RICH L1 Technical Manual

Clive Barham, Sven Katvars, Steve Wotton
University of Cambridge

Contents

RICH L1 Technical Manual 1
Contents 2
1 Introduction 4
2 L1 implementation 5
2.1 Clock domains 5
2.2 Ingress FPGA 6
2.3 Data Links 6
2.4 External ingress DRAM 6
2.5 Egress FPGA 6
2.6 External egress DRAM 6
2.7 TFC network 7
2.8 DAQ network 7
2.9 Board ID EEPROM 7
2.10 Temperature probes 7
2.11 ECS interface 7
2.11.1 Local bus 7
2.11.2 Summary of I2C devices 7
2.11.3 JTAG 7
2.12 Power supply and crate reset 8
2.13 L1 board mechanics 8
2.13.1 LED assignments (ingress) 9
2.13.2 LED assignments (egress) 10
2.14 Number of modules 10
2.15 Barcode 10
3 Ingress firmware 11
3.1 Front-end data characteristics 11
3.2 Deserialisation 11
3.3 L0 data ingress 11
3.4 Pixel masks 12
3.5 Input pre-processing 12
3.6 Zero-suppression 13
3.7 Ingress multiplexing 13
3.8 Transmission to the egress FPGA 14
4 Egress firmware 14
4.1 TFC interface 14
4.1.1 L0 emulation 14
4.1.2 MEP generation 14
4.1.3 Throttle 15
4.1.4 Sequence error detection 15
4.2 Ingress to egress data transmission 15
4.3 Egress multiplexing and MEP building 15
4.4 Ethernet formatting 16
4.5 Ethernet fragmentation 16
4.6 Data transmission 16
4.7 Flow-control 16
4.8 Reset strategy 16
5 Special run modes 16
5.1 ALICE mode 17
5.2 Running without TFC broadcasts 17
5.3 Fast resettable channel inhibit 17
6 Data format 18
6.1 The raw bank 18
6.1.1 L1 ingress header format 18
6.1.2 HPD data block formats 18
6.1.3 Non-zero-suppressed 19
6.1.4 Zero-suppressed 19
6.2 Zero-suppression scheme 20

6.2.1 Zero-suppressed pixel HPD data 20
6.3 Additional L1 monitoring data 21
7 Configuration registers 22
7.1 Miscellaneous control register 1 22
7.2 Miscellaneous control register 2 23
7.3 Resets 23
7.4 TFC latency compensation 23
7.5 Ingress control 23
7.6 Ingress configuration mailbox 23
7.6.1 Type[2..0]=000, Set operation mode 24
7.6.2 Type[2..0]=001, Set selected mask bits 24
7.6.3 Type[2..0]=010, Set all mask bits 24
7.6.4 Type[2..0]=011, Unset all mask bits 24
7.6.5 Type[2..0]=100, Actions 24
8 Status registers 25
8.1 General status 0 25
8.2 Inhibits 25
8.3 Flow control status 26
8.4 MEP builder status 26
8.5 Miscellaneous status 26
8.6 Egress input FIFO errors 26
8.7 TFC L0 and ROT counters 26
8.8 MEP counters 27
8.9 MEP ingress status 27
8.10 Alarms 27
8.11 Egress input FIFO 0,1 occupancy 27
8.12 Egress input FIFO 2,3 occupancy 27
8.13 Egress input FIFO 0,1 RX count 27
8.14 Egress input FIFO 2,3 RX count 28
8.15 Egress input FIFO 0,1 TX count 28
8.16 Egress input FIFO 2,3 TX count 28
8.17 GT frame lengths 28
8.18 Ethernet fragment counters 28
8.19 SPI3 flag counters 28
8.20 State machine states 28
8.21 FE per-channel registers 29
8.21.1 FE channel register 0x0, (0) 29
8.22 FE general status registers 29
8.22.1 FE general register 0x0, (0) 29
9 The 6U 12-channel module 30
9.1 The configuration interface 30
9.2 Software installation 30
9.3 Installing the USB drivers 30
9.4 Starting the GUI 30
9.5 Configuring the L1 board 31
9.6 Data acquisition 31
9.7 Other options 31
9.8 Data recording 31
9.9 Bugs, features and workarounds 31

1 Introduction

This document describes the implementation of the RICH L1 readout module. The main functions of the module are to perform data reduction on the incoming front-end events by performing zero-suppression and to multiplex and format the events for transmission to the global acquisition system.
Table 1 summarises the expected data flow requirements. The LHCb L0 accepted event rate budget allows for up to 3 additional clocks for header words plus one idle word between frames (800ns data + 75ns headers + 25ns idle). The table assumes that all 4 clocks are used and that 32 bits are transferred on each clock so that the quoted bandwidth into L1 represents an absolute maximum. The bandwidth to DAQ assumes the naïve estimate for the zero suppressed event size assuming 1% occupancy discussed in Section 6.2.

Bunch crossing rate	40MHz
L0 accept rate (average)	1MHz
L0 accepted event size (maximum)	(1024+4*32)*484 =557568bit
Bandwidth into L1	500Gbit/s
Number of links into L1	484
L1 accepted event size (average)	10.5kB
Input links per L1 module	36
Number of links into DAQ	4 per L1 board

Table 1 Summary of dataflow parameters

The following tables summarise the main parameters that influence the design of the module.

	No. of columns/rows per plane	No. of HPDs per column/row	No. of planes	HPD total
RICH1	7	14	2	196
RICH2	9	16	2	288

Table 2 Numbers of HPDs

Average event rate	1MHz
Number of detector channels	1024
Number of bits per digitised detector channel	1
Average channel occupancy	<10% (Estimated 1%)
Multiplexing at LHCbPIX output	1024:32
Multiplexer output clock speed	40MHz

Table 3 LHCbPIX output parameters

Number of LHCbPIX chips per module	2
Data frame aspect ratio	35×32bit@40MHz
Minimum interframe gap	1×32bit

Table 4 L0 front-end module output parameters (PINT)

Data frame aspect ratio	70×16bit@80MHz
Minimum interframe gap	2×16bit IDLE codes
Multiplexing factor at output link	20:1 (8B/10B encoding)
Output serial link clock speed	1.6GHz
Effective maximum output bandwidth/fibre	1.28Gbit/s
Number of output data links	2 (1 per LHCbPIX)

Table 5 Gigabit optical link parameters

Maximum output bandwidth per link	1.00Gbit/s
Number of outputs per link mezzanine	4
Link standard	IP on Gbit Ethernet
Maximum IP frame size	64kbyte
Maximum Gbit Ethernet packet payload	1500byte
Number of events per MEP	Variable

Table 6 Output (DAQ) link parameters

2 L1 implementation

The L1 module is implemented using the Xilinx Virtex-II Pro FPGA series which incorporate up to 20 multi-gigabit serializer/deserializer circuits within each FPGA. Full details of the features available in the VirtexII Pro FPGAs can be found in the Xilinx Virtex II Pro data sheet, DS083 and the Xilinx RocketIO Transceiver User Guide, UG024.

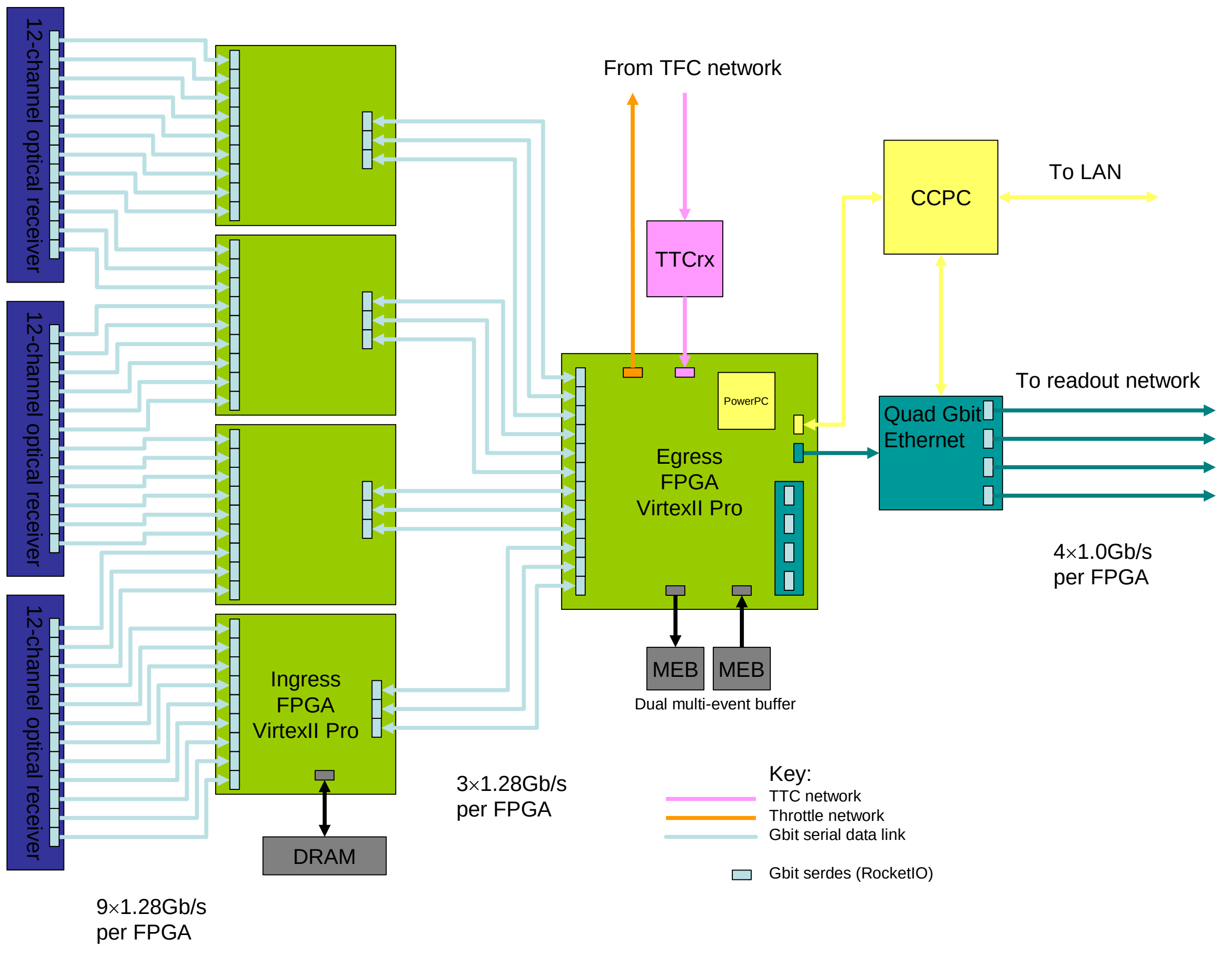


Figure 1 Block diagram of the L1 module

A block diagram of one of the modules is shown in Figure 1 which shows the principal components, data and control paths. L0 accepted events are received on optical links at the left of the diagram. The data flows from left to right in the diagram and are transmitted to the readout network using the Gbit Ethernet interface at the right. The various components are described in more detail in the following sections.

