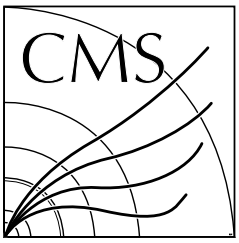


Global Trigger firmware Specification for MP7 platform for Upgrade Phase I

Herbert Bergauer, Babak Rahbaran, Johannes Wittmann
Institute of High Energy Physics (HEPHY)
http://www.hephy.at
http://globaltrigger.hephy.at

November 19, 2021

Revision History

Doc Rev	Description of Change	Revision Date
6.4	Updated data structure for jets with DISP bit (4.2.1).	2021/11/17
6.3	Inserted "Calculation of look-up-tables (LUTs) for correlation cuts" (4.4.7).	2021/11/10
6.2	Removed "VHDL-Templates for VHDL-Producer".	2021/09/24
6.1	Updated (and renamed) description of "Invariant mass over delta R calculation" (see 4.4.6).	2021/09/14
6.0	New structure of document for firmware versions 1.12.x and higher.	2021/02/10
5.7	Fixed typo in section "Invariant mass calculation for three objects" 4.4.6.	2020/12/03
5.6	Updated text in section "VHDL-Templates for VHDL-Producer".	2020/09/31
5.5	Inserted links to VHDL modules.	2020/09/18
5.4	Updated text in section "Correlation conditions" 4.4.13. Description is for v1.10.0 of Global Trigger Logic.	2020/09/17
5.3	Inserted description of "Invariant mass divided by delta R calculation" (see 4.4.6).	2020/09/10
5.2	Fixed typo (unconstrained pt).	2020/09/09
5.1	Inserted text for new muon structure in sections 4.2.2, 4.4.4 and 4.4.13.1. Added subsections in section "VHDL-Templates for VHDL-Producer"	2020/08/04
5.0	Additional text in section for calo calo overlap remover condition module.	2020/05/25
4.9	Inserted text in section Calorimeter Overlap Remover conditions and Calo Calo Overlap Remover Correlation conditions.	2020/04/16
4.8	Updated text in sections Calorimeter conditions, Muon conditions and Correlation conditions for changes which have been done for GTL VHDL version 1.8.0 (module names without version number, "five eta cuts").	2019/08/13
4.7	Inserted "Asymmetry" and "Centrality" of "Energy sums" (GTL VHDL version 1.6.0). Therefore updated sections 4.1, 4.2.1, 4.4.9 added section "Centrality condition" 4.4.12 and updated Table 2	2018/08/13
4.6	Updated text in section "Global Trigger Logic" (4) according to firmware version v1.5.0 of gtl_module.vhd.	2018/02/21
4.5	Updated text in section "Framework" (3) according to firmware version v1.2.3 of frame.vhd.	2018/01/19

Doc Rev	Description of Change	Revision Date
4.4	New "icons" ET^{HF}_{miss} and HT^{HF}_{miss} in Table 2 and Section 4. Updated glossary (see Section **??**).	2016/11/11
4.3	Updated table "μFDL register map" (21) and section "Register map" (5.4.1.1). Moved "List of Tables" and "List of Figures" to the end of document. Inserted link to "Scales for inputs to μGT" (4.3). Moved section "Software reset" to section "Framework" as subsection (3.6). Removed empty sections "IPBus", "Firmware Configuration" and "Bibliography".	2016/11/03
4.2	Updated sections "Calo-Layer2 optical interfaces" (4.2.1) and "Energy sum quantities conditions" (4.4.9) for towercount trigger bits. Inserted section "Towercount condition" (4.4.11).	2016/10/25
4.1	Updated section "Calo-Layer2 optical interfaces" (4.2.1) for new energy sum quantities and minimum bias trigger bits. Updated sections "Firmware" (2), "Framework" (3) and "Final Desicion Logic" (5).	2016/06/09
4.0	Updated Text in section "Muon Muon Correlation condition module".	2016/01/15
3.9	Removed "Double objects requirements condition with spatial correlation", because not used anymore in the future, replaced by Correlation conditions.	2016/01/08
3.8	Minor changes in text and updated Figure 10.	2016/01/08
3.7	Changed colour in Figure 11 and updated text for correlation conditions (see section 4.4.13.	2016/01/07
3.6	Updated Figures 10 and 9 and text in calo calo correlation condition module.	2015/12/21
3.5	Inserted drawing of VHDL structure of cuts for correlation conditions (see Figure 12).	2015/11/18
3.4	Updated muon η ranges (Table 9) and inserted correlation conditions. Created scheme for conversion of calorimeter η and φ to muon scale for calo-muon-correlation conditions.	2015/11/17
3.3	Added Text in sections calo comparator module and muon comparator module.	2015/10/08
3.2	Updated Text in section "Final Desicion Logic" (5).	2015/10/06
3.1	Updated Figure 13 and Tables 21. Remaned section "Calorimeter conditions module - version 2" to "Calorimeter conditions module - version 3", section "Muon conditions module" to "Muon conditions module - version 2" and section "Muon comparators module" to "Muon comparators module - version 2".	2015/10/02

Doc Rev	Description of Change	Revision Date
3.0	Updated text and tables of η ranges for Calorimeter objects (see 4.4.2).	2015/09/22
2.9	Renewed Figures in GTL and FDL (see Figure 8, 9 and 10) and FDL(see Figure 13 and 14). Added register bits description of FDL Register map (see section 5.4.1.1).	2015/09/16
2.8	Updated text, tables and listings of section "VHDL-Templates for VHDL-Producer".	2015/09/15
2.7	Corrected calculation of muon η step width (see 4.4.4).	2015/09/10
2.6	Edited text in Table 16.	2015/08/28
2.5	Updated definition of η ranges for Calorimeter objects and Muon objects.	2015/08/20
2.4	Added section Calo Muon Correlation condition.	2015/08/19
2.3	Added section "Register map" (5.4.1.1) for μFDL.	2015/06/26
2.2	Updated figures (8, 9 and 10) for GTL and edited section "Correlation conditions" (see 4.4.13).	2015/05/08
2.1	Added tables for calorimeter isolation-bits and for muon quality- and isolation-bits definition (8, 11 and 13). Edited section glossary (**??**) and acronyms.	2015/05/07
2.0	Added text for "Energy sum conditions" (4.4.9) and updated chapters for "Calorimeter conditions" for version 2. Inserted isolation bits for electron/γ and tau objects (4.4.2).	2015/05/06
1.9	Minor changes "Demux Lane Data" (see 3.2) and "Muon data" (see 4.4.4).	2014/11/06
1.8	Edited Section "Energy sum quantities conditions" (see 4.4.9).	2014/10/08
1.7	Added sections "Configuration of optical connections" (3.1), "Demux Lane Data" (3.2) and "Lane Mapping Process" (3.3) to framework. Removed tables of optical interfaces from gtl and referenced to tables in framework.	2014/10/07
1.6	Minor changes in "Calorimeter conditions" and "Muon conditions" .	2014/07/01
1.5	Updated with minor changes in "Muon conditions".	2014/06/17
1.4	Fixed bug in Figure 11.	2014/04/30
1.3	Updated section "Muon conditions".	2014/04/22
1.2	Removed section "Muon charge module" and added new section "Muon charge correlation module" (see 4.4.6). Edited text in section and subsections "Muon conditions definition".	2014/04/15
1.1	Changed Figure 11 and minor changes in text for anti-clockwise behaviour in φ.	2014/04/04
1.0	Added definition for "calorimeter conditions over bx", see section.	2014/03/12

Doc Rev	Description of Change	Revision Date
0.9	Changed text of condition description in subsections Calo conditions definition and Muon conditions definition.	2014/02/12
0.8	Updated calorimeter data structure in 4.2.1.	2013/12/03
0.7	Updated muon data structure in 4.2.2	2013/12/02
0.6	Moved decription of VHDL templates for TME to "VHDL-Templates for VHDL-Producer".	2013/11/18
0.5	Subsection 4.2 added to section 4.	2013/11/11
0.4	GTL and FDL firmware implemented for new data structure (GTL firmware version v1.0.0 [fix part of GTL], FDL firmware version v1.0.0)	2013/11/06
0.3	New framework implementation based on new object types definition. Additionally, the ROP is implemented based on production requirements.	2013/10/13
0.2	First framework implementation + ROP .	2012/07/01
0.1	Document created.	2012/02/22

Contents

1 Global Trigger System overview

Entire document is "under construction"!

The Global Trigger System is based on uTCA technology and 10Gbps optical links. A set of 6 MP7 boards with FPGAs of the powerful Xilinx Virtex-7 family is available. The Global Trigger firmware is implemented on these FPGAs. Every FPGA contains a part of the VHDL representation of a L1 Menu, the partitioning is done by VHDL Producer tool. The trigger decision of every MP7 board is collected on an AMC502 board to generate the "final OR" signal which triggers the readout of the detector.

2 Firmware overview

The figure 1 shows the architecture of μGT payload. It consists of framework and the algorithm logic which it consists of the following modules:

1. Global Trigger Logic Data Mapping
2. μGTL
3. μFDL

The output mux (part of framework) collects data for read-out record which are send via MP7 read-out to AMC13.

The IPBus system allows the control of hardware via a 'virtual bus', using a standard IP-over-gigabit-Ethernet network connection.

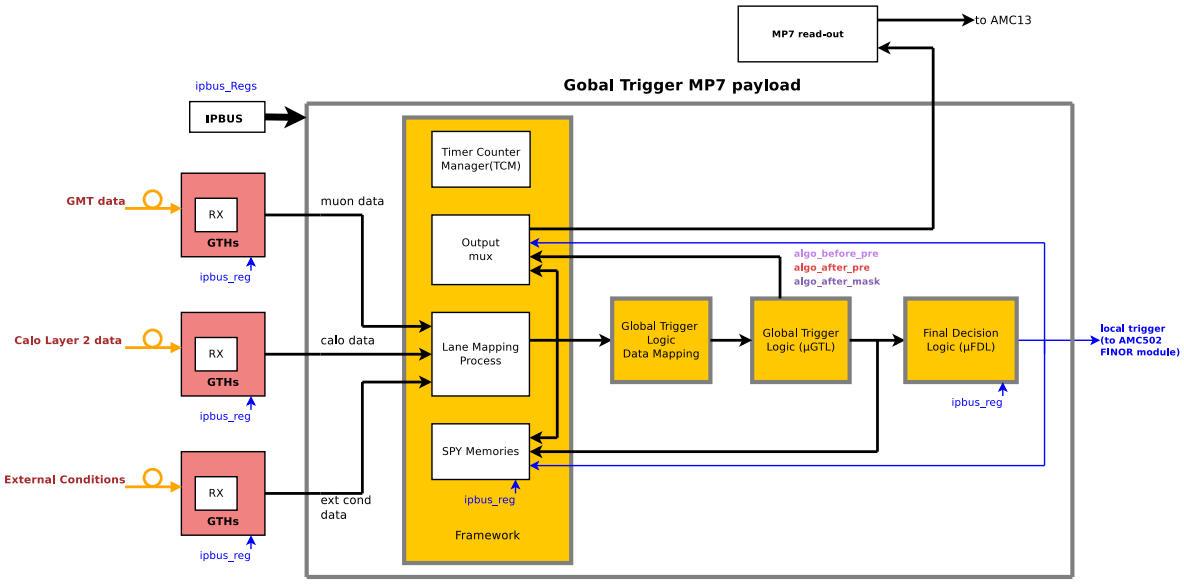


Figure 1: μGT payload

2.1 Firmware version

This firmware description is valid for version 1.17.0 of Global Trigger firmware, containing the following module versions:

- Framework: 1.2.4
- Global Trigger Logic: 1.17.0
- Final Decision Logic: 1.3.6

2.2 Directory structure of Global Trigger firmware

INSERT TEXT !!!

2.2.1 Package: lhc_data_pkg

The VHDL record `lhc_data_t` (shown in Listing 1) is used as a container for all object streams processed by the system. It is declared in the VHDL package `lhc_data_pkg`. For debugging and simulation purposes a second package (`lhc_data_debug_util_pkg`) is created which contains functions to convert the `lhc_data_t` to a hexadecimal string representation and vice versa. The testbench of the design uses this functions to load the contents of the SIM memory from a file.

Listing 1: lhc_data_t record specification

```
type lhc_data_t is record
        muon : muon_array_t;
        eg : eg_array_t;
        tau : tau_array_t;
        jet : jet_array_t;
        ett : std_logic_vector(ETT_DATA_WIDTH-1 downto 0);
        ht : std_logic_vector(HT_DATA_WIDTH-1 downto 0);
        etm : std_logic_vector(ETM_DATA_WIDTH-1 downto 0);
        htm : std_logic_vector(HTM_DATA_WIDTH-1 downto 0);
        etmhf : std_logic_vector(ETMHF_DATA_WIDTH-1 downto 0);
        htmhf : std_logic_vector(HTMHF_DATA_WIDTH-1 downto 0);
        link_11_fr_0_data : std_logic_vector(LINK_11_FR_0_WIDTH-1 downto
            0);
        link_11_fr_1_data : std_logic_vector(LINK_11_FR_1_WIDTH-1 downto
            0);
        link_11_fr_2_data : std_logic_vector(LINK_11_FR_2_WIDTH-1 downto
            0);
        link_11_fr_3_data : std_logic_vector(LINK_11_FR_3_WIDTH-1 downto
            0);
        link_11_fr_4_data : std_logic_vector(LINK_11_FR_4_WIDTH-1 downto
            0);
        link_11_fr_5_data : std_logic_vector(LINK_11_FR_5_WIDTH-1 downto
            0);
        external_conditions : std_logic_vector(
            EXTERNAL_CONDITIONS_DATA_WIDTH-1 downto 0);
end record;
```

3 Framework

Remark:
with frame v1.2.3 "Delay Manager" (dm.vhd) and "Data Source Multiplexer" (dsmux.vhd) are removed because these features were never used in production system, only for tests. Simmem data not useable anymore, because of removed dsmux. The reason of removing is to get more available resources.

Figure 2 shows the basic components the framework together with Readout-Process.

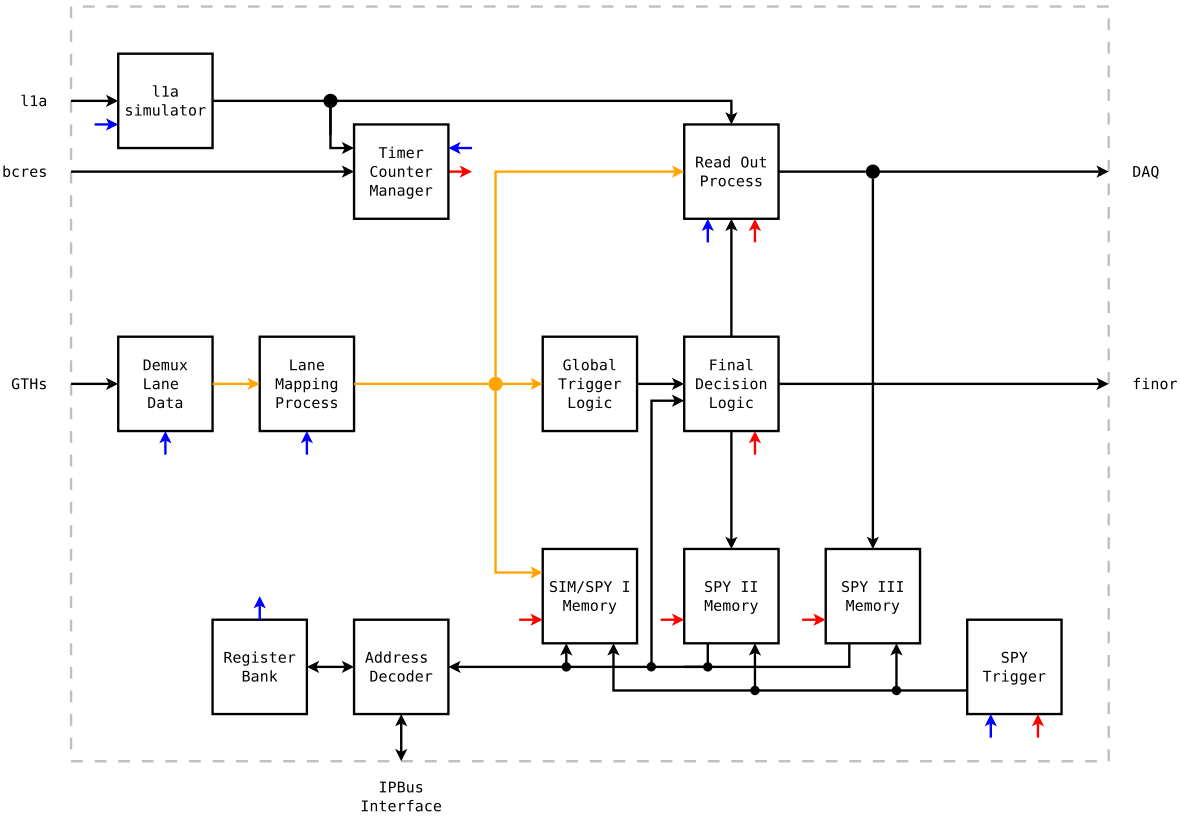


Figure 2: System architecture overview

The central data type of the framework is shown in Listing 1 (see Section 2.2.1 for details). In the current configuration it comprises 2304 bits (288 Bytes). Data from the GTH interfaces is demultiplexed (from 240 MHz clock domain to LHC clock domain, see Demux Lane Data 3.2) and mapped to this data type in the LMP (Lane Mapping Process). It is also used as input and output type for the SIM/SPY I memory.

3.1 Configuration of optical connections

The configuration of the optical connections to Calo-Layer2 is (currently) done as described in Table 2, where frame means the 32 bits data (240 MHz) within a LHC clock period.

3.2 Demux Lane Data

Data from GTH interfaces is in the 240 MHz clock domain. The demultiplexing to the LHC clock domain (about 40 MHz) is done in demux_lane_data.vhd, which is instantiated in frame.vhd as often as lanes are used (currently 16 lanes are used).

3.3 Lane Mapping Process

In the Lane Mapping Process module data from the lanes are mapped to objects structure defined in lhc_data_pkg.vhd.

3.3.1 Implementation

Currently lane mapping is "fixed" in lmp.vhd module, see Table 3

3.4 SIM and SPY Memory

Remark:
with frame v1.2.3 Simmem data not useable anymore, because of removed "Data Source Multiplexer". The reason of removing "Data Source Multiplexer" is to get more available resources.

Under construction!!!

Figure 3 shows the SIM/SPY memory subsystem of the framework. It is used to calibrate the system, i.e. to set the correct delays in the Delay Manager, to record results of the GTL/FDL and output packages of the ROP and to provide simulation data for the system. All source files for the memory subsystem are located in `src/mem` directory.

