UK L1 prototype revision 3 module

Clive Barham, Sven Katvars and Steve Wotton

Contents

Figure 1 Dataflow and buffering

Status register summary

Register number	Description
0	General status
1	E100(0)
2	E100(16)
3	E100(1)
4	Buffer remainder
5	TTCrx L0 (L1ACC) count
6	Buffer count 0-2
7	Buffer count 3-5
8	Event count low
9	Event count high
10	Parity error counters
11	Parity error counters
12	E100 GT CRC
13	E100 GT CRC
14	TTC ID sense
15	Flow control counters
16	In[0] status
17	In[1] status
18	In[2] status
19	In[3] status
20	In[4] status
21	In[5] status
22	In[6] status
23	In[7] status
24	In[8] status
25	In[9] status
26	In[10] status
27	In[11] status
28	TTC[0]
29	TTC[1]
30	TTC[2]
31	TTC[3]

Status register details

Register 0, General status

Bit	Description	Notes
0	Global reset status	0 = not reset, 1 = reset
1	Global ready status	0 = ready, 1 = not ready
2	Top system clock DLL status	0 = not locked, 1 = locked
3	Bottom system clock DLL status	0 = not locked, 1 = locked
4	TTCrx ready status	0 = not ready, 1 = ready
8	SDRAM initialisation flag	0 = not ready, 1 = ready
9	Transmitter fault status	0 = OK, 1 = fault
10	100baseTX management status	0 = ready, 1 = not ready
11	Receiver signal detect status	0 = no signal, 1 = signal detected

Register 1, 100baseTX register 0

Bit	Description	Notes
15..0	100base TX MDI register 0	See manual

Register 2, 100baseTX register 16

Bit	Description	Notes
15..0	100base TX MDI register 16	See manual

Register 3, 100baseTX register 1

Bit	Description	Notes
15..0	100base TX MDI register 1	See manual

Register 4, Buffer remainder

Bit	Description	Notes
7..0	Word remainder of last used buffer row, 0-2	
15..8	Word remainder of last used buffer row, 3-5	

When reading data from the L1 buffers, the last occupied memory row may not be fully occupied with valid data. This register contains the number of valid 32-bit words in the last row containing valid data or 0 if the last row contains no valid data.

Register 5, TTCrx L0 (L1ACC) counter

Bit	Description	Notes
15..0	Number of L0 triggers seen	

Register 6, L1 buffer row count, memories 0, 1 and 2

Bit	Description	Notes
14..0	Number of complete rows written	1,2,3

1. The register contains the number of completed memory rows for memories 0, 1 and 2. Partial rows are not counted so an additional row should always be read to ensure that all valid data are extracted.
2. Memories 0, 1 and 2 share the same memory controller so there is only one counter for the three memories.

3. Each memory contains data from two input channels.

Register 7, L1 buffer row count, memories 3, 4 and 5

Bit	Description	Notes
14..0	Number of complete rows written	1,2,3

1. The register contains the number of completed memory rows for memories 3, 4 and 5. Partial rows are not counted so an additional row should always be read to ensure that all valid data are extracted.
2. Memories 3, 4 and 5 share the same memory controller so there is only one counter for the three memories.
3. Each memory contains data from two input channels.

Register 8, event counter low

Bit	Description	Notes
15..0	Number of events written, low word	1,2

1. The register contains the lower 16 bits of the event counter.
2. The event counter is only sensitive to the lower 6 input channels.

Register 9, event counter high

Bit	Description	Notes
7..0	Number of events written, high byte	1,2

1. The register contains the high 8 bits of the event counter.
2. The event counter is only sensitive to the lower 6 input channels.