2.1 Clock domains

The principal clock domain on the L1 module is defined by an 80.44MHz local oscillator having a slightly higher frequency than the 80.157MHz LHC clock frequency.
Each input channel defines a unique clock domain. The frequencies of all the input clock domains are identical (equal to the LHC clock frequency) but the phase is in general different for each due to variations in the channel-to-channel signal propagation delays. Other clock domains include:

- The TTC clock domain sourced by and synchronous to the TTCrx CLK40 signal.
- The ECS local bus clock sourced by a 20MHz crystal oscillator.
- The GBE SPI3 bus clock sourced by a crystal oscillator or FPGA clock multiplier.

2.2 Ingress FPGA

The Xilinx Virtex II Pro FPGA is used for the implementation of the L1 board. This series of FPGAs contain integrated Gbit serial data transceivers that are well matched to the serial data arriving from the front-end electronics. Cost optimisation has resulted in the choice of the P40 VirtexII Pro FPGA for the ingress path. Each P40 contains 12 RocketIO Gbit transceivers. 9 transceivers are used to receive data from the front end and the remaining 3 transceivers are used to implement the control and data path between the ingress and egress FPGA. 4 FPGAs of this type therefore allow the serial paths from 3 12-channel receivers to be distributed across 4 P40 FPGAs.

The P40 also contains sufficient integrated dual-port RAM to allow the input buffering to be done inside the FPGA without the need for external memory. The ingress firmware requires only a small fraction of the available P40 FPGA programmable logic resources for the implementation.

Each ingress FPGA has a dedicated flash memory (Xilinx XCF32P, Xilinx datasheet DS123) containing the FPGA bitstream. When "master serial" configuration mode is selected (switch SW6 set to "MS"), the bitstream is downloaded on power up or by pressing the frontpanel "CFG" button. In "boundary scan" configuration mode (switch SW6 set to "BS") download of the bitstream is by JTAG only.

2.3 Data Links

The L0 accepted events arrive on unidirectional optical links operating at 1.6Gb/s. There is no flow control at the link level. Industry (and LHCb) standard 12-fibre ribbons are used. The data are received at the L1 modules by Agilent 12-channel optical receivers (HFBR-782B) and the data are deserialised using the VirtexII Pro RocketIO interfaces. Each RocketIO interface performs the following functions:

- Implements a single 16:1 serialiser and 1:16 deserialiser channel of which the serialiser channel is not used.
- Performs 8b/10b decoding.
- Synchronises the received data to the system clock using an integral "elastic buffer" and integrated clock correction logic.

12 RocketIO interfaces are therefore required for each 12-channel optical receiver. The RocketIO deserializers require a reference clock whose frequency is within 100ppm of the clock that is used as the reference clock for the data transmission (the LHC clock via the TTC network). An 80.157MHz crystal oscillator is used for this purpose. The signals from the optical receivers are AC coupled.

2.4 External ingress DRAM

256Mbit of external DRAM memory (Micron MT46V16M16 DDR) are mounted per input channel. Independent control and data lines are routed to each memory chip but groups of three memories share address lines. The memory is deep enough to buffer about 250k events per channel in LHCb mode. The large memory buffer was designed to accommodate the large latency of the L1 trigger in the LHCb architecture. The change to 1MHz readout in LHCb has removed the requirement for a large L1 latency buffer therefore the external ingress buffers are not expected to be used although they remain on the board in case of need.

The memory is organised in three banks of three parallel memories. Each bank can be operated independently. Each bank is configured to operate as a 48-bit wide, dual ported, DDR memory operating at a clock speed of 80MHz to give an effective read and write bandwidth of 48bit@160MHz per bank before overheads (refresh cycles, turnaround).

2.5 Egress FPGA

A single P50 VirtexII Pro FPGA having 16 RocketIO Gbit transceivers provides the resources for the implementation of the output multiplexing, formatting and control of the L1 board. 3 RocketIO transceivers are connected to each of the 4 ingress FPGAs to provide the principle data and control path between ingress and egress FPGAs. These links operate synchronously to the system clock. Each serial link is equivalent to a 16bit bus operating at 80MHz.

The egress FPGA has a dedicated flash memory (Xilinx XCF32P, Xilinx datasheet DS123) containing the FPGA bitstream. When "master serial" configuration mode is selected (switch SW6 set to "MS"), the bitstream is downloaded on power up or by pressing the front-panel "CFG" button. In "boundary scan" configuration mode (switch SW6 set to "BS") download of the bitstream is by JTAG only.

2.6 External egress DRAM

4 256Mbit Micron MT46V16M16 DDR memories provide buffering for formatted data before transmission to the DAQ network if required. In fact, there may be sufficient memory resources within the egress FPGA so that the external memory may not be required. The memory is organised in two banks of two parallel memories. Each bank can be operated independently. Each bank is configured to operate as a 32-bit wide, dual ported, DDR memory operating at a clock speed of 80MHz to give an effective read and write bandwidth of 32bit@160MHz before overheads (refresh cycles, turnaround).

2.7 TFC network

Interfacing to the TFC network is through the TTCrx mounted directly on the L1 board and connected to the egress FPGA. The TTCrx is AC coupled to the optical receiver mounted at the back of the board. The TTCrx address is defined using pull-ups and pull-downs attached in FPGA firmware to the TTCrx IO pins and can therefore be set to any desired value. Decoding of TTC channel A and long and short broadcasts of TTC channel B are supported.
The egress FPGA also sources two LVDS signals that are routed to a back panel connector for connection to the TFC throttle network.

2.8 DAQ network

The interface to the DAQ network is provided by the LHCb GBE mezzanine card. Configuration of the GBE is done via the CCPC interface while the data transmission is implemented in the egress FPGA through the GBE SPI3 interface. The GBE TX and RX clocks are sourced from separate IOs of the egress FPGA. The clock speed is programmable in firmware by selecting either an external crystal oscillator source or using the Virtex II Pro clock multiplier.

2.9 Board ID EEPROM

A serial EEPROM (Microchip 24LC024) is mounted to store the board ID according to the LHCb convention [EDMS 508771]. The EEPROM is connected to ECS I2C bus number 1 at address 0x50. The first 8 nibbles of the EEPROM contain the hexadecimal code (LSB at right):

`601rssss`

where `r` is the board hardware revision ID and `ssss` is the board serial number. For production boards `r=2.` The preseries boards have `r=1`.The serial number is the same as the numeric part of the serial number on the module's barcode. A write-protect switch is provided on the back edge of the module.

2.10 Temperature probes

Three temperature probes (Texas/Burr-Brown TMP100) are mounted on the underside of the board and are connected to I2C bus 1 at addresses 0x48, 0x4a and 0x4c.

2.11 ECS interface

The CCPC and gluecard provide the primary configuration and control interface. The UKL1 board routes the gluecard PLX local bus to the GBE card and also to the egress FPGA. The serial port of the CCPC/gluecard is routed to a back panel connector but is not foreseen to be mounted on production boards. In the remainder of the document no distinction will be drawn between the CCPC and gluecard; they are referred to as the "ECS interface".

2.11.1 Local bus

The ECS local (PLX) bus clock is sourced by a 20MHz crystal oscillator. Only address bits A[13..0] are connected to the egress FPGA. CS1* is also connected.

2.11.2 Summary of I2C devices

All I2C devices are on bus number 1 (gluecard convention).

Device	I2C addresses	Notes
Board ID	0x50	Fixed
Temperature probe 0	0x48	Fixed
Temperature probe 1	0x4a	Fixed
Temperature probe 2	0x4c	Fixed
GBE	0x57	3 LSB set by pull-ups/pull-downs
TTCrx	0x42-0x43	Set in firmware

Table 7 I2C addresses

2.11.3 JTAG

The FPGA configuration is done using JTAG. The FPGA flash memory devices and FPGAs are connected to ECS TAP 3 (gluecard convention). A local connector is also provided. The two sources (ECS remote, or local connector) are selectable with a front-panel switch.
The FPGAs and their configuration PROMs appear on the JTAG chain as follows:

TDI → PROM1 → FPGA1 → PROM2 → FPGA2 → PROM3 → FPGA3 → PROM4 → FPGA4 → PROM5 → FPGA5 → TDO.
The TTCrx and GBE are normally bypassed using soldered links on the UKL1 board.

2.12 Power supply and crate reset

Power is supplied through the standard LHCb connector. Only 5V and 3.3V are used. Distributed local voltage regulation provides other voltage levels. Each module requires about 13A at 3.3V and around 1A at 5V.
The crate reset derived from the LHCb power connector resets only the ECS interface. A front panel reset button is OR'ed with the crate reset.
Reload of the FPGA bitstreams from the on-board flash memory is triggered by the ECS GC_CONFIG signal or by using a front-panel button.

2.13 L1 board mechanics

The UKL1 module is a 9U high module designed to be accommodated in the LHCb "TELL1" crate (see photo Figure 2). The front and back panel connector configuration is illustrated in Figure 3.

