LDMX NO.	REV.	VALIDITY
0000000	**0.8**	**DRAFT**

Document date: 2022-03-08
Render date: 2022-03-08

SPECIFICATION

ECAL DATA FORMATS SPECIFICATION

ABSTRACT:
This document provides the detailed specification of the LDMX Ecal optical data formats.

DOCUMENT PREPARED BY:	DOCUMENT TO BE CHECKED BY:	DOCUMENT TO BE APPROVED BY:
Jeremiah Mans (UMN)	Erich Frahm (UMN) Nhan Tran (FNAL)	David Hitlin (CIT)

DOCUMENT SENT FOR INFORMATION TO:

Torsten Akesson (Lund)

REV. NO.	DATE	PAGES	DESCRIPTIONS OF THE CHANGES
0.1	2020-04-02	6	Initial version including documentation gleaned from previous talks
0.5	2021-10-19	10	Added information on the HGCROCV2 and the appendix including the superblock format used for the 2021 testbeam
0.8	2022-03-08	11	Added the superblock format used for the 2022 testbeam

Contents

1 Introduction

The exact data rates of the LDMX ECAL data links is dependent on several aspects. The first is the beam or bunch clock. The downlink must be implemented synchronously with the beam clock, as it carries the bunch-by-bunch fast control. Given this requirement, we expect the uplink to be synchronous as well, to simplify the FPGA firmware. However, the uplink and downlink data formats are different from each other, given the asymmetry of controls and other bandwidth requirements. The specification of the data formats will be given in terms of words transmitted, which would result in a different bit rate for a 37.5 MHz bunch clock or a 23.1 MHz bunch clock, which are the two primary rates under discussion.

There are three data formats expected to be in use – the downlink data format, which carries fast control and configuration information to the front-end, the DAQ uplink data format, which carries the DAQ data and responses to configuration and monitoring requests, and the trigger uplink data data format, which carries the data to the trigger processor.

All data formats are based on the industry-standard 8b10b encoding, which is well-supported by the PolarFire FPGA. In addition, the PolarFire includes a deterministic delay mode which is expected to stabilize the phase of the received fast-control data.

The LDMX ECAL uses a downlink format which carries a minimum of eight bytes of payload for each bunch. This implies a data rate of approximately 3 Gbps for a 37.5 MHz bunch rate. The format for the downlink packet is shown in Table 1. Each packet is sixteen bytes long, so requires two bunches for data transmission. There are two fast-control bytes contained within the packet (one for each bunch), each of which is protected by (8,4) Hamming encoding. The Wishbone portion of the packet is protected by a CRC-16 using the CCITT-16 polynomal ($x^{16} + x^{12} + x^5 + 1$).

Table 1: Downlink data format

i	**Data Contents**					
0	Comma (K28.5)					
1	0	Reserved (0)	WB_RST	WB_WE	WB_STR	WB_CYC
2	Hamming(8,4) of Fast Command [7:4] *N*					
3	Hamming(8,4) of Fast Command [3:0] *N*					
4	WB_DATO[31:24]					
5	WB_DATO[23:16]					
6	WB_DATO[15:8]					
7	WB_DATO[7:0]					
8	Comma (K28.5)					
9	1	WB_Target[4:0]			WB_ADDR[17:16]	
10	Hamming(8,4) of Fast Command [7:4] *N+1*					
11	Hamming(8,4) of Fast Command [3:0] *N+1*					
12	WB_ADDR[15:8]					
13	WB_ADDR[7:0]					
14	CRC_16[15:8]					
15	CRC_16[7:0]					

The uplink data format, shown in Table 2, contains a fixed portion associated with the Wishbone bus and a variable portion which depends on the specific link utilization. The fixed portion is kept as small as possible

Table 2: Uplink data format. The last section of the packet carries the DAQ data. The DAQ data blocks are significantly longer than one packet and their format is shown in Table 3.

i	*Contents*				
0	Comma (K28.5)				
1	BXID[2:0]	CRCErr	WB-Byte(X)[1:0]	WB-ERR	WB-ACK
2	WB-DataFromSlave[X+7:X]				
3	CRC-8				
4-N	DAQ Data				

to reserve space for the primary DAQ or trigger data stream. As a result, the readback of any register on the bus requires four BX packets to receive all 32 bits of register data.

