**Chapter 3**

# CRU common firmware

|  |  |
| --- | --- |
| 2 | CRU firmware |

# 3.1 Common requirements

The CRU firmware is divided into two parts. The first part is the common firmware which (i) provides the interfaces to PCIe, trigger, and timing, and up to 24 front-end links via the GBT protocol, (ii) provides the possibility to read out all detectors in ‘raw mode with no data processing in the CRU, (iii) allows reference clock and trigger signals distribution, and (iv) permits FEE configuration. The second part is the user logic which is only needed for those detector systems that need detector-specific data processing for instance baseline correction or zero suppression. An important ALICE requirement is to be able to switch between raw-mode and user-logic at any moment without reloading different firmware versions. Self-testing capabilities to ease commissioning and system maintenance are implemented. They are detailed later in this paper. From a system point of view, the different requirements for the common firmware are due to the different GBT bus mode (packet or stream), the information it is supposed to carry, the integration of a user logic or not, and the type of slow control protocol required to configure the FEE. It is important to mention that when the data reach the DMA engine in the CRU the format is identical, regardless of the data-taking configuration of the card. Having the same data format simplifies the firmware logic of the CRU as well as the software required for physics analysis. The GBT links are used to send downstream trigger messages and/or the reference clock to the FEE, while upstream they are used for data readout and optionally to acknowledge specific slow control transactions. The CRU firmware supports the operation of the GBT links in GBT-mode (80 bits of payload and 32 bits of forwarding error correction) or in wide-bus (112 bits of payload and no forward error correction). For almost all the GBT detectors the standard GBT-model provides sufficient data bandwidth.

# 3.2 Firmware description

An overview of the firmware is given in Fig. 3.1. The main parts are shown: the FEE interface through GBT links, the TTS interface, the board support Package (BSP), the data path, and the PCIe endpoints. Starting from the front-end side on the left, the GBT\_wrapper interface is shown; it is the interface with the FEE. On the downstream path (CRU to FEE), depending on the detector requirements or test requirements, several sources can be selected to supply the GBT\_wrapper. These are the Trigger and Timing System interface, the Dedicated Data Generator (DDG), or the slow control.

## 3.2.1 Board Support Package

The board support package features several I2C masters that are directly controlled through the PCIe interface. They allow the readout of the various optical transceiver parameters such as temperature or optical power, the settings of external PLLs, and access to the board serial number stored in an EEPROM. Additionally, it permits access to the FPGA serial number (fixed by the FPGA manufacturer) and monitoring of the FPGA die temperature. The BSP functionality also includes the reconfiguration of the QSPI flash and the possibility to trigger an FPGA reboot from the slow control (through the PCIe interface). The chosen strategy is to have two reserved areas in the flash memory, one for a golden and one for the application firmware. The golden firmware will never be modified outside of the lab, while the application firmware can be modified when deployed on-site. At cold startup, the FPGA always

|  |  |
| --- | --- |
| 2 | CRU firmware |

boots on the golden firmware. Then, by issuing a PCIe BAR command, specifying the memory offset to use, it is possible to load the application firmware in the FPGA. In the case of a configuration failure (e.g. loss of connection, power cut, or faulty firmware) the CRU can be easily recovered.