3.4.1 Implementation

The memory subsystem consists of four main parts, which will be discussed in more detail in the following sections

- SPY Trigger
- SIM/SPY Memory
- SPY Memory II
- SPY Memory III

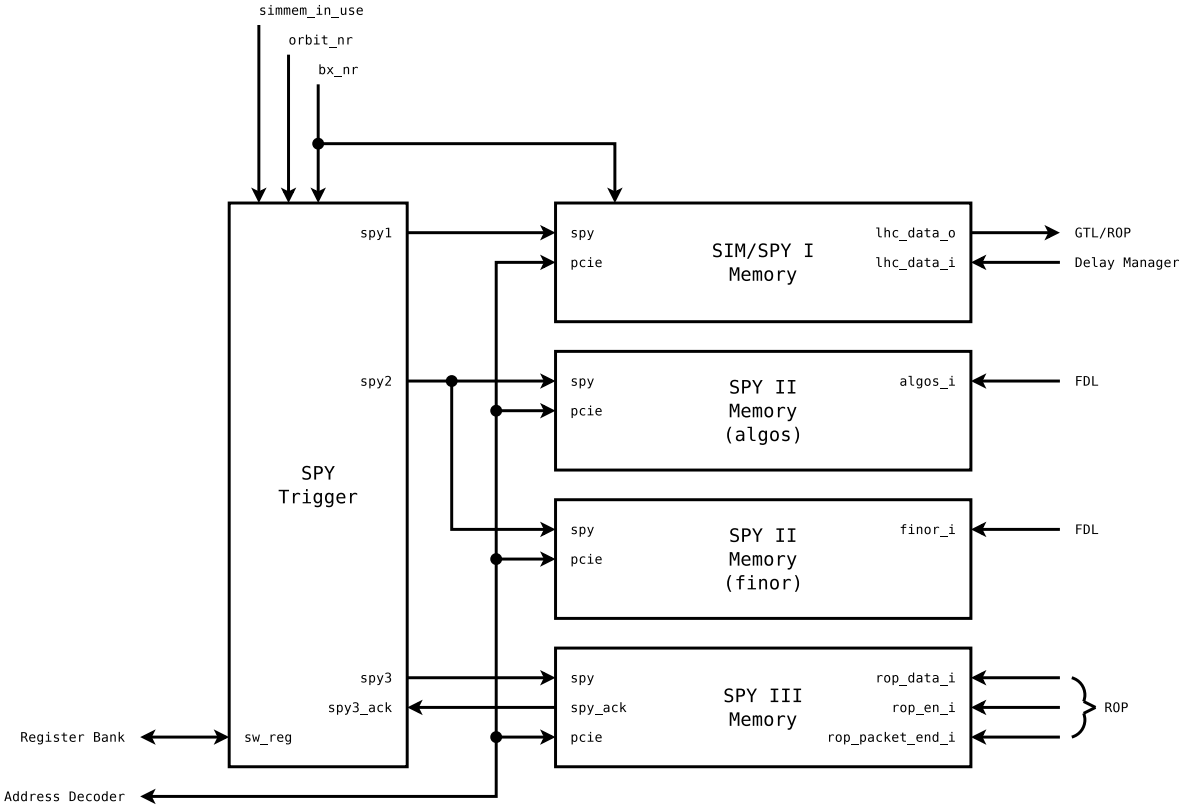


Figure 3: Memory subsystem

3.4.1.1 SPY Trigger

The SPY trigger controls the SPY memories and decides when data is recorded. It can be configured and controlled using software registers 3.1 and 3.2 provided by the register bank.

When the SPY trigger receives a spy12 command (next or once) over the software register interface it asserts the *spy*1 and *spy*2 signals for the appropriate orbit. This means that the *spy* signals go high with the bunch crossing counter reaching the value zero and stay high until it reaches zero again (overflow). Note that when the SIM memory is being used (indicated by the *simem_in_use_i* input provided by the DSMUX component) the *spy*1 output will not be asserted.

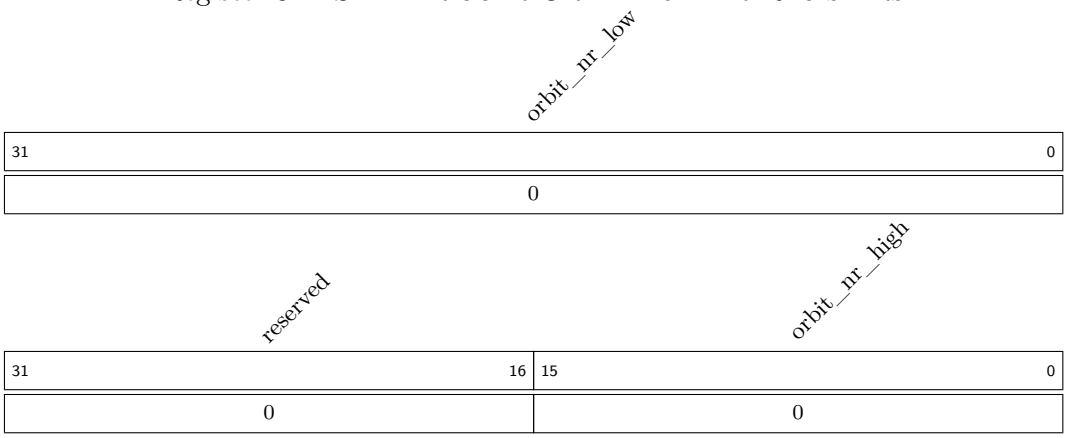
When a spy3 command is received the SPY trigger asserts the *spy*3 signal and waits until the *spy*3*_ack* signal is asserted.

3.4.1.2 SIM/SPY memory

This component combines the SIM memory and the SPY memory I. This optimization is possible because these two memories are never used at the same time. There are basically two use cases for this memory.

- SIM memory: Data is read form the memory and provided to GTL and ROP to test these components.

Register 3.1: SPY Trigger Orbit Number Registers



orbit_nr_low The 32 low bits of the 48 bit orbit number, used for the spy once trigger.

orbit_nr_high The 16 high bits of the 48 bit orbit number, used for the spy once trigger.

- SPY memory: External data is received by the GTX and stored in the memory to check the alignment of the data.

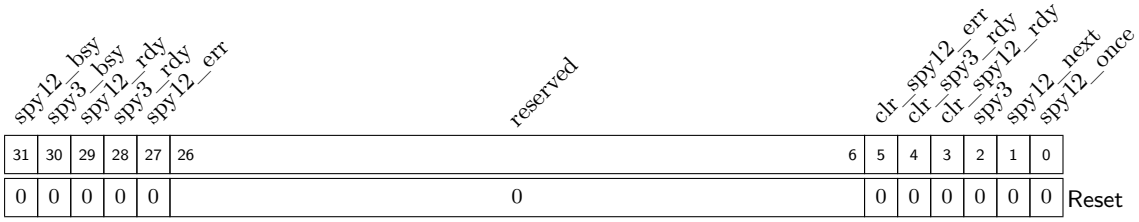
It is very important to guarantee that the spy input signal is not asserted, as long as the memory is used as SIM memory. Note that this functionality is implemented in the SPY trigger component.

The SIM/SPY memory converts the *lhc_data_i* input signal to a `std_logic_vector` using the converter function provided by the `lhc_data_pkg`. This vector is then divided into chunks of 32 bits (the PCIe data width). For each of these chunks a 32 bit true-dual-port memory (2 read ports, 2 write ports, 2 clock domains) is instanciated. Thus, every memory has a read/write port in both clock domain, the 125MHz PCIe clock domain and the 40MHz LHC clock domain, which can be used simultaneously. The PCIe data-in signal (*sw_i.data*) is connected to PCIe-clock domain write port of the memories. A memory select signal is generated form the LSBs of the software address (*sw_i.addr*). The memory select signal also controls the multiplexer on the output of the memories to generate the *sw_o.data* signal.

Depending on whether the SIM/SPY memory is used to provide simulation data or to store/spy data the address on the LHC-clock domain port of the internal memories is adjusted. If data is recorded (SPY) the bunch crossing counter is used as memory address directly. When the memory is read the read latency (two clock cycles) must be taken into account. This is achieved by subtracting 2 form the bunch crossing number before using it as address. To generate the *lhc_data_o* signal the LHC-clock domain data out ports of the internal memories are concatenated and converted back to the `lhc_data_t`.

If the `lhc_data_t` is changed (e.g. new objects added) no modifications in the SIM/SPY memory are required. The SIM/SPY memory only depends on the (auto-generated) functions

Register 3.2: SPY Trigger Configuration Register



spy12_once Triggers the recording of the selected orbit to SPY memories I and II, when written with 1.

spy12_next Triggers the recording of the next whole orbit to SPY memories I and II, when written with 1.

spy3 Triggers the recording of the next package that will be sent by the ROP to SPY memory III, when written with 1.

clr_spy12_rdy Clears the ready flag of the SPY trigger for SPY memories I and II, when written with 1.

clr_spy3_rdy Clears the ready flag of the SPY trigger for SPY memory III, when written with 1.

clr_spy12_err Clears the error flag, when written with 1.

spy12_bsy Indicates that the SPY trigger for SPY memories I and II is busy.

spy3_bsy Indicates that the SPY trigger for SPY memory III is busy.

spy12_rdy Indicates that one orbit has been recorded in SPY memories I and II and that the SPY trigger is ready for new commands.

spy3_rdy Indicates that packet has been recorded in SPY memory III and that the SPY trigger is ready for new commands.

spy12_err Indicates an error condition (Set only when the selected orbit number for the spy once trigger lies in the past and can therefore not be recorded).

used to convert a `lhc_data_t` signal to `std_logic_vector` and vice versa (see Section 2.2.1 for details).

In the current implementation the size every object in the `lhc_data_t` is a multiple of 32 bit. This is also expected by the SIM/SPY memory. If objects with 16 bit sizes are added the SIM/SPY memory must be modified to support this situation (e.g. zero pad the `lhc_data_t`). Furthermore take into account that the PCIe memory bus is 32 bits wide. So 16 bit objects should be added to the end of the `lhc_data_t` (as last entry) to keep software memory access simple.

3.4.1.3 SPY memory II

The SPY memory II is divided into two subcomponents, to store the *algos* and *finor* outputs of the FDL. Both memory can only be read over the SW interface. A write access has no effect. The algos memory uses the same architecture as the SIM memory. The finor memory uses a true-dual-port memory with asymmetric ports. This memory can be written with a data width of one bit and read with a data width of 32 bit.

3.4.1.4 SPY memory III

The SPY memory III stores the output of the ROP, which is sent to the DAQ. The input data width is configurable to bus widths of 16, 32 or 64 bits. Depending on the input data width the memory uses different architectures.

- 16 Bit
  A true-dual-port memory with asymmetric ports (16 and 32 bits) is used.
- 32 Bit
  A true-dual-port memory with 32 Bit data width is used.
- 64 Bit
  Two true-dual-port memories with 32 Bit data width are used.

3.4.2 Interface Specification

Listing 2: SPY trigger interface specification

```
entity spytrig is
        port
        (
                lhc_clk     : in  std_logic;
                lhc_rst     : in  std_logic;
                orbit_nr    : in  orbit_nr_t;
                bx_nr       : in bx_nr_t;
                sw_reg_i    : in  sw_reg_spytrigger_in_t;
                sw_reg_o    : out sw_reg_spytrigger_out_t;

                spy1_o      : out std_logic;
                spy2_o      : out std_logic;
                spy3_o      : out std_logic;
                spy3_ack_i  : in  std_logic;

                simmem_in_use_i : in std_logic
        );
end;
```

3.5 TCM

Under construction!!!

The Timer Counter Manager (TCM) provides different counters, listed in table 4.

3.5.1 Counter Overview

Table 4: counters of the timer counter manager

Counter	range	increase condition	reset condition	Comments
bx_nr	$0 to 3563$	rising_edge(lhc_clk)	overflow	
event_nr	$0 to 2^{32}-1$	l1a=1 and rising_edge(lhc_clk)	BGOS: event counter reset	
trigger_nr	$0 to 2^{48}-1$	l1a=1 and rising_edge(lhc_clk)	BGOS: start run	
orbit_nr	$0 to 2^{48}-1$	overflow of bx_nr	BGOS: orbit counter reset	
luminosity_seg_nr	$0 to 2^{32}-1$	rising_edge(orbit_nr(18))	BGOS: orbit counter reset	
start_lumisection	$0 to 1$	luminosity_seg_nr increases	after 25ns	'1' for 25ns
bx_nr_d_fdl	$0 to 3563$	rising_edge(lhc_clk)	overflow	

3.5.2 Bunch Crossing Number and counters derived from it

All counters except for event_nr and the trigger_nr (which are trivial because they are increased with l1a) are dependent on the bunch crossing counter bx_nr as stated in table 4. The bx_nr is zero at startup, then waits for the the first bcres_d (bunch crossing reset delayed) and starts counting as depicted in figure 4. It's maximal value is 3563 (0xdeb), then it automatically overflows and starts at zero again (see figure 5). Exactly when bx_nr = 0,

bcres_d has to be asserted. Otherwise the counter is out of synchronization. If this happens, the software register err_det is set and the counter waits for the next bcres_d to synchronize again. Note that the value of the counter is invalid until it has synchronized again.

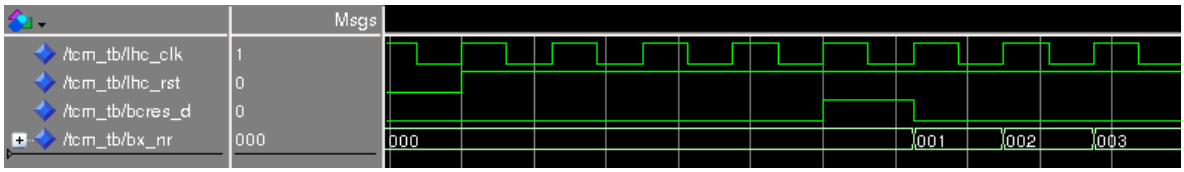


Figure 4: start of the bunch crossing number with the first bcres_d

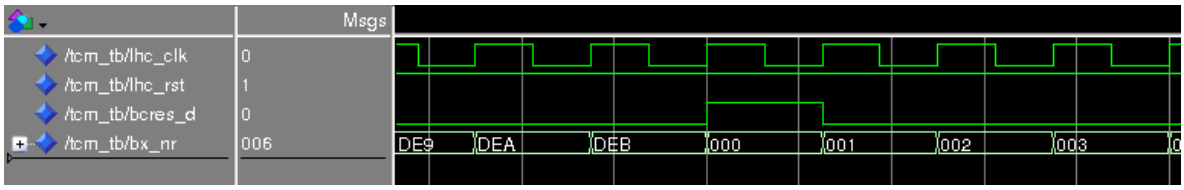


Figure 5: normal operation of the bunch crossing number

3.5.3 Special counter: bx_nr_d_fdl

The bx_nr_d_fdl is derived from bcres_d_fdl in the same manner as bx_nr is derived from bcres_d. bx_nr_d_fdl will automatically resync if the logic described in subsection 3.5.5 detects a synchronization error for bx_nr.

3.5.4 Counters derived from l1a

The counters event_nr and trigger_nr are increased with l1a, i.e. they are increased twice if l1a is high for 2 clock cycles, etc. They differ only in their value range and the condition that resets the counters, see table 4.

3.5.5 Errors

As stated above, bcres_d has to be asserted exactly when bx_nr = 0, otherwise the counter is out of sync. Then the software register err_det is set as depicted in figure 6. err_det can be reset via the software event register err_det_reset_event as depicted in figure 7. Furthermore err_det is set if bgos = Resync-0x1 and the counter value is not 3563.

The TCM implements two additional counters (bx_nr_chk and bx_nr_max) for debugging purposes. These counters are not visible by any other module but readable via software. bx_nr_chk is a 32bit Counter that increases with every LHC clock cycle and resets with bcres_d. bx_r_max holds the highest value bx_nr_chk ever reached (should be 3563 if the link is aligned).

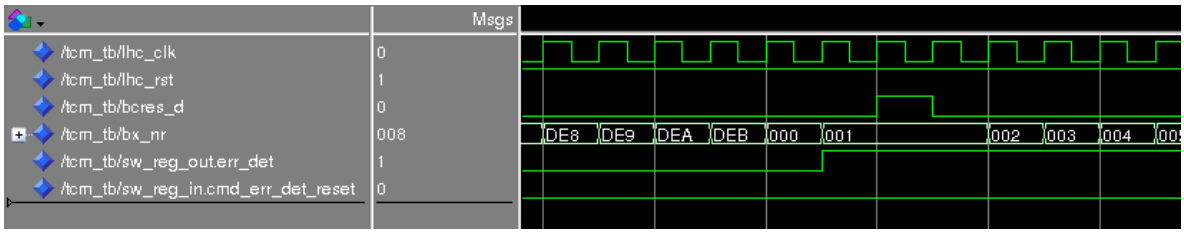


Figure 6: set of the software register err_det when bc_res_d is not asserted correctly

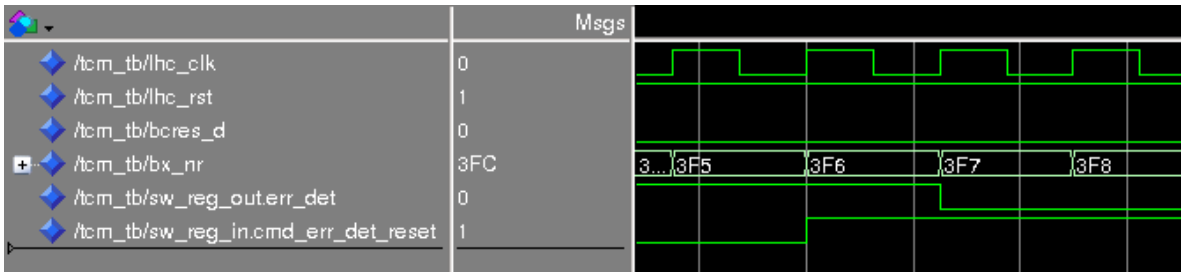
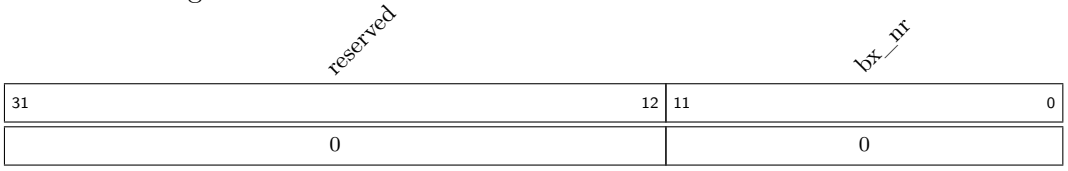


Figure 7: reset of the software register err_det when err_det_reset_event toggles

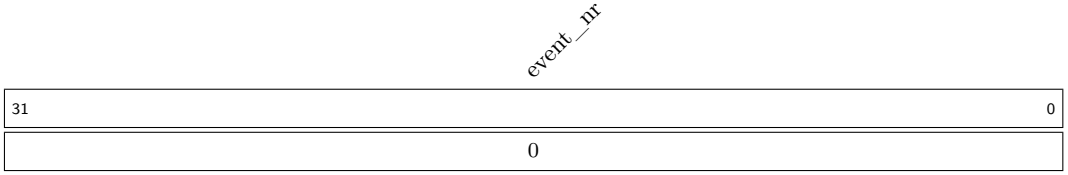
3.5.6 SW-Registers

All counters except for the start_lumisection described in table 4 can be read by software via the following sw registers:

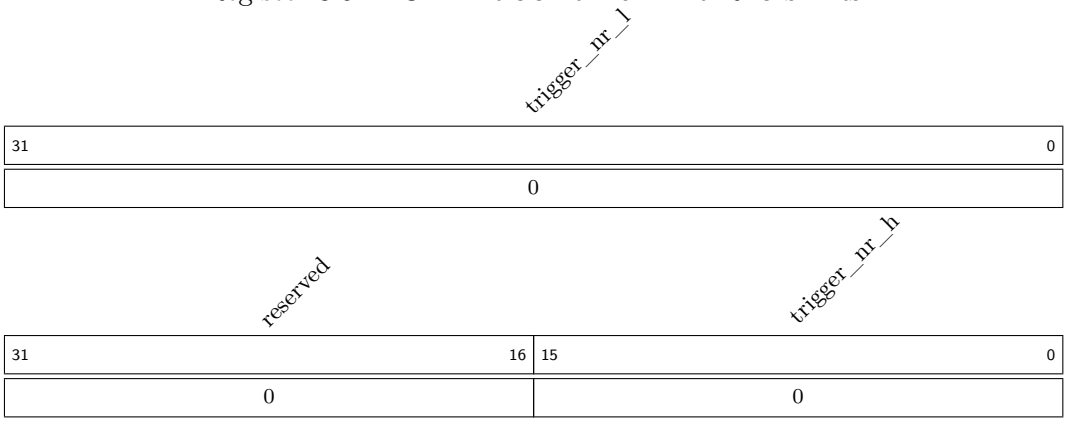
Register 3.3: TCM Bunch Crossing Number Register



Register 3.4: TCM Event Number Register



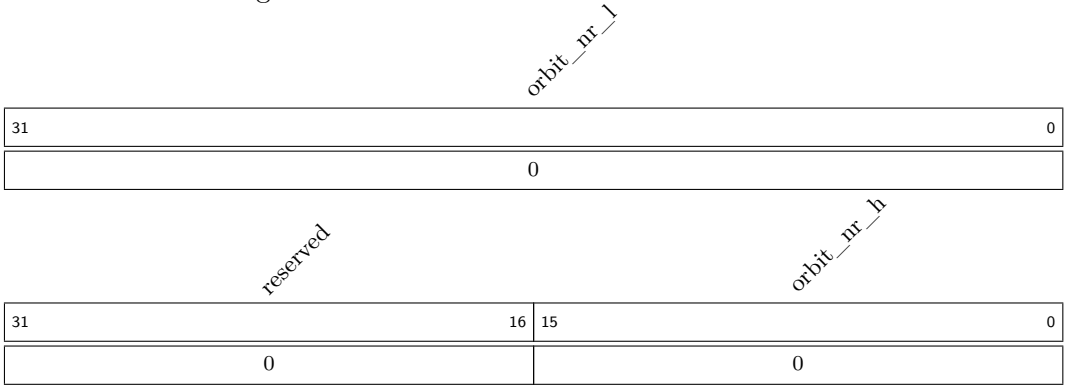
Register 3.5: TCM Trigger Number Registers



trigger_nr_l The 32 low bits of the 48 bit trigger number.

trigger_nr_h The 16 high bits of the 48 bit trigger number.

