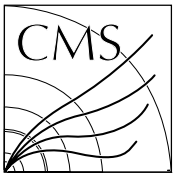
Available on CMS information server

CMS NOTE 2005/021



The Compact Muon Solenoid Experiment

CMS Note

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

Nov 23, 2005

Electromagnetic Calorimeter Raw Data Format

N. Almeida, R. Alemany, A. Jain, J.C. da Silva and J. Varela

LIP Lisbon, Portugal

P. Busson, M. Cerutti, Y. Geerebaert, P. Paganini, M. Bercher

LLR, CNRS/IN2P3 Palaiseau, France

M. Dejardin, J.L. Faure, P. Gras, I. Mandjavidze

DAPNIA, CEA Saclay, France

N. Marinelli

IASA Athens, Greece

T. Camporesi, M. Hansen

CERN, Geneva, Switzerland

E. Vlassov

ITEP, Moscow

Abstract

This document describes the Electromagnetic Calorimeter (ECAL) raw data format, which is generated by the ECAL Data Concentrator Card (DCC) sitting on the ECAL Off-Detector electronics crates. The DCC actions in response to possible errors in the input data are discussed. A software package responsible for the event decoding and data integrity monitoring, developed for the ECAL On-line and Off-line systems, is presented.

1 Raw Data Requirements

A CMS event consists of several subdetector data blocks. The Electromagnetic Calorimeter (ECAL) data are made up of 54 event fragments processed in the Data Concentrator Cards (DCC)[1][2] (36 for the ECAL Barrel (EB) and 18 for the ECAL End-Cap (EE)). The main ECAL raw data requirements are listed bellow:

- **2 KBytes of data size per DCC**: In total a CMS event should not exceed the target data size of 1 MByte, 10% of the data volume was attributed to the ECAL detector. The detector is served by 54 DCCs. For a 100 KHz trigger rate, each DCC will contribute with a maximum average throughput of near 200 MBytes/s, or a data volume of 2 KBytes per event.
- **Selective Readout (SR) flags, crystal and trigger data collection:** Each DCC is a common collection point of crystal and trigger data. The crystal data are used for precise cluster reconstruction and energy measurement, while the trigger data may be used along with crystal data to establish the complete energy flow in the calorimeter. The selective readout flags must also be present in the collected data in order to track the filtering conditions applied to each Front-End (FE) readout board.
- **Channel characterization:** The DCC input channels can be enabled or disabled by configuration. The DCC performs input data verification and can flag the channels with errors. These flags must be known to the off-line analysis and therefore must be present in the event format.
- **Minimal data formatting**: In order to avoid the possibility of data corruption, only a minimal input data formatting should be performed by the DCC. In particular, the possibility of using complex on-line data compression schemes was rejected.
- **Output DAQ format**: All event fragments must be transmitted to the central acquisition system (DAQ) with the common format specified by the CMS DAQ group [3]. In particular, data fragments are required to be organized in 64 bit data words and include a standard header and trailer words.

2 Raw Data Structure

The ECAL raw data format is shown in Figure 1.Here and in the following the fragment of a CMS event produced by a single DCC is called “DCC Event”. Each DCC event consists of five block types:

- DCC Header Block
- TCC Block
- SR Block
- Tower Block
- Crystal Block

These blocks are organized in 64 bit words and are enclosed by the standard CMS DAQ header and trailer. Each block is characterized by a unique bit field identifier, which helps to track possible data formatting errors.

The standard CMS DAQ header is followed by the specific ECAL DCC Header Block. Trigger data received from the Trigger Concentrator Card (TCC) [2] are grouped in the TCC Blocks. A DCC event contains one TCC block for the EB DCCs and four for the EE DCCs.

ECAL Front-End (FE) data must be reduced by a factor of near 20. Data filtering is based on a Selective Readout algorithm finding the high interest areas of ECAL that must be read out with low zero suppression or without zero suppression at all. The other areas are readout with TPG granularity and precision or read out with a higher zero suppression threshold. The Selective Readout Processor (SRP) [2][4] produces for each readout unit (RU) a flag identifying the level of zero suppression the DCC must apply to the RU data. In the barrel an RU corresponds to a Trigger Tower (TT), in the endcap an RU corresponds to a Super Crystal. These Selective Readout (SR) flags are grouped in the SR Block.

The number of Trigger Tower (or Super Crystal) Blocks in each event depends on the enabled FE channels and on the SR flags, which can suppress the entire FE channel data. The number of Crystal Blocks belonging to a given Trigger Tower Block (or Super Crystal) is dynamic and depends on the deposited energy and on the associated SR flag.