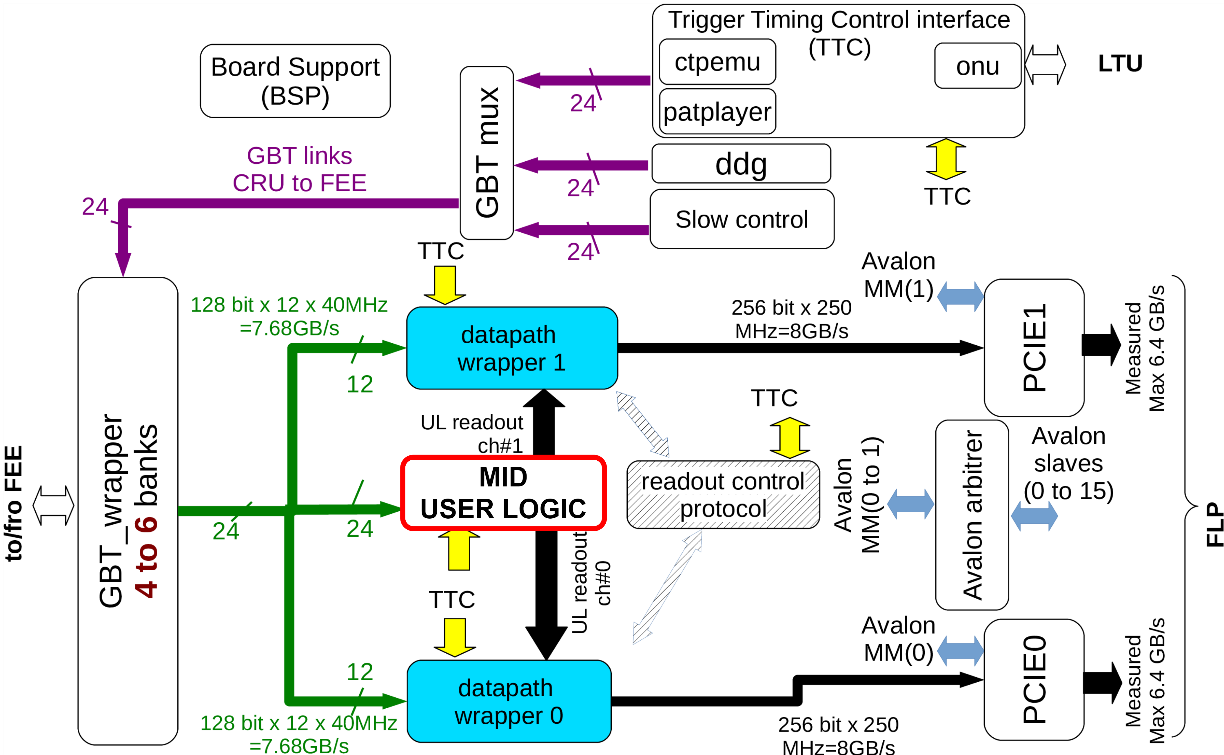


Figure 3.1: cru firmware overview [ ]

## 3.2.2 GBT wrappers

The GBT\_wrapper is a modified version of the GBT-FPGA developed at CERN [3, 6], see Fig. 5. The main differences are that (i) it has a user data path operating at 240MHz (six times the machine clock), (ii) the clock domain crossing between the transceiver domain and the user domain is achieved with timing constraints instead of a phase scan at link start-up, (iii) dynamic switching is possible between GBT-mode and wide-bus mode to cover all detector requirements and (iv) the test data pattern generator is shared between all links to save resources. Moreover, the GBT\_wrapper permits external (with optical fibers) and internal (inside the FPGA transceivers) loop-back tests which allow the validation of the CRU-FEE communication and the CRU datapath operation once installed in the system. In external loop-back, the data generator enables the emission of representative data towards the FEE that can be looped back by them into the CRU; while in internal loopback mode, this feature allows stressing the system without relying on the availability of detector FEE. The strategy used to maintain a constant latency and to avoid the phase scanning on the transmission path (CRU to FEE) is (i) to rely on the zero delay buffer provided by the hardware PLL and to feed the extracted transmission delays due to the PCB in the constraint file, and (ii) to use the 6 time faster rate to properly sample and transfer the data from one clock domain to the other (120 bits transferred at 40 MHz). On the receiving, which is used only for readout, a non-constant latency can be accepted, and thus a FIFO was implemented to cope with the clock domain transfer from the recovered clock domain to the user part clock domain. This solution was extensively tested across several scenarios and proved to be reliable. The scenarios tested were: CRU reference clock switching between local and remote clock source, warm reboot (FPGA reconfiguration), and cold reboot (CRU and FLP turned off and rebooted).

