29/04/2004

Updates: 01 May[1] , 06 May[2], 20-21 May[3], 05 June[4]

H6 Beam Crate ROD fragment format. Version 1

P.Gorbunov (University of Toronto & ITEP)

1. Intoduction

The proposed beam fragment format is based on specifications [1] and describes the data as it arrives to the ROS from the Beam Crate ROD. Some fragment components, like a sub-fragment directory of the Status block (Section 3), or sub-fragments 0xF1, 0xF2 and 0xFF (Sections 4.7, 4.8) are optional and can be removed from the format in future versions, depending on the outcome of a discussion of the present draft.

Depending on the Beam ROD – ROS link implementation, some formal parts of the fragments (the header and the trailer, the sub-fragment directory) can be added in the ROS itself, to reduce the link traffic.

Basic storage units in the fragment are full 32-bit words of type (unsigned int). However, data parts of most of the sub-fragments (beam detector data, meta data) are internally structured as arrays of 16-bit (unsigned short) words or 8-bit bytes.

The overall Beam ROD fragment structure is as follows:

[Header]
[Status elements]
[Data elements]=[sub-fragment](s) (one or more)
[Trailer]

Details of these entities are given in the following sections.

---

1 Corrections: format version number. Minor changes: run number, Ext Lev1 ID, Stat0.Specific, CRC32, TrigFlag. No s/fragment directory by default. Mistakes corrected in 4.2.

2 Updates: Flag word in the Status block, other minor changes. Appendix A is added.

3 Updates: More details (and modifications) to “Hdr”, “Wtail”, “Beam” and “Mwpc” sub-fragment descriptions (sections 4.1, 4.3, 4.5 and 4.6). A few typos are fixed.

4 Final specification for the beam-related sub-fragments. New sub-fragments: miniRODs and the run header.

2. Header

Word	*Name*	*Value*	*Remarks*
0	Start of header Marker	0xee1234ee	[1], Sect. 5.1
1	Header size	9	[1], Sect. 4
2	Format Version Number	0x02040000	[1], Sect. 5.7; major version 2.4
3	Source ID	0x00007000	[1], 5.2, App. B
4	Run Number	...	presumably, as specified in [1], 5.5, with run types 0x00=physics, 0x01=calibration, 0x02=pedestals
5	Extended Level1 ID	...	sequential L1 trigger in a run[5] (for synchronization with FEB RODs)
6	Bunch Crossing	...	12-bit (optional, can be used for synchronization with the FEB RODs)
7	Level1 Trig Type	...	undefined
8	Detector event type		

=0 special =1 physics
=2 f/e calibration =3 random (pedestal)
=4 BPC calibration

3. Status block

The status block preceeds the data block. Its elements beyond the compulsory first status element include a CRC32, trigger/read-out bits and an optional sub-fragment directory. The CRC32 is used to validate the data received over the ROD-ROS link. The directory plays a rôle of the offset element block which is missing from the ROD format specification. By default `MaxFrag=0` (no directory included).

Word	*Name*	*Value*	*Remarks*
9	Stat0	...	A compulsory status element, [Specific][Generic], see Ref. [1], 5.9

Specific = if non-zero: fatal error (event should be discarded)

Word	*Name*	*Value*	*Remarks*
10	CRC32	....	a 32-bit checksum over the entire ROD fragment with CRC32=0 (a code to compute CRC32 is available)
11	Flag	....	(Byte 0) `Off-spill` non-zero for events taken out of spill (Byte 1) `Trigger bits` TIU inputs at a trigger time (App. A) (Bytes 2-3) `Read-out bits` Read-out pattern, see Appendix A.
12	[MaxFrag][Nfrag]	[0][...]	(Bytes 0-1) `Nsfrag` actual number of sub-fragments (Bytes 2-3) `MaxFrag` the length of the sub-fragment directory

5 This element is set to 0 for events without FEB readout (e.g., for BPC calibration events)

13...13+(MaxFrag-1)	(optional) sub-fragment directory A fixed list of offset elements for MaxFrag possible sub-fragments, formed according to Ref.[1], Sections 3.1.1, 5.1

