1

TRIGGER & DATA CONCENTRATOR MODULE

REFERENCE MANUAL

D. Calvet

Version 1.6

1 GENERAL OVERVIEW

The Trigger and Data Concentrator Module – TDCM – is a versatile electronic back-end module for detector readout systems based on the AFTER [1], AGET [2], ASTRE [3] family of chips. It performs the following functions: 1) distribution of a primary reference clock to a set of front-end cards or modules, 2) distribution of a common trigger signal and global synchronization of the front-ends, 3) configuration and read-back of the front-end and back-end sides, 4) data collection from the front-end, 5) slow control and monitoring of the front-end and back-end sides.

The TDCM is primarily designed for the neutrino-less double beta decay experiment PandaX-III, but it is also intended to use it for other applications that have comparable requirements.

The TDCM can be used as a master device where it uses its own local reference clock, or it can be used as a slave device where it receives the primary clock and trigger from another master device, e.g. another TDCM or some other module. Master and slave TDCMs are noted M-TDCM and S-TDCM respectively. A system may include at most one M-TDCM, and can have one or several S-TDCMs.

Depending on hardware configuration, a TDCM can control up to 32 front-end units. In PandaX-III, the basic front-end unit is a front-end card, FEC, which is equipped with 4 AGET chips and FPGA logic to communicate with the TDCM. In T2K-II, the elementary front-end unit can be a front-end card, called the ARC, which is equipped with 4 AFTER or AGET chips and FPGA logic for communication with the TDCM, but it can also be a front-end module, which is composed of two front-end cards, also called FEC, equipped with 8 AFTER chips each, and a front-end mezzanine card, FEM, which includes the FPGA logic for controlling the two FECs and communicating with the TDCM.

In this document, each front-end unit is referred to as a "front-end', noted FE", although the underlying hardware may be a (PandaX-III) FEC, an ARC, or a (T2K-II) FEM and its (T2K-II) FECs. There are nonetheless some differences between the different FEs, and specific features are mentioned when appropriate. Using PandaX-III FEC (256 channels), a TDCM can read out up to 8192 channels. In T2K-II, the maximum channel count per TDCM is 9216 and 36,854 using the ARC (288 channels) and the FEM and its two FECs (576 channels each) respectively.

The TDCM has a number of interfaces to connect to its partner devices. Four NIM level inputs are available to input a reference clock, a baseband trigger signal, and two other signals (e.g. timestamp reset and clock synchronization). Two NIM level outputs are provided to output a busy signal and a user defined signal. Three general purpose TTL level input and three outputs are also available. Alternatively, the primary clock,

trigger and synchronization signals can be provided over a RJ45 cable carrying serially encoded information over LVDS, or an optical interface can be used. Both the RJ45 electrical interface and optical port use a proprietary protocol and encoding. Multiple TDCMs can be cascaded using their master and slave RJ45 port. The TDCM has four high speed transceivers connected to SFP cages. One of these SFP ports is normally used to interface to a control and DAQ PC over Gigabit Ethernet (optical, or electrical using a GBIC – Gigabit Ethernet Interface Converter). The three other SFPs have no specifically assigned function and are available to the user. A second Gigabit Ethernet port (RJ45) is also available. A PCI-Express (Gen 2 x 4 lanes), is also present on the TDCM, but it is not supported by default (and even it is currently untested).

The interface to the front-end devices normally uses optical media, although a physical layer based on copper may also be developed in the future if it is needed.

2 BOARD DESCRIPTION

The TDCM is a composite assembly made of three types of electronic boards:

- An FPGA module. For a fast and cost effective development, a commercially available FPGA module was selected: the Mercury ZX1 from Enclustra [4].
- A main carrier board. This custom made board is designed to carry a Mercury ZX1 FPGA module and up to two physical layer mezzanine cards. For development, it is also possible to use the evaluation platform sold by Enclustra, the Mercury PE1. With this platform, some of the interfaces of the full size TDCM are not available, and only one physical layer mezzanine card can be used. This configuration is intended for tests and development only.
- Physical layer mezzanine cards. This card is also a custom design. It houses the physical layer part of the interface to the front-ends, for example optical transceivers. Other types of physical layer mezzanine cards may be designed in the future if some application require copper media, or some specific type of optical transceiver. A physical layer mezzanine card can have up to 16 transceiver/interface ports.

2.1 MERCURY ZX1 FPGA MODULE

A complete description of the Mercury ZX1 FPGA Module is available in [4]. The model that has initially been used for the development of the TDCM is ME-ZX1-45-2C-D10-P or ME-ZX1-45-2I-D10-P. Later versions have switched to a cheaper model from the same family: ME-ZX1-30-2I-D10 or ME-ZX1-30-2C-D10. The main difference between the XC7z030 and the XC7z045 part is that only 4 MGT transceivers at up to 6.6 Gbps are available instead of 8 MGT at up to 10 Gbps. Consequently, the PCIE interface is not available on the TDCM equipped with a ME-ZX-30 FPGA module.

Alternatively, the ME-ZX1-35-1C-D10 or ME-ZX1-35-1I-D10 may also be adequate – although this has not been checked.

2.2 MERCURY PE1 EVALUATION KIT

A development version of the TDCM is implemented on the commercially available Mercury PE1 Evaluation kit made by Enclustra. This hardware configuration can be equipped with only one physical layer mezzanine card and it cannot be interfaced to a master trigger and clock distributor. This setup is normally only used for firmware and software development. A picture of the TDCM mock-up built with the Enclustra PE1 evaluation kit is shown in Fig. 1. Connection to the data acquisition PC can be done with the Gigabit Ethernet port of the Mercury PE1 board but this configuration does not support Jumbo frames. Alternatively, an adapter board that converts the PCI Express connector into several SFP+ and SMA connectors can be purchased from Enclustra. This hardware allows the implementation of Gigabit Ethernet using 1000-baseX physical

layer and can support Jumbo frame with the appropriate MAC device. This configuration of the TDCM mock-up is shown in Fig. 2.
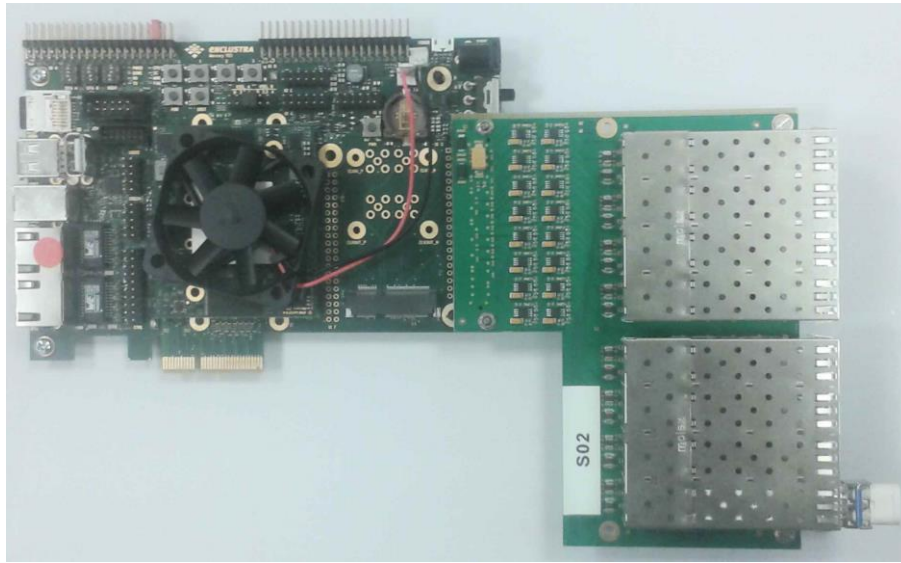

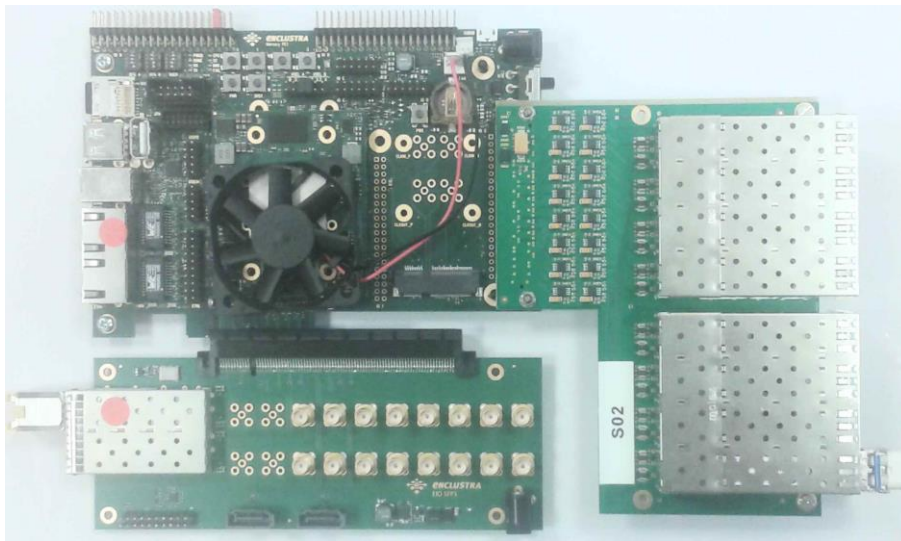Fig. 1.   TDCM mock-up on Enclustra PE1 evaluation kit.

Fig. 2.   TDCM mock-up on Enclustra PE1 evaluation kit with the PCIe breakout.

The physical layer mezzanine card is not mandatory, but in this case, no FE can be connected. If present, the physical layer mezzanine card must be screwed on the PE1 evaluation kit using M2.5 screws and 8 mm standoff. On the front side of the physical layer mezzanine card, M3 screws and 18 mm standoff shall be used. A heat sink and a fan is required for the FPGA module. When using the on-board Ethernet, connect the host to the RJ45 port closest to the USB connector on the front panel side of the PE1 evaluation kit (marked with a red sticker on the picture in Fig. 1). When the PCIe breakout board is used, a GBIC should be used, in the SFP cage closest to the PCIe

connector (see Fig. 2). Also, in this configuration, the board must be powered from the circular jack on the PCIe breakout board instead of the power jack of the PE1 board.

For both hardware variants, jumper VSEL_A should be set to position "A" while jumper VSEL_B should be set to position "B". Switches CFG_A should be set OFF; OFF; OFF; ON. This corresponds to a default boot from the MicroSD flash. Switches CFG_B should be set OFF; OFF; ON; ON. The PE1 evaluation kit contains 4 user defined DIP switches and 4 push buttons. Their usage is shown in Table 1.

Table 1 . Usage of PE1 card user DIP switches and push buttons.

Resource	TDCM mock-up usage
User switch #1	DBG_SEL<0>
User switch #2	DBG_SEL<1>
User switch #3	DBG_SEL<2>
User switch #4	BIOS_N
Push button #0	RESET_N
Push button #1	MAN_TRIG_N
Push button #2	MAN_SYNCH_N
Push button #3	MAN_CLR_N

For proper operation of the TDCM mock-up on the PE1, a jumper must be installed between pin #35 and pin #36 of connector IOA. This connects and internally generated 25 MHz clock to the primary external clock input of the TDCM mock-up.

The firmware of the TDCM mock-up defines 16 debug output pins that can be used to spy-on internal signals with an external oscilloscope or logic analyzer. Up to 8 groups of 16 signals can be observed according to the position of user switches DBG_SEL<3..0>. The observable signals are mapped to FPGA I/O pins connected to connector IOA as shown in Table 2.

Table 2 . Location of debug output pins (connector IOA).

Resource	Pin#	Pin#	Resource
DBG<0>	31	32	DBG<1>
DBG<2>	29	30	DBG<3>
DBG<4>	27	28	DBG<5>
DBG<6>	25	26	DBG<7>
DBG<8>	21	22	DBG<9>
DBG<10>	19	20	DBG<11>
DBG<12>	17	18	DBG<13>
DBG<14>	15	16	DBG<15>

The observable internal signals are defined in the VHDL source code of the TDCM mock-up and is changed as needed during firmware development.

Connection to a RS232 terminal console is done through a USB converter on-board the PE1 evaluation kit. A micro-USB cable should be connected on the connector close to the power supply input. It may be necessary to install the FTDI drivers for the USB-RS232 bridge. Console settings are 115.000 bauds, 8 bit, no parity. Note that the USB section of the PE1 board and some other logic is powered from USB. In some cases, it can be necessary to power off the PE1 board and disconnect it from the USB to entirely reset the board.

The TDCM mock-up on the PE1 evaluation kit requires a specific version of the TDCM firmware and software (note also that it is different depending on whether the PCIe breakout board is used or not), but besides this, the mock-up supports almost exactly the same set of commands, and uses the same tools and methods as the full-size TDCM.

2.3 TDCM CARRIER CARD

The TDCM carrier card is a 6U form factor electronic board designed to accommodate a Mercury ZX1 FPGA module, one or two physical layer mezzanine cards (with up to 16 links each). The width of TDCM assembly with a physical layer mezzanine on each side is 8 cm (i.e. 16 HP or the equivalent of 4 slots in a Eurocard chassis). The TDCM carrier receives an external 12V power input and distributes the adequate power voltages to the FPGA module and the mezzanine cards. The TDCM carrier includes a large variety of I/O's to interface to a data acquisition and control PC, to connect to a master trigger and clock distributor module, and to optionally cascade several TDCMs.
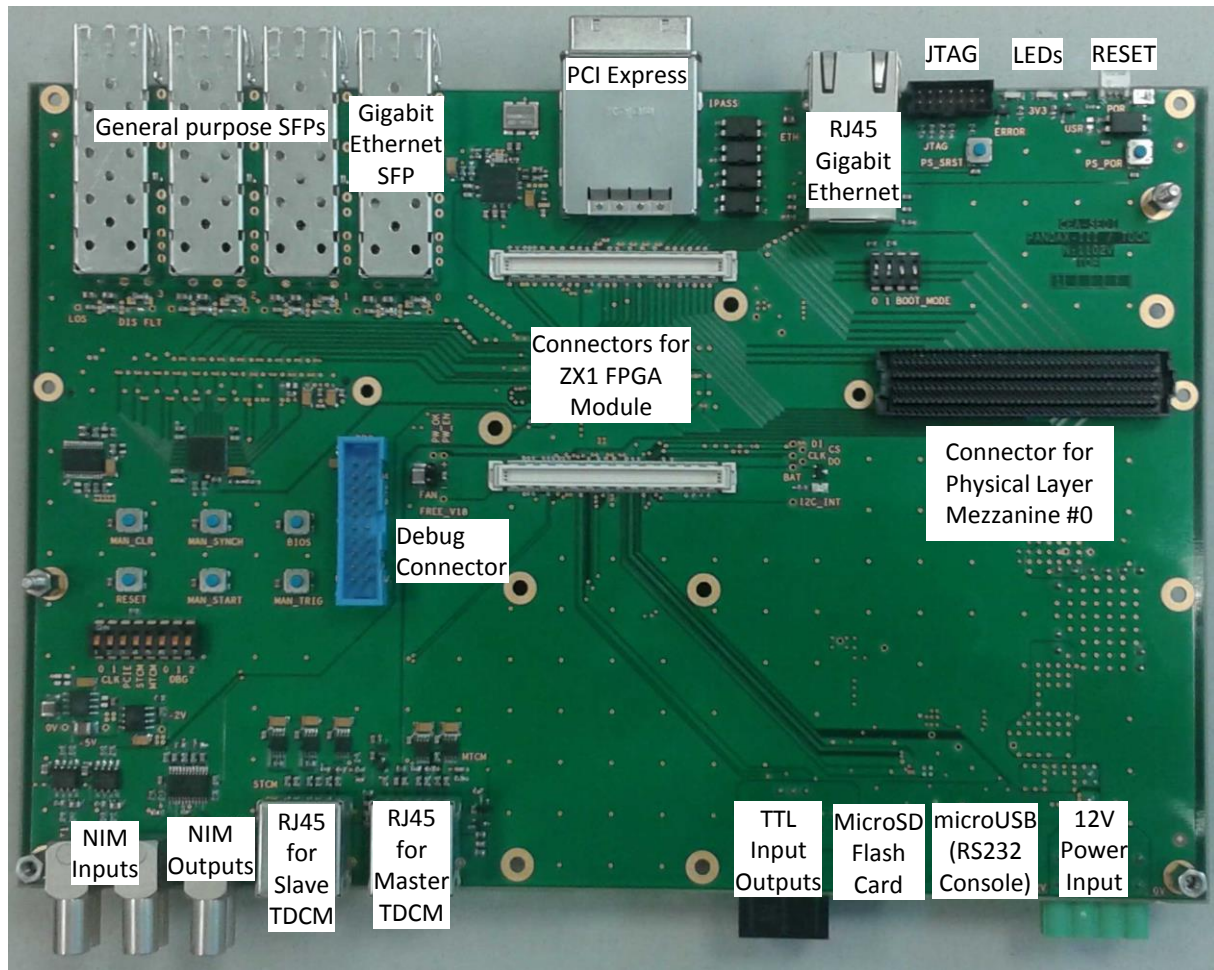
Fig. 3.   Front view of the TDCM carrier.

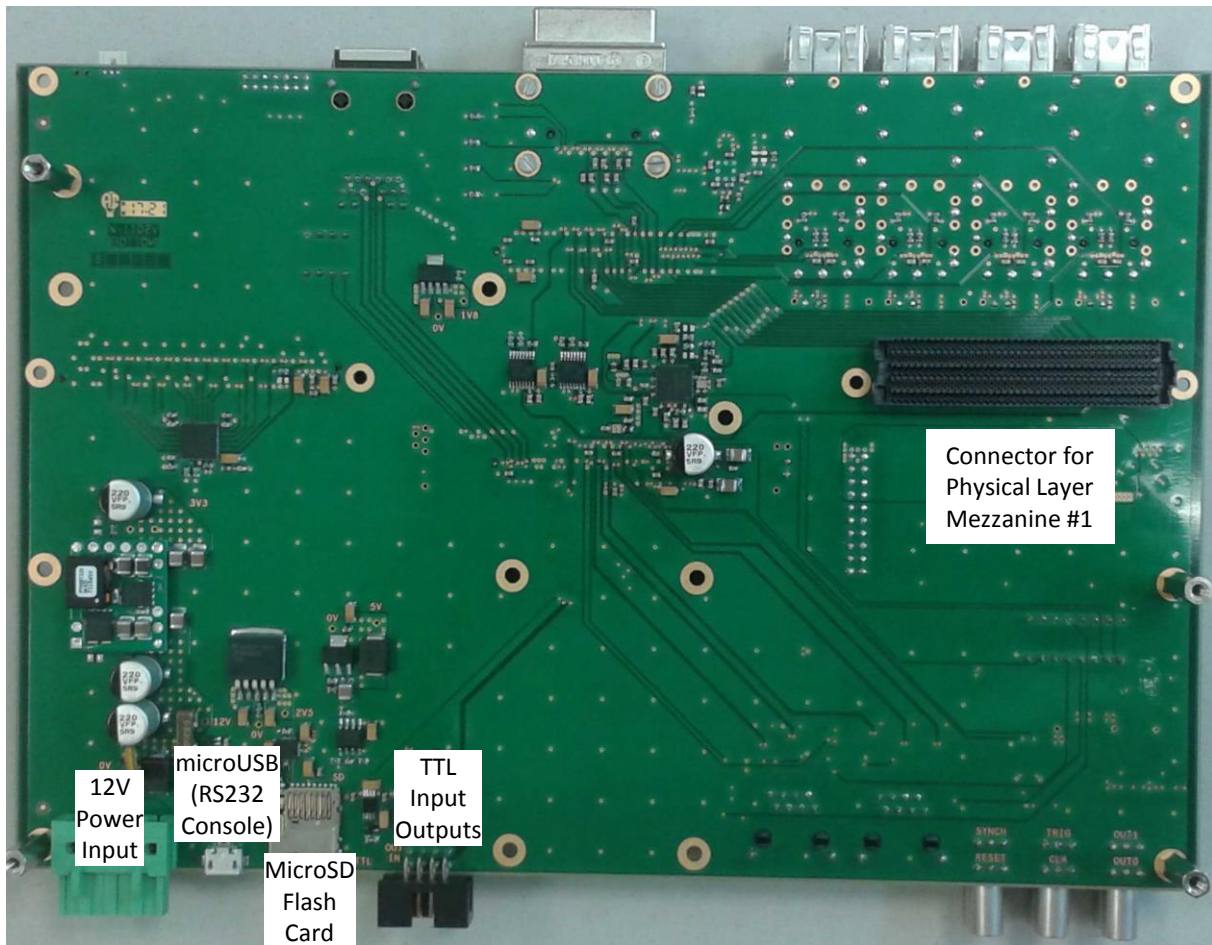A picture of the rear side of the TDCM is shown in Fig. 4.

Fig. 4.   Rear side view of the TDCM carrier.

2.4 MULTI-SFP MEZZANINE CARD

The multi-SFP Mezzanine Card is a custom made board that can house up to 16 small form factor pluggable optical transceivers (SFP) as specified in the Multi-Source Agreement (MSA) document signed by many manufacturers in 2000 and that has become a de-facto standard.

The host side of the multi-SFP Mezzanine Card uses LVDS signaling for the 16 transmit and 16 receive differential transceiver signals. The card also contains some peripheral I/O components controllable via I2C. These are used to control front-panel LED's (two per transceiver), drive the TX disable pin of each transceiver, and read-back the LOS indicator of each transceiver. In addition, it is also possible to access through an I2C multiplexer the Digital Diagnosis Monitoring interface of each transceiver.

The multi-SFP Mezzanine card uses a low pin count FMC connector as defined in VITA 57.1 standard. However, the card is **not compatible with this standard**, and its form factor is also different. Do not connect a multi-SFP Mezzanine Card to a carrier platform others than those recommended without checking in advance that it is compatible.

Fig. 5.   Front view of the multi-SFP Mezzanine Card.

A front view picture of the multi-SFP Mezzanine Card is shown in Fig. 5. Each of the 16 slots can accept a SFP transceiver. A port number is used to identify each SFP. If two multi-SFP Mezzanine Cards are connected to a carrier board, the port number of the second multi-SFP Mezzanine are offset by 16. During the procedure of detection and enumeration of the FEs connected to the TDCM, each FE is assigned the ID that corresponds to the port number where it is connected.

There are two front-panel LEDs associated with each transceiver. The green (or yellow) LED indicates the following:

- Permanently OFF: the corresponding port is not active
- Slow blinking: communication is not established with the partner FE
- Permanently ON: communication with the partner FE is established

The yellow (or green) LED illuminates during 100 ms whenever a data packet is received on the corresponding port. The functions of the LED may be changed in the future.

3 HARDWARE FEATURES AND INSTALLATION

3.1 POWER SUPPLY REQUIREMENTS

The TDCM requires a single +12 V power supply. Without any mezzanine card, the typical current drawn by the TDCM carrier and FPGA module is 1 A. Each mezzanine cards (without the optical transceivers) draws an additional 0.5 A. Each optical transceiver is expected to draw ~250 mA from 3.3V, i.e. the equivalent of ~100 mA from 12V. Assuming that 32 transceivers are used, these will draw ~3.2 A from 12 V. In total, a power supply rated 6 A should be adequate (to be confirmed).

The power dissipation of the FPGA module is rather high and it is mandatory that both a heat sink and a fan are used on the TDCM. The fan takes power (12 V) directly from the TDCM carrier board.

3.2 PRIMARY CLOCK SELECTION

The embedded processor of the TDCM and Ethernet communication rely on internal oscillators. Communication with the front-ends is synchronous to a 100 MHz reference clock that is derived from three possible sources selected by clock selection switches as shown in Table 3. When the NIM clock input or the local generator are selected, the 25 MHz supplied clock is multiplied by a PLL to produce the final 100 MHz reference clock.

Table 3   . Reference clock selection switches.

CLK < 1..0>	Reference clock source
OFF - OFF	Master TCM RJ45 connector (100 MHz)
OFF - ON	NIM Clock input (25 MHz)
ON - OFF	Local clock generator (25 MHz)
ON – ON	Illegal

When the primary clock is provided by the Master TCM, the DIP switch MTCM must also be ON.

3.3 BOOT MODE SELECTION

The TDCM is programmed via JTAG during the debugging and development phase. For self-configuration at power up, the board can boot from the micro-SD flash memory card, or the SPI card embedded on the FPGA module. Booting from the flash NAND memory should be possible but was not tested so far. The BOOT_MODE switches determine which boot device is used as indicated in Table 3.

Table 4   . Boot mode selection switches.

BOOT_MODE < 1..0>	Boot device
OFF - OFF	Micro SD card
OFF - ON	QSPI flash

ON - OFF	NAND flash
ON - ON	JTAG

3.4 MULTI-SFP MEZZANINE CARD

For some development and debugging tasks, it is possible to use the TDCM without any physical layer mezzanine card. However, communication with some front-ends require at least one of the physical layer mezzanine cards. Either the top or the bottom mezzanine card can be installed, or both of them. When only one mezzanine card is installed it is however preferable to connect it to the connector on the top side of the TDCM for reasons of compactness. However, on the first prototype TDCM, only the mezzanine on the bottom side can be plugged due to a connector orientation error for the layout of the PCB.

3.5 ETHERNET CONNECTION

The TDCM has several Ethernet ports. The RJ45 copper port on the front panel of the TDCM is connected to the Ethernet MAC embedded in the ZYNQ processor sub-system of the FPGA module. This device supports 10/10/1000 Mbps speeds but it does not support Jumbo frames. This Ethernet port is therefore disabled by default in the firmware of the TDCM. Instead of the basic ZYNQ MAC, a Xilinx TEMAC coupled to a GTP serial transceiver is used. The translation from the Ethernet 1000-BaseX interface to RJ45 copper requires a GBIC. The device used is AVAGO ABCU-5730ARZ, but other compatible devices should also work. Alternatively, an optical SFP transceiver can be installed if Ethernet over fiber is used. On the TDCM, SFP location #0 is reserved for the SFP/GBIC of the second Ethernet port. Note that the jumper next to SFP location #0 must be installed to enable that SFP port.

Most configurations will use a direct point-to-point connection between the Ethernet SFP/GBIC port and the PC used for DAQ and control, or a connection through a switch if the same PC is used to control multiple TDCMs. Note that the TDCM may not be directly connected to a large Ethernet network, but should reside in a small local private network.

3.6 CONNECTION OF A RS232 TERMINAL

The use of a RS232 terminal is optional except for the first time when a blank FPGA module is used on a TDCM. Several parameters (MAC and IP address, card ID, etc) must be set in the flash memory of the FPGA module before a TDCM becomes operational. Setting these parameters require a RS232 terminal connection. Settings are the following: speed = 115.200 bauds, 8 bits, no parity, no hardware flow control. Locally typed characters must be echoed and outgoing CR characters should be mapped to CR-LF. The TDCM uses a Microchip 2221 USB to RS232 converter. The corresponding drivers may need to be installed on the control PC. A micro-USB cable is required for the

connection. Note that on TDCM version 0, it is necessary that the card is powered ON in order for the PC to detect the new USB device properly. When the TDCM is turned OFF, the connection will need to be restored at the next power ON phase. On TDCM version 1 and beyond, the USB bridge of the TDCM is powered via the USB cable to the host. Hence the device is detected when the cable is plugged and the connection persists independently of the power ON/OFF state of the TDCM itself.

At power-up, messages are displayed on the RS232 console. During operation, fatal error messages may be printed in case of failure. An example of a normal startup is shown in Fig. 6.

Fig. 6.   Example of terminal printout after a normal boot.

3.7   RESET MEANS

In addition to the power-on reset, the TDCM has several other means for reset. The "PS_SRST" push button is a software reset for the embedded processor only. It does not reload or re-program the FPGA part of the TDCM. In practice, this button is seldom used. The "PS_POR" push button is equivalent to the power-on reset for the processor system: pressing this button reloads the embedded software, re-programs the FPGA and starts the execution of the main TDCM application program. A second push button with the same function is accessible on the front panel of the TDCM, and alternatively a small connector for a remote contactor is available. Note that these last two are not operational on the first prototype of the TDCM. The "RESET" push button is used to reset the user logic part of the FPGA part of the TDCM. Only experienced users may have to press this button.

3.8   LEDS

The FPGA module contains four yellow LEDs. These LEDs are not visible when the TDCM is placed inside a crate. These LEDs are used for development and debugging only.

The TDCM carrier has four LEDs, among these, three are visible from the front panel. A green LED (not visible from the front-panel) is connected to the DONE pin of the FPGA. It indicates that the FPGA part of the system on module has been programmed correctly. If the TDCM boots successfully, this LED will be ON. The green LED on the front panel illuminates when the 3.3V on-board converter is ON.

The yellow LED visible from the front panel is controlled by the FPGA logic. It normally indicates the presence of the primary clock and it should be blinking at ~1 Hz. If this LED stays still or blinks at a different rate, a missing or inappropriate primary clock may be the cause. Note that the function of this LED may be changed in the future.

The red LED visible from the front-panel is connected to an error signal of the application logic. Currently it is used to signal that the TDCM and one or several FEs have lost their common synchronization. When this error LED illuminates, the TDCM will not accept or forward any triggers to the front-ends until the error condition is cleared and some re-synchronization is performed. Note also that the signification of this LED indicator may be changed in the future.

4 SOFTWARE INSTALLATION

4.1 EMBEDDED FIRMWARE AND SOFTWARE

During exploitation, the TDCM will normally boot from the external micro-SD card or from the SPI memory on-board the FPGA module. A bootable file comprises a First Stage Boot Loader (FSBL) for the ZYNQ device on-board the FPGA module, the configuration bitstream for the FPGA, and the executable application program.
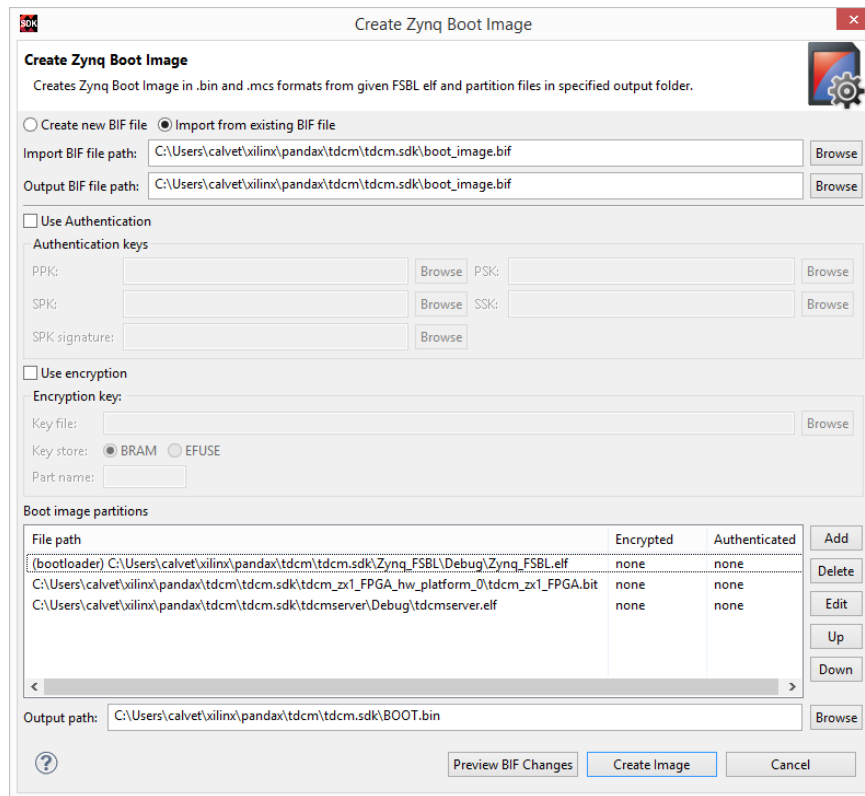
Fig. 7.   Window for creating a boot image file.

The bootable file is generated with Xilinx SDK tool (menu "Xilinx Tools" -> "Create Boot Image"). After loading the existing BIF file, the pop-up window shown on Fig. 7 should be displayed. Press the "Create Image" button to generate a new image file. The boot image file can then be copied to the Micro SD card. Note that the file name must be "BOOT.BIN" and cannot be changed. The file must also reside in the root directory of the card, although the card may contain other files and directories.

The "BOOT.BIN" file can also be copied to the SD Card remotely from the client PC program. This is accomplished in pclient (see section 11) by the commands below:

```
be sd mount
be sd wena 1
rcp <boot_version_data> BOOT.BIN
```

After mounting the SD card, the user must enable write access to the SD card. Then, he can perform a remote copy of the bitstream file to the SD Card with the corresponding command. To keep track of versions, the local copy normally has the target FPGA, board version and revision date contained in the name of the file. However, the target file must always be called "BOOT.BIN" exactly. The transfer takes ~1,5 minute (for a binary file of 15 Mbytes). Take extreme care when doing the copy because the original boot file of the TDCM will be erased. If a corrupted file is loaded, or if the upload fails, the TDCM will no longer boot. If such situation occurs, the micro SD memory card will need to be removed from the TDCM and some proper boot file be restored.

Alternatively the BOOT.BIN file can be programmed in the on board SPI device of the FPGA module via JTAG. This can be accomplished using Xilinx SDK via the menu "Xilinx Tools" -> "Program Flash". Note that if the SPI flash memory is entirely erased, the MAC address, the IP address and other settings of the TDCM will also be erased.

4.2 THE MINI-BIOS CONFIGURATION UTILITY

When a TDCM is boot for the first time, its IP address, MAC address and some other parameters must be set. This operation is accomplished via a RS232 terminal using the "mini-BIOS" utility. To enter mini-BIOS, hold the push button "BIOS" down and power ON the TDCM until the menu of mini-BIOS shown on Fig. 8 appears.

The options M, I, T and E are used to set the MAC address, IP address, Maximum Transfer Unit (MTU), and speed of the Ethernet connection to the DAQ and control PC. The TDCM must reside on a private network. The maximum transfer unit should be set to 1500 bytes at most if Jumbo frames are not supported, and can be set to a maximum of 8100 bytes. Note that the native Ethernet controller of the ZYNQ does not support Jumbo frames but the MAC that drives the SFP Ethernet port does. It is recommended to set the speed to 1000 Mbps, although 100 Mbps and even 10 Mbps should work.

The card ID should be set in a way that each TDCM in the system has a different ID. The allowed range is from 0 to 31.

The PLL type and output clock delay should be left to 0.
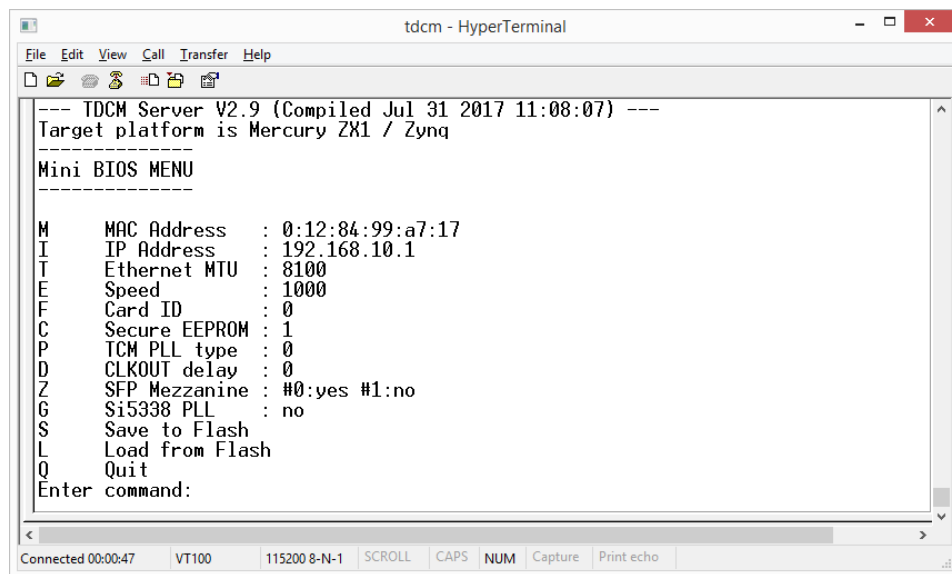
The SFP mezzanine parameter should be set to 0, 1, 2 or 3 depending on whether no, one, or the two SFP mezzanine cards are installed. Note that on the TDCM prototype, mezzanine #0 is the one that plugs on the bottom side while on subsequent models, mezzanine #0 refers to the one on the top side.

Different types of secure EEPROM may be installed on the FPGA module. Refer to the documentation provided by Enclustra for details. The TDCM does not use the MAC address and other information stored in the secure EEPROM.

The Si5338 PLL is only present on the Enclustra PE1 evaluation kit. This option must not be enabled on the TDCM.

After all the desired changes have been made, they must be saved in the on-board flash memory with the command "S". To exit mini-BIOS and boot the TDCM, press "Q".

An example of a successful boot of the TDCM is shown in Fig. 9. After displaying the version number and compilation date, various peripheral circuits and devices are configured. Finally, the Ethernet port is configured. At this time, the TDCM shall respond to ping commands and to the commands sent by the DAQ PC following the protocol and syntax defined in the relevant sections of this document.

Fig. 8.   Mini-BIOS menu.

Fig. 9.   Console messages after a normal boot.

4.3 EMBEDDED SOFTWARE COMPILATION

The embedded software of the TDCM has originally been developed using Xilinx SDK release 2015.4. Migration to more recent versions of the tool should work, but this was not tested so far.

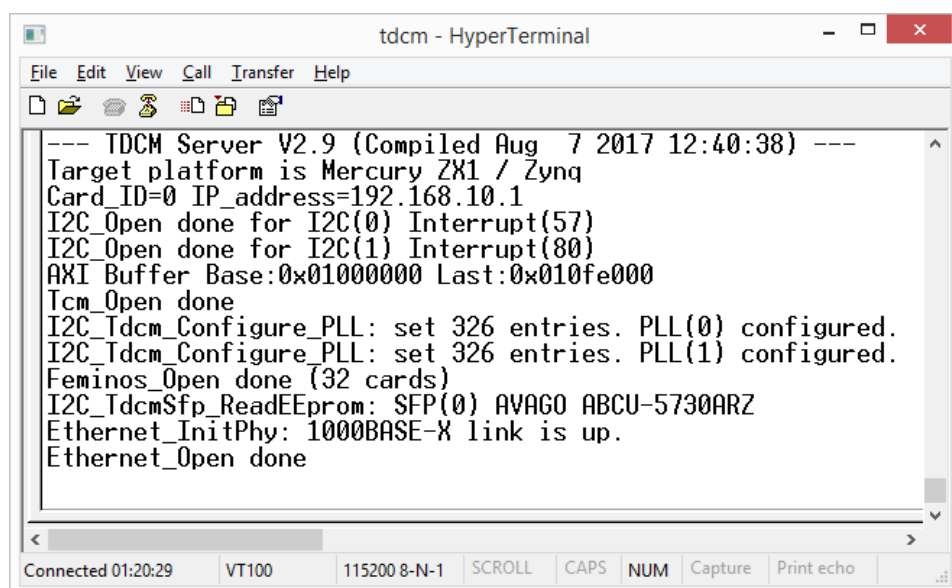Selecting which Ethernet MAC is used (hard IP in the ZYNQ of the FPGA module or Xilinx TEMAC synthetized in FPGA logic) is defined at the compilation time and cannot be changed at run-time. Normally, the TDCM software is compiled in the Xilinx TEMAC flavor because it supports Jumbo frames while the hard IP of the ZYNQ does not. However, both options are supported. The firmware instantiates both types of controller and need not be changed to select which controller is used. However, producing the desired flavor of the software requires a few changes in the TDCM SDK project itself. Those changes are listed in Table 5.

Table 5 . Software generation settings for Ethernet MAC selection.

Item	ZYNQ hard IP	TEMAC soft IP
Include search path	/network/zynq	/network/temac
Pre-processor definitions	USE_EMAC_PS ETHERNET_PHY_KSZ9031	USE_AXI_ETHERNET_DMA
Source files	ethernet_xemac.c ksz9031.c	ethernet_axidma.c 1000baseX.c

Note that on TDCM version 0, the micro SD memory card interface can operate at 6.25 MHz maximum instead of the default value of 25 MHz which is set in Xilinx support library. The speed must be reduced from 25 MHz to 6.25 MHz and the speed of the bus must not be changed to high speed. The modifications to perform in the board support package file xsdps.c are the following:

```
InstancePtr->BusSpeed = 6250000; //SD_CLK_25_MHZ;
…
//Status = XSdPs_Change_BusSpeed(InstancePtr);
```

The modification persists until the BSP source files are re-generated. The SD memory card interface for TDCM version 1, and beyond, can operate at the nominal 25 MHz rate. No modification to the BSP provided by Xilinx is needed.

5 MASTER TRIGGER/CLOCK MODULE INTERFACE (PANDAX-III SPECIFIC INTERFACE)

5.1 PHYSICAL LAYER

When the TDCM is used as a slave device of a master device, e.g. in PandaX-III a Mater Trigger Clock Module, M-TCM, it is called a S-TDCM. The interface between the S-TDCM and the M-TCM uses a four-pair standard category 6 shielded twisted pair (STP) RJ45 cable. This type of cable is commonly used for Fast/Gigabit Ethernet networking. One pair is split into two wires that carry unipolar 3.3V LVTTL signals while the three other pairs use LVDS. The signals from the M-TCM to the S-TDCM are:

- TCM_CLK: this signal is a free running 100 MHz reference clock (LVDS).
- TCM_MOSI (Master Out, Slave In): this signal transports the serially encoded trigger and other synchronous control signals (LVDS).
- TCM_ENAREM (ENAble REMote): this is an active high 3.3V LVTTL signal to indicate to the S-TDCM that the M-TCM is present and requests the S-TDCM to respond.