|  |  |
| --- | --- |
| 2 | CRU firmware |

## 3.2.3 TTC interfaces

The first is the Optical Network Unit (ONU) [9, 10] which recovers the machine clock from the PON and forwards it through the GBT to the FEE. The ONU is also used to receive the trigger and timing message at each clock cycle (trigger bits, bunch crossing number, Heart Beat ID) from the central system. The 200 bit word contains the trigger information on its lower 116 bits and the trigger decision message on its 80 bits upper part. The CRU uses the upstream direction to send the HBACK or HBNACK trigger message. As the optical network is passive, the upstream communication is time multiplexed and a message can only be sent every 125 ns times the total number of ONUs connected on the PON. The second component is a trigger emulator (ctpemu) that is used for tests and system diagnostic purposes. It can produce trigger messages, like the ones provided through the ONU, and simulate readout flow control by producing HBa and HBr commands. The third component is the pattern player (patplayer) that can generate a programmable sequence to be transmitted to the FEE, it is started by a trigger bit issued either by the ONU or ctpemu. The fourth is the trigger router (trgrouter) which remaps, duplicates and forwards some trigger bits received via the PON from the CTP to the FEE boards via the GBT links.

## 3.2.4 Detector Data Generator

The DDG is the component in the CRU able to emulate detector behavior and data throughput. This component plays a main role during the test and validation of the CRU firmware and the readout software chain when the detector hardware is not yet widely available. The DDG has different configuration parameters and it can be dynamically configured to produce either streaming or packet type data. The data packet can be produced for different trigger types with fixed or random packet duration, generating a realistic detector throughput. The DDG can be used to test the firmware at any moment in time. Configuring the GBT links using the internal loop-back connection makes the injection of DDG data in the system possible without the requirement to change the physical optical connection at the card input. DDG is a powerful self-test feature to verify the correct behavior of the hardware and the software without relying on external hardware elements like the FEE.

## 3.4.5 Slow Control

The SWT protocol has been introduced to increase the bandwidth for the slow control operation. It uses the GBT-DATA path to deliver up to 3200 Mb/s (80 bits per 40MHz clock cycle) to the FEE whereas the GBT-EC path provides 80 Mb/s. In practice the slow control read/write speed is limited to 36 Mb/s by the time taken for software to access the PCIe BAR. To send the detector configuration data over the GBT link using the SWT protocol, the CRU must be configured to switch to the SWT traffic with the GBT-MUX component (see Fig. 4). This is a static selection, as there is no dynamic packet switching for the downstream path. In the opposite direction, from the FEE towards CRU, the SWT and FEE information is interleaved within the same link. In order to distinguish GBT words that contain physics information from the control words, like the SWT, the CRU uses two types of information, the isdatasel flag decoded from the GBT header and a part of the GBT word embedded in the GBT data field. When the detector sends physics data the flag isdatasel is set to 1 and the whole 80-bit data field is used to transfer the information. When this flag is 0 the CRU considers the GBT word a control word

|  |  |
| --- | --- |
| 2 | CRU firmware |

and it uses the four most significant bits of the data field to distinguish between the different control words (Fig. 7):

* IDLE: 0x0. This word is used to pause the data transfer in case the FEE has no data to send to the CRU.
* SOP: 0x1. This word identifies the Start Of Packet during a GBT data transfer from a GBT packet type detector.
* EOP: 0x2. This word identifies the End Of Packet during a GBT data transfer.
* SWT: 0x3. Single Word Transfer, it is used to identify a configuration word.

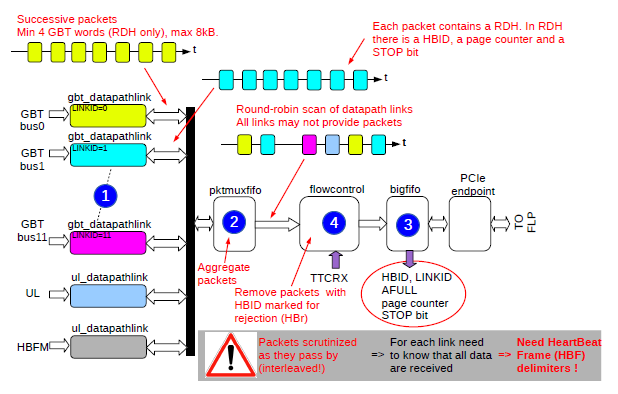
The CRU extracts the SWT information from the data stream before it reaches theDMAengine and stores it in a dedicated FIFO which is accessed by DCS.