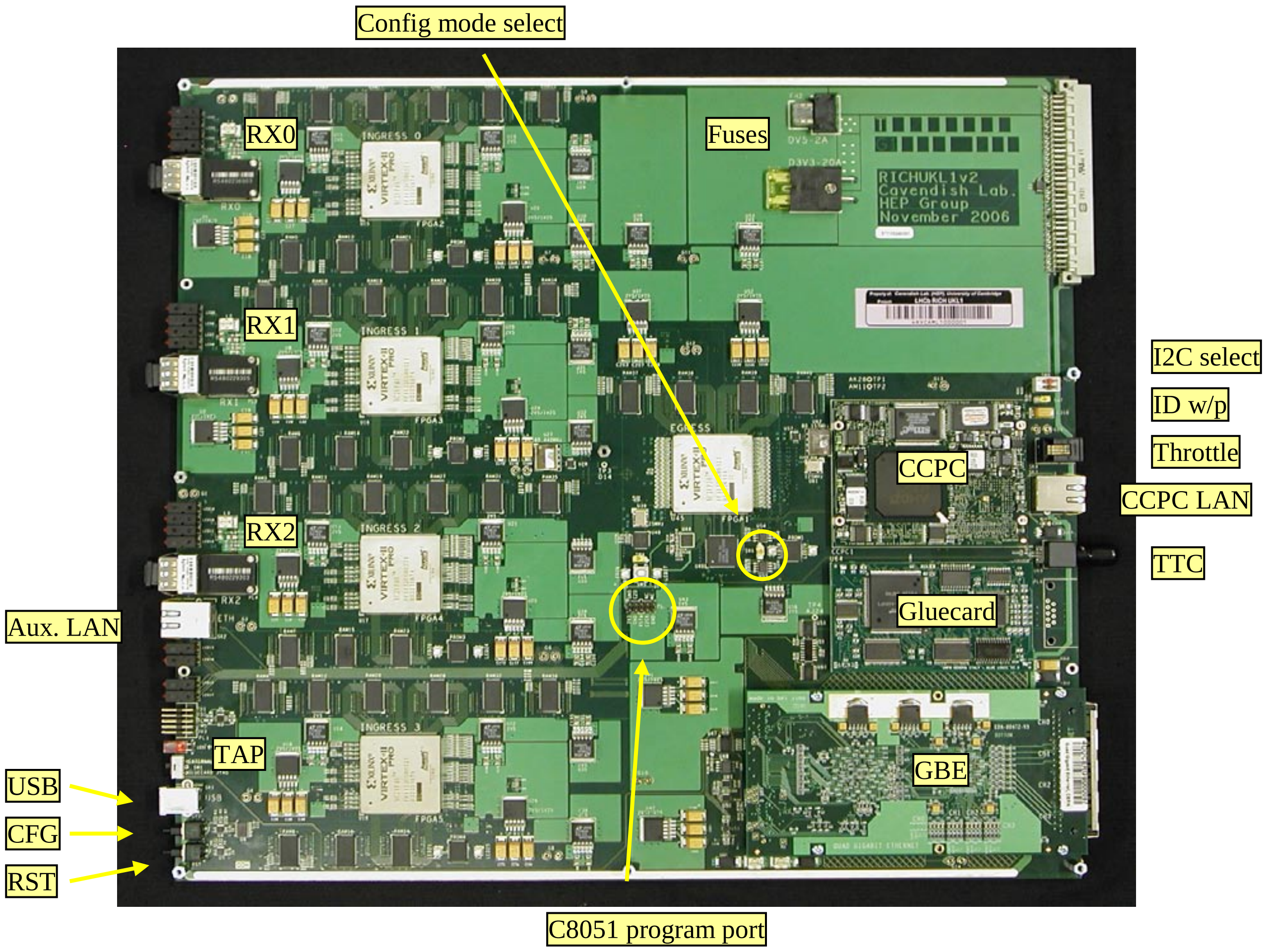


Figure 2 Photograph of UKL1

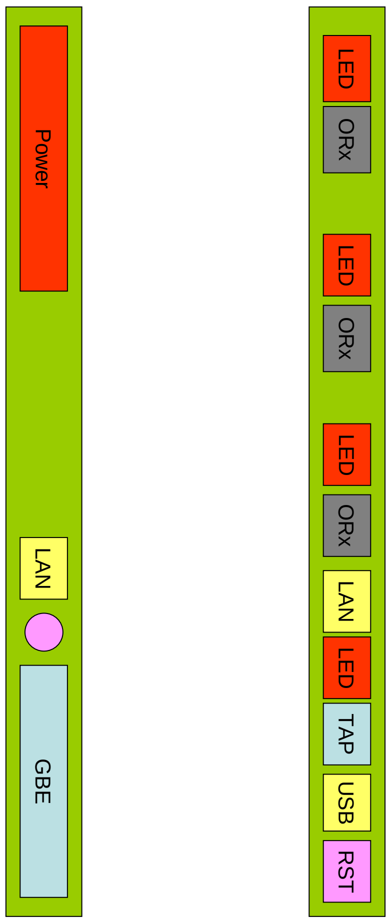


Figure 3 UKL1 back and front panel

The front panel incorporates:

- 3 12-channel Agilent optical receiver modules,
- 4 banks of 9 programmable status LEDs driven by the 4 ingress FPGAs,
- 12 programmable LEDs driven from the egress FPGA,
- USB connector,
- Auxiliary 100Mbit Ethernet connector,
- Local TAP connector,
- TAP source selection switch (local/remote),
- Reset button and FPGA configuration buttons.

The back panel incorporates:

- GBE interface,
- ECS 100Mbit LAN connector,
- TFC TTC fibre connector,
- TFC throttle connector

2.13.1 LED assignments (ingress)

Three groups of 12 red LEDs are used to indicate the status of each input channel. The assignment of the input channels to LEDs is not intuitive and is illustrated in the diagram below. Each pair of numbers in a cell identifies the ORx#/fibre# of the input.

0/2	0/1	0/0
0/6	0/7	0/8
0/11	0/10	0/9
1/11	1/10	1/9

0/5	0/4	0/3
1/6	1/7	1/8
2/9	2/10	2/11
1/2	1/1	1/0

1/5	1/4	1/3
2/5	2/4	2/3
2/8	2/7	2/6
2/2	2/1	2/0

2.13.2 LED assignments (egress)

A group of 12 LEDs (3 green, 3 yellow and 3 red) indicate various status signal driven by the egress FPGA. The assignments are summarised in the diagram below.

DCM	TFC	MEP full
LCK	MEP	Throttle
RDY	GBE	GBE busy
TTC	FE	TFC SQE

Key	Description
DCM	Digital clock managers ready
LCK	DCMs locked
RDY	L1 ready
TTC	TTCrx ready
TFC	TFC broadcast
MEP	MEP ready for transmission
GBE	GBE SPI3 Tx bus active
FE	Ingress FPGA link activity
MEP full	MEP buffer full
Throttle	Throttle asserted
GBE busy	No GBE ports available
TFC SQE	Trigger sequence error

2.14 Number of modules

Since the advent of the 1MHz operation mode and with the possibility of higher luminosity running the number of required L1 modules is now driven by the output bandwidth requirement. Before the 1MHz running scenario the number of modules required was 6 for RICH1 and 8 for RICH2. As a result of a load-balancing exercise for 1MHz running an additional 10 modules are foreseen to accommodate the increased output bandwidth. For partitioning reasons, the L1 modules for RICH1 and RICH2 will occupy separate racks. A single crate in each rack is therefore sufficient to accommodate the full complement of modules.

2.15 Barcode

The RICH UK L1 boards carry the following LHC bar-coded identifier:

`4RXCAML1xnnnnn`

The last 8 characters are the module serial number. The first two characters of the serial number are set to `L1` and the third character , `x`, encodes the module subtype. For prototypes `x=P` while for production modules `x` is the most significant digit (normally `x=0`) of the numeric serial number, `nnnnn`.

3 Ingress firmware

The following sections outline the implementation of the firmware in the ingress FPGAs. These FPGAs receive the L0 data, perform zero-suppression and multiplex the 9 input channels event-wise before transmitting the multiplexed data to the egress FPGA.

3.1 Front-end data characteristics

In the absence of errors, data from the front-ends arrive as contiguous blocks of fixed length with a minimum gap of 2 16-bit words between blocks. However, the front-end data sources may be subject to SEU so the receiver logic should be robust against errors in the front-end data.
Two classes of error can be distinguished:

- Temporary errors such as bit errors that do not disrupt the overall event synchronisation;
- Permanent errors caused by malfunction of the L0 electronics that result in loss of event synchronisation. This category includes errors which require the assertion of the TFC FE reset signal in order to re-synchronise the front-ends.

3.2 Deserialisation

The serial outputs from the optical receivers are directly AC-coupled to the RocketIO transceiver serial IO ports of the ingress FPGAs. The RocketIO transceivers are configured to present the received data to the FPGA fabric as a deserialised 16-bit wide parallel stream resynchronised to the local 80.44MHz clock.
The RocketIO transceiver configuration and operation is through a set of Xilinx library macros. A number of link IO standards are supported through specific macros and a fully customisable macro is also provided. The serial encoding used by the front-end GOLs is compatible with the GT_ETHERNET_2 macro. However, one additional feature of the RocketIO transceiver that is not configurable using the GT_ETHERNET_2 macro allows the deserialised data to be 16-bit word aligned. As this results in some additional simplification of the downstream logic, the fully customisable GT_CUSTOM macro is instead used. It is configured with the same default settings as the GT_ETHERNET_2 macro but has the 16-bit alignment feature enabled.
The RocketIO transceivers incorporate clock correction logic that synchronises the received data with the system clock. IDLE characters are inserted or deleted from the received stream as required to maintain synchronisation. The clock correction logic is configured to insert extra IDLE characters only where IDLE characters already exist in the data stream so that the blocks of received data are not fragmented. The system clock is higher frequency than the data transmission clock so that clock correction never results in deletion of IDLE characters and the RocketIO elastic buffer cannot overflow.
When it is necessary to reset the receive logic of the RocketIO transceivers, the scheme outlined in Xilinx Application Note 670 is used. Reset is triggered whenever the transceiver detects loss of synchronisation. The reset sequence requires some tens of clock cycles to complete in order that the receiver elastic buffer is flushed and so that the read and write pointers are correctly positioned.

3.3 L0 data ingress

Although the input data cannot be assumed to be error-free, the time of arrival of correct data can be predicted by emulating the behaviour of the L0 electronics using the L0 trigger information from the TFC system. This deterministic behaviour is exploited in the ingress logic to provide immunity from errors induced by the input data.
The ingress logic generates data frames of the correct length using the L0 emulation logic to determine the frame timing. This logic is independent of the data arriving on the input links and therefore cannot be disrupted by the incoming data. The content of the data frame is extracted from a “co-synchronisation” buffer by reading a fixed amount of data from the beginning of the buffer for each new generated frame. As data arrive from the front-end, they are written to the co-synchronisation buffer. Each new data packet is written starting from the beginning of the buffer. Figure 4 illustrates a possible pattern of frames as predicted by the L0 emulator and a corresponding pattern of ingress frames. The figure also indicates the buffer write address for ingress frames and read address for emulated frames assuming that the pixel chips are operating in LHCb mode.