Register 10, parity error counters

Bit	Description	Notes
7..0	Parity error counter	channel 0
15..8	Parity error counter	channel 1

Register 11, parity error counters

Bit	Description	Notes
7..0	Parity error counter	channel 2
15..8	Parity error counter	channel 3

Register 12, Last but one transmitted word

Bit	Description	Notes
15..0	Last but one word transmitted	GT transceiver RXD

Register 13, Last transmitted word

Bit	Description	Notes
15..0	Last word transmitted	GT transceiver RXD

Register 14, TTCrx ID sensed

Bit	Description	Notes
7..0	TTCrx ID sensed from pullup/downs	1

1. The value 0x41 is currently set for no particular reason.

Register 15, egress flow control counters

Bit	Description	Notes
3..0	Events out of egress RAM	1
7..4	Events into egress RAM	1
11..8	Events out of egress multiplexer	1
15..12	Events into egress multiplexer	1

1. These 4 counters must normally take the same value and increment by one for each frame transmitted.

Register 16+n, channel n status

Bit	Description	Notes
0	Inhibit status	0 = not inhibited, 1 = inhibited
1	Loss of sync status	0 = synced, 1 = not synced
7..4	RX buffer overflow counter	Normally unchanging
11..8	RX clock correction counter	Normally counting
15..12	ZS FIFO event counter	Increments per event

Register 28-31, TTCrx register

Bit	Description	Notes
7..0	Register value	See TTCrx manual
14..8	Register ID	See TTCrx manual
15	Read/write flag	0 = write, 1 = read

Control register summary

Register number	Description	
0	DRAM read select	
1	DRAM read address	
2	L0 emulator control	
3	TTC A channel pulser count	
4	TTC A channel pulser interval	
5	TTC encoder control	
6	TTC B channel data (low)	
7	TTC B channel data (high)	
8		
9		
10		
11		
12		
13		
14		
15		
16	In[0] control	
17	In[1] control	
18	In[2] control	
19	In[3] control	
20	In[4] control	
21	In[5] control	
21	In[6] control	
23	In[7] control	
24	In[8] control	
25	In[9] control	
26	In[10] control	
27	In[11] control	
28		
29		
30		
31	IP source address	

Control register details

Register 0, read control

Bit	Description	Notes
2..0	Memory select	1
3	Transmission trigger	2
7..4		
15..8	Number of rows requested	3

1. There are 6 memories, numbered 0 to 5.
2. A **transition** from low to high triggers the transmission of the desired number of memory rows from the selected memory. Therefore the normal procedure would be to first update the selected memory and number of rows keeping bit 3 low, then write the same value to the register again with bit 3 high.
3. The number of rows transmitted will be one more than the number written to the register.

Register 1, read start address

Bit	Description	Notes
14..0	Row number of first row	1
15	Request all rows	Not implemented yet

1. Each group of requested rows starts from the beginning of the memory. To read more rows than can be done in a single transaction, update this register with the start row before triggering the transmission.

Register 2, L0 emulator control

Bit	Description	Notes
0	Emulator inhibit	0 = not inhibited, 1 = inhibited
1	LHCb/ALICE mode select	0 = LHCb, 1 = ALICE
12..8	Burst count request	

This register controls the L0 emulator which can generate events, or a burst of events, for each L0 trigger received via the TTCrx. Events can be either ALICE or LHCb format. When sent as a burst, the inter-event gap is the minimum allowed according to the LHCb specification.

Register 3, TTC A channel pulser count

Bit	Description	Notes
14..0	Pulse count	[0,32767]
15	Pulse train trigger	Trigger on low to high transition

Registers 3 and 4 control a pulser that is synchronous to the clk40 signal of the TTCrx. Do not expect the pulser to work properly if the TTCrx is not properly locked to the incoming TTC clock.

Register 4, TTC A channel pulser interval

Bit	Description	Notes
15..0	Pulse interval	[0,65535]

Register 5, TTC encoder control

Bit	Description	Notes
0	B command trigger	Trigger on low to high transition
1	B mode	0=short, 1=long
2	Auto channel A pulse after calibration	0=disabled, 1=enabled
3	External A channel trigger enable	0=disabled, 1=enabled
4	External A channel trigger mode	0=edge, 1=clocked

Register 6, TTC B channel data (low)

Bit	Description	Notes
15..0	D[15..0]	1

1. For short commands only D[7..0] are significant.

Register 7, TTC B channel data (high)

Bit	Description	Notes
15..0	D[31..16]	

Register 16+n, channel n configuration register

Bit	Description	Notes
0	Zero suppression enable	0 = not enabled, 1 = enabled
1	Channel inhibit	0 = not inhibited, 1 = inhibited
2	LHCb or ALICE mode	0 = LHCb, 1 = ALICE
3	Mode force flag	0 = header detection, 1 = force

This group of registers provide some control over the behaviour of the 12 input channels.