|  |  |
| --- | --- |
| 2 | CRU firmware |

## 3.4.6 Datapath wrappers

Two datapath wrapper blocks are implemented in the firmware. They receive the trigger messages, collect and aggregate the FEE data provided by the GBT link or use directly the user logic input if implemented. They also provide monitoring information to the readout control protocol component (see Fig. 8). The first task of each datapath\_wrapper is to receive in parallel the data either from up to 12 GBT busses, from the readout protocol component (trigger acknowledge or decision message) and/or from one user logic link. The gbt\_datapathlink is compatible with stream or packet type format. When selecting the stream mode this component constructs data packets, by chopping the data stream and inserting the Raw Data Header (RDH). The RDH describes the readout packet content and contains the Heart Beat ID (HBID), the Link ID, the page counter and the stop bit as well as other information. For each Link ID the page counter gives the packet identification within the corresponding HBF transmitted and the stop bit indicates whether the last packet for this HBF is being transmitted.

The ul\_datapathlink receives already correctly formatted packets, either from the User Logic or from the Readout Protocol block, i.e. Heart Beat Acknowledge and Decision Messages (HBFM). At the output of this first stage the packets have a maximum size of 8 KB. The second stage, named pktmuxfifo, performs data aggregation by scanning in a round-robin based manner possible data sources (gbt\_datapathlink and ul\_datapathlink) and collects data packets. At the output of this stage, the packets from the various links are interleaved. If required by the CTP, this is followed by the removal of all packets from the data flow with a HBr message (number 4 in Fig. 8). Then, the packets are stored in a large buffer (bigfifo, 16 kwords of 256 bits)) to be made available to the PCIe endpoint. While being stored, the packets are scrutinized and useful parameters (HBID, LINKID, FIFO status) are presented to the readout control protocol component. The readout protocol uses the information provided by both datapath\_wrapper blocks to check the interleaved packets. The HBF reception is declared successful only if for each LINKID included in the readout, start (page counter is 0 in RDH) and stop packets (stop bit is 1 in RDH) were received in consecutively and properly stored in the bigfifo buffers before a pre-defined timeout for reception elapsed. Then a HBACK or HBNACK message is transmitted to the CTP, which assembles the messages from all CRUs and updates the HBa/HBr messages to communicate whether a given HBF should be maintained or be deleted in the FLP. [8, 13].



**Figure 2.2**: Geometry of the readout electronic

### 3.4.6.1 Raw Data Header

|  |  |
| --- | --- |
| 2 | CRU firmware |

## 3.4.7 Readout protocol

|  |  |
| --- | --- |
| 2 | CRU firmware |

## 3.4.8 PCIe DMA

In the firmware the CRU input data stream is divided between the two PCIe endpoints in order to avoid dynamic switching of the data flow between the two PCIe endpoints. Thus, half of the GBT links implemented in the CRU are connected to one single endpoint via its datapath-wrapper instance that receives data from a maximum of 12 GBT links. In this way, the data throughput is evenly distributed between the two PCIe gen3 x8 interfaces.

The communication with the software [14] happens through the PCIe BAR interface. There are two BAR interfaces: BAR0 and BAR2. BAR0 is dedicated to DMA operations, it passes the page descriptors and monitors the status of the data taking, while BAR2 is used to access the card configuration and to monitor the other components of the firmware. The CRU transfers the physics data into the server memory using DMA transfers. In order to do so the CRU needs to know in which buffer data should be stored. For that, the software prepares (or allocates) the buffers in the FLP memory and then passes the addresses of each of these buffers to the CRU. This information is called the DMA page descriptors. Therefore, to maximise the throughput by reducing wait states, the descriptors are prepared in advance by software and delivered through a FIFO to the CRU DMA engine. If the FIFO holding the page descriptors becomes empty, the DMA transaction are paused and data are dropped in full DMA pages until new page descriptors are provided. To reduce the software interaction with the hardware, the concept of super-pages has been introduced. A super-page is a buffer of contiguous spaces in memory of usually 1 or 2MB size (configurable based on the event size of the detector). The software stores the address of this buffer in the CRU descriptor FIFO and the DMA engine fills up the buffer with the data coming from the input links. Once the super-page is full a new descriptor is fetched and data is stored in a new super-page. It is the responsibility of the CRU firmware to divide the super-page in smaller DMA pages, of a maximum size of 8 KB. The CRU can collect the data from a maximum of 24 GBT links. During data taking the CRU is loaded with 128 super-page descriptors for each GBT link active in the data taking. Data coming from different links are organized in dedicated super-pages, therefore the software fetches data from specific GBT links in dedicated link memory buffers.