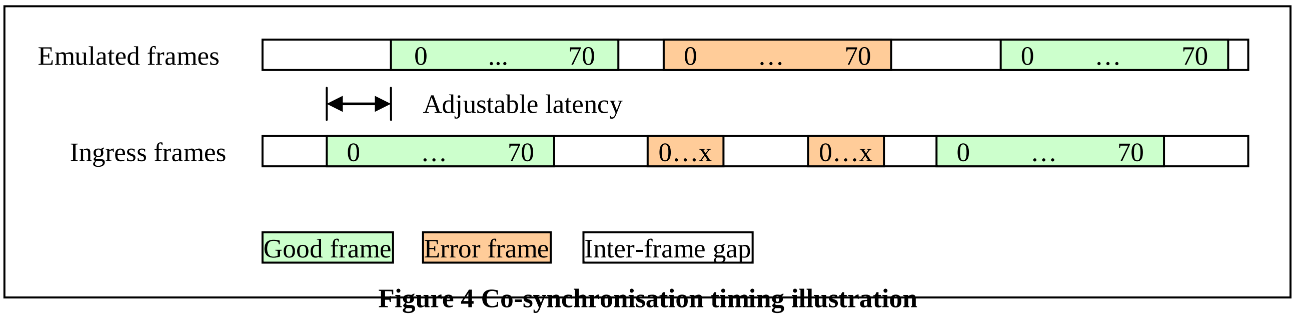


Figure 4 Co-synchronisation timing illustration

In the absence of errors, and by tuning the latency of the L0 emulation so that the data reading follows the data writing by a few clock cycles, the generated data frames will then normally contain the correct data from the front end. Precise tuning of the latency is not required. The only requirement is that the start of the read cycle follows the start of the write cycle but not by more than 36 clocks (1 LHCb event). By setting the latency to somewhere in the middle of the allowed range, correct operation is maintained over a wide range of input latency.
Errors present in the input data will result in corruption of the data content of the frames emitted by the co-synchroniser but cannot result in frame de-synchronisation. Neither data underrun (short frames) nor overrun (long frames) can permanently corrupt the buffer because the write pointer is always reset at the start of each frame and the buffer wraps around on overrun. Therefore, once the errors are no longer present and the L0 is once again sending data frames at the correct time, the data frame content will again be correct without the need for any special reset or resynchronisation logic. Corrupt data will typically contain invalid event ID, bunch ID or data parity information and these are detected in downstream logic. In the example of Figure 4, the first ingress frame is correctly transmitted and therefore appears correctly at the output of the co-synchroniser buffer. The second frame is incorrect so the data extracted from the co-synchroniser buffer will not be valid although the output frame length and timing are correct. The third ingress frame is again correct so the data are correctly extracted from the co-synchroniser buffer. Figure 4 illustrates the co-synchroniser write and read pointer positions for correctly timed data from the front end (left) and for an example of erroneous data (right). Note that, in this error example, the read pointer is ahead of the write pointer so the extracted data is invalid. Other misalignments of the write and read pointer are also possible; however, since the write pointer is reset for each new frame independently of the read pointer, the correct relationship will be automatically re-established for the next correctly timed frame.
In the case of permanent loss of synchronisation of the L0 electronics a TFC FE reset re-establishes the correct operation as this causes a reset of both the L0 electronics and the L0 emulator logic in the L1 board.
The co-synchroniser buffer is implemented using a single Virtex2Pro dual-port block RAM per input channel with 16-bit read and write ports.

3.4 Pixel masks

The data emerging from the cosynchroniser buffer are combined word-by-word with a stored pattern that is initialised through the configuration interface. This allows certain pre-defined bits in each data frame to be either forced on or forced off according to a configuration setting. This feature allows noisy pixels to be masked in the L1 electronics or may be used to generate arbitrary patterns of hits for testing.

3.5 Input pre-processing

At this stage the data frames are guaranteed to be the correct length and to have the correct timing although the content of the frames may contain errors. The correct operation of the subsequent zero-suppression stage does not depend on the frames having the correct internal structure so the error checking is postponed until later.
In order that the zero-suppression algorithm is applied only to frames for which the zero-suppressed data size would be less than the non-zero-suppressed size, a first pass is performed that calculates the size based on the content. A 32-bit L1 header word containing this information and also containing event ID and bunch ID information copied from the L0 header words is added at the beginning of each frame (the format is described in Section 6.1.2). The pre-processed data frames are buffered in dual-ported block RAM with 16-bit read and write ports. The use of the dual-port RAM allows the L1 header to be easily added at the beginning of the frame after the rest of the frame has already been buffered. The buffer must be large enough to contain at least one (ALICE) event as the first word read is the L1 header word and this is written only after all the other words of the frame have been written. There is no flow-control applied to this buffer (or the preceding co-synchronisation buffer) and events are presented to the zero-suppression algorithm as soon as each complete event is available.
The addition of the L1 header word removes the inter-frame gap but a start-of-frame signal synchronised to the extraction of the pre-processed frames from the buffer is generated in its place.

The L1 input pre-processor also computes the pixel data parity from the data frame content and replaces the L0 parity words with the XOR of the L1 and L0 computed parity. In the absence of parity errors, the XOR operation would yield 0 otherwise a bit is set in the L1 header to indicate that the parity words (and other header words) should not be suppressed in the downstream logic so that they are available for off-line checking later.

3.6 Zero-suppression

The zero-suppression algorithm is fully pipelined and operates in parallel on all input channels at the data ingress rate. For this reason, flow control is not required for the preceding buffers. A start-of-frame signal triggers the start of the zero-suppression algorithm. Whether zero-suppression is applied to the current frame is determined from the information in the L1 header word (the first word of the frame). At this stage, the data are 16-bit wide with the first 6 16-bit words containing the L1 and L0 header followed by 64(512) 16-bit words of LHCb(ALICE) pixel data and ending with 2 16-bit words containing the pixel data column-wise parity. The zero-suppression algorithm is not applied to the 6 header words or to the 2 parity words. An intermediate dual buffer is used so that the two bytes of each 16-bit data word can be processed in parallel and so avoids the need to multiply the clock speed in order to satisfy the latency requirement.

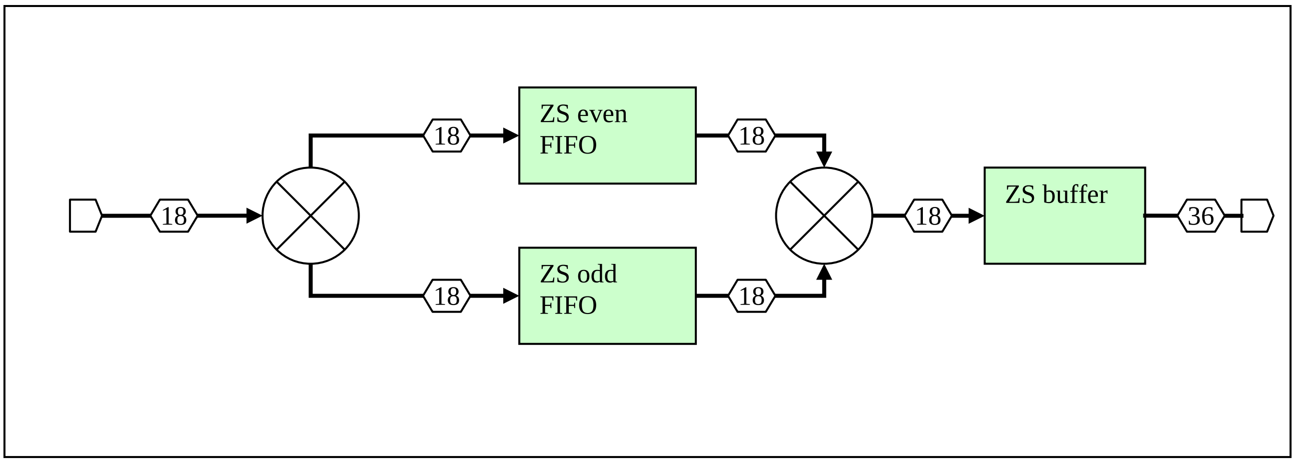


Figure 5 Zero-suppression buffer structure

An additional feature of the zero suppression logic is that data are suppressed from ingress channels for which the link status is not OK. This action is flagged using a bit reserved in the L1 header.
The zero-suppressed frame is buffered in dual-port block RAM having an 18-bit write port and 36-bit read port. As soon as a complete frame has been buffered, the data are extracted from this buffer and written to the main ingress data buffer without flow control. The Virtex2Pro block RAM parity bits are used to mark significant data frame features. In particular one of the bits is used to mark the end of each frame.
More details about the zero-suppression algorithm and the zero-suppressed data format can be found in Sections 6.2 and 6.1.4.

3.7 Ingress multiplexing

The main ingress data buffer is implemented using block RAM configured with 36-bit write and read ports. Each ingress channel is served by an independent buffer each of which is large enough to contain 56 non-zero-suppressed frames. Each ingress FPGA serves 9 ingress channels. The ingress multiplexer extracts frames from the corresponding 9 buffers in round-robin fashion and concatenates them into an output buffer with the addition of an ingress header word at the beginning of each block and a trailer word at the end of each block. The concatenation cycle is controlled by commands derived from TFC signals in the egress FPGA and queued in each ingress FPGA.

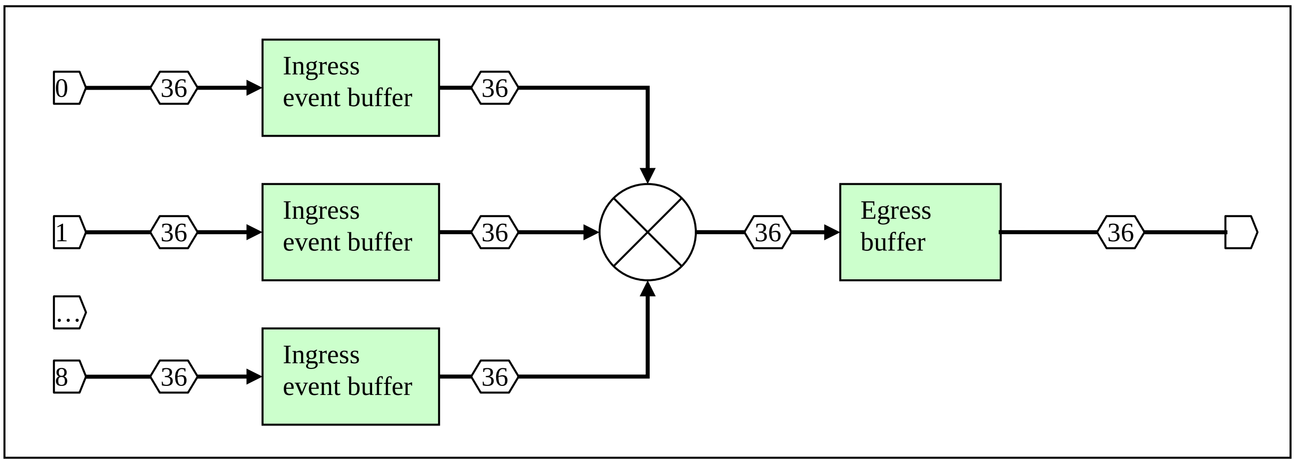


Figure 6 Ingress 9:1 multiplexing buffer structure

Each concatenation command contains the lowest 8 bits of the predicted event ID and bunch ID. These are compared with the corresponding bits of the L1 header word and any discrepancy results in a bit being set in the L1 header word that inhibits the subsequent suppression of headers and trailer data which would normally be applied when no errors are detected.
In case any ingress event buffer is nearly full, a throttle signal is asserted. The OR of the 9 throttle signals sources a dedicated throttle signal to the egress FPGA which results in the assertion of the TFC throttle for the module.
Only ingress channels that are configured to be enabled are included in the concatenated block and the mask of configured channels is contained in the block header word.

3.8 Transmission to the egress FPGA

A simple configuration of the ingress-to-egress data links is to use two of the three available RocketIO transceivers in parallel. The link then effectively operates as a 32-bit wide parallel bus at 80MHz (about 2.5 Gbit/s effective aggregate). This matches the egress architecture where it is natural to use a 32-bit wide data path due to the GBE interface.
An event-wise flow-through protocol is used with flow-control across the transmission link.

4 Egress firmware

The following sections outline the implementation of the firmware in the egress FPGA. This FPGA receives the multiplexed event data from the ingress FPGA, builds multi-event packets and performs Ethernet formatting and fragmentation before transmission to the DAQ network using the GBE interface.

4.1 TFC interface

The dataflow is driven by the broadcast signals received through the TFC interface (trigger, FE reset, trigger type, MEP destination broadcast and throttle signals).