The signals from the S-TDCM to the M-TCM are:

- TCM_REMDET (REMote DETected): this is an active high 3.3V LVTTL signal asserted by the S-TDCM to signal its presence and readiness to the M-TCM.
- TCM_MISO (Master In, Slave Out): this signal carries serially encoded synchronous data (e.g. trigger acknowledge, busy/release, etc.) from the S-TDCM to the M-TCM (LVDS).

A schematic view of the M-TCM to S-TDCM interface is shown in Fig. 10.
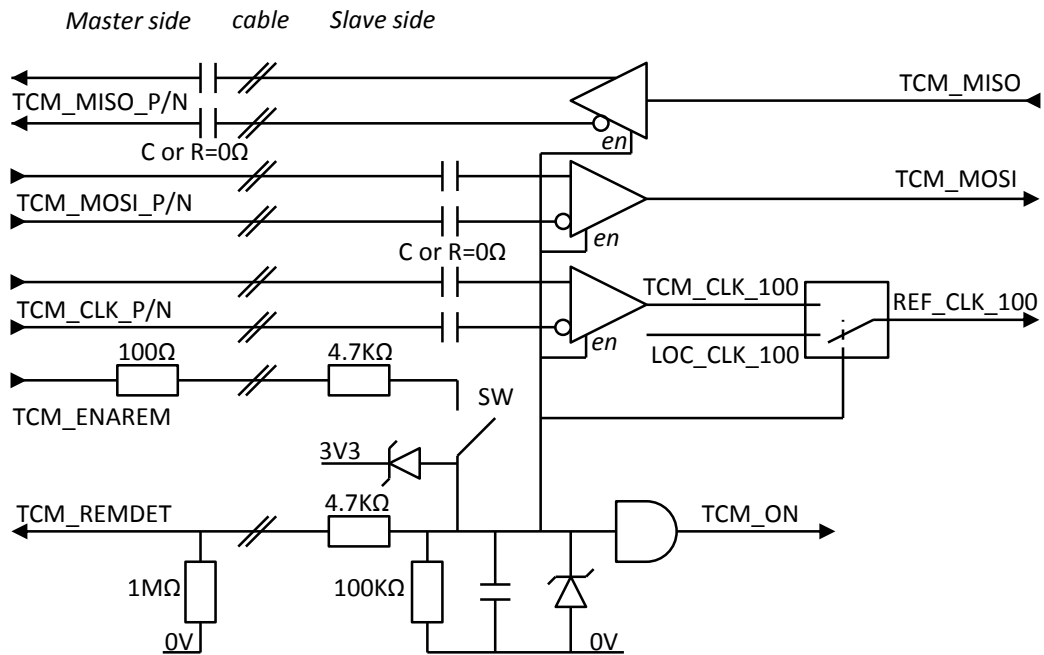
Fig. 10.   M-TCM – S-TDCM electrical interface.

The M-TCM de-activates communication with the S-TDCM when it detects a low level on the TCM_REMDET line. This happens in the following situations:

- The M-TCM sets a low level on TCM_ENAREM to voluntary disable communication with the S-TDCM,
- The M-TCM has set a high level on TCM_ENAREM but no S-TDCM is connected at the other extremity of the cable, or an S-TDCM is connected but it is currently not powered, or it is present and powered but SW is open meaning that it is desired that the S-TDCM operates on its own local clock.

When SW is closed, the S-TDCM takes its locally generated clock as a reference if the M-TCM is not present or if TCM_ENAREM is low, which happens if the M-TCM is present but not powered, or present and powered but TCM_ENAREM is not activated. The S-TDCM takes TCM_CLK_100 as a reference when both SW is closed and the M-TCM has asserted TCM_ENAREM high. The signal TCM_ON indicates to the FPGA logic of the S-DTCM that the M-TCM is present and ready.

Differential pairs may use AC coupling or DC coupling by choosing between capacitors or 0Ω resistors at the receiving end. The TCM_CLK_P/N pair will preferably use AC coupling. Initially, it is proposed to use DC coupling for the TCM_MOSI_P/N and TCM_MISO_P/N pairs although this may choice may be changed in the future. All differential receivers must be terminated with a 100Ω differential impedance. This resistor may be integrated in the receiver or it may be external. A failsafe circuit may also be needed.

If DC coupling is used for differential pairs, the M-TCM is not allowed to actively drive any of its LVDS output lines unless a high level is detected on TCM_REMDET. Similarly, the S-TDCM is not allowed to actively drive the TCM_MISO_P/N lines unless a high level is sensed on TCM_REMDET. If AC coupling is used, the common mode voltage of the differential drivers will be blocked at the receiver end, and differential drivers can be enabled even when there are connected to a remote partner board that is not powered. It is however preferable that differential drivers transmit a constant logic level when no active receiver is detected. Because series resistors are placed on TCM_ENAREM, limited current will flow from the M-TCM to the S-TDCM if the M-TDCM drives this line to a high level while the S-TDCM is present but not powered and SW is closed.

The pin assignment of the RJ45 cable between the M-TCM and the S-TDCM is shown in Fig. 11. Note that colors depend on compliance with one of two possible standards.

			EIA/TIA T568B		EIA/TIA T568A
1	TCM_CLK_P		Wh/ Or		Wh / Gr
2	TCM_CLK_N		Orange		Green
3	TCM_ENAREM		Wh / Gr		Wh / Or
4	TCM_MOSI_P		Blue		Blue
5	TCM_MOSI_N		Wh / Bl		Wh / Bl
6	TCM_REMDET		Green		Orange
7	TCM_MISO_P		Wh / Br		Wh / Br
8	TCM_MISO_N		Brown		Brown

Male RJ45 connector
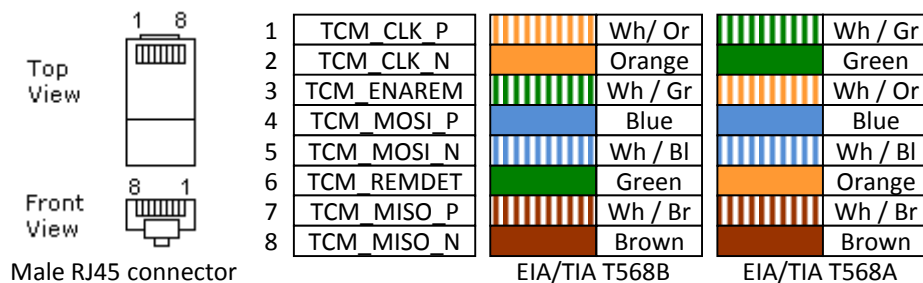
Fig. 11.   M-TCM – S-TDCM cable pin assignment.

5.2 MESSAGE FORMAT

The format of the frames sent from the M-TCM to the S-TDCM before encoding is shown in Fig. 12.
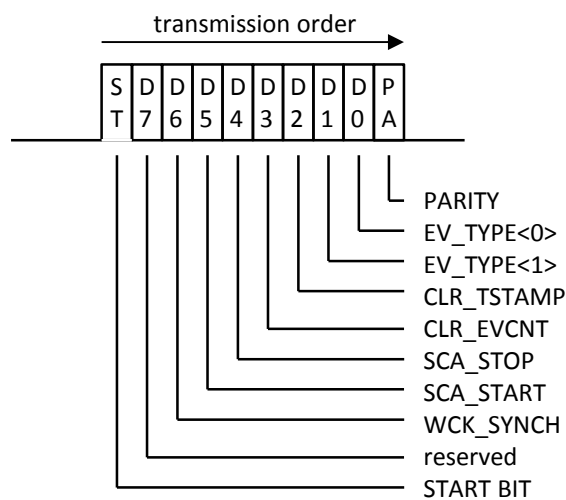
Fig. 12.   M-TCM to S-TDCM frame format.

The marking level of the TCM_MOSI line is a low level. The START_BIT is a high level. The PARITY bit is the exclusive OR of the START_BIT and bit D0 to D7, i.e. the total number of bits equal to 1 in a message taken from the start bit to the parity bit (included) is always made even. The bit WCK_SYNCH is used to synchronize the clock generator of the SCA write clock in each front-end card. The SCA_START bit is used to synchronously start sampling in the SCA of the ASICs of the front-ends. The SCA_STOP bit is the trigger. Four types of event can be distinguished by EV_TYPE<1..0>. The CLR_EVCNT bit and CLR_TSTAMP bit are used to synchronously clear the event counter and time stamp counter of all slave cards simultaneously. The CLR_TSTAMP bit actually performs a preset of the timestamp counter to an initial programmable value. After the last bit of data, the M-TCM must insert a gap of at least 6 symbols before transmitting the next frame. Consequently, the maximum frame rate sent by the M-TCM to the S-DCM is 1/160 ns = 6.25 MHz. However, the absolute time of synchronization signals can be resolved with 10 ns resolution.

The format of the frames sent from the S-TDCM to the M-TCM before encoding is shown in Fig. 13.
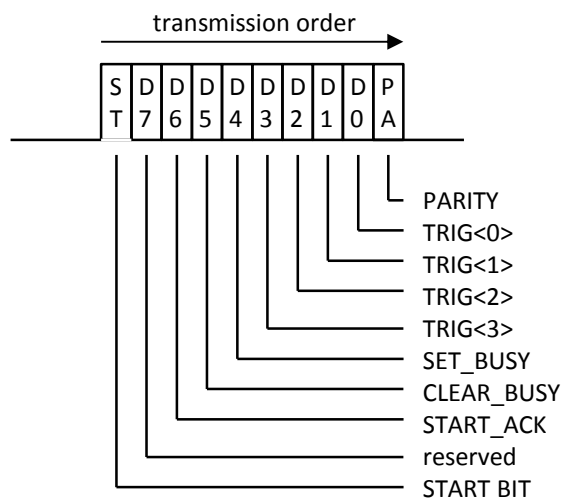
Fig. 13.   S-TDCM to M-TCM frame format.

The marking level of the TCM_MISO line is a low level. The START_BIT is a high level. The PARITY bit is the exclusive OR of the START_BIT and bit D0 to D7, i.e. the total number of bits equal to 1 in a message taken from the start bit to the parity bit (included) is always made even. After the last bit of data, the S-TDCM inserts a gap of 6 bits before the next frame. Consequently, the maximum frame rate sent by the S-TDCM to the M-TCM is 1/160 ns = 6.25 MHz.

The bit START_ACK indicates that the S-TDCM has successfully received the SCA_START command from the M-TCM and that the ASICs on the front-end side are now

sampling data. The SET_BUSY bit is used to acknowledge the reception of the current trigger and indicates that the S-TDCM cannot perform a SCA_START command until it sends CLEAR_BUSY. The bit CLR_BUSY is used to indicate that the FEs have completed processing of the previous event and are now ready to resume the SCA write operation.

The TRIG<3..0> field is used to transfer self-trigger primitives from the S-TDCM to the M-TCM. This is not defined at present.

5.3 LINE ENCODING

The TCM_CLK_P/N pair carries a free running 100 MHz clock. The TCM_MOSI_P/N and TCM_MISO_P/N lines carry 10-bit frames (1 start bit + 8 bit data + 1 parity bit) at 100 Mbps. The transitions on TCM_MOSI_P/N are phase aligned by the M-TCM with the transitions (rising edge or falling edge) of the primary clock TCM_CLK_P/N. The messages sent by the S-TDCM on TCM_PISO_P/N are synchronous to the clock supplied by the M-TDCM, but the phase on the M-TCM side is unknown. It depends on cable length and the various delays of the active components on the path. To recover the messages sent by the S-TDCM, the M-TCM must adjust locally the phase of the received signal with respect to its own local clock.

Operation without any line encoding is only possible using DC-coupling on the corresponding transport lines.

Optionally, a DC balanced protocol is also supported. This protocol can be used with either AC-coupled or DC-coupled lines. (Note: a trivial, almost DC-balanced encoding can be obtained by making a XOR between the non-encoded serial stream with a locally generated bit pattern of alternating 1 and 0's. A more sophisticated encoding scheme will be defined later if needed). The DC balanced protocol can be enabled independently for the S-TDCM to M-TCM link and for the M-TCM to S-TDCM link. Obviously, the settings must be consistent between the M-TCM and the S-TDCM for correct operation.

5.4 BIT ERROR RATE TESTER

A bit error rate tester for checking the quality of the communication link between the M-TCM and S-TDCM is implemented. The bit error rate tester supports four different standard patterns: PRBS7, PRBS15, PRBS23 and PRBS31. Before any PRBS can be received or transmitted, the S-TDCM must be set first to operate in the bit error test mode by programming the MTCM_BERT_ENA bit to one. Once this is done, the bit error test mode can also be engaged on the M-TCM side.

When the bit MTCM_BERT_RXEN is set to 1, the S-TDCM tries to look on the received PRBS. As soon as lock is achieved, it compares the received bit stream with the expected pattern, which is generated locally, and counts the eventual errors. Note that the repetition period of the PRBS31 pattern at 100 Mbps is ~21 s. It may take up to this time to gain synchronization.

When the bit MTCM_BERT_TXEN is set to 1, the S-TDCM starts sending the selected PRBS to the M-TCM. On the rising edge of bit MTCM_BERT_DOERR, a single bit error is inserted.

The bit error rate tester can be run in either direction independently or both directions at the same time, for as long as it is desired. In the current implementation, the bit error counter of the receiver saturates at 255, and the received and transmitted bit counters roll over at 2^{24} Mbit, which corresponds to approximately 2 days at 100 Mbps. At the end of a test, the receiver side should be disabled first to avoid that stopping the PRBS on the transmitter side is misinterpreted as bit errors by the receiver side. After the PRBS transmitter and receiver sides of the S-TDCM have been disabled, the bit MTCM_BERT_ENA should be cleared. This restores the normal operation of the link between the S-TDCM and the M-TCM.

6 INTERFACE TO A FRONT-END UNIT

6.1 PHYSICAL LAYER

The interface between the TDCM (standalone of slave) and each FE normally uses an optical fiber media. Small Form factor Pluggable (SFP) transceivers in compliance with the Multi-Source Agreement (MSA) are a recommended choice. Alternatively, or for debugging purposes, a copper cable interface is also supported.

6.1.1 OPTICAL TRANSCEIVER AND OPTICAL FIBER INTERFACE

The physical layer mezzanine card supports standard SFP transceivers, either single fiber bi-directional transceivers or common dual fiber transceivers. If bi-directional transceivers are used, it is recommended to use 1310 nm TX / 1490 nm RX transceivers on the TDCM side. These transceivers can operate with single mode or multimode fiber. Alternatively, dual-fiber transceivers at 850 nm wavelength are also adequate. These devices require multimode fiber. Other types of optical media (e.g. LED transceivers for plastic optical fiber) will be supported if this is needed.

6.1.2 COPPER INTERFACE USING A RJ45 CABLE

The copper interface between the TDCM and a FE is physically almost identical to the interface between the M-TCM and the S-TDCM. The four pairs of an RJ45 cable are assigned the same functions. Cable usage is defined in Fig. 14.

Pin	Signal	T568B color	T568A color
1	TDCM_CLK_P	Wh/ Or	Wh / Gr
2	TDCM_CLK_N	Orange	Green
3	TDCM_ENAREM	Wh / Gr	Wh / Or
4	TDCM_MOSI_P	Blue	Blue
5	TDCM_MOSI_N	Wh / Bl	Wh / Bl
6	TDCM_REMDET	Green	Orange
7	TDCM_MISO_P	Wh / Br	Wh / Br
8	TDCM_MISO_N	Brown	Brown

Male RJ45 connector   EIA/TIA T568B   EIA/TIA T568A
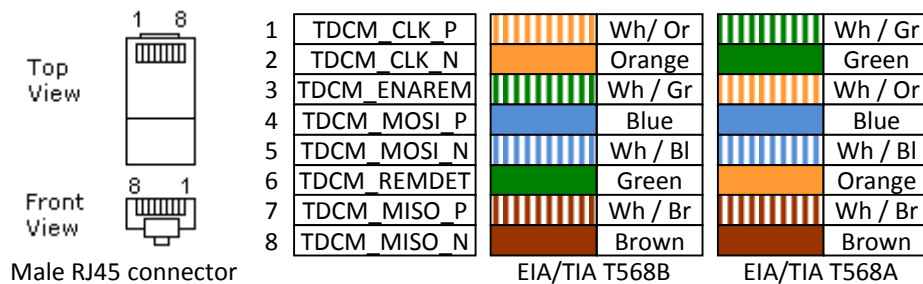
Fig. 14.   TDCM – FE cable pin assignment.

The same electrical schematic as the one shown in Fig. 10 can be used. However, it is recommended to use AC-coupling by default for all differential pairs. In this case, LVDS transceivers need not the tristate capability. The TCM_CLK_P/N carries the primary 100 MHz reference clock and the TCM_MOSI-P/N serial stream is synchronous to that clock. Similarly, the serial stream sent by the FE to the TDCM on the TCM_MISO_P/N pair must be synchronous to the same 100 MHz reference clock. Message format and line encoding on TCM_MISO and TCM_MOSI for the S-TDCM – FE link are different from that used for communication between the M-TCM and S-TDCM.

6.2 TDCM – FE COMMUNICATION PRINCIPLES

Communication between the TDCM and the FEs use an asymmetric network. In the TDCM to FE direction, a low-skew, multi-cast network is used. The same bit stream is sent by the TDCM to all FEs simultaneously. In the FE to TDCM direction, each FE has its own private point-to-point communication link with the TDCM.

Communication between the TDCM and the FEs requires transporting different types of messages: isochronous information (trigger, time-stamp clear order), runtime configuration parameters, read-back parameters, periodic monitoring requests, responses with the value of the monitored variable, request for data, response messages containing event data fragments, etc. All these different types of messages are classified in three different categories. Using three virtual channels implemented over the physical links using time division multiplexing, messages from the three different classes are exchanged concurrently in both directions between the TDCM and the FEs. The following virtual channels are defined:

- Virtual Channel A is used to transport isochronous messages (mostly trigger information) from the TDCM to all FEs. In the opposite direction, it transports trigger primitives, trigger acknowledge and other synchronous information.
- Virtual Channel B is used for runtime parameter configuration and read-back as well as slow control monitoring of the variables measured by each FE (e.g. voltage, current, temperature).
- Virtual Channel C is used for event data collection from the FEs under the control of the TDCM.

Each virtual channel is assigned a fraction of the total available link bandwidth. In the TDCM to FE direction, 1:2 of the link bandwidth is allocated to virtual channel A, and 1:4 of the link bandwidth to virtual channel B and C. In the FE to TDCM direction, 1:2 of the available link bandwidth is reserved for virtual channel C, while virtual channel A and B are assigned 1:4 of the link bandwidth each. On the physical TDCM to FE link, each group of four consecutive transmitted bits is composed of 1 bit from virtual channel A, followed by one bit from virtual channel B, then one bit from virtual channel A again, and then one bit from virtual channel C, i.e. the sequence is A, B, A, C, A, B, A, C, etc. For the FE to TDCM links, the sequence of transmitted bits is the following: A, B, C, C, A, B, C, C, etc.
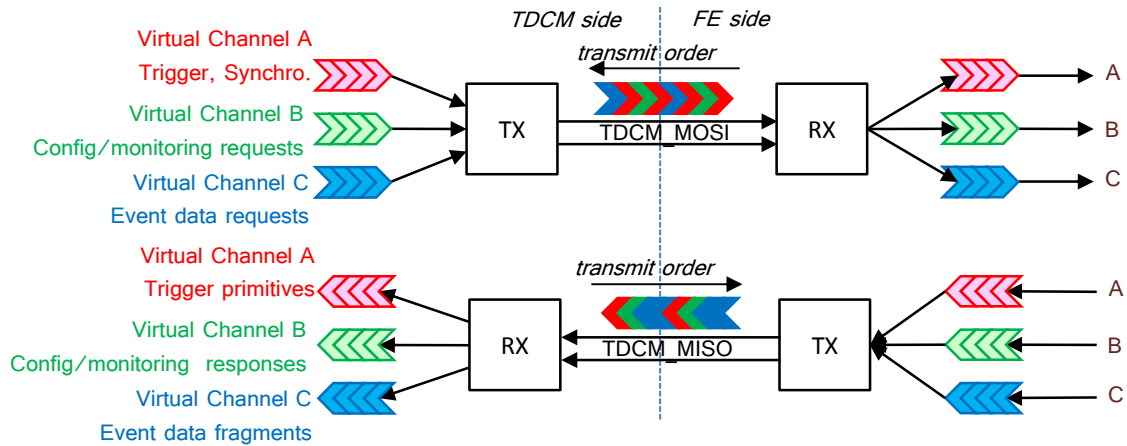
Fig. 15.   TDCM – FE communication over time-multiplexed virtual channels.

Proper synchronization is required to ensure that the original bit streams on each virtual channel in both directions are reconstructed properly. An illustration of the principle of virtual channel multiplexing is shown in Fig. 15.

Table 6 . Virtual Channel link bandwidth allocation.

Direction	Virtual Channel	Fraction allocated	Actual net bandwidth
TDCM to FE	A	1:2	50 Mbps
	B	1:4	25 Mbps
	C	1:4	25 Mbps
FE to TDCM	A	1:4	100 Mbps
	B	1:4	100 Mbps
	C	1:2	200 Mbps

The bandwidth allocated to each virtual channel is fixed and cannot be changed at run time. The allocation of link bandwidth for the three virtual channels is shown in Table 6. It is assumed that link speed is 100 Mbps (before encoding) in the TDCM to FE direction and that the return link runs at 400 Mbps (also before encoding).

6.3 MESSAGE FORMAT – VIRTUAL CHANNEL A

The format of messages exchanged over Virtual Channel A is almost similar to those exchanged between the M-TCM and the S-TDCM. The format of the frames sent from the TDCM to the FEs before line encoding is shown in Fig. 16.
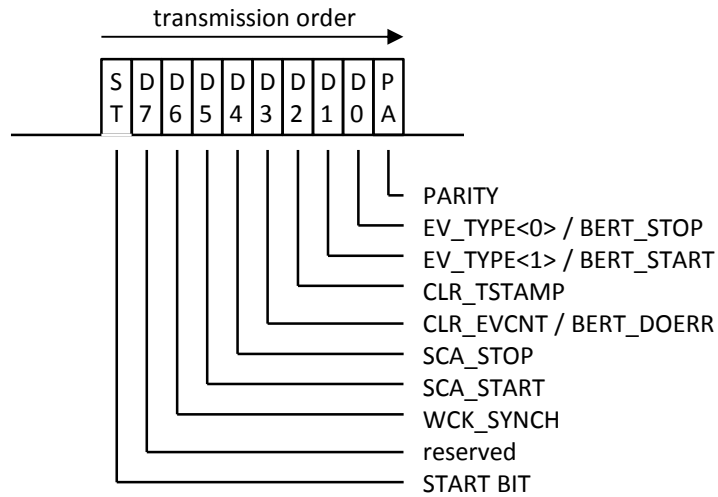
Fig. 16.   TDCM to FEs Virtual Channel A frame format.

The signification of the different bits is identical to that of the M-TCM to TDCM link. Refer to the corresponding section for details. Note that some bits have a dual function and are used by the bit error rate tester of the FE to TDCM link as detailed in section 6.10.

The format of the messages sent by the FEs to the TDCM on Virtual Channel A is shown in Fig. 17.
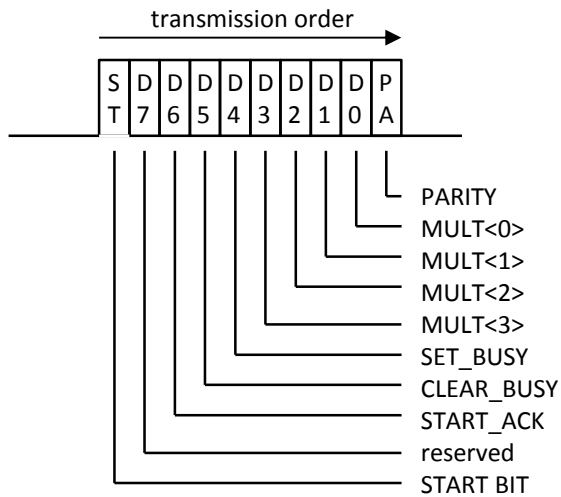
Fig. 17.   FE to TDCM Virtual Channel A frame format.

Each of the MULT bits corresponds to a self-trigger primitive elaborated by the FE by applying a programmable threshold to the digitized hit multiplicity signal of an AGET chip. The TDCM will combine this information among all FEs to elaborate a self-trigger signals. The details of the algorithm is not defined yet.

6.4 MESSAGE FORMAT – VIRTUAL CHANNEL B

Logically, the TDCM identifies each FE by a 5-bit ID that corresponds to the ID of the physical port where the fiber or cable of that FE is attached. Port are numbered from 0 and up to 31 on the TDCM. A message sent on Virtual Channel B will normally be interpreted only by the FE that has the ID that matches that indicated in the message. When the broadcast bit (BC) is set in the message, all FEs shall receive and decode the message. The TDCM sees the parameter configuration and variable monitoring registers as a flat 16-bit address, 32-bit data read/write virtual memory space. All configuration setting or retrieving operations and monitoring operations are performed by one or series of read/write operations to the virtual memory space of the FE accessible by the TDCM via Virtual Channel B.

The format of messages sent from the TDCM to the FEs on Virtual Channel B is shown in Fig. 18. The message is composed of one start bit (ST), 62 payload bits, and one parity bit (PA). The parity bit is computed so that the number of bit equal to 1, including the start bit, is always even. A minimum gap of 4 bits equal to 0 must be present between two successive frames on this virtual channel.
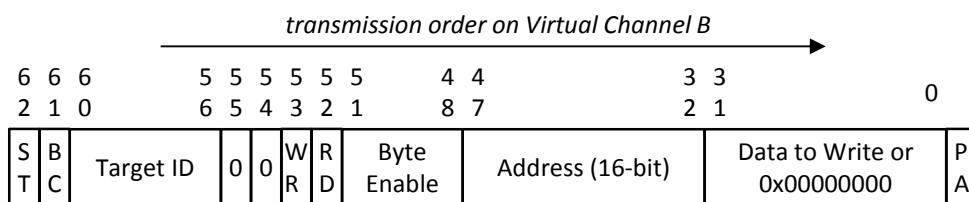
Fig. 18.   TDCM to FE Virtual Channel B frame format.

The address supplied is a byte address but only transactions aligned on 32-bit boundaries are supported. Access to individual bytes is possible using the four individual Byte Enable. The address covers a range from 0x0000 to 0xFFFC. The operation to perform, read or write, is indicated by setting the appropriate RD or WR bit. The FE shall respond to messages where both RD and WR are set to 0, although this operation does not perform any action. A transaction where both RD and WR are set to 1 is illegal. The FE shall return an error in this case. The data field is set with the data to write in the case of a write transaction. In case of a read, the 32 bits are still present, but they are set to a value (normally 0x00000000) that should be ignored by the FE.

The FE is not allowed to send a frame on Virtual Channel B until it has received a request from the TDCM. The FE must respond to each message received on Virtual Channel B with a matching ID or the broadcast bit set by sending one and only one frame on the same virtual channel. The format of a frame sent by a FE to the TDCM on Virtual Channel B is shown in Fig. 19.

transmission order on Virtual Channel B

6 2	6 1	6 0			5 6	5 5	5 4	5 3	5 2	5 1		4 8	4 7		3 2	3 1	0
S T	B E		Target ID		P E	F E	W R	R D		Byte Enable		Address (16-bit)			Data written or read		P A
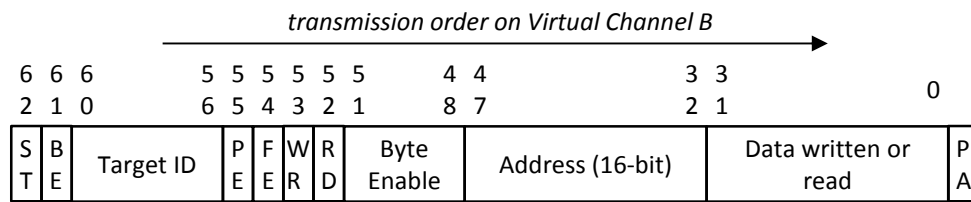
Fig. 19.   FE to TDCM Virtual Channel B frame format.

The frame is composed of a start bit (ST) followed by 62 payload bits and one parity bit (PA). The BE bit (Bus Error) is set by the FE to indicate that the requested operation was not successful: no device present at the requested address, misaligned access, etc. The PE bit (Parity Error) is used to indicate that the FE has received the request frame from the TDCM, but the parity was incorrect. The FE bit (Format Error) indicates that the request did not have the expected number of gap bits. If the requested operation was successful, bits BE, PE and FE will all be 0. The RD, WR, Byte Enable and Address fields must be identical to that of the request. In case of a write, the Data field shall also echo the data supplied in the request. In case of a read, the data field contains the content retrieved from the local resource of the FE being addressed. The data byte corresponding to the different Byte Enable may be set to 0x00 if the corresponding byte enable is set to 0. It is also permitted to put valid data even for those bytes where Byte Enable was inactive. However, the TDCM will ignore the bytes corresponding to disabled bits in the Byte Enable field. If a parity error is detected by the TDCM on the received frame, the response from the FE shall be ignored. If the TDCM does not receive any response from a FE after a certain amount of time, a timeout error will be issued. All errors are propagated to the upper level that may initiate a retry or inform the user to perform diagnosis and recovery actions.

The resource map of each front-end is intended to be uniform across the different hardware implementations. The core of the resource map of the FE is described in Section 8 of this document. There may be specific variations for the different flavors of front-end hardware. Access to specific resources should be assigned the address locations that are currently not affected.

6.5 MESSAGE FORMAT – VIRTUAL CHANNEL C

This virtual channel is dedicated to the transfer of event data from the FEs to the TDCM. The format of the frames sent by the TDCM to the FEs on Virtual Channel C is shown in Fig. 20.

```
              transmission order on Virtual Channel C
              ─────────────────────────────────────────▶

              3  3          3  3
              6  5          2  1                      0
              ┌──┬──────────┬────────────────────┬──┐
              │S │ Operation │    Target FEs      │P │
              │T │  OP-Code  │  (one bit per FE)  │A │
              └──┴──────────┴────────────────────┴──┘
```
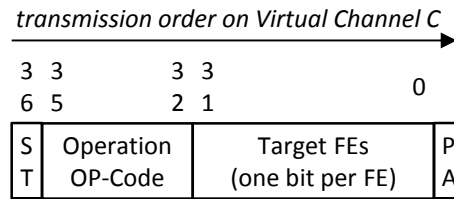
Fig. 20.   TDCM to FE Virtual Channel C frame format.

Each message is composed of a start bit, a 4-bit operation code, 32 bits indicating to which FE the operation applies, and a parity bit. When receiving messages on Virtual Channel C, each FE checks if the bit of the Target FE field that corresponds to its own ID is set or not, and executes the operation specified in the body of the message accordingly. The different operations to perform are listed in Table 7.

Table 7 . Commands sent by the TDCM to the FEs on Virtual Channel C.

Op-Code	Value	Action
NOP	0x0	No action. For test purpose only.
INC_CREDIT	0x1	Increments by 1 unit the number of packets the FE is allowed to send.
CLR_CREDITS	0x2	Clear the number of credits to send packets.
-	0x3 to 0xF	Reserved for future use

The protocol over Virtual Channel C implements the appropriate flow control to avoid that the TDCM is overflowed by event data from the FEs. All data transfers are initiated by the TDCM. The FEs are not allowed to send any data on this virtual channel until they have obtained the required credits from the TDCM. Initially, the FEs do not have any credit to send data packets to the TDCM. To begin data collection, the TDCM sends an INC_CREDIT operation to the FEs. Each of the FE concerned shall increment by one unit the counter of credits that corresponds to the number of packets it is allowed to send. If event data are ready, a FE can send over Virtual Channel C the next packet of event data, and it decrements by one unit its counter of credits. If no data is available, the FE must sent an empty data packet to the TDCM after a pre-defined timeout selectable among 1 ms, 10 ms, 100 ms or 1 s. Failure to do is considered as an error by the TDCM. Note that the FE must decrement by one unit its credit counter even when sending an empty data packet. When a FE has no credit left, it is not allowed to send any data packet, empty or not, to the TDCM, even after the programmed timeout. The FE must wait for credits to be received from the TDCM to resume data transmission on Virtual Channel C. The TDCM may transmit several INC_CREDIT messages before it receives the first data packet from a FE. Pipelining multiple data requests and data packets may increase global throughput, but initially, the TDCM will not pipe-line multiple requests to the same FE and the credit count at both ends should always be 0 or 1. In case of

error, or to initialize the system, the TDCM can send a CLR_CREDITS instruction. All the FE concerned must clear their credit counter and halt transmission on Virtual Channel C after the packet currently under transmission (if any) has been sent.

Other operations may be defined in the future.

The structure of the data packets – when these are non-empty – sent by a FE to the TDCM are shown in Fig. 21. A non-empty packet is composed of an even number of 16-bit words. The header is composed of a START_OF_PACKET constant (currently 0xAC0F) followed by a 16-bit word. The SOE flag (Start Of Event) is set to 1 to indicate that the current packet is the first packet of data for this event. The EOE flag (End Of Event) is set to indicate that the current packet is the last one for this event. The Packet Size field indicates the total size, in bytes, of the payload words. The number of payload words can be 0, must be a multiple of 2, and is limited to the capacity of the receive FIFOs on the TDCM side, which is currently 2 KBytes. If the number of payload words to send is odd, a null padding word must be added, but the Packet Size field must still indicate the total number of payload words, including the null padding word.
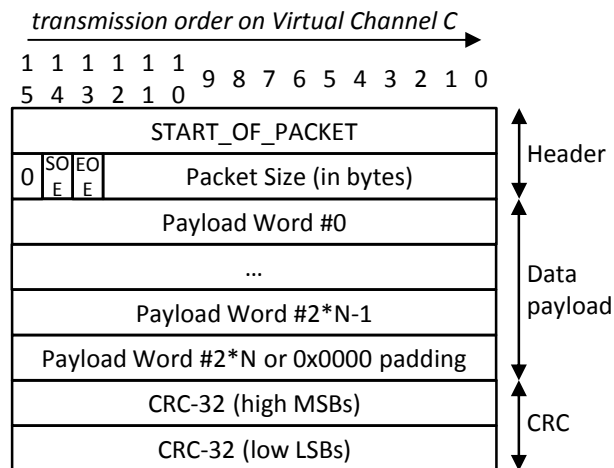
Fig. 21.   FE to TDCM Virtual Channel C packet format.

After the last payload word, a cyclic redundancy check word is appended. The standard CRC-32 polynomial commonly used by many standard applications (ISO 3309, ANSI X3.66, FIPS PUB 71, FED-STD-1003, ITU-T V.42, Ethernet, SATA, MPEG-2, Gzip, PKZIP, POSIX cksum, PNG, ZMODEM…) is adopted:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0$$

The CRC-32 is computed over all the header and data payload words. Optionally, CRC-32 computation can be disabled on the FE and two null words should be instead in this case. Verification of the CRC-32 is also optional on the TDCM side, although it is strongly recommended to enable it.

When a FE has a positive number of credits but no data to send, it must send an empty data packet after a programmable timeout has elapsed. An empty data packet is simply composed of a START_OF_PACKET header followed by a null 16-bit word indicating that the size of the payload of this packet is null. This two word packet is followed by its CRC-32, or 32 bits equal to zero if CRC-32 computation is disabled.

6.6 LINK CLOCKING AND GLOBAL CLOCKING

The diagram of global system clock distribution is shown in Fig. 22 for the case of PandaX-III. The main reference clock of the system, *M_REF_CLK*, is supplied by a master trigger and clock module (M-TCM) to all S-TDCM over point-to-point copper links (RJ-45 cables). This clock is transmitted without any encoding at the appropriate desired frequency. By default, the TDCM assumes **the reference clock is 100 MHz**. In standalone mode, the TDCM uses for *M_REF_CLK* a local clock derived from an on-board oscillator. The TDCM use *M_REF_CLK*, or a multiple of *M_REF_CLK* obtained by multiplication with a PLL, to produce *TDCM_TX_CLK*, which is the clock used to serialize the messages sent to the FEs on the TDCM to FE link. When the physical media of the TDCM to FE link is RJ45 copper cables, *M_REF_CLK* is transmitted on one cable pair that is distinct from the data pair. When using optical fiber, *TDCM_TX_CLK* is encoded with data.

Each FE receives *M_REF_CLK* directly from the TDCM when copper cables are used, or recovers *TDCM_TX_CLK* using a Clock and Data Recovery (CDR) circuit when using optical fiber media. The clock recovered from the TDCM link by each FE, *RX_REC_CLK$_i$*, is synchronous to *TDCM_TX_CLK* and consequently *M_REF_CLK*. After the appropriate division, each FE derives from *RX_REC_CLK$_i$* the clock for sampling detector signals, *SAMP_CLK$_i$* and optionally the clock for the local ADC. All *SAMP_CLK$_i$* have the same frequency and are synchronous to *M_REF_CLK*. Because *SAMP_CLK$_i$* may be obtained in each FE by a divider that is initially not synchronous with that of the other FEs, the phase of *SAMP_CLK$_i$* needs to be aligned among all FEs. The synchronization of the clock divisor of each FE is performed by the TDCM by sending a message on Virtual Channel A with the bit WCK_SYNCH set to one.
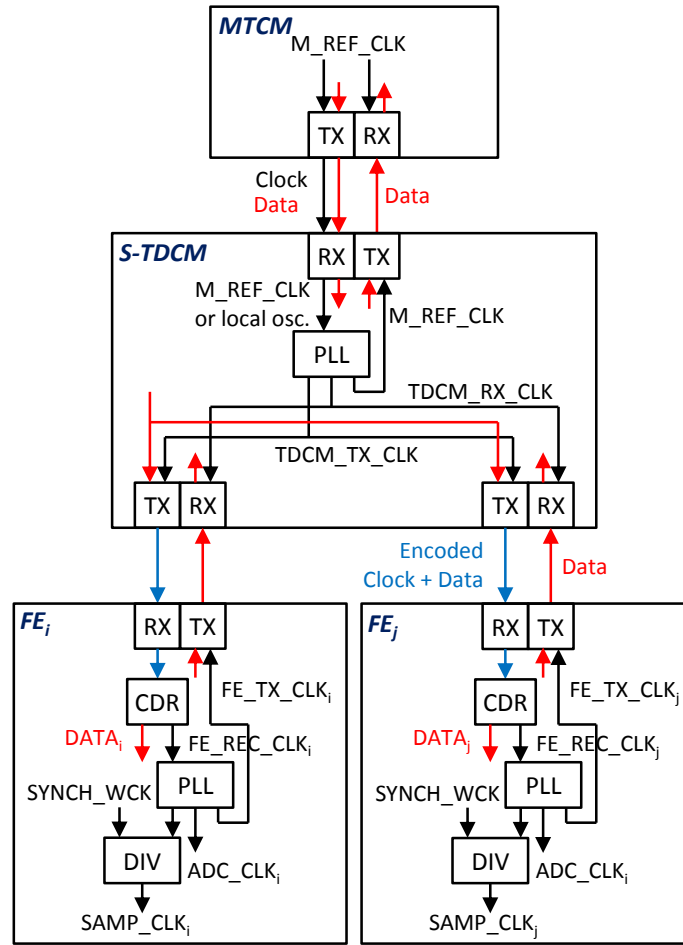
Fig. 22.   Global system clocking scheme.

Each FE also uses its own REC_CLK_i, after an optional multiplication, to derive $FE\_TX\_CLK_i$, which is the clock used on the TX side of the serializer transmitting messages back to the TDCM.

The TDCM uses M_REF_CLK, or a clock derived from it to produce TDCM_RX_CLK, the clock used to de-serialize the data received from each FE. Because each of the transmit clock $FE\_TX\_CLK_i$ used by the FEs are synchronous to M_REF_CLK, the TDCM can de-serialize data from all FEs using only one clock, $TDCM\_RX\_CLK$, and furthermore, this is a local clock. The TDCM does not have sufficient resources (e.g. clock trees) to recover a different receive clock from each of the FE and transport de-serialized data back in a common clock domain synchronous to M_REF_CLK. Data capture from all FEs can only be done directly in a clock domain synchronous to M_REF_CLK. This is not a limitation and, when the system is configured for self-trigger using front-end data, the latency of the links from the FEs to the TDCM must also be fixed so that trigger primitives from the different FEs are combined in a time-coherent way. Hence the FE to TDCM links must also be synchronous to the same common reference clock.

The TDCM also uses *M_REF_CLK* to serialize messages back to the M-TCM and the M-TCM also uses its own copy of this reference clock to de-serialize messages sent by the TDCM.

6.7 LINE ENCODING

6.7.1 LINK FROM THE TDCM TO THE FES

As previously explained, the link from the TDCM to the FEs transports time-interleaved messages from three virtual channels, named Virtual Channel A, B, and C. Line encoding before transmission over the physical media is performed in two steps. If the bit to be transmitted pertains to virtual Channel A or C, the first step of the encoding does not change this bit. If the bit to be transmitted pertains to Virtual Channel B, it is inverted. In other words, the data of Virtual Channel A and C are transmitted as they are and the data of Virtual Channel B are inverted before transmission. This first step of encoding allows the delineation of the three virtual channels at the receiver side. When no message is sent over the three virtual channels, the data send over the media (before the second step of encoding) is a fixed repetitive pattern composed of 4 bit, "0100", where the bits equal to '1' indicate the bits of Virtual Channel B. After the bits of Virtual Channel B are identified, the previous bit and the next bit belong to Virtual Channel A, and the second bit after the one that marks the position of Virtual Channel B is from Virtual Channel C.