Unused channels are normally automatically sensed and inhibited on reset so the inhibit control is not normally necessary. However, the loss-of-synchronisation indicator uses the inhibit control bits in these registers to mask off the corresponding channels and therefore gives a more useful indication of a problem if unused channels are manually inhibited.

The default operation is to automatically sense whether the incoming data is LHCb or ALICE mode. To force a particular mode use bit 3 of these registers. Bit 2 then selects the desired mode. Bit 2 is ignored if bit 3 is zero.

Register 31, IP source address

Bit	Description	Notes
14..0	Id	2 low octets of IP source address
15	Update trigger	Trigger on high to low transition

The low 15 bits of this register are used to set the low 15 bits of the IP source address and serve also as a module serial number. The high 16 bits of the IP address are hard-coded in the FPGA firmware. This register is intended to be set automatically on reset of the C8051 microcontroller and the value should be set uniquely. However, the register can be modified through the configuration register if required. A low-to-high transition of bit 15 triggers the update.

JP pin assignments

Many of the signals on the monitor pins are for diagnostic use and are subject to change. The `ttc_pulse` signals however can be assumed to be stable.

JP (1)	Description	IO standard (2)
2−	`ttc_a_channel-`	LVDS out (3)
2+	`ttc_a_channel+`	LVDS out (3)
3−	`ram_rdfifo_we[0]`	LVCMOS25 out
3+	`TXCLOCK_E100`	LVCMOS25 out
4−	`ttc_a_channel-`	LVDS out
4+	`ttc_a_channel+`	LVDS out
5−	`ttcrst`	LVCMOS25 out
5+	`reset`	LVCMOS25 out
6−	`ttc_b_channel-`	LVDS out
6+	`ttc_b_channel+`	LVDS out
7−	`ttc_pulse`	LVCMOS25 out
7+	`reset_in`	LVCMOS25 out
8−	`e100_txclkdv`	LVCMOS25 out
8+	`SERBCHAN_TTC`	LVCMOS25 out
9−	`cfg_ttc_auto_pulse`	LVCMOS25 out
9+	`mclk`	LVCMOS25 out
10−	`trigger_a_channel-`	LVDS in
10+	`trigger_a_channel+`	LVDS in
11−	`ttci2c_reset_i2c`	LVCMOS25 out
11+	`ttci2c_en`	LVCMOS25 out
12−	`ttci2c_ready`	LVCMOS25 out
12+	`ttci2c_active`	LVCMOS25 out
13−	`ttci2c_sda`	LVCMOS25 out
13+	`ttci2c_scl`	LVCMOS25 out

1. The −+ symbols indicate the polarity when used for differential signalling. The left-hand pin of each pair is the positive side.
2. 2.5V signalling is used for all monitor pins.
3. On some boards these header pins are removed and the signal routed to the front-panel RJ-11 connector.

Front panel LEDs

Front panel LEDs are numbered 0,1,2 with 0 at the top of the module.

LED	Description
Green[0]	Always on
Green[1]	Top **and** bottom system clock DCMs locked.
Green[2]	Global ready status.
Yellow[0]	TTCrx L0 (L1ACC) trigger.
Yellow[1]	L1 buffer writing. Or of all buffers.
Yellow[2]	L1 buffer reading. Or of all buffers.
Red[0]	RX loss of sync. Or of all unmasked input channels.
Red[1]	TTCrx not ready.
Red[2]	TTCrx I2C ack error.