4.1.1 L0 emulation

The trigger and FE reset broadcasts are used to drive L0 emulation logic which predicts the event ID, BX ID and timing of each data frame transmitted by the front-end electronics. This derived information is broadcast to the ingress FPGAs where it used to control the extraction of the data from the co-synchronisation buffer (Section 3.2) and also to validate the ingress data (which may be subject to corruption due to SEU) during multiplexing and formatting (Section 3.7).

4.1.2 MEP generation

MEP generation is controlled by the TFC readout type and destination broadcasts. A new event is extracted from the input FIFOs and appended to the MEP buffer when a readout type broadcast is received. Both the data and the multiplexing commands are queued in FIFOs so that the precise timing and ordering of their arrival does not affect the operation of the multiplexer; the new event is added to the MEP buffer as soon as the readout type information and ingress data are both available. The MEP is closed when the destination broadcast is received. This command is also queued in the command FIFO and the sequence of readout type broadcasts and destination broadcasts determines the number of events per MEP.
It is possible to run in a mode where the TFC MEP destination broadcasts are not used. In this mode, the L1 logic generates the necessary MEP generation commands using the TFC trigger together with the MEP packing factor and destination address information stored in the static configuration registers.

4.1.3 Throttle

The TFC throttle is formed by the logical OR of the 4 dedicated ingress throttle signals with a throttle asserted by the L0 emulator logic.

4.1.4 Sequence error detection

The RICH L0 electronics does not allow operation with triggers in consecutive bunch crossings as this causes de-synchronisation of the internal buffers. In this case, any further events from the L0 electronics must be considered invalid until the next FE reset broadcast. The readout supervisor enforces a gap between triggers to avoid this situation. The sequence of FE triggers is checked in the L0 emulator logic of the L1 modules in order to verify that this condition is not violated. In case consecutive triggers are seen, an error flag is set and is subsequently cleared on receiving a FE reset broadcast.
[Flag condition in ingress header word? Assert throttle?]

4.2 Ingress to egress data transmission

Data arrive from the four ingress FPGAs in event-wise packets and are immediately buffered in small event FIFOs. The data from each ingress FPGA arrive in parallel on two serial links (see Section 3.8) and are written in parallel to two event FIFOs having 16-bit read and write ports. By keeping the data from the two serial streams separate at this stage, any misalignment of the two streams due to clock skew is automatically removed when the buffers are read. The buffers also allow flow control to be easily implemented across the ingress-to-egress links; an internal pause signal is generated and transmitted to the ingress FPGA when the FIFO becomes nearly full.

4.3 Egress multiplexing and MEP building

The generation of multi-event packets is controlled by TFC broadcasts (Section 4.1.2). The readout type broadcast is used to signal the extraction of the next event packet from the input buffers. The data are extracted from the 4 input sources in round-robin fashion and concatenated into the MEP buffer. The MEP building buffer is implemented using block RAM resources configured as dual-port RAM. The event header information can therefore be added to the beginning of the event block after the complete event has been buffered. Further events are appended in the same way until the MEP destination broadcast is processed. At this point the MEP header information is added to the beginning of the buffer and the MEP is ready for Ethernet formatting. An additional word at the beginning of the MEP contains the destination information. This is used in the downstream Ethernet formatting stages but does not form part of the transmitted MEP.

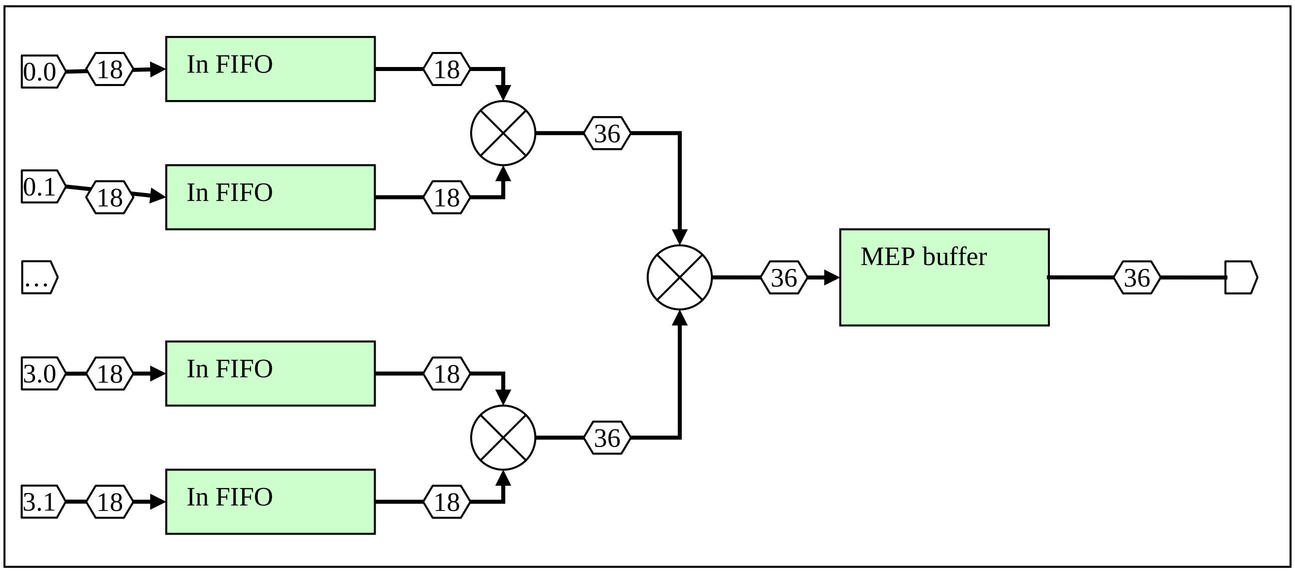


Figure 7 Ingress and MEP buffer structure

The pathological maximum size of the concatenated data for one event for one 36-input RICH L1 board is about 40kbit. Therefore, for some MEP packing factors it is possible that the size could exceed the maximum size imposed by the constraint that the MEP must fit in a single IP message. In this case the ingress packets are truncated.
If, while writing the MEP buffer, the event could exceed the maximum MEP size before the destination broadcast is received, all further event packets extracted from the ingress buffer have their data part truncated and a status bit is set in the corresponding ingress header word. Enough headroom is reserved in the MEP buffer to accommodate any remaining headers that may need to be written taking proper account of the maximum allowed MEP packing factor.

4.4 Ethernet formatting

The raw data transport protocol is specified in the document EDMS 499933 and is based on the IP transport protocol. IP frames are formed by the concatenation of the Ethernet header, IP header, MEP header and MEP data.
Some parts of the header information are derived from information obtained via the ECS interface. This information remains static for the duration of a run. To this is added the part of the destination address sent via the TFC system that identifies the final destination of the MEP.

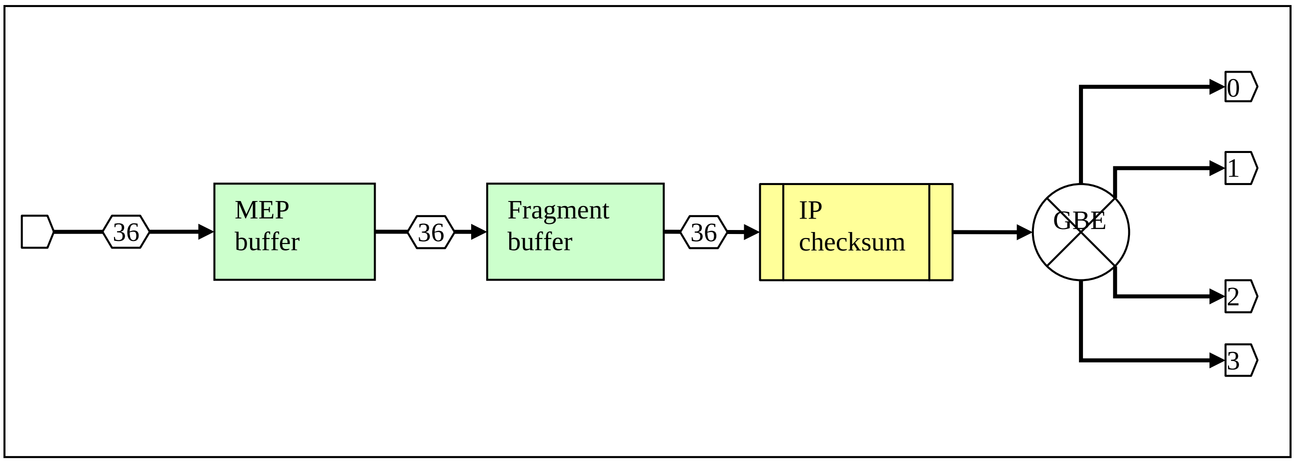


Figure 8 Egress buffer structure

4.5 Ethernet fragmentation

The IP size frequently exceeds the Ethernet MTU so packets are divided into Ethernet fragments before transmission to the readout network.

4.6 Data transmission

The LHCb GBE interface services four bidirectional Gbit Ethernet links through a common 32-bit data bus operating at up to 133MHz. Each output link has sufficient buffering for a maximal size MTU but not enough for a complete maximal MEP. Therefore, in order to make optimal use of the output bandwidth, the Ethernet fragments comprising the MEP are distributed in round-robin fashion across the four available links. Any of the four links may be masked using static configuration information.

4.7 Flow-control

The L1 electronics will assert a throttle signal in the event that it cannot accept further events without risking buffer overflow conditions. This signal will be routed via a dedicated network to the readout supervisor, which will suppress further triggers and allow the L1 buffers to empty. When sufficient buffer space is again available, the throttle signal is deasserted and the readout supervisor will resume sending triggers. The throttle must be asserted sufficiently early to allow any events already in the system to be buffered.

4.8 Reset strategy

Several reset types are available:

- A global reinitialisation causing a complete reconfiguration of the L1 modules similar to a power up reset. Generated using a gluecard GPIO via ECS or a front-panel pushbutton.
- An ECS reset (initiated via the crate reset function or pushbutton) that resets the CCPC and gluecard but does not reset any L1 logic and therefore does not interrupt data flow.
- An L1 reset generated using a gluecard GPIO via ECS that does a general reset of the L1 logic.
- A GBE reset generated using a gluecard GPIO via ECS that resets only the GBE card.
- A front-end (L0) reset in which is also used in the L1 logic to reset the L0 emulator functions.

5 Special run modes

The RICH L1 electronics is not required to do any special processing for special triggers during normal LHCb data acquisition. All trigger types are treated the same.

5.1 ALICE mode

Outside of normal LHCb data taking it may be necessary to read out the RICH detector in ALICE mode. The L1 electronics will support this operation mode but only under restricted trigger conditions. Typically this would mean setting the MEP packing factor to 1 and ensuring that the readout supervisor L0 gap generator is used to enforce a gap of several clocks between L0 triggers. In ALICE mode, the L1 module generates the throttle in such a way as to ensure that an ALICE event must have propagated completely through the logic before a subsequent trigger is allowed (by deasserting the throttle). The gap generator prevents further triggers in the time taken for the throttle signal to propagate to the readout supervisor.