Word	*ID*	*sub-fragment contents*
13	0x01	beam header
14	0x03	trigger time
15	0x04	warm Tail Catcher
16	0x05	BPC (ITEP beam chambers)
17	0x06	beam counters
18	0x07	MWPCs
19	0xF1	run header meta data
20	0xF2	run trailer meta data
21	0xFF	calibration "stamps"

A number `Nfrag` of sub-fragments is equal to the number of r/o bits set in the `RoBits` field (word 11). If the directory is present, however, it always contains `MaxFrag` offset elements, with missing sub-fragments indicated by zero offsets.

4. Data block

The H6 beam fragment is structured as a sequence of an arbitrary number of sub-fragments. The actual number of sub-fragments in an event can be either retrieved from the Status block (`Nfrag`), or computed by scanning the entire fragment. If a sub-fragment directory is present in the Status block, then any sub-fragment can be directly accessed by using its offset relative to the fragment header.

A sub-fragment consists of:

word 0 = size (in full words)
word 1 = sub-fragment ID
words 2...(size-1) = data words

4.1 0x01 Beam Header sub-fragment

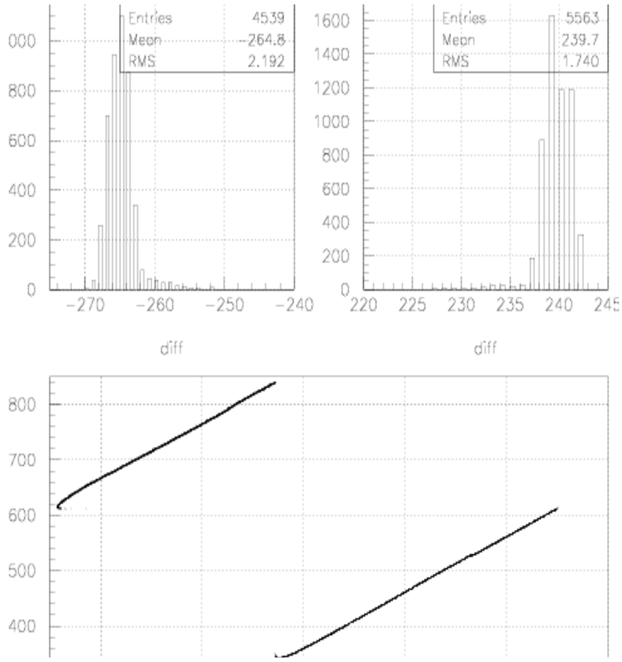
Size = 2+4
Structured as an array of full 32-bit words.

data-word	*Name*	*Meaning*
0	ev_number	sequential event number in a run
1	ev_type	event type, after trigger ambiguity resolution
2	ev_clock	a value from a 10 MHz counter restarted at a Start-of-Spill.
3	ev_trigger	various trigger bits latched by PUs. See [2], Sect 1.2.3.3 for a tentative definition.

4.2 0x03 Trigger Time sub-fragment

Size = 2+ 3
Structured as an array of unsigned 16-bit words.

data-word	*Name*	*Meaning*
0	L1_cl40 ("TTC1")	"40 Mhz" clk from PDG, TDC 2228A[6], N=2.
1	L1_cl40_del ("TTC2")	same, delayed by ~12 ns
2-3	scaler1[2]	time elapsed since the previous particle crossing (in units of ?? ns)
4-5	scaler2[2]	same, downscaled (DSC factor = ??)[7]



The plots illustrate the use of trigger time measurements. Two top histograms show typical distributions of the two possible "values" of (TTC1 – TTC2). The intrinsic resolution is about 100 ps. The difference between the two peaks, corresponding to the TTC clock period, can be used to calibrate the TDC (roughly, 50 ps/ch).

The bottom plot shows the correlation between TTC1 and TTC2. The distortion of a single measurement in cases when the trigger crosses the clock edge is clearly seen.