Data may be transmitted directly over the media after the first step of encoding when using DC-coupled copper cables between the TDCM and the FEs. Because the first step of encoding does not provide a null DC component, it may not be adequate for AC-coupled media or transmission over optical fibers. In this case, a second step of encoding is performed. The second step uses Manchester encoding. For every original bit to be sent, two bits of data are transmitted: the original bit followed by its inverted value. The bit stream is therefore guaranteed to have exactly an equal number of 1's and 0's. However, the baud rate over the physical media is twice that of the original bit rate.

When no data are sent over Virtual Channel A, B and C, the fixed repetitive pattern "01100101" is continuously transmitted on the physical media. Transmitter signals after the two steps of the encoding are shown in the timing diagram Fig. 23.
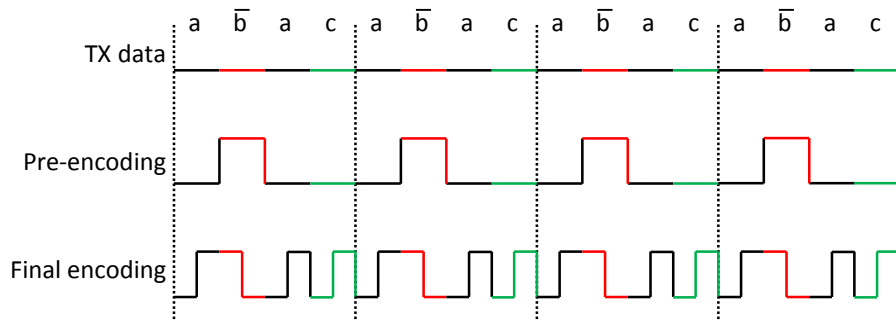
Fig. 23.   Pattern on the media when no messages are transmitted by the TDCM.

To achieve data reception and alignment, a receiver scans the received bit stream and perform bit slips until the above synchronization pattern is found. When synchronization is reached, the receiver retains one bit every two, and invert the bits that correspond to Virtual Channel B to retrieves the original bit stream. The serial bit stream of each virtual channel is then fed to its own de-serializer to obtain the messages sent over the three virtual channels.

One of the main advantages of the dual-step line encoding scheme proposed is that it allows the receiver to gain synchronization easily and without the need to reset or perform any specific training period on the transmitter side. Recall that in the TDCM to FE direction, a hardwired broadcast network is used. If the transmitter part of the TDCM is reset, all FE receivers must also be re-synchronized. With the proposed scheme, each FE can be reset at any time without the need to reset the TDCM which would impose to re-synchronize all the other FEs. To synchronize the receive link of a FE, it is sufficient that the TDCM stops sending messages to the FEs, so that the synchronization pattern is transmitted instead. The receiver on the FE side can gain synchronization on-the-fly, i.e. determine the optimal sampling time for the received data, and find the appropriate frame alignment of the received series of bits. Because the transmitted synchronization pattern is constant and repetitive, this operation is simplified compared to other encoding techniques like 8B/10B encoding or scrambling that do not produce both a constant and short period pattern when no data are transmitted.

By default, the line rate for the **TDCM to FE links is 100 Mbit/s, i.e. 200 Mbaud after encoding**. Sub-multiple or multiples of this base rate may also be supported in the future.

The TDCM can send data without inversion (default normal operation), or with inversion, to correct for the swap of the two sides of a differential pair for example.

Alternatively, instead of performing the inversion of data of Virtual Channel B, a different option is to invert instead the data of Virtual Channel A. In this case, the default pattern after the first step of encoding is "1010" which is precisely the clock to be transmitted to the FEs and is already a perfectly DC-balanced pattern. Hence, the second step of encoding may not be needed. This is shown in Fig. 24.
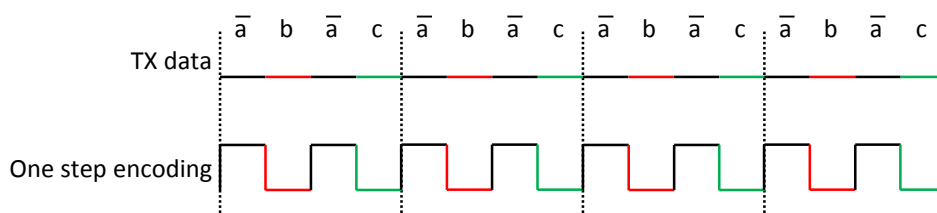
Fig. 24.   Line encoding with inversion of Virtual Channel A (no data being transmitted over any virtual channel).

However, this scheme does not provide a method to differentiate the bits of Virtual Channel B and C. Distinguishing if a message belongs to Virtual Channel B or C can easily be done by adding an extra bit to each message after the start bit: for example, a second '1' after the start bit indicates that the message belongs to Virtual Channel B, and a '0' after the start bit indicates that it belongs to Virtual Channel C. A potential limitation of this scheme is that the data stream becomes imperfectly DC-balanced when messages are sent over any virtual channel if no further encoding is done. Tests are needed to investigate if this imbalance can affect the operation or the robustness of the transmission over AC-coupled electrical cables or optical fiber media. The advantage of this scheme is that the baud rate on the transmission line is equal to the initial bit rate, i.e. there is no additional overhead caused by line encoding. However, until further investigations are made, this scheme is not selected and the baseline design of the TDCM uses data inversion on Virtual Channel B followed by Manchester encoding leading to a line rate of 200 Mbaud for the original 100 Mbps bit stream.

6.7.2 LINK FROM THE FE TO THE TDCM

In the front-end to TDCM direction, a set of point-to-point links is used. Each link carries messages from the three Virtual channels A, B, C, as defined earlier in this document. In the same way as it is done for the TDCM to FE link, message encoding before transmission is done in two steps. At the first step, the bits that correspond to Virtual Channel B are inverted. Consequently, when no traffic is sent by a FE, the data transmitted after the first step of encoding is a repetitive fixed pattern equal to "0100". This allows the receiver, on the TDCM side, to synchronize and determine which bit corresponds to which virtual channel. The bits equal to '1' correspond to Virtual Channel B. The bits preceding them belong to Virtual Channel A, and the two bits after the ones marking Virtual Channel B belong to Virtual Channel C.

The second step of encoding uses a self-synchronizing scrambler. Although other polynomials or a different encoding technique may be supported in the future, the baseline design of the TDCM uses the polynomial $x^{43} + 1$. This polynomial is used by the Asynchronous Transfer Mode (ATM) standard in telecommunication networks. As it is shown in Fig. 25, the implementation of the scrambler and de-scrambler are very simple.
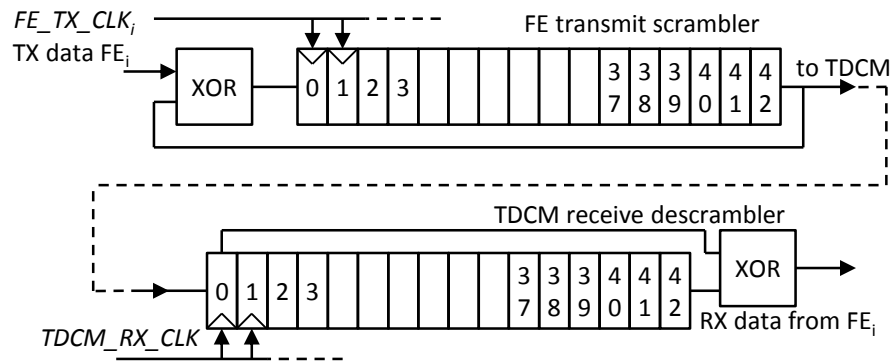
Fig. 25.   FE Transmit scrambler and TDCM receive descrambler.

Having a simple decoder is advantageous for the TDCM, because there must be one decoder per port, and the TDCM is planned to support 32 ports. Other advantages are that the encoding does not add any overhead and receivers are self-synchronized, which means that the scrambler and the de-scrambler can be started at a different time and with different initial seeds in their respective shift register. Self-synchronizing scramblers are commonly used in telecommunications and various other applications. They are generally preferred over additive scramblers because of their self-synchronizing property. The main disadvantage of self-synchronizing scramblers, also called "multiplicative scramblers", is that a single bit error at the input causes multiple bit errors at the output of the de-scrambler. In our case, additional protection on the data (parity, CRC-32) and the expected low bit error rate will mitigate that effect.

Both ends of the link must use the same clock: in standalone mode, the TDCM uses its own local reference clock or the clock supplied by the M-TCM to de-serialize and de-scramble serial data from the FEs. Reciprocally, the FEs use the clock recovered from the RX side of their FE-TDCM link not only to de-serialize the data received from the TDCM but also to encode and serialize data transmitted back to the TDCM. See the section on clocking for details.

By default the **FE shall transmit data to the TDCM at 400 Mbps, leading to 400 Mbaud** on the physical media. Running at a sub-multiple or a multiple of the base rate may be supported in the future.

6.8 ESTABLISHING COMMUNICATION BETWEEN THE TDCM AND THE FES AND HANDLING LOSS OF SYNCHRONIZATION

After powering up the system, none of the communication links between the TDCM and the FEs are operational. By default, each FE constantly tries to establish communication with the TDCM by searching for the synchronization pattern "01100101" on its receiver. The transmitter part of the FE is kept idle, i.e. at a fixed level without any transition.

When the TDCM is ready, the default synchronization pattern starts to be sent to all FEs. As long as the TDCM does not send any message over the three different virtual channels, the fixed synchronization pattern appears on the physical media. After a FE detects the synchronization pattern and successfully locks on it, the transmitter part of that FE is enabled and it sends a synchronization pattern composed of alternating 1's and 0's for a pre-defined fixed duration. After that, it switches to the scrambler mode. The TDCM captures the data received from each FE using its own local clock which is also used for transmitting data to the FEs. The TDCM does not have means to recover the clock from the serial stream received from each FE. Consequently, all the FEs must use the clock recovered from the TDCM to FE link to also clock the transmitter part of the FE to TDCM link. The whole system must be synchronous to the clock supplied by the TDCM, which originates from the M-TCM or some other master device. Nonetheless, the phase shift introduced by a FE between the primary clock in the TDCM and the serial data sent by this FE when it reaches the TDCM cannot be predicted. The TDCM must therefore determine on a per-FE basis the optimal time for sampling the serial data transmitted by the FEs. The initial value of this delay is determined by the TDCM when communication with the FE is established using a training pattern composed of alternating 1's and 0's. After a FE has acquired synchronization on the receive side, it must transmit the synchronization pattern 010101… for a duration of 660 ms before it switches to the self-synchronizing scrambler mode used for normal operation. During the link training period, the TDCM determines the optimal delay for sampling the received data. After this initial training period, the TDCM may need to adjust dynamically the data sampling point to compensate for the possible drifts of the delay of the TDCM-FE-TDCM communication loop. When the TDCM starts to find transitions on the receiver side of a link connected to a FE, it determines the optimal delay to apply on the received data to recover an alternating series of 1's and 0's. At 400 Mbps, the range for adjusting the delay line is 2.5 ns. When the initial delay calibration phase has been completed, the receiver part of the TDCM corresponding to the appropriate FE is switched to de-scrambler mode. Synchronization is gained when the TDCM detects the repetitive pattern "0100" in the received de-scrambled data. The '1' indicates the position of Virtual Channel B, and after these bits are inverted, the received data can be fed to the own de-serializers of the three virtual channels of the appropriate FE in the TDCM.

Assuming that synchronization in both directions is established, the TDCM starts transmitting messages to the FEs and gets replies back. This continues as long as communication remains stable. If a FE or the TDCM receives an excessive number of errors, or if the TDCM fails to receive a response from a FE, this indicates that the communication is broken at some place.

To signal to the TDCM an abnormal situation, a FE should stop sending any data on its transmitter. The TDCM monitors that transitions occurs on each of its receivers. If no

transition occur during more than a certain number of unit intervals, the TDCM considers that communication with this FE is mal-functioning and an alarm flag is raised. The TDCM may try to reset the FE by sending it some appropriate message. However, it is possible that this operation will not succeed because the communication link may be broken in the TDCM to FE direction. In this case, the only recovery action that can be made is to power cycle the faulty FE and perform link synchronization again. If synchronization still cannot be obtained, it may be a permanent hardware failure, on either side, and the appropriate actions have to be made by the user: more advanced diagnosis, hardware replacement, or fault isolation. Note that if the TDCM is reset, this will also reset the communication with all the FEs and all links will need to be re-synchronized.

The FE is also supposed to check that transitions occur on its receiver. It may verify that every received bit is followed by its inverted value (Manchester encoding rule). The exact method is not specified, but the absence of a sufficient number of transitions, the observation of repeated violations of the encoding rules, the detection of parity errors on received messages are some of the indicators that can be used by each FE to determine a link fault condition. When faults exceed some alarm level, the FE shall simply stop its transmitter: the TDCM will detect this situation and trigger the appropriate recovery actions.

6.9 TDCM to FE Link Bit Error Rate Tester

The TDCM comprises several pseudo-random bit stream generators for testing the quality of the communication link from the TDCM to the FEs. The industrial and telecommunication standard patterns PRBS7, PRBS15, PRBS23 and PRBS31 are supported by the TDCM. These patterns can easily be generated with a linear feedback shift register whose taps correspond to the coefficients of the polynomial:

PRBS7 = $x^7 + x^6 + 1$

PRBS15 = $x^{15} + x^{14} + 1$

PRBS23 = $x^{23} + x^{18} + 1$

PRBS31 = $x^{31} + x^{28} + 1$

These PRBS can be generated with or without the additional Manchester encoding step used for the TDCM to FE link. The FE shall at least support checking errors with each of these standard patterns at 100 Mbps data rate with Manchester encoding, i.e. 200 Mbaud on the physical media. This is the normal default mode of operation of the TDCM to FE links. Other speeds and PRBS sent without further encoding are primarily meant to test the serializers of the TDCM with an oscilloscope or some standard serial data analyzer equipment.

In the TDCM, the bit error rate tester in the TDCM to FE transmit direction is enabled by setting the bit MT_BERT_ENA to one. The speed and pattern generated are determined by the fields MT_RATE and MT_BERT_PAT respectively. The bit MT_MANCH_ENA determines if Manchester encoding is enabled or disabled. For operation with the FE, the bit should always be set to 1, but it may be set to 0 when an oscilloscope or other test equipment is connected to the TDCM. The bit MT_INVERT is used to optionally invert the data transmitted. The different patterns, speeds, and encoding options supported by the TDCM are listed in Table 8 and Table 9.

Table 8 . TDCM to FE link bit error rate tester patterns.

MT_BERT_PAT (TDCM side) FE_RX_BERT_PAT (FE side)	Resulting pattern
00	PRBS7
01	PRBS15
10	PRBS23
11	PRBS31

Table 9 . TDCM to FE link bit error rate speed and encoding.

MT_RATE	MT_MANCH_ENA	Resulting pattern
00	0	PRBSx 100 Mbps (100 Mbaud)
00	1	PRBSx 100 Mbps + Manchester (200 Mbaud)
01	0	PRBSx 200 Mbps (200 Mbaud)
01	1	PRBSx 200 Mbps + Manchester (400 Mbaud)
10	1	PRBSx 400 Mbps (400 Mbaud)
10	0	Not supported – generates PRBSx 400 Mbps (400 Mbaud)
11	0	100 MHz clock
11	1	200 MHz clock

The TDCM can force the generation of a bit error by making a transition from 0 to 1 on MT_BERT_DOERR.

For proper operation of the bit error rate tester, it must be first enabled on the FEs. This is accomplished by setting the bit FE_RX_BERT_ENA in each FE after the appropriate pattern has been selected by setting FE_RX_BERT_PAT. The polarity inversion and Manchester decoder must also be selected. When FE_RX_BERT_ENA is enabled on the FE, it no longer routes the bit stream received from the TDCM to the de-serializers of Virtual Channel A, B and C. Instead, the FE continuously tries to synchronize its own PRBS pattern generator waiting for a specific value to appear in the received bit stream, for example 7, 15, 23 or 31 consecutive bits equal to 1 according to the selected pattern. When this seed value is found, the FE starts its own local PRBS generator and compares

its output on the fly against the received bit stream in order to detect and count errors. The FE shall accumulate bit errors in a 16-bit saturating counter and count the number of received bits, divided by 10^6, in a 32-bit counter. Assuming a bit rate of 100 Mbps, the counter of received bits will roll-over after more than one year. The FE shall send to the TDCM every 10^7 received bits a specific packet on Virtual Channel C that contains the number of bit received and the number of errors detected. See packet format details in the appropriate section. At 100 Mbps, the TDCM will receive one such packet from each FE every 10 s. This procedure continues as long as it is desired.

To end a bit error test, the user must first stop the PRBS generator on the TDCM by clearing bit MT_BERT_ENA. This causes the TDCM to switch transmission to the default fixed repetitive synchronization pattern "01100101". While in bit error rate tester mode, each FE must not only detect bit errors but it must also search for multiple consecutive occurrences of the periodic default synchronization pattern. When the number of the consecutive occurrences of this pattern exceed the number of bits of the shift register of the PRBS generator, it indicates that the TDCM has stopped the test. Then each FE shall also stop running in bit error test mode and shall redirect received bits to the de-serializers of Virtual Channel A, B and C so that normal communication with the TDCM is restored. The TDCM ends the procedure by clearing the FE_RX_BERT_ENA bit in each FE. Note that the FE must delay the accumulation of bit errors by the appropriate amount of time to guarantee that when the TDCM switches from the PRBS pattern back to the default synchronization pattern to indicate the end of the test period, the FE does not interpret the occurrences of the synchronization pattern as bit errors.

6.10 FE TO TDCM LINK BIT ERROR RATE TESTER

In order to test the integrity of the FE to TDCM communication links, a bit error rate tester is implemented. In many aspects, it is similar to the BERT used in the TDCM to FE direction as described in the previous section. There are however a few differences. A first difference comes from the fact that the TDCM to FE communication uses a broadcast topology while the return path is based on a series of point-to-point links. A second difference is that for the TDCM to FE BERT, normal communication from the TDCM to the FE is interrupted during the test, while it is FE to TDCM communication path that is suspended when the FE to TDCM BERT is running.

The FE to TDCM BERT implements the same PRBS as its counterpart, namely PRBS7, PRBS15, PRBS23 and PRBS31. It is required that the FE supports at least the nominal speed of 400 Mbps. Other speeds, may be supported in the future. When the BERT is running, the output scrambler on the transmit side of the FE must be disabled. The selected PRBS pattern drives directly the serial transmitter of the FE without any further encoding. Note that the PRBS is a DC-balanced pattern that can be transmitted over AC-coupled or optical media.

The proper sequence of operation for using the FE to TDCM link BERT is the following. Firstly, the same PRBS must be set at both ends by the FE_TX_BERT_PAT configuration bits in the FEs (see the description of FE Register#6 on Fig. 68 for details) and BE_RX_BERT_PAT configuration bits in the TDCM. The pattern generated according to these configuration bits is given in Table 10.

Table 10. FE to TDCM link bit error rate tester patterns.

BE_RX_BERT_PAT (TDCM) FE_TX_BERT_PAT (FEs)	Resulting pattern
00	PRBS7
01	PRBS15
10	PRBS23
11	PRBS31

Secondly, the BERT must be armed in each of the FEs participating to this test. This step is accomplished by setting the bit FE_TX_BERT_ENA to 1 in each of the desired FE. Thirdly, the BERT must be armed in the TDCM by setting the bit BE_RX_BERT_ENA to 1. At this point, none of the FEs have started the actual transmission of any PRBS and the communication between the FEs and the TDCM is still operational in both direction.

The BERT is effectively started synchronously in all the participating FEs and the TDCM by sending over Virtual Channel A a message with the bit BERT_START equal 1 during one clock period. At this time, each FE shall switch the input of its serializer from the normal data scrambler mode to the selected PRBS generator. Simultaneously, the TDCM shall disconnect the Virtual Channel deserializers from its receiver links and shall try to acquire synchronization on the PRBS pattern. The TDCM only uses one PRBS generator to check the pattern received from each FE. It is mandatory that all FEs start simultaneously and use the same seed. This initial seed must be 11111..11 (the number of 1 is 7 for PRBS7, 15 for PRBS15, etc.). To acquire synchronization, the TDCM applies a programmable delay to the pattern received from each FE and locks when the received pattern matches the locally generated PRBS. If the latency from the TDCM to each FE and back is deterministic and is equal for every branch – which is what the system is designed to normally achieve – the delay applied to the received PRBS of each FE shall settle to the same value. Reading back this per-FE delay allows checking whether the actual link latency is equal or not on every branch and quantifying any mismatch.

As soon as BERT_START is received, the TDCM starts incrementing a counter at a rate of 400 million units per seconds. This represents the number of bit received. This is done independently on whether locking on the PRBS for any FE is done or not. The benefit of this scheme is that only one counter is needed for all FEs, while one receive bit counter per FE would be needed to count exactly the number of bits received from each FE

starting from the instant when lock was achieved for each particular FE. In principle, if the latency path from each FE is identical, locking on the PRBS for all FEs occurs simultaneously, so the same offset on the received bit count applies to all FEs. This offset corresponds to the number of bit received from the instant BERT_START was pulsed and the TDCM achieves lock on the PRBS pattern. This number is small compared to the total number of bits transmitted during a BER test measurement because locking normally takes a small fraction of a second while a BER test lasts from minutes to hours.

When locking is achieved (on a per-link basis), the TDCM compares the received pattern from each FE to its own locally generated version of the expected PRBS. It increments a 16-bit saturating error counter in case of mismatch. A separate error counter is used for the link of each FE.

The BER test continues until it is stopped by the TDCM synchronously by sending over Virtual Channel C a message with the bit BERT_STOP equal to 1. When BERT_STOP is received, each FE shall stop sending the PRBS pattern and return to the normal data and scrambler operation. Similarly, the TDCM shall stop checking the received data against the PRBS pattern and re-establish normal communication with the FEs.

It is permitted to start and stop a BER test an indefinite number of times by pulsing BERT_START and BERT_STOP successively. To really finish a BER test, it is necessary to disengage the tester after the last BERT_STOP by setting the bit FE_TX_BERT_ENA back to 0 in every FE and setting the bit BE_RX_BERT_ENA back to 0 in the TDCM.

During a BER test, each FE shall force the generation of a single bit error when receiving over Virtual Channel A a message with the bit BERT_DOERR equal to 1. This feature is used to verify that the BER tester is effectively capable of detecting single bit errors.

Note that the bits BERT_START, BERT_STOP and BERT_DOERR are dual function bits in the messages sent by the TDCM to the FEs over Virtual Channel A. Sending messages with any of these dual function bits set to FEs that are not running in BER tester mode does not have any impact for them, except that the counter of events received will be cleared when BERT_DOERR is pulsed. Anyway, this counter shall be cleared at the beginning of every normal data taking run, so this small side effect occurring during system test is insignificant.

6.11 FRONT-END IDENTIFICATION AND ENUMERATION PROCEDURE

The TDCM identifies each FE by a 5-bit index that corresponds to the index of the physical port where the fiber or cable of that FE is attached. Port are numbered from 0 and up to 31 on the TDCM. Besides this index, each FE is required to have a unique serial number that identifies it. This serial number is referred to as the DNA number of that FE. There are many different options for implementing this DNA number. It may be

implemented in a dedicated chip (e.g. Maxim Integrated DS28CM00), it could be programmed in a One-Time Programmable (OTP) register of the on-board FPGA (e.g. Xilinx e-Fuse technology), it could be coded directly in the firmware or the FPGA, or it can be set on-board by switches, etc. For example, Xilinx series 7 FPGAs implement a 57-bit DNA number that identifies each chip – however it may not be unique and Xilinx says that up to 32 chips can have the same DNA number, but this probability is supposed to be extremely small in practice.

The TDCM assumes that the DNA number of each FE is 64-bit. However, provided that this DNA number is different for each FE, it could be shorter (or even longer). If less than 64-bit of DNA number can be provided, these should be prepended with the appropriate number of 1's or 0's. If the DNA number available by the FE is more than 64 bits, the extra bits should be truncated, although it has to be verified that the 64 remaining bits are only found in one FE among all of those connected to the TDCM.

At power-up, the TDCM does not know which FEs are present, and the FEs themselves do not know to which port of the TDCM they are connected. The purpose of the present procedure is for the TDCM to know which FEs are connected and assign them the appropriate 5-bit ID. Each FE will be assigned the 5-bit ID that corresponds to the port of the TDCM where it is connected. The TDCM performs FE enumeration in two steps: firstly, it requests the DNA number of all FEs. Secondly, it associates each of the DNA number received with the appropriate port index. Then the TDCM sends all FEs the correspondence table between each DNA number and its assigned FE index. In the second step of the procedure, each FE captures from this table the index that is associated with its own DNA number.

Both steps of the procedure are conducted over Virtual Channel B by series of read and write operations broadcasted to every FEs. These operations are performed in the appropriate configuration register of the FEs to implement the serial protocol defined later in this text. The serial interface to retrieve the DNA number of the FE and obtain the FE index assigned by the TDCM takes the signals given in Table 11. These signals are internal to the FPGA of the FE and are mapped to Register #9 as shown in Fig. 70.

Table 11   . Signals to interface to the DNA number and FE index block.

Signal	Direction	Function
DNA_CLK	TDCM -> FE	DNA block serial clock
DNA_SEL	TDCM -> FE	DNA block active high select
DNA_RD_WR	TDCM -> FE	DNA block active low read / active high write
DNA_ID_KEY	TDCM -> FE	DNA block serial input for DNA and ID table keys sent to all FECs
MY_DNA_ID	FE -> TDCM	DNA block serial output for the DNA and ID of this FE

To retrieve, the DNA number of the FEs, the TDCM generates the sequence shown in Fig. 26. After asserting, DNA_SEL high and DNA_RD_WR low, a series of 70 serial clock cycles are made by pulsing DNA_CLK. At every clock cycle, from cycle #0 to #63, the FE places on MY_DNA the next bit of its own DNA number, starting from the MSB. On clock cycle #64, the FE places a bit set to 0 if the ID of this FE was not set, and a 1 if the ID of this FE was set. On clock cycles #65 to #69, the FE places the five bits of its assigned ID (MSB-first) if this ID was assigned, and 0 otherwise. Note that the FE asynchronously places its DNA number and ID bits after the rising edge of the serial clock. These data bits are therefore captured by the TDCM on the falling edge of the serial clock.
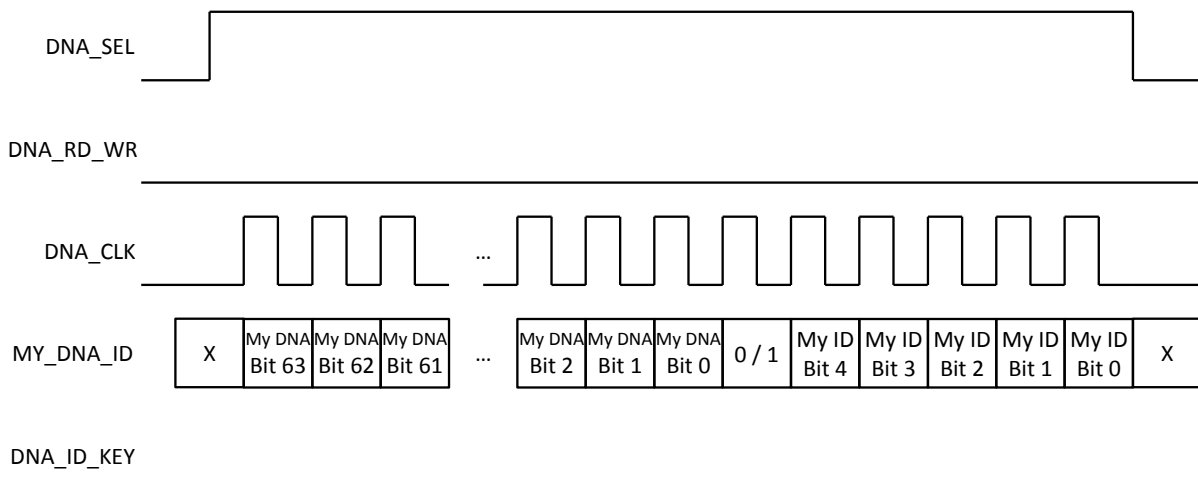
Fig. 26.   FE DNA number read operation.

After retrieving all DNA numbers of the FEs, the TDCM broadcasts to all FEs on Virtual Channel B the keys that are composed each of one DNA number and its corresponding assigned FE index. The FEs must listen to all keys, and when each individual FE detects its own DNA, it captures its associated index. The procedure of sending one DNA and index key also takes 70 serial clock cycles as shown in Fig. 27. After asserting DEV_SEL high and DEV_RD_WR high, the TDCM places on DNA_ID_KEY successively the 70 bits of a DNA value at each serial clock cycle. On clock cycle #64, the TDCM normally places 1 to clear in the corresponding FE the bit that tells that the ID of this FE was set. It can also place a 0 instead to preserve the state of this bit. On clock cycles #64 to #69, the TDCM places the 5-bit FE index (MSB first) matched to the DNA number it has just sent. Note that the TDCM places serial data on DNA_ID_KEY on the rising edge of the serial clock. The FE should therefore capture this data on the falling edge of the serial clock. The FE that recognized its own DNA in the first part of the message must assign its index to the supplied value, and it must set to 1 the bit that tells that its own index was set after the 5-bits of the Key ID have been received.
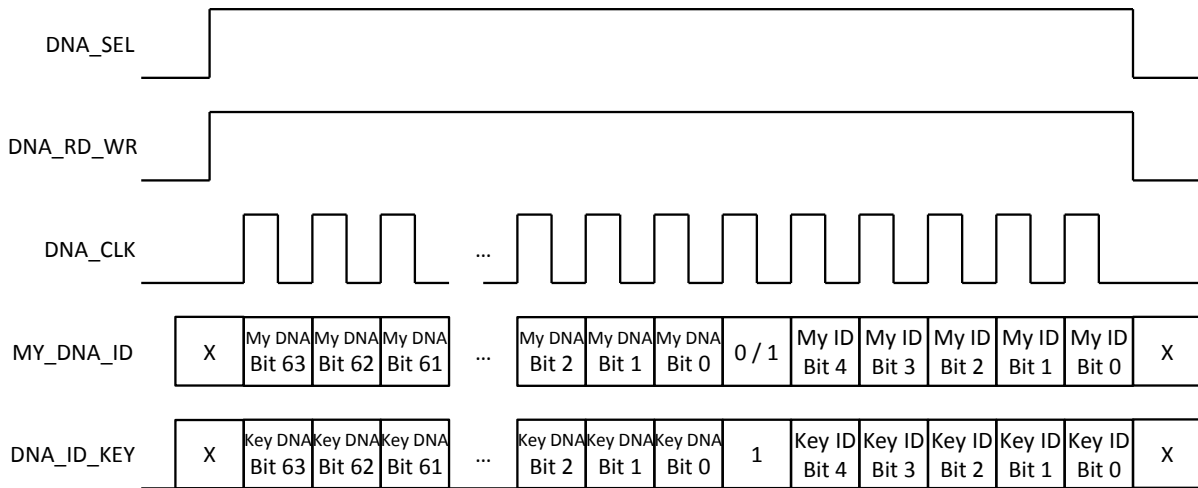
Fig. 27.   DNA number and FE index key write operation.

After the TDCM has pushed to the FEs all the DNA-index keys, every FE should have its index assigned. To verify this, the TDCM requests again all DNA numbers of the FEs and checks that in all cases the 64th bit is set and the 5-bit index that follows effectively corresponds to the expected port index. If power is lost on any FE or on the TDCM, the FE identification and enumeration procedure must be re-run for proper operation.

The 5-bit index of the FE is available in read-only in Register #9 as shown in Fig. 70 (field CARD_ID). This index is also placed in the header of every packet of event data sent from a FE to the TDCM to identify the origin of the packet. For coherent event building and global event data consistency, it is critical that each FE is assigned the correct index. The 5-bit index of the FE is also used to determine to which particular FE the operations transmitted over Virtual Channel B apply (except for broadcast operations). For proper FE configuration and monitoring, it is also critical that each FE knows its correct ID. Finally, the ID of the FEs are also used to determine to which FEs the commands sent on Virtual Channel C apply. Again, it is critical that every FE knows its correct ID so that transfer of event data to the TDCM works properly.

7 TDCM LOCAL REGISTER AND MEMORY MAP

The run-time configuration parameters of the TDCM and the interaction between the embedded processor integrated in the local FPGA and the logic implemented in its fabric are controlled via a set of registers and memory blocks.

7.1 CONFIGURATION REGISTERS

The TDCM implements a bank composed of thirty-two 32-bit wide control and status registers. The first sixteen registers are accessible in read/write by the local processor, with the exception of a few bit fields that are read-only, while the other half are read-only registers. These give access to the processor to the status and monitoring information on the downstream logic.

The configuration and status register bank is accessible over the AXI-4 bus and is mapped to the user address space of the local processor. The list of registers is given in Table 12 and Table 13.

Table 12. TDCM register bank – Configuration registers.

Register	Name	Main function
#0	-	Various configuration fields
#1	TX_B_DATA	Virtual Channel B transmit data
#2	TX_B_ADDR	Virtual Channel B transmit address
#3	-	Various configuration fields
#4	DATA_PUMP	Determine the set of FE included in DAQ
#5		Various configuration fields
#6	-	Various configuration fields
#7	EVENT_GEN	Local event data generator settings
#8	TRIG_LATENCY_01	Trigger latency settings (first half)
#9	TRIG_LATENCY_23	Trigger latency settings (second half)
#10	FE_ACTIVE	Determine the set of active FE
#11	MULT_TRIGGER	Self-trigger multiplicity settings
#12	-	Available for future use
#13	MTCM	Controls the interface to the Master TCM
#14		Various configuration fields
#15	-	Available for future use

Table 13. TDCM register bank – Status and monitoring registers.

Register	Name	Main function
#16	FE_UP	Set of FE with whom communication is established
#17	FE_SAMPLING	Set of FE currently sampling detector signals
#18	FE_BUSY	Set of FE busy with the readout of the current event
#19	RX_BERT	Mbit received by TDCM in bit error rate tester mode
#20	RX_BERT_ERR	Received bit error count in bit error rate tester mode

#21	STAT_CNT	Message statistics counter
#22	TX_CNT	Count of message transmitted to FE
#23	PUMP_RUNNING	Set of data pumps in the running state
#24	PUMP_STALLED	Set of data pumps in the stalled state
#25	EB_SOE	Set of Start-Of-Event expected by the event builder
#26	EB_EOE	Set of End-Of-Event expected by the event builder
#27	-	Event builder, packet mover, trigger generator status
#28	EVENT_RX	Number of events/triggers received from M-TCM
#29	EVENT_TX	Number of events/triggers sent to FE
#30	MTCM Status	Multi-function register for status of M-TCM interface
#31	-	Available for future use

General Configuration (Register #0):

This register is shown Fig. 28.
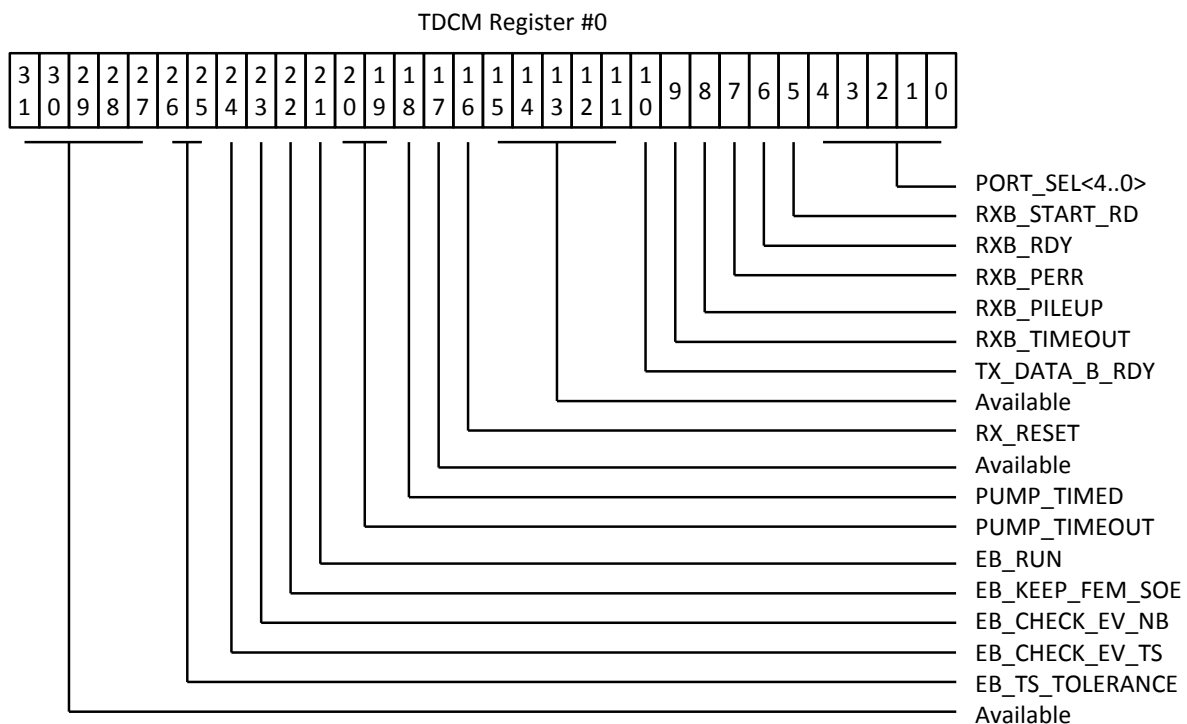
TDCM Register #0

Fig. 28.   TDCM General Configuration Register #0.

The field PORT_SEL determines which of the 32 front-end ports is currently selected for various types of operation such as sending a message over Virtual Channel B, reading a message counter, etc. All the operations that use this field must set it to the desired value before performing the actual operation because it may have been changed by some other previous operation.

The field START_RX_RD is used to start the readout of a message received over Virtual Channel B from the FE port selected by PORT_SEL. The field RXB_RDY (read-only)

49

indicates that a message on Virtual Channel B from the selected port is available for receive. The field RXB_PERR (read-only) indicates that the current message received over Virtual Channel B has a parity error. The field RXB_PILEUP (read-only) indicates that a new message was received on Virtual Channel B of the selected port before the previous message was readout. This is not allowed by the protocol. Only the last message can be read-out and all the other previous messages received over Virtual Channel B on that port are lost. The field RXB_TMEOUT (read-only) indicates that no response message was received over Virtual Channel B from the selected port within the expected time after the transmission of a request message. This is an error situation because every front-end shall return a response message to each request received over Virtual Channel B.

The field TX_DATA_B_RDY is used to initiate the transmission of a message over Virtual Channel B. Transmission occurs when this bit is changed from 0 to 1. It must be cleared to 0 before the next message can be sent. The destination and the content of the message being sent is determined by the registers TX_B_DATA and TX_B_ADDR.

The field RX_RESET is used to reset the receive link of the port selected by PORT_SEL. The receive link is held in reset as long as this bit is set to 1. When it is cleared, the TDCM attempts to synchronize the receive link on the expected null traffic pattern.

The bit PUMP_TIMED determines if the Data Pump operates in timeout mode or infinite wait mode. The field PUMP_TIMEOUT determines the timeout value when operating in timeout mode. The values 00, 01, 10 and 11 correspond to 1 ms, 10 ms, 100 ms and 1 s respectively. See the description of the Data Pump for details.

The bit EB_RUN is used to start and stop the local event builder. The field EB_KEEP_FEM_SOE determines if the event builder forwards to the DAQ or drops the Start Of Event packet transmitted by each front-end at the beginning of each event. The field EB_CHECK_EV_NB determines if the local event builder performs the verification of the event numbers sent by each front-end in their respective Start-Of-Event packet. The bit EB_CHECK_EV_TS determines if the local event builder performs the verification of the event timestamp sent by each front-end in their respective Start-Of-Event packet. The field EB_TS_TOLERANCE determines the allowed mismatch between timestamp values when timestamp verification is enabled in the local event builder. The values 00, 01, 10 and 11 corresponds to an allowable mismatch of 0 units, 1 unit, 2 units and 4 units respectively.

Virtual Channel B Transmit Data (Register #1):

This register is shown in Fig. 29.

TDCM Register #1
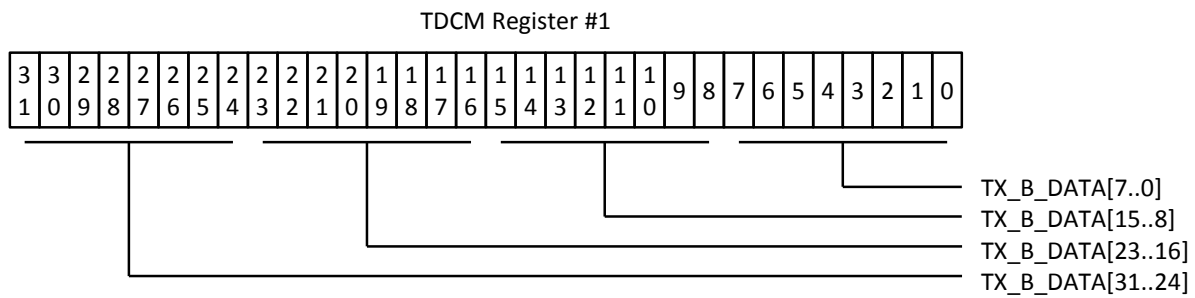
Fig. 29.   Virtual Channel B Transmit Data.