5.2 Running without TFC broadcasts

It is possible to run without using the TFC readout type and destination address broadcasts that are normally used to control the building of MEPs. To enable this mode, assert bits 0 and 1 of register 15. The number of events per MEP is then determined by register 18. Each MEP is queued for transmission as soon as the requested number of events has been concatenated. Note that in this mode, FE resets should not be sent while triggers are flowing as this can cause loss of synchronisation. Readout type and MEP destination broadcasts are discarded.

5.3 Fast resettable channel inhibit

A channel inhibit is provided that can be used to selectively disable individual channels while allowing the HPDs to be quickly re-enabled using a global reset signal. This is intended to be used during normal LHCb running so that an input channel can be disabled in the event that the downstream monitoring algorithms detect that the connected front-end has been upset. All channels that have been disabled during a run can then be quickly re-enabled at the start of a new run. The procedure to disable a channel is as follows:

- Assert bit 12 of the appropriate front-end channel configuration register.
- Deassert bit 12 of the front-end channel configuration register

This causes an internal channel inhibit to be latched. The status of the internal inhibit can be monitored in bit 8 of the corresponding front-end channel status register 0.

To globally deassert the internal channel inhibit for all channels:

- Assert bit 5 of configuration register 0x15.
- Deassert bit 5 of configuration register 0x15.

6 Data format

The formatted data from the RICH UKL1 boards are encapsulated inside the LHCb MEP structure which itself forms the payload of an IP packet. The Ethernet, IP headers, MEP headers and event headers are described elsewhere [EDMS 499933]. The format of the RICH data is described in the following sections.

6.1 The raw bank

31..0
Ingress[0] header
HPD[0]
…
HPD[n]
Ingress[1] header
HPD[0]
…
HPD[n]
Ingress[2] header
HPD[0]
…
HPD[n]
Ingress[3] header
HPD[0]
…
HPD[n]

Each event can contain a number of subdetector raw banks [EDMS 565851]. Presently only one type of RICH raw data bank is defined. The standard bank header contains 8 header bytes and is followed by the concatenated data from the front-ends. The data structure, excluding the standard header, is illustrated in the diagram. It reflects the UKL1 hardware architecture having up to 4 groups of up to 9 HPDs for each UKL1 board.

6.1.1 L1 ingress header format

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
`R`	`T`	`ID`		`Channel active [11..0]`												`BX ID[7..0]`								`Evt ID[7..0]`							

Key:

R=Reserved T=Truncation (0: not truncated, 1: truncated)

ID=Ingress ID

The 9 LSB of the channel active flags indicate which of the 9 input channels are configured to be in the readout and therefore how many blocks of HPD data are to follow. Note that the field is large enough for 12 channels. This is to allow the same format to be applicable to data from the 12-channel prototypes.
The T flag indicates that the data from this ingress FPGA (including HPD headers) have been suppressed. The channel active flags indicate which HPDs would have been in the data had they not been suppressed. This truncation may be applied to ensure that the MEP size does not exceed the maximum size of one IP packet (64 kbyte).

6.1.2 HPD data block formats

Each HPD data block has the general structure shown below.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
R	G	F	M	Z	NZ count[10..0]											Evt ID[4..0]					HPD ID[10..0]										
L0[0]																															
L0[1]																															
Pixel data[0]																															
…																															
Pixel data[n]																															
Column parity																															

Key:

R=Reserved, internal empty flag	M=ALICE/LHCB mode (0: LHCb, 1: ALICE)
Z=Zero suppression (0: Non-ZS, 1: ZS)	F=Format (0: Compact, 1: Extended)
G=GT inhibit (0: not inhibited, 1: inhibited)	

Two variants are defined; one having extended header information (F=1) and another, more compact format (F=0). In the compact header format, the two L0 header words, L0[1,0], and the column parity word are suppressed. The compact format is generated except when an error has been detected.
The column parity word is the XOR of the L0 column parity and the L1 calculated column parity. All bits are therefore zero in the absence of parity errors.
The pixel data are either in zero-suppressed format (Z=1) or non-zero-suppressed format (Z=0). The UKL1 board only performs zero-suppression on the front-end data if this would reduce the size of the block of data from the front-end. Whether zero-suppression is performed is determined for each individual HPD for each event so the raw data will in general contain a mixture of zero-suppressed and non-zero-suppressed data blocks.
When the GT inhibit flag is set in the data, F is not set and Z is set. In this case, only the L1 header word is present and has all other bits undefined.
"NZ count" is the number of bytes with at least one hit. It serves as the block length word for zero-suppressed blocks. It is foreseen that blocks with "NZ count" equal to zero may be suppressed completely although this is not currently implemented.

6.1.3 Non-zero-suppressed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
R	G	F	M	Z	NZ count[10..0]											Evt ID[4..0]					HPD ID[10..0]										
L0[0]																															
L0[1]																															
Row 0																															
Row 1																															
Row 2																															
Row 3																															
…																															
Row 30																															
Row 31																															
Column parity																															

The format of the data arriving from the front-end electronics is defined in the PINT specification document [EDMS 696530]. The L1 format of the non-zero-suppressed data is the same as this with the exception that the L1 header word is added at the beginning and the L0 column parity word is replaced with the value XORed with the L1 calculated column parity.
Non-zero-suppressed blocks have Z=0. The table illustrates LHCb mode. In ALICE mode there are 256 rows instead of 32.

6.1.4 Zero-suppressed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
R	G	F	M	Z	NZ count[10..0]											Evt ID[4..0]					HPD ID[10..0]										
L0[0]																															
L0[1]																															
Address 1								Byte 1								Address 0								Byte 0							
Address 3								Byte 3								Address 2								Byte 2							
Address 5								Byte 5								Address 4								Byte 4							
Address 7								Byte 7								Address 6								Byte 6							
…								…								…								…							
Address n-1								Byte n-1								Address n-2								Byte n-2							
Undefined																Address n								Byte n							
Column parity																															

The zero-suppressed blocks have Z=1. If “NZ count” is odd, the block is padded to preserve 32-bit alignment before the “Column parity” word (as is the case in the example). The content of the padding word should not be assumed to be zero. The number of 32-bit words of zero-suppressed data between L0[1] and “Column parity” is given by (“NZ count” + 1)/2 of which the first “NZ count” 16-bit words are valid..
The zero-suppressed pixel data encoding is described in the following sections.

6.2 Zero-suppression scheme

The expected average signal occupancy is 1% and is not expected to locally exceed 10%. Zero-suppression algorithms typically exhibit a break-even point at which the zero-suppressed data size exceeds the non-zero suppressed data size. For a simple algorithm this occurs at occupancies of about 5% as illustrated in Table 8. These numbers are for illustrative purposes; care should be taken when using them as they assume a particular algorithm. Because the above-threshold signals are not uniformly distributed improved performance can be achieved using a suitably optimised algorithm. The zero-suppression performance is also affected by different assumptions about the addressing method, packing of bits into words and header content.

	Zero-suppressed				**Non Zero-suppressed**
Average occupancy	1%	3%	5%	8%	100%
Event Size.	10.5Kbytes.	31.4Kbytes.	52.3Kbytes.	83.7Kbytes.	55Kbytes
L1 output bandwidth	3.36Gbits/s	10.05Gbits/s	16.74Gbits/s	26.78Gbits/s	18Gbits/s

Table 8 Zero-suppression performance

Zero-suppression is required in the 1MHz readout scheme to reduce the bandwidth requirements for the following stages of data transport.
The following zero-suppression strategy is adopted:

- For each event accepted by the L1 electronics, a zero-suppression algorithm will be applied to each block of data from the front end.
- Zero-suppression will only be applied to blocks of data whose non-zero-suppressed size exceeds the zero-suppressed size.
- A configuration switch allows zero-suppression to be disabled.

This strategy has the following benefits:

- It guarantees that the maximum event size can never exceed the non-zero-suppressed event size.
- It handles fluctuations in occupancy in the detector in an optimum and dynamic way.
- It automatically suppresses the 25% of pixels that are outside of the photocathode image and never generate hits.

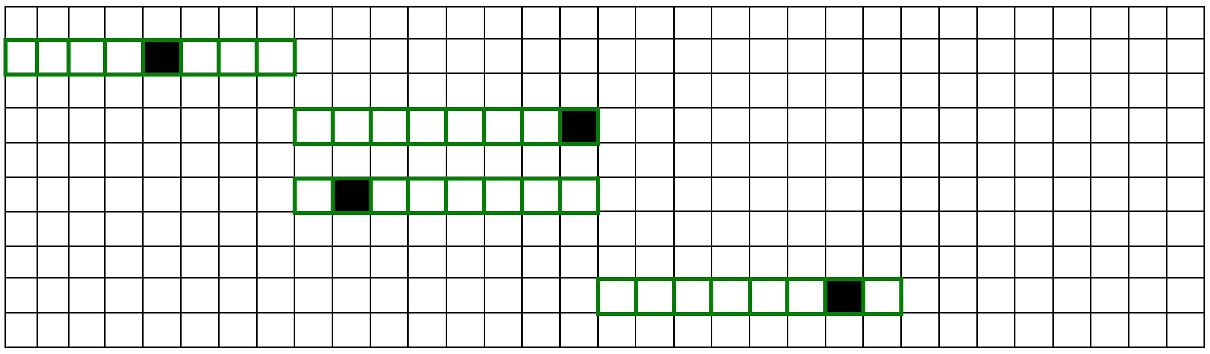
6.2.1 Zero-suppressed pixel HPD data

The zero-suppressed data will be encoded as a list of those bytes having at least one bit set. For LHCb-mode data, 7 address bits are sufficient to uniquely identify a byte within an HPD. Therefore, each non-zero byte can be encoded in a 16 bit word containing the 7 bits of address and 8 bits of data. Note also that the current scheme has all the even-addressed bytes followed by all the odd-addressed bytes. In ALICE mode the address field is not large enough to uniquely identify the location of a non-zero byte. To overcome this, the ALICE pixel data are divided into sub-blocks and the first byte of each sub-block is forced to be written with its sub-address whether or not the byte contains hits. By counting the occurrences of these sub-block boundary markers, the decoding routine can keep track of the current location within an ALICE block.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
`Address 1`								`Byte 1`								`Address 0`								`Byte 0`							
`Address 3`								`Byte 3`								`Address 2`								`Byte 2`							
`Address 5`								`Byte 5`								`Address 4`								`Byte 4`							
`Address 7`								`Byte 7`								`Address 6`								`Byte 6`							
…								…								…								…							
`Address n-1`								`Byte n-1`								`Address n-2`								`Byte n-2`							
`Undefined`																`Address n`								`Byte n`							

Table 9 LHCb mode zero-suppressed data format

The following array represents the pattern of hits from an HPD in an event. The full array is 32 by 32 pixels but only the first 10 rows are shown and the other rows contain no hit pixels in this hypothetical event. Rows are horizontal, columns are vertical (row,column)=(0,0) is at top right of the array. The groups of bits highlighted are those that would appear in the zero-suppressed format.