4.3 0x04 Warm Tail Catcher sub-fragment

Size = 2+ 24
Structured as an array of unsigned 16-bit words.

data-word	*Name*	*Meaning*
0-47	wtc_adc[]	readings from an ADC 2249A[8] ADC, N=9-12. See Appendix B for channel mapping details.

4.4 0x05 Beam Chamber sub-fragment)

Size = 2+ 18
Structured as an array of unsigned 16-bit words.

See a detailed description in Ref. [2], Sect 3.2.

---

[6] 11 bits, useful value range: 1-2047 , 50 ps resolution, zero value when no S1 signal was detected (typical of random triggers).

[7] The scalers were disabled till the run 258, at least.

[8] 10 bits, useful value range: 1-2047, zero value means a h/w problem (no gate).

4.5 0x06 Beam counters

Size = 2+ 9
Structured as an array of unsigned 16-bit words.

data-word	*Name*	*Meaning*
0-4	beam_adc[5]	B, Halo, S3, S2 and S1 ADC 2249A[8], N=7
5-12	muonveto_adc[8]	4 Veto (61-64) and 4 Muon (65-68) counters, ADC 2249A[8], N=8
13-17	beam_tdc[5]	S2, S3, B, Halo, VetoOR, TDC 2228A[6], N=2.

Remarks:
- the definition is still preliminary
- the order of data words is as indicated in the "Meaning" column.

4.6 0x07 Dubna/MPI MWPCs

Size: variable
Structured as an array of unsigned16-bit words.

The last non-zero 16-bit word is the status word formatted as follows:

Bytes 0: *0*
Byte 2: *bit 0 = h/w time-out error (should never happen in our setup)*
bit 1 = h/w overflow error
bit 2 = s/w time-out error (the PCOS controller did not respond in 500 μs)
bit 3 = s/w overflow (too many clusters, data is truncated to 39 clusters)
bit 4 = 1

Thus, under normal circumstances, the status word should be equal to 0x1000.
If necessary, one zero-valued 16-bit word is added after the status word, to have a whole number of full words in the sub-fragment.

The data words have the following format (note that the MWPC data is encoded in terms of r/o *channels*):

Bits 0-11 (12 bits): the channel number at the center of a cluster
Bits 12-15 (4 bits): the cluster width (in wires)

The C-code below illustrates the MWPC data decoding (S.Karev):

```
unsigned short p; // a MWPC data word
int w; // cluster width
int c; // the starting channel of a cluster (0<start<4096)
int chamber; // 0=X2, 1=Y2, 2=X3, 3=Y3, 4=X4, 5=Y4, 6=X5, 7=Y5
int wire; // wire number
p = ...;  // take next word (a cluster)
w = (int) (p>>12); c = (int) ((0xfff & p) - w/2 + !(w&1));
if( c < 768 ) {chamber = c/128; wire=c-chamber*128;}
else {chamber=6+(c-768)/64; wire=c-768-chamber*64;}
```

4.7 0xF1, 0xF2 Run-header and Run-trailer sub-fragments

Size: variable
Structured as a whole number of char[64] arrays ("lines").

These two sub-fragments are highly optional. They represent ASCII dumps of the entire set of run configuration parameters. See Ref.[2], Sections 1.2.1 and 2 for details.

The 2004 run headers have several minor differences with the 2003 format:

- the configuration file names are specified with the keynames in brackets, e.g. `<RunConf>;`
- the `$PathName` record (spcifying the root directory for the configuration files);
- the `BeamParticle` and `BeamSpot` records have text (non-numerical) values. For the beam particle, a obvious notation is used: `e- pi- mu- e+ pi+ mu+ p`. For the beam impact position, the notations proposed by P.Schacht are used: a letter (A,B,C,D,E,F,G) denoting a major beam spot postion and a whole number (0-12) indicating a minor deviation. For X- and Y-scans, the notations Xn and Yn are used, where “n” denotes the scan point number. A blank means “undefined”, or “a shifter was lazy or reluctant to enter it”.