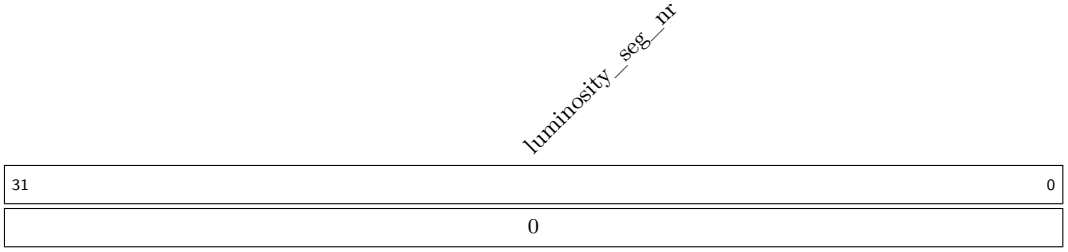
Register 3.6: TCM Orbit Number Registers



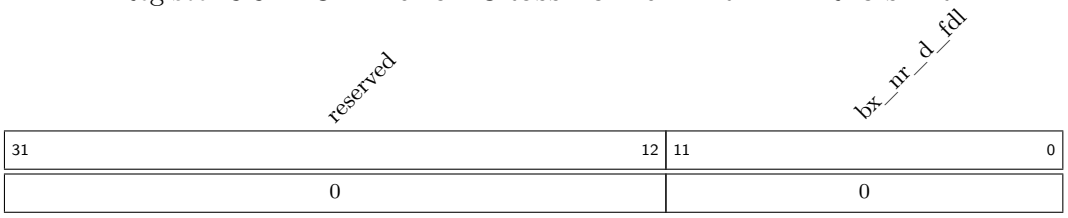
orbit_nr_l The 32 low bits of the 48 bit orbit number.

orbit_nr_h The 16 high bits of the 48 bit orbit number.

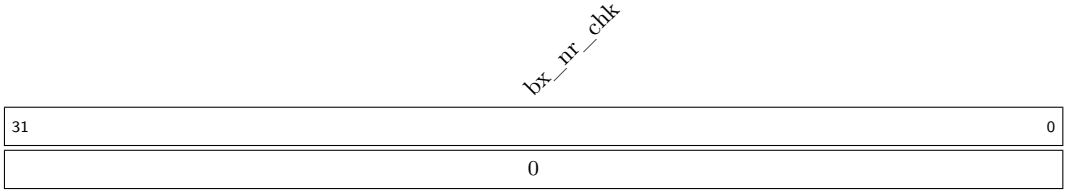
Register 3.7: TCM Luminosity Segment Number Register



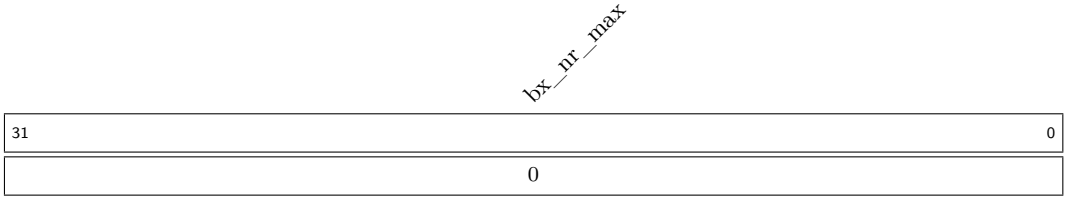
Register 3.8: TCM Bunch Crossing Number FDL Register



Register 3.9: TCM Bunch Crossing Number Check Register

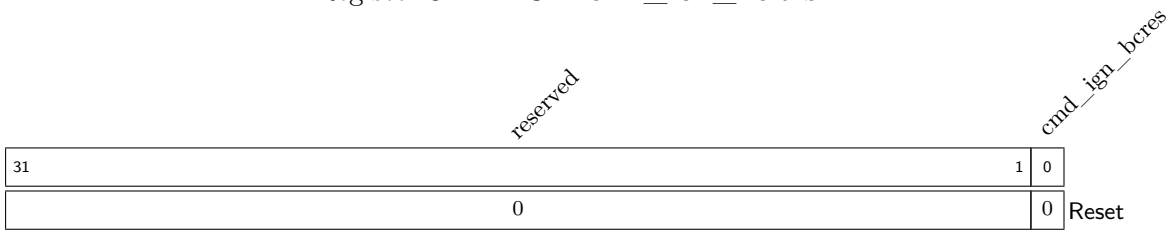


Register 3.10: TCM Bunch Crossing Number Max Register



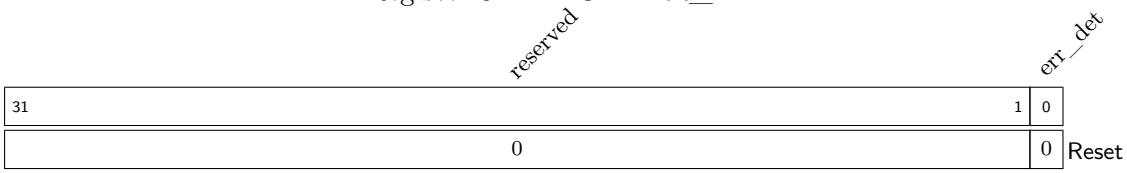
Some additional control register can be used to check and reset err_det, disable the check of bcres_d and bcres_d_fdl (bx_nr and bx_nr_d_fdl automatically reset when they overflow if cmd_ign_bcres is set, bcres_d is ignored) and simulate the bgos signal. To do this, a value of the orbit signal has to be written to sw-register bgos. The value of the input signal bgos is replaced by the value of the sw-register for exactly one clock cycle, when "1" is written to the event register bgos_event.

Register 3.11: TCM cmd_ign_bcres



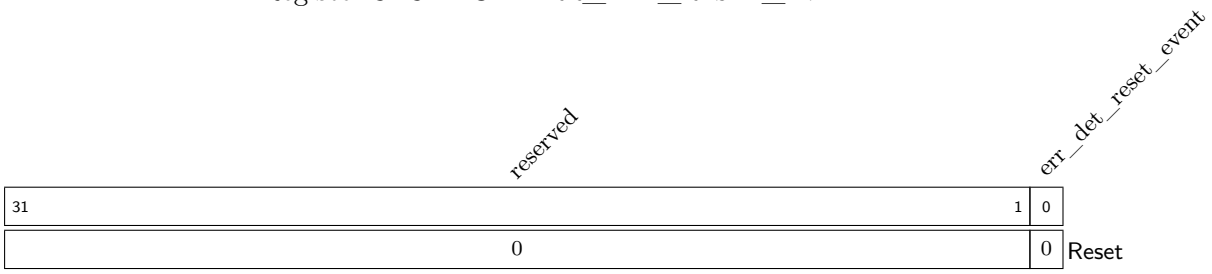
cmd_ign_bcres bcres is ignored (not checked) when this is set.

Register 3.12: TCM ERR_DET



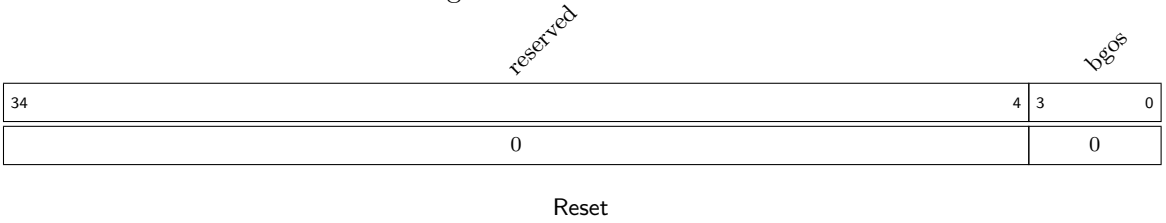
err_det Set when out of synchronization.

Register 3.13: TCM ERR_DET_RESET_EVENT



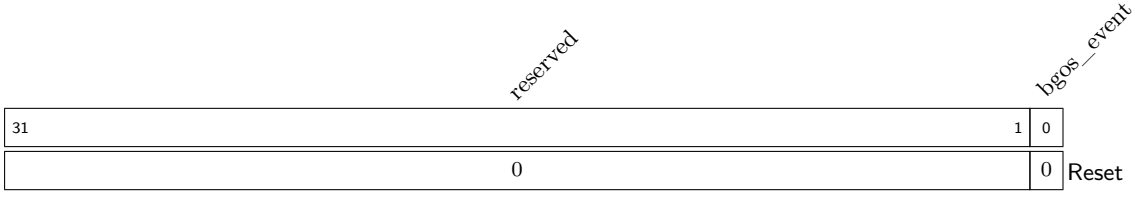
err_det_reset_event Event register: resets err_det.

Register 3.14: TCM BGOS



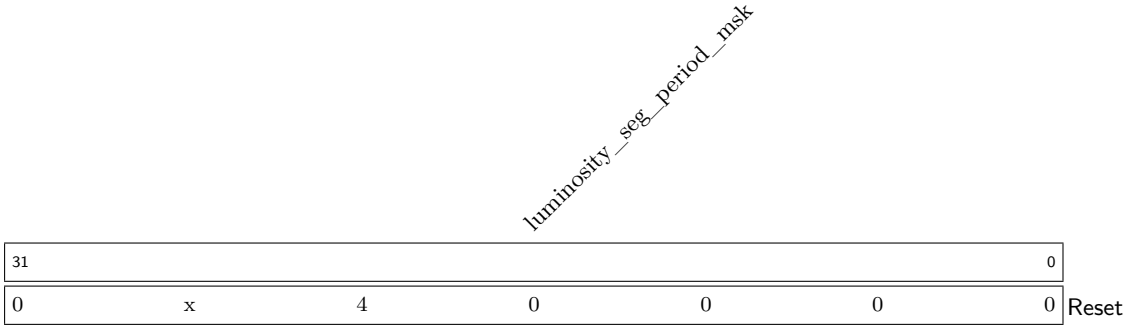
bgos For simulation of the bgos signal.

Register 3.15: TCM BGOS_EVENT



bgos_event Event register: replaces the input signal bgos by the sw-register bgos for exactly one clock cycle.

Register 3.16: TCM LUMINOSITY_SEG_PERIOD_MSK



luminosity_seg_period_msk luminosity_seg_nr is increased when the orbit_nr mod lum_seg_period_mask = 0.

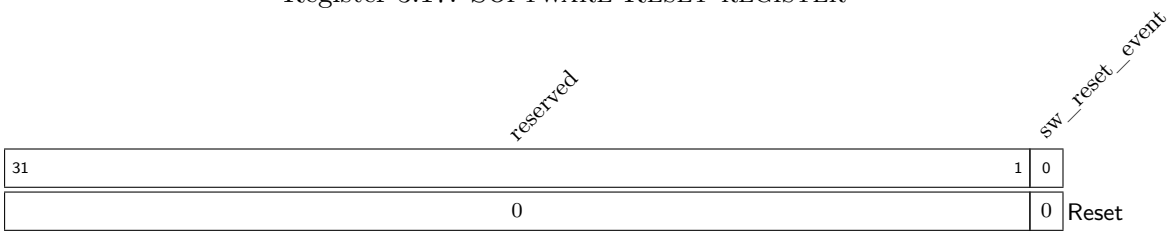
3.5.7 Hardware Test

There are various python scripts located in the software/GtControl/branches/fpga-design-2013/python/GtControl directory for testing the tcm module. Please refer to the output of the scripts for information how the tests are performed in detail. See table 5.

3.6 Software Reset

The software reset module (sw_reset) provides the possiblity for a software reset via the software event register sw_reset_event.

Register 3.17: SOFTWARE RESET REGISTER



sw_reset_event Event register: Generates a reset signal for exactly one clock cycle.

Table 2: Configuration of optical connections

	frame					
link	**0**	**1**	**2**	**3**	**4**	**5**
0	reserved	reserved	muon obj. 0 [0..31]	muon obj. 0 [32..63]	muon obj. 1 [0..31]	muon obj. 1 [32..63]
1	reserved	reserved	muon obj. 2 [0..31]	muon obj. 2 [32..63]	muon obj. 3 [0..31]	muon obj. 3 [32..63]
2	reserved	reserved	muon obj. 4 [0..31]	muon obj. 4 [32..63]	muon obj. 5 [0..31]	muon obj. 5 [32..63]
3	reserved	reserved	muon obj. 6 [0..31]	muon obj. 6 [32..63]	muon obj. 7 [0..31]	muon obj. 7 [32..63]
4	electron/γ obj. 0	electron/γ obj. 1	electron/γ obj. 2	electron/γ obj. 3	electron/γ obj. 4	electron/γ obj. 5
5	electron/γ obj. 6	electron/γ obj. 7	electron/γ obj. 8	electron/γ obj. 9	electron/γ obj. 10	electron/γ obj. 11
6	jet obj. 0	jet obj. 1	jet obj. 2	jet obj. 3	jet obj. 4	jet obj. 5
7	jet obj. 6	jet obj. 7	jet obj. 8	jet obj. 9	jet obj. 10	jet obj. 11
8	tau obj. 0	tau obj. 1	tau obj. 2	tau obj. 3	tau obj. 4	tau obj. 5
9	tau obj. 6	tau obj. 7	tau obj. 8	tau obj. 9	tau obj. 10	tau obj. 11
10	ET ETTEM MBT0HFP	HT TOWER-COUNT MBT0HFM	ET_{miss} ASYMET MBT1HFP	HT_{miss} ASYMHT MBT1HFM	ET^{HF}_{miss} ASYM-ETHF CENT[3:0]	HT^{HF}_{miss} ASYM-HTHF CENT[7:4]
11	free	free	free	free	free	free
12	external-conditions [0..31]	external-conditions [32..63]	reserved	reserved	reserved	reserved
13	external-conditions [64..95]	external-conditions [96..127]	reserved	reserved	reserved	reserved
14	external-conditions [128..159]	external-conditions [160..191]	reserved	reserved	reserved	reserved
15	external-conditions [192..223]	external-conditions [224..255]	reserved	reserved	reserved	reserved

Table 3: Current lane mapping

lane	**objects**
0	muon objects 0..1
1	muon objects 2..3
2	muon objects 4..5
3	muon objects 6..7
4	electron/γ objects 0..5
5	electron/γ objects 6..11
6	jet object 0..5
7	jet object 6..11
8	tau object 0..5
9	tau object 6..11
10	energy sum quantities (incl. minimum bias trigger bits and towercounts)
11	n/a (currently not used)
12	external-conditions [0..63]
13	external-conditions [64..127]
14	external-conditions [128..191]
15	external-conditions [192..255]

Table 5: scripts for testing the tcm

script	purpose
`tcm_counter_values.py`	outputs the values of all counters defined above
`tcm_produce_err_det`	produces an err_det by manipulating bgos
`tcm_err_det_reset`	resets the err_det software register
`tcm_trigger_test`	tests trigger_nr and event_nr by generating l1a signals using l1asim
`tcm_lum_seg_nr_test`	checks the period of two successive increases of the luminositiy_seg_nr

4 Global Trigger Logic

SECTION STILL UNDER CONSTRUCTION

Remark:
This description is for version 1.17.0 of Global Trigger Logic.

The Global Trigger Logic (μGTL) firmware contains conditions and algorithms for trigger decision.

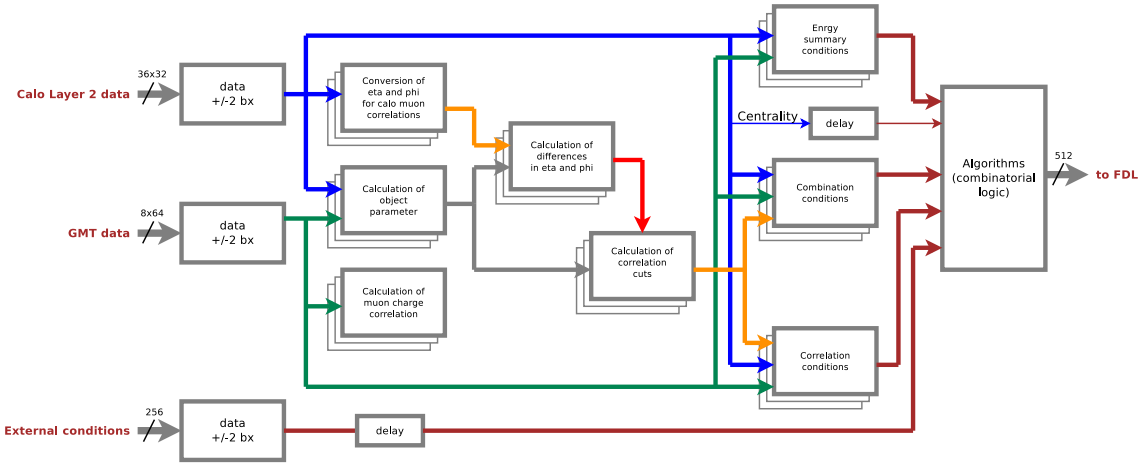


Figure 8: μGTL firmware

4.1 μGTL Interface

Inputs:

- Calo-Layer2 data
  - Electron/γ objects
  - Jet objects
  - Tau objects
  - Energy summary information:
    - Total Et (ET)
    - total Et from ECAL only (ETTEM)
    - total calibrated Et in jets (HT)
    - missing Et (ET_{miss})
    - missing Et including HF (ET^{HF}_{miss})
    - missing Ht objects (HT_{miss})
    - missing Ht including HF (HT^{HF}_{miss})

    - "Asymmetry" information (ASYMET, ASYMHT, ASYMETHF, ASYMHTHF)
  - Minimum bias HF bits (included in energy summary information data structure)
  - Towercount bits (number of firing HCAL towers, included in energy summary information data structure)
  - "Centrality" bits
- Global Muon Trigger data
- External conditions

Outputs:

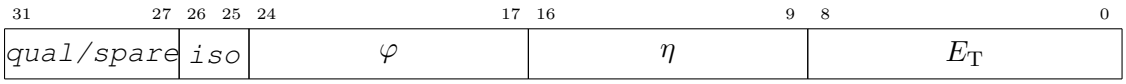
- Algorithms

4.2 Definition of optical interfaces

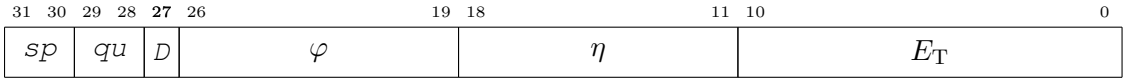
Remark:
All definitions for scales in the following chapters are from a CMS Detector Note: "Scales for inputs to μGT" (see current version in https://raw.githubusercontent.com/cms-l1-globaltrigger/mp7_ugt_legacy/master/doc/scales_inputs_2_ugt/pdf/scales_inputs_2_ugt.pdf).

4.2.1 Calo-Layer2 optical interfaces

The data structure of an electron/γ object (bits 27..31 are not defined yet, reserved for quality, ...):

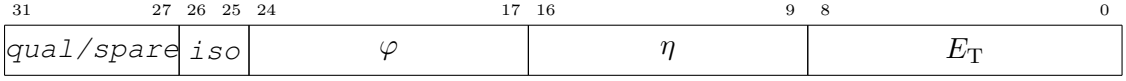


The data structure of a jet object (bits 30..31 are spare bits):

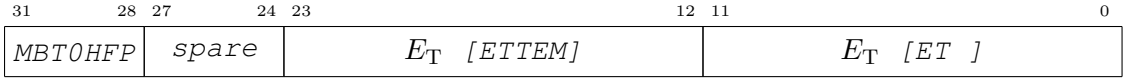


D = DISP bit, qu = quality flags, sp = spare bits.