Assuming that the zero-suppression algorithm is applied, the following data block shows the resulting encoding (note all even addresses followed by all odd addresses):

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0x16								0x40								0x0e								0x01							
0x21								0x02								0x07								0x08							

6.3 Additional L1 monitoring data

Various counters and flags can be concatenated with the data in a format yet to be defined. The LHCb data format allows for this by defining a new raw bank type. This extra data would then be transparent to the readout network and would only need to be sent infrequently.

7 Configuration registers

Register address	Description	Notes
0x00(0)	Ethernet destination address [15..0]	
0x01(1)	Ethernet destination address [31..16]	
0x02(2)	Ethernet destination address [47..32]	
0x03(3)	Ethernet source address [15..0]	
0x04(4)	Ethernet source address [31..16]	
0x05(5)	Ethernet source address [47..32]	
0x06(6)	IP destination address [15..0]	
0x07(7)	IP destination address [31..16]	
0x08(8)	IP source address [15..0]	
0x09(9)	IP source address [31..16]	
0x0a(10)	Protocol type	
0x0b(11)	Type of service	
0x0c(12)	Time-to-live	
0x0d(13)	Type	
0x0e(14)	Version	
0x0f(15)	Miscellaneous control	Section 7.1
0x10(16)	Partition ID [15..0]	
0x11(17)	Partition ID [31..16]	
0x12(18)	MEP event count	
0x13(19)	Miscellaneous control register	Section 7.2
0x14(20)	Undefined	
0x15(21)	Resets	Section 7.3
0x16(22)	TFC latency compensation	Section 7.4
0x17(23)	L1 ID [7..0]	
0x18(24)	Ingress control	Section 7.5
0x19(25)	Undefined	
0x1a(26)	Undefined	
0x1b(27)	Undefined	
0x1c(28)	Undefined	
0x1d(29)	MTU	
0x1e(30)	MEP truncation HWM	
0x1f(31)	Ingress configuration mailbox	Section 7.6

7.1 Miscellaneous control register 1

0xf(15)		
Bit	**Description**	**Notes**
0	Force fixed MEP destination address	
1	MEP building mode	0: Use TFC broadcasts; 1: Auto-generate TFC
3..2	Enable throttle outputs	
5..4	Throttle output polarity	0: Normal; 1: Inverted
6	Enable TTC decoding	
7	Enable GBE TPA polling	
9..8	GBE port number	
10	Use fixed GBE port number	
15..11	Undefined	

7.2 Miscellaneous control register 2

0x13(19)		
Bit	**Description**	**Notes**
3..0	Undefined	
4	LHCb/ALICE mode (0=LHCb, 1=ALICE)	
15..5	Undefined	

7.3 Resets

0x15(21)		
Bit	**Description**	**Notes**
3..0	Reset FE status read pointers	One reset bit per Ingress FPGA
4	Reset SPI3 RX read pointer	
5	L1 (partial) asynchronous reset	Reset various dataflow logic blocks
6	Reset DCMs	Reset clock managers
7	TFC asynchronous reset	Reset TFC logic
8	Synchronous reset	Various uses

7.4 TFC latency compensation

0x16(22)		
Bit	**Description**	**Notes**
11..0	Latency	
15..12	Undefined	

This register controls the delay added to the TFC data that is broadcast to the ingress FPGAs. The following procedure to reset the latency pipeline should be followed after changing the latency value to avoid spurious data being broadcast:

1. Set the desired latency.
2. Assert bit 6 of register 15 to reset the TFC decoding logic.
3. Deassert bit 6 of register 15.

For small lab systems (similar TFC and data cable lengths) a value of around 0x20 is suitable. In LHCb the value must be increased to around 0x68 to compensate for the longer cable runs to the L0 electronics.

7.5 Ingress control

0x18(24)		
Bit	**Description**	**Notes**
3..0	Inhibit	**Error! Reference source not found.**
4	Ingress mailbox write pointer reset	Section 7.6
5	Ingress mailbox read pointer reset	Section 7.6
6	Ingress configuration buffer transmit trigger	Section 7.6
7	Broadcast flag	Section 7.6
[9..8]	Ingress configuration target ID	Section 7.6

1. Each bit inhibits the corresponding ingress FPGA.

7.6 Ingress configuration mailbox

0x1f(31)		

This 16-bit register is used to transfer configuration data (operation mode, pixel masks) to the input channels. Repeated writes to this register cause consecutive locations of an internal buffer to be initialised with the written data. The allowed formats of the data block are illustrated in the following subsections. Four types of block are defined and they are differentiated by the Type field of the header word. Each block targets one of the 36 input channels.
The target channel is identified by the channel ID (Channel[3.0]) in the header word (range 0-8). Channel ID=0xf is interpreted as a broadcast address and the settings will be applied to all channels of the targeted ingress FPGA. The target Ingress FPGA ID is set using bits [9..8] of register 24.
The recommended sequence of operations to configure one ingress channel is:

1. Write the ID of the target ingress FPGA to bits [9..8] of register 24.

2. Assert bit 4 of register 24 (reset the buffer write pointer).
3. Deassert bit 4 of register 24.
4. Write header word to register 31.
5. Repeated write to register 31 to set data.
6. Assert bit 5 of register 24 (reset the buffer read pointer).
7. Deassert bit 5 of register 24.
8. Assert bit 6 of register 24 (trigger transfer of data to ingress).
9. Deassert bit 6 of register 24.

The sequence assumes that bits 4, 5 and 6 of register 21 are all deasserted at the start of the sequence.

7.6.1 Type[2..0]=000, Set operation mode

The second word contains configuration flags for the addressed ingress channel.

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Channel[3..0]				R	R	0									
Empty threshold			R	R	R	R	Mm	Ee	R	Me	Ie	Si	Zi	Mi	Ii

Key:

Ii: Channel inhibit (1=inhibit)
Mi: Channel mode (0=LHCb, 1=ALICE)
Zi: Channel zero-suppression (0=disabled, 1=enabled)
Si: Channel empty event suppression (0=disabled, 1=enabled)
Mm: Mask mode (0=force off, 1=force on).
Ie: L0 emulator inhibit (1=inhibit)
Me: L0 emulator mode (0=LHCb, 1=ALICE)
Ee: L0 emulator enable (1=send events)

Empty threshold is the threshold in number of bytes with hits above which the “empty” HPDs are transported.

7.6.2 Type[2..0]=001, Set selected mask bits

The words following the header contain pixel mask data for the addressed ingress channel.

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Channel[3..0]				R	R	Type[9..0]									
0	Addr[0]											Mask[0]			
0	Addr[1]											Mask[1]			
0	…											…			
0	Addr[n]											Mask[n]			
1															

The MSB of each valid Address/Mask pair must be 0 and the block is terminated by a word having the MSB set.

7.6.3 Type[2..0]=010, Set all mask bits

This dataless command causes all mask bits of the target channel to be set.

7.6.4 Type[2..0]=011, Unset all mask bits

This dataless command causes all mask bits of the target channel to be unset.

7.6.5 Type[2..0]=100, Actions

The bits in the second word trigger various actions in the Ingress FPGA. Write 1 to the desired bit to trigger the action.

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Channel[3..0]				R	R	4									
R	R	R	R	R	R	R	R	R	R	R	R	Rs	Ss	Rh	Sh

Key:

Ih: Set channel inhibit
Ss: Set channel squash
Rh: Reset channel inhibit
Rs: Reset channel squash

8 Status registers

Register address	Description	
0x20	TTC ID [7..0]	Section 8.1
0x21	Inhibits	Section 8.2
0x22	Buffer flags	Section 8.3
0x23	MEP builder status	Section 8.4
0x24	Ethernet formatter state	
0x25	Miscellaneous status	Section 8.5
0x26	Last received TTC long broadcast	
0x27	Egress input FIFO errors	Section 8.6
0x28	FE0 status	
0x29	FE1 status	
0x2a	FE2 status	
0x2b	FE3 status	
0x2c	ROT broadcast counter & L0 trigger counter	Section 8.7
0x2d	MEP counters	Section 8.8
0x2e	MEP ingress status	Section 8.9
0x2f	Alarms	Section 8.10
0x30	Egress input FIFO 0,1 occupancy	Section 8.11
0x31	Egress input FIFO 2,3 occupancy	Section 8.12
0x32	Egress input FIFO 0,1 RX count	Section 8.13
0x33	Egress input FIFO 2,3 RX count	Section 8.14
0x34	Egress input FIFO 0,1 TX count	Section 8.15
0x35	Egress input FIFO 2,3 TX count	Section 8.16
0x36	GT 0,1,2,3 length	Section 8.17
0x37	Ethernet fragment counters	Section 8.18
0x38	SPI3 flag counts	Section 8.19
0x39	State machine states	Section 8.20
0x3a	TFC ROT counter [15..0]	
0x3b	TFC ROT counter [31..16]	
0x3c	TFC MEP counter [15..0]	
0x3d	TFC MEP counter [31..16]	

8.1 General status 0

0x20 (32)		
Bit	**Description**	**Notes**
7..0	TTC ID	Sensed by L1 on TTCrx reset
8	TFC FIFO request	1=request pending
9	TTCrx ready	
11..10	Undefined	
15..12	Event merger read FSM state	

8.2 Inhibits

0x21 (33)		
Bit	**Description**	**Notes**
3..0	GT 0 (Data low word) inhibit status	1
7..4	GT 1 (data high word) inhibit status	1
11..8	(GT 0) OR (GT 1) OR CFG inhibit status	1
15..12	GT 2 inhibit status	1

1. Each bit encodes the status of one of the 4 ingress FPGAs.

8.3 Flow control status

0x22 (34)		
Bit	**Description**	**Notes**
0	SPI3 (GBE) TX busy	
1	MEP FIFO full	
3..2	Throttle OR	
7..4	GT FIFO full	
11..8	GT 1 FIFO data ready	
15..12	GT 0 FIFO data ready	

8.4 MEP builder status

0x23 (35)		
Bit	**Description**	**Notes**
0	TFC L0 reset	
1	TFC L1 reset	
7..2	Last received short broadcast	
11..8	GBE transmit port available	1 bit per GBE port
12	Ethernet fragment ready	
13	MEP ready	
14	Fragment request	1=request pending
15	Event Qvalid	

8.5 Miscellaneous status

0x25 (37)		
Bit	**Description**	**Notes**
0	Ethernet Qvalid	
1	Sequence error	Cleared on L0 reset
3..2	MEP count	
7..4	GT FIFO TX wait flags	1 bit per ingress
15..8	TFC FIFO occupancy	