The Appendix D shows an example of a run header recorded in one of the first 2004 physics runs, interpreted by a rhlib package coming together with a demo bytestream file reader (Section ...).

The run header/trailer appear (if at all) in the the very first (run-header sub-fragment) and the very last (run-trailer subfragment) events of a run, either as single sub-fragments in dedicated events, or in combination with regular sets of sub-fragments. The run-trailer can appear only in standalone beam runs.

4.8 0xFF Calibration stamps

Size = 2+ 6
Structured as an array of 8-bit bytes.

This (optional) sub-fragment contains a complete pulser board information (a "stamp") for a given event. The data is a direct copy of the byte string read from the pulser board after it had been prepared to deliver the calibration pulse for the given event. This sub-block appears only in pulser events (type 2).

See Ref. [2], Sect. 3.4 for technical details. Note that the bit-patterns in this sub-fragment correspond to the pulser board channels, not FEB channels. The correspondence between the pulser board and FEB channels is different for different FEBs. The simplest case is that of the Fcal FEBs (all 3 FEBs have the same CalBoard <-> FEB channel mapping: Ref.[3], Table 1).

bytes 0-15 (16 bytes, 128 bits): the bit-pattern of the pulsed channels

bytes 16-19 (4 bytes, 32 bits) : the DAC value (pulse amplitude)
byte 16 = LSByte
...
byte 19 = MSByte
For example: f8 2a 00 00 means DAC=11000

byte 20 (1 byte, 8 bits) : the delay value, in units of ~1 ns (common for all 8 delay channels of a pulser board)
byte 21 (1 byte, 8 bits) : error word (OK=0, if non-zero, the event should be discarded)

bytes 22-23: undefined — can be used for calibration board ID,
e.g. 0xFF01 = Fcal, 0xFF02 = EMEC, 0xFF03 = HEC

4.8 0x02 miniROD data

Size = variable (`nBoards *(3+nSamples*(1+nGains)+2)/2 + 2`)
Structured as an array of 16-bit words.

See Section 3.1 of [2] for a detailed description of the data part. A distinct feature of the this sub-fragment is that each FEB data block starts with a sequence of 16 words containing `0xffff` and ends with a sequence of 16 words containing `0x0000`.[9] The order of FEBs in the sub-fragment corresponds to the `miniROD` record in the run header. By default, it is just 1,2,3,4,5,6,7,8, which means HEC3, HEC4, HEC5, HEC6, EMEC, Fcal1, Fcal1/CTC, Fcal2.

5. Trailer

The trailer conforms to the Ref. [1], Sect 4:

Word	*Meaning*	*Value*
0	Number of status elements	1+3+9(optional) = 4 or 13
1	Number of Data elements	...
2	Status block position	0

6. Disk file formats

The Beam DAQ, when working in a stand-alone mode, writes a stream of ROD fragments on disk, to the directory /raid/data/... Each event in this stream is the ROD fragment, as described in the previous sections, preceeded by a header similar to the ones of sub-fragments:

word 0 = full event size (in full words)
word 1 = the ID (`0xCAFE`)

A standalone format converter program `dress_rod` reads a ROD-stream file and produces a bytestream file readable with ATHENA.

The Appendix F shows an example of the bytestream event header dump corresponding to run 240. This dump is produced by `my_ef_dump` function of the `my_ef` package, a collection of simple tools for working with e-format. The converter `dress_rod`, as well as a demo bytestream file reader `rd_eformat` are made on top of that package. `rd_eformat` reads a bytestream file with the H6 data and unpacks all the subfragments. Currently, MWPCs and miniROD unpacking codes are not yet there but will be added soon.

All source files needed to make the programs mentioned in this Section are available from /afs/cern.ch/user/p/petr/public/Eformat.

[9] In the version 1.5 of Ref. [2] there is a mistake in p. 16 (top two paragraphs). The FEB trailer and header “records” are of 16 short words, not 8.