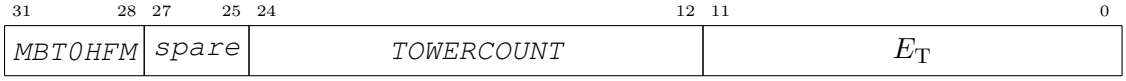
The data structure of a tau object (bits 27..31 are not defined yet, reserved for quality, ...):



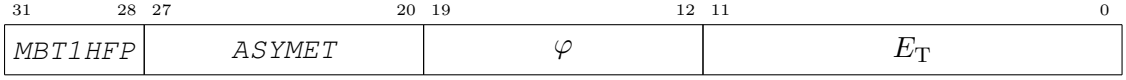
The data structure of "total Et" (ET) quantity [including "total Et from ECAL only" (ETTEM) and "minimum bias HF+ threshold 0" bits]:



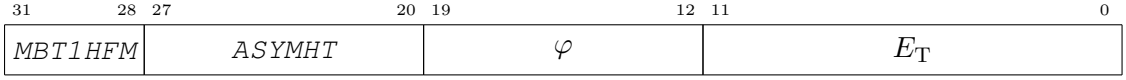
The data structure of "total calibrated Et in jets" (HT) quantity [including "towercount" and "minimum bias HF- threshold 0" bits]:



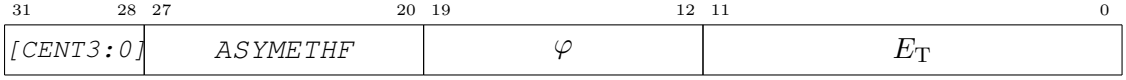
The data structure of "missing Et" (ET_{miss}) quantity [including "Asymmetry" ASYMET and "minimum bias HF+ threshold 1" bits]:



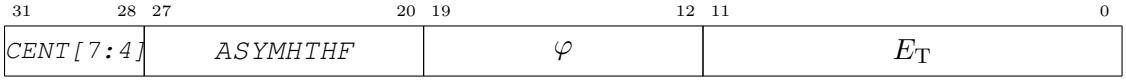
The data structure of "missing Ht" (HT_{miss}) quantity [including "Asymmetry" ASYMHT and "minimum bias HF- threshold 1" bits]:



The data structure of "missing Et including HF" (ET^{HF}_{miss}) quantity [including "Asymmetry" ASYMETHF and "Centrality" bits (3:0)]:

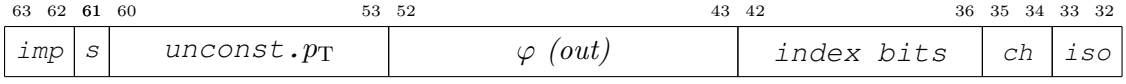


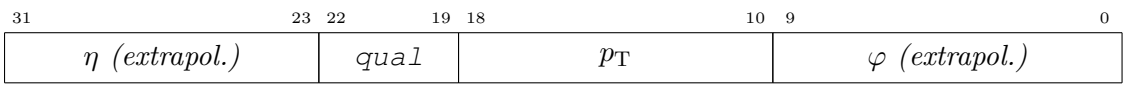
The data structure of "missing Ht including HF" (HT^{HF}_{miss}) quantity [including "Asymmetry" ASYMHTHF and "Centrality" bits (7:4)]:



4.2.2 Global Muon Trigger optical interfaces

The data structure of a muon object (64 bits - bit 34 = charge sign, bit 35 = charge valid, bit 61 is a spare bit, bit 63..62 = impact parameter):





ch = charge bits, s = spare bit, imp = impact parameter.

4.3 Implementation in firmware

The firmware of μGTL consists of two main parts:

- A top-of-hierarchy file (`gtl_module.vhd`), which contains the pipeline for ±2bx data, the instantiations of calculators for differences in η and φ, the instantiations of conditions, the instantiations of charge correlation logic of muons and the Algorithms logic for 512 Algorithms, as well as a package file (`gtl_pkg.vhd`) for declarations. Actually 6 AMC board are used to contain 512 Algorithms. Therefore the 512 Algorithms are **partitioned by VHDL Producer**. The VHDL Producer for every Trigger Menu creates VHDL snippets files (`algo_index.vhd`, `gtl_module_instances.vhd`, `gtl_module_signals.vhd`, `ugt_constants.vhd`), these snippets are inserted into templates for `gtl_module.vhd` (`gtl_module_tpl.vhd`) and `gtl_pkg.vhd` (`gtl_pkg_tpl.vhd`) during simulation and synthesis.
- A set of VHDL-files exists for all the modules instantiated in top-of-hierarchy and the modules in the hierarchy. These files, called the "fixed part", are not influenced by VHDL Producer.

The latency of μGTL is fixed to 5 bunch-crossings, 2 bunch-crossings for the pipeline of ±2bx data (for data with +2bx and +1bx), 2 bunch-crossings for conditions (fixed), also for the conditions requested in the future, 1 bunch-crossing for the logic of Algorithms (See Figure 10).

4.3.1 Top-of-hierarchy module

The top-of-hierarchy module (`gtl_module.vhd`) contains

- the pipeline for ±2bx data
- the instantiations of charge correlation logic of muons (generated by VHDL Producer)
- the instantiations of calculators for differences in η and φ (generated by VHDL Producer)
- the instantiations of conditions (generated by VHDL Producer)
- a boolean logic for Algorithms (generated by VHDL Producer)

Listing 3 contains the entity-declaration of the top-of-hierarchy file (`gtl_module.vhd`).

Listing 3: Entity declaration of `gtl_module.vhd`

```
entity gtl_module is
    port(
        lhc_clk : in std_logic;
        eg_data : in calo_objects_array(0 to NR_EG_OBJECTS-1);
        jet_data : in calo_objects_array(0 to NR_JET_OBJECTS-1);
        tau_data : in calo_objects_array(0 to NR_TAU_OBJECTS-1);
```

```
        ett_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        ht_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        etm_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        htm_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
--
    ****************************************************************************************

-- HB 2016-04-18: updates for "min bias trigger" objects (quantities) for Low-
    pileup-run May 2016
        mbt1hfp_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        mbt1hfm_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        mbt0hfp_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        mbt0hfm_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
-- HB 2016-06-07: inserted new esums quantities (ETTEM and ETMHF).
        ettem_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        etmhf_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
-- HB 2016-09-16: inserted HTMHF and TOWERCNT
        htmhf_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        towercount_data : in std_logic_vector(MAX_TOWERCOUNT_BITS-1 downto 0);
-- HB 2018-08-06: inserted signals for "Asymmetry" and "Centrality" (included in
    esums data structure).
        asymet_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        asymht_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        asymethf_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        asymhthf_data : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        centrality_data : in std_logic_vector(NR_CENTRALITY_BITS-1 downto 0);
--
    ****************************************************************************************

        muon_data : in muon_objects_array(0 to NR_MUON_OBJECTS-1);
        external_conditions : in std_logic_vector(NR_EXTERNAL_CONDITIONS-1 downto
            0);
        algo_o : out std_logic_vector(NR_ALGOS-1 downto 0));
end gtl_module;
```

All the declarations for arrays ('type'), parameters ('constant') and look-up-tables ('constant') used in modules are available in `gtl_pkg.vhd` package-file.

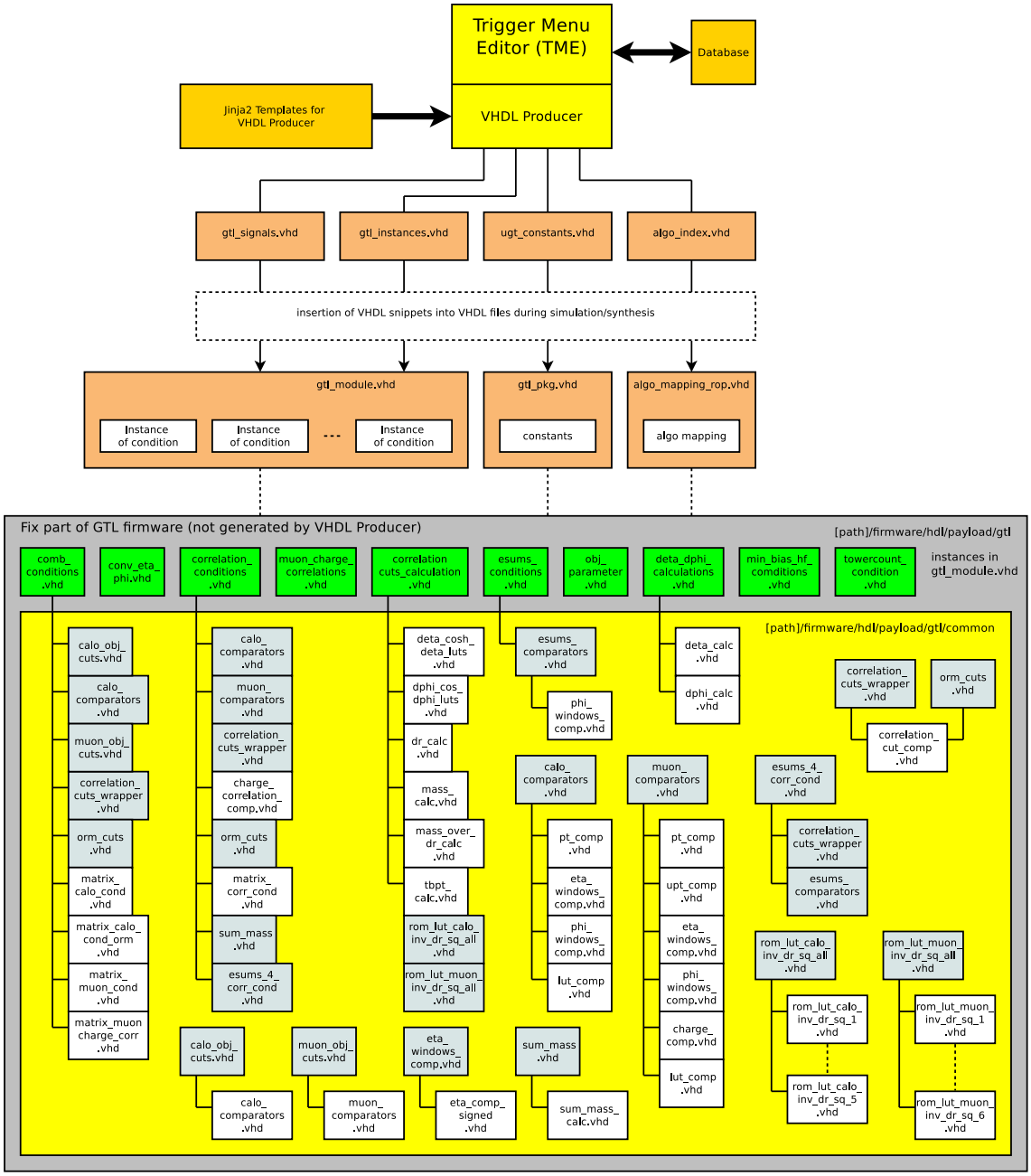


Figure 9: VHDL file generation by VHDL Producer

4.4 μGTL structure

4.4.1 Data ±2bx

The μGTL input data flow through a register pipeline of four stages. With those data it is possible to have conditions with objects from different bunch-crossings (within ±2 bunch-crossings), electron/γ for Correlation conditions.
See Figure 10 for a scheme of μGTL pipeline structure. The data "data_p_1bx" and "data_p_2bx" occur 1 respectively 2 bunch-crossings after data for a certain bunch-crossing, therefore we got 2 bunch-crossings of latency from those data. The data "data_m_1bx" and "data_m_2bx" have no influence on latency, because coming before data for a certain bunch-crossing.

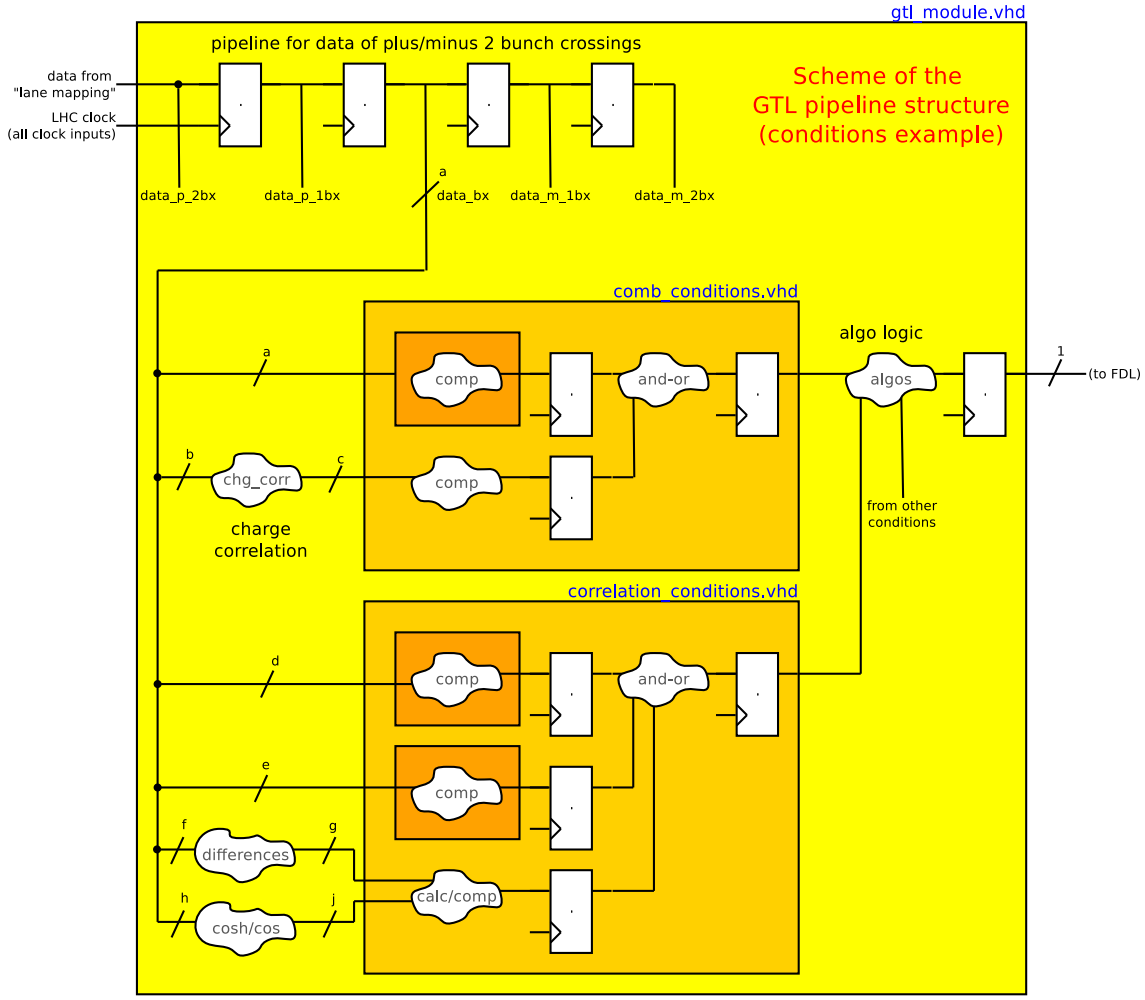


Figure 10: Scheme of μGTL pipeline structure

4.4.2 Definitions of Calorimeter data

The calorimeter trigger processing identifies **electron/γ, jet and tau** objects and **energy sum quantities**.
See also 4.2.

electron/γ:
Twelve objects are passed to the μGT for each event.
For each selected object, the Calo-Layer2 sends parameters for p_{T} and for position and isolation - encoded in 32 bits:

- 9 bits p_{T}, range = 0..255 GeV (HW index = 0..0x1FF), step = 0.5, the highest bin will mark an overflow (HW index 0x1FF): meaning has to be defined
- 8 (7+1 sign) bits pseudo-rapidity (η) position, range = -5.0 to 5.0, step = 0.087/2, linear scale, 138 bins (HW index = 0xBC..0x44)
- 8 bits azimuth angle (φ) position, range = 2π, step $\approx 2\pi/144$ ($\triangleq$2.5°), 144 bins (HW index = 0..0x8F), HW index starting at 0° (anti-clockwise)
- 2 bits isolation
- 5 bits spare

jet:
Twelve objects are passed to the μGT for each event.
For each selected object, the Calo-Layer2 sends parameters: p_{T}, for position information, a DISP bit and quality information - encoded in 32 bits:

- 11 bits p_{T}, range = 0..1023 GeV (HW index = 0..0x7FF), step = 0.5, the highest bin will mark an overflow (HW index 0x7FF): meaning has to be defined
- 8 (7+1 sign) bits pseudo-rapidity (η) position, range = -5.0 to 5.0, step = 0.087/2, linear scale, 230 bins (HW index = 0x8E..0x72)
- 8 bits azimuth angle (φ) position, range = 2π, step $\approx 2\pi/144$ ($\triangleq$2.5°), 144 bins (HW index = 0..0x8F), HW index starting at 0° (anti-clockwise)
- 1 DISP bit (will be used to flag a jet as delayed / displaced based on HCAL timing and depth profiles that are indicative of a LLP decay. If this bit is set to 1, then the jet has been tagged as an LLP jet.)
- 2 bits for "quality flags" - currently not used.
- 2 bits spare

tau:
Twelve objects are passed to the μGT for each event.
For each selected object, the Calo-Layer2 sends parameters for p_{T} and for position information and isolation - encoded in 32 bits:

- 9 bits p_{T}, range = 0..255 GeV (HW index = 0..0x1FF), step = 0.5, the highest bin will mark an overflow (HW index 0x1FF): meaning has to be defined
- 8 (7+1 sign) bits pseudo-rapidity (η) position, range = -5.0 to 5.0, step = 0.087/2, linear scale, 138 bins (HW index = 0xBC..0x44)
- 8 bits azimuth angle (φ) position, range = 2π, step $\approx 2\pi/144$ ($\hat{=}$2.5°), 144 bins (HW index = 0..0x8F), HW index starting at 0° (anti-clockwise)
- 2 bits isolation
- 5 bits spare

The representation of the 8 bits (called "hardware index [HW index]") in η is expected as Two's Complement notation as shown below.

Table 6: η scale of calorimeter objects

HW index	η **range**	η **bin**
0x72	114∗0.087/2 to 115∗0.087/2	114
...	...	...
0x01	0.087/2 to 2∗0.087/2	1
0x00	0 to 0.087/2	0
0xFF	0 to -0.087/2	-1
0xFE	-0.087/2 to -2∗0.087/2	-2
...	...	...
0x8E	-114∗0.087/2 to -115∗0.087/2	-115

The representation of the 8 bits in φ is expected as shown in Table 7.

Table 7: φ scale of calorimeter objects

HW index	φ range	φ range [degrees]	φ bin
0x00	0 to $2\pi/144$	0 to 2.5	0
0x01	$2\pi/144$ to $2*2\pi/144$	2.5 to 5.0	1
...	...	...	...
0x8F	$143*2\pi/144$ to 2π	357.5 to 360	143

The representation of the 2 bits for isolation (e/γ and tau) is expected as shown in Table 8.

Table 8: Definition of e/γ and tau isolation bits

bits [26..25]	definition
00	not isolated
01	isolated
10	TBD
11	TBD

4.4.3 Definitions of Energy sum quantities data

See also 4.2.
energy sum quantities:
Consists of following quantities (naming convention see in "Glossary"):

- **ET**
- **HT**
- ET_{miss}
- HT_{miss}
- **ETTEM**
- $\mathbf{ET}^{HF}_{miss}$
- $\mathbf{HT}^{HF}_{miss}$
- **ASYMET**
- **ASYMHT**
- **ASYMETHF**
- **ASYMHTHF**
- **CENT0**
- ..
- **CENT7**

Calo-Layer2 sends 6 frames (each 32 bits) with Energy sum quantities containing the following information:

- E_{T}, 12 bits, range = 0..2047 GeV (HW index = 0..0xFFF), step = 0.5, the highest bin will mark an overflow (HW index 0xFFF): meaning has to be defined
- azimuth angle (φ) position, 8 bits, range = 2π, step $\approx 2\pi/144$ ($\hat{=}2.5°$), 144 bins (HW index = 0..0x8F), HW index starting at 0° (anti-clockwise)
- "Towercount", 13 bits, range = 0..8191
- "Minimum bias", 4 bits, range = 0..15
- "Asymmetry", 8 bits, range = 0..255 (used 0..100)
- "Centrality", 8 bits, used as signals

Frame 0: The data structure of "total Et" (ET) quantity [including "total Et from ECAL only" (ETTEM) and "minimum bias HF+ threshold 0" bits].

