ATLAS

EDMS Number:

EDMS Id:

ATLAS Level-1 Calorimeter Trigger

FEX Hub Firmware

Working document

Document Version: Draft 0.0

Document Date: 06 April 2015

Prepared by: D. Edmunds, Y. Ermoline, W. Fisher, P. Laurens,
Michigan State University, East Lansing, MI, USA

Document Change Record

Version	Issue	Date	Comment
0	0	06 April 2015	Initial document layout

TABLE OF CONTENTS

1. INTRODUCTION

1.1. CONVENTIONS

The following conventions are used in this document:

- Hub – or “FEX Hub” is FEX system ATCA switch (hub) module.
- eFEX – electron Feature EXtractor.
- jFEX – jet Feature EXtractor.
- gFEX – global Feature EXtractor.
- ROD – or “Hub-ROD” is Readout Driver (ROD) mezzanine on the FEX Hub.

1.2. RELATED PROJECTS

[1] FEX System Switch Module (FEX Hub) Prototype (v0.3), 21 September 2014, http://www.pa.msu.edu/hep/atlas/l1calo/hub/specification/1_preliminary_design_review/Hub_Spec_v0_3.pdf

[2] Electromagnetic Feature Extractor (eFEX) Prototype (v0.2), 6 February 2014, https://twiki.cern.ch/twiki/pub/Atlas/LevelOneCaloUpgradeModules/eFEX_spec_v0.2.pdf

[3] Jet Feature Extractor (jFEX) Prototype (v0.2), 14 July 2014, http://www.staff.uni-mainz.de/rave/jFEX_PDR/jFEX_spec_v0.2.pdf

[4] Global Feature Extractor (gFEX) Prototype (v0.3), 16 October 2014, https://edms.cern.ch/file/1425502/1/gFEX.pdf

[5] Hub-based ReadOut Driver (L1Calo ROD) Prototype (v0.9.5), 1 July 2014, https://edms.cern.ch/file/1404559/2/Hub-ROD_spec_v0_9_5.docx
http://www.pa.msu.edu/hep/atlas/l1calo/hub/reference/ROD/

[6] The Gigabit Link Interface Board (GLIB) ecosystem, TOPICAL WORKSHOP ON ELECTRONICS FOR PARTICLE PHYSICS 2012, 17–21 SEPTEMBER 2012, OXFORD, U.K.

[7] FELIX: Interfacing the GBT to general purpose networks, https://twiki.cern.ch/twiki/bin/view/Atlas/GBT2LAN

[8] GBT: Giga Bit Transceiver, https://espace.cern.ch/GBT-Project/default.aspx

[9] IPbus: a flexible Ethernet-based control system for xTCA hardware, TOPICAL WORKSHOP ON ELECTRONICS FOR PARTICLE PHYSICS 2014, 22–26 SEPTEMBER 2014, AIX EN PROVENCE, FRANCE

[10] Development of an ATCA IPMI controller mezzanine board to be used in the ATCA developments for the ATLAS Liquid Argon upgrade, http://cds.cern.ch/record/1395495/files/ATL-LARG-PROC-2011-008.pdf

Hub : ATCA Hub Module page at MSU: http://www.pa.msu.edu/hep/atlas/l1calo/hub/

Hub Module FPGA Firmware Topics by Dan:
http://www.pa.msu.edu/hep/atlas/l1calo/hub/hardware/details/hub_fpga_firmware_topics.txt

1.3. FEX HUB OVERVIEW

The FEX Hub [1] is the FEX ATCA shelf switch module. Its primary function is to support FEX readout system, provide switching functionality for module control and DCS IPbus networks and to distribute timing and control signals to the FEX modules [2] [3] [4] .

Figure 1 shows a sketch of the Hub modules within the FEX ATCA shelves.

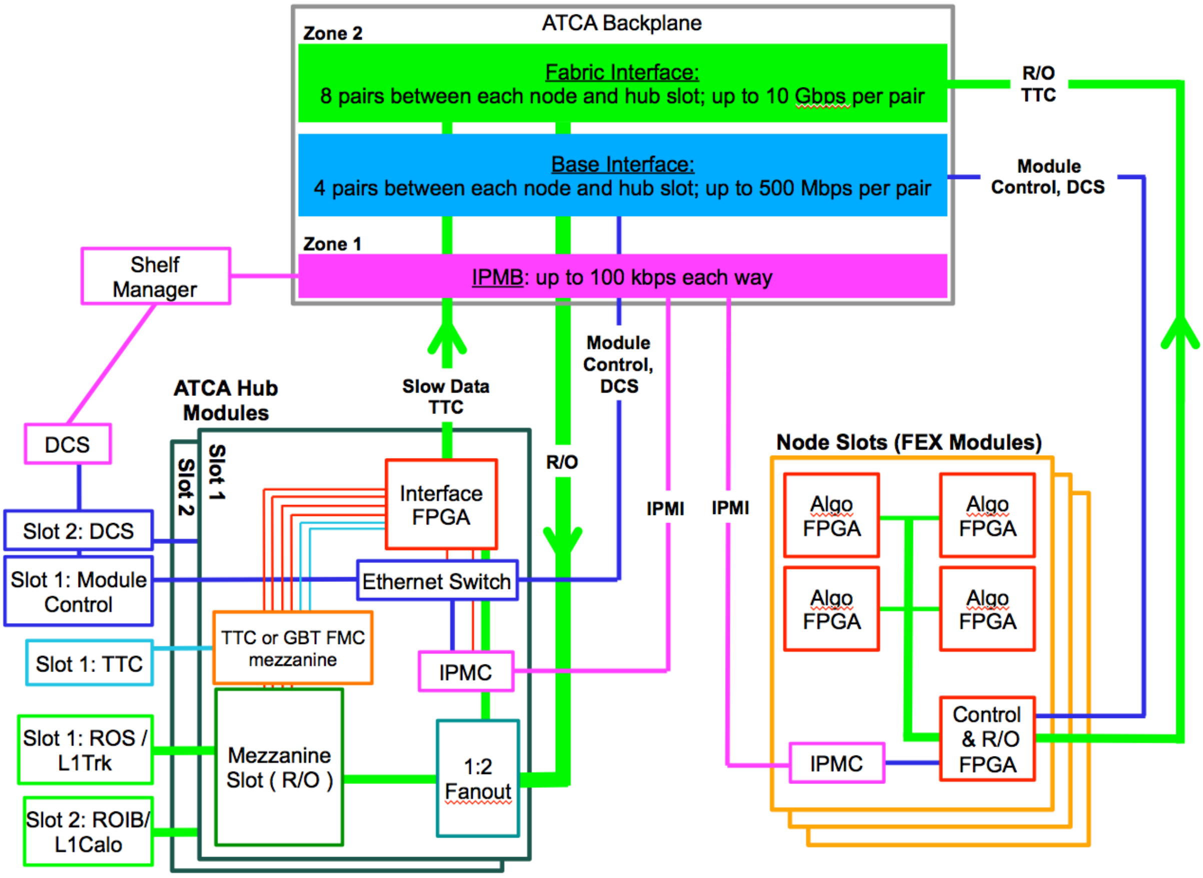


Figure 1: Illustration of the functions of FEX Hub modules within the FEX readout system.

There are to be two Hub modules per shelf. Both Hub modules will receive high-speed FEX data over the ATCA Fabric Interface, which will be fanned out to a ROD mezzanine on the Hub and to the Hub's own FPGA. This high-speed data path will include two data channels from the other Hub module. The Hub module in logical slot 1 will provide switching capability for a network that routes module control signals on the base interface, while the Hub in logical slot 2 will provide switching for a network that routes DCS information. The Hub module in slot 1 will further host a TTC or GBT mezzanine card, whose signals will be decoded and fanned out to the FEX modules and also the Hub in slot 2. The fanned-out TTC control data stream will be interleaved with ROD-to-FEX communications including, for example, back-pressure signals.

The Hub module has connections to the other slots in the ATCA shelf over three distinct electrical interfaces, as illustrated in Figure 1. ATCA backplane Zone-2 consists of the Fabric Interface and the Base Interface. The Fabric Interface provides 8 differential pairs (channels) from each node slot to each Hub slot (8 to Hub-1 and 8 to Hub-2). There are a total of 8 Fabric Interface channels between Hub-1 and Hub-2. The Fabric Interface pairs have a nominal bandwidth specification of 10 Gbps / channel. The Base Interface provides 4 differential pairs between each node slot and each Hub slot. There are a total of 4 Base Interface channels between Hub-1 and Hub-2. The Base Interface lines have a nominal bandwidth specification of 500 Mbps / channel, suitable for Gbps Ethernet protocol.

Finally, ATCA backplane Zone-1 provides each node and Hub slot with a connection to the Intelligent Platform Management Bus (IPMB) with a total bandwidth of 100 kbps.

The L1Calo FEX Hub system will consist of eight Hub modules. There will be two eFEX shelves (each of 12 eFEX modules), one jFEX shelf (holding 7(still under discussion) jFEX modules) and one gFEX shelf (with 1 gFEX module).

Figure 2 shows a possible Hub module PCB layout.