Appendix A Trigger and read-out bits

Trigger bits in the status word *Flag* correspond to the inputs of the Trigger Input Unit (TIU) . A non-zero bit value means that the corresponding input was fired when the trigger was detected by the TIU.
Normally, only one input is set and the Detector event type (ROD fragment header, Section 2) is directly associated with that input ("trigger bit"), as shown in the table below. However, theoretically, trigger overlays (clashes) can occur. If TIU control software can resolve the trigger ambiguity, the Detector event type is attributed using the internal Beam DAQ look-up tables. In pathological cases, when the ambiguity cannot be resolved, a fatal error flag is written to *Stat0* word and a hardware reset is performed.

Table A1: Trigger bits

Trigger Input/Bit	***Meaning***	***Detector event type***	***Readout***
0	Start of Spill	-	none
1	Physics beam trigger	1 (physics)	full
2	End of Spill	-	none
3	Random (pedestal) trigger	3 (pedestals)	full
4	Calibration	2 (f/e calib)	only FEBs (and, optionally, calibration boards)
5	BPC calibration	4 (BPC calib)	Only BPCs

Read-out bits in the status word *Flag* corresond to the read-out pattern for a given event and define what sub-fragments should be formed within the Beam ROD fragment. The default read-out patterns are defined in Beam DAQ as functions of the run type and event type. They can be changed via a run configuration file.

Table A2: Read-out bits

	Hdr 1	***Time 3***	***Tail 4***	***BPC 5***	***Beam 6***	***Mwpc 7***	***RunH 11***	***RunT 12***	***Calb 15***
physics	x	x	x	x	x	x	x (*)	x (*)	
random	x	x	x	x	x	x	x (*)	x (*)	
F/e calibration	x						x (*)	x (*)	x (**)
BPC calibration	x			x			x (*)	x (*)	

(*) Run Header and Run Trailer sub-fragments can be added, optionally, to the first and to the last events in a run, respectively. The event-carrier can be of any type.

(**) Calibration stamps can be added, optionally to the f/e calibration events.

Appendix B Warm Tail Catcher

Figure B.1 below shows schematically the layout of the Warm Tail Catcher layers, viewed from the Gex side (along the beam direction). The numbers on the PMs correspond to the data words in the Wtail sub-fragment (section 4.3): PM 1 corresponds to wtc_adc[0], ..., PM 48 corresponds to wtc_adc[47].

