FastTracKer Second Stage Board Spybuffer Parsing

Samuel, WONG Hiu Wing
Department of Physics, the Chinese University of Hong Kong
CERN Summer Student Program
Supervised by Prof. Stefania Xella and Dr. Alessandra Camplani

***Abstract*— The implementation of a monitoring tool for the FastTraKer (FTK) system is described in this report. Particular focus is given to one of the FTK system boards, the Second Stage Board (SSB). Circular buffer memories, called spybuffers, are used for monitoring purposes when the FTK data are dumped. A script to parse and evaluate the quality of the data at the output of the SSB board has been implemented and tested for this project.**

I. What is FTK?

The ATLAS experiment [1], at the Large Hadron Collider (LHC) [2], is preparing to exploit the new Physics opportunities that will be offered by the increase of luminosity, foreseen for the next years [3].

The new running conditions pose a challenge on the trigger system. For this reason, the Fast TracKer (FTK) system has been developed. FTK is a hardware processor built to reconstruct tracks with transverse momentum above 1 GeV at a rate of up to 100 kHz and to provide them to the High Level Trigger (HLT). To achieve this goal, the system uses a parallel architecture with algorithms designed to exploit the computing power of custom Associative Memory chips and modern Field Programmable Gate Arrays (FPGAs) [4].

The FTK makes use of the 12-layers of hit information from the Inner Detector to reconstruct the tracks. The system as a whole has 6 different types of board, the Input clustering Mezzanine (IM) and Data Formatter (DF) which are both responsible for taking care of the input signals from the Inner Detector.

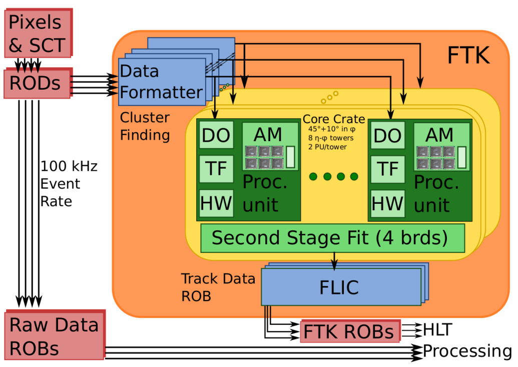


Fig. 1. Data flow in FTK [5]

The Auxiliary Card (AUX) and Associative Memory Board (AMB) are responsible for first stage track fitting using 8 layers of hits.

The Second Stage Board (SSB) uses the tracks from AUX and 4 additional hits from DF to extrapolate and refit the track using 12-layer hits.

Finally, the FTK Level-2 Interface Crate (FLIC), receives the tracks from SSB, further processes and sends them to the HLT in ATLAS. The data flow in FTK is shown in Fig.1.

The SSB makes use of spybuffer memories. A spybuffer is a circular buffer that holds event data that can be read out for monitoring. This project mainly focus on the parsing the spybuffer dump of the SSB output. [1]

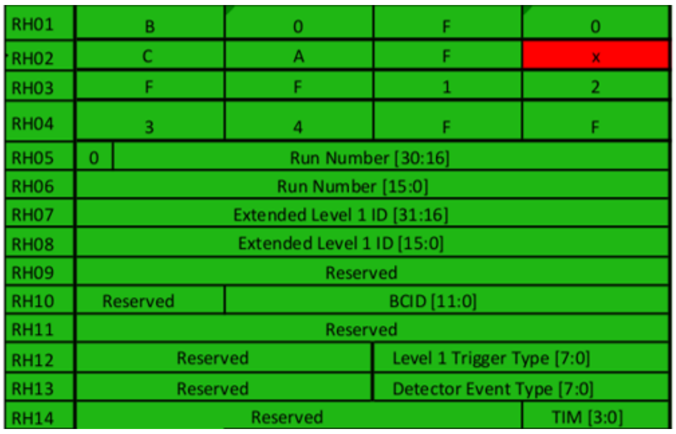
II. SSB spybuffer Format

The SSB spybuffer dumps data in words of 16 bits each [6]. The packet comes in three parts: header, track information and trailer.