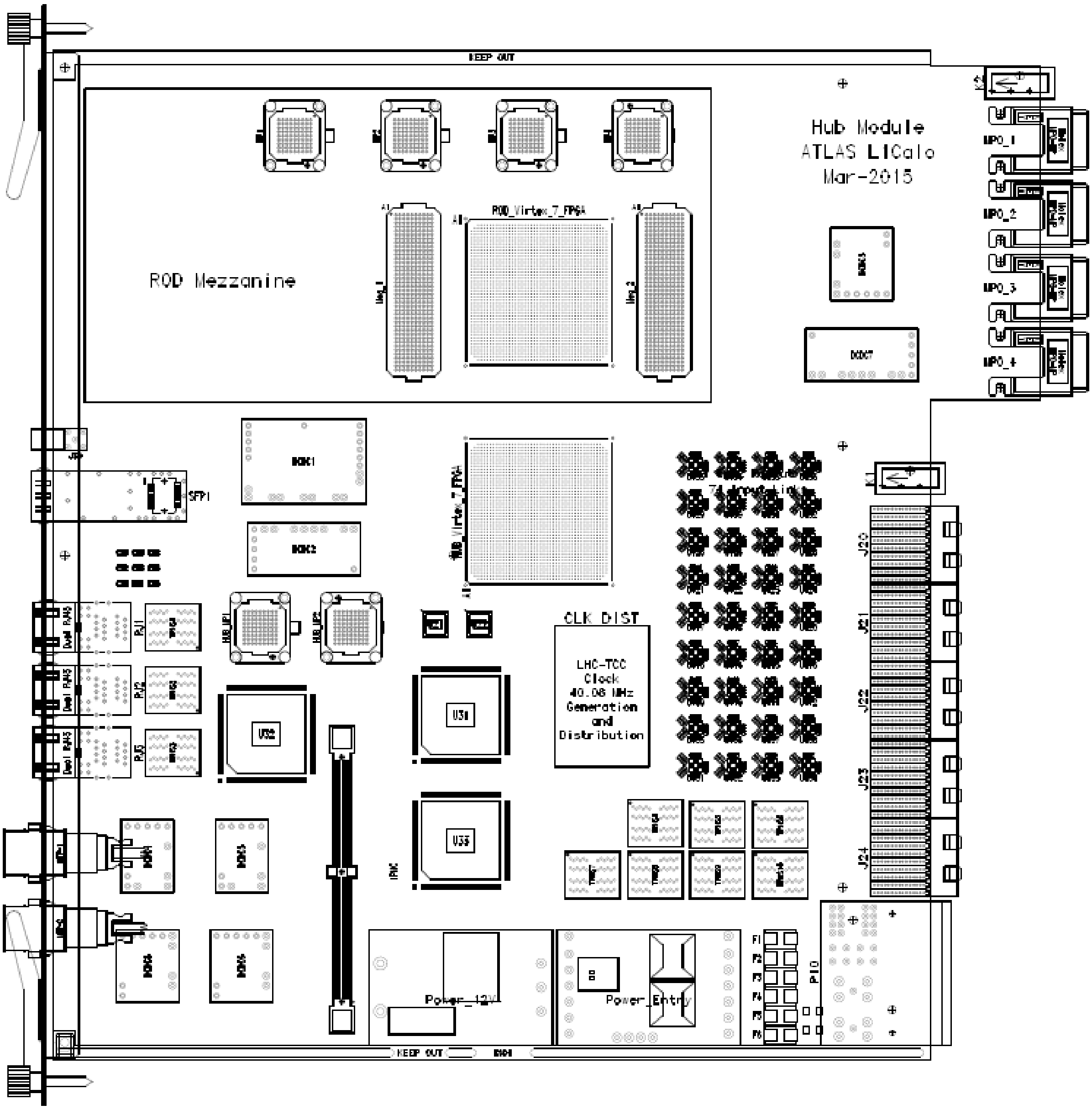


Figure 2: Illustration of the Hub module PCB layout as of 25 Mar 2015.

The main Hub FPGA will be a large Xilinx Virtex-7 device, such as an XC7VX550T-1FFG1927. This offers large logic resources and Block RAM, and adequate fast Multi Gigabit transceivers. In fact it is the number of receivers that is critical: input data from the FEXs and the second Hub module requires 74 inputs. A few more inputs are needed for Ethernet and TTC signals. The XC7VX550T is the smallest device with sufficient transceivers (80 GTH's). The XC7VX690T is pin compatible, and offers a modest increase in Logic and Block RAM.

Following chapters discuss individual interfaces to the Hub FPGA.

2. INTERFACE TO FEX/ROD DATA

2.1. FEX/ROD DATA DISTRIBUTION

The Hub receives over the ATCA Fabric Interface 6 serial streams of Readout Data from each FEX Module – 72 maximum in total for the eFEX shelves with 12 eFEX modules. Each Hub also receives over the Fabric Interface 2 serial streams of Readout Data from the other Hub in the crate. These 74 high speed serial streams are fanned out on the Hub. One copy of each stream is sent to the ROD and one copy is sent to the Hub's own Virtex-7 FPGA.

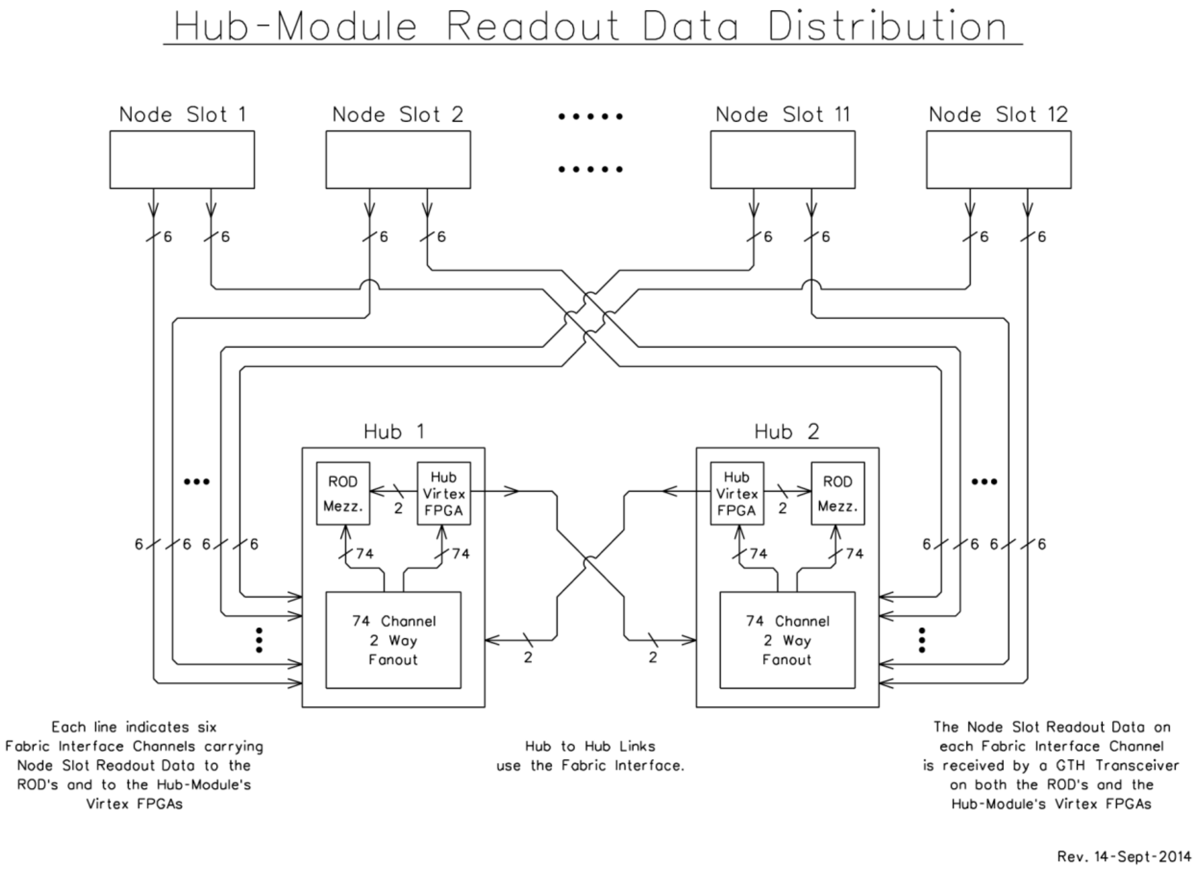

Hub-Module Readout Data Distribution
Node Slot 1
Node Slot 2
•••••
Node Slot 11
Node Slot 12
6
Hub 1
Hub 2
ROD Mezz.
Hub Virtex FPGA
2
74
74 Channel 2 Way Fanout
•••
Each line indicates six Fabric Interface Channels carrying Node Slot Readout Data to the ROD's and to the Hub-Module's Virtex FPGAs
Hub to Hub Links use the Fabric Interface.
The Node Slot Readout Data on each Fabric Interface Channel is received by a GTH Transceiver on both the ROD's and the Hub-Module's Virtex FPGAs
Rev. 14-Sept-2014

Figure 3: Illustration of FEX-Hub distribution of high-speed data signals.

The Hub FPGA also sends 2 serial streams with its own Readout Data to its own ROD and to the other Hub FPGA. The data rate per readout stream will be 10 Gbps or less.

- 72 GHT receivers from the FEX modules + 2 GHT receivers from the other Hub FPGA.
- 2 GHT transmitters to Hub’s own ROD + 2 GHT transmitters to the other Hub.

The fan-out of the readout data in the Hub is implemented with On-Semi 2-way fan-out chips NB7VQ14M - http://www.onsemi.com/pub_link/Collateral/NB7VQ14M-D.PDF

2.1.1. eFEX Data format ([2] chapter 4.2, 5.1 and 5.3)

On receipt of an L1A signal, the eFEX provides RoI data and DAQ data to the Hub (to the ROD mezzanine on the Hub and to the Hub’s FPGA). Collectively, these data are referred as readout data. For each L1A, data from a programmable time frame of up to three bunch crossings can be read out.