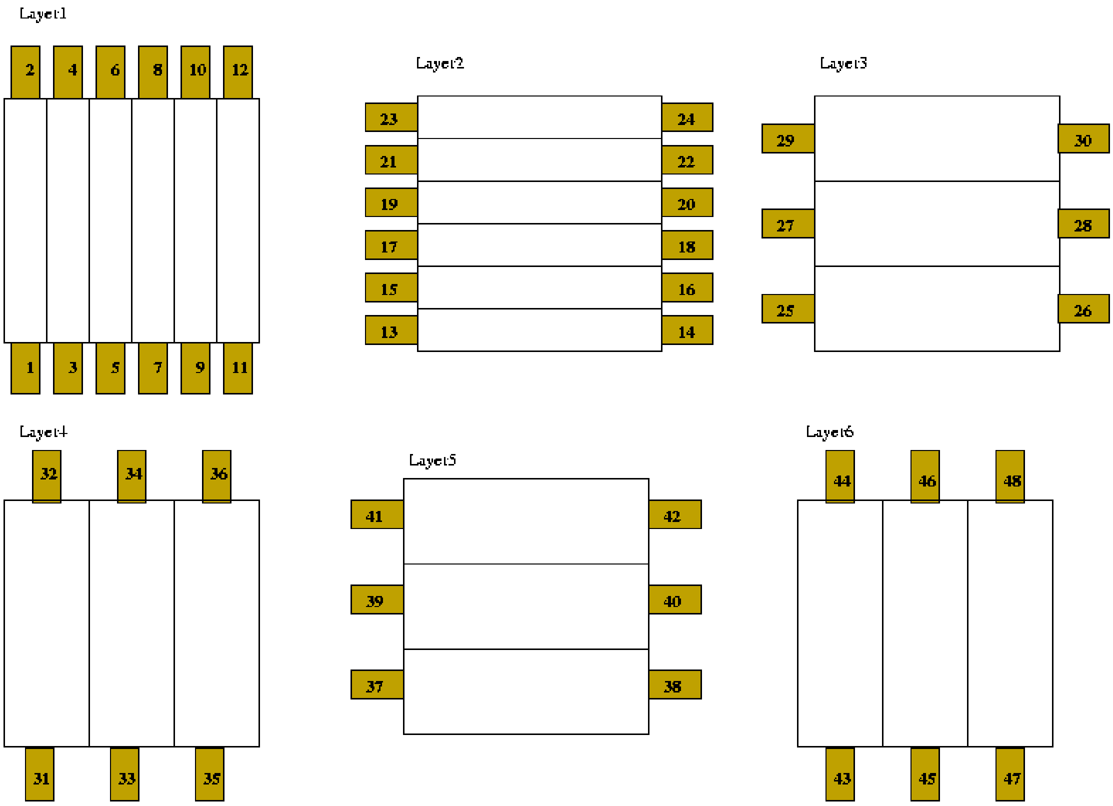


Figure B.1 A schematical drawing of the Warm Tail Catcher based on Leonid's slides of 26/03/2004.

Appendix C Allocation of CAMAC channels (the table by S.Savin).

"N" means a CAMAC slot number.

Latch 4448 (N=13)		**Pattern Unit** (N=14)		**Run Statistics Scaler** (N=20)		**Beam TDC** (N=2)		**Beam ADC** (N=7)		**Beam ADC 2** (N=8	
1	S1	**0**	Early PuP flag	**0**	DAQ event	**0**	TTC1	**0**	B	**0**	V-left
2	S2	**1**	Late PuP flag	**1**	Spill #	**1**	TTC2	**1**	Hol e	**1**	V-up
3	S3	**2**	In-Spill flag	**2**	B2	**2**	S2	**2**	S3	**2**	V-right
4	B2	**3**	Off-Spill flag	**3**	S1	**3**	S3	**3**	S2	**3**	V-down
5	V-left	4	CEDAR 6/8	**4**	S2	**4**	B	**4**	S1	**4**	Mu-2L
6	V-right	5	CEDAR 7/8	**5**	S3	**5**	Hole	**5**		**5**	Mu-2R
7	V-down	6	CEDAR 8/8	**6**	Beam	**6**	Veto	**6**		**6**	Mu-1L
8	V-up			**7**	Hole	**7**	S1?	**7**		**7**	Mu-1R
9	Mu-1L			**8**	Hole & Beam			**8**		**8**	
10	Mu-1R			**9**	Veto & Beam			**9**		**9**	
11	Mu-2L			**10**	Mu1 & Mu2 & Beam			**10**		**10**	
12	Mu-2R			**11**	W1 & W2			**11**		**11**	
13	Hole							**12**		**12**	
14	W1 & W2										
15	B1										
16	Scaler reset type										