[1]The original python script can be found here.

A. *Header format*

The header contains 14 words. The first four words are fixed, in order to identify the beginning of an event. The header also contains other information such as Level 1 Identifier (L1ID) and the run number. The header format is shown in Fig. 2.

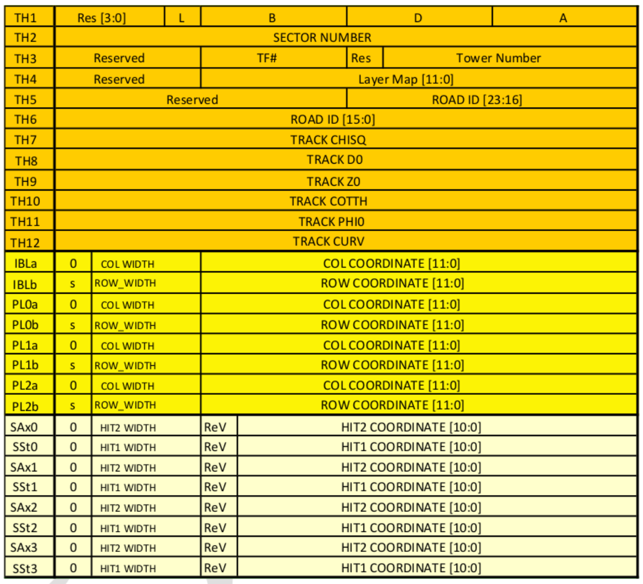


RH01	B	0	F	0
RH02	C	A	F	x
RH03	F	F	1	2
RH04	3	4	F	F
RH05	0	Run Number [30:16]		
RH06	Run Number [15:0]			
RH07	Extended Level 1 ID [31:16]			
RH08	Extended Level 1 ID [15:0]			
RH09	Reserved			
RH10	Reserved	BCID [11:0]		
RH11	Reserved			
RH12	Reserved		Level 1 Trigger Type [7:0]	
RH13	Reserved		Detector Event Type [7:0]	
RH14	Reserved			TIM [3:0]

Fig. 2. Header Format [7]

B. *Track Frame Format*

The track frame consists of 28 words, as shown in Fig. 3. It contains information about the sector/tower number, which correspond to the physical location of the detector, the helix parameters of the track and the hit coordinates.