This registers contains the four bytes of data to be transmitted in a message over Virtual Channel B. The content is only relevant for write operations. It is recommended to set it to 0 for read operations.

Virtual Channel B Transmit Address and Control (Register #2):

This register is shown in Fig. 30.

TDCM Register #2
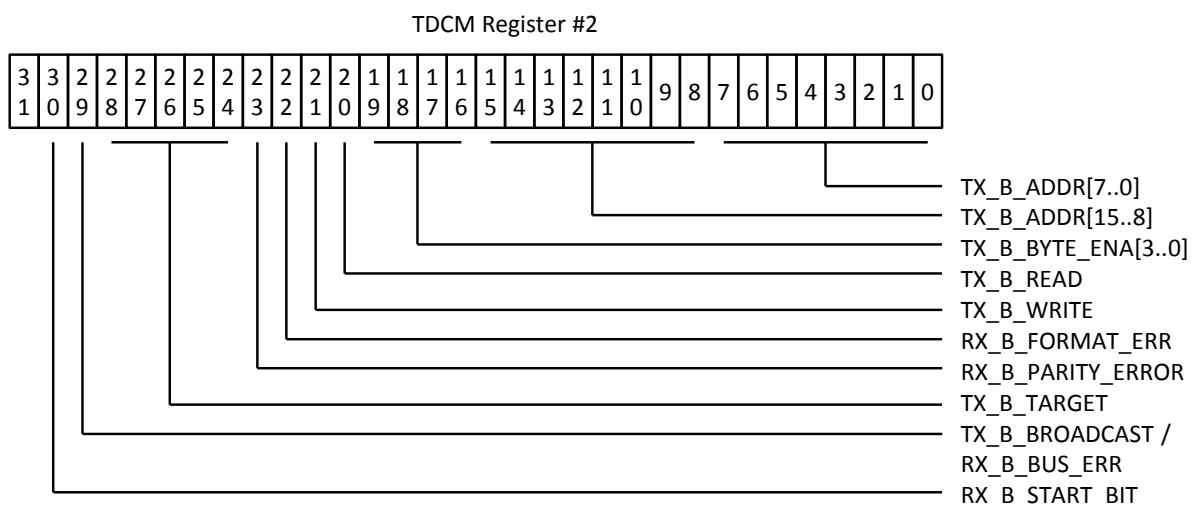
Fig. 30.   Virtual Channel B Transmit Address and Control.

This registers contains the four bytes of the address to be transmitted in a message over Virtual Channel B as well as the destination and control information.

The field TX_B_ADDR specifies the 16-bit address to which the remote read or write operation will take place over Virtual Channel B. The field TX_B_BYTE_ENA determines which of the 4 bytes in the 32-bit wide location addressed are concerned by the remote operation. The field TX_B_READ specifies a remote read operation. The field TX_B_WRITE specified a remote write operation. It is permitted to send a message with neither TX_B_READ nor TX_B_WRITE set. No operation is performed in this case, but a response message must be sent by the target front-end. However, setting

51

simultaneously TX_B_READ and TX_B_WRITE is not permitted. The field RX_B_FORMAT_ERR and RX_B_PARITY_ERR are not used in the transmit direction and are only assigned by the front-end in response messages. The field TX_B_TARGET determines the destination front-end of the message. When the TX_B_BROADCAST bit is set to 0, the destination field is a single node, and when this bit is se to 1, the message targets all front-end nodes. In the receive direction, the RX_B_BUS_ERR bit indicates that the requested operation could not be completed successfully. This typically indicate an attempt to access an invalid remote address location. The bit RX_B_START_BIT indicates the state of the receive start bit. In the transmit direction, the start bit is automatically inserted by the firmware. Note also that the parity bit is automatically computed and inserted at the end of a message, so that the total length of messages exchanged over Virtual Channel B is 64 bits.

General Configuration and Control (Register #3):

This register is shown in Fig. 31.
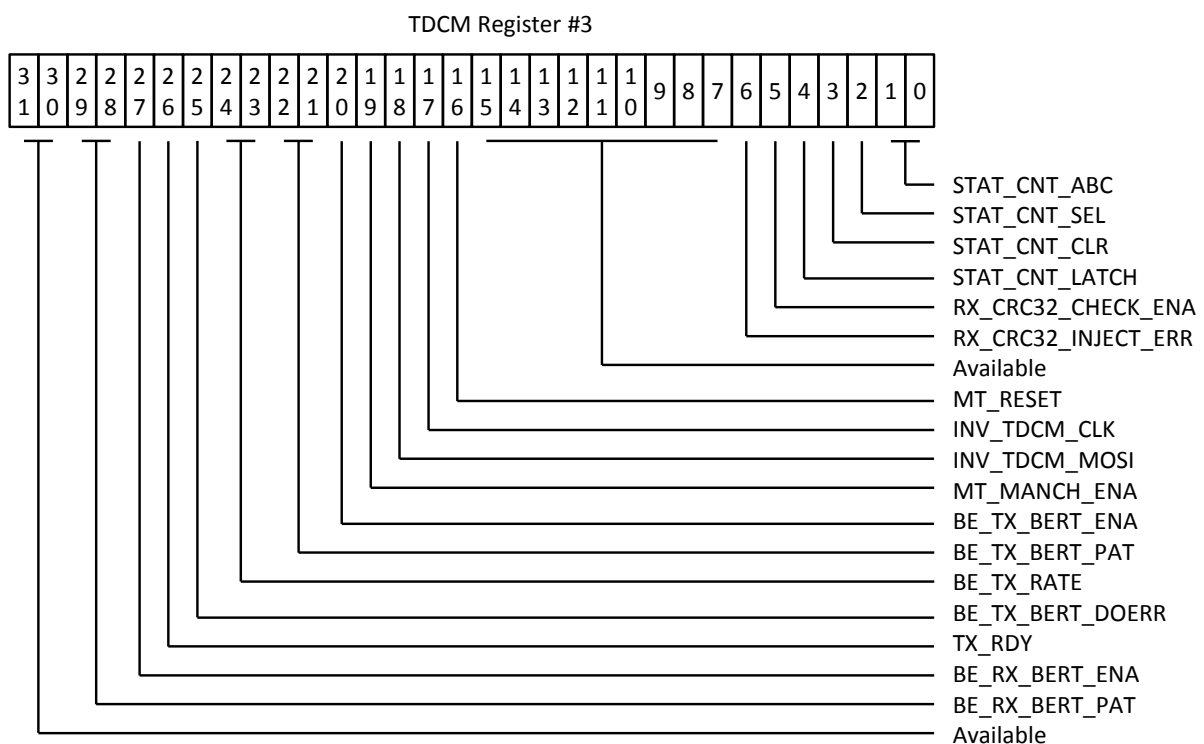
TDCM Register #3

Fig. 31.   General Configuration and Control Register #3.

The STAT_CNT_ABC field is used to determine from which Virtual Channel A, B, or C, the transmit and receive message counters and the receive error message counters will be captured. The capture of the selected message and error counters occurs when STAT_CNT_LATCH is changed from 0 to 1 and latching occurs on all ports simultaneously on RX and on TX. For the receive direction, two counters per port are latched when the

selected Virtual Channel is A or B. These are: 1) the number of correct messages received, 2) the number of messages received with an error. For Virtual Channel C, the following receive counters are latched on each port: 1) the number of packets correctly received, 2) the number of packets with overrun error, 3) the number of packets with a format error, 4) the number of packets with a size error, 5) the number of packets with a CRC-32 error. In the transmit direction, there is only one port because a multi-cast network is used. For Virtual Channel A, B and C the number of transmitted messages or packets is counted. After the RX counters of all ports on the selected Virtual Channel have been latched, STAT_CNT_SEL and PORT_SEL can be changed to select for read-out the RX message counter or the RX error counters of one or several of the desired ports. Neither STAT_CNT_SEL nor PORT_SEL have any effect on the TX counter that was latched. Once the set of receive and transmit message counters of a Virtual have been captured, the latched values remain unchanged until a new transition from 0 to 1 occurs on STAT_CNT_LATCH. The value of the TX and RX latched counters are available in the Transmit Message Counter and Statistics Message Counter Register of the TDCM respectively. See Register #22 and Register #21 for definition and details.

The capture of message counters within the TDCM is summarized in Table 14.

Table 14. Capture of TDCM receive message counters.

STAT_CNT_ABC	STAT_CNT_LATCH	Counters latched
XX	0 or 1 stable or ↓	unchanged
00	↑	Virtual Channel A TX and RX message and error counters for all ports
01	↑	Virtual Channel B TX and RX message and error counters for all ports
10	↑	Virtual Channel C TX and RX message and error counters for all ports
11	↑	Reserved

When the bit STAT_CNT_CLR is set to 1, this clears simultaneously the TX and RX counters of all ports for the Virtual Channel selected by STAT_CNT_ABC. The bit STAT_CNT_CLR must be cleared to 0 to enable the message counters. When STAT_CNT_ABC is set to "11"' and STAT_CNT_CLR is activated, all message and error counters for both TX and RX on all ports and the three Virtual Channels are cleared in the TDCM.

The bit RX_CRC32_CHECK_ENA is used to enable (1) or disable (0) the verification of the CRC-32 on the packets received over Virtual Channel C. Except for testing purposes, it is advised to always enable CRC-32 verification. Obviously, the CRC-32 insertion feature must also be enabled on all the front-ends.

The bit RX_CRC32_INJECT_ERR is used to force the verification of the CRC-32 of a received packet to fail. The error is introduced for the next non-null packet received over Virtual Channel C after this bit is set from 0 to 1. This feature is useful for testing the behavior of the local event builder of the TDCM in case of errors.

The bit MT_RESET is used to reset the main transmitter of the TDCM. Because a multi-cast network is used, setting this bit to 1 will affect the transmit link from the TDCM to every FE. The transmit path is held in reset as long as MT_RESET is held at 1. When it is cleared to 0, the TDCM starts transmitting the null-data synchronization pattern. If the transmit path of the TDCM is reset, it is also necessary to reset the receive path of every front-end and re-synchronize all links.

The bit INV_TDCM_CLK is used to optionally invert the clock transmitted from the TDCM to the FEs. This setting is only used when the physical layer between the TDCM and the FEs uses RJ45 cables where a dedicated wire-pair is used to transport the reference clock. With the optical physical layer, the clock is embedded with data over the same physical media.

The bit INV_TDCM_MOSI is used to optionally invert the serial stream transmitted by the TDCM to the FEs. This inverts only the data pair when the physical layer between the TDCM and the FEs are RJ45 cables and it inverts the encoded clock and data with the optical media. The inversion applies to all TDCM to FE links because the same copy of the serial bit stream is used for multi-cast.

The bit MT_MANCH_ENA is used to enable or disable DC-balanced encoding of the serial data sent to the FEs. At present, setting this bit to 1 enables Manchester encoding as explained in section 6.7.1 of this document. Non-DC balanced encoding may only be used when the interface between the TDCM and the FEs uses DC-coupled media or for testing purposes in bit error tester mode.

The bit BE_TX_BERT_ENA is used to enable or disable the bit error rate tester of the TDCM in the transmit direction. The field BE_TX_BERT_PAT, BE_TX_RATE and MT_MANCH_ENA determine the type of pseudo-random bit sequence generated and the transmission rate as explained in Table 8 and Table 9.

The bit BE_TX_BERT_DOERR is used to force the transmission of a single bit error to the FEs in bit error tester mode. This bit is used to verify the error detection capability of the FEs in the receive direction. Currently, BE_TX_BERT_DOERR has no effect in normal running mode, but this may be changed in the future to be able generating errors also in normal running mode.

The bit TX_RDY is a read-only value that indicates that the transmit path from the TDCM to the FEs is operational after a reset.

The bit BE_RX_BERT_ENA is used to enable (1) or disable (0) the bit error rate tester function of the TDCM in the FE to TDCM direction. It is used in conjunction with the field BE_RX_BERT_PAT to determine which pseudo-random bit pattern is expected to be received. See section 6.10 for details on this bit error rate tester.

Data Pump Enable (Register #4):

This register is shown in Fig. 32.
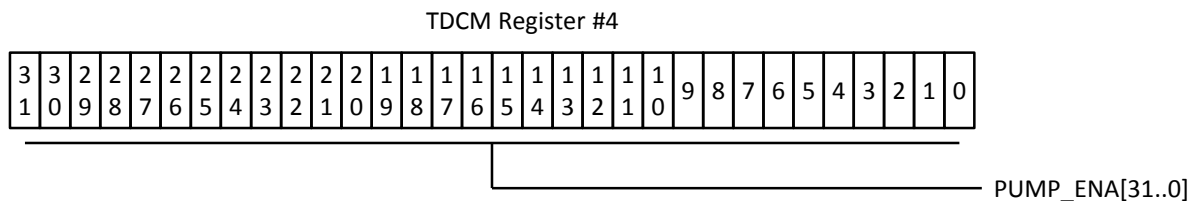
TDCM Register #4

Fig. 32.   Data Pump Enable Register.

This register determines from which set of FEs the TDCM shall gather data to assemble the events that are forwarded to the external data acquisition system. Every bit in this register corresponds to one physical port to a FE. Setting a 1 at one position means that the corresponding FE is part of the data acquisition of the current run.

When at least one bit is set in this register, the TDCM sends data requests over Virtual Channel C to the FEs that are activated provided that enough space is available in the corresponding receive FIFO buffer for storing at least one data packet of the maximum expected size. Assuming that the Data Pump is set to operate in timeout mode, each of the FE to whom data have been requested must respond within the allowable timeout period with an empty or non-empty data packet. Failure to do so is considered as an error that will freeze the local event builder until the underlying problem is solved.

General Configuration (Register #5):

This register is shown in Fig. 33.
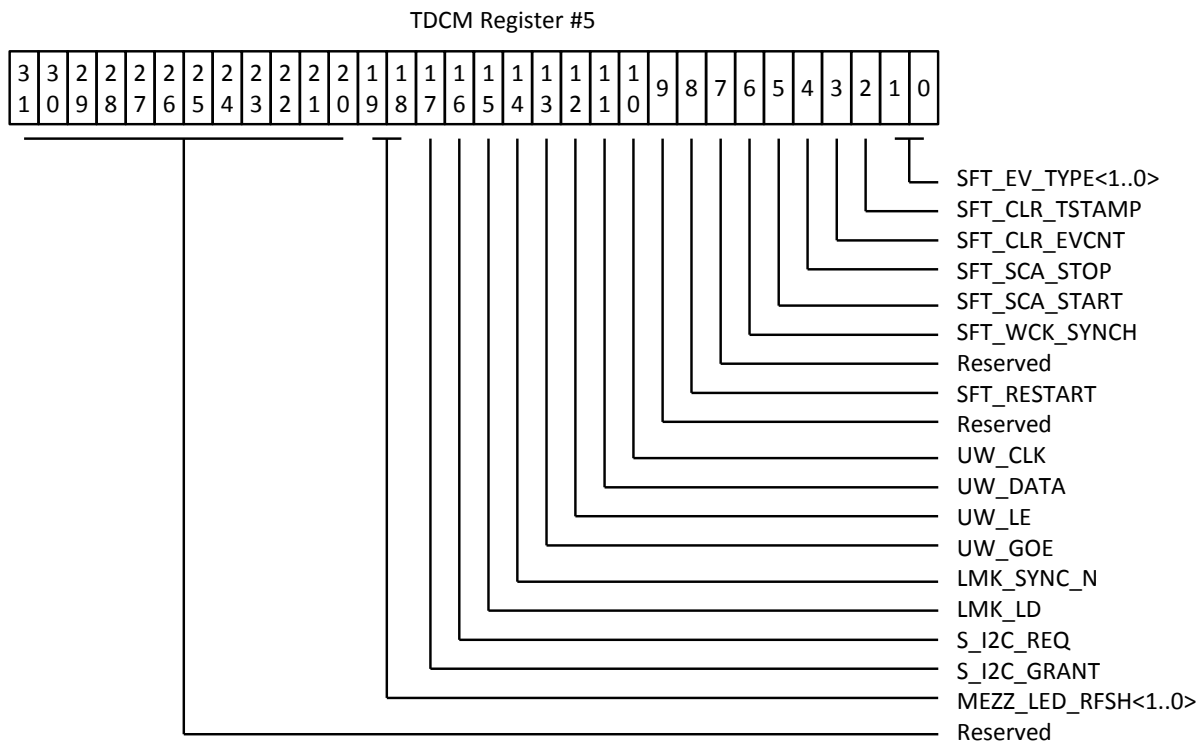
TDCM Register #5

Fig. 33.   TDCM General Configuration Register #5.

The field SFT_EV_TYPE, SFT_CLR_TSTAMP, SFT_CLR_EVCNT, SFT_SCA_STOP, SFT_SCA_START and SFT_WCK_SYNCH are used to transmit a software made trigger or command to all FEs using Virtual Channel A. Although it cannot be predicted precisely when this software made command will be transmitted, it reaches all FEs synchronously.

The bit SFT_RESTART is used to restart the operation of the trigger generator state machine.

The bit UW_CLK, UW_DATA, UW_LE, UW_GOE are used to implement a micro-wire serial interface. This port is only used to configure boards equipped with a PLL of the LMK032xx family. The LMK_SYNC_N is used to synchronize this PLL if it is needed. The bit LMK_LD is a read-only bit indicating the status of the PLL Lock Detect pin.

The bit S_I2C_REQ is used by the local processor to gain access to the master I2C controller of the TDCM. When access is granted to the processor, the line S_I2C_GRANT is activated by the underlying firmware. Operations initiated by the processor on the I2C bus must always request and obtain access to that bus before attempting to read or write some I2C slave device. The S_I2C_REQ line must remain active during the complete transaction and must be released afterwards.

The field MEZZ_LED_RFSH is used to enable (1) or disable (0) the automatic refresh of the front panel LEDs of the physical layer mezzanine boards of the TDCM. One configuration bit is available for each of the two mezzanine cards. When refresh is

enabled, the firmware of the TDCM will continuously detect the lock state of each FE link and the activity on that link to update the front panel LEDs accordingly.

General Configuration (Register #6):

This register is shown in Fig. 34.
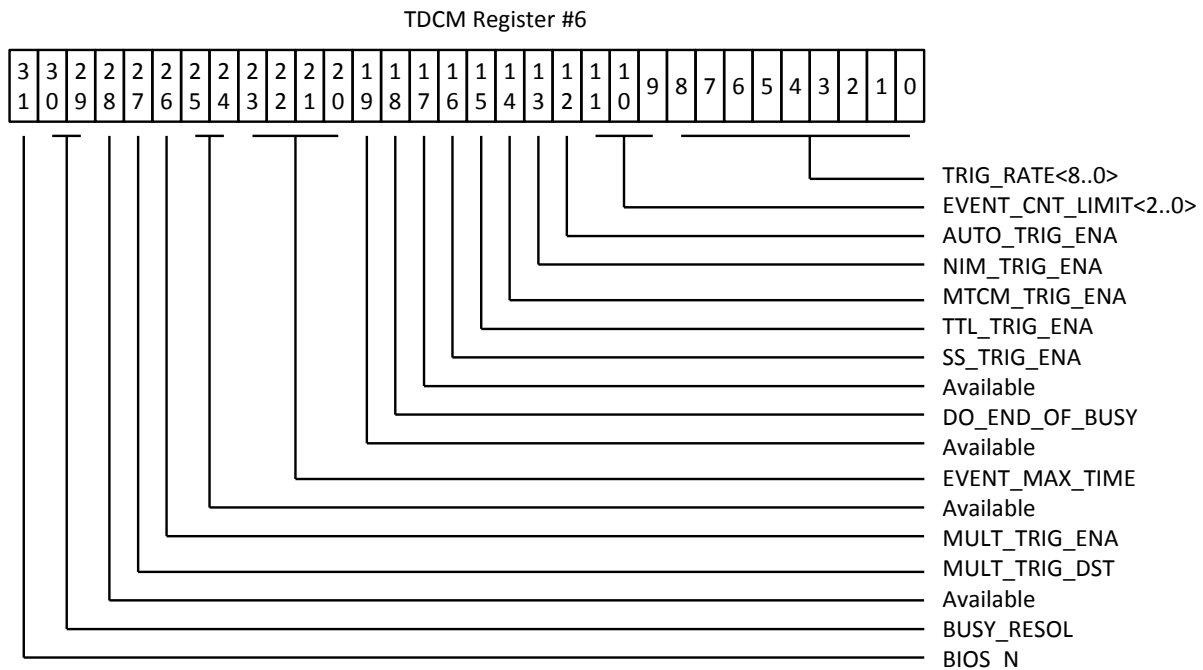
Fig. 34.   TDCM General Configuration Register #6.

The field TRIG_RATE determines the rate of the periodic trigger generator embedded in the TDCM. The two upper bits determine the rate range and resolution, while the seven lower bits determine the relative rate from 0 to 100. This is shown in Table 15.

Table 15 . Trigger generator range and resolution.

TRIG_RATE<8..7>	Range	Resolution
00	0 to 10 Hz	0.1 Hz
01	10 to 1000 Hz	10 Hz
10	100 Hz to 10 kHz	100 Hz
11	1 kHz to 100 kHz	1 kHz

The field EVENT_CNT_LIMIT determines the maximum number of event that can be generated by the local trigger generator or some external trigger input. Settings and the corresponding event count are given in Table 16. When the allowable event limit is reached, the local trigger state machine will stop generating new events or accepting external triggers until it is restarted. Limiting the number of event to a precisely defined value can be useful for system test and debugging.

Table 16 . Event count limit.

EVENT_CNT_LIMIT<2..0>	Event limit
000	infinite
001	1
010	10
011	100
100	1000
101	10,000
110	100,000
111	1,000,000

The bit AUTO_TRIG_ENA is used to enable (1) or disable (0) the local periodic trigger generator. The bit NIM_TRIG_ENA, MTCM_TRIG_ENA and TTL_TRIG_ENA are used to enable (1) or disable (0) the corresponding source of external triggers. Several trigger sources may be enables simultaneously. The TDCM supports two bits for defining the type of trigger. Usage depends on actual firmware implementation.

The bit SS_TRIG_ENA is used to enable or disable the "Single Shot Trigger" of the TDCM. When this mode is enabled, the TDCM generates automatically a single shot trigger with controlled latency from SCA_START. The total latency is the sum of the delay programmed in SS_TRIG_LAT and TRIG_LAT_2. The Single Shot Trigger mode is intended to be used when the calibration pulser is enabled on the front-end side.

The bit DO_END_OF_BUSY determines the behavior of the external BUSY pin. When DO_END_OF_BUSY is 0, the BUSY pin remains in the active state as long as the TDCM and the FEs are not available to take the next trigger. When DO_END_OF_BUSY is set to 1, the BUSY pin will only be activated for a short period when the TDCM and the FEs are ready to take the next trigger.

The EVENT_MAX_TIME field determines the maximum amount of time that is allowed for the readout of an event from the FEs. It can be programmed from 1 s to 17 s. Following a trigger, if a FE has not returned to the TDCM a SET_CLEAR_BUSY message over Virtual Channel A within the allowable event read-out time, the trigger generator state machine will reach an error state.

The MULT_TRIG_ENA bit is used to enable (1) or disable (0) the self-trigger based on the multiplicity bits sent by the FEs. The bit MULT_TRIG_DST determines the action to perform when a multiplicity trigger occurs. When MULT_DST_TRIG is set to 0, self-triggers based on multiplicity are transmitted to all FEs. When MULT_TRIG_DST is set to 1, the multiplicity trigger is sent to the M-TCM or some other external master trigger module.

The field BUSY_RESOL determines the resolution and range of the system dead-time histogram accumulated by the TDCM as shown in Table 17.

Table 17 . Dead-time histogram range and resolution.

BUSY_RESOL<1..0>	Range	Resolution
00	0 to 1.023 ms	1 µs
01	0 to 10.23 ms	10 µs
10	0 to 102.3 ms	100 µs
11	0 to 1023 ms	1 ms

The bit BIOS_N is a read-only bit that indicates when low that the external BIOS button has been pushed. The state of this button is scanned by the TDCM at startup to enter the "minibios" program that is used to configure various parameters of the TDCM that need to persist after reboot. See the relevant section for details.

Event Data Generator Configuration (Register #7):

This register determines the configuration and behavior of the event data generator embedded in the TDCM. This register is shown in Fig. 35.
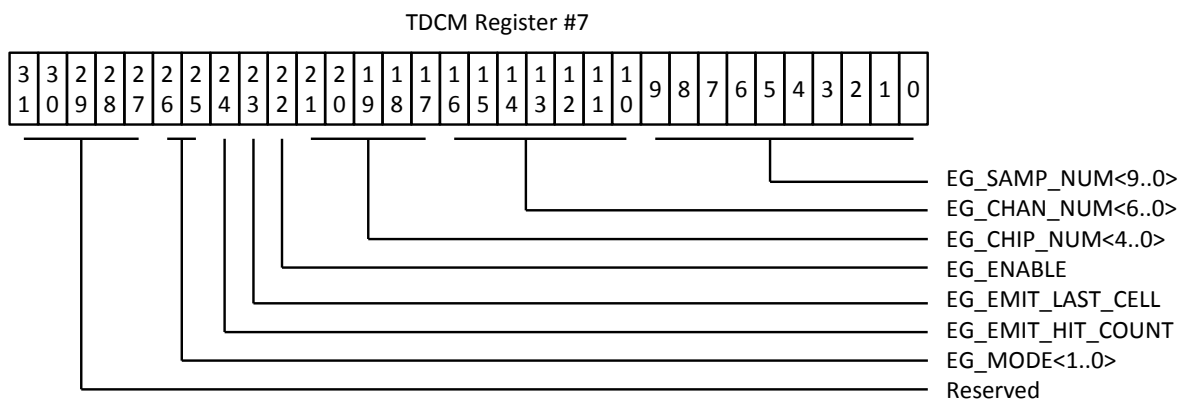
Fig. 35.   Event Data Generator Configuration Register #7.

The field EG_SAMP_NUM determines the number of samples contained in every channel data packet that are generated. This number can be set from 0 to 512 inclusive. When EG_SAMP_NUM is set to an odd value, two null bytes are added after the actual samples to guarantee that payload size is a multiple of four bytes. The field EG_CHAN_NUM and EG_CHIP_NUM determine the equivalent number of channels per chip and the number of chips per FE respectively for the data generator. The field EG_CHIP_NUM can be set from 0 to 16 inclusive, while the EG_CHAN_NUM can be set from 0 to 79 inclusive. When full event are produced, the generator makes events that comprise EG_CHIP_NUM x EG_CHAN_NUM packets of data per simulated FE. Each data

packet contains EG_SAMP_NUM data samples coded on two bytes each, and two null bytes when EG_SAMP_NUM is odd.

The bit EG_ENABLE is used to enable (1) or disable (0) the local event data generator. The generator may not be used when physical FEs are present and activated.

The bit EG_EMIT_LAST_CELL is used to optionally emit the last cell indicator of the AFTER/AGET chip in the event header packet. The bit EG_EMIT_HIT_COUNT is used to optionally emit the AFTER/AGET channel hit count in the event header packet. When these settings are enabled, constant values are put at the corresponding locations in the event header packets. The field EG_MODE determines the operating mode of the event data generator. The event generator can produce events of constant or variable size. The size of each data packet can be made constant for all channels, or it can also vary from one channel to the next within the same event. However, the packets generated always have the same size across all simulated FEs. Possible settings for EG_MODE are listed in Table 18.

Table 18 . Event Data Generator operating mode.

EG_MODE<1..0>	Event Generated
X0	Constant size
01	Fragment of random size but constant within one event
11	Fragment of a different random size from one channel to the next within every event

When the event data generator is enabled, all the possible sources of trigger, provided they are enabled, can trigger the generation of one event. Note that if the FIFO buffer of any FE fills-up, the event generator skips received triggers until memory space becomes available.

Trigger Latency (Register #8 and #9):

These registers determines the additional latency applied to trigger signals for up to four types of triggers. The value programmed in each field is expressed in clock units of 10 ns. The added latency for each type of trigger is programmable from 0 to 655.350 µs. These registers are shown in Fig. 36 and Fig. 37. The values TRIG_LAT_0, TRIG_LAT_1, TRIG_LAT_2 and TRIG_LAT_3 apply to trigger type 0, 1, 2 and 3 respectively.

TDCM Register #8

Fig. 36.   Trigger Latency Register #8.

TDCM Register #9

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0

TRIG_LAT_2[15..0]
TRIG_LAT_3[15..0]
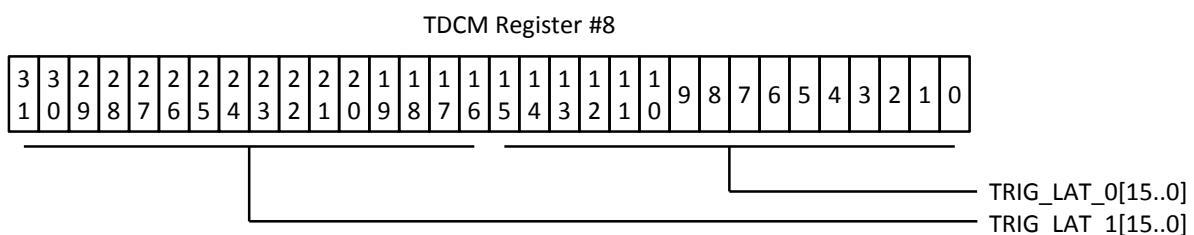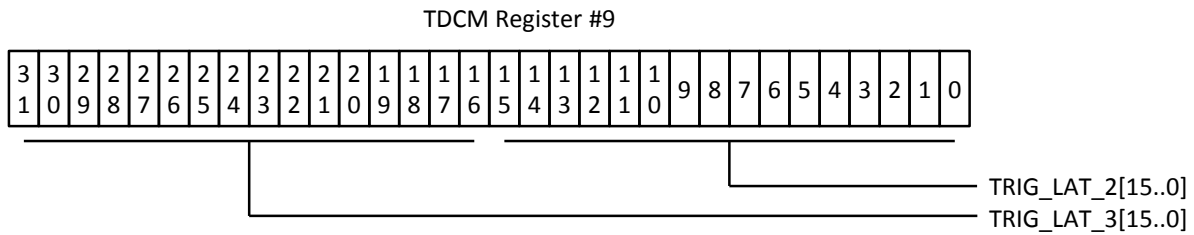
Fig. 37.   Trigger Latency Register #9.

Front-end active Register (Register #10):

This register determines which front-ends are active and which are either not present or disabled. This register is shown in Fig. 38. Setting bits to 1 in this register indicates that the corresponding FEs are currently active.

TDCM Register #10

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0

FE_ACTIVE[31..0]
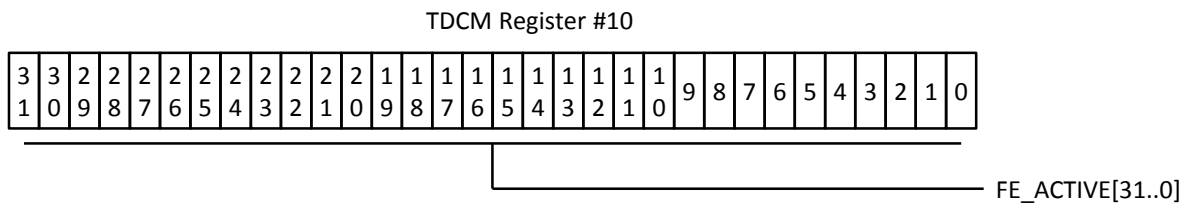
Fig. 38.   Front-End active Register #10.

When a FE is not active (i.e. masked) at the TDCM level, all messages will still be sent to the associated port because a hardwired fanout to all FEs is used. However, any incoming message coming from a masked port will be ignored.

Trigger Multiplicity Register (Register #11):

This register determines the conditions for the generation of a self-trigger based on the multiplicity signals received from the front-ends. This register is shown in Fig. 39.

TDCM Register #11

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0

MULT_MORE_THAN[6..0]
MULT_LESS_THAN[13..7]
Available
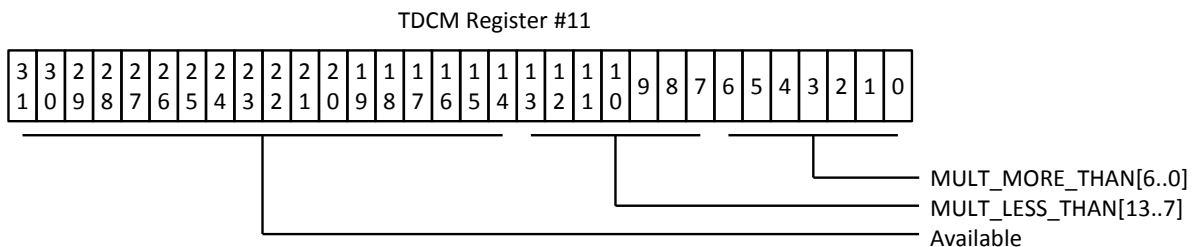
Fig. 39.   Trigger Multiplicity Register #11.

When the multiplicity trigger is enabled at both ends, the TDCM continuously receives from each FE 4 bits indicating if the multiplicity of the corresponding AGET chip is above or below threshold. The TDCM makes the arithmetic sum of all these bits and applies the above programmable thresholds to determine when a self-trigger occurs.

Register #12:

This read-write registers is currently unassigned and is available for future use.

M-TCM interface (Register #13):

This register is used to control various functions of the M-TCM interface. It is shown in Fig. 40.

TDCM Register #13
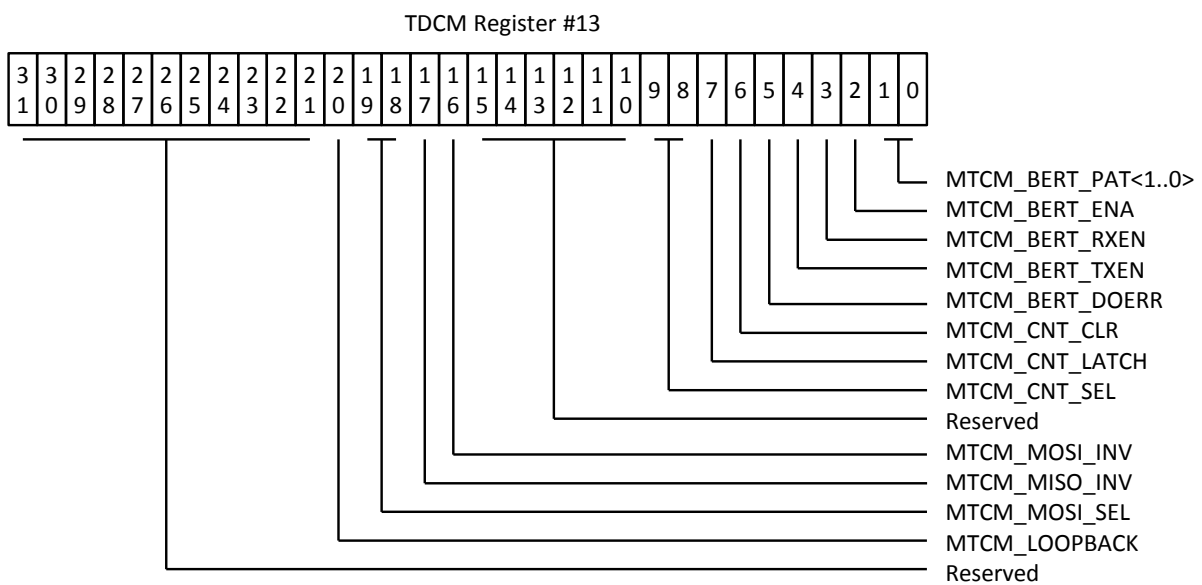
Fig. 40.   M-TCM interface register.

The field MTCM_BERT_PAT is used to select which pseudo-random bit pattern is expected and sent when the bit error rate tester is engaged. Possible values: "00": PRBS7; "01": PRBS15; "10": PRBS23; "11": PRBS31.

The bit MTCM_BERT_ENA is used to enable or disable the bit error rate tester function between the TDCM and the M-TCM. When MTCM_BERT_ENA is set to 1, the bit stream received from the M-TCM is not forwarded to the trigger logic, but is interpreted by the bit error rate tester. The bit MTCM_BERT_ENA shall return to 0 only after the M-TCM has stopped sending the PRBS pattern and the communication link has returned to its normal operation mode.

The bit MTCM_BERT_RXEN is used to effectively start the receive part of the bit error rate tester after MTCM_BERT_ENA has been set to 1. When MTCM_BERT_RXEN is set to 1, the internal logic tries to capture from the received bits a particular seed value that

62

occurs only once in the selected PRBS pattern. If the seed pattern can be found, the local pseudorandom pattern generator is started and received bits are compared against the expected ones. When MTCM_BERT_RXEN is cleared, received bits are no longer compared with the expected pattern. MTCM_BERT_RXEN may be set to 1 again to continue the test.

The bit MTCM_BERT_TXEN is used to enable the transmitter part of the bit error rate tester after MTCM_BERT_ENA has been set. When MTCM_BERT_TXEN is set to 1, the TDCM starts transmitting the selected PRBS to the M-TCM. Transmission continues until MTCM_BERT_TXEN is cleared. When a transition from 0 to 1 is detected on MTCM_BERT_DOERR, the transmitter sends the complement value of the current bit of the PRBS to generate a single bit error. The MTCM must be able to detect this error.

Pulsing the bit MTCM_CNT_CLR from 0 to 1 clears the message counter selected by MTCM_CNT_SEL. After a counter has been cleared, MTCM_CNT_CLR shall be cleared.

Pulsing the bit MTCM_CNT_LATCH from 1 to 0 captures the content of the counter selected by MTCM_CNT_SEL. The value of the captured counter is readable on Register #30. It is recommended to make a transition, 0, 1, 0 on MTCM_CNT_LATCH to capture each counter.

The field MTCM_CNT_SEL is used to select which message counter is cleared or captured. The possible values are: "00": RX count from M-TCM; "01": RX error count from M-TCM; "10": TX count to TCM; "11": reserved.

The bit MTCM_MOSI_INV is used to optionally invert the serial data received from the M-TCM. This setting applies in the normal mode of operation of the link and in bit error rate tester mode.

The bit MTCM_MISO_INV is used to optionally invert the serial data sent to the M-TCM. This setting applies in the normal mode of operation of the link and in bit error rate tester mode.

The bit MTCM_MOSI_SEL is used to select which of the 4 samples captured from the MTCM_MOSI line are de-serialized. This setting can be used to compensate for some static phase offset between the clock received from the M-TCM and the serial data.

The bit MTCM_LOOPBACK is used to set the M-TCM interface of the S-TDCM in loopback mode. In this mode, when received a message "SCA_START", the S-TDCM returns "START_ACK" with the shortest possible delay. And when received a message "SCA_STOP", the S-TDCM returns a message "SET_BUSY" followed a few clock cycles later by "CLR_BUSY". The loopback mode is intended to test the normal M-TCM to S-TDCM protocol without forwarding any of the traffic to the front-end side.

Register #14:

This register is shown in Fig. 41.
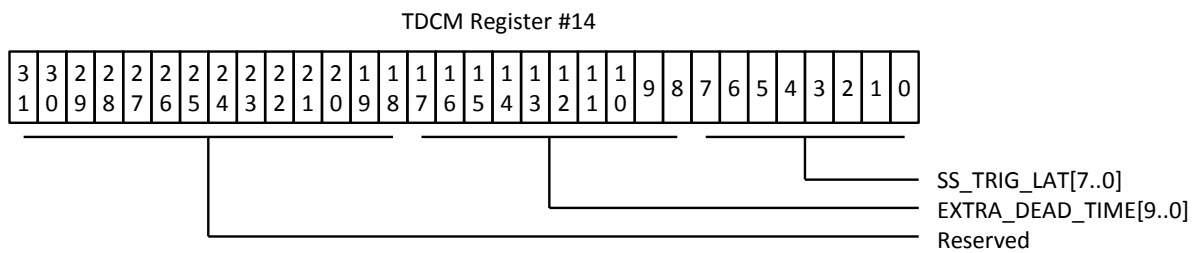
TDCM Register #14

Fig. 41.   TDCM General Configuration Register #14.

The field SS_TRIG_LAT is used to program the latency of the generated trigger in the "Single Shot Trigger" mode. The value is expressed in units of 10 µs. The programmed value must be at least 1, leading to a minimum 10 µs delay between SCA_START and the generated trigger. The maximum delay is 25.5 ms. Note that in addition to the delay set by SS_TRIG_LAT, the value programmed in TRIG_LAT_2 is also applied. The Single Shot trigger mode should be used when the calibration pulser is enabled on the front-end side. The latency of the pulser needs to be set so that injection occurs within the data capture window. It is not recommended to set the total latency of the single shot trigger to less than one complete turn in the SCA matrix of the front-end chip because previous data and the transient signal caused by the activation of SCA_WRITE should be completely erased before a new event is captured.

The field EXTRA_DEAD_TIME is used to program additional dead-time after sampling in all FE has been restarted and before the TDCM releases its external BUSY pin, and it posts CLEAR_BUSY to the M-TCM (when it is used). The additional dead-time is expressed in µs and it can be set from 0 to 1023 µs.