The eFEX outputs a single stream of readout data, which contains the super-set of the RoI and DAQ data. For each event that is accepted by the Level-1 trigger, the eFEX can send three types of data to the readout path:

- Final Trigger Object words (TOBs) - copies of those transmitted to L1Topo, in normal running mode these are the only data read out.
- Expanded TOBs (XTOBs) - words that contain more information about trigger candidates than can be transmitted on the real-time data path. The number of XTOBs may be larger than the number of TOBs, XTOBs are not normally read out (this functionality can be enabled via the slow control interface).
- Input Data - all data received from the calorimeters after serial-to-parallel conversion and after the CRC word has been checked. There are a number of programmable parameters, set via slow control, that determine which Input Data are read out.

On receipt of an L1A, the eFEX transmits to the ROD a packet of data, format shown in Figure 4:

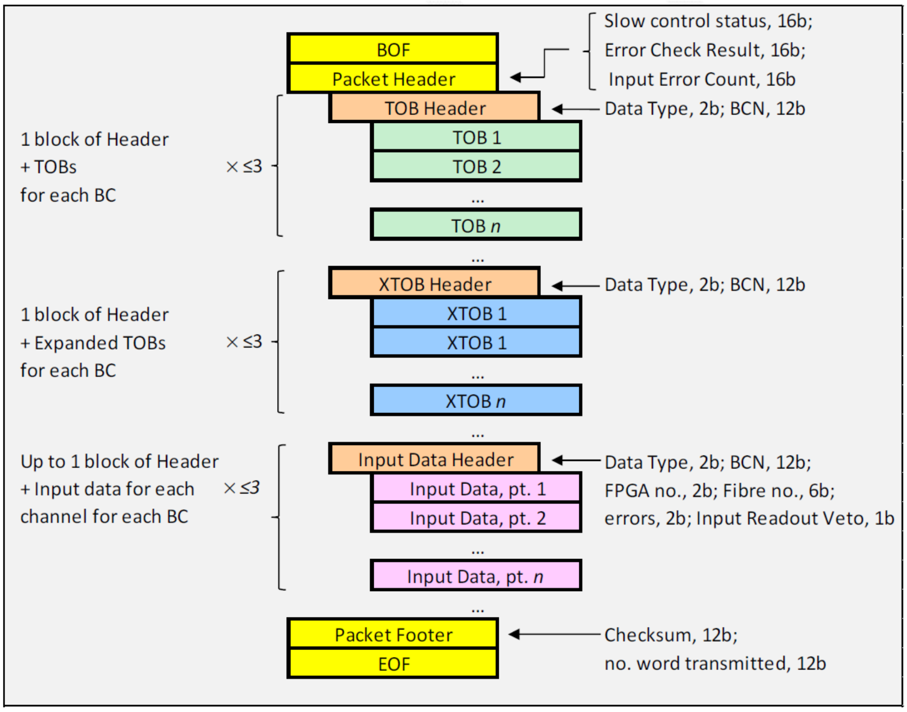
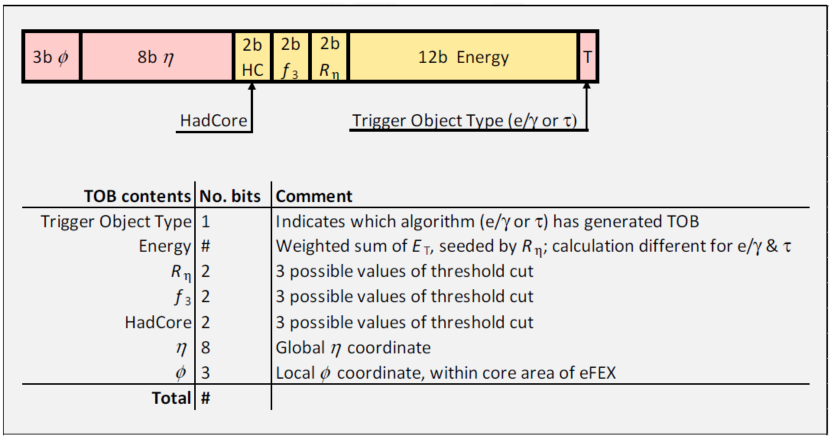


Figure 4: A provisional format for a readout data packet.

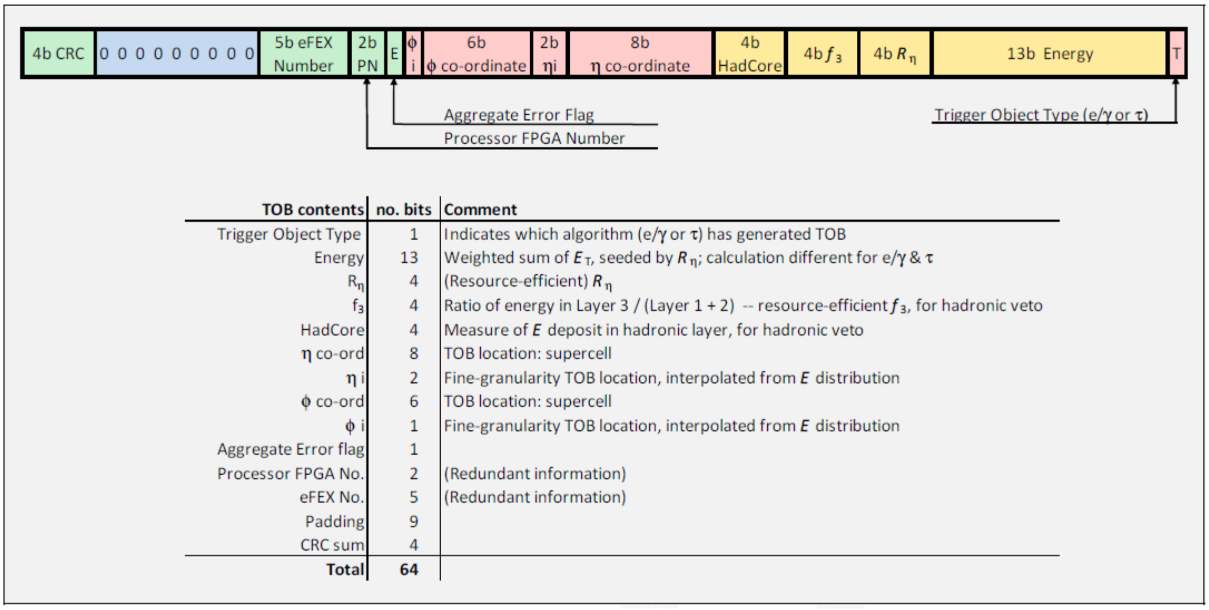
Figure 5 shows a draft format of the TOB. It is 30 bits wide.



TOB contents	No. bits	Comment
Trigger Object Type	1	Indicates which algorithm (e/γ or τ) has generated TOB
Energy	#	Weighted sum of E_T, seeded by R_η; calculation different for e/γ & τ
R_η	2	3 possible values of threshold cut
f_3	2	3 possible values of threshold cut
HadCore	2	3 possible values of threshold cut
η	8	Global η coordinate
ϕ	3	Local ϕ coordinate, within core area of eFEX
Total	#	

Figure 5: Draft TOB Format.

Figure 6 shows a draft format of the XTOB. It is 64 bits wide.



TOB contents	no. bits	Comment
Trigger Object Type	1	Indicates which algorithm (e/γ or τ) has generated TOB
Energy	13	Weighted sum of E_T, seeded by R_η; calculation different for e/γ & τ
R_η	4	(Resource-efficient) R_η
f_3	4	Ratio of energy in Layer 3 / (Layer 1 + 2) -- resource-efficient f_3, for hadronic veto
HadCore	4	Measure of E deposit in hadronic layer, for hadronic veto
η co-ord	8	TOB location: supercell
η i	2	Fine-granularity TOB location, interpolated from E distribution
ϕ co-ord	6	TOB location: supercell
ϕ i	1	Fine-granularity TOB location, interpolated from E distribution
Aggregate Error flag	1	
Processor FPGA No.	2	(Redundant information)
eFEX No.	5	(Redundant information)
Padding	9	
CRC sum	4	
Total	**64**	

Figure 6: Draft XTOB Format.

Input Data - the eFEX modules receive data from the electromagnetic and hadronic calorimeters on optical fibres. For the baseline line rate of 6.4 Gb/s, the data are encoded as specified below:

Electromagnetic Calorimeter Input Data Format for a 6.4 Gb/s Link:

- 10-bit data are provided for each of the 10 super-cells in a tower of 0.1 × 0.1 (h × f).
- The data from neighbouring trigger towers of 0.1 × 0.1 (h × f) are BC-multiplexed to enable them to be transmitted on a single fibre.
- A 10-bit cyclic redundancy check is used to monitor transmission errors.
- 8b/10b encoding is used to maintain the DC balance of the link and ensure there are sufficient transitions in the data to allow the clock recovery.
- Word-alignment markers (8b/10b control words) are inserted periodically, as substitutes for zero data.

Using 8b/10b encoding, the available payload of a 6.4 Gb/s link is 128 bits per bunch crossing (BC). The above scheme uses 121 bits (data from 10 supercells, plus 10 BCMUX flags, plus a 10-bit CRC). The remaining 7 bits/BC are spare. The order of the data in the payload is not yet defined.

Hadronic Calorimeter Input Data Format for a 6.4 Gb/s Link:

- For each of eight trigger towers of 0.1 × 0.1 (h × f), 10 bits of data are provided, summed in depth over the trigger towers.
- A 10-bit cyclic redundancy check is used to monitor transmission errors.
- 8b/10b encoding is used to maintain the DC balance of the link and ensure there are sufficient transitions in the data to allow the clock recovery.
- Word-alignment markers (8b/10b control words) are inserted periodically, as substitutes for zero data.

The above scheme uses 80 bits/BC, leaving 48 bits/BC spare (per link). The order of the data in the payload is not yet defined.

2.1.2. jFEX Data format

2.1.3. gFEX Data format

2.1.4. Data format to/from other Hub

2.1.5. Data format to ROD

2.2. FEX/ROD DATA PROCESSING

What ROD is doing?