Usage notes

Setting up

The card is 6U VME but won't fit in an old standard VME crate because the card guides are too narrow. I think the VME64 standard has wider card guides and these should be OK. However, the card guides can be removed from the corresponding slot and the module secured with front panel screws. Alternatively, bench mount the board and use a 5V/5A power supply and a fan to cool the board. The fan is mandatory as both the FPGA and the optical receivers get hot.

The non-volatile memory for the FPGA configuration requires the Xilinx Parallel IV cable or USB cable. It is possible to download directly to the FPGA whenever the module is powered using the ordinary (Xilinx Palallel III) cable.

For the Ethernet use a second 100Mbit NIC in the PC so that the L1 board is isolated from the normal network. For a point-to-point connection a special NIC-to-NIC Ethernet cable is needed (the standard ones won't work because the TX and RX have to be crossed). Alternatively use a standard Ethernet switch with standard cables. A switch is also convenient for applications using multiple L1R3 boards.

The configuration and control interface

All configuration and control proceeds via the sending and receiving of messages on the USB interface. Messages are sent to the module to request an action and some request types result in the module sending a response message. Command messages have the following structure:

```
struct devicePacket_t
{
  unsigned char cmd;
  unsigned char dummy;
  unsigned short length;
  unsigned char uc[4];
  unsigned char RegisterId;
  unsigned char dummy2;
  unsigned short ConfigurationData;
};
```

The `length` field contains the length of the data part of the message in bytes and should always be 4 though is ignored at present as all messages sent to the module have the same fixed length. The `cmd` field contains the message type and these are briefly summarised below:

Type	Description	Response
StatusRequest	Request status block	Yes
ResetRequest	Request system reset	Yes
ConfigurationData	Request configuration update	Yes
L1ResetRequest	L1 fast reset	No

The `RegisterId` and `ConfigurationData` fields are used only for `ConfigurationData` commands and contain the configuration register ID and data respectively.

When the command requires a response, the returned message has the following structure:

```
struct devicePacket_t
{
  unsigned char cmd;
  unsigned char dummy;
  unsigned short length;
  unsigned short status[32];
};
```

The received `cmd` field will always contain the value `deviceHeader_t::Status`. The returned status array contains the current values of the 32 16bit status registers. Any response resulting from a command must be read by the application before the next command request.

Readout

The board uses USB for configuration and control and the 100Mbit Ethernet for the data transmission.

A simple L1 Ethernet reader based on the same library used by Ethereal exists. It compiles under cygwin and maybe under VC++. It can be used as a model for other applications.

The board auto-senses the connected inputs on reset and auto-senses LHCb or ALICE mode so no configuration is required. After reset, the board is ready to receive triggers. System reset causes a complete reset (similar to power-up reset except that the FPGA firmware is not reloaded) and is fairly slow. A faster "L1 reset" resets only the buffer read and write pointers without wiping the L1 buffer contents.

After sending a burst of triggers, the data can be transmitted across the 100baseTX link. The amount of data that has been buffered can be determined by reading the buffer counter and event counter status registers. The general idea is that a block of data is requested via the USB interface by interacting with the read control registers. The 6 memory buffers (each containing data from two input channels) are read out individually. The chosen memory, the read start address and amount of data must first be set then the actual data transmission can be triggered. The transmission proceeds without any flow control so care must be taken not to request more rows than might overflow a NIC or kernel buffer. A typical readout sequence might therefore proceed as follows:

1) General reset
2) L1 (fast) reset
3) Wait 10ms
4) Enable triggers
5) Acquire events
6) Inhibit triggers
7) Read number of buffer rows written (status register 6,7)
8) Read number of events (status register 8,9)
9) Set read start address to beginning of buffer (write 0 to control register 1)
10) Select memory and number of rows to transmit (e.g. to request 10 rows from memory 2, write 0x0902 to control register 0)
11) Trigger the transmission by writing the same word with the trigger bit (bit 3) set (i.e. write 0x090a to control register 0)
12) Go to 9) and repeat until you have extracted all the data.
13) Go to 2).