Register #15:

This read-write register is currently unassigned and is available for future use.

Front-End Link OK Register (Register #16):

This register indicates to which FEs communication is currently established. Every bit in this register correspond to one physical port on the TDCM. When a bit is 0 in this register, it indicates that no FE appears to be present or communication with that FE could not be established or was lost. This register is shown in Fig. 42.

64

TDCM Register #16

FE_LINK_OK[31..0]
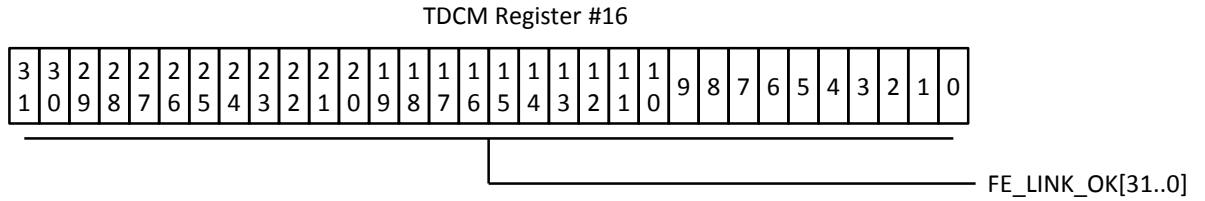
Fig. 42.   Front-End Link OK Register #16.

Front-End is Sampling Register (Register #17):

This register indicates which FEs are currently in the "Sampling" state. Every bit corresponds to one port on the TDCM. This register is shown in Fig. 43.

TDCM Register #17

FE_IS_SAMPLING[31..0]
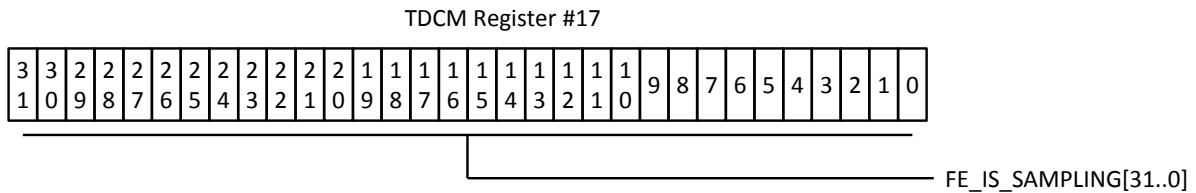
Fig. 43.   Front-End Sampling Register #17.

The "Sampling" state is when front-end ASICs are writing in the SCA the analog signals from the associated detector.

Front-End is Busy Register (Register #18):

This register indicates which FEs are currently in the "Busy" state. Every bit corresponds to one port on the TDCM. This register is shown in Fig. 44.

TDCM Register #18

FE_IS_BUSY[31..0]
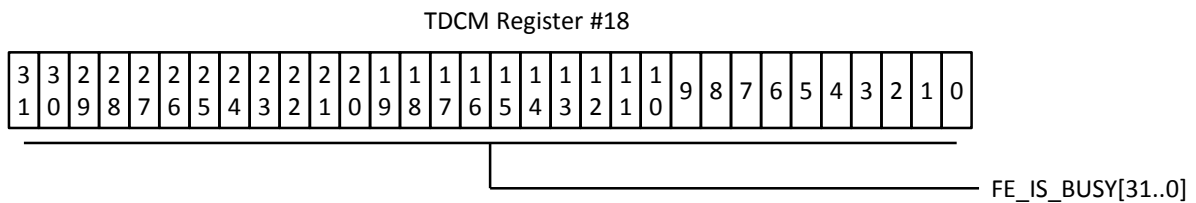
Fig. 44.   Front-End is Busy Register #18.

Following a trigger, when a FE is in the "Sampling" state it changes it state to "Busy". The "Busy" state includes the time needed for the digitization on the SCA and all the additional time until writing in the SCA resumes, i.e. the FEC returns to the "Sampling" state.

Bit Error Rate Tester – Receive Mbit Count (Register #19):

This register contains the number of Mbit receive from the front-end side in bit error rate tester mode. This register is shown in Fig. 45.

TDCM Register #19

BERT_RX_MBIT[31..0]
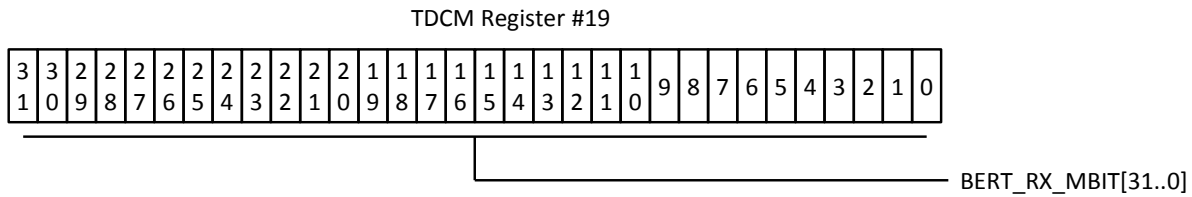
Fig. 45.   Bit Error Rate Tester – Received Mbit (Register #19).

The content of this register starts incrementing in bit error rate tester mode as soon as BERT_START is detected. The nominal link rate from the FE to the TDCM is 400 Mbit/s. Consequently, this counter is incremented by one unit every 2.5 ms. It will roll-over after $10^6 \times 2^{32}$ bit are received, i.e. ~4 months.

Bit Error Rate Tester – Receive State and Error Counter (Register #20)

This register contains the number of Mbit receive from the front-end side in bit error rate tester mode. This register is shown in Fig. 46.

TDCM Register #20

RX_BERT_ERR_CNT[7..0]
RX_BERT_DELAY[15..8]
RX_BERT_LOCKED
Available
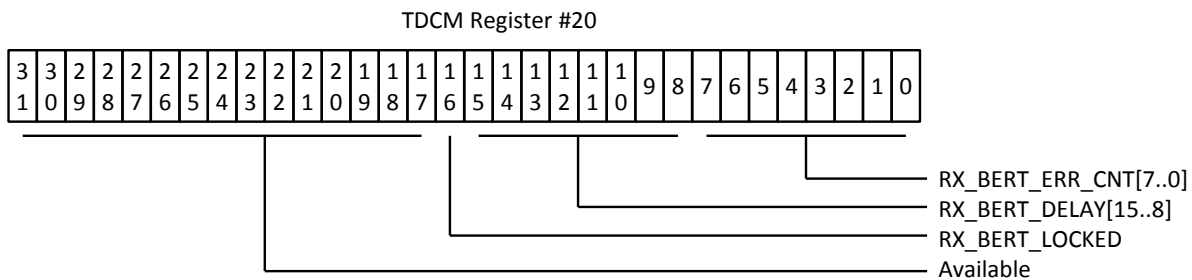
Fig. 46.   Bit Error Rate Tester – Receive State and Error Counter (Register #20).

The field RX_BERT_ERR_CNT contains the number of bit errors detected from the selected FEC when in bit error rate tester mode. The field RX_BERT_DELAY contains the delay applied on the pseudo-random received bit stream to align it to the locally generated pattern. This is expressed in UI (i.e. 2.5 ns at the nominal rate of 400 Mbit/s). The bit RX_BERT_LOCKED indicates whether the bit error rate tester has achieved lock or not for the selected FE. Other fields in this register are invalid unless lock was achieved.

Received Messages Statistics Counters (Register #21)

This register contains the receive statistics counters for one selected FE and one of the three Virtual Channels. This register is shown in Fig. 47.

66

TDCM Register #21

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0

RX_STAT_CNT[31..0]
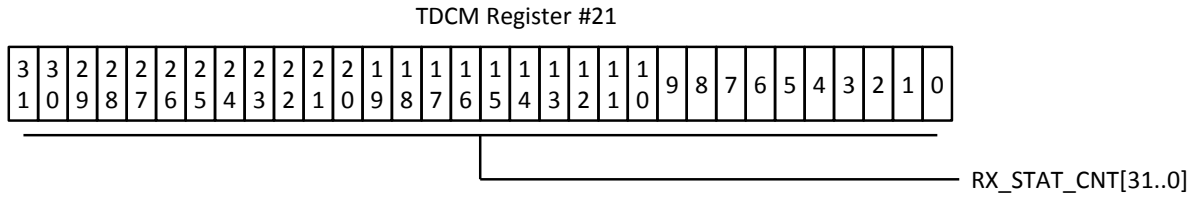
Fig. 47.   Received Messages Statistics Counters (Register #21).

At first, the statistics counters of one Virtual Channel must be latched. This is accomplished by setting STAT_CNT_ABC as desired and making a transition from 0 to 1 on STAT_CNT_LATCH. After this is done, the field RX_STAT_CNT is updated to the statistics of the FE selected by PORT_SEL. The field and STAT_CNT_SEL determines which of the received message counter or received message error counters are readout.

Transmit Message Counter (Register #22)

This register contains the counter of messages transmitted to the front-end. This register is shown in Fig. 48.

TDCM Register #22

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
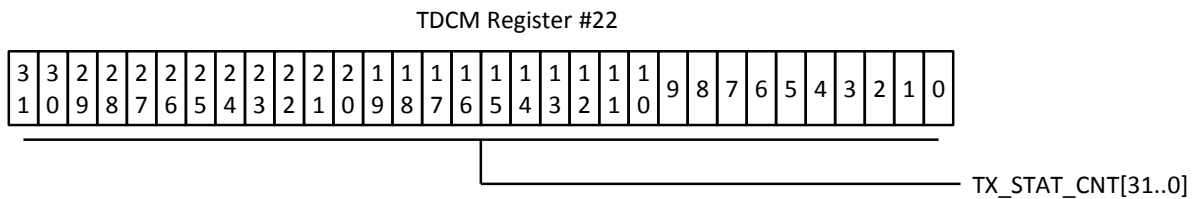
TX_STAT_CNT[31..0]

Fig. 48.   Transmit Message Counter (Register #22).

The field TX_STAT_CNT contains the number of messages sent over Virtual Channel A, B or C according to STAT_CNT_ABC after the corresponding counter has been latched by a transition from 0 to 1 on STAT_CNT_LATCH.

Data Pump Running (Register #23)

This register indicates which of the Data Pumps are in the "Running" state. This register is shown in Fig. 49.

TDCM Register #23

3 1	3 0	2 9	2 8	2 7	2 6	2 5	2 4	2 3	2 2	2 1	2 0	1 9	1 8	1 7	1 6	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
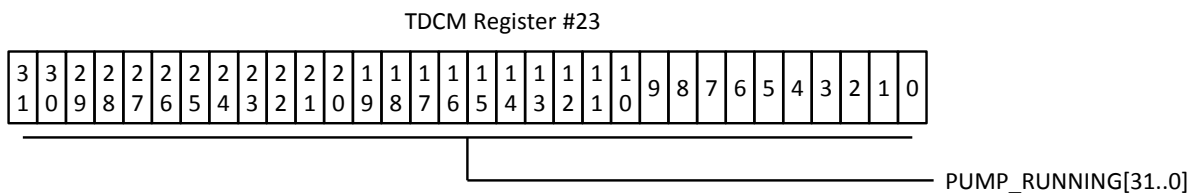
PUMP_RUNNING[31..0]

Fig. 49.   Data Pump Running (Register #23).

Every bit in this register corresponds to one physical port on the TDCM. Each Data Pump Running bit indicates that the TDCM is currently trying to get event data from the corresponding FE. This is the normal expected state during data taking.

Data Pump Stalled (Register #24)

This register indicates which of the Data Pumps are in the "Stalled" state. This register is shown in Fig. 50.

TDCM Register #24

| 3 1 | 3 0 | 2 9 | 2 8 | 2 7 | 2 6 | 2 5 | 2 4 | 2 3 | 2 2 | 2 1 | 2 0 | 1 9 | 1 8 | 1 7 | 1 6 | 1 5 | 1 4 | 1 3 | 1 2 | 1 1 | 1 0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
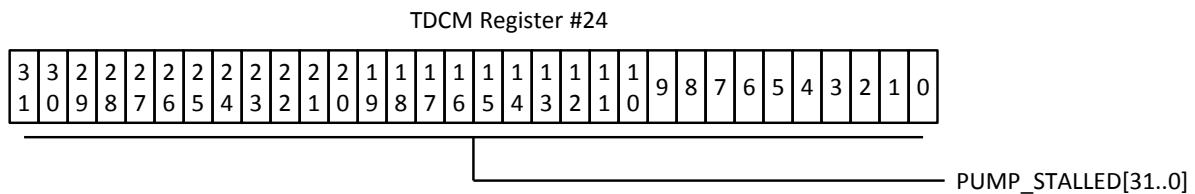
PUMP_STALLED[31..0]

Fig. 50.   Data Pump Stalled (Register #24).

Every bit in this register corresponds to one physical port on the TDCM. Each Data Pump Stalled bit indicates that the TDCM failed to get event data from the corresponding FE. This is an error situation during data taking that indicates that the corresponding FE is not responsive or did not return the expected empty data packets within the allowable timeout period. This error must be resolved before normal data acquisition can resume.

Event Builder – Start Of Event expected (Register #25)

This register indicates the set of FEs from which a Start Of Event packet is expected for the event currently being read out. This register is shown in Fig. 51.

TDCM Register #25

| 3 1 | 3 0 | 2 9 | 2 8 | 2 7 | 2 6 | 2 5 | 2 4 | 2 3 | 2 2 | 2 1 | 2 0 | 1 9 | 1 8 | 1 7 | 1 6 | 1 5 | 1 4 | 1 3 | 1 2 | 1 1 | 1 0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
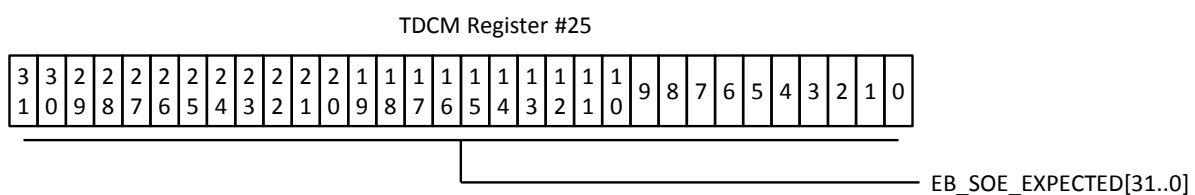
EB_SOE_EXPECTED[31..0]

Fig. 51.   Event Builder – Start Of Event expected (Register #25).

Every bit in this register corresponds to one physical port on the TDCM. Each EB_SOE_EXPECTED bit indicates that the TDCM still expects the Start Of Event packet from the corresponding FE to start the assembly of the current event. When the local Event Builder is blocked because of an error, reading this register can help localize the faulty FE or communication port.

Event Builder – End Of Event expected (Register #26)

This register indicates the set of FEs from which an End Of Event packet is expected for the event currently being read out. This register is shown in Fig. 52.

TDCM Register #26

| 3 1 | 3 0 | 2 9 | 2 8 | 2 7 | 2 6 | 2 5 | 2 4 | 2 3 | 2 2 | 2 1 | 2 0 | 1 9 | 1 8 | 1 7 | 1 6 | 1 5 | 1 4 | 1 3 | 1 2 | 1 1 | 1 0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
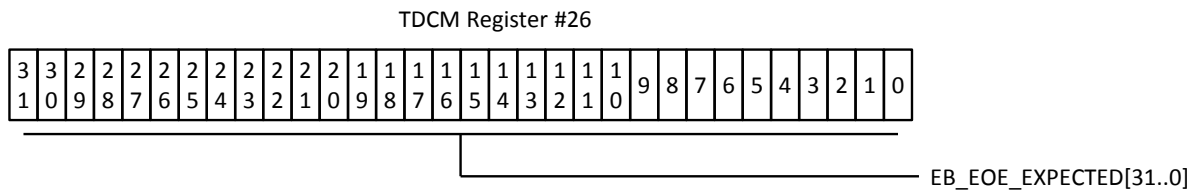
EB_EOE_EXPECTED[31..0]

Fig. 52.   Event Builder – End Of Event expected (Register #26).

Every bit in this register corresponds to one physical port on the TDCM. Each EB_EOE_EXPECTED bit indicates that the TDCM still expects the End Of Event packet from the corresponding FE to finish the assembly of the current event. When the local Event Builder is blocked because of an error, reading this register can help localize the faulty FE or communication port.

General Status Register (Register #27)

This register indicates the internal state of various components of the firmware of the TDCM. This register is shown in Fig. 53.

The bit EB_COLLECTING_SOE indicates that the local event builder is collecting Start Of Event packets for the current event.

The bit EB_WAIT_FE_PKT indicates that the local event builder is waiting for a packet from the front-end indicated by the field EB_CURRENT_FE.

The status bit EB_MISSING_SOE indicates that the local event builder is in error because it has not received the expected Start Of Event packet. The status bit EB_MISSING_EOE indicates that one or several End Of Event packets are missing.

The status bit EB_EV_NB_MISMATCH indicates that the event builder detected that at least one event number among all the Start Of Event packets for the current event do not match. The status bit EB_EV_TS_MISMATCH indicates a timestamp mismatch that exceeds the allowable limit.

The status bit PM_WAIT_GET_NXT_PKT indicates that the Packet Mover is currently waiting before it can get the next data packet. The status bit PM_WAIT_PKT_FIFO_NE indicates that the Packet Mover is currently waiting for a packet FIFO to be non-empty. The status bit PM_WAIT_BUF_DESC indicates that the Packet Mover is currently waiting for a buffer descriptor in order to start moving a received packet. The status bit PM_WAIT_PKT_MOVED indicates that the Packet Mover is currently waiting for the end of a message transfer from a Receive FIFO to some other memory location.
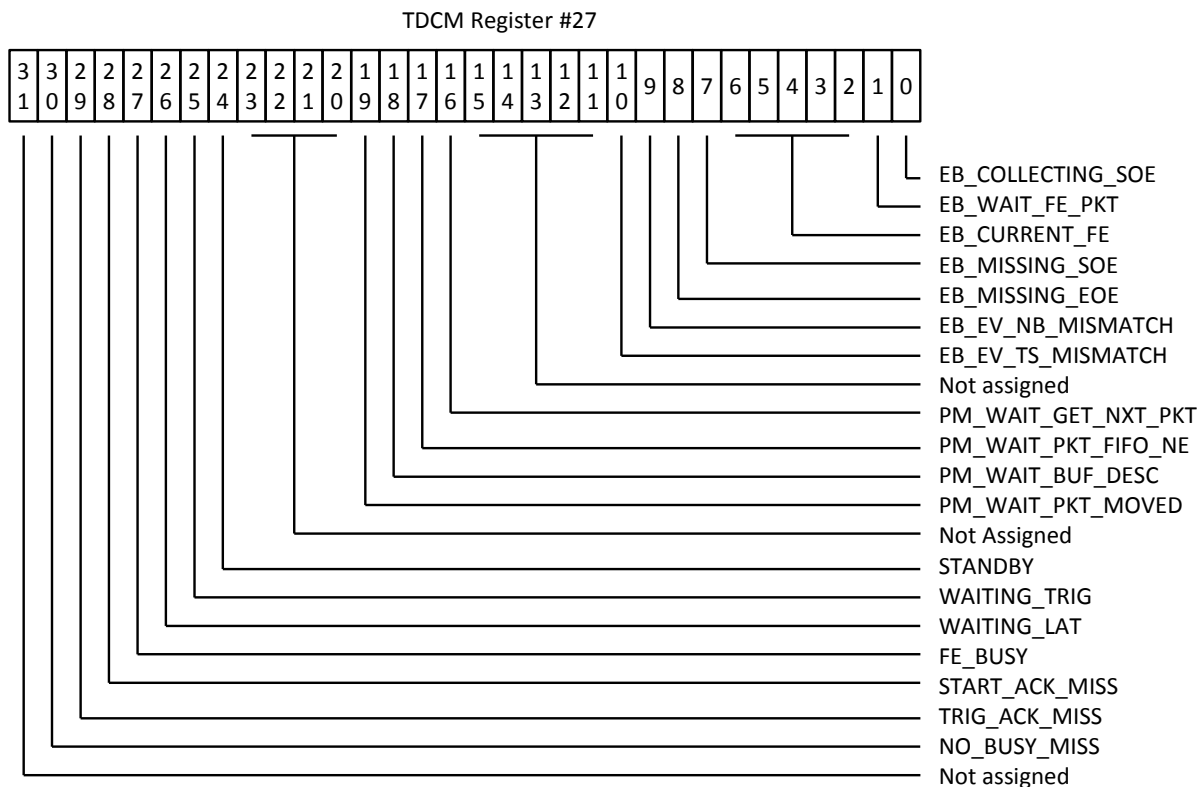
TDCM Register #27

Fig. 53.   General Status Register (Register #27).

The status bit STANDBY indicates that the Trigger Handler is currently in some idle state. The status bit WAITING_TRIG indicates that the Trigger Handler is currently waiting for a trigger. This state is reached when all the front-ends are in the Sampling state. The status bit WAITING_LAT indicates that the Trigger Handler has received a valid trigger and is currently delaying this trigger by the programmed latency. This state is transient and may not be observed. The status bit FE_BUSY indicates that one or several FEs are currently busy. The status bit START_ACK_MISS indicates that one or several FEs have not returned to the TDCM the expected acknowledge message following SCA_START. The status bit TRIG_ACK_MISS indicates that one or several FEs have nor returned to the TDCM the expected acknowledge message following SCA_STOP. The status bit NO_BUSY_MISS indicates that one or several FEs have not sent a message with the CLEAR_BUSY flag set within the allowable maximum event read out time.

Event Received Count (Register #28)

This register counts the number of valid trigger received by the TDCM. This register is shown in Fig. 54.

70

TDCM Register #28

EVENT_RX_CNT[31..0]
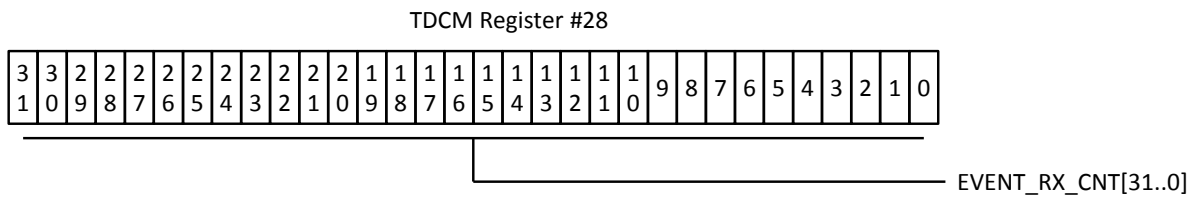
Fig. 54.   Event Received Count (Register #28).

This counter is incremented by one unit for each trigger, internally generated or externally received.

Event Transmit Count (Register #29)

This register counts the number of valid trigger transmitted by the TDCM to the front-end side. This register is shown in Fig. 55.

TDCM Register #29

EVENT_TX_CNT[31..0]
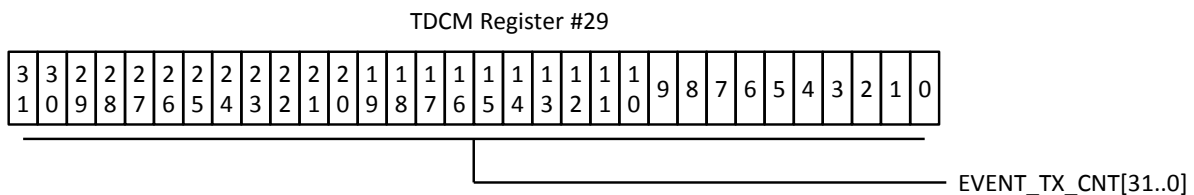
Fig. 55.   Event Transmit Count (Register #29).

This counter is incremented by one unit for each trigger transmitted to the FEs. It cannot exceed the number of events received.

Register #30

This register is used to display the status of the interface to the M-TCM. It is a multi-format register as shown in Fig. 56. The 24 LSBs are updated when MTCM_CNT_LATCH is pulsed, according to which counter is selected by MTCM_CNT_SEL. The TX and RX counters count the number of "items" being sent or received. An item means one message when the link between the S-TDCM and M-TCM operates in the normal mode. Each item means 10^6 bits when the bit error rate tester mode is engaged.

When the RX error counter is selected, the 8 LSB's contain the number of parity errors detected in received messages when the link runs in normal mode, and the number of bit errors when running in bit error rate tester mode. The bit RX_BERT_RUNNING indicates that lock onto the received PRBS is achieved. In normal mode, this bit is unused.

The 8 MSB's of this register have a fixed signification. The bit MTCM_DETECTED is set to 1 after a stable high level is detected on the cable signal MTCM_ENAREM and the DIP switch "MTCM" is manually set to the ON position on the S-TDCM.
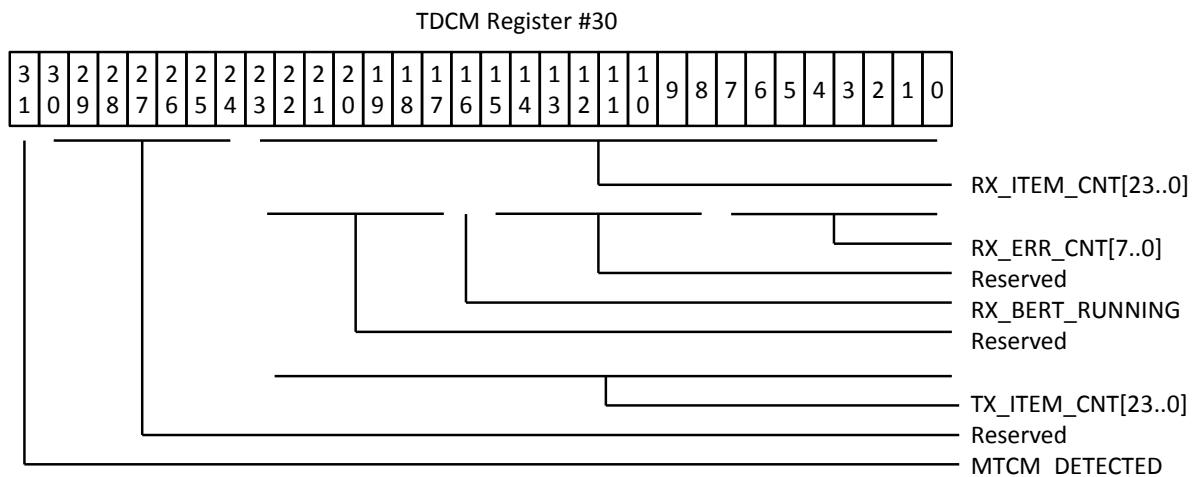
Fig. 56.　　M-TDCM Status Register (Register #30)

Register #31

This register is currently not assigned and is reserved for future use.

7.2 RING BUFFER INTERFACE

Data transfers from the receivers of the links connected to front-end electronics to the scatter-gather DMA Ethernet controller embedded in the FPGA of the TDCM are handled by the "Ring Buffer Interface" where descriptors pointing to buffers located in the SDRAM attached to the SoC are exchanged between the local processor and the local FPGA logic. At system initialization, a pool of buffers is allocated in the SDRAM of the SoC to contain event data. Each buffer is coupled to a buffer descriptor that contains a pointer to it. Free buffer descriptors are posted by the local processor to the programmable logic via a FIFO, called the out-bound FIFO – OFIFO in short, accessible through the AXI-4 bus. Free buffer descriptors are consumed by the FPGA logic as they are needed to store the event data received from the front-end link receivers. When a buffer does not have sufficient remaining space to store the next block of event data, or when no data has been received after a programmable time out value has elapsed, its associated buffer descriptor is posted by the FPGA logic to the local processor via a FIFO, called the in-bound FIFO, IFIFO in short. The local processor is responsible for unloading buffer descriptors from the IFIFO, pre-pend header information (e.g. Ethernet, IP and UDP headers) to the filled buffers and post them to the Ethernet controller for DMA transmission to the DAQ PC. When a buffer has been sent, the local processor returns its associated buffer descriptor to the OFIFO for re-use.

The resource map of the ring buffer interface is shown in Table 19.

Table 19. Ring Buffer Interface Resource Map.

Address range	Access	Function
RBF_Base+0x0	R/W	Ring Buffer Control/Configuration
RBF_Base+0x4	R	Logic to processor in-bound FIFO
RBF_Base+0x4	W	Processor to logic out-bound FIFO

The RBF_Base is the 32-bit address assigned for this IP block via the Address Editor panel of Xilinx tools at synthesis time. It cannot be changed afterwards. The corresponding address range must not be made cacheable. At system initialization, the local processor must program the Ring Buffer Configuration register first and then write to the out-bound FIFO the set of free buffer descriptors. The Ring Buffer Control/Configuration register is shown in Fig. 57.

Ring Buffer Control/Configuration Register
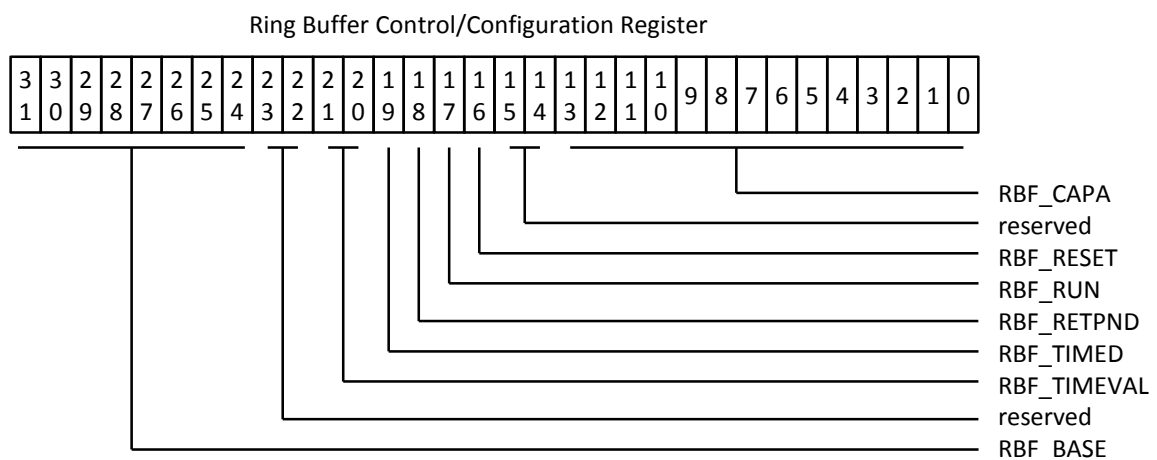
Fig. 57.   Ring Buffer Control/Configuration Register.

The field RBF_CAPA specifies (in bytes) the size of the buffers pointed by the buffer descriptors exchanged over the in-bound and out-bound FIFOs. All buffers must have an equal size and must be aligned to 8 KByte boundaries. The size must be a multiple of 4 bytes and must be greater than the Ethernet MTU. It is recommended to use 8 KByte buffers so that Ethernet Jumbo frames up to that size can be accommodated.

The RBF_RESET field should be set and cleared by the local processor at startup to initialize the programmable logic side of the interface properly.

The RBF_RUN bit must be set to 1 to start data transfers after configuration and after a sufficient number of free buffer descriptors have been posted to the hardware. This bit can be cleared to stop data transfers. The RBF_RETPND bit can be pulsed to force the local logic to abort filling the current buffer descriptor and post it to the processor in-bound FIFO.

The RBF_TIMED bit should be set to 1 to operate the ring buffer in a mode where partially filled buffers are returned to the IFIFO after a timeout period has elapsed. The

field RBF_TIMEVAL determines the time to wait before sending a partially filled buffer. The four possible timeout values are 1 ms, 10 ms, 100 ms and 1 s. When RBF_TIMED is cleared, the ring buffer logic will wait indefinitely for data to optimally fill buffers before sending them (unless RBF_RETPND is pulsed).

The RBF_BASE field gives the 8 MSB's of the 32-bit memory base address where the array of data buffers is placed. This base address must start on a 16-MByte boundary and normally points to a region in the SDRAM attached to the SoC. Assuming 511 buffers of 8 KByte are allocated, 4 MByte of memory storage have to be allocated.

The buffer descriptors exchanged over the in-bound and out-bound FIFOs are 32-bit words. Hence fetching or posting a descriptor only requires a single 32-bit access. The fields of a buffer descriptor are shown in Fig. 58.
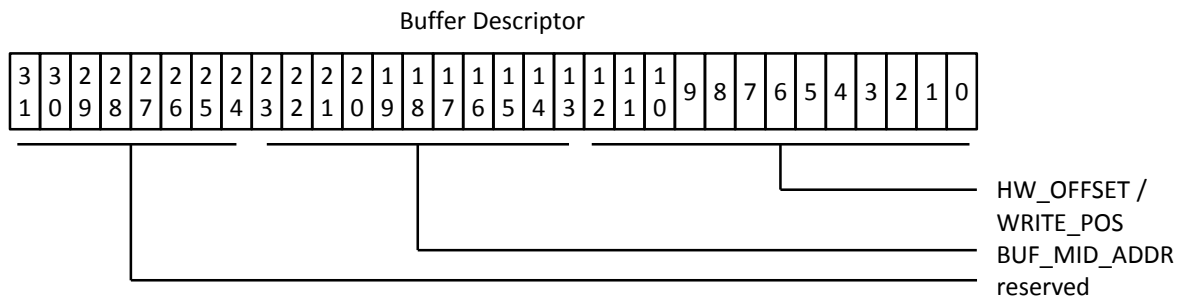
Fig. 58.   Buffer Descriptor.

The HW_OFFSET field is used in descriptors posted to the out-bound FIFO to indicate to the embedded logic the location of the first free location in the corresponding buffer. The embedded software should set a non-null value in this field to reserve the necessary space to put Ethernet and other protocol headers (i.e. Ethernet frame header, IP and UDP or TCP headers). Additional space can be reserved to pre-pend specific and user defined header information in software after the core of data buffer has been filled by the hardware. The value of HW_OFFSET must be multiple of 4. Assuming standard Ethernet and UDP/IP framing, the header information is actually only 42 bytes. This must be rounded up to 44 bytes and the first two bytes of the UDP payload must be set by the local processor in the present implementation.

When descriptors are read from the in-bound FIFO, the 13 LSB's of the buffer descriptor are the current position of the pointer where additional data may be written. Software may append trailing information starting from this offset. This field is also used to determine the size of data that has been filled in the buffer pointed by the descriptor (HW_OFFSET must be subtracted to get the actual size of data that was filled in the buffer by the programmable logic side).

The BUF_MID_ADDRESS is used in conjunction with the RBF_BASE address to determine the physical memory base address of the buffer pointed by this descriptor. The 32-bit base address of the buffer is formed by concatenating RBF_BASE, BUF_MID_ADDRESS, and twelve padding 0's. The resulting address is therefore aligned on 8 KByte boundaries (i.e. the maximum capacity of a buffer). The maximum number of buffer descriptors is 2048 corresponding to up to a 16-MByte buffer array. However, the in-bound and out-bound FIFOs have only 511 positions each and it is recommended to set the number of descriptors to 511 or less to avoid any overflow.

The relations between the Ring Buffer Configuration Register, a buffer descriptor and the actual buffer are shown schematically on Fig. 59 and Fig. 60 for an empty buffer and for a buffer filled with data respectively.
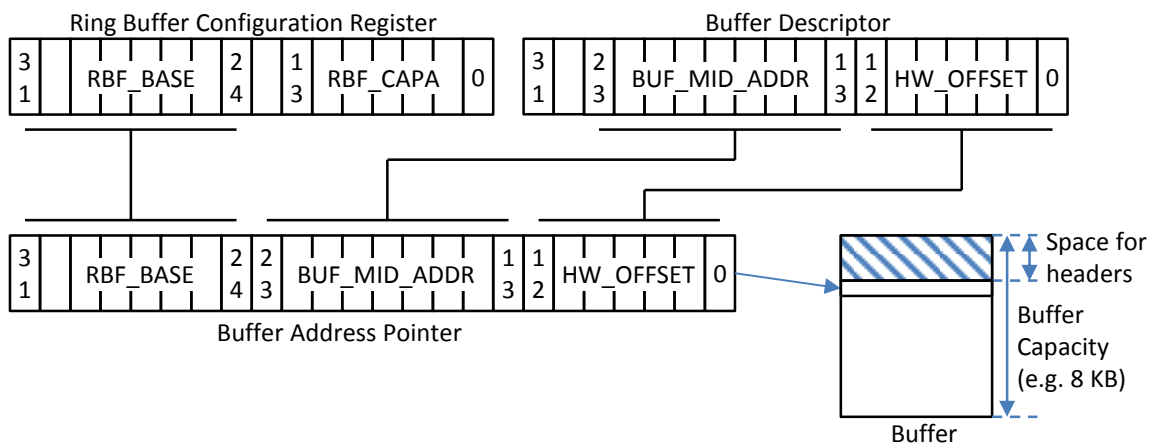
Fig. 59.   Forming the address of a free buffer.
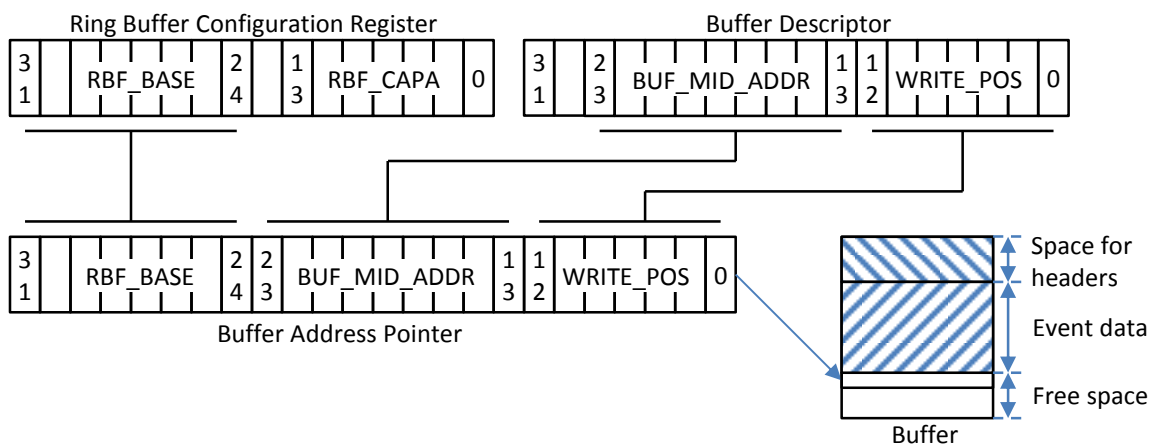
Fig. 60.   Forming the address of a buffer filled with data.

7.3 DUAL-PORTED MEMORY BLOCKS

In addition to its bank of configuration registers and the Ring Buffer Interface, the processor side of the TDCM also shares with the programmable logic side several dual-ported memory blocks. The base address of these memory blocks is assigned via Xilinx tools at synthesis time and cannot be changed afterwards. The corresponding address range must not be made cacheable. The dual-ported memory blocks used by the TDCM are listed in Table 20.

Table 20. TDCM variables mapped to dual-ported RAM locations.

Variable	Element Type	Function
hbusy[1024]	unsigned int	Dead-time histogram bins
hevper[1024]	unsigned int	Inter-event time histogram bins

hbusy:

This memory region is used to store the bins of the dead time histogram. Histogram accumulation is handled by the firmware side but software is responsible for clearing histogram bins. The dynamic range of each bin is 32 bits. Bin size is programmable among four values: 1 µs, 10 µs, 100 µs and 1 ms. The last bin of the histogram (#1023) is used to accumulate overflows. The range is selectable from 0 to 1.022 ms with 1 µs resolution to 0 to 1.022 s with 1 ms resolution.

hevper:

This memory region is used to store the bins of the inter-event time histogram. Histogram accumulation is handled by the firmware side but software is responsible for clearing histogram bins. The dynamic range of each bin is 32 bits. Bin size is fixed to 1 ms. The range of this histogram is [0; 1.022 s] because the last bin (#1023) is used to count overflows.

8 FRONT END NODE RESOURCE MAP

The TDCM sees the configuration and monitoring register space of each FE as a virtual memory organized in 16K-words of 32 bits. This memory space is accessible remotely by the TDCM via read and write transactions serialized on the communication link between the TDCM and the FEs using Virtual Channel B.

The configuration parameters of a FE and locally monitored variables are accessible via:

- a set of sixteen 32-bit wide registers,
- six dual-port memory blocks.

The address map of these different resources is shown in Table 21.

Table 21. Mapping of FE resources remotely accessible by TDCM.

Address range	Region	Type	Function
0x0000-0x1FFF	#0	Registers	Configuration Registers
0x2000-0x3FFF	#1	DPRAM	Hit Register Rules Table
0x4000-0x5FFF	#2	DPRAM	Test Data Table
0x6000-0x7FFF	#3	DPRAM	SPI Flash controller data buffers
0x8000-0x9FFF	#4	DPRAM	Pedestal and Threshold Table
0xA000-0xBFFF	#5	DPRAM	Dead-time histogram bins
0xC000-0xDFFF	#6	DPRAM	Histograms of per event channel hit count
0xE000-0xFFFF	#7	-	available

8.1 INTERNAL CONFIGURATION REGISTERS

The internal register map of a generic FE is shown in Table 22. The base address is 0x0000. All addresses are aligned on 32-bit boundaries and little-endian byte ordering is assumed.

Table 22. Internal registers of a FE.