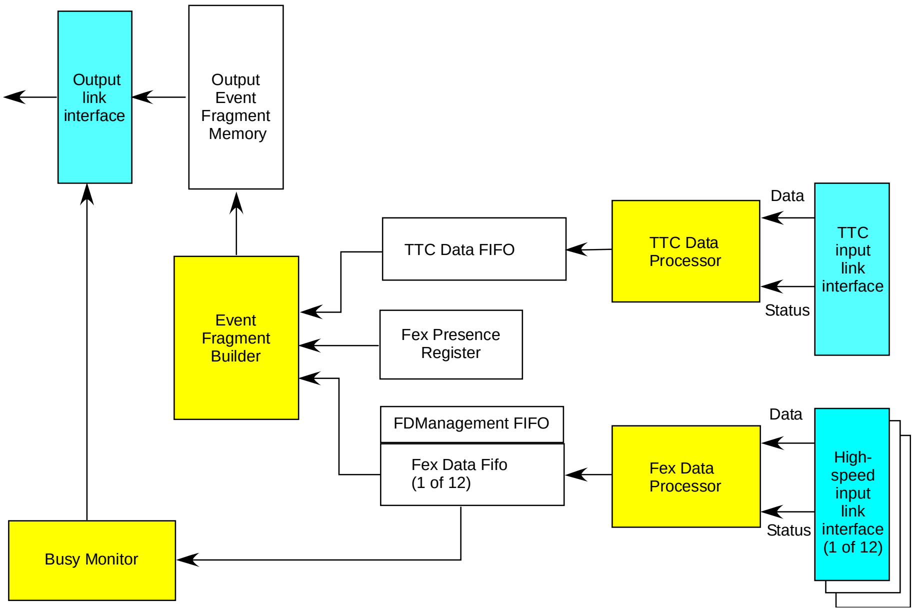
Look at ROD 3.2.1(Functional Requirements), 4 (Algorithms and Resources), 8 (Programming model)

From [3] 4.1. Event Data Processing

Figure below shows an overall summary of the main event data processing algorithms.

Input data from the FEX modules is received via High-Speed Link interfaces, which monitor the backplane link status, and perform clock recovery and 8b/10b decoding. The outputs from the interfaces are the FEX data and the status of each link (up or down).

The FEX Data Processor performs CRC checking, separates the event and bunch numbers from other FEX data, and performs any other required FEX data reformatting. On completion, the (variable length) output FEX data are inserted into the FEX Data FIFO. The bunch number, event number, and FEX Data length are inserted into the FDManagement FIFO.



Timing information (TTC input) is also received via a High-Speed Link interface, which monitors the link status and performs high-quality clock and data recovery. This block produces TTC data for each L1A, and link status (Link up or down).

The TTC Data Processor performs any required processing of TTC data, and stores the result as a fixed-length TTC Data block in the TTC Data FIFO for each L1A.

The Busy Monitor logic compares the occupied depth of all FIFOs to individual limit registers (one for each type of FIFO), and maintains an internal BUSY signal which is active when one or more of the thresholds is exceeded (i.e. when one or more of the memories is approaching full). When BUSY changes state, a BUSY_Activate or BUSY_Deactivate signal is sent to the Output Link interface. This will be sent on to the CTP, causing triggers to be suspended until the FIFO levels fall again.

The most complex logic is in the Event Fragment Builder. This monitors the TTC and FEX data management FIFOs, the FEX presence map (loaded by software), the FEX input link status, and the output fragment memory status. When there is space available in the output event fragment memory and TTC data waiting in the TTC FIFO, a timer is started. When every FEX present has either link down or data waiting, or when the timer expires, the following Event Building steps are performed and the resulting data stored in the Output Event Fragment memory:

- Construct Felix Header (from computer-loadable registers) and copy to output;
- Construct ATLAS Standard Event Fragment header (from TTC data and computer-loadable registers) and copy to output. This action removes the TTC data from the TTC FIFO.
- For each FEX in sequences from 1 to 12:
    - Ignore a disabled FEX
    - If Link Down is asserted, set a bit in the fragment status word
    - If data is absent due to a timeout, set a bit in the fragment status word
    - If the FEX bunch number and event number do not match the header bunch number and event number, set a mismatch bit in the fragment status word, and copy the event and bunch numbers to output;
    - Copy the FEX data to output (removing the data from the FIFO)
- Construct the ATLAS Standard event fragment trailer and copy to output;
- Construct the Felix trailer and copy to output.

It should be noted that the detailed event processing logic will be more complex than described here. Each FEX may provide data on up to six backplane links, potentially requiring separate handling, and there may be more than one category of output data processed through separate event building logic. The overall scheme will however be as described.

In Phase-II, RoI (TOB) output has to be provided to L1Track and L1Calo via a low-latency, possibly synchronous route. This will use a streamlined form of the above logic, omitting the ATLAS standard event headers.

From [3] 4.1.1. Event monitoring

In addition to the CRC checking of event data, the ROD must maintain statistics of link failures so that low-rate errors can be detected. Logic will be in place to monitor the status of all links by counting errors (e.g. 8b/10b code violations) on a bunch-by-bunch basis. This information will be available in computer-readable registers.

From [1] 4.8 Future Use Cases

The FEX-Hub module is intended to be used in the L1Calo and L0Calo trigger systems through Run 4. As such, future use cases in which the Hub may need to augment the capacity of the FEX-Hub-ROD readout path have been identified. This extra functionality is being implemented on the FEX-Hub so long as it does not complicate the core Hub functions and design. These extra Hub functions are as follows:

- The Hub main FPGA receives a fanned-out copy of all high-speed FEX data being sent to the ROD mezzanine card, allowing at a minimum the monitoring of FEX data. This feature can also support Hub commissioning and diagnostics, as it further provides a Fabric Interface connection to the other Hub module.
- The Hub main FPGA provides additional MGT links to the ROD mezzanine, which will be instrumented on the ROD if sufficient input MGT links are available. Similarly, MGT links from the ROD to the Hub main FPGA are defined on the HUB-ROD interface.
- External data output paths from the Hub main FPGA are provided electrically via Ethernet and optically via one Minipod transmitter. The Minipod socket and routings are implemented by default, but the Minipod transmitter is only installed if required.

Together, this Hub functionality can provide supplemental trigger processing if required. However, all of this functionality could instead be ignored or disabled with no negative impact on the Hub core functions.

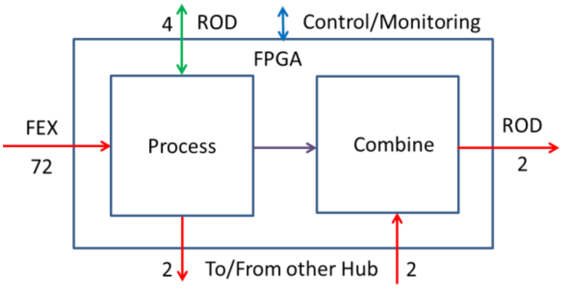


Figure 7: FEX/ROD data processing.

2.2.1. FEX/ROD Data interface

INPUTS

GTH receivers:

- 72 receivers - 6 Links Receive Readout Data from each of 12 FEX
  - Line rate: 6.412592 Gb/s Ref. clock: 320.6296 MHz
- 2 receivers - 2 Links Receive Readout Data from the Other Hub
  - Line rate: x.x Gb/s Ref. clock:

INPUT/OUTPUT

General IO:

- 4 signals - Spare HP I/O signals to/from ROD - 4 LVDS pairs

OUTPUTS

GTH transmitters:

- 2 transmitters - Send Readout Data to the ROD on This Hub
  - Line rate: x.x Gb/s Ref. clock:
- 2 transmitters - Send Readout Data to the ROD on Other Hub
  - Line rate: x.x Gb/s Ref. clock:

2.2.2. FEX Data processing

At a minimum the monitoring of FEX Data – what to do with this monitoring data? Send over IPbus?

2.2.3. Data generation to other Hub

Generates this Hub Data (from FEX Data?) and sends to other Hub

2.2.4. Data generation to ROD

Receives the other Hub Data

Combines FEX Data with the other Hub Data

Sends it to this Hub ROD

2.2.5. ROD Geographic Address

8-bit System Geographic Address (GA) coming to the ROD from the Hub FPGA. The Hub FPGA determines this System Geographic Address by combining:

- 8-bit J10 Geographic Address pins from ATCA backplane Zone 1,
- Shelf Address (Shelf Number) retrieved from the Shelf Manager by the IPMC (see 5.2.1).

The Hub needs to pass its unique location in the overall L1Calo system to the ROD over 8 lines (not to be confused with the 8 backplane Zone 1 Geographic Address pins). Is there now a defined way that the Slot Number plus Crate Number should be encoded in 8 bits?

INPUTS

General IO:

- 8-bit J10 Geographic Address pins from ATCA backplane Zone 1

OUTPUTS

General IO:

- 8-bit System Geographic Address to this Hub's ROD

2.2.6. ROD control/monitoring FPGA interface

INPUTS

General IO:

- 1 signal - Receive the "ROD Present" signal from the ROD
- 1 signal - Receive the "I Have Powered Up" Power Control signal from the ROD
- 8 signals - Listen to ROD's front panel signals as a Life Boat (?)

INPUT/OUTPUT

General IO:

- 2 signals - Spare Power Control signals to/from the ROD

3. GBT/TTC INTERFACE

3.1. GBT/TTC DATA DISTRIBUTION

FELIX/GBT link for TTC

Since PDR, now plan to not include the TTC-FMC mezzanine card on Hub PCB in order to free up precious floor space for other constraints (important as ROD form factor evolves). The plan is to receive the Optical Timing signal (TTC data over GBT) from FELIX with an SFP/SFP+ optical module and send this signal into a GTH Transceiver input on the Hub's Virtex 7 FPGA. Inside the FPGA the "Clock" and the "TTC Information" content will be extracted from the incoming Optical signal.