2 The Wishbone Bus

The Wishbone Bus follows the open Wishbone standard, allowing for a straightforward implementation of cores for various purposes if required. While the bus is fully documented in the standard (`https://opencores.org/cdn/downloads/wbspec_b3.pdf`), this specification reviews how the bus is implemented into the ECAL data links.

The data link appears as a Wishbone master within the on-detector FPGA firmware. The standard defines several important signals (all shown from the master point of view):

- `CLK_I` – The Wishbone clock is the bunch clock of the experiment.
- `CYC_O` – The cycle signal indicates which target is being addressed. This signal is produced by putting the WB_CYC signal from the data packet through a demultiplexer driven by the WB_Target field of the data packet.
- `STB_O` – The strobe signal is asserted to indicate the beginning of a bus cycle and deasserted to indicate that the handshake from the target was received in the backend electronics. Given the encoding and flight times, there are significant wait states in the process of `STB_O` being deasserted after the target acknoweledges the transfer. This signal is produced by putting the WB_STB signal from the data packet through a demultiplexer driven by the WB_Target field of the data packet.
- `WE_O` – Active-high signal indicating the bus cycle is a write (low during read cycles). Carried in the downlink packet in the (WB_WE) field.
- `ADDR_O` – The 24-bit address field is driven from the packet contents (WB_ADDR). Some targets may use fewer than the full set of address bits.
- `DAT_O` – The 32-bit data from master to target is carried in each downlink packet (WB-DataToTarget).
- `RST` – A reset of the Wishbone state machines, global across all targets. Carried in the downlink packet (WB-Reset).

- `DAT_I` – The 32-bit data from the target to the master, which is valid once `ACK_I` is asserted. This data is transferred to the master over four uplink packets in the (WB-DataFromTarget) field, with the specific byte indicated by the (WB-Byte) field.
- `ACK_I` – The acknowledge from the target is carried in the uplink data format in the (WB_ACK) field.
- `ERR_I` – An optional error signal from the target, carried in the uplink data format in the (WB_ERR) field.

The master signals are registered in the on-detector FPGA and all transitions occur synchronously after the packet is validated. The packet validation requires a successful calculation of the CCITT-16 CRC, calculated over the nine Wishbone-related bytes of the downlink packet. If the CRC is invalid, this failure is indicated on the uplink (CRCErr) and the signals to the Wishbone bus are not updated. Since the signals are generally in place for an extended period of time, a loss of one packet will not have any impact.

3 DAQ Data Format

3.1 Readout Requests

It is possible that LDMX will wish to capture multiple BX of data for each trigger. From the point of view of the ECAL front-end, each request must be considered as an L1A and will be handled independently. In particular, the front-end FPGA will not bundle complete events in this case – this is the responsibility of the off-detector electronics. We follow this strategy to minimize the complexity in the front-end electronics, which is exposed to radiation and is harder to debug. Of particular concern is handling the case where two events overlap in time when consider presamples and postsamples, a case which requires duplication of samples into multiple events.

Therefore, the ECAL DAQ data format refers to the BX readout requests which are encoded as L1A messages to the HGCROC. To limit confusion, we do not refer to event numbers but readout request numbers in the data format.

3.2 Format Details

The DAQ data for a readout request is built of an overall packet structure beginning with a header and ending with a CRC checksum as shown in Table 3. The format begins with a four-bit version field for the use of the offline software in decoding any modifications to the format. The version documented here is version 1.

The header includes an eight-bit FPGA_ID field, the number of subpackets from ROCs ("NLINKS") and the number of total 32-bit words in the packet, including the header and the CRC checksum at the end ("LEN"). The header includes alignment information with the bunch id ("BXID"), readout request number ("RREQ"), and a ten-bit bunch-train/orbit counter ("OR"). After the two header words is a series of words containing eight bits per ROC readout link including the number of 32-bit words from that link and two data-quality bits. The "RIDok" bit indicates that the BXID, RREQ, and OR match between the FPGA and the ROC. The "CRCok" indicates that the CRC calculated by the FPGA matches that in the packet from the FPGA.