Address	Register	Access	Function
Base+0x00	#0	R/W	General configuration
Base+0x04	#1	R/W	Front-end ASIC control
Base+0x08	#2	R/W	Trigger configuration
Base+0x0C	#3	R/W	Pulser configuration
Base+0x10	#4	R/W	Multiplicity thresholds
Base+0x14	#5	R/W	Configuration Register
Base+0x18	#6	R/W	S-TDCM interface
Base+0x1C	#7	RO	Free running clock cycle counter
Base+0x20	#8	R/W	Multiplicity Limit
Base+0x24	#9	R/W	Extended Configuration
Base+0x28	#10	R/W	Link message counter

Base+0x2C	#11	R/W	Multiplicity Control
Base+0x30	#12	R/W	SPI Flash Controller
Base+0x34	#13	R/W	XADC Interface
Base+0x38	#14	R/W	Available
Base+0x3C	#15	RO	Firmware Version

General Configuration (Register #0):

This register is shown in Fig. 61.

The SCA_CNT field determines the number of SCA cells to readout. This parameter can be set from 1 to 511 and 1 to 512 in the AFTER mode and AGET mode respectively.

The MODE_AFTER bit determines if the FE is being used to readout AFTER chips (MODE_AFTER=1) or AGET chips. This bit is 0 by default. Both modes may not be supported by some hardware flavors of the FE.

The RST_CHAN_CNT bit is used in the AGET mode to define the number of reset cycles that appear at the output of the device when switching from one column to the next. The AFTER chip has a fixed number of reset cycles which is equal to 3. The AGET chip can be programmed to have 2 or 4 reset cycles. In addition to programming the AGET chips with the desired value, the configuration register of the FE must be set accordingly: RST_CHAN_CNT=0 (default value) corresponds to 2 reset cycles.
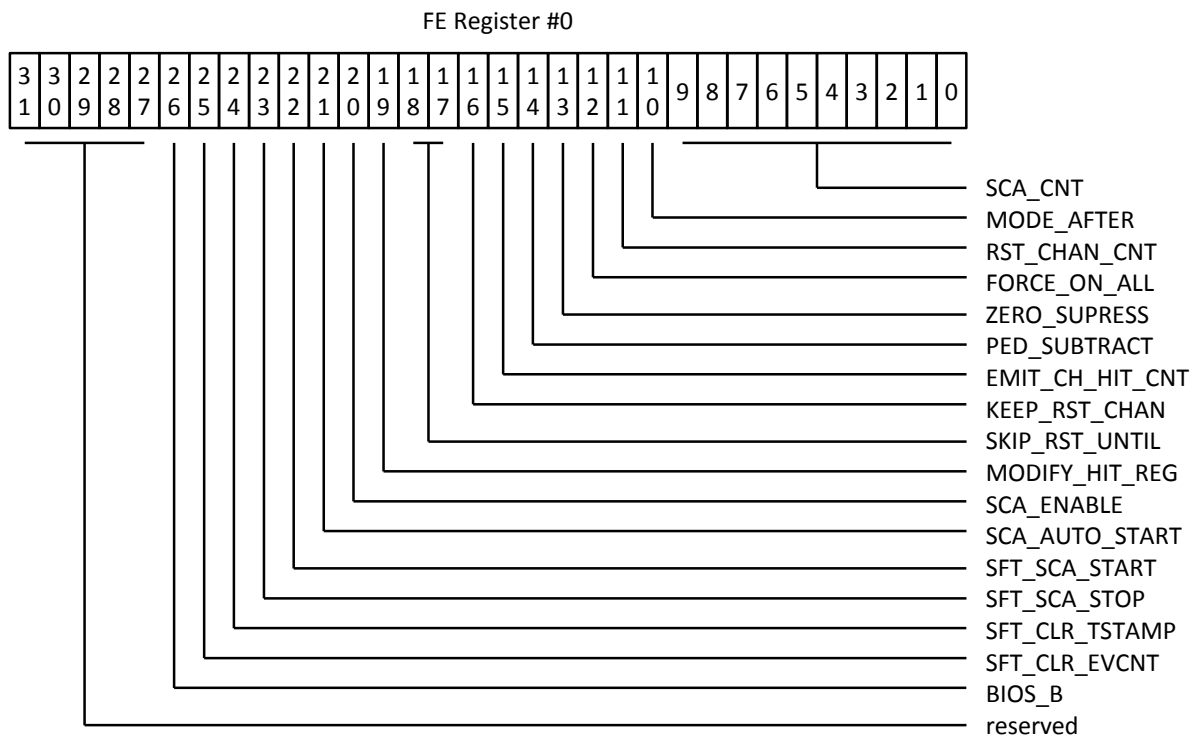
FE Register #0

Fig. 61.   General Configuration Register (Register#0).

When set, the FORCE_ON_ALL bit is used in the AGET mode to inform the local logic that all 64 physical channels and 4 FPN channels of the AGET chips are being readout. The AGET chips must be programmed accordingly. When this option is selected, the logic of the FE expects to receive 70 or 72 ADC samples per time-bin (corresponding to 2 or 4 reset values + 64 channels + 4 FPN). The readout of the channel hit register is skipped when this option is active. In the AFTER mode, this option need not be specified because the AFTER chip does not support sparse channel readout and, in this mode, the FE logic always expects 79 samples per time-bin (3 reset phases + 72 physical channels + 4 FPN).

The ZERO_SUPPRESS bit is used to optionally apply a zero-suppression algorithm on channel data. When this bit is not set, all data are kept. See zero-suppression algorithm details for more information. This option is valid in both AGET and AFTER modes.

The PED_SUBTRACT bit is used to optionally subtract a pre-loaded constant to each channel. See pedestal subtraction section for details. This option is valid in both AGET and AFTER modes.

The EMIT_CH_HIT_CNT is used to optionally add in the data stream sent to the TDCM the number of channels that were hit in each of the ASICs controlled by the FE. This information is useful for debugging. In the AFTER mode, the number of channel hit will always be 79. In the AGET mode, the value returned will vary from 6 to 72 depending on the number of reset channel count and hit register count (if it is not by-passed by other settings). Note that the number of reset phases and FPN channels count are always added to compute the total number of channel hit in a chip.

The KEEP_RST_CHAN bit can be optionally set to keep in the data stream sent to the TDCM the data corresponding to the reset channels of the AFTER or AGET chips. Keeping the data of these non-physical channels can be used for some specific debugging tasks.

The SKIP_RST_UNTIL field is used when KEEP_RST_CHAN is disabled to determine which of the reset channel data are discarded and which are sent to the TDCM. Most users will not want to keep any of the reset channel data. In the AGET mode, the corresponding value for this field is "01" when the number of reset phases is programmed to 2 and "11" when the number of reset phases is 4. In the AFTER mode, this field should be set to "10" to skip transmitting the data of 3 reset channels.

The MODIFY_HIT_REG bit is used to enable/disable modification of the hit register content before SCA digitization (only available in the AGET mode). When the MODIFY_HIT_REG bit and FORCE_ON_ALL bit are cleared, the logic of the FE reads the content of the Hit Channel Register of the active AGET chips it controls to determine the subset of channels that are readout for the current event. When MODIFY_HIT_REG is set to 1 while FORCE_ON_ALL is cleared, the logic of the FE reads the content of the Hit Channel Register of the active AGET chips, preserves forces to 1 or forces to 0 each

individual bit, and writes to the AGET chips the altered value. After all Channel Hit Registers have been updated, SCA digitization is initiated. The per-channel, PRESERVE, FORCE_ON and FORCE_OFF settings are programmed during system configuration. See the relevant section for details.

The SCA_ENABLE bit is used to enable/disable the operation of the ASICs controlled by the FE. This bit must be set to 1 for normal operation, after system configuration.

The SCA_AUTO_START bit is used to determine when the write operation in the SCA of the front-end ASICs can be started. When the SCA_AUTO_START bit is set to 1, the FE will start the SCA write operation for the next event as soon as possible after the digitization of the current event. When this bit is cleared, the FE will not restart the write operation in front-end SCAs until this order has been received from the TDCM. In a single, standalone FE setup, SCA_AUTO_START should be set to 1. It leads to the shortest possible dead-time. In a setup with several FEs controlled by a TDCM, this bit should be set to 0 for operation in a common dead-time mode, but it can also bit set to 1 if each FE is allowed to handle its own dead-time independently.

The SFT_SCA_START bit is used to asynchronously start writing in front end SCAs. The operation takes place on the transition from 0 to 1 of this bit. The user need not act on this setting when SCA_AUTO_START is active or when the SCA start order is received from the S-TDCM. This option is mainly used when the pulse generator of the FE is being used.

The SFT_SCA_STOP bit is used to generate software triggers. It should be set to 1 and then returned to 0. The SCA_START order must have been received (via the TDCM or the SCA_AUTO_START flag) prior to SFT_SCA_STOP to effectively trigger SCA digitization.

The SFT_CLR_TSAMP bit is used to asynchronously clear the event time stamp counter. The clear is performed when this bit is set from 0 to 1. The SFT_CLR_TSAMP bit must be cleared to 0 before this function can be re-used.

The SFT_CLR_EVCNT bit is used to asynchronously clear the event counter. The clear is performed when this bit is set from 0 to 1. The SFT_CLR_EVCNT bit must be cleared to 0 before this function can be re-used.

The bit BIOS_B is a read-only bit that reflect the state of the corresponding push-button on the FE. When this push button is pressed, BIOS_B is 0, and it is 1 when this button is released. In the version of the ARC that uses an embedded processor, this push button can be used to enter a special mode at startup which is used to configure some specific parameters (e.g. IP and MAC address, card ID, etc). Other implementations of the FE may not have this push button, or may use it for a different purpose.

Front-End ASIC Control (Register #1):

Initially, the interface to read and write to the registers of the ASICs of a front-end was designed to support up to 4 ASICs per front-end card. This interface has been upgraded to support up to 16 ASICs per front-end. The mapping of this register is shown in Fig. 62 and Fig. 63 for the 4 ASIC per front-end and 16 ASIC per front-end versions respectively. Even if a front-end only includes 4 ASICs, the 16-ASICs capable version of the interface should be used. The 4-ASIC version is now deprecated.

FE Register #1 (4 ASIC version)
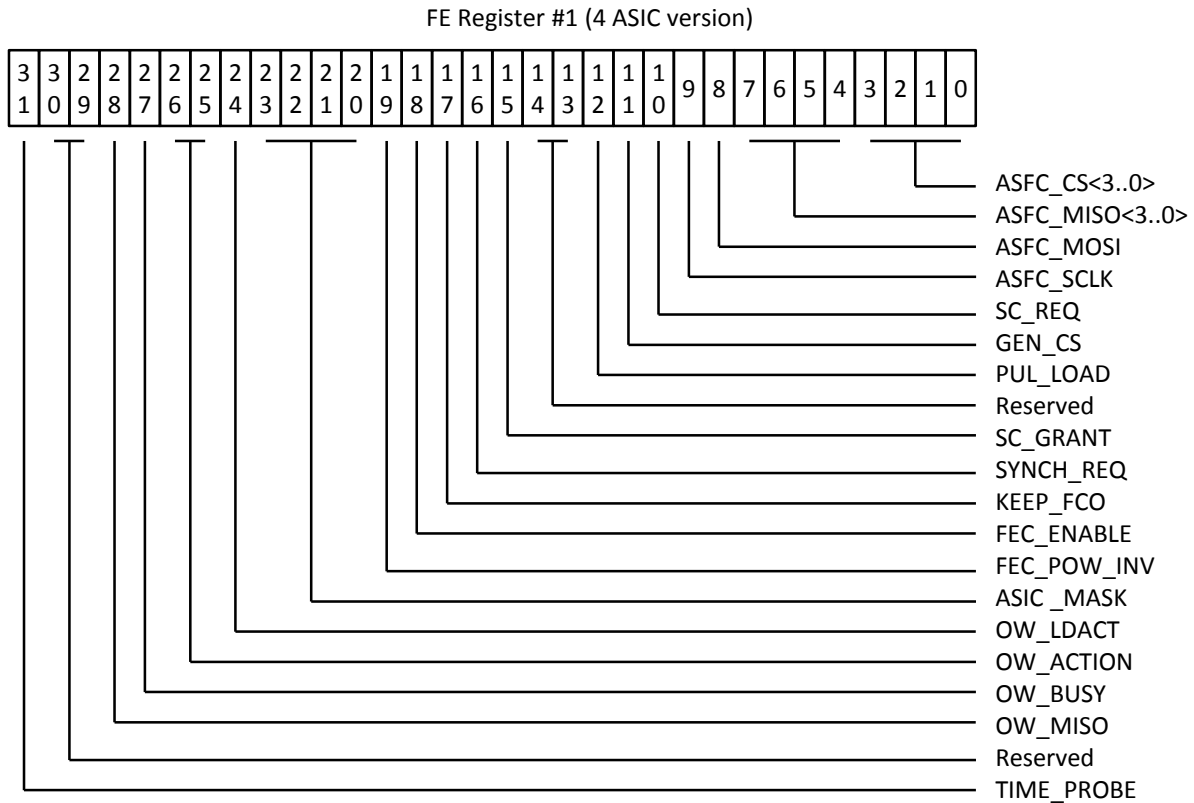
Fig. 62.   Front-end ASIC control register (Register #1) – 4 ASIC version. Deprecated.

In the 4-ASIC version of the interface, the ASFC_CS<3..0> bits are directly connected electrically to the slow control Sc_en line of each individual ASIC controlled by the FE. The ASFC_MISO<3..0> bits maps to the Sc_dout line of each individual ASIC. In the 16-ASIC version of the interface, the ASFC_CS_MUX<3..0> bits drive a 16-bit demultiplexer (implemented in firmware logic) that applies the state of bit ASFC_CS_VAL to one selected ASIC among up to 16. The bit ASFC_MISO takes the state of the Sc_dout line of the ASIC which is selected by ASCF_CS_MUX<3..0>. A 16-to-1 multiplexer implemented in firmware logic selects the Sc_dout line of one ASIC among up to 16.

In both the 4-ASIC and 16-ASIC versions of the interface, the ASFC_SCLK bit maps directly to the serial clock line (Sc_ck) of all ASICs, and the ASFC_MOSI bit maps to the Sc_din line which is common to all ASICs.

Embedded software running on the TDCM is responsible for controlling these lines in order to implement the serial protocol to read and write to the configuration registers of the ASICs of front-ends. For performance reasons, read/write operations from/to the Hit Channel Registers are implemented in firmware.

FE Register #1 (16 ASIC version)

		ASFC_CS_MUX<3..0>
		ASFC_CS_VAL
		Reserved
		ASFC_MISO
		ASFC_MOSI
		ASFC_SCLK
		SC_REQ
		GEN_CS
		PUL_LOAD
		Reserved
		SC_GRANT
		SYNCH_REQ
		KEEP_FCO
		FEC_ENABLE<1..0>
		reserved
		OW_LDACT
		OW_ACTION
		OW_BUSY
		OW_MISO
		OW_SEL
		TIME_PROBE
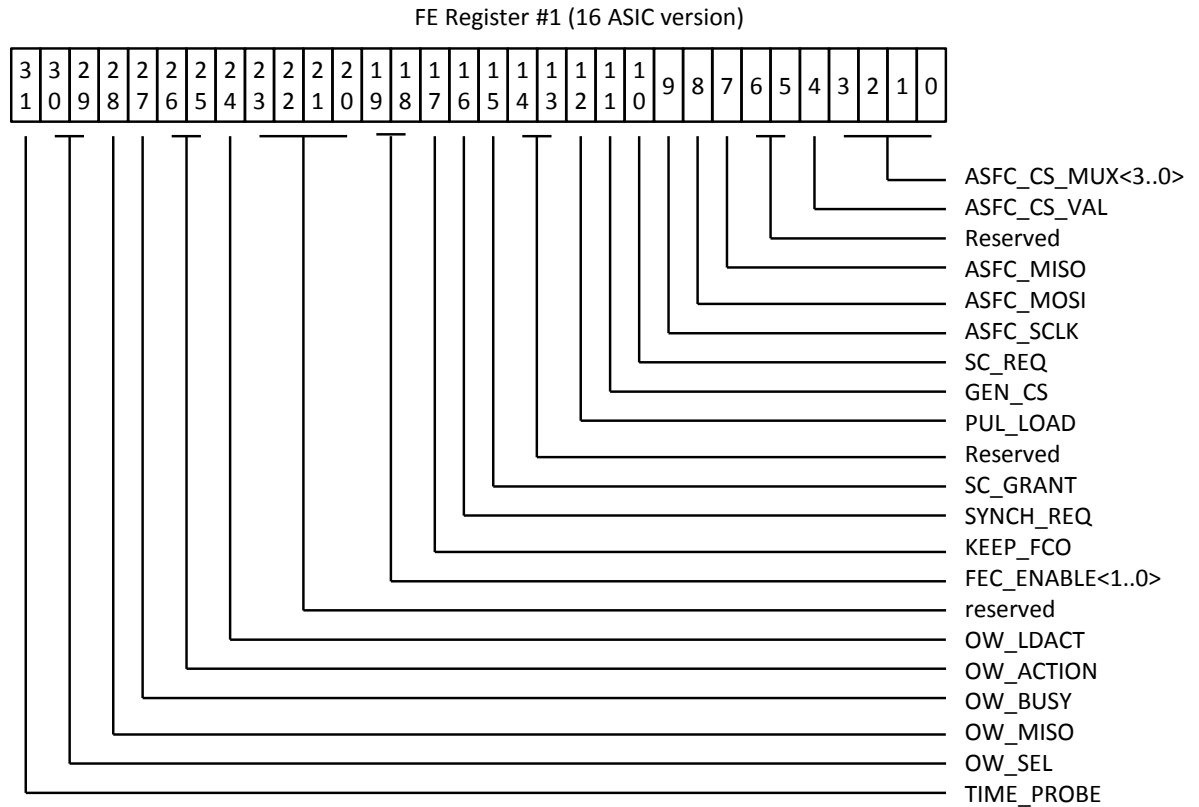
Fig. 63.   Front-end ASIC control register (Register #1) – 16 ASIC version – Active.

The SC_REQ bit is used to determine which of the remote processor or local firmware can access the slow control lines of the ASICs. When the TDCM wishes to perform a slow control operation on ASIC registers, it sets the SC_REQ bit to 1. The SC_GRANT bit will then be set by the local firmware to indicate that the slow control lines of the ASICs can be used. The SC_REQ should be held high until the end of the slow control operation. In the AGET mode, software is responsible for setting the slow control lines in the "slow control mode" just after access to these lines has been granted, and must configure these lines in the "channel address control mode" before releasing them. After system power-up, the embedded software should at least perform one slow control operation in each ASIC to ensure that the slow control lines are set in the "channel address control mode".

The GEN_CS bit is used as a chip select signal for programming the amplitude value of the on-board pulse generator of the FE. The lines ASFC_MOSI and ASFC_SCLK are used as the serial data input and serial clock when programming this generator. For

implementation reasons, the value programmed in the pulse generator cannot be read back. In order to program the DAC of the pulse generator, access to slow control lines must be obtained using the SC_REQ/SC_GRANT hand-shake protocol.

The PUL_LOAD bit is used to immediately update the DAC of the pulse generator to the pre-loaded value by pulsing signal GEN_GO. This bit is normally used to set/restore the baseline output level of the DAC of the pulse generator before the desired pulse amplitude is pre-loaded and automatically updated when pulse injection occurs.

The SYNCH_REQ bit is used to re-synchronize the ADC de-serializer logic. Only expert users should use this bit.

The KEEP_FCO flag is used to optionally replace the samples of ADC channel #3 (or #7 and #15 in the T2K-II FEM) by the framing pattern delivered by the ADC (signal "FCO" = "111111000000"). This option is used for advanced debugging of the ADC interface.

The FEC_ENABLE bit field is used to control power on up to two FECs attached to the front-end unit. By default, this bit is cleared and both FECs are OFF. One or both bits must be set to 1 to power ON the corresponding FECs. This operation must be done prior to any read/write operation in ASIC registers and data taking. For a hardware configuration that have only one FEC, only the LSB of this field is active. The control of power via this bit is not supported by all hardware implementations of the front-end card.

The OW_LDACT bit is used to initiate an action on the OneWire controller that drives the FEx_ID pin selected by OW_SEL (if the FE contains several OneWire devices). The action to be performed is specified by the OW_ACTION field as follows: "00" performs reset and presence pulse detection; "01" reads one bit from the OneWire device, "10" and "11" write 0 and 1 respectively to the device. After posting an action to the OneWire controller, the flag OW_BUSY is active until completion and the bit read from the OneWire device is available in OW_MISO.

The bit TIME_PROBE bit is intended to be routed to a spare FPGA output pin on the FE. Changing the state of this field in software can be used to measure precise time intervals with an oscilloscope connected to the corresponding pin. This function shall only be used by developers and a comparable pin may not be available in other front-end designs.

Trigger Control (Register #2):

This register is shown in Fig. 64.
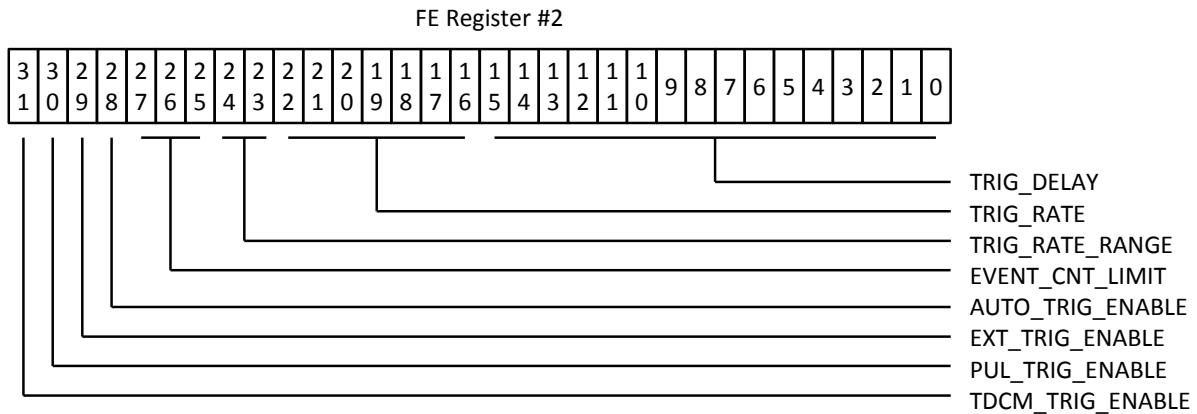
FE Register #2

Fig. 64.   Trigger Control Register (Register #2).

The TRIG_DELAY field is used to apply a fixed latency to the SCA_STOP order. The delay is expressed in 10 ns units and is programmable from 0 to 1.3 ms. This field supports two ranges, determined by the MSB of TRIG_DELAY. When TRIG_DELAY(15) is 0, the supported range is [0; 327.67 µs] in steps of 10 ns. When TRIG_DELAY(15) is 1, the supported range is [0; 1.31068 ms] in steps of 40 ns.

The TRIG_RATE field and TRIG_RATE_RANGE field are used to specify the trigger rate when the embedded periodic trigger generator is used. Four frequency ranges are available with 100 values per range. Range "00" is from 0.1 Hz to 10 Hz in steps of 0.1 Hz; range "01" is from 10 Hz to 1 kHz in steps of 10 Hz; range "10" is from 100 Hz to 10 kHz in steps of 100 Hz; range "11" is from 1 kHz to 100 kHz in steps of 1 kHz.

The EVENT_CNT_LIMIT is used to allow that only a pre-defined number of events flow through the system after SCA_ENABLE is asserted. This feature is useful for system debugging. When EVENT_CNT_LIMIT is set to "000", there is no limitation on the number of events that are allowed to pass though the FE. When EVENT_CNT_LIMIT is set to "001" or other values up to "111", the FE card will not respond to triggers after 1; 10; 100; 1000; 10,000; 100,000 or 1,000,000 events have been acquired. To resume operation, the SCA_ENABLE bit must be cleared and set back to 1. Setting a pre-defined number of events is useful for system development and debugging.

The AUTO_TRIG_ENABLE bit is used to start/stop the embedded periodic trigger generator. When enabled, the generator will start after an initial delay of ~1 s. Note that the readout system may not be able to sustain the desired trigger rate and some of the periodic triggers may be lost. The periodic generator is mostly used for system test and performance evaluation.

The EXT_TRIG_ENABLE bit is used to enable or mask triggers coming from the EXT_TRIG pin of the FE. This pin may not be available in other front-end card designs.

84

The PUL_TRIG_ENABLE bit is used to enable or disable the generation of a trigger when the pulse generator of the FE is fired. When this bit is active, the AUTO_TRIG_ENABLE bit and SCA_AUTO_START bits must be cleared and the PUL_ENABLE bit must be set.

The TDCM_TRIG_ENABLE bit is used to enable or mask the triggers received from the TDCM via its optical or cable link.

Pulser and General Control (Register #3):

The content of this register is shown in Fig. 65.

FE Register #3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PUL_DELAY
PUL_DELAY_RANGE
PUL_ENABLE
GEN_FT_TEST
Available
POLARITY
Available
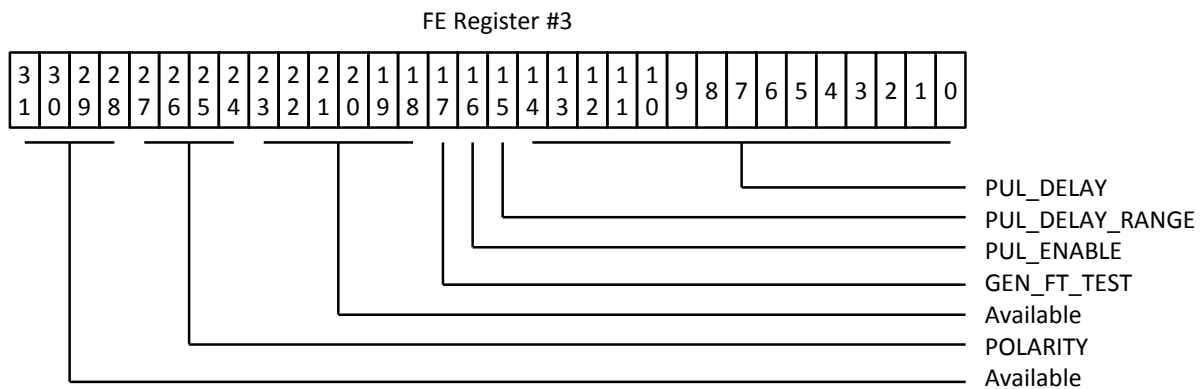
Fig. 65.   Pulser and General Control (Register #3)

The field PUL_DELAY is used to specify the delay from when the SCA write signal is asserted to when the pulse generator is fired. The delay is expressed in 10 ns and 40 ns units when PUL_DELAY_RANGE is 0 and 1 respectively. The maximum value is 327.67 µs with 10 ns resolution and up to 1.31068 ms with 40 ns resolution. Note also that it takes several clock cycles for the internal logic to actually fire the pulser. Pulse injection in the very first SCA cells is not possible when the SCA write clock is 100 MHz.

The bit PUL_ENABLE is used to enable/disable the use of the pulser. When this bit is set and PUL_TRIG_ENABLE is set, a trigger will automatically be generated when the pulser fires. When PUL_ENABLE is set without PUL_TRIG_ENABLE being set, the multiplicity trigger should be enabled.

The bit GEN_FT_TEST is used to optionally inject the output of the pulser to the functional test input of the front-end ASIC of the FE. When this bit is set to 0, the functional test input of the front-end ASICs shall be grounded. Nonetheless, the output of the pulser shall still be fed via a precision capacitor to the calibration input of the front-end ASICs. When GEN_FT_TEST is set to 1, the output of the calibration pulser is fed to both, the functional test input and the calibration input of the front-end ASICs. The actual behavior of this bit depends on the implementation of the pulser of the FE.

The POLARITY field is used to determine, on a per-chip basis, the behavior of the zero-suppressor depending on the polarity of detector signals. When a POLARITY bit is 0, the zero-suppressor keeps samples that are above threshold. This is intended to be used with detectors that deliver negative signals. When a POLARITY bit is 1, the zero-suppressor keeps samples that are below threshold. This mode is used with detectors that deliver positive signals. The polarity of each of the four AFTER or AGET chips controlled by the ARC can be programmed independently. This capability may differ in other front-end designs.

Multiplicity Thresholds (Register #4):

The FE can generate a local multiplicity trigger by applying a programmable threshold to each of the multiplicity signal of the four AGET chips. Each threshold is an 8-bit unsigned integer. Each threshold is only applied if it is greater than 0. The content of Register #4 is shown in Fig. 66.
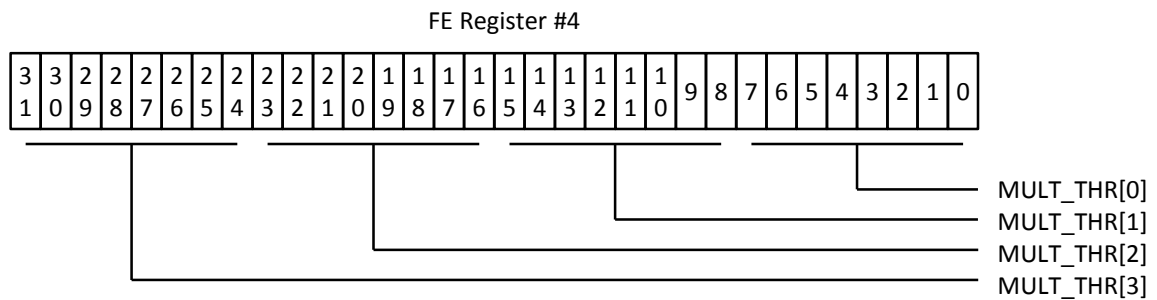
FE Register #4

Fig. 66.   Multiplicity Enable and Threshold (Register #4).

Configuration (Register #5):

This register is described in Fig. 67.
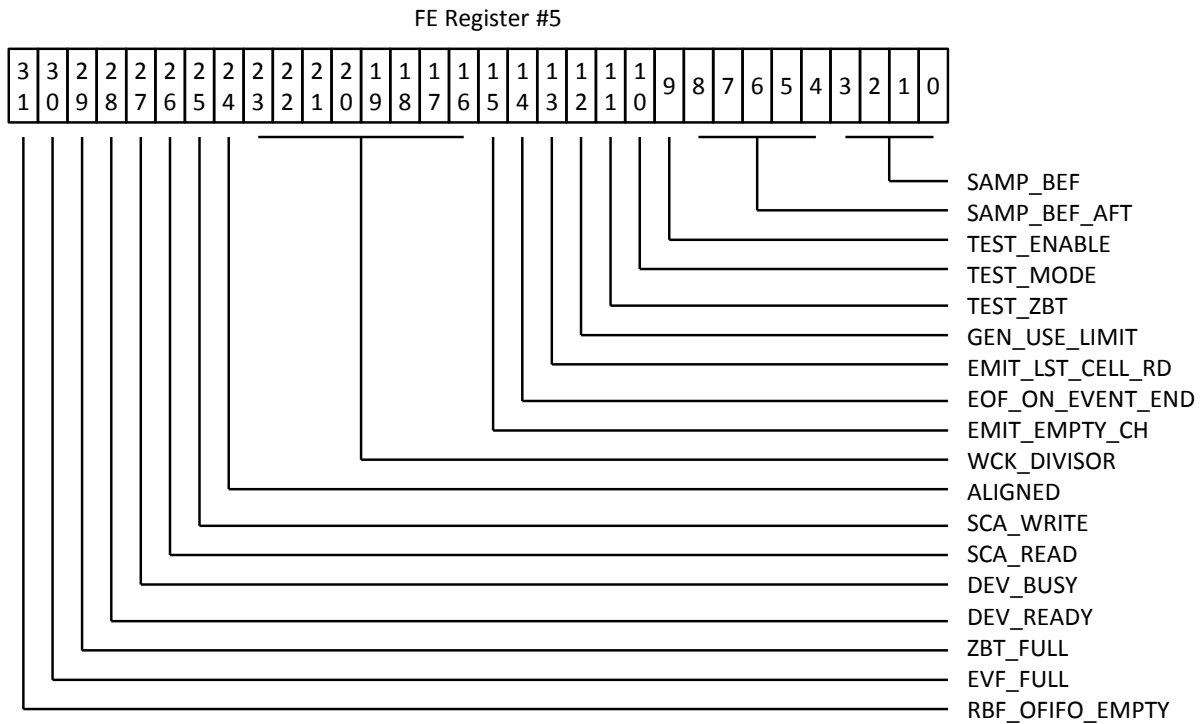
FE Register #5

Fig. 67.   Configuration (Register #5)

The field SAMP_BEF is used to specify how many samples should be kept before threshold is passed in zero-suppressed readout mode. The acceptable value is in [0; 15].

The field SAMP_BEF_AFT specifies the sum of the sample count kept before and after threshold is passed in zero-suppressed readout mode. The acceptable value is in [0; 31].

The TEST_ENABLE bit is used to select the source of data before the zero-suppression block. This bit should be cleared for normal operation where the data to process comes from the ADC of the external front-end card. When the TEST_ENABLE bit is set to 1, internally generated test data are used instead of ADC data.

When TEST_ENABLE is set to 1, the TEST_MODE bit is used to choose one of two types of test data. When TEST_MODE is 0, test data are formed with the 12-LSBs of the memory address of the ADC sample being readout. This setting is for advanced users only. When TEST_MODE is 1, test data are formed by sequentially reading-out an internal memory array. This memory array can be configured by the user with arbitrary values or some pre-defined simple patterns. Refer to the relevant section for details.

The bit TEST_ZBT is used to select the data written to the ZBT SRAM event buffer. When TEST_ZBT is 0, the data received from the ADC of the FE are written to this memory. When TEST_ZBT is set to 1, the 12-LSB's of the write pointer to the event buffer are written instead. This setting shall only be used for debugging or diagnosis.

The bit GEN_USE_LIMIT determines how the field EVENT_CNT_LIMIT is interpreted. When GEN_USE_LIMIT is set to 1, it is assumed that the internal event generator determines the number of events to generate. When SCA are enabled, the generator will make the desired number of events at the programmed rate and will stop. Events that could not be recorded by the SCAs will be dropped. When GEN_USE_LIMIT is 0, the internal generator or other sources of trigger will remain active until the SCAs have captured the programmed number of events.

The bit EMIT_LST_CELL_RD determines if the last cell read pointer retrieved from each ASIC are added or not to the stream sent to the TDCM. The value of the last cell read pointer of all ASICs can be compared at the DAQ level or later off-line to verify the correct synchronization of the front-end ASICs.

The bit EMIT_EMPTY_CH determines the content of the output stream sent to the TDCM for channels that were digitized but whose data has been entirely removed by the zero-suppression stage of the FE. When EMIT_EMPTY_CH is set to 1, empty channels will be sent to the TDCM. The information send for an empty channel comprises a 16-bit word to identify the channel index followed by a null 16-bit word to indicate that there is no data for this channel. When EMIT_CH_EMPTY is cleared, channels that have no data above threshold are not sent to the TDCM. This makes events more compact.

The field WCK_DIVISOR is used to set the value of the divisor to generate the SCA write clock. The SCA write clock is obtained by dividing the 100 MHz reference clock by WCK_DIVISOR. Valid values for WCK_DIVISOR are [1; 255] leading to discrete frequency values of 100 MHz, 50 MHz, 33 MHz, 25 MHz, etc., down to ~392 kHz.

The ALIGNED flag indicates that the ADC correctly delineates received data from the framing signal FCO. The binary pattern received on this line is six 1's followed by six 0's.

The SCA_WRITE and SCA_READ flags indicate that the front-end ASICs are currently in the write mode and read mode respectively.

The DEV_BUSY flag indicates that the FE is currently busy with the readout of the current event and is not ready to put front-end SCAs in the write state. The DEV_READY flag indicates that the FE is ready to change its internal state to put front-end SCAs in the write state.

The ZBT_FULL flag indicates that the memory for storing event data is full and cannot accept the next event. The EVF_FLAG indicates that some of the internal FIFOs used to store intermediate event information are full. The FE will not resume SCA write until space is available in the event memory and in these FIFOs.

Configuration (Register #6):

This register contains various settings related to the communication link with the TDCM. It is described in Fig. 68.

FE Register #6

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

STAT_CNT_ABC
STAT_RX_TX_SEL
STAT_CNT_SEL
STAT_CNT_LATCH
STAT_CNT_CLR
TX_CRC32_INSERT_ENA
Available
FE_RX_BERT_ENA
FE_RX_BERT_PAT
FE_TX_BERT_ENA
FE_TX_BERT_PAT
I2C_TARGET<1..0>
ADC_CS
ADC_SCLK
ADC_SDO
ADC_SDI
ADC_SDT
MMPOL<1..0>
I2C_SCL_O/T
I2C_SDA_O/T
I2C_SDA_I
Available
TSTAMP_CLR_ISSET (W)
TSTAMP_ISSET (R)
TS_INIT_SCLK
TS_INIT_SDAT
FRA_TIMEOUT<1..0>

Fig. 68.   TDCM interface register.

The STAT_CNT_ABC field is used in conjunction with STAT_RX_TX_SEL to determine on which Virtual Channel A, B, or C and on which direction, transmit to the TDCM or receive from the TDCM, the message counter and error message counter will be captured and read-out. The capture of the selected message and error counter occurs when STAT_CNT_LATCH is changed from 0 to 1. In the transmit direction, there is no error counter and only the number of transmitted messages over Virtual Channel A, B or C will be latched. In the receive direction, three counters will be latched for the selected Virtual Channel. These are: 1) the number of correct messages received on the selected Virtual Channel, 2) the number of messages received with a parity error, 3) the number of messages received with a format error. After the RX counters of one Virtual Channel has been latched, STAT_CNT_SEL can be changed to select for read-out the RX message counter or the two RX error counters. The field STAT_CNT_SEL has not effect when the TX counter is read-out. Once a counter, or a set of counters have been

captured, the latched value remains unchanged until a new transition from 0 to 1 occurs on STAT_CNT_LATCH. The value of the latched counter(s) is available in the Message Count Register of the FE, see Register #10 definition for details.

The behavior of the various settings for controlling the read out of the messages counters of the FE are summarized in Table 23.

Table 23. FE message counters readout control settings.

STAT_CNT_ABC	STAT_CNT_SEL	STAT_TX_RX_SEL	Counter for readout
00	0	0	RX message count A
00	1	0	RX error count A
01	0	0	RX message count B
01	1	0	RX error count B
10	0	0	RX message count C
10	0	0	RX error count C
11	X	0	Reserved
00	X	1	TX message count A
01	X	1	TX message count B
10	X	1	TX message count C
11	X	1	Reserved

The capture of message counters is summarized in Table 24.

Table 24. FE message counters capture.

STAT_CNT_ABC	STAT_TX_RX_SEL	STAT_CNT_LATCH	Counter latch
XX	X	0 or 1 stable or ↓	unchanged
00	0	↑	RX message and error counters A
01	0	↑	RX message and error counters B
10	0	↑	RX message and error counters C
11	0	↑	Reserved
00	1	↑	TX message count A
01	1	↑	TX message count B
10	1	↑	TX message count C
11	1	↑	Reserved

When the bit STAT_CNT_CLR is set to 1, this clears simultaneously the TX and RX counters of the Virtual Channel selected by STAT_CNT_ABC. The bit STAT_CNT_CLR must be cleared to 0 to enable the message counters. When STAT_CNT_ABC is set to "11"' and STAT_CNT_CLR is activated, all message and error counters on both TX and RX and the three Virtual Channels are cleared in the FE.

The bit TX_CRC32_INSERT_ENA determines if CRC-32 is appended at the end of packets transmitted by the FE to the TDCM over Virtual Channel C. It is recommended to always keep this bit equal to 1 so that CRC-32 generation is enabled on the FE transmitter side.

The field FE_RX_BERT_ENA, FE_RX_BERT_PAT, FE_TX_BERT_ENA and FE_TX_BERT_PAT are used to enable/disable and select the PRBS pattern of the bit error rate tester for the TDCM to FE and FE to TDCM direction respectively.

The field I2C_TARGET is used on FE that have multiple I2C ports and ADC slow control ports. It selects which I2C external bus and ADC slow control bus are active for the transaction to be accomplished. The T2K FEM has three I2C interfaces and two ADC slow control interfaces. Setting I2C_TARGET to "00" and "01" activates the I2C and ADC slow control interfaces of FEC#0 and FEC#1 respectively. Setting I2C_TARGET to "10" activates the I2C interface of the FEM. When I2C_TARGET is set to "11", all I2C and ADC slow control interfaces are disabled. For FE that only have one I2C interface, I2C_TARGET must be set to "00".

The bit fields ADC_CS, ADC_SCLK, ADC_SDO, ADC_SDI and ADC_SDT are used to control a SPI like port used to configure special functions of the on-board ADC of the FE, if the equipped model supports this interface (e.g. Analog Devices AD9928). Note that the bit ADC_CS is inverted before driving the corresponding FPGA output pin which is therefore name ADC_CS_B: setting the bit ADC_CS to a logic "1" translates into a low level on the associated FPGA output pin ADC_CS_B. The bit ADC_SDI is read-only. The bit ADC_SDT controls the tri-state function of the bi-directional FPGA I/O pin ADC_SDIO which is physically connected to the external ADC. This interface can be left unconnected if the model of ADC used on the FE is not configurable via this kind of serial interface, or it can be used to control some other serial device on-board the FE. Some FE may have several ADC slow control interface ports. The ADC slow control port which is active is determined by the state of I2C_TARGET<1..0> (commonly shared with I2C port selection).