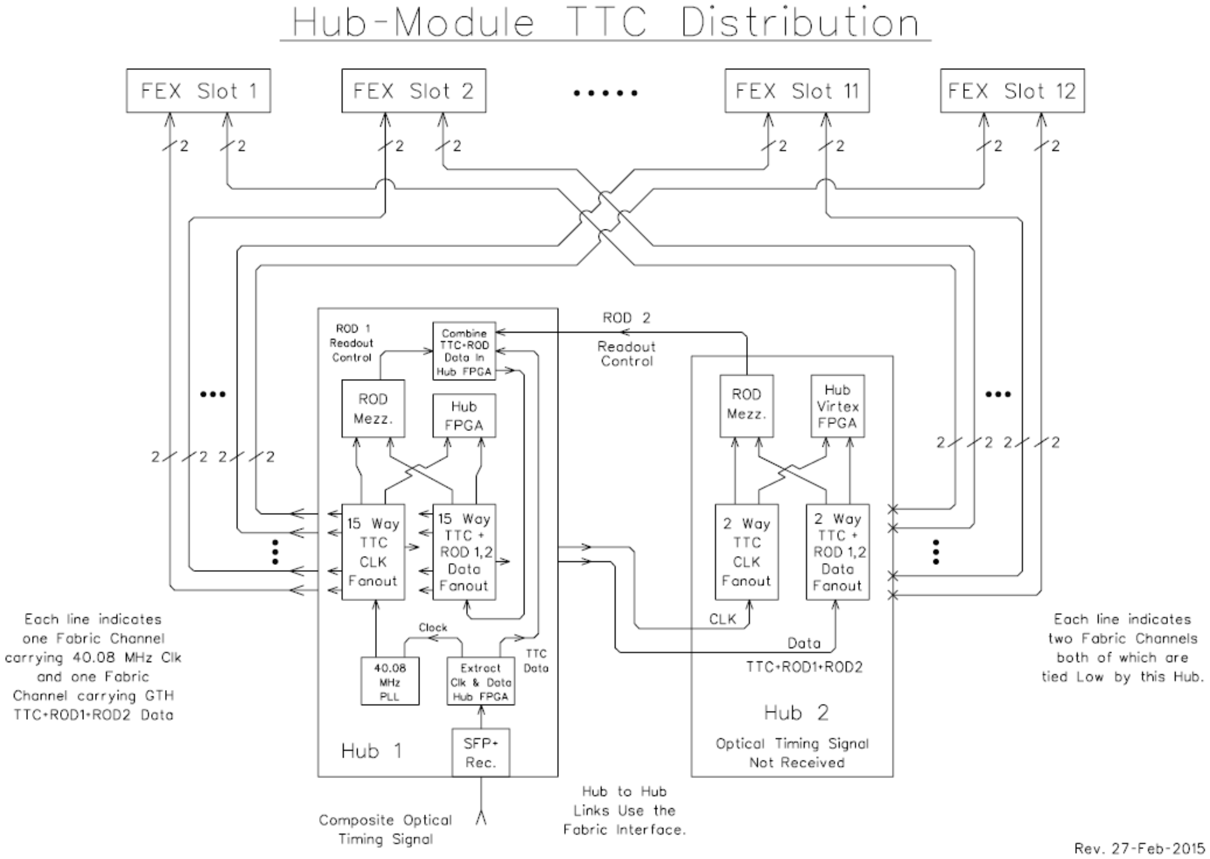

Hub-Module TTC Distribution
FEX Slot 1
FEX Slot 2
•••••
FEX Slot 11
FEX Slot 12
2
ROD 1 Readout Control
Combine TTC+ROD Data In Hub FPGA
ROD 2 Readout Control
ROD Mezz.
Hub FPGA
ROD Mezz.
Hub Virtex FPGA
15 Way TTC CLK Fanout
15 Way TTC + ROD 1,2 Data Fanout
2 Way TTC CLK Fanout
2 Way TTC + ROD 1,2 Data Fanout
Clock
40.08 MHz PLL
Extract Clk & Data Hub FPGA
TTC Data
SFP+ Rec.
Hub 1
CLK
Data
TTC+ROD1+ROD2
Hub 2
Optical Timing Signal Not Received
Each line indicates one Fabric Channel carrying 40.08 MHz Clk and one Fabric Channel carrying GTH TTC+ROD1+ROD2 Data
Each line indicates two Fabric Channels both of which are tied Low by this Hub.
Composite Optical Timing Signal
Hub to Hub Links Use the Fabric Interface.
Rev. 27-Feb-2015

Figure 8: Illustration of FEX-Hub distribution of TTC clock and control data stream signals.

The Clock will be immediately sent out of the FPGA. Outside of the FPGA the recovered 40.08 MHz Clock will be cleaned up by an external 40.08 MHz PLL. This PLL is also guaranteed to be running within 50 ppm of the LHC frequency even when there is no incoming Optical Timing signal. The output of this PLL will be fanned out as needed, e.g. to a Global Clock input on the Hub's FPGA, to the ROD mounted on the Hub, and over Fabric Interface lines to the FEX cards and to the Other Hub.

The "TTC Information" content that is recovered from the incoming Optical Timing signal will be combined with the “readout-control data” (former "back data") from the ROD on This Hub and with the “readout control data” from the ROD on the Other Hub. This combined stream “TTC Information” from the Optical Timing signal + ROD1 “readout-control data” + ROD2 “readout-control data” will come out of the Hub's FPGA on a GTH Transceiver output. On the Hub this GTH signal will be fanned out and sent to the ROD on This Hub and sent over the Fabric Interface to the FEX cards and to the Other Hub.

The Hub Virtex FPGA must have firmware to recover the Clock and the Information from the incoming Optical Timing signal and it must have firmware to make the Combined Information signal that is distributed to the other cards in the Shelf.

Therefore, the Hub FPGA receives Timing signal from FELIX (via SFP) and "readout-control data" coming from both the ROD on Hub-1 and the ROD on Hub-2 (and the copy of its own “combined data stream” output?). It sends the “combined data stream”…?:

- 1 GHT receiver from FELIX (TTC data over GBT),
- 2 GHT receivers for the readout-control data from this Hub’s ROD and other Hub’s ROD,
- 1 recovered 40.08 MHz TTC Clock output (diff),
- 1 GHT transmitter to the Hub fan-out.

From the 1st external PLL a "clean 40.08 MHz clock" is distributed to everyone who needs a stable LHC locked clock and it is used as a reference for a 2nd higher frequency external PLL. This 2nd external PLL would be something like 8x i.e. 320.64 MHz. The output of this 2nd external PLL is sent into the Hub's FPGA as the GTH Reference. The "clean 40.08 MHz clock" from the 1st PLL is guaranteed to always be running even when the Optical Timing signal is not there it will be within 50 ppm of the LHC frequency. The 2nd external PLL is always locked to the output of the 40.08 MHz external PLL so even without an Optical Timing signal everything runs normally (i.e. the 40.08 and the 320.64 remain locked to each other even without the Optical Timing signal).

The frequency of the 2nd external PLL must be optimally selected so that the GTH Transceiver clock generator can match the required GTH "line rate" with rational values of "M" and "N" or whatever these integers are called in the Virtex-7 GTH clock generator. The optimal GTH reference frequency, i.e. the frequency that will be used for the Hub's 2nd external PLL is not yet known. I assume that people want to use LHC locked GTH line rates - but I do not know this for certain. As you know CMX-Topo use a LHC locked GTX line rate around 6.4 Gb/s and a 320.64 MHz reference was good for the Virtex-6 GTX generator at this line rate. If the two proposed line rates are: Phase I 4.8 Gb/s and Phase II 9.6 Gb/s then it is quite likely that the Hub could run at either, using the same GTH reference frequency, by just changing "M" and "N" values in the GTH clock generator (while still using nice comfortable values of M and N). If this is not the case then we would either need to swap the 2nd higher frequency PLL between Phase 1 and Phase 2 or more likely just put both on the Hub to start with. The GTH clock generator has a mux at its reference input so it is trivial to switch from one external GTH reference to another.

GBT in FPGA https://espace.cern.ch/GBT-Project/GBT-FPGA/default.aspx

The GBT chip is a radiation tolerant ASIC that can be used to implement bidirectional multipurpose 4.8 Gb/s optical links for high-energy physics experiments. It will be proposed to the LHC experiments for combined transmission of physics data, trigger, timing, fast and slow control and monitoring. Although radiation hardness is required on detectors, it is not necessary for the electronics located in the counting rooms, where the GBT functionality can be realized using Commercial Off-The-Shelf FPGAs.

Having a transposition of the GBTserdes chip into FPGAs would thus be very useful, not only for the counting room GBTs, but also to emulate the GBT chip before its actual release, and to design test platforms for GBT testing and system validation. A team located in Marseille (CPPM) and at CERN implemented the GBT protocol in Altera and Xilinx FPGAs and made it available to users via SVN. The current implementations are based on Altera StratixIIGX and Xilinx Virtex5 and will progressively be completed with StratixIV and Virtex6 designs and optimization techniques. They constitute a Firmware Starter Kit to get familiar with the GBT protocol.

FELIX

FELIX (Front-End LInk eXchange) [7] https://twiki.cern.ch/twiki/bin/view/Atlas/GBT2LAN is an interface between custom data links connected to on-detector electronics (in our case – to the Hub) and an off-detector industry standard network. It uses the CERN Giga Bit Transceiver (GBT) development

[8] and its capability to carry detector data, timing and fast control information, and slow control and monitoring data.