Inside the packet are a sequence of blocks sent by the readout chips as shown in Table 5. The DAQ FPGA receives a block from each ROC and carries out optional zero suppression. The FPGA provides the ROC_ID label and checks the CRC sent by the ROC, setting the "CRC ok" bit in the per-ROC header. The per-ROC

Table 3: DAQ Data packet structure from an ECAL front-end FPGA.

i	Data Contents											
0	FMTVER=1		FPGA_ID[7:0]			NLINKS[5:0]			Rsvd (0)	LEN[11:0]		
1	BXID[11:0]					RREQ[9:0]				OR[9:0]		
2	RID ok3	CRC ok3	LEN3[5:0]	RID ok2	CRC ok2	LEN2[5:0]	RID ok1	CRC ok1	LEN1[5:0]	RID ok0	CRC ok0	LEN0[5:0]
⋯	*Additional header lines as needed*											
	ROC subpacket 0											
	ROC subpacket 1											
	• • •											
$N-1$	CRC-32 (FPGA)											

header also contains a forty-bit vector “ReadoutMap” which indicates for each 32-bit word sent by the ROC whether it was copied into the output packet.

Table 4: Data subpacket from a single e-link of a single readout chip for the HGCROCV2, as used in 2021 tests. Each readout chip produces two such blocks.

j	*i*	Data Contents				
	0	ROC_ID[15:0]		CRC ok=0	Reserved (0)	ReadoutMap[39:32]
	1	ReadoutMap[31:0]				
0	2	1 0 1 0	1 0 1 0	BXID[11:0]	WADD[8:0]	1 0 1
1	3	TOT (Ch 0)		TOA (Ch 0)	ADC (Ch 0)	
⋯	⋯	• • •				
18	20	TOT (Ch 17)		TOA (Ch 17)	ADC (Ch 17)	
19	21	1 0	0 (10 bits)	Common Mode ADC (0)	Common Mode ADC (1)	
20	22	TOT (Calib Cell)		TOA (Calib Cell)	ADC (Calib Cell)	
21	23	TOT (Ch 18)		TOA (Ch 18)	ADC (Ch 18)	
⋯	⋯	• • •				
38	40	TOT (Ch 35)		TOA (Ch 35)	ADC (Ch 35)	
39	41	IDLE or missing				

The index j refers to the ReadoutMap[39:0] array. For each element j, ReadoutMap[j]=1 indicates that the 32-bit word indicated is present in the readout after zero suppression. The WADD field refers to the write address of the sample in the memory of the HGCROCV2 and is effectively duplicative with the BXID.

The first 32-bit word sent by the ROC contains the bunch id (“BXID”), the low six bits of the readout request number (“RREQ”), the low three bits of the bunch train counter (“OR”), as well as three bits indicating if Hamming errors were seen in the ROC memory.

The majority of the payload words carry the data for the channels : ten bits of previous time slice ADC, ADC(t-1), ten bits of current time-slice ADC or TOT (as appropriate), and ten bits of TDC (or time-of-arrival) for the pulse. The first two bits of each 32-bit word ($f_1 f_0$) indicate whether the channel data is valid ADC, TOT, or neither. A value of “00” indicates normal ADC while a value of “11” indicates normal TOT.

Table 5: Data subpacket from a single e-link of a single readout chip. Each readout chip produces two such blocks.

j	i	Data Contents					
	0	ROC_ID[15:0]	CRC ok	Reserved (0)	ReadoutMap[39:32]		
	1	ReadoutMap[31:0]					
0	2	0 1 0 1	BXID[11:0]	RREQ[5:0]	OR[2:0]	HE[2:0]	0 1 0 1
1	3	1 0	0 (10 bits)	Common Mode ADC (0)	Common Mode ADC (1)		
2	4	f_1 f_0	ADC(t-1) (Ch 0)	ADC/TOT (Ch 0)	TOA (Ch 0)		
…	…	• • •					
19	21	f_1 f_0	ADC(t-1) (Ch 17)	ADC/TOT (Ch 17)	TOA (Ch 17)		
20	22	f_1 f_0	TOT (Calib Cell)	TOA (Calib Cell)	ADC (Calib Cell)		
21	23	f_1 f_0	ADC(t-1) (Ch 18)	ADC/TOT (Ch 18)	TOA (Ch 18)		
…	…	• • •					
38	40	f_1 f_0	ADC(t-1) (Ch 35)	ADC/TOT (Ch 35)	TOA (Ch 35)		
39	41	CRC-32 (ROC)					

The index j refers to the ReadoutMap[39:0] array. For each element j, ReadoutMap[j]=1 indicates that the 32-bit word indicated is present in the readout after zero suppression. The f_1f_0 field indicates whether the channel data is valid ADC, TOT, or neither. A value of 00 indicates normal ADC, a value of 11 indicates normal TOT, while 01 indicates TOT-underway so ADC is invalid and 10 should not occur.