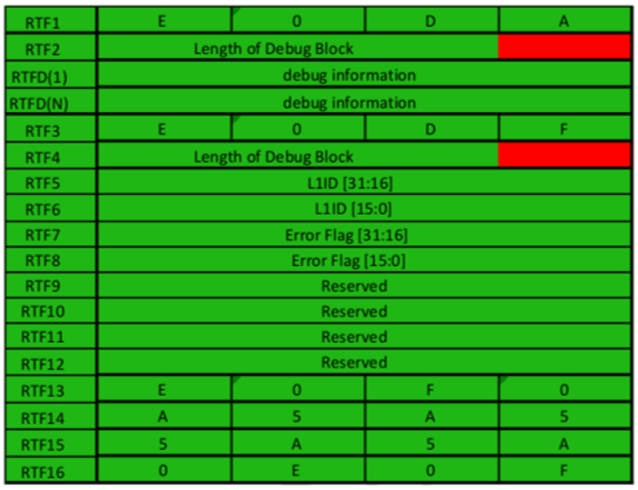


TH1	Res [3:0]	L	B	D	A
TH2	SECTOR NUMBER				
TH3	Reserved		TF#	Res	Tower Number
TH4	Reserved		Layer Map [11:0]		
TH5	Reserved			ROAD ID [23:16]	
TH6	ROAD ID [15:0]				
TH7	TRACK CHISQ				
TH8	TRACK D0				
TH9	TRACK Z0				
TH10	TRACK COTTH				
TH11	TRACK PHI0				
TH12	TRACK CURV				
IBLa	0	COL WIDTH	COL COORDINATE [11:0]		
IBLb	s	ROW_WIDTH	ROW COORDINATE [11:0]		
PL0a	0	COL WIDTH	COL COORDINATE [11:0]		
PL0b	s	ROW_WIDTH	ROW COORDINATE [11:0]		
PL1a	0	COL WIDTH	COL COORDINATE [11:0]		
PL1b	s	ROW_WIDTH	ROW COORDINATE [11:0]		
PL2a	0	COL WIDTH	COL COORDINATE [11:0]		
PL2b	s	ROW_WIDTH	ROW COORDINATE [11:0]		
SAx0	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]	
SSt0	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]	
SAx1	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]	
SSt1	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]	
SAx2	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]	
SSt2	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]	
SAx3	0	HIT2 WIDTH	ReV	HIT2 COORDINATE [10:0]	
SSt3	0	HIT1 WIDTH	ReV	HIT1 COORDINATE [10:0]	

Fig. 3. Track Frame Format [7]

C. *Trailer Format*

The trailer contains 16 words, as shown in Fig. 4. Specific words are use to identify the end of data file and the packet itself. The trailer stores the error flag, debug information and L1ID.



RTF1	E	0	D	A
RTF2	Length of Debug Block			🟥
RTFD(1)	debug information			
RTFD(N)	debug information			
RTF3	E	0	D	F
RTF4	Length of Debug Block			🟥
RTF5	L1ID [31:16]			
RTF6	L1ID [15:0]			
RTF7	Error Flag [31:16]			
RTF8	Error Flag [15:0]			
RTF9	Reserved			
RTF10	Reserved			
RTF11	Reserved			
RTF12	Reserved			
RTF13	E	0	F	0
RTF14	A	5	A	5
RTF15	5	A	5	A
RTF16	0	E	0	F

Fig. 4. Trailer Format [7]

III. Implementation in the FTK_Solo tool

The SSB spybuffer parser has been implemented and tested in the FTK_Solo tool, a monitoring script used by all the FTK boards, dumping status registers and spybuffers and saving the information as text files.

Whenever the FTK_Solo tool is called, the SSB spybuffer parser automatically checks the dumps produced for the SSB, creating the two output log files described in the next section.

IV. SSB Spybuffer Parser

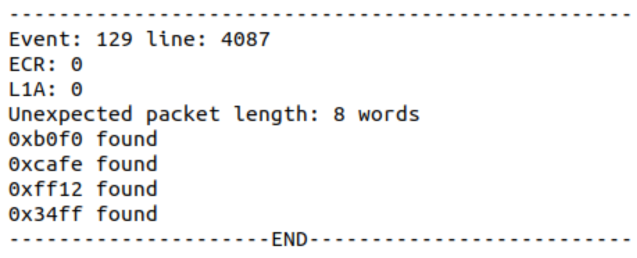
The SSB spybuffer parser has two functions.

First is to identify corrupted packets, locate all the errors and store the main information and the error message in a text file called ssb_spybuffer_parse_format.log.

Second is to convert the track helix parameters from floating point hexadecimal representation into fixed point decimal representation. The helix parameters are then saved in a file called ssb_spybuffer_parse_helix.log.

A. *Output file: ssb_spybuffer_parse_format.log*

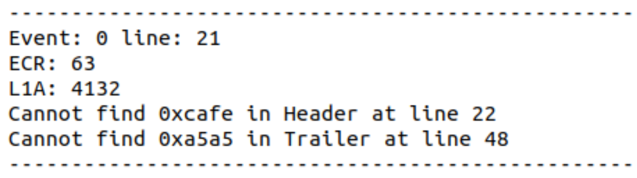
1) Packet integrity: As a first step, the parser checks the length of the packet, since header, track and trailer blocks have fixed length. The packet is expected to have $30 + 28 \times n$ words (where n is the number of tracks). If the packet length is wrong, the parser checks the existence of all key words. An example is shown in Fig. 5.



```
------------------------------------------------
Event: 129 line: 4087
ECR: 0
L1A: 0
Unexpected packet length: 8 words
0xb0f0 found
0xcafe found
0xff12 found
0x34ff found
--------------------END-------------------------
```

Fig. 5. Error message: unexpected packet length

If the packet has a proper length, the parser checks the key words and their relative position. In the header, for example, 0xcafe should be right after 0xb0f0. If the parser is not able to find the expected word in specific location, the line number of the error is recorded. An example is shown in Fig. 6.