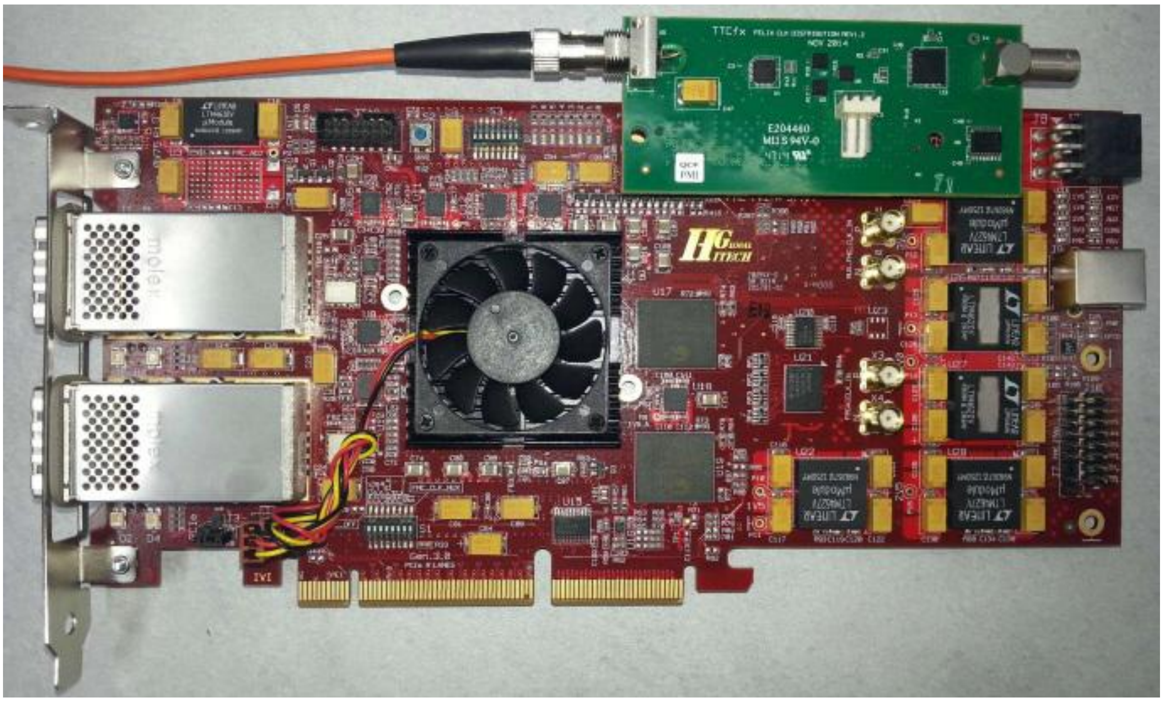


Figure 9: TTCfx FMC (green) with connectors (mounted on the Hitech Global HTG-V7-PCIE-CXP): ST fiber (left), cleaned clock (centre) and LEMO (right).

The TTC FMC for FELIX, TTCfx, works with the legacy LHC TTC system to provide TTC information and a jitter-cleaned Bunch Crossing clock for an FPGA board with a standard FMC connector. A clock and data recovery chip (CDR) provides a 4x BC clock and the data stream consisting of the TTC "A" and "B" channels inter-leaved. In addition there is a jitter cleaner chip. The CDR is the same as that used in the GLIB project's TTC FMC and the jitter cleaner is the same as that used on the GLIB board itself.

3.1.1. GBT link data format (FELIX to Hub)

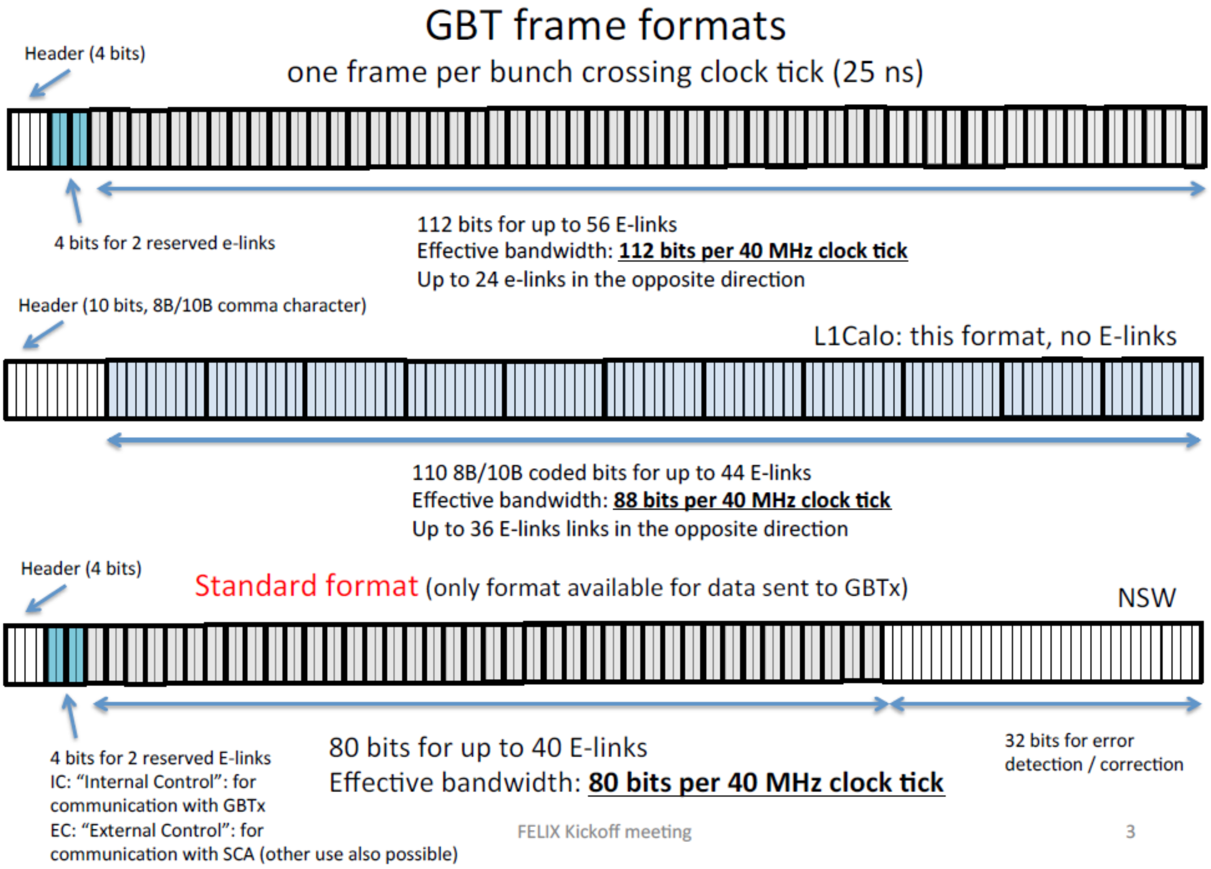
The standard encoded TTC signal will arrive to FELIX via a standard TTC fiber and will be decoded by a TTCrq or equivalent FPGA firmware. TTC data will be stuffed, on each BC clock, with fixed latency, directly into all "GBT frame format" with the "TTC" attribute, as follows:

- 2-bit E-link: the raw TTC A and B channels. Destination must decode the two serial streams.
- 4-bit E-link: L1A, BCR, ECR, system[3] from the 8-bit TTC broadcast packet
- 8-bit E-link: L1A, BCR, ECR, system[3..0], user[7] from the 8-bit TTC broadcast packet

The 120-bit "GBT frame format" is transmitted during a single LHC bunch crossing interval (25 ns), resulting in a line rate of 4.8 Gb/s. In the "Standard" format four bits are used for the frame Header (H) and 32 are used for Forward Error Correction (FEC). This leaves a total of 84 bits for data transmission corresponding to a user bandwidth of 3.36 Gb/s. Of the 84-bits, 4 are always reserved for Slow Control information (Internal Control (IC) and External Control (EC) fields), leaving 80-bits for user Data (D) transmission. The 'D' and EC fields use is not pre-assigned and can be used indistinguishably for Data Acquisition (DAQ), Timing Trigger & Control (TTC) and Experiment Control (EC) applications.

What is the GBT/TTC data format, sent by FELIX to the Hub - Standard?

How to extract Clock and TTC information from the GBT data - on the Hub or in the FPGA?



GBT frame formats

one frame per bunch crossing clock tick (25 ns)

Header (4 bits)

4 bits for 2 reserved e-links

112 bits for up to 56 E-links
Effective bandwidth: **112 bits per 40 MHz clock tick**
Up to 24 e-links in the opposite direction

Header (10 bits, 8B/10B comma character)

L1Calo: this format, no E-links

110 8B/10B coded bits for up to 44 E-links
Effective bandwidth: **88 bits per 40 MHz clock tick**
Up to 36 E-links links in the opposite direction

Header (4 bits)

Standard format (only format available for data sent to GBTx)

NSW

4 bits for 2 reserved E-links
IC: "Internal Control": for communication with GBTx
EC: "External Control": for communication with SCA (other use also possible)

80 bits for up to 40 E-links
Effective bandwidth: **80 bits per 40 MHz clock tick**

32 bits for error detection / correction

FELIX Kickoff meeting

3

Figure 10: GBT frame formats.

3.1.2. *ROD readout-control data format (ROD to Hub FPGA)*

3.1.3. *Hub FPGA output data format*

The format and content of the TTC-info stream broadcast from the Hub to the FEX shelf should be defined in the L1Calo ATCA Backplane Usage specification. The information broadcast should include the source of the clock transmitted by the Hub (TTC or local crystal) and whether or not a valid TTC input was received by the Hub. This information should be broadcast regularly.