The value "01" indicates that the TOT is still integrating charge, so the ADC is invalid (either saturated or in undershoot) and the TOT is not valid at this timeslice. The value "10" is invalid and should not appear.

The CRC-32 at the end of the ROC data can only be checked if no words were dropped from the ROC payload. Therefore, if any zero suppression is planned, the transmission of the CRC should be dropped as a useless overhead.

3.3 Analysis of Data Volume

The baseline design of the LDMX motherboard has 42 elinks coming to each DAQ FPGA. A readout-request which reads out the full detector therefore requires transferring

$$2 + \left\lceil \frac{42}{4} \right\rceil + (42 \times 42) + 1 = 1778 \text{ 32 bit words}$$

Assuming that the uplink is configured to carry sixteen bits of payload (6 Gbps at 37.5 MHz bunch rate), each bunch's packet will transfer three 32-bit words (see Table 2). Therefore, 593 crossings will be required to fully transfer one full-detector readout request. When there is no DAQ data to transfer, the link will transfer K(23.7) control characters as padding.

For a configuration where each LDMX event produces a single readout request, the maximum possible trigger rate is 63.237 kHz. In addition, a safety factor is required to allow the event buffers to clear in the FPGA in a timely manner, which limits the maximum trigger rate without zero suppression to approximately 50 kHz. This is well-matched to the rest of the LDMX experiment and provides a substantial safety factor with respect to the target readout rate of 5 kHz or the system requirement to handle up to 25 kHz.

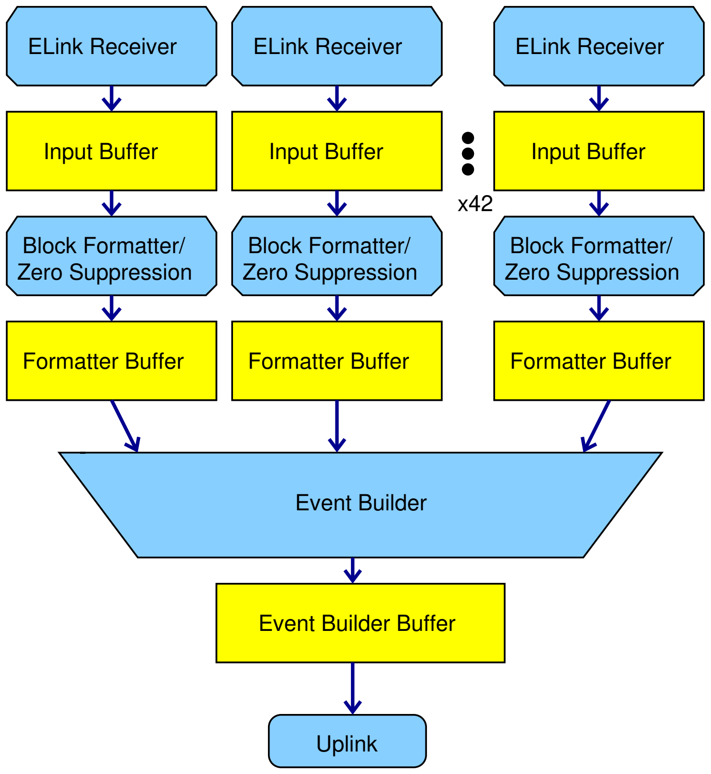

Figure 1: Structure of the blocks and data flow of the DAQ firmware for the LDMX ECal.

However, if multiple readout requests are necessary to provide presamples or post-samples, then the maximum trigger rate would be reduced proportionally. In this situation, the zero-suppression capability of the FPGA would probably be required to substantially reduce the data volume from the on-detector FPGA. In any case, zero-suppression would be necessary either in the on-detector electronics or in the off-detector electronics to limit the data volume written to disk. The off-detector electronics is generally simpler to debug, which suggests carrying out such data reductions at the off-detector stage if practical.

3.4 Trigger Rules

The structure of the DAQ firmware is shown in Fig. 1. The following design information is useful to consider when defining trigger rules.