The bit field MMPOL is used to control the PhotoMos relay that ground or leave floating the polarization resistors connected every input channel of the FE. Assuming a 256-channel FE, each PhotoMos relay controls the grounding of 128 channels. During normal operation, MMPOL should be set to "11" for the correct polarization of the resistors connected to input channels. In case of an unacceptable spark rate of the detector, or a permanent short between the high voltage mesh and one or several pads, the polarization of a group of 128 pads can disconnected from ground by setting the corresponding MMPOL bit to 0. Note that these pads do not produce meaningful data in this case. The feature to control protection circuits is primarily intended for use with

metallic Micromegas detectors. It is normally not needed with resistive Micromegas and other types of detectors.

The bit I2C_SCL_O/T, I2C_SDA_O/T and I2C_SDA_I control the two external pins that implement an I2C master interface within the FPGA of the FE. The bit I2C_SCL_O/T maps directly to the serial clock line. An external pull-up resistor is normally required. The bit I2C_SDO/T maps to the serial output line and the tri-state control of the corresponding bi-directional I/O pin of the FPGA. To configure the serial data line in input for the master, the bit I2C_SDO/T must be set to 0. The serial data is then retrieved in the read-only bit I2C_SDA_I. This I2C port is used to control some of the devices on-board the FE (e.g. the monitoring interface of the SFP optical transceiver). Some FE may have several I2C interface ports. The I2C port which is active is determined by the state of I2C_TARGET<1..0> (commonly shared with ADC slow control port selection).

The field FRA_TIMEOUT is used to select the value of the timeout for sending an empty data packet over Virtual Channel C when no event data is available to be sent. Setting FRA_TIMEOUT to "00", "01", "10" and "11" sets the timeout value to 1 ms, 10 ms, 100 ms and 1 s respectively.

Free running clock cycle counter (Register #7):

This register is a free running counter which is incremented every 20 ns. It provides the time reference needed to measure time intervals in software. This timer rolls over every 1' 25".

Multiplicity Limits (Register #8):

Two comparators are used on the multiplicity signal delivered by each AGET chip. To generate a multiplicity hit, it is required that the multiplicity signal is above a programmable threshold and is simultaneously less than a programmable limit.

The multiplicity limits are mapped to Register #8 as shown in Fig. 69. Each multiplicity limit is an 8-bit unsigned integer. It must be different from 0 if multiplicity trigger is used.
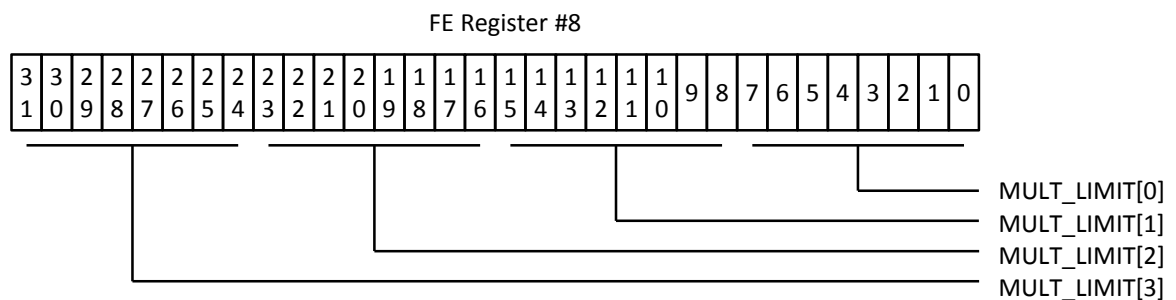
Fig. 69.   Multiplicity limit register.

Extended Configuration Register (Register #9):

This register is shown in Fig. 70.
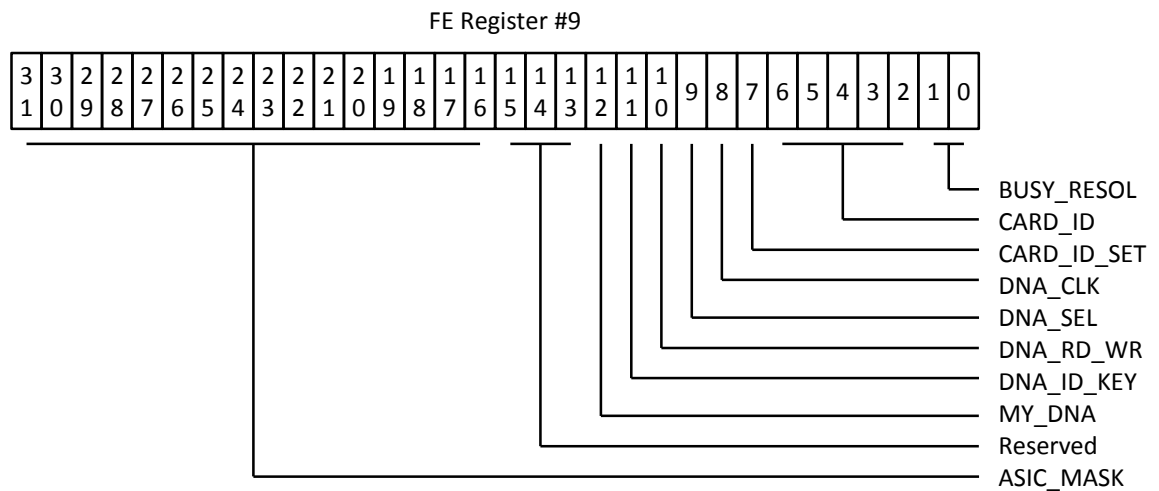
FE Register #9

Fig. 70.   Extended configuration register (Register#9).

The field BUSY_RESOL sets the resolution of the timer for the accumulation of the dead-time histogram. Four resolutions are available: 1 μs ("00"), 10 μs ("01"), 100 μs ("10") and 1 ms ("11").

The field CARD_ID contains the index of the card assigned by the TDCM after the card enumeration based on the DNA number of each card has been completed. This field is cleared after power-up. The bit CARD_ID_SET indicates if the index stored in CARD_ID is valid or not.

The bit DNA_CLK is the serial input clock for the operation to read/write the DNA number and FE index. The bit DNA_SEL is set to 1 to perform a read or write operation in the DNA number block. The bit DNA_RD_WR is set to 0 and 1 to perform a read or write operation respectively in the DNA number block. The bit DNA_ID_KEY is the input to serially load a DNA number to be compared with the local DNA number, as well of the corresponding FE index. The bit MY_DNA is used to serially output the local DNA and FE index.

The ASIC_MASK field is used to optionally disable one or several ASICs. Masked ASICs cannot be configured and readout. Reading and modifying their Channel Hit Register is skipped. Masking ASICs should be done when using partially equipped FE or to increase readout performance when only a fraction of the available channels of a FE are used. The user should always leave at least one un-masked ASIC for proper operation.

Message Counter Register (Register #10):

This register contains the value of the message counter that was latched. Note that the TX or RX message counters will roll over to 0 after reaching 2^{32}-1 while the RX error counters will saturate at 255 until they are cleared. The content of this register is shown in Fig. 71.
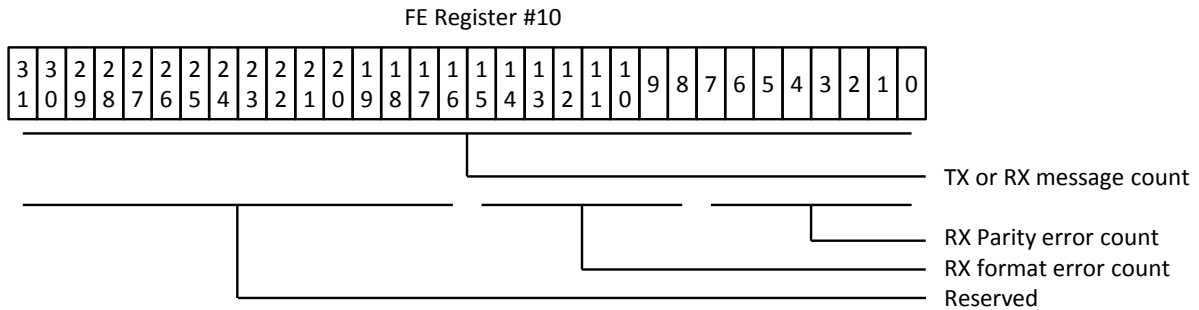
Fig. 71.   Message Counter Register.

Multiplicity Control Register (Register #11):

This register controls several functions related to multiplicity processing. The content of this register is shown in Fig. 72.
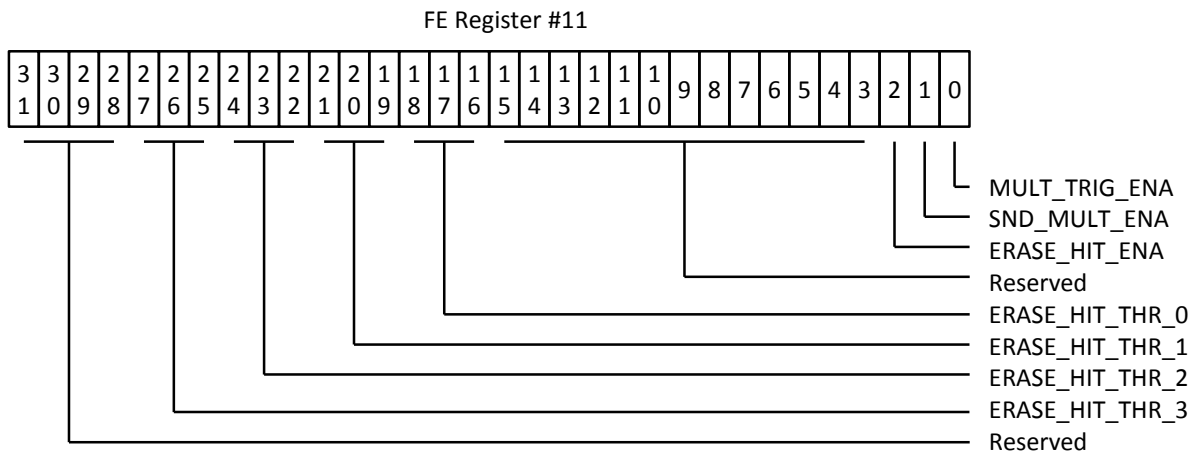
Fig. 72.   Multiplicity Control Register.

The bit MULT_TRIG_ENA is used to enable or disable the self-trigger based on multiplicity. When this bit is set, a self-trigger will be generated whenever the multiplicity output of one or several of the four AGET chips controlled by the FE passes the multiplicity threshold programmed.

The bit SND_MULT_ENA is used to enable or disable the transmission of the multiplicity-over-threshold bits to the TDCM. When this bit is active, the TDCM can be programmed to generate a trigger based on system-level multiplicity-over-threshold bits.

The bit ERASE_HIT_ENA is used to enable the function that clears the content of the hit register before digitization for chips that have a number of channel hit above a programmable threshold. This bit is only active in the AGET mode and when MODIFY_HIT_REG is also set.

The fields ERASE_HIT_THR specify for each AGET chip the maximum number of channel hit that are retained when ERASE_HIT_ENA and MODIFY_HIT_REG are set. In this mode, the content of the hit channel register is cleared if the number of hit channel is above the specified threshold. The threshold is a 3 bit value programmable from 0x0 to 0x7 and corresponds to a maximum allowable number of hit channels of 4 to 32 in increments of 4 channels.

SPI Flash Controller (Register #12)

This register is used with its associated DPRAM block by the SPI Flash Controller block that allows read-back and programming of the non-volatile memory device that stores the FPGA bitstream, and possibly other parameters. The content of this register is shown in Fig. 73.
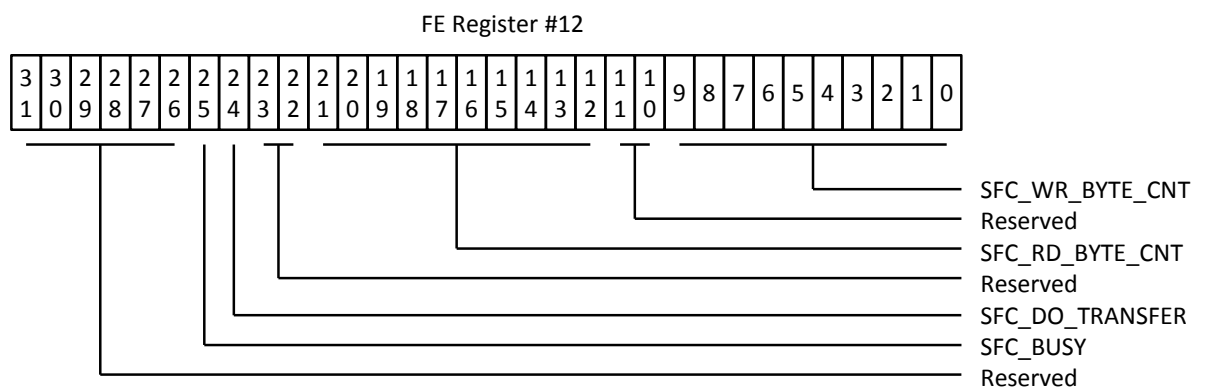
Fig. 73.   SPI Flash Controller Register.

The fields SFC_WR_BYTE_CNT and SFC_RD_BYTE_CNT are used to define respectively the number of bytes to write to and read from the SPI Flash memory. The write data must be programmed in the associated BRAM before the transfer is initiated and the read data is available in the BRAM buffer after the transfer is complete. The bit SFC_DO_TRANSFER initiates a transfer to/from the SPI flash memory when a transition from 0 to 1 is performed. The user must return SFC_DO_TRANSFER from 1 to 0 before a new transfer can be initiated. This can be done at any time. The bit SFC_BUSY is a read only bit which is set to 1 when a transfer is in progress and cleared when it is completed. No transaction request can be accepted when SFC_BUSY is active. Note that when SFC_BUSY is released, this only means that the desired command has been posted to the flash memory, but this does not mean that this device has effectively completed the

execution of this command and is ready to accept a new one. Polling on the status register of the flash memory is needed to determine the completion status of some commands (e.g. write, erase sector, etc.).

XADC Interface (Register #13)

This interface register is used to program the ADC embedded in the FPGA, "XADC" of the FE. The same interface is used to read back monitored variables. The content of this register is shown in Fig. 74.
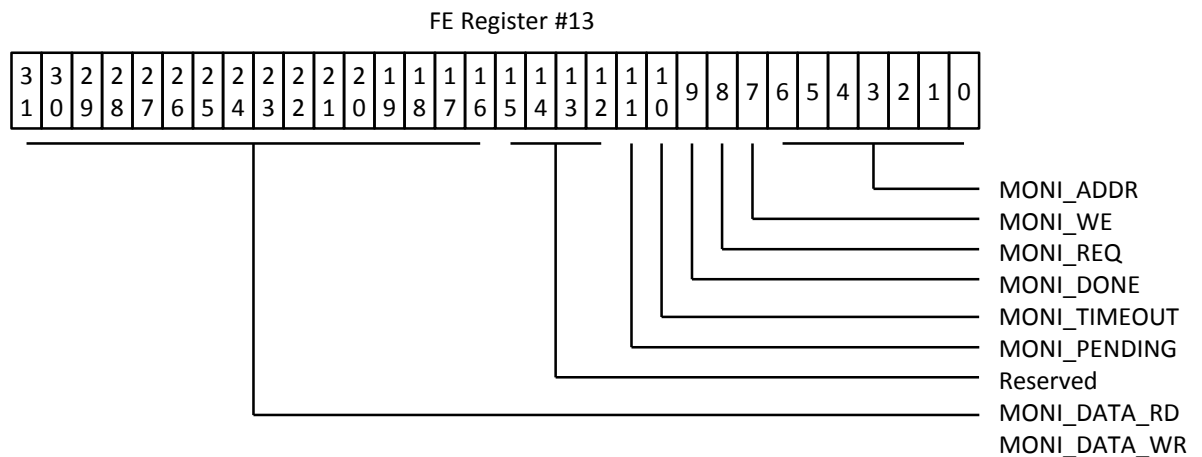
Fig. 74.   XADC Interface Register.

The field MONI_ADDR is used to select the desired XADC register on the Dynamic Reconfiguration Port (DRP). The bit MONI_WE is used to specify that the transaction is a write. If MONI_WE is set low, it is a read transaction. The field MONI_DAT_RD / MONI_DATA_WR is a written with the data to be loaded in the XADC for write transactions and contains the data read from the XADC upon read transactions. The MONI_REQ bit is set to a high level to initiate the actual transaction and shall be cleared upon completion. The bit MONI_PENDING is a read only field that is set to a high level when a transaction is pending. When the current transaction is completed the bit MONI_DONE is set. In case of error, the bit MONI_TIMEOUT is set.

Register #14:

This register is currently unassigned and are available for future use or specific functions of a particular implementation of the front-end.

Firmware Version Register (Register #15):

This read-only register contains the version number of the firmware currently loaded in the front-end. The content of this register is shown in Fig. 75.
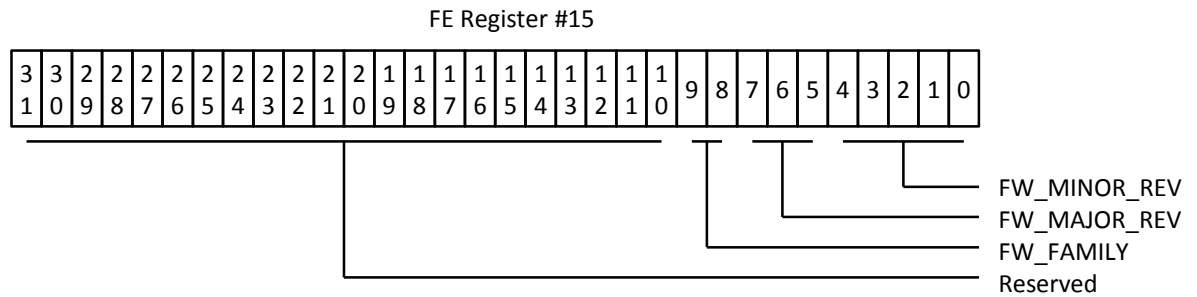
FE Register #15

Fig. 75.   Firmware Version Register.

The field FW_FAMILY is used to distinguish among several types of FE of several different projects. This field should be set to zero it is not used. The fields FW_MAJOR_REV and FW_MINOR_REV are used to specify the major and minor revision number of the current firmware. The firmware revision number can range from version 0.0 to version 7.31. The firmware revision number is managed manually at compilation time. The designer of the front-end firmware is responsible for assigning the major and minor revision number and updating them appropriately.

8.2 Dual-ported Memory Blocks

In addition to a bank of configuration registers, the FE is also controlled through several dual-ported memory blocks. Each memory block is mapped to an 8 Kbyte region within the 64 Kbyte virtual memory space accessible by the TDCM on Virtual Channel B.

Table 25. FE variables mapped to dual-ported RAM locations.

Variable	Element Type	Function
pedthrlut[4][128] or pedthrlut[16][128]	short, ushort	Pedestal and Threshold Table
hitregrule[4][128]	ushort, ushort	Hit Register Rules Table
testdata[4096]	unsigned short	Test Data Table
sfcbuffer[2][1024]	unsigned char	SPI Flash Controller W/R data buffers
hbusy[1024]	unsigned int	Dead-time histogram bins
hhitcnt[4][128]	unsigned int	Histogram of per event channel hit count

pedthrlut:

This table is used to store the constant pedestal and threshold value for each channel of the 4 (or 16) ASICs controlled by the FE. Although 128 entries are available for each ASIC, only 70, 72 or 79 entries are used depending on the selected mode (AGET or

AFTER) and the number of reset cycles (2 or 4 in the AGET mode). The per channel pedestal value is a 9-bit signed integer that is added to channel data when the bit PED_SUBTRACT of the configuration register is set. The available range for pedestal values is [-4096; +4095] ADC counts. Pedestals are coded in 2's complement. Threshold values are used to perform zero-suppression when the ZERO_SUPPRESS configuration bit is set. The zero-suppression phase takes place after the optional pedestal subtraction. Threshold values are 9-bit unsigned integers. The available range is [0; 4095] ADC counts.

hitregrule:

When operating in the AGET mode, this table is used to store the rules that determine how to optionally alter the content of the Hit Channel Register before SCA digitization. Each 32-bit entry in this table maps to two binary fields, ForceOn (bit 0) and ForceOff (bit 16). When ForceOn and ForceOff are cleared, the content of the corresponding bit in the Hit Channel Register is unaltered. When ForceOn is set to 1, the corresponding channel will be readout even if it was not hit. Reciprocally, when ForceOff is set to 1, the corresponding channel will be skipped from readout even if it was hit. ForceOn and ForceOff bits may not be set simultaneously. The table contains 128-entries per AGET but only 70 or 72 entries are used. Note that the channels corresponding to the reset sequence of the SCA (2 or 4 channels) cannot be skipped at this stage and the ForceOn/ForceOff flags have no effect on these channels. The readout of FPN channels is controlled by register settings in the AGET chips but the relevant ForceOn/ForceOff entries must also be programmed in a coherent way.

testdata:

This table is used to store a programmable pre-defined pattern of data to operate the FE in test mode. Each table entry is interpreted as a 12-bit unsigned ADC value corresponding to the successive time-bins of the channels being hit. The table has 4096-entries. Assuming that the user wishes to exercise the readout with 512 time-bins per channel, different arbitrary data for up to 8 channel hit can be programmed. Table entries are re-scanned from the first address for each event, and wraparound occurs when the number of channel hit multiplied by the number of time bins per channel exceeds table size. When accessing this table from the TDCM, the supplied address must be aligned on 4-byte boundaries. Selecting the two upper bytes or lower bytes allows the access to the desired 16-bit word.

sfcbuffer:

This memory area is the write buffer (lower half) and read buffer (upper) half needed by the SPI Flash Controller. The write buffer and read buffer can contain up to 1024 bytes each. The address supplied by the TDCM to read or write this area must be aligned on

4-byte boundaries. The four byte enable bits supplied with the address must be used to select which of the four addressed bytes are affected.

hbusy:

This memory region is used to store the bins of the dead time histogram. Histogram accumulation is handled by the firmware side but software is responsible for clearing histogram bins. The dynamic range of each bin is 32 bits.

hhitcnt:

Some implementations of the FE (e.g. the ARC) accumulates the histogram of the number of channels hit per event and per ASIC (i.e. 4 histograms are stored). The maximum number of channels hit is 72 and 79 in the AGET and AFTER mode respectively (recall that pedestal channels and reset channels are also counted), but enough space for 128 bins per histogram is reserved. Each histogram bin is a 32-bit unsigned value.

9 FORMAT OF MESSAGES AND DATA PACKETS

The front-ends communicate with the TDCM over point-to-point links with proprietary encoding while the TDCM communicates with the DAQ PC over a standard UDP/IP socket interface. Commands sent from the DAQ PC to the TDCM are encoded in plain ASCII text according to the syntax described in section 10. Commands are executed locally in the TDCM or translated into binary messages sent to the front-ends over virtual channels A, B or C. Responses from the front-ends are coded in binary format and are either reformatted in ASCII by the TDCM or simply packed and encapsulated in Ethernet frames before they are sent to the DAQ PC.

Data sent from the front-end to the TDCM assumes Little-Endian byte ordering. The same byte order is kept for communication with the DAQ PC. Contrary to the Internet convention, data sent by the TDCM use Little-Endian byte ordering.

The TDCM card supports standard Ethernet frame length (up to ~1500 bytes) for 10/100/1000 Mbps speed and Jumbo frames up to 8 KByte in Gigabit Ethernet mode. Messages sent by the TDCM to the front-ends have a fixed size. Messages sent from the front-ends to the TDCM also have a fixed size on virtual channel A and B, but data packets sent on virtual channel C have a variable size. The maximum size of a packet sent on virtual channel C is typically around 2 KByte. It corresponds to the size of the data of one front-end channel in uncompressed readout mode.

9.1 PREFIX-CODE FORMAT

The basic information datum is a 16-bit short word. For coding compactness, the front-end cards and TDCM uses a prefix-code where a variable fraction of the 16-bits of an information datum is used to define the type of data being encoded and the remaining bits (followed by one or several 16-bit words if needed) are used to encode the data itself. In order to decode data, application software needs to identify the prefix of each data element. The scan should be done starting from the shortest prefix then trying longer ones until a match is found. Once the prefix has been identified, the size of the data element is determined implicitly or can be retrieved from the data itself. If decoding software does not find a matching prefix in a data item, this is a serious error that indicates data transmission corruption, software bugs, non-synchronized protocol versions, or similar problems that need to be resolved for proper operation. The list of available prefix is listed in Table 26.

Table 26. List of prefix.

Prefix	Name	Function and Data Field
2-bit prefix	*14-bit data field*	
11xxxxxyyzzzzzzz	PFX_CARD_CHIP_CHAN_HIT_IX	Specify that sub-sequent data belong to channel <zzzzzzz> of chip <yy> of front-end <xxxxx>

10xxxxxxxxxxxxxx		Available for future use
01xxxxxyyzzzzzzz	PFX_CARD_CHIP_CHAN_HISTO	Specify that sub-sequent data is pedestal histrogram of channel <zzzzzzz> of chip <yy> of front-end <xxxxx>
4-bit prefix	***12-bit data field***	
0011xxxxxxxxxxxx	PFX_ADC_SAMPLE	Encodes a 12-bit ADC sample value
0010xxxxxxxxxxxx	PFX_LAT_HISTO_BIN	12-bit count of the current bin of the dead-time latency histogram
5-bit prefix	***11-bit data field***	
00011xxyyyyyyyyy	PFX_CHIP_LAST_CELL_READ	The last cell read pointer of ASIC <xx> is <yyyyyyyyy>
6-bit prefix	***10-bit data field***	
000101xxxxxxxxxx		Available for future use
7-bit prefix	***9-bit data field***	
0000111xxxxxxxxx	PFX_TIME_BIN_IX	The ADC samples that follow start at time-bin <xxxxxxxxx>
0000110xxxxxxxxx	PFX_HISTO_BIN_IX	Specifies the bin index in the current histogram
0000101xxxxxxxxx	PFX_PEDTHR_LIST	List of pedestals or thresholds
0000100xxxyzzzzz	PFX_START_OF_DFRAME	Start of data frame. Encoding version is <xxx> source type is <y> source index is <zzzzz>
0000011xxxyzzzzz	PFX_START_OF_MFRAME	Start of frame with monitoring information. Encoding version is <xxx> source type is <y> source index is <zzzzz>
0000010xxxyzzzzz	PFX_START_OF_CFRAME	Start of configuration reply frame. Encoding version is <xxx> source type is <y> source index is <zzzzz>
0001001xxyyyyyyy	PFX_CHIP_CHAN_HIT_CNT	Specifies that the total channel hit count of chip <xx> is <yyyyyyy>
0001000xxxxxxxxx		Available for future use
8-bit prefix	***8-bit data field***	
00000011xxyzzzzz	PFX_START_OF_EVENT	Start of event of type <xx> sent by source of type <y> and index <zzzzz>
00000001xxxxxxxx	PFX_ASCII_MSG_LEN	Specifies that the length of the ASCII string that follows is <xxxxxxxx> bytes.
10-bit prefix	***6-bit data field***	
0000001011yzzzzz	PFX_END_OF_EVENT	End of event sent by source of type <y> and index <zzzzz>
0000001010yzzzzz	PFX_BERT_STAT	Bit error rate tester statistics packet sent by source of type <y> and index <zzzzz>
0000001001		Available for future use
0000001000		Available for future use
12-bit prefix	***4-bit data field***	
000000001111xxxx	PFX_START_OF_EVENT_MINOS	(deprecated) Start of event. Even type is <xxxx>. 48-bit event time-stamp and 32-bit event count follow.
000000001110xxxx	PFX_END_OF_EVENT_MINOS	(deprecated) End of Event. MSB of 20-bit event size is <xxxx>, 16-LSB of event size follow
000000001101xxxx	PFX_EXTD_CARD_CHIP_LAST_CELL_READ	Last cell read in SCA of chip <xxxx>. Supports up to 16 chips per front-end
000000001100xxxx	unspecified	Available for future use
000000001011xxxx	unspecified	Available for future use
000000001010xxxx	unspecified	Available for future use
000000001001xxxx	unspecified	Available for future use
000000001000xxxx	unspecified	Available for future use

14-bit prefix	2-bit data field	
00000000011111xx	PFX_CH_HIT_CNT_HISTO	Channel hit count histogram. ASIC index is <xx>
0000000001111110	Unspecified	Available for future use
…	…	…
00000000010000xx	unspecified	Available for future use
15-bit prefix	1-bit data field	
000000000011111x	unspecified	Available for future use
…	…	…
000000000001010x	unspecified	Available for future use
16-bit prefix	Implicit data	
0000000000010011	unspecified	Available for future use
0000000000010010	PFX_EXTD_CARD_CHIP_CHAN_H_MD	Header for pedestal mean and standard deviation. Card, chip and channel indexes are given in the following 16-bit word
0000000000010001	PFX_EXTD_CARD_CHIP_CHAN_HIT_IX	Header for channel data. Card, chip and channel indexes are given in the following 16-bit word
0000000000010000	PFX_EXTD_CARD_CHIP_CHAN_HISTO	Header for channel pedestal histogram. Card, chip and channel indexes are given in the following 16-bit word
0000000000001111	PFX_END_OF_FRAME	End of frame indicator
0000000000001110	PFX_DEADTIME_HSTAT_BINS	Dead time statistics and histogram follow
0000000000001101	PFX_PEDESTAL_HSTAT	Pedestal histogram full statistics follow
0000000000001100	PFX_PEDESTAL_H_MD	Pedestal histogram short statistics follow
0000000000001011	PFX_SHISTO_BINS	(deprecated) Channel hit probability versus threshold histogram follow
0000000000001010	PFX_CMD_STATISTICS	Command statistics counter follow
0000000000001001	PFX_START_OF_BUILT_EVENT	(deprecated) Event start boundary when event builder active
0000000000001000	PFX_END_OF_BUILT_EVENT	(deprecated) Event end boundary when event builder active
0000000000000111	PFX_EVPERIOD_HSTAT_BINS	Inter-event time statistics and histogram follow
0000000000000110	PFX_SOBE_SIZE	Start Of Built Event with Size – only used with final event builder stage
0000000000000101	PFX_LONG_ASCII_MSG	Long ASCII message – The size of the message is given in the following 16-bit word
0000000000000100	PFX_EXTD_PEDTHR_LIST	List of pedestal or thresholds with up to 16 chips per front-end
…	…	…
0000000000000001	unspecified	Available for future use
0000000000000000	PFX_NULL_CONTENT	Null word to be skipped

The TDCM sends frames that can be classified in three different groups:

- Frames that contain the reply to a configuration command. These contain an error status code and a string message in ASCII format.
- Frames that contain event data for the DAQ. These are encoded in binary format.
- Frames that contain monitoring information (e.g. pedestal histograms, latency measurements, etc). These are also encoded in binary format.

Frame classification is done by scanning the first 16-bit word of the frame. The TDCM card tries to optimally fill Ethernet frames with data but it guarantees that the data samples of one channel (up to ~512-time bins) never spans across two frames.

9.2 FRAME ENCODING FOR CONFIGURATION COMMAND REPLIES

This type of frame is typically exchanged between the DAQ PC and the TDCM or a front-end that supports configuration over Ethernet. The format of the frame sent in response to a configuration command is shown in Fig. 76.
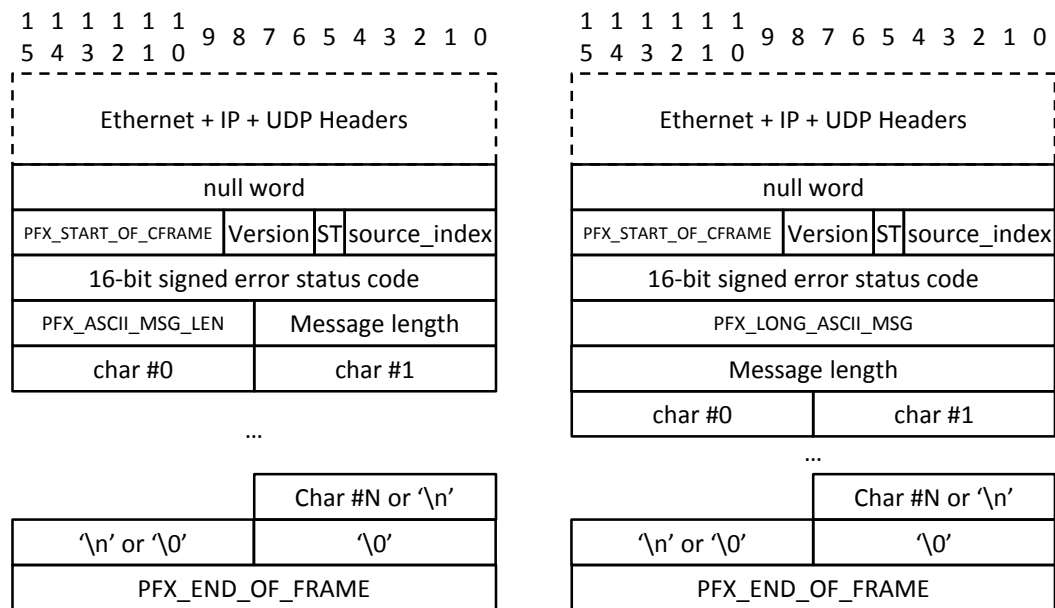
Fig. 76.   Reply frame to a configuration command.

A null word is always present after the UDP header. It is followed by the start of configuration frame prefix, PFX_START_OF_CFRAME, the protocol encoding version, the type of source sending this frame (0: front-end; 1: back-end) and the index of the TDCM. The following word is 16-bit signed error code. A negative value indicates that the command failed. A null or positive value indicates that the command completed successfully. The error code is followed by a prefix that indicates that subsequent data is an ASCII string. There are two format flavors: either the 8-bit prefix PFX_ASCII_MSG_LEN followed by the size of the ASCII string coded with 8-bit, or the 16-bit prefix PFX_LONG_ASCII_MSG followed by the size of the ASCII string coded on a 16-bit short integer. The length includes the size of the trailing carriage return (if any) but it does not include the size of the null terminating character which is always appended. If the length of the encoded string is even, two null characters (instead of one) are added so that the length of the message is always an even number of bytes. Depending on which prefix is used, strings of up to 255 or 65535 characters can be encoded. However, it is currently required that a response string fits in a single Ethernet frame, reducing the

maximum length to ~1480 characters with the standard MTU and ~8100 bytes with 8 KB Jumbo frames.

9.3 EVENT DATA ENCODING

The data of an event is composed of:

- An event header that contains the type of the event, a 48-bit timestamp, a 32-bit event count, and some optional fields.
- The body of the event which is composed of a series of packets, where each packet contains the data of one front-end channel.
- An event trailer that contains the total size of the event and other information.

The front-ends send the event header, each part of the event body and the event trailer in distinct individual packets. The TDCM performs local event building to assemble fragments that have the same event number and time stamp. The individual event header and trailer of each front-end are normally dropped and are replaced by a common event header and trailer. The different packets of the body of the event are simply concatenated. Although a packet of event data sent from a front-end to the TDCM can only contain an event header, the data of one channel, or an event trailer, an Ethernet frame sent by the TDCM to the DAQ PC may contain an event header and the data packets of several channels and an event trailer. The TDCM tries to optimize the filling of Ethernet frames depending on the allowable Maximum Transfer Unit (MTU) and the size of the packets received from the front-ends. The following rules apply:

- The front-ends are not allowed to split an event header, the data of one channel, and the event trailer across several packets. Each must be entirely contained in one packet.
- The TDCM is not allowed to split any event header, the data of one channel, or the event trailer across several Ethernet frames.
- Each front-end normally delivers event data in the incrementing order of channels, starting from the first channel of the first chip until the last channel of the last chip. However, sending channels in a different order is also permitted.
- The TDCM does not guarantee that the same order is kept. Normally, if the data packets of all channels are of equal size, the order will be: first channel of the first chip of the first front-end, followed by the first channel of the first chip of the second front-end, until the last channel of the last chip of the last front-end is reached. However, if a front-end responds faster than the others, if zero-suppression is enabled and makes the size of the data of each channel variable, or if only hit channels are read out, channel ordering may vary.

Nonetheless, for each given front-end, channels are guaranteed to appear in the same incremental order that is used for transmission between the front-end and the TDCM.

Although the content of the packets sent from the front-ends to the TDCM remain unchanged when they are transferred by the TDCM to the DAQ PC, encapsulation differs because the front-end links use a proprietary communication protocol while the TDCM uses Ethernet to communicate with the DAQ PC. The encapsulation of a packet on these two different type of links is shown in Fig. 77.
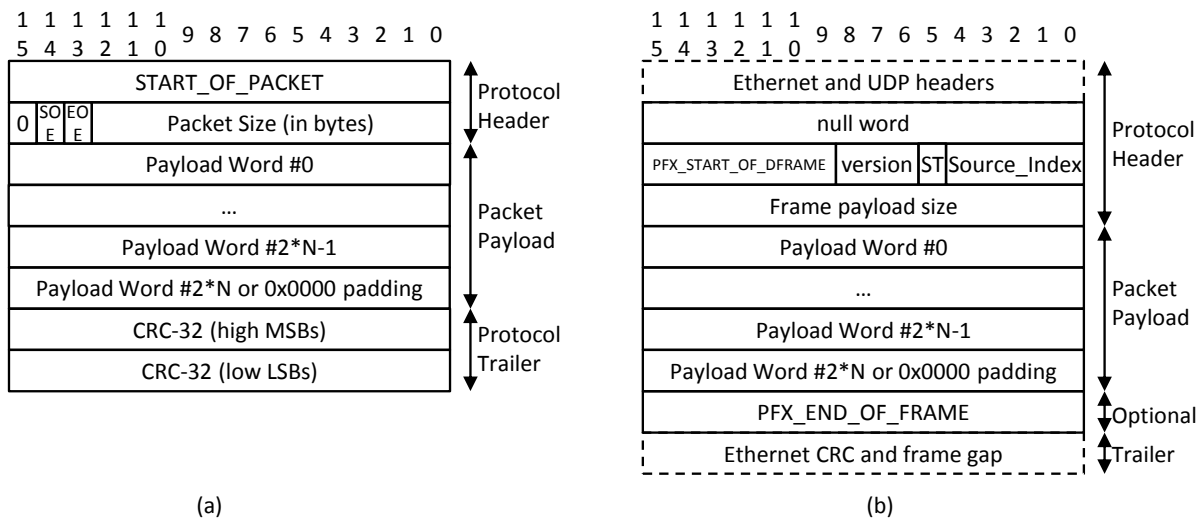
Fig. 77.   Packet encapsulation. (a) on the front-end links. (b) on the Ethernet link.

When a packet is transported by the link from a front-end to the TDCM, the protocol header is composed of a START_OF_PACKET synchronization word (16 bits) followed by a second 16-bit word where the field SOE indicates if this packet is a start-of-event, the field EOE indicates if this packet is an end-of-event packet, and the field PACKET_SIZE (13 bits) indicates the size of the packet payload in bytes. The size excludes the size of the protocol header words themselves and the size of the protocol trailer. The protocol trailer is composed of the CRC-32 of the packet, computed from the first word of the protocol header until the last word of the packet payload. If CRC-32 generation is disabled in the front-end, two null 16-bit words must be emitted instead. The TDCM shall normally be set to check CRC-32 on the fly and will only accept packet that have a correct CRC-32. For test purposes, CRC-32 verification can be disabled in the TDCM, and all packets will be accepted in this case.

When a packet is sent from the TDCM to the DAQ PC, the standard Ethernet, IP and UDP headers are immediately followed by a 16-bit word which is normally null. If not, it transports a sequence number to help identifying potential Ethernet frame losses. This word is followed by a START_OF_DFRAME prefix followed by a 3-bit protocol version

number, a one bit indicator for the type of source emitting the frame (0: front-end; 1: back-end) and the index of the source coded with 5 bits. It is followed by a payload size word which represents the size in bytes of the packet starting from the null word of the protocol header until the last payload word or the optional PFX_END_OF_FRAME trailer word. The rest of the protocol trailer, Ethernet checksum and frame gap, are automatically inserted by the MAC layer of the TDCM and stripped by the receiving PC.