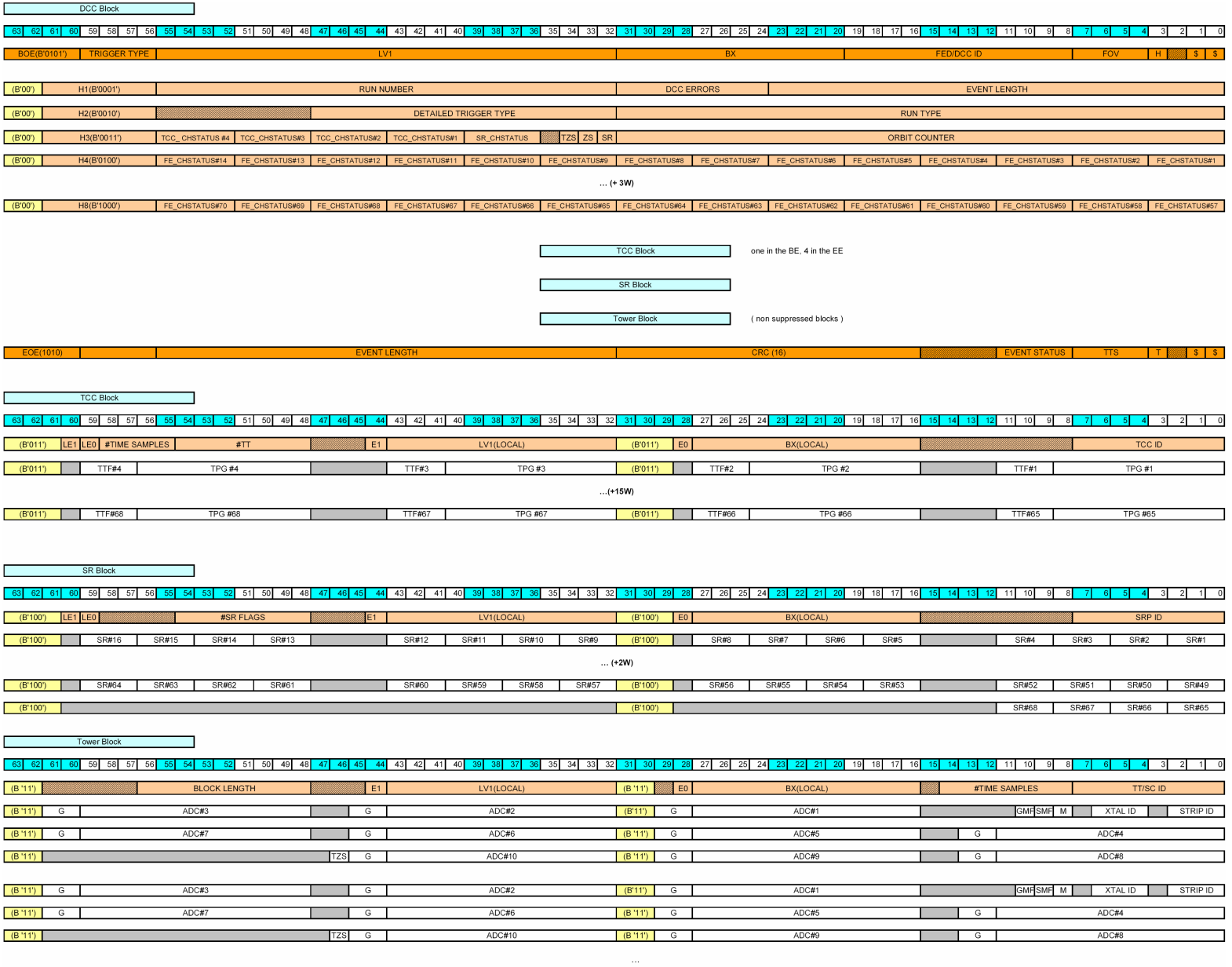


Figure 1. The DCC raw data format.

2.1 DCC Header Block

The DCC Header Block consists of eight words (of 64 bits) characterized by a block identifier (B´00´) at the beginning of each 64 bit word, and a header word number (from 1 to 8). The block carries the following information:

- EVENT LENGTH: Number of 64 bit words in the event, also present in the CMS DAQ trailer. This number includes the CMS DAQ header and trailer words. The event length is repeated at the beginning of the event to ease fast decoding and event characterization.

- DCC ERRORS: DCC error field described in Table 1.

Table 1: DCC error description.

DCC Error (binary)	Description
00000000	No Errors
00000001	Empty event condition. An empty event is produced : DAQ header + the first two DCC header words + DAQ trailer
xxxxxx1x	Data timeout : at least one channel has a data presence timeout[1]
0xxxx1xx	Channel error: at least one channel in error
…	To be defined
1xxxxxxx	No error check performed

- RUN NUMBER: Sequential run number or encoded run conditions as written in the DCC Run ID register by the DCC Supervisor software via VME.

- Hi (i = 1 … 8): Header word count identifier.

- RUN TYPE: Field encoded by the DCC supervisor software in local data taking mode[2] and written into the DCC Run Type register (Figure 2). The content of the different bit fields in Figure 2 is the following:

- **RT_DCCID:** The DCC ID as encoded in the CMS DAQ header (DCC/FED ID).

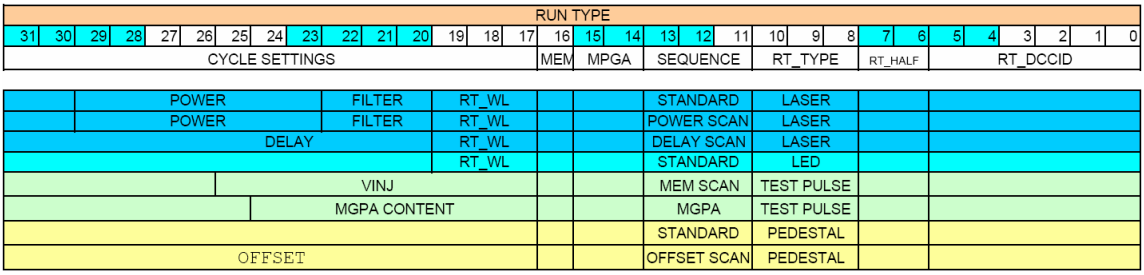


Figure 2. The Run Type word format. The CYCLE SETTINGS encoding depends on the selected SEQUENCE.

[1] At every input channel a timeout is implemented to ignore channels that do not get data from the FE boards before the timeout is reached.

[2] Local data taking mode has standalone timing and trigger (generated by the Local Trigger Controller or by the Trigger FPGA of the Clock and Control System card), and raw data readout via VME.

- **RT_HALF:** Defines which part of the Super Module was readout (Table 2).
- **RT_TYPE:** The run type (Table 3).

Table 2: RT_HALF description.