Data in the FLP memory is grouped before being sent to the EPN for further analysis. The grouping is done at the boundary of Time Frames. To simplify the job of the software the HBFs are stored by the CRU in dedicated super-pages. In this way it is simpler to group the different buffers belonging to a single Time Frame and send them over RDMA (Remote DMA) to the EPN farm. Assigning a dedicated buffer for each GBT link removes any dependency between them, improving the stability of the system and the total throughput.

## 3.4.9 Avalon arbitrer

|  |  |
| --- | --- |
| 2 | CRU firmware |

## 3.4.10 User logic

**Port specifications**

The port speciations of the user logic are defined below.

**TTC Downlink Ports (from LTU to UL direction)**

|  |  |  |  |
| --- | --- | --- | --- |
| Port Name | Direction | Clock Domain | Description |
| ttc\_rxclk | input | 240 MHz | input clock |
| ttc\_rxrst | input | ttc\_rxclk | reset |
| ttc\_rxready | input | ttc\_rxclk | ready |
| ttc\_rxvalid | input | ttc\_rxclk | valid |
| ttc\_rxd | input | ttc\_rxclk | data |

**GBT Uplink Ports (from FEE to UL direction)**

|  |  |  |  |
| --- | --- | --- | --- |
| Port Name | Direction | Clock Domain | Description |
| gbt\_rx\_ready | input | ttc\_rxclk | gbt ready |
| gbt\_rx\_isdatasel | input | ttc\_rxclk | gbt payload flag |
| gbt\_rx\_valid | input | ttc\_rxclk | gbt payload valid |
| gbt\_rx\_data | input | ttc\_rxclk | gbt payload |

**GBT Downlink Ports (from UL to FEE direction)**

|  |  |  |  |
| --- | --- | --- | --- |
| Port Name | Direction | Clock Domain | Description |
| gbt\_tx\_ready | input | ttc\_rxclk | gbt ready |
| gbt\_tx\_isdatasel | input | ttc\_rxclk | gbt payload flag |
| gbt\_tx\_valid | input | ttc\_rxclk | gbt payload valid |
| gbt\_tx\_data | input | ttc\_rxclk | gbt payload |

|  |  |
| --- | --- |
| 2 | MID readout chain |

**LED Ports (from UL to CRU)**

|  |  |  |  |
| --- | --- | --- | --- |
| Port Name | Direction | Clock Domain | Description |
| BlueGreenRed\_LED\_1 | output | n/a | LED 1 |
| BlueGreenRed\_LED\_2 | output | n/a | LED 2 |
| BlueGreenRed\_LED\_3 | output | n/a | LED 3 |
| BlueGreenRed\_LED\_4 | output | n/a | LED 4 |

**Avalon-MM Slave Port - PCIe BAR Read/Write access (UL from/to CRU)**

|  |  |  |  |
| --- | --- | --- | --- |
| Port Name | Direction | Clock Domain | Description |
| mms\_clk | input | 100 MHz | input clock |
| mms\_reset | input | mms\_clk | reset |
| mms\_waitreq | output | mms\_clk | wait flag |
| mms\_addr | input | mms\_clk | address bit |
| mms\_wr | input | mms\_clk | write flag |
| mms\_wrdata | input | mms\_clk | write data |
| mms\_rd | input | mms\_clk | read flag |
| mms\_rdval | output | mms\_clk | read valid |
| mms\_rddata | output | mms\_clk | read data |

**PCIe Link Ports (UL from/to CRU)**

|  |  |  |  |
| --- | --- | --- | --- |
| Port Name | Direction | Clock Domain | Description |
| fclk | output | ttc\_rxclk | clock |
| fval | output | ttc\_rxclk | valid |
| fsop | output | ttc\_rxclk | start of packet |
| feop | output | ttc\_rxclk | end of packet |
| fd | output | ttc\_rxclk | packet data |
| afull | input | ttc\_rxclk | full flag |

# 3.3 Summary

# 3.4 References