Note that when using several boards at the same time, the data source can be distinguished by the IP source address which is supposed to be unique and which is normally configurable in firmware but can be changed through the configuration interface if required. Note that no sleeps should be needed at any time other than after the fast reset. Simultaneously receiving and transmitting events is not likely to work so it is important to ensure that triggers are inhibited before requesting data transmission.

Ethernet frame capture

The WinPcap Ethernet frame capture tools can be used to capture Ethernet frames from the L1 board and the Ethereal program is a convenient tool for debugging. As there is no flow control in the data transmission from the L1 boards and it is possible that Ethernet infrastructure will drop frames when congestion occurs, it is necessary that the frame capture application implements a retry mechanism when frames are lost by requesting the retransmission of frames. For maximum performance it is best to choose a small value of the read timeout parameter in the call to pcap_open(). A value of 10ms should be OK.

Each Ethernet frame contains 1024bytes (one row) of data from a memory buffer. Therefore the last frame with valid data might not be fully occupied since the frame size is not an integral number of event blocks when headers are taken into account. To simplify the extraction of the valid data without the application having to keep track of the number of events and readout mode a register is provided that keeps track of the number of valid words in the last occupied buffer. A value of zero in the register means that the entire last frame contains no valid data. A non-zero value means that there was at least some valid data in the last frame.

Using the Dell PowerConnect 2716

The switch needs to be configured in "managed mode" as some configuration settings must take non-default values.

The L1 cards use IP addresses in the range 192.168.2.16-192.168.2.21. The switch IP address (and NIC IP address) should use addresses on the same network. The default address of the Dell switch is 192.168.2.1 which is therefore OK.

The L1 card fast Ethernet port has auto-negotiation disabled and is set to 100Mbs, full-duplex. The network switch ports should be set similarly.

The "storm control" option must be disabled on all ports used for L1 traffic otherwise the switch may discard packets at high rate.

The TTC encoder

The module is capable of being a TTC source encoder. Both A and B channel LVDS outputs are provided that can be coupled via LVDS-to-ECL translators to the TTCvx module or similar.

The A-channel encoder can generate a stream of regularly spaced pulses. Consecutive pulses are supported. The pulser can be triggered in software or using an external LVDS input. The external input can be either edge triggered or level sensitive (clocked).

The B-channel encoder can generate any short or long command on the B-channel. A special mode is supported that automatically triggers the A-channel pulser when an LHCb calibration command is received on the B-channel. The delay between the

calibration command and A-channel pulse is set using the A-channel pulser interval. It is not possible to synchronise the B-channel commands to a particular bunch-crossing.

Ethernet frame format

Each Ethernet frame is formatted according to the C++ struct `l1frame_t` and its sub-structs. The LHCb MEP header is padded with a few extra bytes. The standard MEP fields contain dummy values. Each frame encapsulates one row of memory containing the RICH data in the field `l1frame_t.uc[1024]`. The complete frame is also padded with some extra bytes beyond the 1024 bytes of RICH data. These must be discarded when concatenating consecutive memory rows.

```
struct l1frame_t
{
  ethernetHeader_t ethernetHeader;
  ipHeader_t ipHeader;
  mepHeader_t mepHeader;
  union
  {
    unsigned char uc[1024];
    unsigned ui[256];
  };

};

struct ethernetHeader_t
{
  unsigned char destinationAddress[6];
  unsigned char sourceAddress[6];
  unsigned short type;
};

struct ipHeader_t
{
  unsigned char versionLength;
  unsigned char dsf;
  unsigned short length;
  unsigned short id;
  unsigned char flags;
  unsigned char fragmentOffset;
  unsigned char ttl;
  unsigned char protocol;
  unsigned short checksum;
  unsigned char sourceAddress[4];
  unsigned char destinationAddress[4];
};

struct mepHeader_t
{
  unsigned char mep[22];
};
```