RT_HALF (binary)	Description
01	First half of Super Module
10	Second half of Super Module
11	Full Super Module

Table 3: RT_TYPE description.

RT_TYPE (binary)	Description
001	Laser
010	Test Pulse
011	Pedestal
100	LED

- **SEQUENCE:** The operation sequence during local data taking. The SEQUENCE encoding depends on the RT_TYPE and its description is shown in Table 4.

Table 4: The SEQUENCE description.

SEQUENCE (binary)	Description
If RT_TYPE is Laser	
000	Standard Laser
001	Laser Power Scan
010	Delay Scan
If RT_TYPE is Test Pulse	
000	MEM Scan
001	MGPA
If RT_TYPE is Pedestal	
000	Standard Pedestal
001	Pedestal Offset Scan
If RT_TYPE is LED	
000	Standard LED

- **MGPA:** The MPGA gain mode (Table 5).
- **MEM:** The MEM channel gain mode (Table 6).
- **CYCLE SETTINGS:** The cycle settings applied during local data taking. The CYCLE SETTINGS encoding depends on the SEQUENCE and is shown in Figure 2. The POWER, FILTER, DELAY, VINJ, MGPA CONTENT and OFFSET fields are binary representations of configuration parameters for each SEQUENCE type. The RT_WL field encodes the Laser/LED wavelength according to Table 7.

- DETAILED TRIGGER TYPE: A Timing and Trigger Control (TTC) "long command" [5] written in the DCC header during calibration triggers expected at the orbit gaps. The content of this word (Figure 3) is the following:

- **DTT_DCCID:** The FED_ID of the DCC which must read the data for this trigger.

- **DTT_HALF:** The same as RT_HALF (Table 2).
- **DTT_TYPE:** The same as RT_TYPE (Table 3).
- **DTT_WL:** The same as RT_WL (Table 7).

Table 5: The MGPA gain.

MGPA (binary)	Description
01	Forced gain 12 (default)
10	Forced gain 6
11	Forced gain 1

Table 6: The MEM gain.

MEM (binary)	Description
0	MEM forced gain 1
1	MEM forced gain 16 (default)

Table 7: The RT_WL description.

RT_WL (binary)	Description
000	Blue (Laser)
001	Green (Laser)
010	Red (Laser)
011	Infra-Red (Laser)
101	λ1 (LED)
110	λ2 (LED)

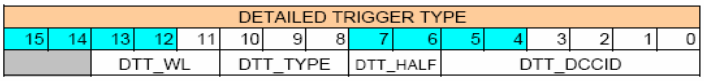


Figure 3. Detailed trigger type word format.

- ORBIT COUNTER: The orbit number in the run.

- SR and ZS: These bits characterize the DCC readout conditions. The SR bit specifies if selective readout is enabled and in the case it is disabled, the ZS field specifies if crystal data are zero suppressed or fully readout.

- TZS: Test Zero Suppression mode. In this operation mode (TZS=1) the zero suppression algorithm (if requested in the DCC configuration or by a SR flag) is applied to the crystal data without data removal. The resulting filter flag is set in the TZS field of the Crystal Block allowing the off-line test of the ZS algorithm.

- SR_CHSTATUS: SRP input channel status (Table 8).

- TCC_CHSTATUS#i: TCC input channel #i status (Table 8).

- FE_CHSTATUS#i: FE input channel #i status (Table 8).

2.2 TCC Block

The TCC Blocks, added after the DCC Block Header, include the trigger data received from the TCC links (Annex A.2). The block is identified by the B(´011´) field at the beginning of each 32 bit word and it consists of the following fields:

- TCC ID: The TCC board id (from 1 to 108).

- BX: TCC local bunch crossing (BX) counter.

- E0: DCC BX error bit (active if there is a mismatch between the TCC BX counter and the DCC BX counter).

- LV1: TCC local level one accept (LV1) counter.

- E1: DCC LV1 error bit (active if there is a mismatch between the TCC LV1 counter and the DCC LV1 counter).

- #TT: Number of TTs associated to this TCC. In the EB each TCC is associated to 68 TTs, which gives a TCC Block size of 18 (words of 64 bits)x64 bits (for one Trigger Primitive Generator (TPG) time sample). In the inner EE each TCC Block includes a maximum of 32 TTs while in the outer EE this number is decreased to 16. The TCC block size in the EE is the same: 9 (words of 64 bits)x64 bits (for one TPG time sample), irrespective of inner or outer EE. The TPG data fields not used are set to zero.

Table 8: DCC Channel status description. If multiple errors are identified the channel status is set according to its priority (a lower channel status value implies a higher priority).

CHSTATUS (binary)	Description
0000	Channel enabled and no errors identified
0001	Channel disabled by configuration
0010	Channel data ignored following to a timeout
0011	Header identifier mismatch
0100	Block word count mismatch
10xx	Sync error (bit 0: LV1 (LeVel 1 accept) , bit 1: BX (Bunch Crossing counter))
11xx	Parity error (bit 0: horizontal parity , bit 1: vertical parity)

Table 9. Trigger tower flags description.

TT Flag (binary)	Description
000	Low interest TT (Tower Et (transverse energy) is bellow the low threshold)
001	Mid-interest TT (Tower Et is between the low and high thresholds)
010	Not used
011	High-interest TT (Tower Et is above the high threshold)
100	Force TT readout (Agilent sync link error)
101	Force TT readout (Hamming code error)
11x	Force TT readout (Other error conditions)

- #TIME SAMPLES: Number of time samples for the TPG data words. In normal data acquisition mode there is only one time sample. For debugging operation mode, 4 or 8 time samples are foreseen. For multiple time samples, TT flags and TPG data are ordered consecutively per set of samples in each TT.