Frame 1: The data structure of "total calibrated Et in jets" (HT) quantity [including "tower-count" and "minimum bias HF- threshold 0" bits].

Frame 2: The data structure of "missing Et" (ET_{miss}) quantity [including "Asymmetry" ASYMET and "minimum bias HF+ threshold 1" bits].

Frame 3: The data structure of "missing Ht" (HT_{miss}) quantity [including "Asymmetry" ASYMHT and "minimum bias HF- threshold 1" bits].

Frame 4: The data structure of "missing Et including HF" (ET^{HF}_{miss}) quantity [including "Asymmetry" ASYMETHF and "Centrality" bits (3:0)].

Frame 5: The data structure of "missing Ht including HF" (HT^{HF}_{miss}) quantity [including "Asymmetry" ASYMHTHF and "Centrality" bits (7:4)].

4.4.4 Definitions of Muon data

Eight Muon objects are provided by Global Muon Trigger. One Muon object has a 64 bits data structure with parameters for p_T, for unconstrained p_T, for impact parameter, for position, charge, quality and isolation information (see also 4.2.2):

- 10 bits azimuth angle (φ) position, range = 2π, step $\approx 2\pi/576$ ($\hat{=}$0.625°), 576 bins (HW index = 0..0x23F), HW index starting at 0° (anti-clockwise)
- 9 bits p_T, range = 0..255 GeV (HW index = 0..0x1FF), step = 0.5, the highest bin will mark an overflow (HW index 0x1FF): meaning has to be defined
- 4 bits quality, 16 types for quality (meaning not defined yet!)
- 9 (8+1 sign) bits pseudo-rapidity (η) position, range = -2.45 to 2.45, step = 0.087/8, linear scale, 452 bins (-225..225, HW index = 0x11F..0x0E1)
- 2 bits isolation, 4 types for isolation (meaning not defined yet!)
- 1 bit charge sign, charge sign = '0' means "positive" charge, charge sign = '1' means "negative" charge
- 1 bit charge valid (='1' means "valid")
- 7 index bits
- 10 bits azimuth angle (φ) position, raw data
- 8 bits unconstrained p_T, range = 0..255 GeV (HW index = 0..0xFF), step = 1.0, the highest bin will mark an overflow (HW index 0xFF)
- 1 spare bit
- 2 bits impact parameter

The representation of the 9 bits (called "hardware index [HW index]") in η is expected as Two's Complement notation as shown in Table 9.
The central value of the bin 0 (-0.010875/2 to +0.010875/2) = 0.0, the left edge of the bins will range from $-255 \times 0.010875 - 0.010875/2 = -2.7785625$ to $+255 \times 0.010875 - 0.010875/2 = 2.7676875$. The central value of the bins will range between ± 2.773125 . The physical η range of the muon detectors is about ± 2.45, so that not all possible η bins will be used.

The representation of the 10 bits in φ is expected as shown in Table 10.

The representation of the 4 bits for quality is expected as shown in Table 11.

The representation of the 2 bits for isolation is expected as shown in Table 13.

The representation of the 2 bits for impact parameter is expected as shown in Table 13.

Table 9: η scale of muon objects

HW index	η range	η bin
0x0E1	224.5∗0.087/8 to 225.5∗0.087/8	225
0x0E0	223.5∗0.087/8 to 224.5∗0.087/8	224
...	...	...
0x001	0.5∗0.087/8 to 1.5∗0.087/8	1
0x000	0.5∗-0.087/8 to 0.5∗0.087/8	0
0x1FF	0.5∗-0.087/8 to 1.5∗-0.087/8	-1
0x1FE	1.5∗-0.087/8 to -2.5∗0.087/8	-2
...	...	...
0x11F	-224.5∗0.087/8 to -225.5∗0.087/8	-225

Table 10: φ scale of muon objects

HW index	φ range	φ range [degrees]	φ bin
0x000	0 to $2\pi/576$	0 to 0.625	0
0x001	$2\pi/576$ to $2*2\pi/576$	0.625 to 1.250	1
...	...	...	...
0x23F	$575*2\pi/576$ to 2π	359.375 to 360	575

Table 11: Definition of muon quality bits

bits [22..19]	definition
0000	quality "level 0"
0001	quality "level 1"
...	...
1110	quality "level 14"
1111	quality "level 15"

Table 12: Definition of muon isolation bits

bits [33..32]	definition
00	not isolated
01	isolated
10	TBD
11	TBD

Table 13: Definition of muon impact parameter bits

bits [63..62]	definition
00	TBD
01	TBD
10	TBD
11	TBD

4.4.5 Calculation of object cuts

List of object cuts:

- p_{T}
- η
- φ
- isolation
- DISP
- charge
- quality
- unconstrained p_{T}
- impact parameter

4.4.5.1 Object cuts

The comparisons for objects cuts are done by:
A comparator between the energy (p_{T}) and a threshold (pt_threshold) with 'mode-selection'. Similar for unconstrained p_{T}.
The comparison in η is done with five "window"-comparators, so one gets max. five ranges for η. The η value (HW index) has a Two's Complement notation, the comparisons is done signed. Number of windows is given for η.
The comparison in φ is done with two "window"-comparators, so one gets two ranges for φ. The comparisons is done unsigned. Number of windows is given for φ.
There are two cases how the limits of one "window"-comparator could be set (see also Figure 11):

- Upper limit is less than lower limit => φ range between the limits, including the φ bin with value = 0 (HW index).
- Upper limit is greater/equal than lower limit => φ range between the limits, not including the φ bin with value = 0 (HW index).

```
phi_comp_w1 <= '1' when phi_w1_upper_limit < phi_w1_lower_limit and
                (phi <= phi_w1_upper_limit or phi >= phi_w1_lower_limit) else
               '1' when phi_w1_upper_limit >= phi_w1_lower_limit and
                (phi <= phi_w1_upper_limit and phi >= phi_w1_lower_limit)
                else '0';
```

Only one of the required ranges ("windows") must be fulfilled by η and φ values ("or").

The comparisons for isolation, quality and impact parameter are done with LUTs.
The comparison for charge is done with requested charge.
If DISP bit is set to 1, then the jet has been tagged as an LLP jet. A one bit requirement is given for DISP for comparison.

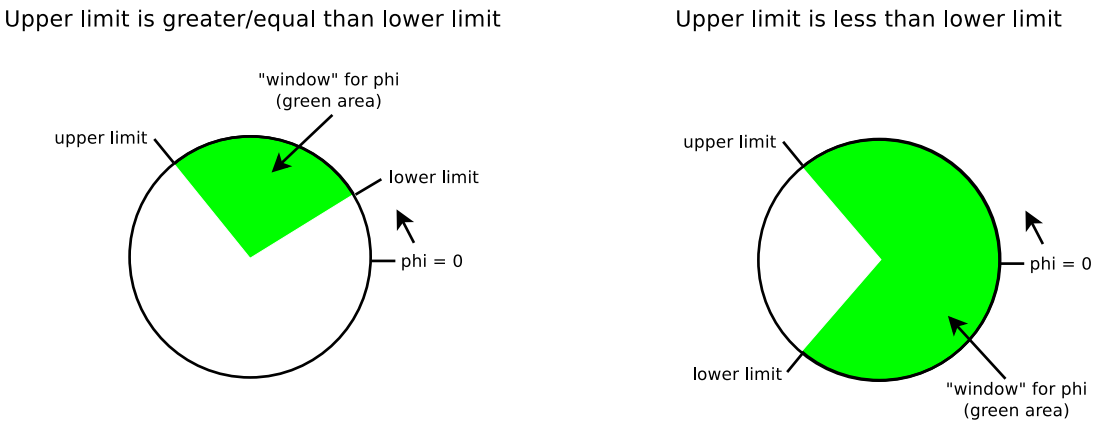


Figure 11: Setting the limits for "window"-comparators for φ

The comparison of isolation (for electron/γ, tau and muon) is done with a LUT (Table 14). [To ignore quality comparison, all bits in the LUT have to be '1'.]

The comparison of impact parameter is done with LUT (Table 15). [To ignore quality comparison, all bits in the LUT have to be '1'.]

The comparison of quality is done with LUT (Table 16). [To ignore quality comparison, all bits in the LUT have to be '1'.]

Charge valid and charge sign bits must be equal to the requested charge.

Table 14: LUT contents for isolation comparison

LUT content (4 bits)	isolation (2 bits)	trigger
X"0"	xx	no trigger
X"1"	00	trigger on isolation bits = 00
X"2"	01	trigger on isolation bits = 01
X"3"	00 or 01	trigger on isolation bits = 00 or 01
X"4"	10	trigger on isolation bits = 10
X"5"	00 or 10	trigger on isolation bits = 00 or 10
X"6"	01 or 10	trigger on isolation bits = 01 or 10
X"7"	00 or 01 or 10	trigger on isolation bits = 00 or 01 or 10
X"8"	11	trigger on isolation bits = 11
X"9"	00 or 11	trigger on isolation bits = 00 or 11
X"A"	01 or 11	trigger on isolation bits = 01 or 11
X"B"	00 or 01 or 11	trigger on isolation bits = 00 or 01 or 11
X"C"	10 or 11	trigger on isolation bits = 10 or 11
X"D"	00 or 10 or 11	trigger on isolation bits = 00 or 10 or 11
X"E"	01 or 10 or 11	trigger on isolation bits = 01 or 10 or 11
X"F"	00 or 01 or 10 or 11	trigger on isolation bits = 00 or 01 or 10 or 11 (= "ignore" isolation)

Table 15: LUT contents for impact parameter comparison

LUT content (4 bits)	impact parameter (2 bits)	trigger
X"0"	xx	no trigger
X"1"	00	trigger on impact parameter bits = 00
X"2"	01	trigger on impact parameter bits = 01
X"3"	00 or 01	trigger on impact parameter bits = 00 or 01
X"4"	10	trigger on impact parameter bits = 10
X"5"	00 or 10	trigger on impact parameter bits = 00 or 10
X"6"	01 or 10	trigger on impact parameter bits = 01 or 10
X"7"	00 or 01 or 10	trigger on impact parameter bits = 00 or 01 or 10
X"8"	11	trigger on impact parameter bits = 11
X"9"	00 or 11	trigger on impact parameter bits = 00 or 11
X"A"	01 or 11	trigger on impact parameter bits = 01 or 11
X"B"	00 or 01 or 11	trigger on impact parameter bits = 00 or 01 or 11
X"C"	10 or 11	trigger on impact parameter bits = 10 or 11
X"D"	00 or 10 or 11	trigger on impact parameter bits = 00 or 10 or 11
X"E"	01 or 10 or 11	trigger on impact parameter bits = 01 or 10 or 11
X"F"	00 or 01 or 10 or 11	trigger on impact parameter bits = 00 or 01 or 10 or 11 (= "ignore" impact parameter)

Table 16: LUT contents for quality comparison of muon objects

LUT content (16 bits)	quality bits (4 bits)	trigger
X"0000"	xxxx	no trigger
X"0001"	0000	trigger on quality "level 0"
X"0002"	0001	trigger on quality "level 1"
X"0003"	0001 or 0000	trigger on quality "level 1" or "level 0"
X"0004"	0010	trigger on quality "level 2"
...	...	...
X"8000"	1111	trigger on quality "level 15"
X"C000"	1111 or 1110	trigger on quality "level 15" or "level 14"
...	...	...
X"FFFF"	xx	trigger on all quality "levels" (= "ignore")

4.4.6 Calculation of correlation cuts

The following cuts are used for two objects correlations:

- $\Delta\eta$ (DETA).
- $\Delta\varphi$ (DPHI).
- ΔR (DR).
- charge correlation (only for muon).
- Cuts for mass (MASS) of following mass types:
  - Invariant mass.
  - Invariant mass with unconstrained pt (for muons only).
  - Invariant mass over ΔR.
  - Transverse mass.
- Two-body pt.

There is one mass cut for correlations with three objects:

- Invariant mass for three objects (MASS).

The generation of look-up-tables (LUTs) for calculations of correlation cuts is described in chapter "Calculation of look-up-tables (LUTs) for correlation cuts" (see 4.4.7).

Calculation of $\Delta\eta$

The calculation of $\Delta\eta$ of two objects is done with formula:

$\Delta\eta$=abs(η1-η2)

where η1 and η2 are represented in signed hardware indices.

Calculation of $\Delta\varphi$

The calculation of $\Delta\varphi$ of two objects is done with formula:

$\Delta\varphi$=abs(φ1-φ2) with ("φ full bin range"-$\Delta\varphi$) when ($\Delta\varphi$ > "φ half bin range").

where φ1 and φ2 are represented in unsigned hardware indices.

ΔR calculation

The calculation of ΔR of two objects is done with formula:

ΔR=$\sqrt{(\eta 1 - \eta 2)^2 + (\varphi 1 - \varphi 2)^2}$.

The calculation of ΔR^2 in VHDL (no square root in VHDL) is done by adding the square of $\Delta\eta$ and $\Delta\varphi$ LUT values.

Invariant mass calculation

The calculation of *invariant mass of two objects* is done with formula:

M=$\sqrt{2pt_1pt_2(\cosh(\eta 1-\eta 2)-\cos(\varphi 1-\varphi 2))}$.

The calculation of $\frac{M^2}{2}$ in VHDL (no square root in VHDL) is done by multiplying LUT values of pt1, pt2 and the difference of cosh($\Delta\eta$) and cos($\Delta\varphi$).

Transverse mass calculation

The calculation of *transverse mass of two objects* is done with formula:

M=$\sqrt{2pt_1pt_2(1-\cos(\varphi 1-\varphi 2))}$.

Calculation similar to "Invariant mass calculation".

Invariant mass over ΔR calculation

The formulas for *invariant mass over* ΔR *of two objects* are:

M=$\sqrt{2pt_1pt_2(\cosh(\eta 1-\eta 2)-\cos(\varphi 1-\varphi 2))}$.

ΔR=$\sqrt{(\eta 1-\eta 2)^2+(\varphi 1-\varphi 2)^2}$.

The calculation of *invariant mass over* ΔR *of two objects* is done with $\frac{M^2}{2}\times(1/\Delta R^2)$ (no square root in VHDL).
A direct calculation of $1/\Delta R^2$ is not possible in firmware (VHDL code), therefore the implementation of the calculation is done by LUTs. In the hardware the values of these LUTs are stored in "large" ROMs, which was realized using the Block RAMs (BRAMs) of the Virtex chip.
Due the limited number of available BRAMs there are some restrictions for creating algorithms with *invariant mass over* ΔR:

- Objects must have the same type (e.g.: "muon muon", "eg eg", ...)
- Objects must be of same bx
- Resolution of $\Delta\eta$ and $\Delta\varphi$:
  - Full resolution for calos (max. deta bins=230, max. dphi bins=72)
  - Half resolution only for muons (max. deta bins=226, max. dphi bins=144)
- If $1/\Delta R^2$=0 ($\Delta\eta$=0 and $\Delta\varphi$=0) then correlation cut *invariant mass over* ΔR is true
- The values of LUTs are only valid for current definitions and restrictions. Every change might cause a recalculation of the values and a regeneration of IPs (representing LUTs in BRAMs) in Vivado (firmware generation tool)

The values of LUTs in firmware are listed in coe files of ROMs (created by same scripts mentioned above), currently 5 ROMs for "calo calo" and 6 ROMs for "muon muon" (see `lut_calo_inv_dr_sq_rom1.coe`, etc. and `lut_muon_inv_dr_sq_rom1.coe`, etc.). The addresses of the BRAMs are given by $\Delta\eta$ and $\Delta\varphi$. All ROMs for calos have 4096 addresses, for muons 8192 addresses. The data width of ROMs is different depending on the highest LUT value in ROM. Because of these different data widths, the partitioning of several ROMs was done to save BRAM resources. Currently 873 BRAMs (36kb) are available per Virtex chip. Following numbers of BRAMs (36kb) are needed for:

- "calo calo": 660
- "muon muon": 672

Currently one calculation of *invariant mass over* ΔR of "calo calo" or "muon muon" is possible in one Virtex chip, but one can have some algorithms containing *invariant mass over* ΔR with different thresholds, but with same objects and same bx.

Invariant mass calculation for three objects

The calculation of *invariant mass calculation for three objects* is done by calculating the invariant mass for all two-object combinations and take the sum of the three invariant masses of the two-object combinations.

Two-body pt calculation

The calculation of *two-body pt* is done with formula:

$$\text{pt}=\sqrt{pt_1^2 + pt_2^2 + 2pt_1pt_2(\cos(\varphi 1)\cos(\varphi 2) + \sin(\varphi 1)\sin(\varphi 2))}$$

The calculation of pt^2 in VHDL (no square root in VHDL) using LUTs for pt1, pt2, $\cos(\varphi)$ and $\sin(\varphi)$.

Muon charge correlation

For definition of muon charge, see 4.4.4.
In the muon charge correlation module (`muon_charge_correlations.vhd`), the charge correlations are made for different muon conditions-types. The module is instantiated in the top-of-hierarchy module (`gtl_module.vhd`) and not inside of a muon conditions module. The charges of objects (number of objects depends on muon condition type) are compared to get "like sign charge" ("LS") or "opposite sign charge" ("0S"), "LS" means that the charges (charge sign) of objects are the same, "0S" means that at least one object has different charge than the others. This information is used in all instatiated muon conditions. There is no charge correlation for single type conditions.
In all cases the "charge valid" bit of the objects must be set.
In TME one can select "LS", "0S" or ignore for charge correlation in muon conditions.

Table 17: Muon charge correlation - Double Muon

x x	I ignore (charge x = +, -, I)
+ +	LS both positive muons
− −	LS both negative muons
I I	LS both muons with the same sign, positive or negative
+ −	OS two muons of opposite sign
− +	OS idem
I I	OS idem

Table 18: Muon charge correlation - Triple Muon

x x x	I ignore (charge x = +, -, I)
+ + +	LS three muons of positive charge
− − −	LS three muons of negative charge
I I I	LS three muons of the same sign (positive or negative)
+ + −	OS a pair plus a positive muon
+ − −	OS a pair plus a negative muon
+ − I	OS a pair plus a negative or positive muon

Table 19: Muon charge correlation - Quad Muon

x x x x	I ignore (charge x = +, -, I)
+ + + +	LS four muons of positive charge
− − − −	LS four muons of negative charge
I I I I	LS four muons of the same sign (positive or negative)
+ + + −	OS a pair plus two positive muons
+ + − −	OS two pairs
+ − − −	OS a pair plus two negative muons
+ − I I	OS a pair plus two negative or positive muons

4.4.7 Calculation of look-up-tables (LUTs) for correlation cuts

LUTs are defined as a VHDL "constant" in `gtl_luts_pkg.vhd` (VHDL package file). The values of precision and step size are given by "scale_set" in XML file of a L1 menu.

Overview of precision types for correlation cuts (an example for electron/γ electron/γ correlation):

- *EG-EG-Delta* relevant for DeltaEta and DeltaPhi LUTs
- *EG-EG-MassPt* relevant for pt and unconstrained pt LUTs (used in mass and two-body pt calculations)
- *EG-EG-Math* relevant for cos(DeltaPhi) and cosh(DeltaEta) LUTs (used in mass calculations)
- *EG-EG-InverseDeltaRMath* relevant for 1/DeltaR LUTs (used in mass over deltaR calculations)
- *EG-EG-TwoBodyPtMath* relevant for cos(Phi) and sin(Phi) LUTs (used in two-body pt calculations)
- *EG-EG-DeltaOverlapRemoval* is obsolete, used EG-EG-Delta (same scales for η and φ)
- *EG-EG-Mass* currently not used
- *EG-EG-TwoBodyPt* is obsolete, used EG-EG-MassPt

Overview of precision names (example for "MassPt"):
EG-EG-MassPt
EG-JET-MassPt
EG-TAU-MassPt
JET-JET-MassPt
JET-TAU-MassPt
EG-ETM-MassPt
JET-ETM-MassPt
TAU-ETM-MassPt
EG-HTM-MassPt
JET-HTM-MassPt
TAU-HTM-MassPt
EG-ETMHF-MassPt
JET-ETMHF-MassPt
TAU-ETMHF-MassPt
EG-MU-MassPt
JET-MU-MassPt
TAU-MU-MassPt
MU-MU-MassPt
MU-ETM-MassPt
MU-HTM-MassPt

MU-ETMHF-MassPt

LUTs for p_{T} and unconstrained p_{T} used in mass and two-body pt calculations

The values of p_{T} or unconstrained p_{T} LUT are calculated by building the half difference of maximum and minimum value of a bin, adding minimum value, rounding at precision position after decimal point and multiplying with $10^{\mathrm{precision}}$ to get integer values.
The address input of the LUT for p_{T} or unconstrained p_{T} is the value of hardware index of p_{T} or unconstrained p_{T}.