Appendix D An example of the run header (interpreted by the rhlib package).

```
$PathName /raid/daq/config/                                   : key=$PathName, 1 values:  0
* ---------   /raid/daq/config/par/runs/run237.par
 RunNumber 237                                                : key=RunNumber, 1 values:  237
 RunType 1                                                    : key=RunType, 1 values:  1
 <RunConf> Physics/narrow.v0                                  : key=<RunConf>, 1 values:  0
* ---------   Physics/narrow.v0
 <ConfFeb> par/fe/default_fe.par                              : key=<ConfFeb>, 1 values:  0
* ---------   par/fe/default_fe.par
 FebSamples      16                                           : key=FebSamples, 1 values:  16
 FebGains        2                                            : key=FebGains, 1 values:  2
 FebAddr         0x2d 0x3a 0x13 0x33 0x26 0x22 0x28 0x21      : key=FebAddr, 8 values:
                                                             45 58 19 51 38 34 40 33
 miniROD         1    2    3    4    5    6    7    8         : key=miniROD, 8 values:
                                                             1 2 3 4 5 6 7 8
 FebTimeout      1000                                         : key=FebTimeout, 1 values:  1000
 FebDacOffset    0xc00                                        : key=FebDacOffset, 1 values:  3072
 FebAutoGainThr  1107 1985                                    : key=FebAutoGainThr, 2 values:
                                                             1107 1985
 FebReadDelay    0x16                                         : key=FebReadDelay, 1 values:  22
 FebFirstSample  0                                            : key=FebFirstSample, 1 values:  0
 TtcCalDly    24                                              : key=TtcCalDly, 1 values:  24
 TtcPdgDly   160                                              : key=TtcPdgDly, 1 values:  160
 TtcFanDly    8*0                                             : key=TtcFanDly, 8 values:
                                                             0 0 0 0 0 0 0 0
 <ConfCam> par/cam/test_cam.par                               : key=<ConfCam>, 1 values:  0
* ---------   par/cam/test_cam.par
 CamBorer  1                                                  : key=CamBorer, 1 values:  1
 Cam2228A  2 3 4 5                                            : key=Cam2228A, 4 values:
                                                             2 3 4 5
 Cam2249A  7:12 21                                            : key=Cam2249A, 7 values:
                                                             7 8 9 10 11 12 21
 CamSc2551 20 19                                              : key=CamSc2551, 2 values:  20 19
 CamOR2088 22                                                 : key=CamOR2088, 1 values:  22
 CamPattB  14                                                 : key=CamPattB, 1 values:  14
 Cam4448   13                                                 : key=Cam4448, 1 values:  13
 CamRTC    6                                                  : key=CamRTC, 1 values:  6
 CamPCOS   16                                                 : key=CamPCOS, 1 values:  16
 CamEvClock 60000 60001                                       : key=CamEvClock, 2 values:  60000
60001
 CamPattern 130200 140200                                     : key=CamPattern, 2 values:  130200
140200
 CamBpc_1 210000 210001 30000:30003                           : key=CamBpc_1, 6 values:
                                                             210000 210001 30000 30001 30002 30003
 CamBpc_2 210002 210003 30004:30007                           : key=CamBpc_2, 6 values:
                                                             210002 210003 30004 30005 30006 30007
 CamBpc_3 210004 210005 40000:40003                           : key=CamBpc_3, 6 values:
                                                             210004 210005 40000 40001 40002 40003
 CamBpc_4 210006 210007 40004:40007                           : key=CamBpc_4, 6 values:
                                                             210006 210007 40004 40005 40006 40007
 CamBpc_5 210008 210009 50000:50003                           : key=CamBpc_5, 6 values:
                                                             210008 210009 50000 50001 50002 50003
 CamBpc_6 210010 210011 50004:50007                           : key=CamBpc_6, 6 values:
                                                             210010 210011 50004 50005 50006 50007
 CamTime  20000:20001 -190010 -190211                         : key=CamTime, 4 values:
                                                             20000 20001 -190010 -190211
 CamTail 90000:90011 100000:100011 110000:110011 120000:120011  : key=CamTail, 48 values:
        90000 90001 90002 90003 90004 90005 90006 90007 90008 90009 90010 90011
        100000 100001 100002 100003 100004 100005 100006 100007 100008 100009 100010 100011
        110000 110001 110002 110003 110004 110005 110006 110007 110008 110009 110010 110011
        120000 120001 120002 120003 120004 120005 120006 120007 120008 120009 120010 120011
 CamBeam  70000:70004 80000:80007 20002:20006                 : key=CamBeam, 18 values:
        70000 70001 70002 70003 70004 80000 80001 80002 80003 80004 80005 80006
        80007 20002 20003 20004 20005 20006
 CamOutReg 221700                                             : key=CamOutReg, 1 values:  221700
 CamScaler -200000 -190000 -190001 -190002 -190003 -190211    : key=CamScaler, 6 values:
                                              -200000 -190000 -190001 -190002 -190003 -190211
 ReadOutMask   Runh Hdr Bpc Time Tail Beam Mwpc Fcal          : key=ReadOutMask, 8 values:
                                                             0 0 0 0 0 0 0 0
 TrigNarrow beam                                              : key=TrigNarrow, 1 values:  0
 RunDebug 1   0    4    0    0    0   0    0    5*0    0    0: key=RunDebug, 15 values:
                                                             1 0 4 0 0 0 0 0 0 0 0 0 0 0 0
 Bpc     1 2 3 4 5 6                                          : key=Bpc, 6 values:
                                                             1 2 3 4 5 6
 DataStore 1 /raid/data                                       : key=DataStore, 2 values:  1 0
 BeamMomentum 120 GeV/c                                       : key=BeamMomentum, 2 values:  120 0
 BeamParticle e+                                              : key=BeamParticle, 1 values:  0
 BeamSpot X                                                   : key=BeamSpot, 1 values:  0
 MaxBursts 1                                                  : key=MaxBursts, 1 values:  1
RunDate 20040605                                              : key=RunDate, 1 values:  20040605
RunTime 031358                                                : key=RunTime, 1 values:  ??
```