- LE0: Local BX error bit. This bit is received from the TCC data and identifies if there is a mismatch (LE0=1) between the TCC local BX counter and the information received from the TTC system.

- LE1: Local LV1 error bit. This bit is received from the TCC data and identifies if there is a mismatch (LE1=1) between the TCC local LV1 counter and the information received from the TTC system.

- TPG#i: 9 bits coding the trigger primitive of TT#i.

- TTF#i: TT flag (Table 9) sent from the TCC to the SRP and included in the DCC for debugging purposes. The first two least significant bits characterize the TT energy state, while the third bit is enabled if the TT flag was produced with error conditions.

2.3 SRP Block

The SRP Block, added after the TCC Blocks, includes the selective readout flags received from the SRP link (Annex A.3). The block is identified by the B(´100´) field at the beginning of each 32 bit word and consists of the following fields:

- SRP ID: The SRP identification.

- BX: SRP local BX counter.

- E0: DCC BX error bit (active if there is a mismatch between the SRP BX counter and the DCC BX counter).

- LV1: SRP local LV1 counter.

- E1: DCC LV1 error bit (active if there is a mismatch between the SRP LV1 counter and the DCC LV1 counter).

- #SRFLAGS: Number of SR flags associated to this link. In the EB the DCC receives a total of 68 SR flags, while in the EE the DCC receives 34, 35 or 36 flags. The SRP block size for the EB and EE has the same size: 6x64 bits. The SRP flag fields not used are set to zero.

Table 10. Selective Readout flags description.

SR Flag (binary)	Description
000	Suppress FE channel
001	Read FE channel with level 1 zero suppression threshold
010	Read FE channel with level 2 zero suppression threshold
011	Read FE channel without zero suppression
1xx	xx action has been forced by an error condition or by configuration: - TT flagged in forced readout mode (from TCC). - Data transmission errors (TCC and Algorithm Board(AB) link errors)

- LE0: Local BX error bit. This bit is received from the SRP data and identifies if there is a mismatch (LE0=1) between the SRP local BX counter and the information received from the TTC System.

- LE1: Local LV1 error bit. This bit is received from the SRP data and identifies if there is a mismatch (LE1=1) between the SRP local LV1 counter and the information received from the TTC System.

- SR#i: 3 bits coding the SR flags for each FE channel (Table 10). The first two LSB bits characterize the SRP algorithm flag, while the third bit is used to indicate if the flag was produced with an error condition.

2.4 Tower Header and Crystal Blocks

The Tower Block collects filtered FE data received from the FE boards (see Annex A.1), and is organized in Crystal Blocks. The Tower (or Super Crystal in the EE) Block contains a maximum of 25 crystals, corresponding to a fully readout tower (or super crystal). The Tower Blocks and the Crystal Blocks are identified by the B(´11´) field at the beginning of each 32 bit word and consist of the following fields:

Tower Header Block:

- TT/SC ID: The FE card id (from 1 to 68).

- #TIME SAMPLES: Number of time samples for the ADC crystal data (default is 10). The possible values are given by: 2 +4n (where n goes from 0 to 15).

- BX: FE local BX counter.

- E0: DCC BX error bit (active if there is a mismatch between the FE BX counter and the DCC BX counter).

- LV1: FE local LV1 counter.

- E1: DCC LV1 error bit (active if there is a mismatch between the FE LV1 counter and the DCC LV1 counter).

- BLOCK LENGTH: Size of the RU block in 64-bit word unit including the header.

Crystal Block:

- STRIP ID: The tower strip identification (from one to five).

- CRYSTAL ID: The crystal identification inside the strip (from one to five).

- M: Monitoring bit, enabled for monitoring triggers.

- GMF and SMF: bit fields received from the FE strip header (see Annex A.1) describing the level of time sample misalignment. SMF is the Strip Misalignment flag, and GMF is set if any of the strips is misaligned.

- ADC#i: 12 bits representing the crystal time sample i.

- G: MGPA gain (B´00´: not used, B´01´: gain 12, B ´10´: gain 6, B´11´: gain 1).

- TZS: When the DCC is configured to operate in the test zero suppression mode, the TZS bit will be enabled whenever the crystal data frame is flagged to be suppressed.

The crystal numbering in the DCC raw data format follows the FE numbering scheme. The FE identifies each crystal by the strip to which it belongs to (from one to five) and the crystal number inside the strip (from one to five). To get the absolute crystal number inside the TT or inside the Super Module, a lookup table should be applied. The lookup table should take into account the particular arrangement of VFE cards inside the FE card, and the particular arrangement of FE cards inside the Super Module.

2.5 CMS DAQ Header and Trailer

The CMS DAQ Header and Trailer are, respectively, the first and last word of the ECAL event represented in Figure 1.

The DAQ Header includes the fields:

- $: Bit used by the S-Link 64 hardware.

- H: Header bit. When set to '0', the current header word is the last one. When set to '1', other header word can follow (in the ECAL DCC, H is set to ´1´) .

- FOV: Version identifier of the FED/DCC data format.

- FED/DCC ID: The Front End Device identification as defined by the Central DAQ group. The FED numbering assigned to ECAL ranges from 601 to 654.

- BX: Bunch crossing counter (from the TTC System).

- LV1: Level one accept counter (from the TTC System).

- TRIGGER TYPE: Event Trigger Type, defined by the central DAQ and shown in Table 11 [6].

- BOE: Begin of event bit field identifier with value B(´0101').

The DAQ Trailer includes the fields:

- $: Bit used by the S-Link 64 hardware.

- T: Trailer bit. When set to '0', the current trailer word is the last one. When set to ´1´, other trailer words can follow, (in the ECAL DCC, T is set to 0).

- TTS: Current Trigger Throttling System bits.