3.2. GBT/TTC DATA PROCESSING

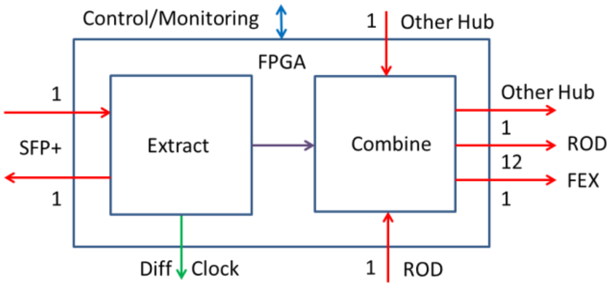


Figure 11: GBT/TTC data processing.

3.2.1. *GBT/TTC Data FPGA interface*

INPUTS

GTH receivers:

- 1 receiver - Receiver the SFP+ Optical Timing signal
  - Line rate: 4.809444 Gb/s Ref. clock:
- 1 receiver - Receiver ROD "Readout Control" Data from the ROD on This Hub
  - Line rate: x.x Gb/s Ref. clock:
- 1 receiver - Receiver ROD "Readout Control" Data from the ROD on the Other Hub
  - Line rate: x.x Gb/s Ref. clock:

OUTPUTS

GTH transmitters:

- 1 transmitter - Signal to the SFP+ Transmitter
  - Line rate: x.x Gb/s Ref. clock:
- 1 transmitter - Send the combined TTC + ROD1 + ROD2 Data to ROD on This Hub
  - Line rate: x.x Gb/s Ref. clock:
- 1 transmitter - Send the combined TTC + ROD1 + ROD2 Data to Other Hub
  - Line rate: x.x Gb/s Ref. clock:
- 12 transmitters - Send the combined TTC + ROD1 + ROD2 Data to 12 FEX
  - Line rate: x.x Gb/s Ref. clock:

General IO:

- 1 recovered 40.08 MHz TTC Clock output (diff)

3.2.2. GBT/TTC Data processing in FPGA

Extract the "Clock" and the "TTC Information" content from the incoming Optical signal. The Clock will be immediately sent out of the FPGA.

The "TTC Information" will be combined with the “readout-control data” (former "back data") from the ROD on This Hub and with the “readout control data” from the ROD on the Other Hub.

This combined stream “TTC Information” from the Optical Timing signal + ROD 1 “readout-control data” + ROD 2 “readout-control data” will come out of the Hub's FPGA on a GTH Transceiver output. On the Hub this GTH signal will be fanned out and sent to the ROD on This Hub and sent over the Fabric Interface to the FEX cards and to the Other Hub.

3.2.3. ROD readout-control data processing

3.2.4. SFP+ control/monitoring interface

Serial link and control of SFP+ optical module

INPUT

General IO:

- 3 SFP+ status signals (MOD_PRESENT, RX_LOST, TX_FAULT)

INPUT/OUTPUT

General IO:

- SDA – SFP+ I2C bidirectional serial data

OUTPUT

General IO:

- SDL – SFP+ I2C clock for the serial data
- 1 SFP+ control signal (TX_DISABLE)

3.2.5. Clock interface

INPUTS

General IO:

- 2 Global Clock inputs from the Hub

4. IPBUS (ETHERNET - NIC)

4.1. IPBUS DESCRIPTION

Hub: An IPbus interface is provided for high-level, functional control of the FEX-Hub module. This allows, for example, any firmware parameters to be set, modes of operation to be controlled and monitoring data to be read. Figure 12 shows the Hub's Base Interface Ethernet Switch in the context of the other cards in the ATCA shelf.

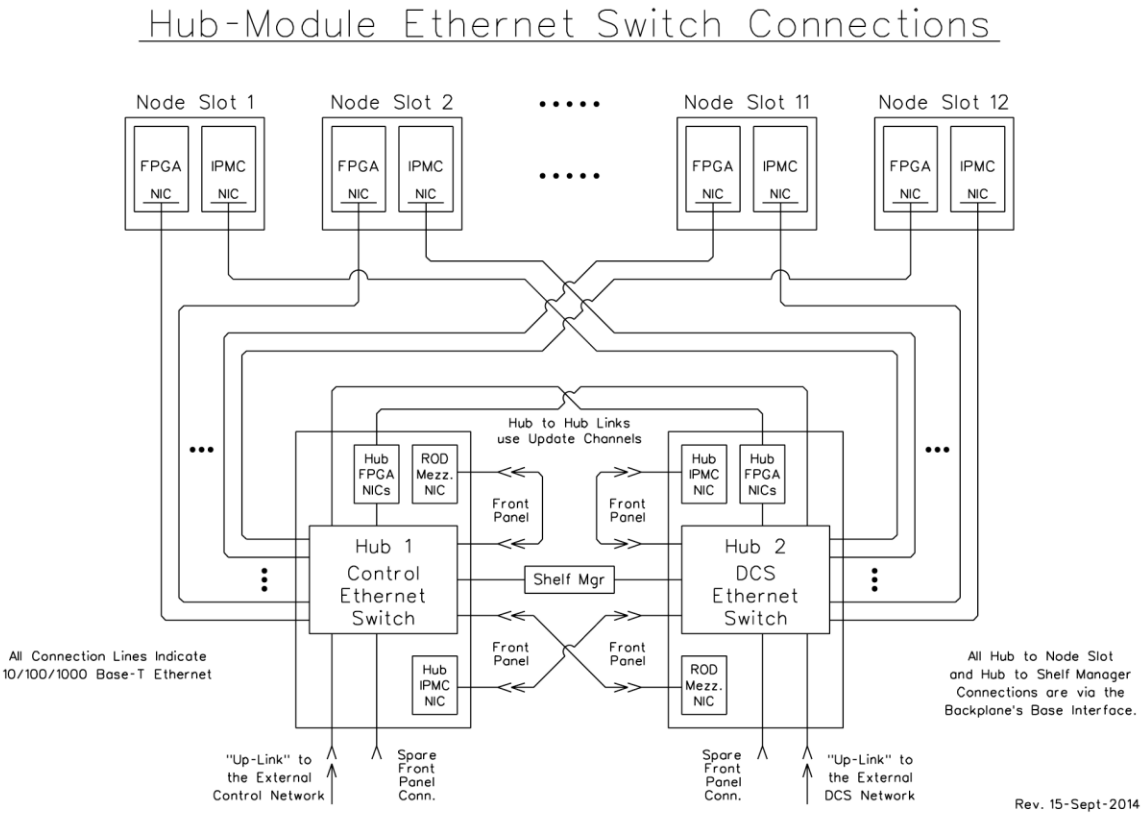


Figure 12: Illustration of FEX-Hub Ethernet network connections.

ROD: An Ethernet link is provided from the main ROD FPGA to the Ethernet switch on the Hub. This will allow a computer using IPbus to:

- Access registers within the ROD FPGA, setting parameters and controlling modes of operation.
- Store FPGA configurations into the SPI-Flash Configuration Memory.
- Initiate the loading of configurations from the SPI-Flash.

This can be used to load a configuration from one of a number of other SPI-Flash sectors. These sectors can be written via IPbus.

4.1.1. IPbus protocol

The IPbus [9] protocol is a simple packet-based control protocol for reading and modifying memory-mapped resources within FPGA-based IP-aware hardware devices which have a A32/D32 bus.

It defines the following operations:

- **Read** A read of user-definable depth. Two types are defined: address-incrementing (for multiple continuous registers in the address space) and non-address-incrementing (for a port or FIFO).

- **Write** A write of user-definable depth. As with reads, two types of write are defined: incrementing and non-incrementing.
- **Read-Modify-Write bits (RMWbits)** An atomic bit-masked write, defined as X := (X &A) jB. This allows one to efficiently set/clear a subset of bits within a 32-bit register.
- **Read-Modify-Write sum (RMWsum)** An atomic increment operation, defined as X := X +A, which is useful for adding values to a register (or subtracting, using two's complement).

The protocol is transactional—for each read, write or RMW operation, the IPbus client (typically software) sends a request to the IPbus device; the device then sends back a response message containing an error code (equal to 0 for a successful transaction), followed by return data in case of reads. In order to minimise latency, multiple transactions can be concatenated into a single IPbus packet.

The protocol lies in the application layer of the networking model and is network protocol agnostic. TCP exhibits various highly-desirable features of a transport protocol, such as reliable, ordered data transmission and congestion avoidance; however, the underlying algorithm is significantly more complex than for the other ubiquitous transport protocol, UDP. Since the IPbus device implementation must have a low FPGA resource usage, UDP has been chosen as the transport protocol. Version 2.0 of the IPbus protocol (finalised in early 2013) includes a reliability mechanism over UDP, through which the client can correct for any packet loss, duplication or reordering. This mechanism is credit-based with a fixed number of packets in flight, giving implicit traffic shaping which can avoid congestion-based performance degradation, such as TCP Incast.

4.1.2. Firmware and software suite

The IPbus software and firmware suite consists of the following components:

- **IPbus firmware** A module that implements the IPbus protocol within end-user hardware
- **ControlHub** Software application that mediates simultaneous hardware access from multiple
- **mHAL** clients, and implements the IPbus reliability mechanism over UDP

End-user instructions and source code for these components are available through the CERN CACTUS (Code Archive for CMS Trigger UpgradeS) website http://cactus.web.cern.ch/ and SVN repository. The software is packaged as RPMs for Scientific Linux versions 5 and 6, and available through a YUM repository.

IPbus firmware

The IPbus 2.0 firmware module is a reference system-on-chip implementation of an IPbus 2.0 UDPserver in VHDL; it interprets IPbus transactions on an FPGA. It has been designed as a common module to run alongside a device's main processing logic (e.g. trigger algorithms) on the same FPGA, only using resources from within the FPGA. Any loss, re-ordering or duplication of the IPbus UDP packets is automatically corrected by the ControlHub using the IPbus reliability mechanism.