The precision values in XML file are given by (an example for electron/γ electron/γ correlation):
<scale>
<object>PRECISION</object>
<type>EG-EG-MassPt</type>
...
<n_bits>1</n_bits>
</scale>

VHDL names of p_{T} and unconstrained p_{T} LUTs:
EG_PT_LUT (used also for tau)
JET_PT_LUT
ETM_PT_LUT (used also for HT_{miss} and ET^{HF}_{miss})
MU_PT_LUT
MU_UPT_LUT

LUTs for $\Delta\eta$

The values of the LUT are calculated by multiplying $\Delta\eta$ in hardware indices with η step size, rounding at precision position after decimal point and multipling the result with $10^{\mathrm{precision}}$ to get integer values.
The address of the LUT is the value of $\Delta\eta$ in hardware indices.

The precision value in XML file is given by (an example for electron/γ electron/γ correlation):
<scale>
<object>PRECISION</object>
<type>EG-EG-Delta</type>
...
<n_bits>3</n_bits>
</scale>
where <n_bits> is the precision value and <type> represents a precision name.

The η (=$\Delta\eta$) step size in XML file is given by (an example for electron/γ):
<scale>
<object>EG</object>
<type>ETA</type>

...
<step>+4.3499999999999997E-02</step>
...
</scale>

VHDL names of $\Delta\eta$ LUTs:
CALO_CALO_DIFF_ETA_LUT
CALO_MU_DIFF_ETA_LUT
MU_MU_DIFF_ETA_LUT

LUTs for $\Delta\varphi$

The values of the LUT are calculated by multiplying $\Delta\varphi$ in hardware indices with φ step size, rounding at precision position after decimal point and multipling the result with $10^{\text{precision}}$ to get integer values.
The address of the LUT is the value of $\Delta\varphi$ in hardware indices.

The precision values of $\Delta\varphi$ are identical with $\Delta\eta$.

The φ (=$\Delta\varphi$) step size in XML file is given by (an example for electron/γ):
<object>EG</object>
<type>PHI</type>
...
<step>+4.3633231299858237E-02</step>
...
</scale>

VHDL names of $\Delta\varphi$ LUTs:
CALO_CALO_DIFF_PHI_LUT
CALO_MU_DIFF_PHI_LUT
MU_MU_DIFF_PHI_LUT

LUTs for $\cosh(\Delta\eta)$ used in mass calculations

The values in the LUT are calculated by multiplying $\Delta\eta$ in hardware indices with η step size, calculating cosine hyperbolic, rounding at "Math" precision position after decimal point and multipling the result with $10^{\text{precision}}$ to get integer values.
The address of the LUT for $\cosh(\Delta\eta)$ is the value of $\Delta\eta$ in hardware indices.
For calo muon correlations one has to use the muon step size.

The precision values in XML file are given by (an example for electron/γ electron/γ correlation):
<scale>
<object>PRECISION</object>
<type>EG-EG-Math</type>

...
<n_bits>3</n_bits>
</scale>
used for cosh($\Delta\eta$) and cos($\Delta\varphi$).

VHDL names of cosh($\Delta\eta$) LUTs:
CALO_CALO_COSH_DETA_LUT
CALO_MUON_COSH_DETA_LUT
MU_MU_COSH_DETA_LUT

LUTs for cos($\Delta\varphi$) used in mass calculations

The values in the LUT are calculated by multiplying $\Delta\varphi$ in hardware indices with φ step size, calculating cosine, rounding at "Math" precision position after decimal point and multipling the result with $10^{\text{precision}}$ to get integer values.
The address of the LUT for cos($\Delta\varphi$) is the value of $\Delta\varphi$ in hardware indices. For calo muon correlations one has to use the muon step size.

VHDL names of cos($\Delta\varphi$) LUTs:
CALO_CALO_COS_DPHI_LUT
CALO_MUON_COS_DPHI_LUT
MU_MU_COS_DPHI_LUT

LUTs for $1/\Delta R^2$ used in mass over deltaR calculations

The calculation of $1/\Delta R^2$ is done by multiplying $\Delta\eta$ in hardware indices with η step size, making the square, doing the same for $\Delta\varphi$, adding the squares, inverting the sum, rounding at "InverseDeltaRMath" precision position after decimal point and multipling the result with $10^{\text{precision}}$ to get integer values. The address of the two-dimensional LUT for $1/\Delta R^2$ consists of values of $\Delta\eta$ and $\Delta\varphi$ in hardware indices.

The precision values in XML file are given by (an example for electron/γ electron/γ correlation):
<scale>
<object>PRECISION</object>
<type>EG-EG-InverseDeltaRMath</type>
...
<n_bits>5</n_bits>
</scale>

Precision names for "InverseDeltaRMath":
EG-EG-InverseDeltaRMath
JET-JET-InverseDeltaRMath
TAU-TAU-InverseDeltaRMath
MU-MU-InverseDeltaRMath

LUTs for $\cos(\varphi)$ used in two-body pt calculations

The values in the LUT are calculated by building the half difference of maximum and minimum value of a φ bin, adding minimum value, calculating cosine, rounding at "TwoBodyPtMath" precision position after decimal point and multipling the result with $10^{\text{precision}}$ to get integer values.

The precision values in XML file are given by (an example for electron/γ electron/γ correlation):
<scale>
<object>PRECISION</object>
<type>EG-EG-TwoBodyPtMath</type>
...
<n_bits>3</n_bits>
</scale>
used for $\cos(\varphi)$ and $\sin(\varphi)$.

VHDL names of $\cos(\varphi)$ LUTs:
CALO_COS_PHI_LUT
MUON_COS_PHI_LUT

LUTs for $\sin(\varphi)$ used in two-body pt cuts

The values in the LUT are calculated by building the half difference of maximum and minimum value of a φ bin, adding minimum value, calculating sine, rounding at "TwoBodyPtMath" precision position after decimal point and multipling the result with $10^{\text{precision}}$ to get integer values.

VHDL names of $\sin(\varphi)$ LUTs:
CALO_SIN_PHI_LUT
MUON_SIN_PHI_LUT

4.4.8 Combination conditions

4.4.8.1 Combination conditions definition

A condition consists of input data and a set of requirements, which contain the requirements to be complied. The requirements are called "object cuts".

The requirement list contains:
thresholds for p_{T}, ranges for η and φ, LUTs for isolation, LUTs for quality, requsted charges, thresholds for unconstrained p_{T}, a LUT for impact parameter. The condition is complied, if every comparison between object parameters and requirements is valid for the following object cuts (only for requested cuts):

For Calorimeter input data:

- p_{T} greater-equal (or equal) threshold
- η in range
- φ in range
- iso LUT

For Muon input data:

- p_{T} greater-equal (or equal) threshold
- η in range
- φ in range
- iso LUT
- requested charge
- quality LUT
- unconstrained p_{T} greater-equal (or equal) threshold
- impact parameter LUT

There are different types of conditions implemented, depending of how many objects have to comply the requirements.

- "Quad objects requirements condition": this condition type consists of requirements for 4 different trigger objects of the same object type. For each object the requirements can be different. To fulfill this condition, there must exist at least one set of 4 different objects, each of which fulfills at least one of the requirements.
- "Triple objects requirements condition": this condition type consists of requirements for 3 different trigger objects of the same object type. For each object the requirements can be different. To fulfill this condition, there must exist at least one set of 3 different objects, each of which fulfills at least one of the requirements.

- "Double objects requirements condition": this condition type consists of requirements for 2 different trigger objects of the same object type. For each object the requirements can be different. To fulfill this condition, there must exist at least one set of 2 different objects, each of which fulfills at least one of the requirements.[1]
- "Single object requirement condition": this condition type consists of one requirement for one trigger object of a given object type. To fulfill this condition, there must exist at least one object which fulfills the requirement.

The values of the requirements are given by VHDL Producer for every Trigger Menu.
The input data objects have to be of same type and same bunch-crossing.

With "Double objects requirements condition" a correlation cut of "two-body pt" can be required (calorimeter and muon objects).
Additionally charge correlation cuts with "Double objects requirements condition", "Triple objects requirements condition" and "Quad objects requirements condition" of muon objects can be required.

[1] "Double objects requirements condition with spatial correlation" not used anymore, replaced by Correlation conditions

Table 20: Explanation of Listing 4

Item	Explanation
et_ge_mode	'mode-selection' for the E_T comparator. Valid strings are 'true' and 'false' (type is boolean), 'true' means comparator works on greater/equal, 'false' means equal (for tests only)
obj_type	valid strings are 'ETT_TYPE', 'HTT_TYPE', 'ETM_TYPE', 'HTM_-TYPE' and 'ETMHF_TYPE'.
et_threshold	threshold value for comparison in E_T. The size of the std_logic_vector depends on the number of E_T bits.
phi_full_range	boolean to set full range of φ.
phi_w1_upper_limits	"upper limit" of "window"-comparator 1 for φ.
phi_w1_lower_limits	"lower limit" of "window"-comparator 1 for φ.
phi_w2_ignore	boolean to ignore "window"-comparator 2 for φ.
phi_w2_upper_limits	"upper limit" of "window"-comparator 2 for φ.
phi_w2_lower_limits	"lower limit" of "window"-comparator 2 for φ.
clk	clock input (LHC clock).
data_i	input data, structure defined in obj_type.
condition_o	output of condition (routed to Algorithms logic, see 4.4.15).

4.4.9 Energy sum quantities conditions

4.4.9.1 Energy sum quantities conditions module (including Asymmetry conditions)

For the entity-declaration of esums_conditions.vhd, see Listing 4.

Listing 4: Entity declaration of esums_conditions.vhd

```
entity esums_conditions is
    generic     (
        et_ge_mode : boolean;
        obj_type : natural := ETT_TYPE; -- ett=0, ht=1, etm=2, htm=3
        et_threshold: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0);
        phi_full_range : boolean;
        phi_w1_upper_limit: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0)
            ;
        phi_w1_lower_limit: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0)
            ;
        phi_w2_ignore : boolean;
        phi_w2_upper_limit: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0)
            ;
        phi_w2_lower_limit: std_logic_vector(MAX_ESUMS_TEMPLATES_BITS-1 downto 0)
   );
    port(
        clk : in std_logic;
        data_i : in std_logic_vector(MAX_ESUMS_BITS-1 downto 0);
        condition_o : out std_logic
    );
end esums_conditions;
```

A comparator between E_{T} and a threshold (et_threshold) and, depending on object type, a comparison in φ with two "window"-comparators is done in this module. The value for E_{T} threshold, the 'mode-selection' for the E_{T} comparator and the limits for the "window"-comparators are given in the generic interface list of the module. The selection whether a comparison in φ is part of the condition is done with the value of the generic parameter 'obj_type' ('ETM_TYPE', 'ETMHF_TYPE', 'HTM_TYPE' and 'HTMHF_TYPE' force a comparison). The comparison in φ is done in the same way as for calorimeter conditions. Additionally the data-structure of input data (data_i in port interface list) is provided as a record in this list. The output signal of the module is in high state, if all comparisons are true.
Data for Asymmetry trigger are received on 4 frames on bits 27..20 (8 bits). For every type a comparision with an 8-bit threshold (greater-equal [or equal]) is done. Asymmetry data are interpreted as counts.

4.4.10 Minimum bias trigger conditions

Data for Minimum bias trigger are received on the 4 MSBs of 4 frames used for Energy sum quantities (see 4.4.9).

- MBT0HFP: "minimum bias HF+ threshold 0" bits
- MBT0HFM: "minimum bias HF- threshold 0" bits
- MBT1HFP: "minimum bias HF+ threshold 1" bits
- MBT1HFM: "minimum bias HF- threshold 1" bits

In minimum bias trigger conditions module there is a comparision with a 4-bit threshold (greater-equal [or equal]).

4.4.11 Towercount condition

Data for Towercount trigger (number of firing HCAL towers) are received on frame HT (see 4.4.9) on bits 24..12 (13 bits) of HT data structure.
In towercount condition module there is a comparision with a 13-bit threshold (greater-equal [or equal]).

4.4.12 Centrality condition

Centrality bits used as a signals for triggers (similar to external signals).

4.4.13 Correlation conditions

The correlation conditions contain a combination of two "Single object requirement conditions" of two object types or one "Double objects requirement condition" of objects of the same type. In addition with object cuts there are correlation cuts for $\Delta\eta$, $\Delta\varphi$, ΔR, mass, mass divided by ΔR and "two-body pt".
The correlation condition of "Invariant mass for three objects" contains one "Triple objects requirement condition" of objects of the same type with one object cut for mass.

List of correlation cuts in 4.4.6.

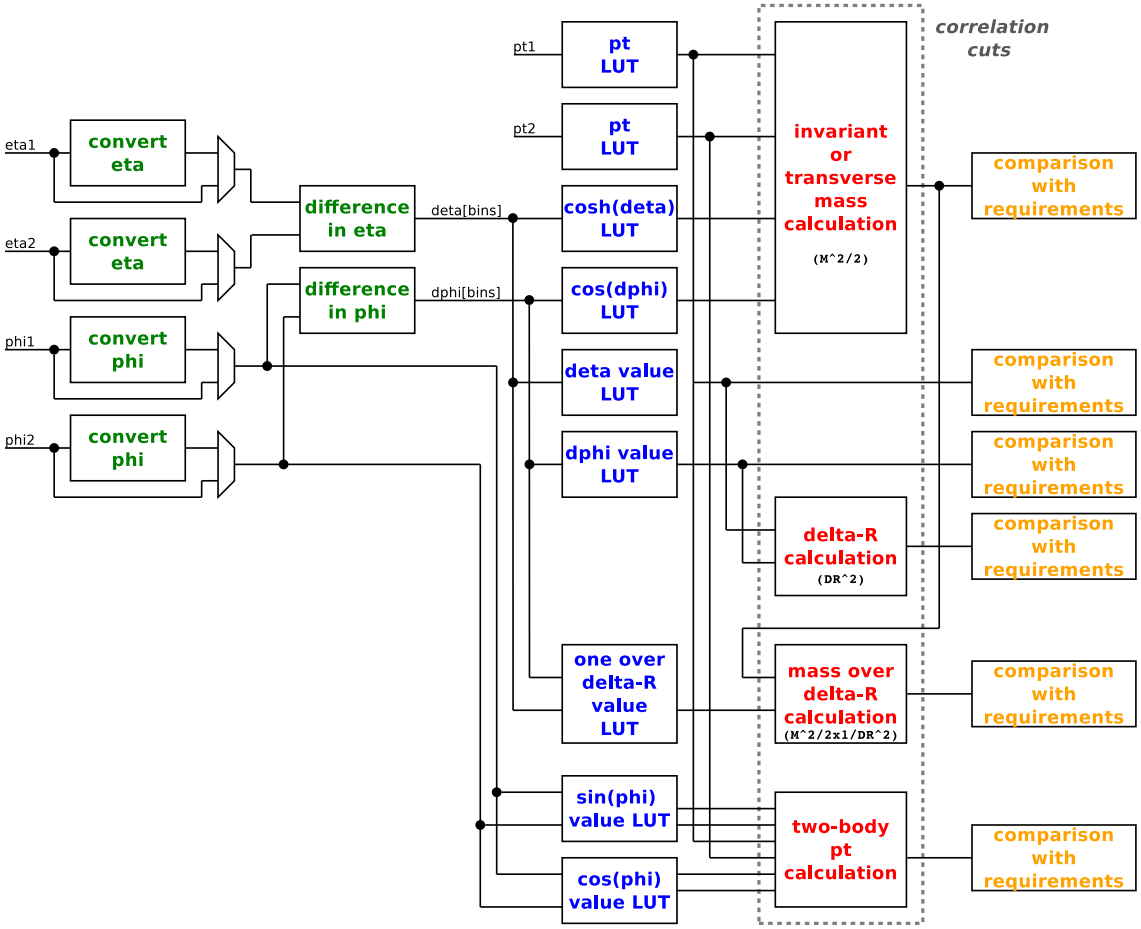


Figure 12: VHDL structure of cuts for correlation conditions

Overview of correlation cuts in conditions

The following list gives an overview of possible correlation cuts in conditions:

- Calo conditions:
  - two-body pt (for double condition)
- Calo conditions overlap removal:

  - $\Delta\eta$ overlap removal
  - $\Delta\varphi$ overlap removal
  - ΔR overlap removal
  - two-body pt (for double condition)
- Muon conditions:
  - charge correlation
  - two-body pt (for double condition)
- Calo calo correlation condition with calo overlap removal:
  - $\Delta\eta$ overlap removal
  - $\Delta\varphi$ overlap removal
  - ΔR overlap removal
  - $\Delta\eta$
  - $\Delta\varphi$
  - ΔR
  - invariant mass
  - two-body pt
- Calo calo correlation condition:
  - $\Delta\eta$
  - $\Delta\varphi$
  - ΔR
  - invariant mass
  - two-body pt
- Calo calo correlation condition for invariant mass divided by ΔR:
  - invariant mass divided by ΔR
- Calo calo correlation condition mass with three objects:
  - invariant mass with three objects
- Calo muon correlation condition:
  - $\Delta\eta$
  - $\Delta\varphi$
  - ΔR
  - invariant mass
  - two-body pt
- Calo esums correlation condition:

  - $\Delta\varphi$
  - transverse mass
  - two-body pt
- Muon muon correlation condition:
  - charge correlation
  - $\Delta\eta$
  - $\Delta\varphi$
  - ΔR
  - invariant mass or invariant mass unconstraint pt
  - two-body pt
- Muon muon correlation condition for invariant mass divided by ΔR:
  - charge correlation
  - invariant mass divided by ΔR
- Muon muon correlation condition mass with three objects:
  - charge correlation
  - invariant mass with three objects
- Muon esums correlation condition:
  - $\Delta\varphi$
  - transverse mass
  - two-body pt

4.4.13.1 Correlation condition module

As described in section Correlation conditions (4.4.13), correlations of two object types are available. Therefore several correlations (objects 1-objects 2) are possible:

- Correlation condition with calorimeter objects
  electron/γ-electron/γ, electron/γ-jet, electron/γ-tau, jet-jet, jet-tau and tau-tau.
- Correlation condition with calorimeter objects and energy sum quantities (ET_{miss}, ET^{HF}_{miss} and HT_{miss} only)
  electron/γ-etm, jet-etm, tau-etm, electron/γ-htm, jet-htm, tau-htm, electron/γ-etmhf, jet-etmhf and tau-etmhf.
- Correlation condition with calorimeter objects and muons objects
  electron/γ-muon, jet-muon and tau-muon.
- Correlation condition with muon objects

- Correlation condition with muon objects and energy sum quantities (ET_{miss}, ET^{HF}_{miss} and HT_{miss} only)
  muon-etm, muon-etmhf and muon-htm.

There are two correlations for mass with three objects:

- Correlation condition for mass with three objects with calorimeter objects (same type, same bunch-crossing)

- Correlation condition for mass with three objects with muon objects

4.4.14 External Conditions

Maximal 256 External Conditions are possible in Global Trigger. They are provided as inputs in the Algorithms logic of μGTL. External Conditions will include the "Technical Trigger" of the legacy system.

4.4.15 Algorithms logic

The outputs of all the instantiated conditions are combined in the Algorithms logic with boolean algebra given by TME for every single Algorithm. These Algorithms are registered and provided as inputs for Final Decision Logic.

5 Final Desicion Logic

The Final Desicion Logic (μFDL) firmware contains algo-bx-masks, suppression of algos caused by calibration trigger, prescalers, veto-masks and rate-counters ("before prescalers", "after prescalers" and "post dead time") for each Algorithm and the local Final-OR- and veto-logic.