- EVENT STATUS: Event fragment status (defined by the central DAQ).

- CRC: Cyclic Redundancy Code of the event fragment including header and trailer (see Annex D).

- EVENT LENGTH: The same meaning has the Event Length in the DCC Header Block.

- EOV: End of event bit field identifier with value B(‘1010‘).

Table 11. The TRIGGER TYPE description.

TRIGGER TYPE	Description
0001	Physics trigger
0010	Calibration trigger
0011	Test trigger
0100	Technical trigger (external trigger)
0101	Simulated events (reserved for DAQ usage)
0110	Traced events (reserved for DAQ usage)
1111	Error

3 Errors on the input data and implications on the ECAL Raw data

The identification of errors in the input data fragments has implications on the way the event is built. The actions taken by the DCC when errors are identified in the input data, are described in this section. The DCC is also able to operate with no error check (DCCERR set to ´1xxxxxxxx´).

Synchronization errors: error on the BX or LV1 in an input channel:

- Global action: The channel data are included in the assembled event and the associated channel status field set with the sync error flag.

- Error identified in the SRP Block: No filtering is applied to crystal data.

- Error identified in the FE Block: No filtering is applied to crystal data from this particular channel.

Data timeout, wrong header identifiers or block word counts:

- These are considered severe errors. Data from the faulty channel are ignored and the channel status is flagged with the identified error condition.

Misaligned FE Data (GMF and SMF bits):

- The crystal data with misaligned time samples are not zero suppressed.

4 Raw Data Parsing and Monitoring

Detector raw data need to be decoded and the data quality monitored. In the Off-line system data will serve physics analysis, while in the On-line system local data taking is used mainly to monitor the detector performance.

A software package was developed to allow on-line and off-line applications access all defined raw data bit fields in an object oriented way. In addition, the package verifies the event data structure identifying possible errors. The class diagram for this package, developed in C++, is represented in Figure 4.

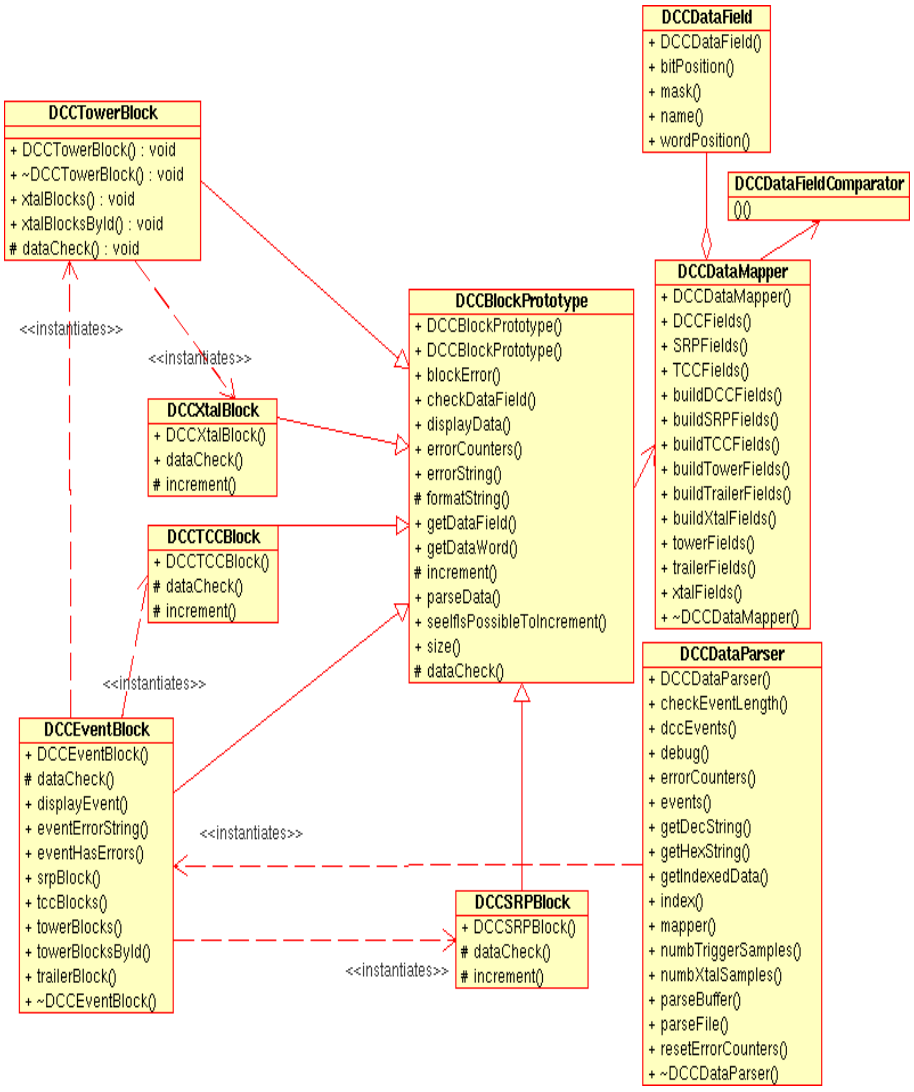


Figure 4. The data parser class diagram

Raw data blocks are characterized by a set of bit fields. Each bit field is defined through a word position inside the block, a bit position inside the word and a defined bit size. The DCCDataField class is a representation of a raw data bit field. These data structures are built and stored in the DCCDataMapper containers. The DCCDataBlockPrototype class encapsulates the knowledge on how to decode a raw data buffer based on the defined bit fields. When parsing the input data buffers the decoded bit fields are stored in a map container, which can be accessed by the applications using the parser. During the parsing process the DCC data blocks classes perform a set of checks on the decoded data. These checks are described in Table 12.