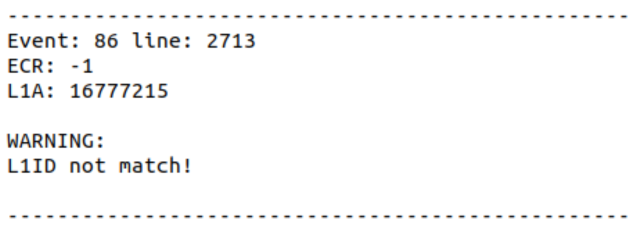
```
------------------------------------------------
Event: 0 line: 21
ECR: 63
L1A: 4132
Cannot find 0xcafe in Header at line 22
Cannot find 0xa5a5 in Trailer at line 48
------------------------------------------------
```

Fig. 6. Error message: cannot find fixed word

2) Level 1 Identifier: L1ID is a 32 bit information that consists of two parts. The first part (from bit 31 to 24) is the Event Counter Reset (ECR) counter. The second part (from bit 23 to 0) is the Level-1 Accept (L1A) counter.

L1As represent the events accepted by the ATLAS first level trigger. When a L1A is received, the L1A counter is increased by 1. If an ECR is received, the ECR counter is increased by 1, and L1A counter is reset to -1. When the next L1A is received, the L1A is set to 0.

In a good packet, the L1ID in both header and trailer must match. The parser also checks if L1IDs match or not. Otherwise, an error message is printed into the file, as shown in Fig. 7.



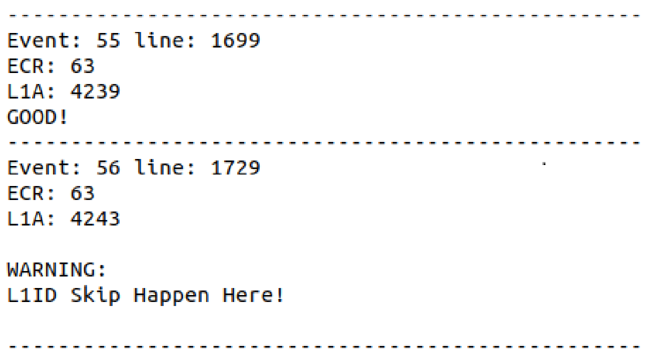
```
------------------------------------------------
Event: 86 line: 2713
ECR: -1
L1A: 16777215

WARNING:
L1ID not match!

------------------------------------------------
```

Fig. 7. Error message: L1ID unmatched

Additionally, L1IDs have to be consecutive. From event to event, either L1A counter is increased by 1 or ECR counter is increased by 1 and L1A counter is reset. If the packet skipped some L1As without increment of ECR or skipped ECRs directly, an error message is left. An example is shown in Fig 8.



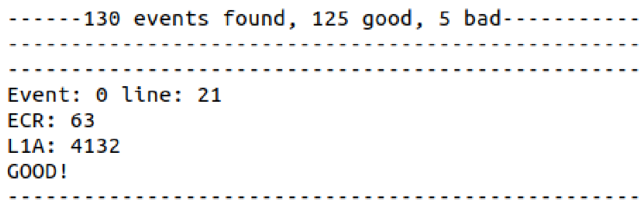
```
------------------------------------------------
Event: 55 line: 1699
ECR: 63
L1A: 4239
GOOD!
------------------------------------------------
Event: 56 line: 1729
ECR: 63
L1A: 4243

WARNING:
L1ID Skip Happen Here!

------------------------------------------------
```

Fig. 8. Error message: L1ID skip

3) Number of corrupted packets: The number of the corrupted packets is printed at the beginning of ssb_spybuffer_parse_format.log, as shown in Fig. 9.