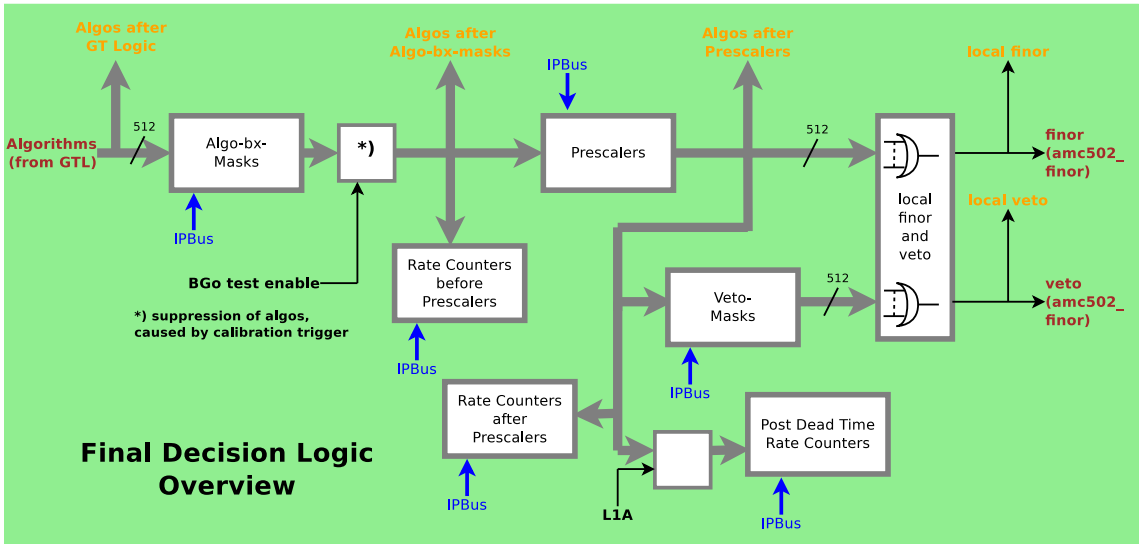


Figure 13: μFDL firmware v1.0.1

5.1 μFDL Interface

Inputs:

- Algorithms from μGTL
- IPBus interface (for registers, counters and memories)
- LHC-clock
- Reset signal
- BC0, BGo test-enable, L1A
- Begin of lumi-section

Outputs:

- Prescale factor set index to Readout-Process
- Algorithms after GTLogic to Readout-Process
- Algorithms after algo-bx-masks to Readout-Process

- Algorithms after prescalers to Readout-Process
- Algorithms after Final-OR-masks to Readout-Process
- Local Final-OR to Readout-Process
- Local veto to Readout-Process
- Local Final-OR with veto to Readout-Process
- Local Final-OR to mezzanine
- Local veto to mezzanine
- Local Final-OR with veto to mezzanine

5.2 MP7 Final-OR hardware solution

The firmware of μFDL in this document is based on a hardware configuration with maximum 6 μGT modules.

5.3 Data flow

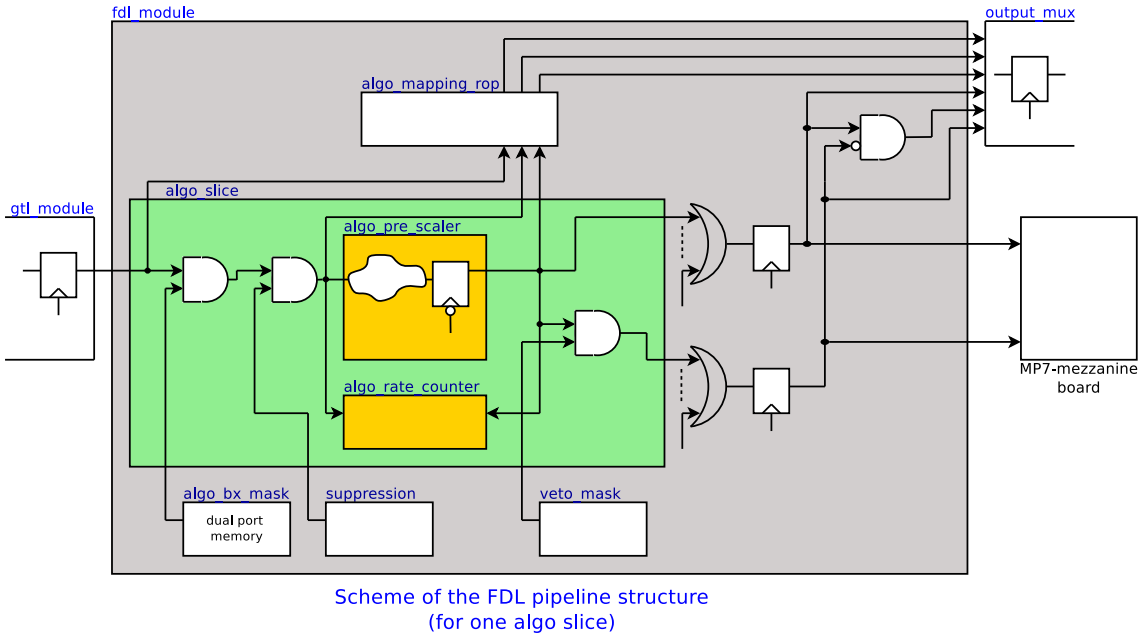


Figure 14: μFDL pipeline v1.0.1

Every Algorithm, in total 512 coming from μGTL, passes a algo-bx-mask, the logic for suppression of algos caused by calibration trigger and a prescaler, which reduces the trigger rate by a given factor. Prescaled Algorithms signals are combined to a local final-or-signal (Final-OR). For every Algorithm there is a rate-counter before prescaler and after prescaler, which

are incremented by LHC-clock if the Algorithm is true. In addition there are post-dead-time counters, one for each Algorithm, which are incremented, if the Algorithm and the L1A-signal are true at the same bunch-crossing. Algorithms after GTLogic, after algo-bx-masks, after prescalers, the local Final-OR- and local veto-signal are provided for read-out-record.
If there are not enough firmware resources in one μGT board, more boards could be used. Therefore the 512 Algorithms are partitioned by TME. TME will set the number of Algorithms as constant in the package module `gtl_pkg.vhd`. This means μGTL and μFDL firmware considered as a unit for synthesis. In the case of more μGT boards, the local Final-OR and local veto are routed via a mezzanine board on MP7 (located on "General Purpose I/O connector") to the FINOR-AMC502 module, where the total Final-OR is created and send to TCDS.
A mapping for Algorithms is provided, to give flexibility for setting the index of Algorithms:

- creating a mapping instance (algo_mapping_rop.vhd) by VHDL Producer, this component will be instantiated in fix part of FDL, and new calculation will done each time over TME.
- TME delivers just the number of Algorithms, which will be built on each card.
- from FDL point of view, FDL see incremented number of Algorithms indexes, e.g. 0, 1, 2, which is e.g. 69, 200, 300.
- TME should take care of assignment of each Algorithm to a number, that means if in card 1 algo_59 is defined, nobody allows to produce the same number again.

5.4 Main parts

The top-of-hierarchy module (`fdl_module.vhd`) contains

- version registers
- a command pulse register
- prescalers for all Algorithms
- registers for prescale factors
- register for prescale factor set index
- rate-counters for all Algorithms, finor, veto, L1A and post-dead-time
- read only registers for rate-counter values
- algo-bx-masks for all Algorithms
- Final-OR-masks for all Algorithms
- veto-masks for all Algorithms
- the Final-OR-logic

5.4.1 Registers and memories

All registers and memories are 32 bits wide. (A first draft of the definition of the relative addresses is shown in Table 21.)

- Dual-port memories for the algo-bx-masks are implemented. For each Algorithm there is a mask bit at every bunch crossing of one orbit. Therefore in total memories of 4096 x 512 bits are implemented. Because of the 32 bit data interface, 16 memories each with a size of 4096 x 32 bits are instantiated.
- Read-only registers for the value of rate-counters (before and after prescalers, post-dead-time counters) are implemented, 512 registers, one for every Algorithm. Rate-counter value has 32 bits.
- Registers for prescale factor of the prescalers are implemented, 512 registers, one for every Algorithm. A prescale factor value has 24 bits.
- Registers for masks (finor- and veto-masks) are implemented, 512 registers.
- One register for prescale factors set index is implemented. This register contains a value, which is unique for a given set of prescale factors. The content of this register is part of Readout-record.
- One register for command pulses is implemented. One bit of this register (bit 0) is used for "setting the request signal for updating prescale factors high", which enables, that the prescale factors and the prescale factor set index are loaded at the begin of a luminosity segment period. (Other bits are not defined yet.)
- One control register is implemented (the content has to be defined).
- 32 register for L1 Trigger Menu name for μGTL is implemented.
- 4 register for L1 Trigger Menu UUID for μGTL is implemented.
- One register for L1 Trigger Menu compiler version is implemented.
- One register for μFDL firmware version is implemented.
- One register for μGTL firmware (fixed code) version is implemented.

5.4.1.1 Register map

The register map for μFDL has a base address of `0x90000000`.

Table 21: μFDL register map

Offset	Register name	Access	Description
0x90000000	`Algo BX masks(0)`	r/w	4096 memory addresses of algo-bx-masks for Algorithms 0-31.

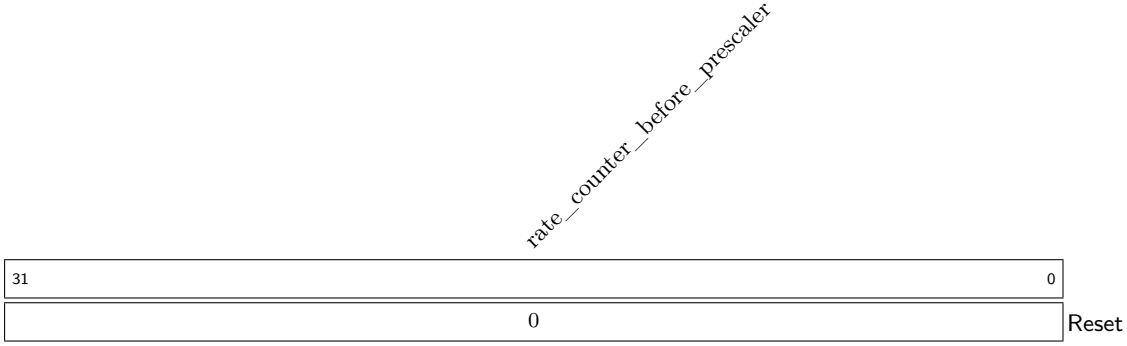
Table 21: μFDL register map

Offset	Register name	Access	Description
0x90001000	`Algo BX masks(1)`	r/w	4096 memory addresses of algo-bx-masks for Algorithms 32-63.
...	...	...	...
0x9000F000	`Algo BX masks(15)`	r/w	4096 memory addresses of algo-bx-masks for Algorithms 480-511.
0x90010000	`Rate counter before prescaler`	r	512 read-only registers for rate-counter values before prescalers.
0x90010200	`Prescale factors`	r/w	512 registers for prescale factors.
0x90010400	`Rate counter after prescaler`	r	512 read-only registers for rate-counter values after prescalers.
0x90010600	`Rate counter post-dead-time`	r	512 read-only registers for post-dead-time rate-counter values.
0x90010800	`Masks`	r/w	512 registers for finor-masks and veto-masks. Bit 0 = finor-mask, bit 1 = veto-mask.
0x90091880	`Prescale factors set index`	r/w	Register for prescale factors set index.
0x900918C0	`L1tm name`	r	32 registers for L1 Trigger Menu name for μGTL.
0x900918E0	`L1tm uuid`	r	4 registers for L1 Trigger Menu UUID for μGTL.
0x900918E4	`L1tm compiler version`	r	Register for L1 Trigger Menu compiler version.
0x900918E5	`GTL FW version`	r	Register for firmware version of μGTL VHDL code.
0x900918E6	`FDL FW version`	r	Register for firmware version of μFDL VHDL code.
0x900918E7	`L1tm FW uuid`	r	4 registers for L1 Trigger Menu FW UUID for μGTL.
0x900918EB	`SVN revision number`	r	Register for SVN revision number.
0x900918EC	`L1tm uuid hash`	r	Register for L1 Trigger Menu UUID hash for μGTL.

Table 21: μFDL register map

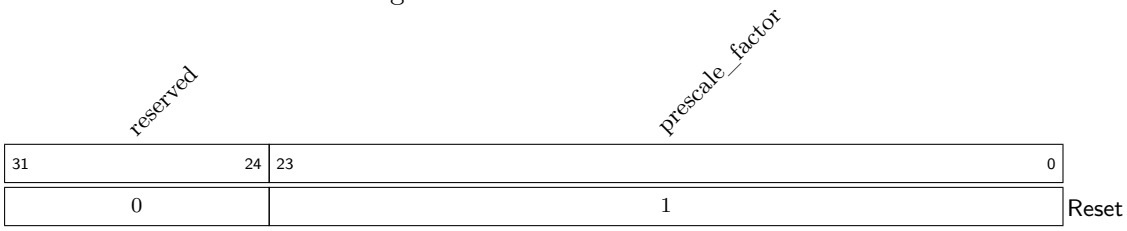
Offset	Register name	Access	Description
0x900918ED	`L1tm FW uuid hash`	r	Register for L1 Trigger Menu FW UUID hash for μGTL.
0x900918EE	`Module ID`	r	Register for Module ID of L1 Trigger Menu.
0x90091900	`Command Pulses`	r/w	Register for command pulses.
0x90091980	`Rate counter finor`	r	One read-only registers for finor rate-counter value.
0x90092200	`L1A latency delay`	r/w	Register for L1A latency delay value (used for post-dead-time counter).
0x90093000	`Rate counter L1A`	r	One read-only registers for L1A rate-counter value.
0x90094000	`Rate counter veto`	r	One read-only registers for veto rate-counter value.
0x90095000	`Current prescale set index`	r	Read-only register for prescale factors set index, which was "updated" with begin of current lumi-section ("prescale_factors_set_index_reg_updated(0)" in VHDL).
0x90095001	`Previous prescale set index`	r	Read-only register for prescale factors set index, which was "updated" with begin of previous lumi-section for monitoring "prescale_factors_set_index_reg_updated(1)" in VHDL).
0x90096000	`Calibration trigger gap`	r/w	Register for begin and end (in Bx) of calibration trigger gap.

Register 5.1: Rate counter before prescaler



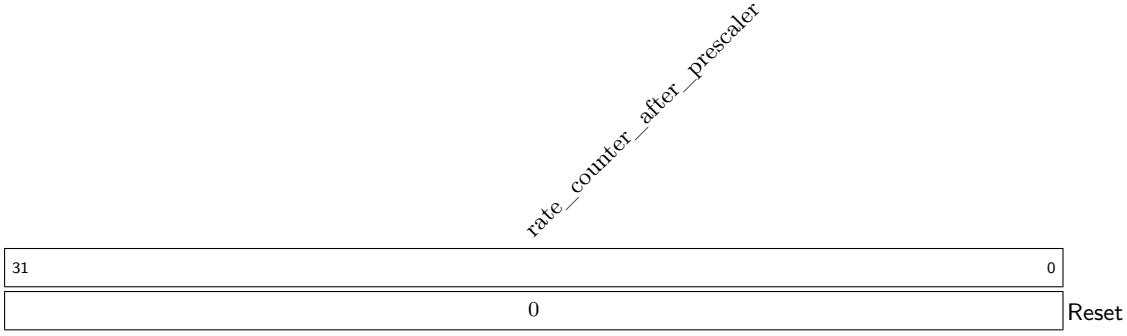
rate_counter_before_prescaler Rate counter before prescaler. Counts the occurancy of an algo (given by register address) in one luminosity segment.

Register 5.2: Prescale factor



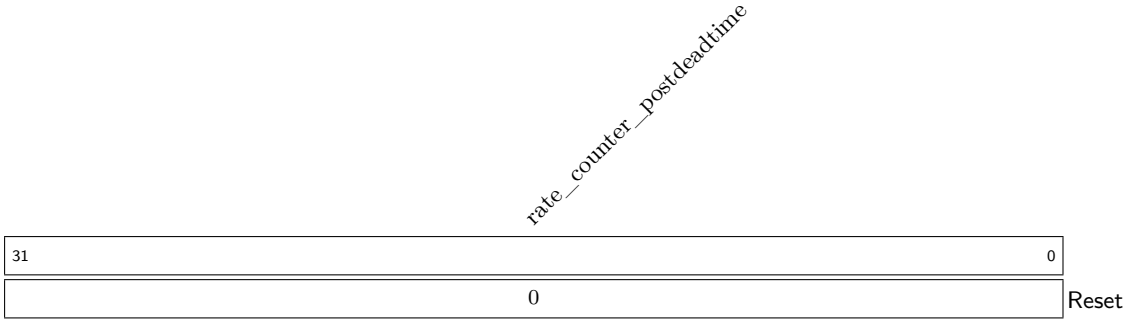
prescale_factor Prescale factor of an algo (given by register address). Prescale factor = 0 means disable algo.

Register 5.3: Rate counter after prescaler



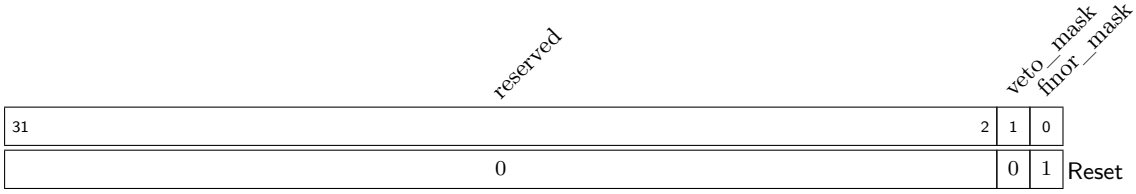
rate_counter_after_prescaler Rate counter after prescaler. Counts the occurancy of an algo (given by register address) in one luminosity segment.

Register 5.4: Rate counter post-dead-time



rate_counter_postdeadtime Rate counter post-dead-time. Counts the occurancy of an algo (given by register address) and L1A at the same bx in one luminosity segment.

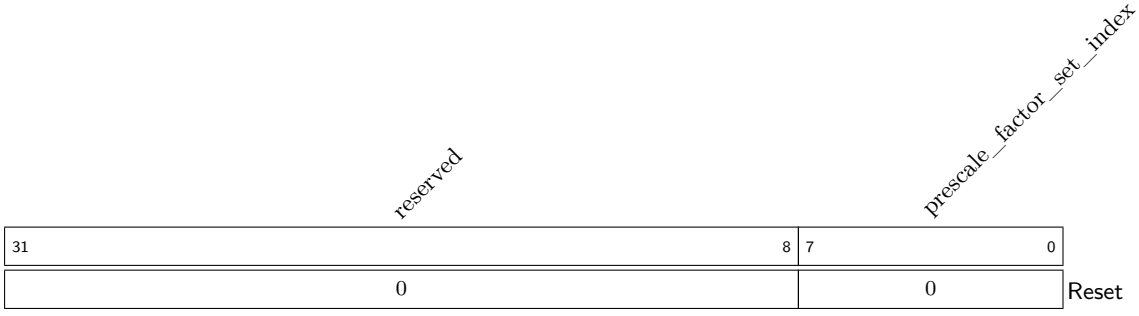
Register 5.5: Masks



veto_mask Selection of a veto (by an algo, given by register address) for veto-or.

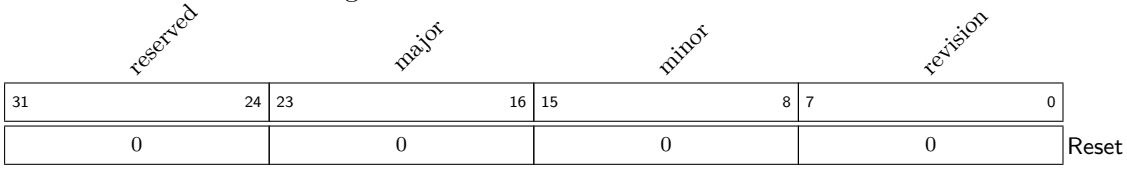
finor_mask Selection of an algo (given by register address) for final-or.

Register 5.6: Prescale factors set index



prescale_factor_set_index Index for a certain set of prescale factors.