- The ROC contains 32 internal buffers, which applies an absolute upper limit of 32 readout-requests in a row. As each readout request requires 40 bunch-periods to empty, the maximum readout-request rate following a burst is one every 40 bunch-periods.
- Each input link from a ROC is assigned one 20kb RAM as an input buffer, which is configured as a total of 512 32-bit words of buffer along with Hamming correction codes. This buffer will hold a total of 12 readout-requests.
- Each input link from a ROC is assigned a post-formatting buffer. This buffer is fed from the input buffer by carrying out zero suppression, verifying the CRC, and providing the header. The data is then stored in the formatting buffer until pulled out by the event builder to construct full events. The formatter

buffer contains a header FIFO containing the number of words and other information for the overall header.

- The event building requires pulling samples from the formatter buffer and putting them on the output link. If we assume a 6 Gbps data link, the event builder will have three times the output bandwidth of the input bandwidth of a single ROC link. However, the event builder must service up to 42 input links, so the total bandwidth is much lower than the input bandwidth. The output buffer is a single 20 kb RAM which serves primarily for flow balanacing.

The following trigger rule is proposed:

- *No more than 10 triggers in 400 bunch-periods.* This rule fully protects the HGCROC buffers in single-readout-per-trigger mode and also provides a substantial time for backpressure to be applied in a multiple-readout-per-trigger configuration.

In addition, the long-term trigger rate must be limited to avoid saturation of the readout data links. This rate, however, will be dependent on average event size and cannot be simply stated as a trigger rule. The on-detector FPGA has sufficient buffering capability to smooth this rate and allow for backpressure to reduce trigger rates when necessary.

4 Trigger Data Format

A Testbeam 2021 Data Format

For the 2021 Testbeam, the header below is used to group the set of readout packets which together represent a full event.

Table 6: DAQ multiple packet structure for a single event, inserted into the data stream by the off-detector FPGA for testbeam 2021. The constant values at the beginning and end of the packet are inserted to simplify identifying the boundary between events in a raw data file. The `LENTOTAL` does not include the recognition words at the beginning and end of the packet.

i	**Data Contents**				
	0x11888811				
	0xBEEF2001				
0	FMTVER=1	FPGA_ID[7:0]	NSAMPLES[3:0]	LENTOTAL[15:0]	
1	Reserved=0	LEN1[11:0]		Reserved=0	LEN0[11:0]
···	*Additional header lines as needed*				
	Sample packet 0				
	Sample packet 1				
	• • •				
	0xD07E2001				
	0x12345678				

B Testbeam 2022 Data Format

For the 2022 Testbeam, the header below is used to group the set of readout packets which together represent a full event. This format differs from the 2021 format in the following ways:

- The format version is updated to 2 for the overall header
- The pre-event header is simplified to a single word `0xBEEF2022`.
- The post-event trailer is removed.
- There are always length entries for 16 samples regardless of how many are actually present. The `NSAMPLES` field remains accurate.
- The total length `LENTOTAL64` is now in 64-bit words rather than 32-bit words.
- Each event consists of a series of samples. A sample block contains all the channels (ROC elinks) with the data for a single "40 MHz" cycle. In the V2 format, these sample blocks are aligned on 64-bit boundaries. If the sample block as a whole is not an even number of 32-bit words, a padding word of zeros will be inserted such that the next sample starts on a 64-bit boundary.

Table 7: DAQ multiple packet structure for a single event in 32-bit format, inserted into the data stream by the off-detector FPGA for testbeam 2022. The constant value at the beginning of the packet is inserted to simplify identifying the boundary between events in a raw data file. The `LENTOTAL64` includes all 64-bit words in the packet. The sample `LEN` values are in units of 32-bit words.

i_{32}	**Data Contents**				
0	0xBEEF2022				
1	FMTVER=2	FPGA_ID[7:0]	NSAMPLES[3:0]	LENTOTAL64[15:0]	
2	Reserved=0	LEN1[11:0]		Reserved=0	LEN0[11:0]
3	Reserved=0	LEN3[11:0]		Reserved=0	LEN2[11:0]
4	Reserved=0	LEN5[11:0]		Reserved=0	LEN4[11:0]
5	Reserved=0	LEN7[11:0]		Reserved=0	LEN6[11:0]
6	Reserved=0	LEN9[11:0]		Reserved=0	LEN8[11:0]
7	Reserved=0	LEN11[11:0]		Reserved=0	LEN10[11:0]
8	Reserved=0	LEN13[11:0]		Reserved=0	LEN12[11:0]
9	Reserved=0	LEN15[11:0]		Reserved=0	LEN14[11:0]
	Sample packet 0				
	Sample packet 1				
	• • •				