The information about the blocks that need to be created is stored in the data itself. The DCCEventBlock decodes the DCC header bit fields identifying the enabled channels. In a first stage the blocks associated to the enabled TCC and SR channels are created. The DCCTowerBlock instantiation is based on the enabled FE channels and on the SR flags that indicate the rejected channels. The size of the DCCTowerBlock and the knowledge of the number of time samples per crystal allow the determination of the number of readout crystals in each tower or super crystal. For each crystal a DCCXtalBlock object is instantiated. The parsing ends with the decoding of the DCC trailer. The software includes methods that allow the raw data display as a function of the defined bit fields (List 1).

The package describing the 2004 H4 Test Beam ECAL raw data is implemented and is included in ORCA_8_8_0[7] (Calorimetry/EcalBarrelTBDaqDigiFormat). The package is responsible for decoding the

raw data and generating the Calo Data Samples Frames.

Table 12 – Error checks performed during parsing.

Data Block	Error check description
DCCEventBlock	Begin Of Event, header identifiers and block identifier
DCCTCCBlock	Channel id, synchronization and block identifiers
DCCSRPBlock	Channel id, synchronization and block identifiers
DCCTowerBlock	Channel id, synchronization, block identifiers and block size (based on the readout conditions and number of crystal samples)
DCCCrystalBlock	Crystal and strip ids and block identifiers
DCCTrailler Block	End Of Event, event size and CRC

==========

Block name : DCCHEADER, size : 1960 bytes, event WOffset : 0

W[00001] BOE = 00005 TTYPE = 00005 L1 = 00001

W[00003] BX = 00094 ID = 00013

W[00005] H1 = 00001 XSAMP = 00010 TSAMP = 00001 ZS = 00000 SR = 00000 TCC4 = 00000 TCC3 = 00000 TCC2 = 00000 TCC1 = 00000

W[00007] DCCERR = 00000 LENGTH = 00245

W[00009] H2 = 00002

W[00011] ORBCNT = 00000

W[00013] H3 = 00003 CHST14 = 00000 CHST13 = 00000 CHST12 = 00000 CHST11 = 00000 CHST10 = 00000 CHST9 = 00000

W[00015] CHST8 = 00000 CHST7 = 00000 CHST6 = 00000 CHST5 = 00000 CHST4 = 00000 CHST3 = 00000 CHST2 = 00000 CHST1 = 00000

W[00017] H4 = 00004 CHST28 = 00000 CHST27 = 00000 CHST26 = 00000 CHST25 = 00000 CHST24 = 00000 CHST23 = 00000

W[00019] CHST22 = 00000 CHST21 = 00000 CHST20 = 00000 CHST19 = 00000 CHST18 = 00000 CHST17 = 00000 CHST16 = 00000 CHST15 = 00000

W[00021] H5 = 00005 CHST42 = 00000 CHST41 = 00000 CHST40 = 00000 CHST39 = 00000 CHST38 = 00000 CHST37 = 00000

W[00023] CHST36 = 00000 CHST35 = 00000 CHST34 = 00000 CHST33 = 00000 CHST32 = 00000 CHST31 = 00000 CHST30 = 00000 CHST29 = 00000

W[00025] H6 = 00006 CHST56 = 00000 CHST55 = 00000 CHST54 = 00000 CHST53 = 00000 CHST52 = 00000 CHST51 = 00000

W[00027] CHST50 = 00000 CHST49 = 00000 CHST48 = 00000 CHST47 = 00000 CHST46 = 00000 CHST45 = 00000 CHST44 = 00000 CHST43 = 00000

W[00029] H7 = 00007 CHST70 = 00000 CHST69 = 00000 CHST68 = 00000 CHST67 = 00000 CHST66 = 00000 CHST65 = 00001

W[00031] CHST64 = 00000 CHST63 = 00000 CHST62 = 00001 CHST61 = 00001 CHST60 = 00000 CHST59 = 00000 CHST58 = 00000 CHST57 = 00000

==========

```
=====================================================================

Block name : TOWERHEADER, size : 616 bytes, event WOffset : 32

W[00000] L1 = 00001 BX = 00094

W[00001] LENGTH = 00076 ID = 00061

=====================================================================

=====================================================================

Block name : XTAL, size : 24 bytes, event WOffset : 34

W[00000] ADC1 = 00215 M = 00000 XTALID = 00001 STRPID = 00001

W[00001] ADC3 = 00217 ADC2 = 00215

W[00002] ADC5 = 00218 ADC4 = 00219

W[00003] ADC7 = 00218 ADC6 = 00217

W[00004] ADC9 = 00217 ADC8 = 00218

W[00005] ADC10 = 00219

=====================================================================

…

=====================================================================

Block name : DCCTRAILER, size : 8 bytes, event WOffset : 488

W[00000] CRC = 00000

W[00001] EOE = 00010  LENGTH = 00245

=====================================================================
```

List 1. 2004 H4 Test Beam raw data fragment display. The data fragment only includes front end data and has some differences in the DCC Header definition with respect to the final format described in Figure 1.

Acknowledgments

N. Marinelli wishes to acknowledge the Financial support provided through the European Community's Human Potential Programme (Research Training Network) under contract HPRN-CT-2002-00326, (PRSATLHC).

References

[1] **Proc. 9th LECC 2003 Workshop, Amsterdam, Holland, September 2003**, " Data Concentrator Card and Test System for the CMS ECAL Readout", N. Almeida et al.

[2] **Proc. Int. Conf. Calorimetry in HEP, Perugia, Italy, March 2004,** "ECAL Off-Detector Electronics" R. Alemany et al.

[3] **CERN/LHCC 2002-28**, "DAQ and High Level Trigger TDR", CMS Collaboration.