Data format

After concatenation of the RICH data from the Ethernet frames the data are formatted according to one of the following structs in non-zero-suppressed mode. The byte ordering of the header and data words in the Ethernet buffer is such that the RICH data will appear correct when read and displayed as 32-bit words assuming that the processor uses Intel byte ordering.

```
struct l1LHCbData_t
{
  unsigned l1Header;
  unsigned l0Header[2];
  unsigned pixelData[32];
  unsigned l0Parity;
};

struct l1ALICEData_t
{
  unsigned l1Header;
  unsigned l0Header[2];
  unsigned pixelData[256];
  unsigned l0Parity;
};
```

The L0 headers, pixel data and parity trailer are transported unmodified from the PINT. Consult the PINT documentation for details. The L1 header word is formatted according to Table 1.

Bit	Description	Notes
31	Reserved	
30..16	Event Id	
15..13	Memory bank Id	
12	LHCb/ALICE flag	
11	Zero suppression flag	
10..0	Zero suppression word count	

Table 1 L1 header format

Non-zero-suppressed pixel HPD data

The format of the data block corresponding to one HPD is shown below for non-zero-suppressed data.

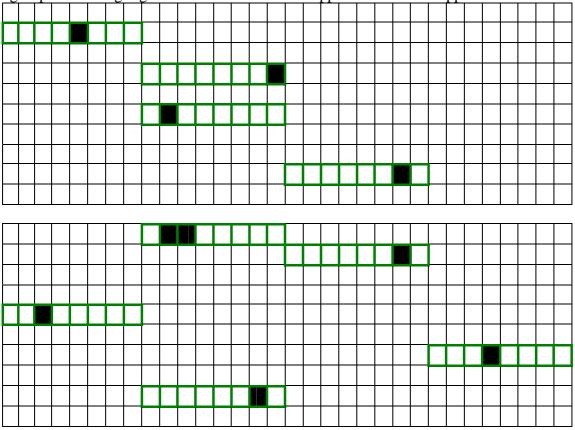| L1 header |
|---|
| L0[0] |
| L0[1] |
| Row 0 |
| Row 1 |
| Row 2 |
| Row 3 |
| … |
| Row 30 |
| Row 31 |
| Parity |

18

Zero-suppressed pixel HPD data

The zero suppressed data will be encoded as a list of bytes having at least one bit set. For LHCb-mode data, 7 address bits are sufficient to uniquely identify a byte within an HPD. There for each non-zero byte can be encoded in a 16 bit word containing the 7 bits of address and 8 bits of data. This is not true in ALICE mode, in this case "fiducial" bytes are always present in the data block.

31	30	29																				9	8	7	6	5	4	3	2	1	0
L1 header																														
L0[0]																														
L0[1]																														
0	Address 1									Byte 1								0	Address 0								Byte 0				
0	Address 3									Byte 3								0	Address 2								Byte 2				
0	Address 5									Byte 5								0	Address 4								Byte 4				
0	Address 7									Byte 7								0	Address 6								Byte 6				
0	…									…								0	…								…				
0	Address n-1									Byte n-1								0	Address n-2								Byte n-2				
0	Unused									Unused								0	Address n								Byte n				
Parity																														

Example of encoding of HPD data (LHCb mode)

The following two arrays represent the pattern of hits from two separate HPDs in an event. The full array is 32 by 32 pixels but only the first 10 rows of each 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 left of each array. The groups of bits highlighted are those that would appear in the zero-suppressed format.

Assuming that the zero-suppression algorithm is applied to the first block but not the second the following data block shows the resulting encoding:

														9	8	7	6	5	4	3	2	1	0
L1 header																							
L0[0]																							
L0[1]																							
0	0x0d				0x80			0		0x04				0x10									
0	0x22				0x40			0		0x15				0x02									
Parity																							
30 pad words																							
L1 header																							
L0[0]																							
L0[1]																							
0x00600000																							
0x00000400																							
0x00000000																							
0x00000000																							
0x40000000																							
0x00000000																							
0x00000080																							
0x00000000																							
0x00040000																							
0x00000000																							
22 words containing 0																							
Parity																							