8.6 Egress input FIFO errors

0x27 (39)		
Bit	**Description**	**Notes**
1..0	Input FIFO RX sequence error flags (Ingress 0)	1
3..2	Input FIFO RX sequence error flags (Ingress 1)	1
5..4	Input FIFO RX sequence error flags (Ingress 2)	1
7..6	Input FIFO RX sequence error flags (Ingress 3)	1
9..8	Input FIFO TX sequence error flags (Ingress 0)	1
11..10	Input FIFO TX sequence error flags (Ingress 1)	1
13..12	Input FIFO TX sequence error flags (Ingress 2)	1
15..14	Input FIFO TX sequence error flags (Ingress 3)	1

1. One bit for each FIFO the pair.

8.7 TFC L0 and ROT counters

0x2c (44)		
Bit	**Description**	**Notes**
7..0	L0 trigger counter	
15..8	Readout type broadcast counter	

8.8 MEP counters

0x2d (45)		
Bit	**Description**	**Notes**
7..0	TFC destination assignment counter	
11..8	MEP sequence counter	
15..12	Undefined	

8.9 MEP ingress status

0x2d (45)		
Bit	**Description**	**Notes**
7..0	Polling sequence	
11..8	GTIN TX waiting for EOF	One bit per ingress
15..12	Undefined	

8.10 Alarms

0x2f (47)		
Bit	**Description**	**Notes**
0	TTCrx ready status	
1	TFC sequence error	
2	Undefined	
3	Undefined	
15..4	Undefined	

8.11 Egress input FIFO 0,1 occupancy

0x30 (48)		
Bit	**Description**	**Notes**
3..0	Occupancy, ingress 0.0	
7..4	Occupancy, ingress 0.1	
11..8	Occupancy, ingress 1.0	
15..12	Occupancy, ingress 1.1	

8.12 Egress input FIFO 2,3 occupancy

0x31 (49)		
Bit	**Description**	**Notes**
3..0	Occupancy, ingress 2.0	
7..4	Occupancy, ingress 2.1	
11..8	Occupancy, ingress 3.0	
15..12	Occupancy, ingress 3.1	

8.13 Egress input FIFO 0,1 RX count

0x32 (50)		
Bit	**Description**	**Notes**
3..0	RX counter, ingress 0.0	
7..4	RX counter, ingress 0.1	
11..8	RX counter, ingress 1.0	
15..12	RX counter, ingress 1.1	

8.14 Egress input FIFO 2,3 RX count

0x33 (51)		
Bit	**Description**	**Notes**
3..0	RX counter, ingress 2.0	
7..4	RX counter, ingress 2.1	
11..8	RX counter, ingress 3.0	
15..12	RX counter, ingress 3.1	

8.15 Egress input FIFO 0,1 TX count

0x34 (52)		
Bit	**Description**	**Notes**
3..0	TX counter, ingress 0.0	
7..4	TX counter, ingress 0.1	
11..8	TX counter, ingress 1.0	
15..12	TX counter, ingress 1.1	

8.16 Egress input FIFO 2,3 TX count

0x35 (53)		
Bit	**Description**	**Notes**
3..0	TX counter, ingress 2.0	
7..4	TX counter, ingress 2.1	
11..8	TX counter, ingress 3.0	
15..12	TX counter, ingress 3.1	

8.17 GT frame lengths

0x36 (54)		
Bit	**Description**	**Notes**
3..0	Ingress 0	
7..4	Ingress 1	
11..8	Ingress 2	
15..12	Ingress 3	

8.18 Ethernet fragment counters

0x37 (55)		
Bit	**Description**	**Notes**
7..0	Fragments from formatter	
15..8	Fragments to GBE	

8.19 SPI3 flag counters

0x38 (56)		
Bit	**Description**	**Notes**
7..0	TEOP	
15..8	TSX	

8.20 State machine states

0x39 (57)		
Bit	**Description**	**Notes**
2..0	Ethsa state	
12..8	Evt_merger state	

8.21 FE per-channel registers

Register address	Bit fields		Notes
0x00	`General status [15..0]`		Section 8.21.1
0x01	`Rx Disparity errors [7..0]`	`Cosync errors [7..0]`	
0x02	`Rx buffer errors[7..0]`	`Rx clock correction [7..0]`	
0x03	`Configuration flags [15..0]`		
0x04	`ZS FIFO count in[7..0]`	`ZS FIFO count out [7..0]`	
0x05	`Ingress_occupancy [7..0]`	`ZS occupancy [7..0]`	
0x06	`ZS “A” counter [7..0]`	`ZS “B” counter [7..0]`	
0x07	`TFC ToA [7..0]`	`L0 data ToA [7..0]`	

8.21.1 FE channel register 0x0, (0)

Bit	Description	Notes
1..0	Rx loss of sync	
3..2	Rx buffer status	
4	GT inhibit	
5	Cosynchroniser error	
6	Ingress buffer ready	
7	Ingress buffer full	
8	Fast re-settable channel inhibit status	
9	Fast re-settable channel squash status	
11..10	Unused	All bits 0.
15..12	Unused	All bits 1.

8.22 FE general status registers

Register address	Description		Notes
0x00	General status		Section 8.22.1
0x01	`cnt_d[7..0]`	`cnt_q[7..0]`	
0x02	`tfc_cnt_d[7..0]`	`tfc_cnt_q[7..0]`	
0x03	`cnt[7..0]`	`tfc_cnt[7..0]`	

8.22.1 FE general register 0x0, (0)

Bit	Description	Notes
0	Reset	
1	ready_b	
2	Up DCM locked	
3	Down DCM locked	
4	DCMS not ready	
5	GT0 inhibit	
6	GT1 inhibit	
7	GT2 inhibit	
8	TFC FIFO data ready	
9	TFC FIFO full	
10	Egress busy	
11	MRX signal detect	

9 The 6U 12-channel module

The 12-channel, 6U prototype module for the production boards described in the document operates in a very similar way and is also largely compatible with the LHCb readout infrastructure. The main differences are:

1. The configuration interface does not use a Credit Card PC but uses instead a USB interface implemented using an on-board C8051 microcontroller.
2. The connection to the readout network is via a single 100Mbit Ethernet link instead of the four 1000Mbit Ethernet links provided by the GBE card.

Architecturally the prototype is the same as the production modules except that the functions of the ingress and egress FPGAs are merged into a single Virtex2Pro FPGA.
The prototype board can therefore be used in nearly all situations where the production boards are employed with the principal operational differences being due to the different mechanical form-factor, the reduced readout bandwidth to the readout network and the need for a different configuration interface.

9.1 The configuration interface

The external configuration link is USB while internally the module implements the configuration and status space as 8-bit SMB (I2C) registers. The layout of the configuration space is the same as for the production boards except that two 8-bit SMB accesses are required per 16-bit register. A Java GUI application exists and provides a convenient way of configuring the boards and displaying their status. A mechanism is provided to configure the TTCrx chip by forwarding SMB accesses from the SMB slave implementation in the L1 FPGA to the separate SMB bus to which the TTCrx is connected. Writing the TTCrx registers via this mechanism is transparent but reading is not and requires an additional step.

9.2 Software installation

The Java GUI is available only for the Windows platform and is distributed as a Microsoft Installer package. The GUI itself would run under Linux but the USB interface device driver is currently only supported on Windows.
To install, simply unpack the distribution into a temporary directory and double-click the **`swdaq.msi`** installer file to launch the installation. Administrator privileges are probably required. The package is installed under **`C:\Program Files\CBSW`** and shortcuts are created in **`Start Menu`** in the **`CBSW`** program group.
The package depends on the Cygwin run-time library. If you already have Cygwin installed, copy **`cygwin1.dll`** from your Cygwin bin directory into **`C:\Program Files\CBSW\java\`**. If you do not have Cygwin, either install it and copy **`cygwin1.dll`** as above or just use the copy of **`cygwin1.dll`** distributed with this package. If you update your Cygwin installation you may need to copy any new version of **`cygwin1.dll`** that may have come with it.

9.3 Installing the USB drivers

Connect the L1 board to a USB port on the PC where you want to run the GUI and switch on the L1 board. The system should detect the new USB device and invite you to install the drivers. Follow the instructions in the wizard to "Install from a specific location". The driver files are located in the directory **`C:\Program Files\CBSW\USBXpress\Driver\`**. Administrator privileges will be needed for this. Multiple L1 boards may be installed in this way. They must have unique USB serial numbers. You will be invited to install drivers each time you connect a new board for the first time.

9.4 Starting the GUI

The GUI should run on any system having Java VM version 1.4 or higher. To start the GUI use the **`Start Menu\CBSW\L1`** shortcut. After a few moments (the Java VM takes a little time to start) a console window will appear containing diagnostic messages and with it the main panel of the GUI. Normally the console will contain a message like **`Found 1 device`** if the USB drivers have been installed correctly and the L1 board is switched on. The serial number of the board should be displayed in the pull-down list next to the **`Open L1`** button. If you have other L1 boards in the system, these should also appear in the list. The GUI can have one device open at a time.
When you click on **`Open L1`** the GUI should connect to the selected L1 board and will try to establish the current status. If the L1 board has not yet been initialised the GUI will initialise it with the last used values for that board or the built-in defaults. If the L1 board had already been initialised the GUI will try to update its internal state to match the hardware state.

9.5 Configuring the L1 board

Many of the default settings should be OK for most applications. However the network parameters will vary from location to location and will need to be compatible with your local network infrastructure. It is highly recommended that a dedicated network interface card on a private network is used to receive the data from the L1 board.
The parameters that you need to set are the source and destination MAC and IP addresses for the data transmission. These can be found in the **`Setup\Network…`** panel. The source addresses should be unique to the L1 board and unique on the network to which the board is connected. The destination MAC and IP addresses are those of the network interface card to where the data should be sent.

9.6 Data acquisition

The default (reset) state of the L1 board is to have all input channels disabled, data transmission disabled and decoding of TFC commands disabled. To acquire data, first enable the input channels (**`Setup\All Ingress\Enable`**), then select **`Enable`** on the main panel to enable TFC decoding and data transmission. At this point it is recommended to issue a TFC front-end reset to synchronise the L0 and L1 bunch and event count IDs. Now the L1 board is ready to receive data and transmit formatted events to the readout network. No further interaction through the USB interface is required during data acquisition as the read out is controlled entirely by the TFC triggers. However, while the GUI has an L1 connection open it will periodically request and display status information from the L1 board.

9.7 Other options

Like the production boards a number of different operation modes are supported. These include:

- ALICE and LHCb mode
- MEP building and destination addressing via TFC ROT and destination broadcasts
- Optional zero-suppression (enabled by default)
- Optional suppression of empty HPDs (disabled by default)

All of the above may be controlled from the GUI. The firmware also supports the L1 masking of pixels but this is not available via the Java GUI.

9.8 Data recording

The data format produced by the L1 board is compatible with the LHCb event-building software. Alternatively simpler set-ups may use the **`ukl1-datacapture`** program which saves data in MDF formatted files that can be read by standard LHCb offline tools.

9.9 Bugs, features and workarounds

The principle known bug in the GUI is that it is not protected against attempts to make USB transactions when the target device is closed. Doing so can cause the application to start using all CPU cycles and the system to become unresponsive. If this happens kill the process using Task Manager.