Appendix F An example of the bytestream event header (interpreted by the my_ef package).

```
  - field -      Full Ev         Sub Det         ROS             ROB
SoHdrMarker     0xaa1234aa      0xbb1234bb      0xcc1234cc      0xdd1234dd
TotFragSize     0x      3d9     0x      3c4     0x      3b8     0x      3ab
Header Size     0x       15     0x        c     0x        d     0x        f
FormatVers #    0x 2040000      0x 2040000      0x 2040000      0x 2040000
Source ID       0x    a0000     0x    a7000     0x    27000     0x    17000
Run Number      0x       f0     0x       f0     0x       f0     0x       f0
NumberStatEl    0x        1     0x        1     0x        1     0x        1
Status 0        0x        0     0x        0     0x        0     0x        0
No. OffsetEl    0x        1     0x        1     0x        1     0x        1
Offset 0        0x70000015      0x        c     0x        d     0x        f
No. FragSpec    0x        a     0x        1     0x        2     0x        4
- fspec --
  DateTime      0x12345678
  GlobEvID      0x        0
  ExtL1 ID      0x        0                     0x        0     0x        0
  L1TrType      0x        0     0x        0                     0x        0
  L2TrInfo      0x        0
  EvFilnfo       0 0 0 0 0
  BunchXng                                      0x        0     0x        0
  DetEvTyp                                                      0x        4

---------------
ROD 0: N stat el=4, N data el=908 Status block pos=0
---------------
 SoH marker     0xee1234ee
 Hdr Size       0x        9
 Form. Vers.    0x 2040000
 Source ID      0x     7000
 Run Number     0x       f0
 Ext L1 ID      0x        0
 Bunch Xng      0x        0
 L1 Tr Type     0x        0
 Det Ev Type    0x        4
-----
Stat. elem 0    0x        0
Stat. elem 1    0xfacedeca
Stat. elem 2    0x 8222002
Stat. elem 3    0x        3
-----
Data  elem 0    0x        6
Data  elem 1    0x        1
Data  elem 2    0x        1
....
```

References

[1] C.Bee et al., The raw event format in the ATLAS Trigger & DAQ,
ATL-DAQ-98-129 (EDMS: ATL-E-ES-0019), version 2.4, 2004-02-23

[2] P.Gorbunov, FCal Test Beam DAQ: description of raw data format,
http://cern.ch/atlas-fcaltb/Memos/DAQ/DataFormat.general.pdf,
version 1.5, 9 January 2004

[3] P.Gorbunov, Calibration run types in FCal beam tests,
http://cern.ch/atlas-fcaltb/Memos/DAQ/Calibration%20run%20types.pdf
Draft 2.0 22-Nov-2003