[4] **Proc. IEEE Nuclear Science Symposium, Rome, October 2004**, "The Selective Read-Out Processor for the CMS Electromagnetic Calorimeter ", N.Almeida et al.

[5] **Proc. 6th Workshop on Electronics for LHC Experiments, Krakow, Poland, September 2000**, "Timing and Synchronization in the LHC Experiments", J.Varela.

[6] **CMS NOTE 2002/033 ,** "CMS L1 Trigger Control System", CMS Trigger DAQ group.

[7] **The ORCA project Web site**: http://cmsdoc.cern.ch/orca .

Annex A: DCC Input Data Formats

A1 : FE DATA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	*Comment*
P	1	1	1	0	0	0	1	TT / SUPER CRYSTAL ID								TT / SC id
P	1	0	1	(Local) BX												*local BX id*
P	1	1	0	(Local) LV1												*local LV1 id*
P	1	1	1	1	# PENDING EVENTS						GMF	SMF	STRIP ID: 001			*strip header*
P	1	0	0	FRAME LENGTH								M	XTAL ID: 001			*xtal header*
P	M	G		ADC#1												*xtal sample*
More samples ...																
P	M	G		ADC#n												*xtal sample*
P	1	0	0	FRAME LENGTH								M	XTAL ID: 010			*xtal header*
P	M	G		ADC#1												*xtal sample*
More samples ...																
P	M	G		ADC#n												*xtal sample*
P	1	1	1	1	#PENDING EVENTS						GMF	SMF	STRIP ID: 010			*strip header*
P	1	0	0	FRAME LENGTH								M	XTAL ID: 001			*channel header*
P	M	G		ADC Value												*xtal sample*
More samples, channels and strips...																
P	1	0	0	FRAME LENGTH								M	XTAL ID: 101			*channel header*
P	M	G		ADC#1												*xtal sample*
More samples																
P	M	G		ADC#n												*xtal sample*
P	1	VERTICAL EVEN PARITY														*checksum*

A2 : TCC DATA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	*Comment*
DF	1	1	1	0	0	0	0	TCC ID								TCC id
DF	1	0	1	(Local) BX												*local BX id*
DF	1	1	0	(Local) LV1												*local LV1 id*
DF	1		LE1	LE0	# TIME SAMPLES/TT				#TTs							*sync errors,number of time samples and TTs*
DF	0			TTF # 1				TPG # 1								*trigger flag and TPG sample*
DF	0			TTF #...				TPG # ...								...
DF	0			TTF #68				TPG #68								*trigger flag and TPG sample*
DF	1	VERTICAL EVEN PARITY														*checksum*

A3 : SR Flags

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	*Comment*
P	1	1	1	0	0	0	0	SRP ID								SRP ID
P	1	0	1	(Local) BX												*local BX id*
P	1	1	0	(Local) LV1												*local LV1 id*
P	1		LE1	LE0					#SRFLAGs							*sync errors and number of SR Flags*
P	0			SRFLAG4				SRFLAG3			SRFLAG2			SRFLAG1		*SRFlags*
P	...															...
P	0			SRFLAG68				SRFLAG67			SRFLAG66			SRFLAG65		*SRFlags*
P	1	VERTICAL EVEN PARITY														*checksum*

DF: Link data validation flag (valid on zero).

P : Horizontal odd word parity.

Notes:

TCC events transmitted to the DCC:

- barrel: 73 words (68 TPGs+4 headers+1 trailer)
- endcap: 37 words (32 TPGs+4 headers+1 trailer) (TPG data fields not used are set to zero)

SRP events transmitted to DCC (barrel and endcap):

- 22 words (17 words with 68 flags+4 headers+1 trailer) (SRP flags not used are set to zero)

Annex B: TCC Input Data Format

B1 : FE Trigger Data

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	*Comment*
GAP	HAMMING				FG	ET										FE data input into TCC

Annex C: AB Input Data Format

C1 : TCC TT Flags

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	*Comment*
DF	1	1	1	0	0	0	0	TCC ID								TCC id
DF	1	0	1	(Local) BX												*local BX id*
DF	1	1	0	(Local) LV1												*local LV1 id*
DF	1		LE1	LE0				#TTs								*sync errors,number of TTs*
DF	0			TTF # 4				TTF # 3				TTF # 2				TTF # 1	*trigger flags*
DF	0			...												...
DF	0			TTF #68				TTF #67				TTF #66				TTF #65	*trigger flags*
DF	1	VERTICAL EVEN PARITY														*checksum*

Note: TCC events transmitted to SRP (barrel and endcap)
- 22 words (17 words with 68 flags+4 headers+1 trailer) (TCC flags not used are set to zero)

Annex D: CRC Algorithm

The CRC[3] field calculation is based in the polynom: $1 + x^2 + x^{15} + x^{16}$.

The initial cycle redundant code is set to 0xFFFF and the raw equations are given by the following expressions:

(D refers to the serial input 64 bit data word , and C to the 16 bit CRC)

NewCRC(00) = D(63) xor D(62) xor D(61) xor D(60) xor D(55) xor D(54) xor D(53) xor D(52) xor D(51) xor D(50) xor D(49) xor D(48) xor D(47) xor D(46) xor D(45) xor D(43) xor D(41) xor D(40) xor D(39) xor D(38) xor D(37) xor D(36) xor D(35) xor D(34) xor D(33) xor D(32) xor D(31) xor D(30) xor D(27) xor D(26) xor D(25) xor D(24) xor D(23) xor D(22) xor D(21) xor D(20) xor D(19) xor D(18) xor D(17) xor D(16) xor D(15) xor D(13) xor D(12) xor D(11) xor D(10) xor D(09) xor D(08) xor D(07) xor D(06) xor D(05) xor D(04) xor D(03) xor D(02) xor D(01) xor D(00) xor C(00) xor C(01) xor C(02) xor C(03) xor C(04) xor C(05) xor C(06) xor C(07) xor C(12) xor C(13) xor C(14) xor C(15);