The IPbus firmware module has been designed to be simple to integrate into variety of platforms, and there are example designs for several development boards and standard platforms. The source code is currently Xilinx-specific, but has been successfully adapted for Altera devices. The firmware is modular, with a core protocol decoder and bus master controlling the interface to the IPbus slaves, and a number of interfaces into the decoder with simple arbitration between them. As well as the UDP interface, there are SPI/I2C interfaces and chip-to-chip bridges allowing control from microcontrollers and between FPGAs. The UDP interface is monolithic, operating at the network layer in order to eliminate unnecessary internal buffering. It also implements: the echo request/reply semantics from ICMP (RFC 792, used in the UNIX ping command); ARP (RFC 826, used for resolving IP addresses into MAC addresses); and RARP (RFC 903, used for requesting an IP address on startup). Several parameters are configurable at build time, including: the Ethernet frame MTU; the number of buffers for incoming/outgoing IPbus packets which determines the maximum possible control throughput; and the method used for IP address assignment—fixed IP address, RARP, or a secondary out-of-band IPbus controller (for instance an onboard microcontroller).

The topologies of an IPbus control system in some common scenarios are shown below. The simplest system (upper left) is a single target running the IPbus firmware, directly connected by a single Ethernet cable to a computer running a C++/Python control application based on the mHAL library. This is the typical layout during early hardware development.

CACTUS: http://cactus.web.cern.ch/
Firmware: https://svnweb.cern.ch/trac/cactus/wiki/IPbusFirmware
Tutorial: https://svnweb.cern.ch/trac/cactus/wiki/uhalQuickTutorial

4.2. IPBUS DATA PROCESSING

4.2.1. IPbus interface

Two FPGA MACs are connected to the Physical chips ksz9031rnx via RGMII ports. This chip has both the RGMII signal connection to the FPGA that is used to move the actual Ethernet data and provides access to internal registers and also has a 2 wire serial "Management Data" port.

INPUT/OUTPUT

General IO:

- 20 signals - 2 RGMII ports (10 fast signals each) to the Hub FPGA's Phys Chips.

4.2.2. IPbus Data processing in FPGA

ROD:

3.2.4 Other Interfaces and Signal Paths:
IPbus for control, performance monitoring, data snooping, and download of configuration data – needs its own NIC to be an Ethernet connection.

5.4.1 IPbus
An Ethernet link is provided from the main ROD FPGA to the Ethernet switch on the Hub. This will allow a computer using IPbus to:
Access registers within the ROD FPGA, setting parameters and controlling modes of operation.
Store FPGA configurations into the SPI-Flash Configuration Memory.
Initiate the loading of configurations from the SPI-Flash.

eFEX:

4.6 Slow Control:
An IPBus interface is provided for high-level, functional control of the eFEX. This allows, for example, algorithmic parameters to be set, modes of operation to be controlled and spy memories to be read.
IPBus is a protocol that runs over Ethernet to provide register-level access to hardware. Here, it is run over a 1000BASE-T Ethernet port, which occupies one channel of the ATCA Base Interface. On the eFEX there is local IPBus interface in every FPGA, plus the IPMC. These interfaces contain those registers that pertain to that device. The Control FPGA implements the interface between the eFEX and the shelf backplane, routing IPBus packets to and from the other devices on as required. The Control FPGA also contains those registers which control or describe the state of the module as a whole. For those devices such as Minipods, which have an I2C control interface, an IPBus-I2C bridge is provided.

4.7 Environment Monitoring:
The eFEX monitors the voltage and current of every power rail on the board. It also monitors the temperatures of all the FPGAs, of the Minipod receivers and transmitters, and of other areas of dense logic. Where possible, this is done using sensors embedded in the relevant devices themselves. Where this is not possible, discrete sensors are used.
The voltage and temperature data are collected by the eFEX IPMC, via an I2C bus. From there, they are transmitted via IPBus to the ATLAS DCS system. The eFEX hardware also allows these data to be

transmitted to the DCS via IPMB and the ATCA Shelf Controller, but it is not foreseen that ATLAS will support this route.

If any board temperature exceeds a programmable threshold set for that device, IPMC powers down the board payload (that is, everything not on the management power supply). The thresholds at which this function is activated should be set above the levels at which the DCS will power down the module. Thus, this mechanism should activate only if the DCS fails. This might happen, for example, if there is a sudden, rapid rise in temperature to which the DCS cannot respond in time.

4.2.3. Switch control/monitoring interface

INPUT/OUTPUT

General IO:

- 6 signals - SPI serial links to each of 3 Switch Chips

5. IPMC INTERFACE

http://www.intel.com/content/www/us/en/servers/ipmi/spec-license-agreement.html

5.1. IPMI DESCRIPTION

The Hub monitors the voltage and current of every power rail on the board. It also monitors the temperatures of FPGAs, of the Minipod transmitter (if installed), and of other areas of dense logic. Where possible, this is done using sensors embedded in the relevant devices themselves. Where this is not possible, discrete sensors are used. The voltage and temperature data are collected by the IPMC, via an I2C bus. From there, they are transmitted via Ethernet to the ATLAS DCS system. The Hub hardware also allows these data to be transmitted to the DCS via IPMB and the ATCA Shelf Controller, but it is not foreseen that ATLAS will support this route.

[6.7 The IPM Controller] For the purposes of monitoring and controlling the power, cooling and interconnections of a module, the ATCA specification defines a low-level hardware management service based on the Intelligent Platform Management Interface standard (IPMI). The Intelligent Platform Management (IPM) Controller is that portion of a module (in this case, the FEX-Hub) that provides the local interface to the shelf manager via the IPMI bus. It is responsible for the following functions:

- interfacing to the shelf manager via dual, redundant Intelligent Platform Management Buses (IPMBs); it receives messages on all enabled IPMBs and alternates transmissions on all enabled IPMBs;
- negotiating the Hub power budget with the shelf manager and powering the Payload hardware only once this is completed;
- managing the operational state of the Hub, handling activations and deactivations, hot-swap events and failure modes;
- implementing electronic keying, enabling only those backplane interconnects that are compatible with other modules in shelf, as directed by shelf manager;
- providing to the Shelf Manager hardware information, such as the module serial number and the capabilities of each port on backplane;
- collecting, via an I2C bus, data on voltages and temperatures from sensors on the Hub, and sending these data, via IPBus(?), to the main Hub FPGA;
- driving the BLUE LED, LED1, LED2 and LED3.

The Hub uses the IPMC mezzanine produced by LAPP as the IPM Controller [1.11]. The form factor of this mezzanine is DDR3 VLP Mini-DIMM.

From Hub PDR:

8. The monitoring scheme designed for the eFEX, which allows data from the sensors to be sent either to the DCS system or read out via IPBus, should be shared with the design teams of the Hub and the other FEX modules (see also item **Error! Reference source not found.**).

11. In addition to sending environment data to the DCS system (lines 341–345 of the specification), a mechanism should be provided by which these data can be read out in the absence of the DCS system, via the IPBus control interface (see also items 8 and 28).

28. The mechanism described in the Hub specification for capturing monitoring data (lines 232–233, 342–345, 387–388, Figure 7 and possibly elsewhere) should be updated to describe the current scheme proposed by the Hub group and L1Calo. That is, the critical environment data (such as power levels and FPGA temperatures) are collected by the IPMC over I2C and then reported over IPMB to the Shelf Manager, from which the DCS system collects them. Additional monitoring data (such as Minipod information) are collected by the Hub FPGA and polled over IPBus (see also items 11 and 31).

31. The Hub specification should be updated to describe the use currently foreseen for the Hub-2 Ethernet network (lines 387-391). Namely, whilst this network does provide an Ethernet interface to

the shelf IPMC cards and could be used to collect data for the DCS, it is not expected to be used for this purpose. Rather, it provides the capacity to implement advanced IPMC functionality (should this become desirable) (See also item 28).

5.2. IPMI DATA PROCESSING

5.2.1. IPMI interface

Shelf Address and I2C access to FPGA from IPMI controller - to the System Monitor?

INPUT

General IO:

- 4-bit (?) Shelf Address (Shelf Number) retrieved from the Shelf Manager by the IPMC
- SDL – I2C clock for the serial data

INPUT/OUTPUT

General IO:

- SDA – I2C bidirectional serial data

Do we need the address pins to set the address in the I2C bus or we will set this address via IPbus?

5.2.2. IPMI Data processing

6. OTHER INTERFACES

6.1. MINIPODS INTERFACE

6.1.1. MiniPOD data interface

INPUTS

GTH receivers:

- 3 receivers - Signals from the Hub's MiniPOD Receiver
    - Line rate: x.x Gb/s Ref. clock:

OUTPUTS

GTH transmitters:

- 12 transmitters - Signals to Hub's MiniPOD Transmitter
    - Line rate: x.x Gb/s Ref. clock:

6.1.2. MiniPOD control/monitoring FPGA interface

INPUT/OUTPUT

General IO:

- SDA – MiniPOD I2C bidirectional serial data

OUTPUT

General IO:

- SDL – MiniPOD I2C clock for the serial data
- 2 MiniPOD control signal (RESET_B ?)

6.2. MISCELLANEOUS

http://www.pa.msu.edu/hep/atlas/l1calo/hub/hardware/details/hub_0_ab_trace_routing_strategy.txt

6.2.1. Hub LEDs

OUTPUT

General IO:

- ~4 – Hub LEDs ?

6.2.2. Front panel access signals

INPUT/OUTPUT

General IO:

- ~4 – Front Panel Access Signals ?

6.2.3. Other control/monitoring

INPUT/OUTPUT

General IO:

- ~1 – All Hub Power OK ?
- ~18 – Use of the FPGA's XADC and it power and reference ?
- ~16 – Life Boat Access Connector ?