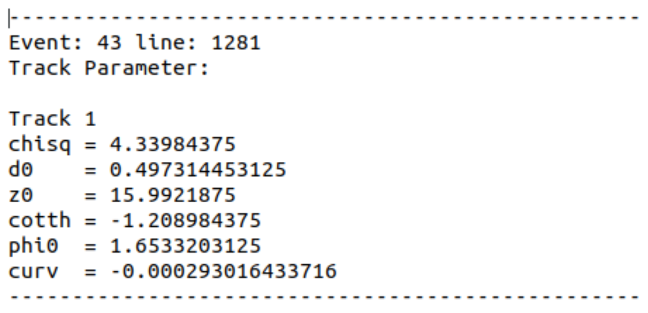


```
------130 events found, 125 good, 5 bad----------
-------------------------------------------------
-------------------------------------------------
Event: 0 line: 21
ECR: 63
L1A: 4132
GOOD!
-------------------------------------------------
```

Fig. 9. Global message: number of good and bad packets

B. *Output file: ssb_spybuffer_parse_helix.log*

The helix parameters are stored in half-precision floating point format, in the spybuffer dump file. The floating point value in hexadecimal representation is converted into fixed point decimal representation, and saved in ssb_spybuffer_parse_helix.log. An example of helix parameters in a track is shown in Fig. 10.

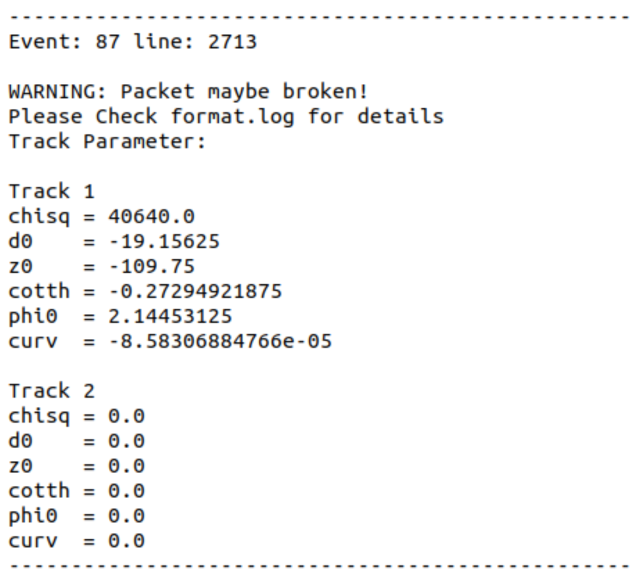


```
|-------------------------------------------------
Event: 43 line: 1281
Track Parameter:

Track 1
chisq = 4.33984375
d0    = 0.497314453125
z0    = 15.9921875
cotth = -1.208984375
phi0  = 1.6533203125
curv  = -0.000293016433716
-------------------------------------------------
```

Fig. 10. Helix Parameters

If the event is broken, for example L1ID skips happened, but the packet still contain helix parameters, the parser leaves a warning message, as shown in Fig. 11.



```
-------------------------------------------------
Event: 87 line: 2713

WARNING: Packet maybe broken!
Please Check format.log for details
Track Parameter:

Track 1
chisq = 40640.0
d0    = -19.15625
z0    = -109.75
cotth = -0.27294921875
phi0  = 2.14453125
curv  = -8.58306884766e-05

Track 2
chisq = 0.0
d0    = 0.0
z0    = 0.0
cotth = 0.0
phi0  = 0.0
curv  = 0.0
-------------------------------------------------
```

Fig. 11. Error message: broken packet with helix parameters

V. Histogram of the Helix Parameters

Histograms of the helix parameters can be seen in Figs.12 and 13, from a spybuffer taken while running test data in lab4. These histogram are generated using pyroot. [2] The range of the helix parameters are shown in Table 1.