Register 5.7: L1tm compiler version

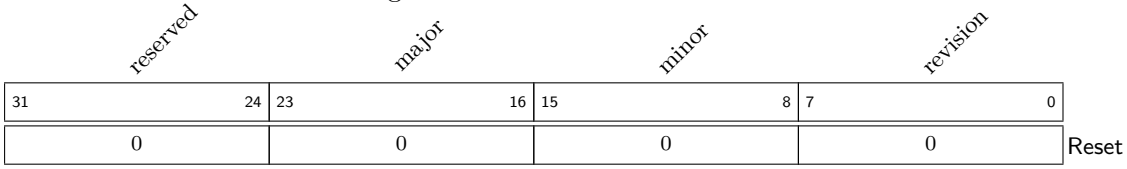


major Major version of L1tm compiler.

minor Minor version of L1tm compiler.

revision Revision version of L1tm compiler.

Register 5.8: GTL FW version

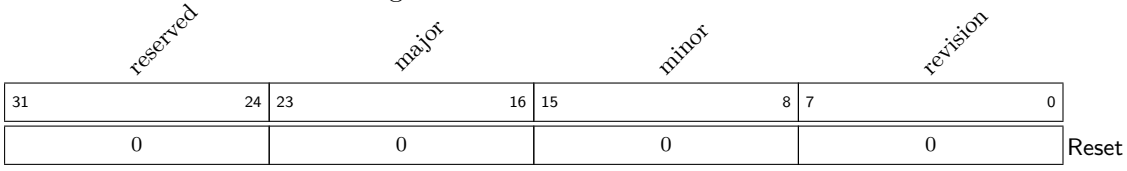


major Major version of GTL firmware.

minor Minor version of GTL firmware.

revision Revision version of GTL firmware.

Register 5.9: FDL FW version

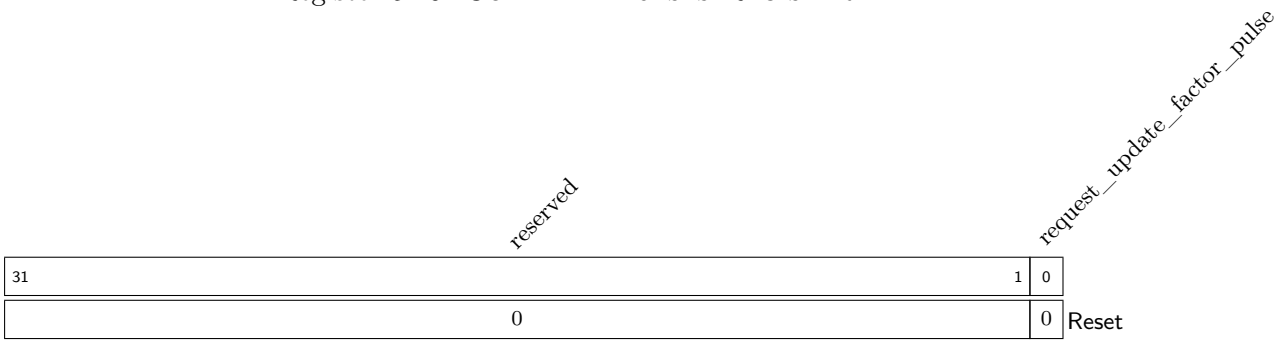


major Major version of FDL firmware.

minor Minor version of FDL firmware.

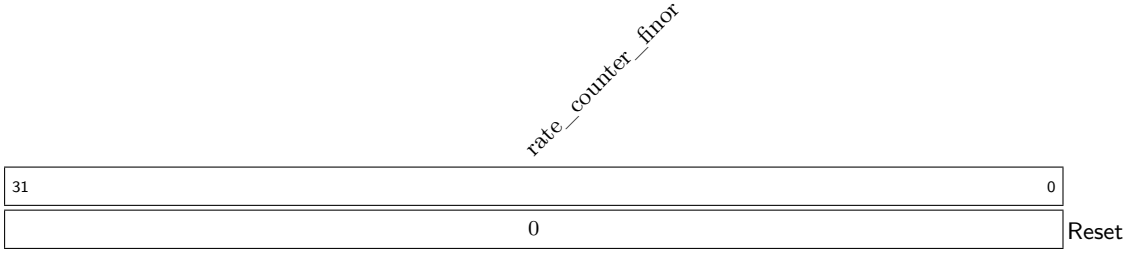
revision Revision version of FDL firmware.

Register 5.10: Command Pulses Register



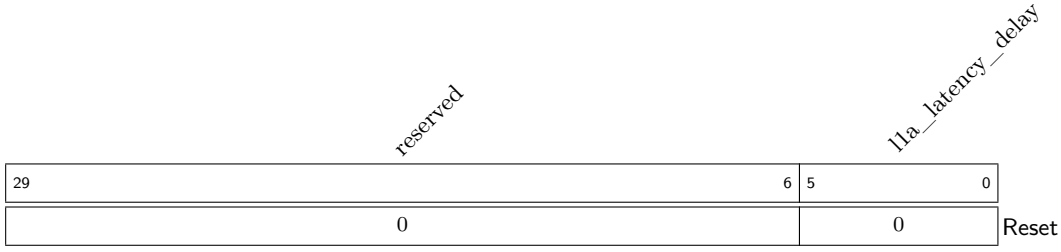
request_update_factor_pulse Sets the request signal for updating prescale factors high. Updating is done at the next "begin of luminosity segment".

Register 5.11: Rate counter finor



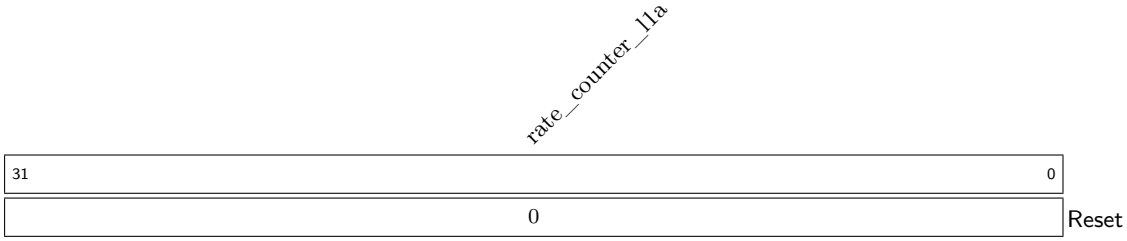
rate_counter_finor Rate counter finor. Counts the occurancy of finor in one luminosity segment.

Register 5.12: L1A latency delay



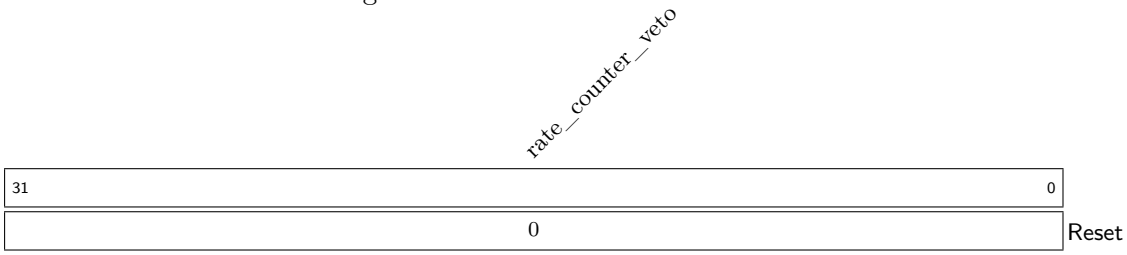
l1a_latency_delay L1A latency delay value (used for post-dead-time counter).

Register 5.13: Rate counter L1A



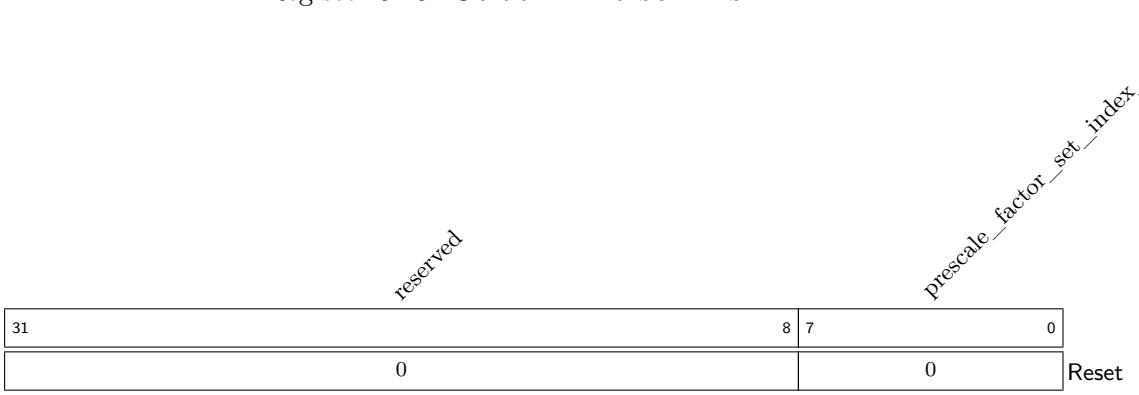
rate_counter_l1a Rate counter L1A. Counts the occurancy of L1A in one luminosity segment.

Register 5.14: Rate counter veto



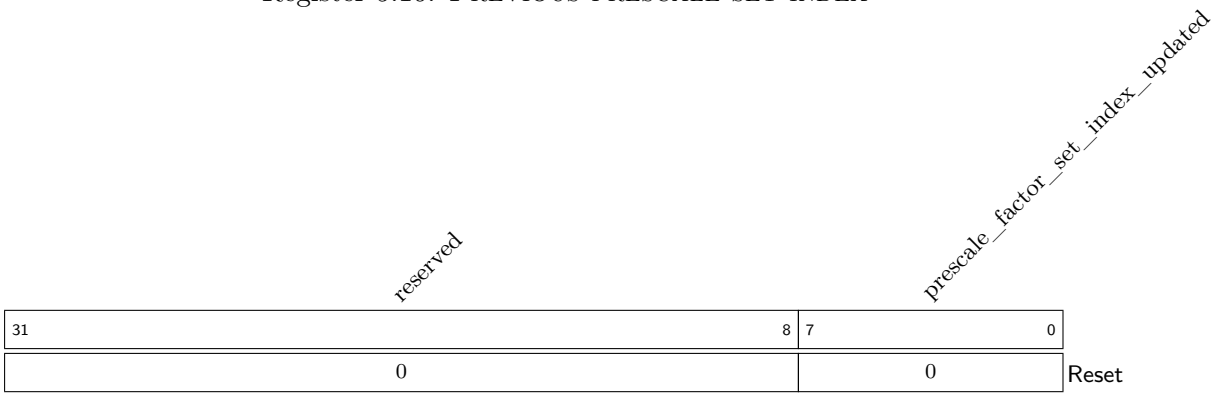
rate_counter_veto Rate counter veto. Counts the occurancy of veto in one luminosity segment.

Register 5.15: Current prescale set index



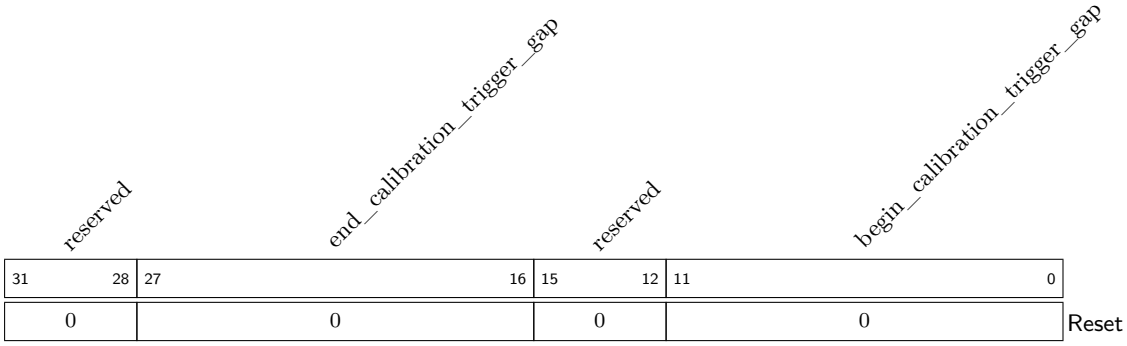
prescale_factor_set_index_updated Index for a certain set of prescale factors, which was "updated" with begin of current lumi-section.

Register 5.16: Previous prescale set index



prescale_factor_set_index_updated Index for a certain set of prescale factors, which was "updated" with begin of previous lumi-section.

Register 5.17: Calibration trigger gap



begin_calibration_trigger_gap Begin of calibration trigger gap (in Bx).

end_calibration_trigger_gap End of calibration trigger gap (in Bx).

5.4.2 Algo-bx-masks

Every Algorithm passes a logic where at every bunch-crossing of the orbit the Algorithm is enabled (or not). The algo-bx-masks are implemented as dual-port memories and loaded at the begin of run. The size of the algo-bx-masks memory is number of bunch-crossings per orbit for address length and number of Algorithms for data-depth (3564 [4096] x 512 bits). The address (bx-number) of the memory for masking the Algorithm is delivered by an address-counter for algo-bx-masks memory, which is reseted with a delay-able bcres signal, to get the correct relations between Algorithms and masks from memory.

5.4.3 Rate-counters

Every Algorithm has a rate-counters with 32 bits, because of the length of one luminosity segment period. There are counters before and after prescalers and post-dead-time counters. The counters before and after prescalers are incremented, if the Algorithm signal is in high state and a positive edge of LHC-clock occur. The post-dead-time counters are incremented, if the Algorithm signal is in high state (delayed by L1A latency delay), a L1A signal and a positive edge of LHC-clock occur. The content of a counter is updated into a register (for reading the counter value) and is reseted at the begin of a luminosity segment period. So there is one luminosity segment period time to read the registers with the counter values by software.

5.4.4 Prescalers

Every Algorithm has a prescaler with a prescale factor of 24 bits. The prescaler reduces the trigger-rate per Algorithm with a factor, so e.g. a factor of 2 passes through every second trigger. A prescale factor of 0 inhibits all triggers of the certain Algorithm. The factor is loaded into a register by software and updated at begin of a new luminosity segment period, if the update was enabled by software ('request_update_factor_pulse' was set in "command_-pulses" register). The prescaler works with the new factor. A register for "prescale factor set index" contains a value which represents a certain set of prescale factors. The content of this register is seen in the Readout-record too. The "prescale factor set index" is loaded into the register by software and updated at begin of a new luminosity segment period.

5.4.5 Finor-masks

Every Algorithm passes a Final-OR-mask, which enables the Algorithm for Final-OR. The Final-OR-masks are implemented as registers and loaded at the begin of a run.

5.4.6 Veto-masks

Every Algorithm passes a veto-mask, if at least one Algorithm, which is enabled by veto-mask, becomes high state, then Final-OR is disabled as long as the Algorithm is in high state. The veto-masks are implemented as registers and loaded at the begin of a run.

5.4.7 Finor

The Final-OR-signal is a disjunction of all Algorithms passed the Final-OR-bx-masks. An Algorithm enabled by veto-mask, disables the Final-OR. This is done on the FINOR-AMC502 module.

5.5 Implementation in firmware

The entity-declaration of `fdl_module.vhd` is shown in 5.3.

Listing 5 contains the entity-declaration of the `fdl_module.vhd`.

Listing 5: Entity declaration of `fdl_module.vhd`

```
entity fdl_module is
    generic(
        SIM_MODE : boolean := false; -- if SIM_MODE = true, "algo_bx_mask" is
            given by "algo_bx_mask_sim".
        PRESCALE_FACTOR_INIT : ipb_regs_array(0 to MAX_NR_ALGOS-1);
        MASKS_INIT : ipb_regs_array(0 to MAX_NR_ALGOS-1);
        PRESCALE_FACTOR_SET_INDEX_WIDTH : positive := 8;
        PRESCALE_FACTOR_SET_INDEX_REG_INIT : ipb_regs_array(0 to 1) := (others =>
             X"00000000");
        L1A_LATENCY_DELAY_INIT : ipb_regs_array(0 to 1) := (others => X"00000000"
            );
        CNTRL_REG_INIT : ipb_regs_array(0 to 1) := (others => X"00000000");
-- Input flip-flops for algorithms of fdl_module.vhd - used for tests of
    fdl_module.vhd only
        ALGO_INPUTS_FF: boolean := false
    );
    port(
        ipb_clk             : in std_logic;
        ipb_rst             : in std_logic;
        ipb_in              : in ipb_wbus;
        ipb_out             : out ipb_rbus;
-- ============================================================================
        lhc_clk             : in std_logic;
        lhc_rst             : in std_logic;
        bcres               : in std_logic;
        test_en             : in std_logic;
        l1a                 : in std_logic;
        begin_lumi_section  : in std_logic;
        algo_i              : in std_logic_vector(NR_ALGOS-1 downto 0);
        bx_nr_out : out std_logic_vector(11 downto 0);
        prescale_factor_set_index_rop : out std_logic_vector(
            PRESCALE_FACTOR_SET_INDEX_WIDTH-1 downto 0);
        algo_after_gtLogic_rop  : out std_logic_vector(MAX_NR_ALGOS-1 downto 0);
        algo_after_bxomask_rop     : out std_logic_vector(MAX_NR_ALGOS-1 downto
            0);
        algo_after_prescaler_rop      : out std_logic_vector(MAX_NR_ALGOS-1
            downto 0);
        local_finor_rop     : out std_logic;
        local_veto_rop      : out std_logic;
```

```
        finor_2_mezz_lemo      : out std_logic; -- to LEMO
        finor_preview_2_mezz_lemo      : out std_logic; -- to LEMO
        veto_2_mezz_lemo      : out std_logic; -- to LEMO
        finor_w_veto_2_mezz_lemo      : out std_logic; -- to tp_mux.vhd
        local_finor_with_veto_o       : out std_logic; -- to SPY2_FINOR
-- HB 2016-03-02: v0.0.21 - algo_bx_mask_sim input for simulation use with
    MAX_NR_ALGOS (because of global index).
        algo_bx_mask_sim    : in std_logic_vector(MAX_NR_ALGOS-1 downto 0)
    );
end fdl_module;
```

6 Readout-Process

The readout is done via TX-links of MP7 to AMC13.

List of Tables

List of Figures

Glossary

electron/γ = electron/gamma objects over Calo-Layer2 (VHDL: eg)

jet = jet objects over Calo-Layer2 (VHDL: jet)

tau = tau objects over Calo-Layer2 (VHDL: tau)

muon = muon objects over μGMT (VHDL: muon)

ET = Scalar sum of transverse energy components over Calo-Layer2 (VHDL: ett)

ETTEM = Scalar sum of transverse energy components from ECAL only over Calo-Layer2 (VHDL: ettem)

MBTxHFy = Minimum bias HF bits (VHDL: MBT0HFP, MBT0HFM, MBT1HFP, MBT1HFM)

HT = Magnitude of the vectorial sum of transverse energy of jets (hadronic) over Calo-Layer2 (VHDL: htt)

TOWERCOUNT = tower counts (VHDL: towercount)

ET_{miss} = 2-vector sum of transverse energy over Calo-Layer2 (VHDL: etm)

HT_{miss} = Missing Total transverse energy of jets over Calo-Layer2 (VHDL: htm)

$\mathbf{ET}^{HF}_{miss}$ = 2-vector sum of transverse energy including HF over Calo-Layer2 (VHDL: etmhf)

$\mathbf{HT}^{HF}_{miss}$ = Missing Total transverse energy of jets including HF over Calo-Layer2 (VHDL: htmhf)

ASYMET = Asymmetry of ET over Calo-Layer2 (VHDL: asymet)

ASYMHT = Asymmetry of HT over Calo-Layer2 (VHDL: asymht)

ASYMETHF = Asymmetry of ET including HF over Calo-Layer2 (VHDL: asymethf)

ASYMHTHF = Asymmetry of HT including HF over Calo-Layer2 (VHDL: asymhthf)

CENTx = Centrality bits [7:0] over Calo-Layer2 (VHDL: cent7, cent6, ...)

p_{T} = transverse momentum of muon objects(VHDL: pt)

E_{T} = energy of calorimeter objects (VHDL: et)

η = pseudo-rapidity position (VHDL: eta)

φ = azimuth angle position (VHDL: phi)

isolation = isolation information (VHDL: iso)

quality = quality information (VHDL: qual)

Acronyms

DAQ Data Acquisition

DM Delay Manager Module

FDL Final Decision Logic Module

GTL Global Trigger Logic Module

ROP Readout-Process Module

TCM Timing Counter Manager Module

TCS Trigger Control System

GCT Calorimeter Trigger Layer-2

GMT Global Muon Trigger

GT Global Trigger

References