NewCRC(01) = D(63) xor D(62) xor D(61) xor D(56) xor D(55) xor D(54) xor D(53) xor D(52) xor D(51) xor D(50) xor D(49) xor D(48) xor D(47) xor D(46) xor D(44) xor D(42) xor D(41) xor D(40) xor D(39) xor D(38) xor D(37) xor D(36) xor D(35) xor D(34) xor D(33) xor D(32) xor D(31) xor D(28) xor D(27) xor D(26) xor D(25) xor D(24) xor D(23) xor D(22) xor D(21) xor D(20) xor D(19) xor D(18) xor D(17) xor D(16) xor D(14) xor D(13) xor D(12) xor D(11) xor D(10) xor D(09) xor D(08) xor D(07) xor D(06) xor D(05) xor D(04) xor D(03) xor D(02) xor D(01) xor C(00) xor C(01) xor C(02) xor C(03) xor C(04) xor C(05) xor C(06) xor C(07) xor C(08) xor C(13) xor C(14) xor C(15);

NewCRC(02) = D(61) xor D(60) xor D(57) xor D(56) xor D(46) xor D(42) xor D(31) xor D(30) xor D(29) xor D(28) xor D(16) xor D(14) xor D(01) xor D(00) xor C(08) xor C(09) xor C(12) xor C(13);

NewCRC(03) = D(62) xor D(61) xor D(58) xor D(57) xor D(47) xor D(43) xor D(32) xor D(31) xor D(30) xor D(29) xor D(17) xor D(15) xor D(02) xor D(01) xor C(09) xor C(10) xor C(13) xor C(14);

NewCRC(04) = D(63) xor D(62) xor D(59) xor D(58) xor D(48) xor D(44) xor D(33) xor D(32) xor D(31) xor D(30) xor D(18) xor D(16) xor D(03) xor D(02) xor C(00) xor C(10) xor C(11) xor C(14) xor C(15);

---

[3] A VHDL component performing the CRC computation can be found in [6].This component can be used "as is" in the FED/DCC firmware. When the trailer is given to the component for CRC computation, as it is containing the final CRC, the CRC field is forced to zero. Then when the event is complete, the current CRC value is substituted in the trailer word before transmission to DAQ.

NewCRC(05) = D(63) xor D(60) xor D(59) xor D(49) xor D(45) xor D(34) xor D(33) xor D(32) xor D(31) xor D(19) xor D(17) xor D(04) xor D(03) xor C(01) xor C(11) xor C(12) xor C(15);

NewCRC(06) = D(61) xor D(60) xor D(50) xor D(46) xor D(35) xor D(34)xor D(33) xor D(32) xor D(20) xor D(18) xor D(05) xor D(04) xor C(02) xor C(12) xor C(13);

NewCRC(07) = D(62) xor D(61) xor D(51) xor D(47) xor D(36) xor D(35)xor D(34) xor D(33) xor D(21) xor D(19) xor D(06) xor D(05) xor C(03) xor C(13) xor C(14);

NewCRC(08) = D(63) xor D(62) xor D(52) xor D(48) xor D(37) xor D(36) xor D(35) xor D(34) xor D(22) xor D(20) xor D(07) xor D(06) xor C(00) xor C(04) xor C(14) xor C(15);

NewCRC(09) = D(63) xor D(53) xor D(49) xor D(38) xor D(37) xor D(36) xor D(35) xor D(23) xor D(21) xor D(08) xor D(07) xor C(01) xor C(05)xor C(15);

NewCRC(10) = D(54) xor D(50) xor D(39) xor D(38) xor D(37) xor D(36) xor D(24) xor D(22) xor D(09) xor D(08) xor C(02) xor C(06);

NewCRC(11) = D(55) xor D(51) xor D(40) xor D(39) xor D(38) xor D(37) xor D(25) xor D(23) xor D(10) xor D(09) xor C(03) xor C(07);

NewCRC(12) = D(56) xor D(52) xor D(41) xor D(40) xor D(39) xor D(38) xor D(26) xor D(24) xor D(11) xor D(10) xor C(04) xor C(08);

NewCRC(13) = D(57) xor D(53) xor D(42) xor D(41) xor D(40) xor D(39) xor D(27) xor D(25) xor D(12) xor D(11) xor C(05) xor C(09);

NewCRC(14) = D(58) xor D(54) xor D(43) xor D(42) xor D(41) xor D(40) xor D(28) xor D(26) xor D(13) xor D(12) xor C(06) xor C(10);

NewCRC(15) = D(63) xor D(62) xor D(61) xor D(60) xor D(59) xor D(54) xor D(53) xor D(52) xor D(51) xor D(50) xor D(49) xor D(48) xor D(47) xor D(46) xor D(45) xor D(44) xor D(42) xor D(40) xor D(39) xor D(38) xor D(37) xor D(36) xor D(35) xor D(34) xor D(33) xor D(32) xor D(31) xor D(30) xor D(29) xor D(26) xor D(25) xor D(24) xor D(23) xor D(22) xor D(21) xor D(20) xor D(19) xor D(18) xor D(17) xor D(16) xor D(15) xor D(14) xor D(12) xor D(11) xor D(10) xor D(09) xor D(08) xor D(07) xor D(06) xor D(05) xor D(04) xor D(03) xor D(02) xor D(01) xor D(00) xor C(00) xor C(01) xor C(02) xor C(03)xor C(04) xor C(05) xor C(06) xor C(11) xor C(12) xor C(13) xor C(14) xor C(15);