TABLE I

Expected Range of Helix Parameter [6]

Helix Parameter	Range
d_0	(-32mm, 32mm)
z_0	(-320mm, 320mm)
cot θ	(-6.55, 6.55)
ϕ_0	(0, 2π)
Curvature	(-0.65m^{-1}, 0.65m^{-1})

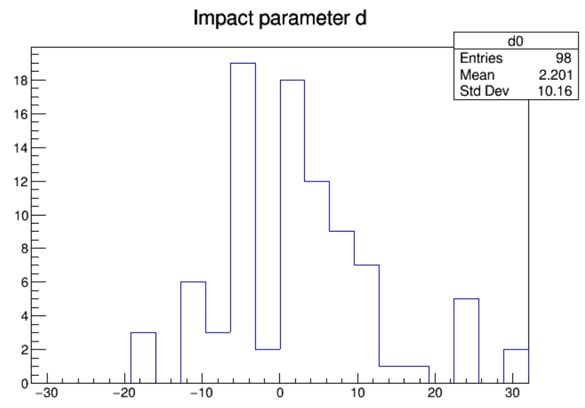


Fig. 12. Impact parameter d_0

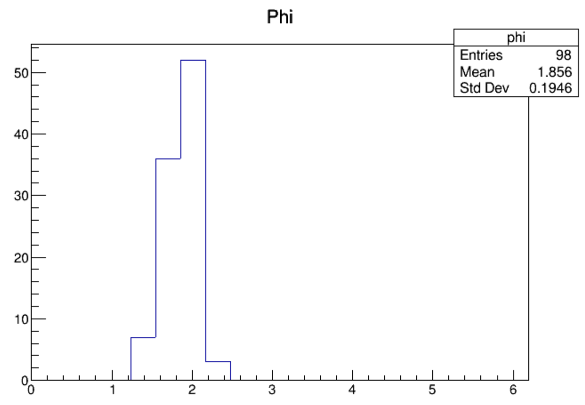


Fig. 13. Azimuthal angle ϕ_0

[2]The pyroot script can be found here

VI. Conclusion

The SSB spybuffer parser takes the spybuffer dump as input, checks the packet integrity, L1ID skips, and extract helix parameters from the SSB output packets. This can greatly reduce the effort of identifying broken packets, especially with L1ID skips. Additionally, the track helix parameters are converted into fixed point decimal format and dumped in a file. This simplifies and makes possible further analysis.

VII. Acknowledgement

I would like to express my great thanks to Stefania Xella and Alessandra Camplani, who supervised and guided me through out this project. I would also like to thank Jonathan Long for providing the framework of this python scripts and helping me during the online software testing.

Finally, I would like to thank the Physics Department at Chinese University of Hong Kong and CERN Summer Student Program Committee for the financial support and organizing this summer program.

References

[1] The ATLAS Collaboration; G. Aad, E. Abat, J. Abdallah, A. A. Abdelalim, A. Abdesselam, O. Abdinov, B. A. Abi, M. Abolins, H. Abramowicz *The ATLAS Experiment at the CERN Large Hadron Collider*. Journal of Instrumentation, Volume 3, August 2008

[2] The LHC Study Group; Pettersson, Thomas Sven (ed.); Lefvre, P (ed.) *Large Hadron Collider : conceptual design*.

[3] The ATLAS Collaboration ; CERN. Geneva. The LHC experiments Committee ; LHCC *Letter of Intent for the Phase-I Upgrade of the ATLAS Experiment*

[4] The ATLAS collaboration; lizawa, Tomoya (Waseda U.) *The ATLAS Fast Tracker system*. Topical Workshop on Electronics for Particle Physics, Santa Cruz, Ca, United States of America, September 2017, pp.139

[5] The LHC experiments Committee (LHCC), The ATLAS Collaboration; Shochet, M; Tompkins, L; Cavaliere, V; Giannetti, P; Annovi, A; Volpi, G *Fast TracKer (FTK) Technical Design Report*. June, 2013

[6] The ATLAS collaboration; Argonne National Laboratory; Jinlong Zhang *ATLAS FTK: The Output Data Format* April, 2016

[7] The ATLAS collaboration; Argonne National Laboratory; John Anderson, Robert Blair, Gary Drake, Andrew Kreps, Jimmy Proudfoot, Jinlong Zhang *Specification of the FTK / L2 Interface Crate (FLIC) for the ATLAS Fast-Tracker*. April, 2016