The format of an event header packet is shown in Fig. 78. The protocol header is mandatory and depends on which physical link is used to transport the packet. After the prefix PFX_START_OF_EVENT, the field ETYPE indicates the type of event. Four different event types are supported. The field ST determines the type of source that emitted this packet: 0 when the packet originates from a front-end, 1 when it originates from a back-end card or device (e.g. the TDCM). The field SOURCE_ID is the index of the emitter, either a front-end node or a back-end node. This is followed by a 48-bit event time stamp and a 32-bit event count. No additional information is present when the packet originates from the back-end. When the packet originates from a front-end, additional information may be appended. The optional data specifies the total number of channel hit in each chip of the front-end, and the last position of the read pointer in the SCA matrix of each chip.
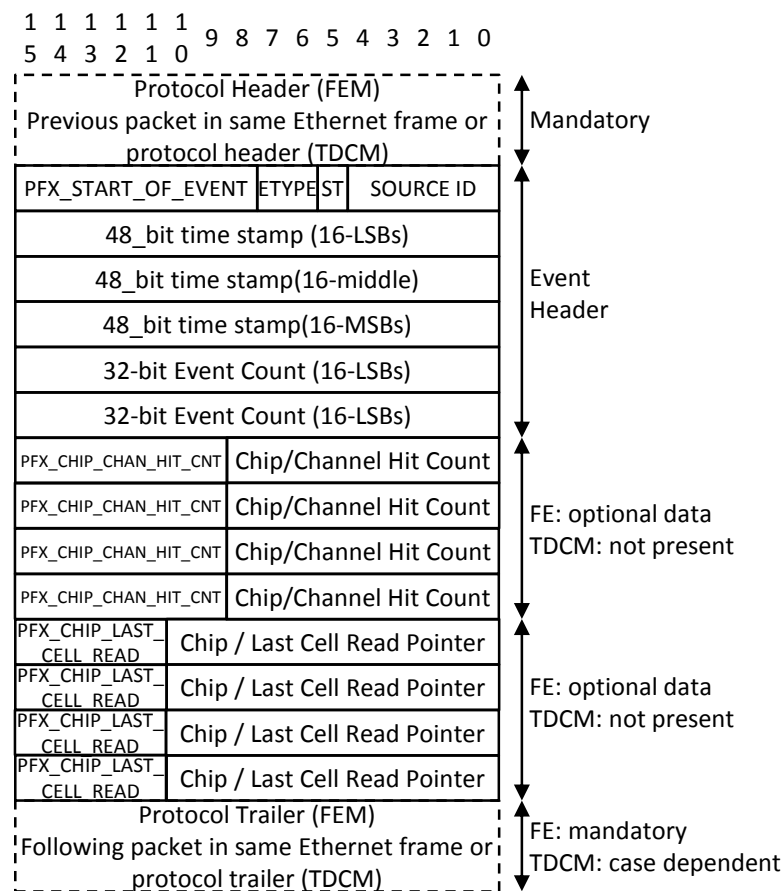
Fig. 78.   Event header packet (4 ASICs per front-end version).

Two versions for encoding the last read pointer are available: for front-end hardware implementations that have 4 ASICs, the prefix PFX_CHIP_LAST_CELL_READ can be used. For front-ends that supports up to 16 ASICs, the prefix PFX_EXTD_CARD_CHIP_LAST_CELL_READ shall be used. Because this type of front-end only supports the AFTER chip, the number of channel hit per ASIC is not encoded in the header because the detection of hit channels is not available in the AFTER chips. The event header for a front-end with up to 16 AFTER chips is shown in Fig. 79. Note that the last cell read pointer may be emitted for 0 or up to 16 ASICs, depending on the hardware configuration, and the EMIT_LST_CELL_RD and ASIC_MASK settings.
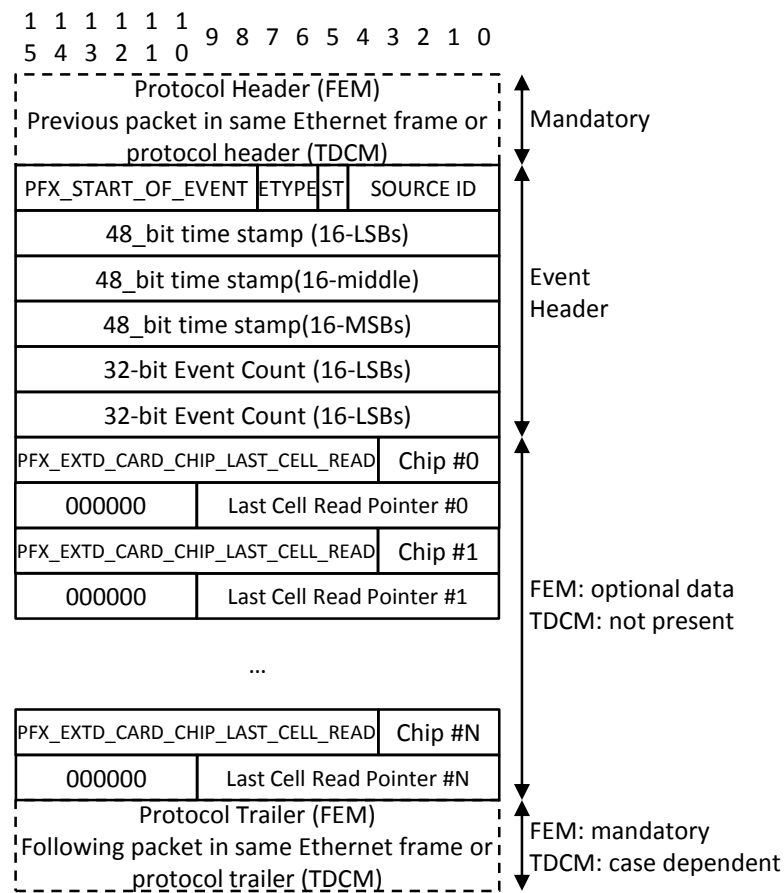
Fig. 79.   Event header packet (16 ASICs per front-end version).

A protocol trailer always terminates the packet when it is transported between the front-end and the back-end. When this packet is transported over Ethernet, the following packet may be stored in the same Ethernet frame, or the corresponding protocol trailer information is placed to terminate this frame. Every front-end card is required to send an event header packet for every event, even if it has no data for this event. The TDCM is also required to send one event header per event. The TDCM

normally drops silently the individual header of each front-end card, but retains all of them for diagnosis in case of event count or time stamp mismatch, or simply when this feature is enable for debugging. Consequently, the event header of the TDCM may be followed by all the individual event headers of each front-end card (including the optional data) before the actual data of all channels.

After the event header(s), the body of an event is composed by a series of packet, where each packet contains the entire data of one channel. The data packet of one channel in non-zero-suppressed read out mode is shown in Fig. 80.
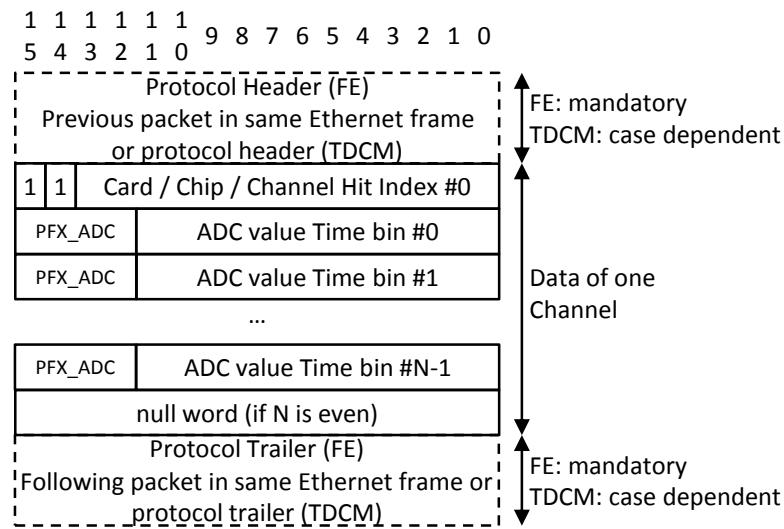
Fig. 80.   Packet of data of one channel in non-zero-suppressed readout mode.

The prefix "11" (PFX_CARD_CHIP_CHAN_HIT_IX) is followed by the ID of the sending front-end card (5 bits), the index of the chip (2 bits), and the index of the channel (7 bits). It is followed by the series of ADC samples (4-bit prefix PFX_ADC and 12-bit ADC data) corresponding to the successive time bins, in incremental order, starting from the oldest one. A null word is added if the number of time bins is even, so that the size of the packet is always an integral number of 32-bit words. After the last ADC sample or the null padding, the protocol trailer (FE to TDCM link) is added or the following packet is placed if the Ethernet frame can accommodate it (TDCM to DAQ PC Ethernet connection). The number of ADC samples may be less than 512 if only a fraction of the SCA depth of the front-end ASIC is read out. Note that time bins are delivered in incremental order and that the data of the corresponding channel must be entirely contained in the packet.

The format of the data packet of one channel in zero-suppressed readout mode is given in Fig. 81. It is similar to the format in non-zero-suppressed mode except that the index of the first SCA cell above threshold (prefix PFX_TIME_BIN followed by a 9-bit SCA cell index) is placed before each group of ADC samples that are kept by the zero-

suppressor. Depending on channel activity and the applied threshold, there may no sample at all above threshold, one series of samples (possibly containing every time-buckets), or multiple series of samples. Note also that the zero-suppressor can be programmed to keep samples below threshold around those that passed threshold. The ADC sample of a given time bucket is never duplicated among two adjacent pulses, but the zero-suppressor may produce several null samples in addition to the ADC values of a channel in the particular case when one or several of the first time buckets are above threshold and the zero-suppressor is programmed to keep samples before threshold is passed. Refer to the section that describes zero-suppressor for details.
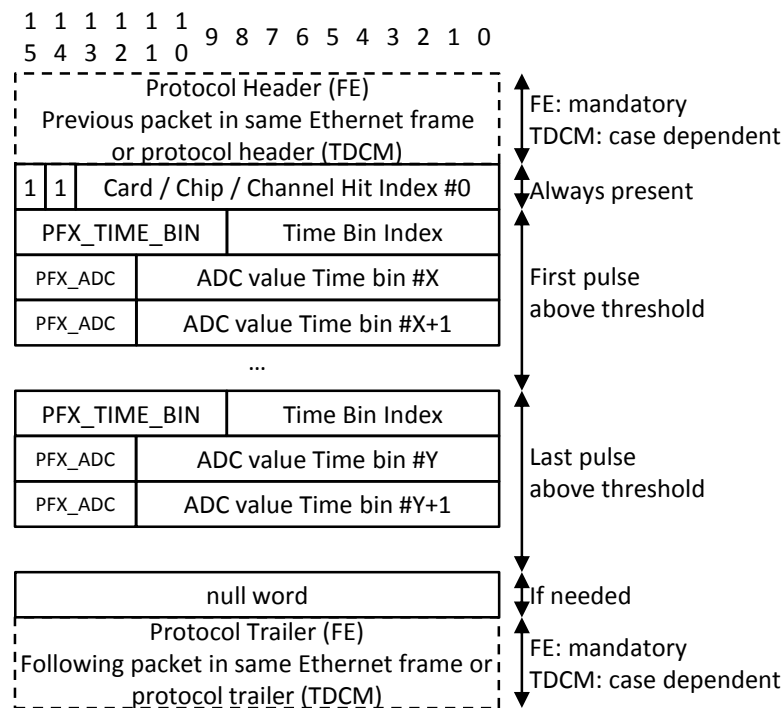
Fig. 81.   Packet of data of one channel in zero-suppressed readout mode.

The body of the event is composed of multiple packets as shown in Fig. 80 or Fig. 81. In full readout mode, the number of these packets is equal to the number of front-end cards (from 1 to 32) multiplied by the number of chip per cards (typically 4) and multiplied by the number of channels per chip (64 or 72 for AGET and AFTER respectively when only physical channels are read out, 72 or 79 if FPN channels and "reset" channels are also read out). In partial read out mode, only hit channels are present.

The format described in Fig. 80 and Fig. 81 is adapted to front-ends that have 4 ASICs. The TDCM also supports and extended data format for front-ends that have up to 16 ASICs. The format of a packet of channel data in the extended format is given in Fig. 82. It can be seen that compared to the regular format, the extended format takes 32 bits instead of 16 bits to specify the type of packet, card index, chip index and channel index.

The same modification applies to channel data packets with zero-suppression in the extended format.
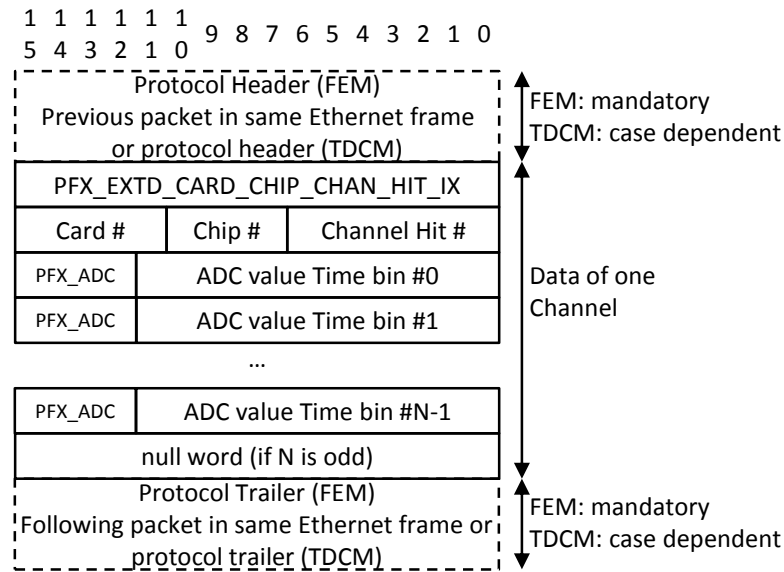
Fig. 82.   Packet of data of one channel, non-zero-suppressed, extended format.

After all the packets of channel data, the event is terminated by an end-of-event trailer packet. The format of this packet is shown in Fig. 83. After the end-of-event prefix, PFX_END_OF_EVENT, the field ST indicates the type of source emitting this packet (0: front-end; 1: back-end), and the field SOURCE_ID indicates the index of the source in the interval 0 to 31. The next two bytes are reserved for future use. These should be set to 0x0000 by front-ends. These bits will indicate error conditions when the packet is sent by a back-end card. This is followed by the size of the event coded on 32 bits. When the end-of-event packet is sent by a front-end, the size indicated is the total size of the data sent for the current event by this front-end. When the end-of-event packet is sent by the TDCM (or some other back-end device), the size indicated is the total event size across all front-ends. Normally, the TDCM does not propagate to the DAQ PC the end-of-event packet of each front-end, but only sends its own end-of-event packet based on the actual size of the event it sent. Optionally, the end-of-event packets of each front-end can be kept for debugging or in case of size mismatch. Note that the total size of an event normally differs from the sum of the event size of all front-ends because the start-of-event header and end-of-event trailer of the front-ends are generally not transmitted to the DAQ PC and because the TDCM also adds its own locally generated start-of-event header and end-of-event trailer.
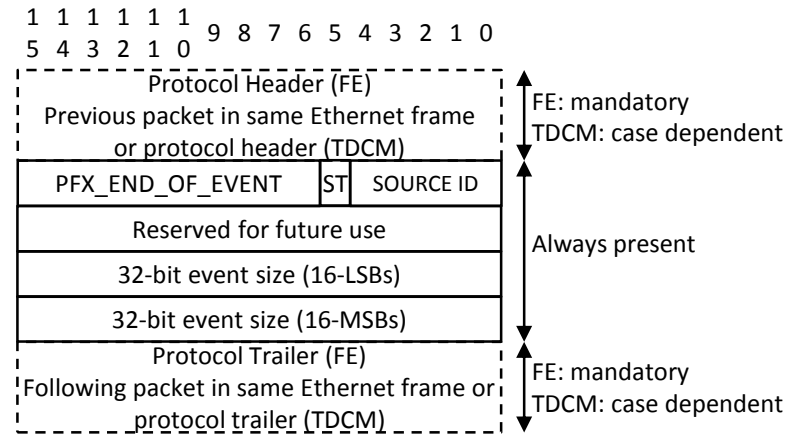
Fig. 83.   End of event packet.

Empty events require at least two packets at the output of each front-end card: one start-of-event packet and one end-of-event packet. At the output of the TDCM, an empty event will also contain at least a start-of-event header and an end-of-event trailer, but these will most likely be stored in the same Ethernet frame which may contain several consecutive events, or a mixture of empty and non-empty events depending on its capacity. An event may be empty because none of the ASIC channels were hit, or because none of the waveforms were retained after zero-suppression.

9.4   Encoding of Frames for Monitoring Information Data

Monitored variables that are simple scalar types (e.g. the supply voltage, current or temperature of a front-end or back-end card) are normally sent by the TDCM to the control PC in ASCII format, like configuration reply frames. More complex data types are transported in binary format in the so-called "monitoring frames". These frames have no equivalent on the front-end to TDCM links because monitoring information is retrieved from the front-ends by the TDCM using messages sent over virtual channel B. It usually takes multiple transactions on virtual channel B to retrieve even a scalar configuration word (e.g. the content of the register of an ASIC on a front-end card), a monitored variable, or a histogram accumulated on-line. But after this information is retrieved by the TDCM, it can be formatted in a single Ethernet frame that is sent at once. The different types of monitoring frames are detailed below.

9.4.1   Pedestal Histograms

The TDCM can accumulate the histograms of pedestal data of every front-end channel. This is accomplished during specific runs. These histograms are used to calculate the equalization constant for pedestal equalization and the thresholds for zero-suppression. These two operations are performed on-line by the front-ends, but the computations of the appropriate equalization constants and thresholds are done by the TDCM. The pedestal histograms of each given channel can be sent to the DAQ PC in

a condensed or detailed format. The detailed format contains the list of all the bins that contain a non-null count. The format of this frame is shown in Fig. 84.
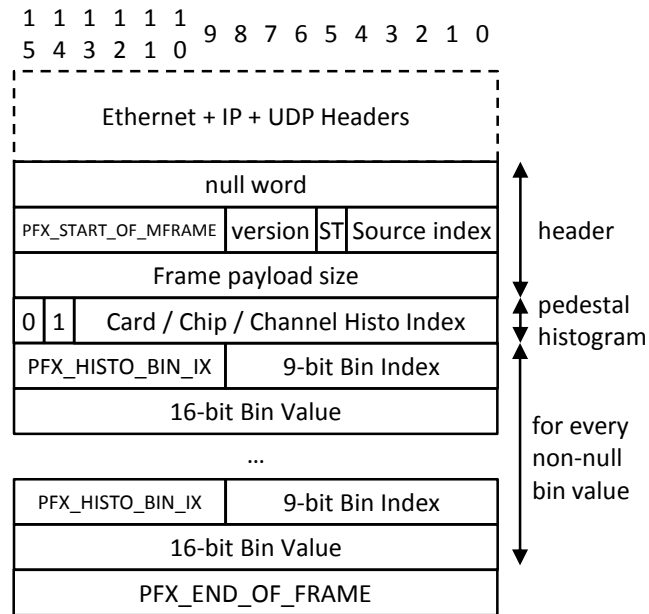
Fig. 84.   Pedestal Histogram in non-null bin list format.

In this format, an Ethernet frame contains the pedestal histogram of only one channel. The list of bins may be truncated if the histogram does not fit in one Ethernet frame (the current limit is fixed at ~1400 bytes because the corresponding section of the current code does not use Gigabit Ethernet Jumbo frames). The list of bins may also be empty. The frame payload size is the size in bytes counted from the START_OF_MFRAME prefix to END_OF_FRAME (including both of them).

The format described in Fig. 84 is adapted to front-end cards that have 4 ASICs. An extended version of the format allows to support up to 16 ASICs per front-end card. This is shown in Fig. 85.

| 1 1 1 1 1 1 9 8 7 6 5 4 3 2 1 0 |
| 5 4 3 2 1 0 |

| Ethernet + IP + UDP Headers |

| null word |
| PFX_START_OF_MFRAME | version | ST | Source index |
| Frame payload size |
| PFX_EXTD_CARD_CHIP_CHAN_HISTO |
| Card # | Chip # | Channel # |
| PFX_HISTO_BIN_IX | 9-bit Bin Index |
| 16-bit Bin Value |
| … |
| PFX_HISTO_BIN_IX | 9-bit Bin Index |
| 16-bit Bin Value |
| PFX_END_OF_FRAME |

header

pedestal histogram header

for every non-null bin value
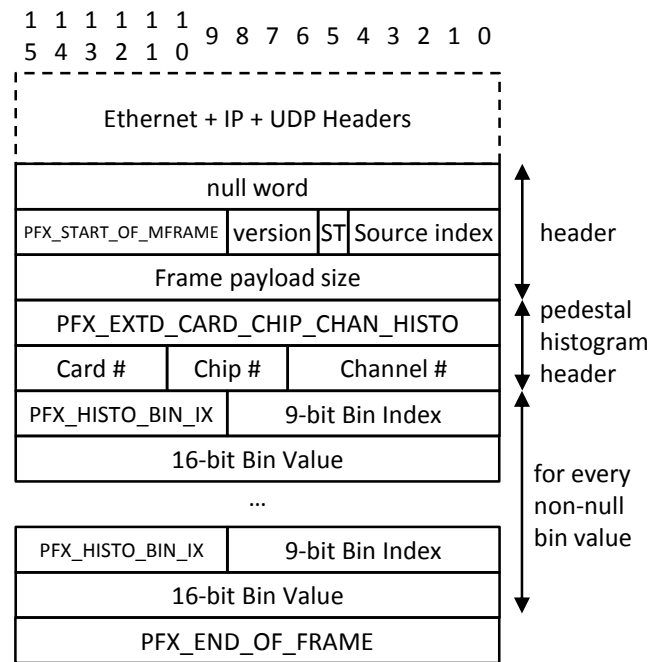
Fig. 85.   Pedestal Histogram, non-null bin list, extended format.

A pedestal histogram can also be sent in a more condensed format as shown in Fig. 86.

```
1 1 1 1 1 1
5 4 3 2 1 0  9 8 7 6 5 4 3 2 1 0
```

Ethernet + IP + UDP Headers

null word	header
PFX_START_OF_MFRAME | version | ST | Source_index	
Frame payload size	
0 | 1 | Card / Chip / Channel Histo Index	pedestal histogram
PFX_PEDESTAL_HSTAT	
32-bit Min_Bin (16-LSBs)	
32-bit Min_Bin (16-MSBs)	
32-bit Max_Bin (16-LSBs)	
32-bit Max_Bin (16-MSBs)	
32-bit Bin_Width (16-LSBs)	
32-bit Bin_Width (16-MSBs)	
32-bit Bin_Count (16-LSBs)	
32-bit Bin_Count (16-MSBs)	for one channel
32-bit Min_Non_Null_Bin (16-LSBs)	
32-bit Min_Non_Null_Bin(16-MSBs)	
32-bit Max_Non_Null_Bin (16-LSBs)	
32-bit Max_Non_Null_Bin (16-MSBs)	
32-bit Mean*100 (16-LSBs)	
32-bit Mean*100 (16-MSBs)	
32-bit Std_Dev * 100 (16-LSBs)	
32-bit Std_Dev * 100 (16-MSBs)	
32-bit Entry_Count (16-LSBs)	
32-bit Entry_Count (16-MSBs)	
PFX_END_OF_FRAME	

Fig. 86.   Pedestal histogram detailed statistics frame.

Pedestal histogram detailed statistics can also be formatted in extended format that supports up to 16 ASICs per front-end. This frame is shown in Fig. 87. Note that the pedestal histogram header includes a 16-bit padding word to guarantee the alignment of subsequent 32-bit integers on 32-bit boundaries.

```
1 1 1 1 1 1
5 4 3 2 1 0  9 8 7 6 5 4 3 2 1 0
```

Ethernet + IP + UDP Headers

null word	header
PFX_START_OF_MFRAME | version | ST | Source_index	
Frame payload size	

PFX_EXTD_CARD_CHIP_CHAN_HISTO	pedestal histogram header
Card # | Chip # | Channel #	
PFX_PEDESTAL_HSTAT	
0x0000	

32-bit Min_Bin (16-LSBs)	Pedestal histogram features for one channel
32-bit Min_Bin (16-MSBs)	
32-bit Max_Bin (16-LSBs)	
32-bit Max_Bin (16-MSBs)	
32-bit Bin_Width (16-LSBs)	
32-bit Bin_Width (16-MSBs)	
32-bit Bin_Count (16-LSBs)	
32-bit Bin_Count (16-MSBs)	
32-bit Min_Non_Null_Bin (16-LSBs)	
32-bit Min_Non_Null_Bin(16-MSBs)	
32-bit Max_Non_Null_Bin (16-LSBs)	
32-bit Max_Non_Null_Bin (16-MSBs)	
32-bit Mean*100 (16-LSBs)	
32-bit Mean*100 (16-MSBs)	
32-bit Std_Dev * 100 (16-LSBs)	
32-bit Std_Dev * 100 (16-MSBs)	
32-bit Entry_Count (16-LSBs)	
32-bit Entry_Count (16-MSBs)	
PFX_END_OF_FRAME	

Fig. 87.   Pedestal histogram detailed statistics frame, extended format.

Pedestals histograms statistics can be sent in a shorter version that only contains the mean value and standard deviation. In this format, a frame can contain the pedestal mean/deviation of several channels. This is shown in Fig. 88. The frame payload size is the size in bytes counted from the START_OF_MFRAME prefix to END_OF_FRAME (including both of them).

```
1 1 1 1 1 1
5 4 3 2 1 0   9 8 7 6 5 4 3 2 1 0
```

Ethernet + IP + UDP Headers

null word

| PFX_START_OF_MFRAME | version | ST | Source_index |

| Frame payload size |

| 0 | 1 | Card / Chip / Channel Histo Index |

| PFX_PEDESTAL_H_MD |

| 32-bit Mean * 100 (16-LSBs) |

| 32-bit Mean * 100 (16-MSBs) |

| 32-bit Std_Dev * 100 (16-LSBs) |

| 32-bit Std_Dev * 100 (16-MSBs) |

...

| PFX_END_OF_FRAME |

header

pedestal histogram short statistics of one channel
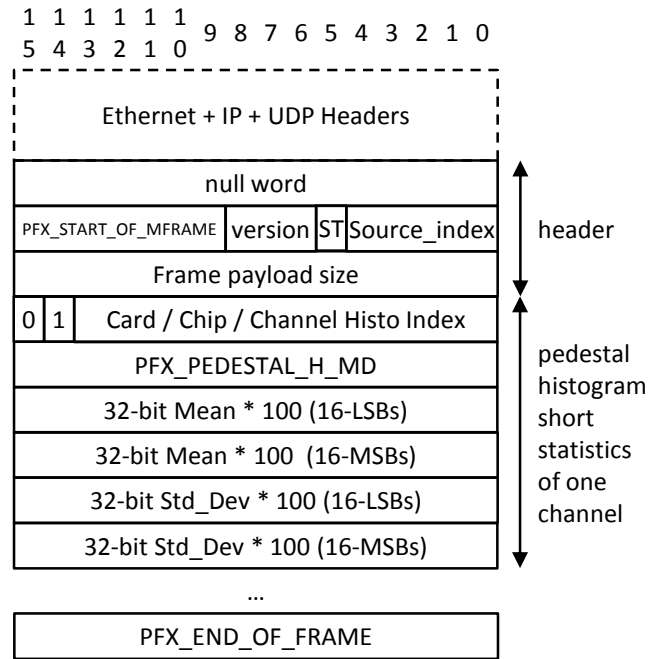
Fig. 88.   Pedestal histogram condensed statistics frame.

The condensed form of the pedestal histogram is also available in the extended format that supports 16 chips per front-end card instead of 4. This is shown in Fig. 89.

```
1 1 1 1 1 1
5 4 3 2 1 0   9 8 7 6 5 4 3 2 1 0
```

Ethernet + IP + UDP Headers

null word

| PFX_START_OF_MFRAME | version | ST | Source_index |

| Frame payload size |

| PFX_EXTD_CARD_CHIP_CHAN_H_MD |

| Card # | Chip # | Channel # |

| 32-bit Mean * 100 (16-LSBs) |

| 32-bit Mean * 100 (16-MSBs) |

| 32-bit Std_Dev * 100 (16-LSBs) |

| 32-bit Std_Dev * 100 (16-MSBs) |

...

| PFX_END_OF_FRAME |

header

pedestal histogram short statistics of one channel
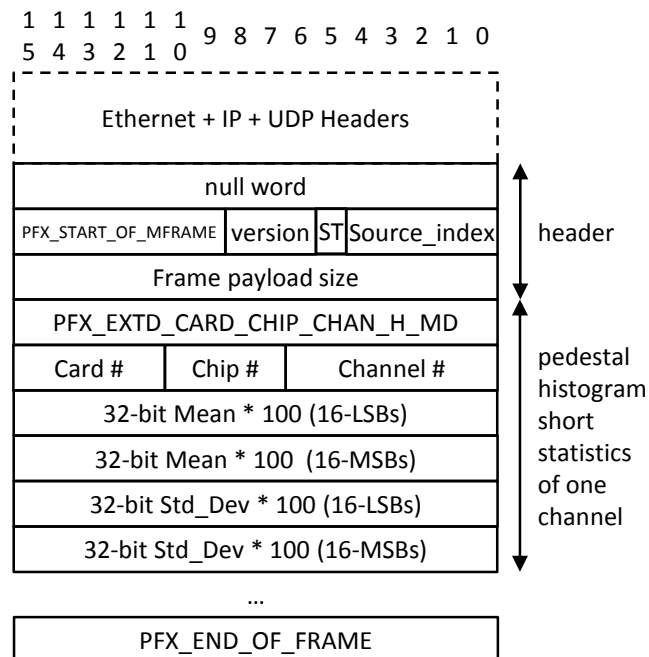
Fig. 89.   Pedestal histogram condensed statistics frame. Extended format.

9.4.2 DEAD-TIME HISTOGRAMS

The TDCM activates its BUSY pin when it receives a valid trigger and releases it when all the active front-end cards have returned a message indicating that they are ready to

capture the next event. The duration of the BUSY signal represents the dead-time of the readout system. The dead-time measured for each event is accumulated in a 1024-bin, 32-bit amplitude count histogram. Resolution is programmable among four values: 1 µs, 10 µs, 100 µs and 1 ms. The measurement range scales accordingly (i.e. 1.022 ms, 10.22 ms, 102.2 ms or 1.022 s. For all resolution settings, the last bin (bin #1023) accumulates overflows. The frame payload size is the size in bytes counted from the START_OF_MFRAME prefix to END_OF_FRAME (including both of them). The format of this histogram is shown in Fig. 90.

| 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 | 9 8 7 6 5 4 3 2 1 0 |
| 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 | |

Ethernet + IP + UDP Headers				
null word				
Frame payload size	PFX_START_OF_MFRAME	version	ST	Source_index
null word	PFX_DEADTIME_HSTAT_BINS			
Min_Bin (16-MSBs)	Min_Bin (16-LSBs)			
Max_Bin (16-MSBs)	Max_Bin (16-LSBs)			
Bin_Width (16-MSBs)	Bin_Width (16-LSBs)			
Bin_Count (16-MSBs)	Bin_Count (16-LSBs)			
Min_Non_Null_Bin(16-MSBs)	Min_Non_Null_Bin (16-LSBs)			
Max_Non_Null_Bin (16-MSBs)	Max_Non_Null_Bin (16-LSBs)			
Mean*100 (16-MSBs)	Mean*100 (16-LSBs)			
Std_Dev * 100 (16-MSBs)	Std_Dev * 100 (16-LSBs)			
Entry_Count (16-MSBs)	Entry_Count (16-LSBs)			
Bin Value (16-LSBs)	0010	10-bit Bin Index		
	Bin Value (16-MSBs)			
…				
	PFX_END_OF_FRAME			

header (rows: Frame payload size through PFX_DEADTIME_HSTAT_BINS)
latency histogram statistics (Min_Bin through Entry_Count)
for all non-null bins (Bin Value rows)

Fig. 90.   Dead-time histogram frame.

9.4.3 INTER-EVENT TIME HISTOGRAM

For each event, starting from the second one, the TDCM measures the interval of time between the current trigger and the next one. This value in accumulated in a 32-bit amplitude, 1024-bin histogram. The resolution of each bin is fixed to 100 µs leading to a measurement range of 0 to 102.2 ms. The last time bin accumulates the overflows. The format of this histogram is identical to that of the dead-time histogram except that the prefix word in the header is PFX_EVPERIOD_HSTAT_BINS instead of PFX_DEADTIME_HSTAT_BINS. The average value of this histogram represents the average event taking rate of the system.

9.4.4 COMMAND STATISTICS

The TDCM accumulates the number of configuration and monitoring command received, number of errors (syntax error in the command or execution failure) and number of command replies. Also, the number of "daq" requests received and served is counted separately. The structure of the monitoring frame that contains the commands statistics counters is show in Fig. 91. The frame payload size is the size in bytes counted from the START_OF_MFRAME prefix to END_OF_FRAME (including both of them).

| 1 1 1 1 1 1 9 8 7 6 5 4 3 2 1 0 |
| 5 4 3 2 1 0 |

Ethernet + IP + UDP Headers	
null word	header
PFX_START_OF_MFRAME | version | ST | Source_index	
Frame payload size	
PFX_CMD_STATISTICS	command statistics counters
32-bit RX command count (16-LSBs)	
32-bit RX command count (16-MSBs)	
32-bit RX daq request count (16-LSBs)	
32-bit RX daq request count (16-MSBs)	
32-bit RX daq timeout count (16-MSBs)	
32-bit RX daq timeout count (16-LSBs)	
32-bit RX daq delayed count (16-LSBs)	
32-bit RX daq delayed count (16-MSBs)	
32-bit RX daq missing count (16-LSBs)	
32-bit RX daq missing count (16-MSBs)	
32-bit RX cmd error count (16-LSBs)	
32-bit RX cmd error count (16-MSBs)	
32-bit command reply count (16-LSBs)	
32-bit command reply count (16-MSBs)	
32-bit daq reply count (16-LSBs)	
32-bit daq reply count (16-MSBs)	
32-bit daq resend count (16-LSBs)	
32-bit daq resend count (16-MSBs)	
PFX_END_OF_FRAME	

Fig. 91.   Command statistics counter frame.

9.4.5 PER ASIC CHANNEL HIT COUNT HISTOGRAM

Some types of front-end cards, e.g. the ARC, accumulate for each of the 4 ASICs controlled the histogram of channel hit count per event. These histograms can be

retrieved by the TDCM and forwarded to the DAQ PC. The structure of the monitoring frame that contains the histogram of the number of channels hit in one ASIC of one front-end is show in Fig. 92. The frame payload size is the size in bytes counted from the START_OF_MFRAME prefix to END_OF_FRAME (including both of them). The index of the chip is indicated after the appropriate prefix and the index of the front-end is placed in the following 16-bit word.
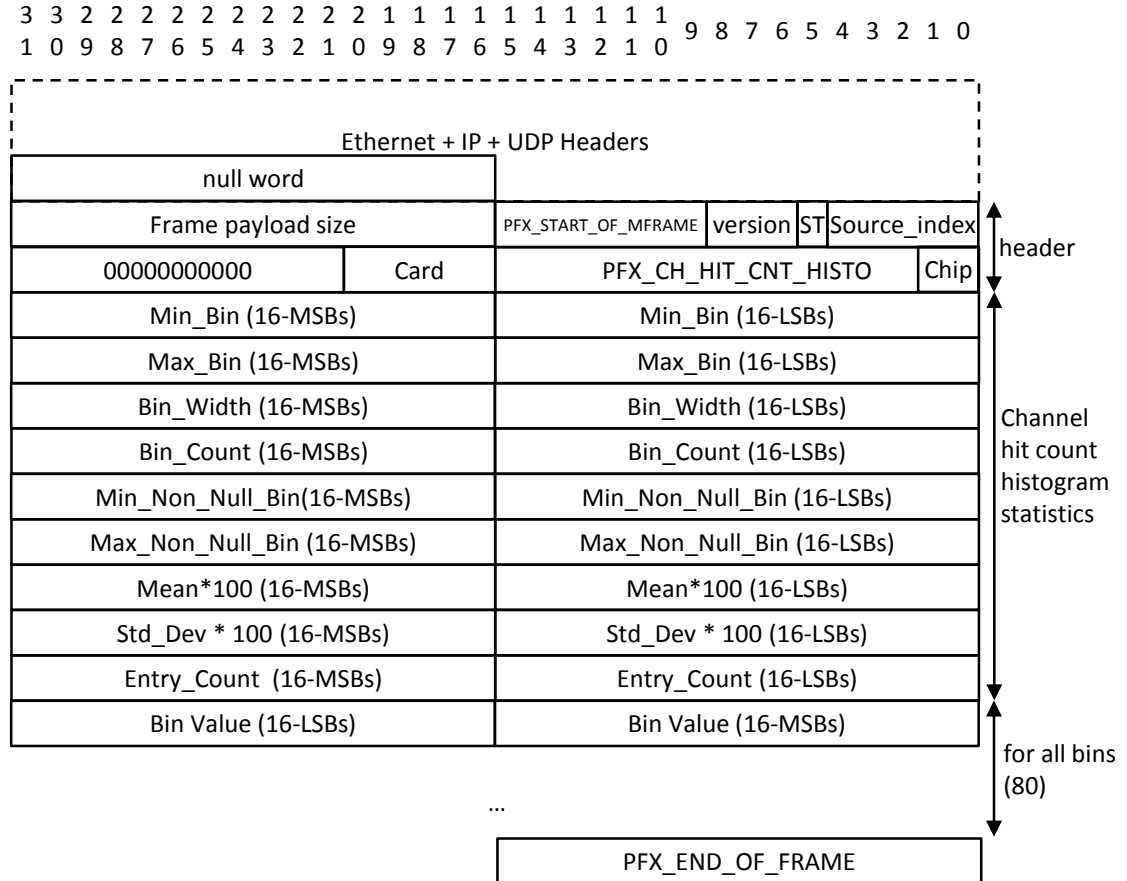
Fig. 92.   Channel hit count histogram frame

9.4.6 LIST OF PEDESTAL EQUALIZATION CONSTANTS AND ZERO SUPPRESSION THRESHOLDS

Assuming that a pedestal run has been performed and that the appropriate commands to compute pedestal equalization constants and thresholds for zero-suppression have been executed, the TDCM can send to the DAQ PC the list of pedestal equalization constants and thresholds that it has computed. The format of the corresponding frame is given in Fig. 93.

119
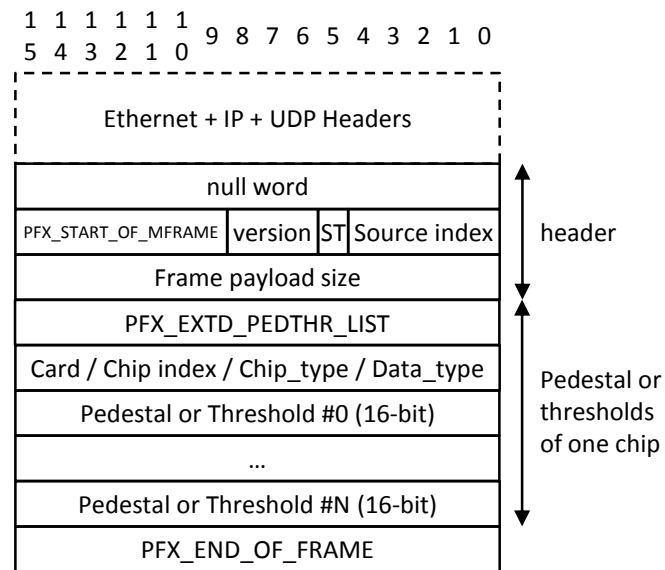
Fig. 93.   Pedestal or threshold list.

The next 16-bit word after the prefix PFX_EXTD_PEDTHR_LIST contains from MSB to LSB: 5 unaffected bits, the index of the front-end (5 bits), the index of the chip (4 bits), the type of chip (1 bit: 0 for AGET and 1 for AFTER), the type of data in the list (1 bit: 0 for pedestals and 1 for thresholds). It is followed by a fixed number of 16-bit values, 72 in the AGET mode and 79 in the AFTER mode, where each value represents a pedestal equalization constant or threshold. Pedestal equalization constants are signed short integers while thresholds are unsigned short integers.

9.4.7 BIT ERROR RATE TESTER STATISTICS

The TDCM implements a bit error rate tester for measuring the stability of the communication links with the FEs. In the TDCM to FE direction, each FE compares the received pattern from the TDCM with the expected locally generated reference pattern to determine if bits were altered during transmission. Periodically, e.g. every 10 million bits have been received, the FE sends to the TDCM over Virtual Channel C a specific packet that contains the number of receive errors and the number of millions of received bits. The format of this packet is shown in Fig. 94.

```
1 1 1 1 1 1
5 4 3 2 1 0  9 8 7 6 5 4 3 2 1 0
```

Protocol Header (FE) Previous packet in same Ethernet frame or protocol header (TDCM)	FE: mandatory TDCM: case dependent
PFX_BERT_STAT · ST · SOURCE ID	
Bit Error Count (16-bit)	Always present
Received bit (in millions) (16-LSBs)	
Received bit (in millions) (16-MSBs)	
Protocol Trailer (FE) Following packet in same Ethernet frame or protocol trailer (TDCM)	FE: mandatory TDCM: case dependent

Fig. 94.   Bit Error Rate Tester statistics packet.

10 COMMAND SERVER REFERENCE

The tcm_server application is a command server program running on the embedded processor of the TDCM. This server processes the commands received from a remote DAQ PC over a standard Ethernet connection and returns results via the same path. All commands are formatted in ASCII and perform basic to complex operations on the TDCM itself, or one or several front-ends via the appropriate virtual channel of the physical distribution links. Commands can be classified into three categories: 1) configuration commands, 2) monitoring and run control commands, 3) data acquisition commands. A distinction must also be made in the scope of the commands: some apply to the TDCM, some apply to one of several FEs, and several of them control the behavior of the client program running on the control PC. Most of the commands that apply to the TDCM are prefixed with the string "be ", meaning back-end, while the commands that apply to the FEs are prefixed with the string "fe ", meaning front-end. A command for the front-end can be sent to a single FE, or it can be duplicated within the TDCM and sent to a subset of FEs, or all of them.

The TDCM uses only one UDP/IP socket when the client program running on the DAQ PC performs all the tasks of system configuration, monitoring, run control and data acquisition. The TDCM uses two UDP/IP sockets when different programs, possibly running on different hosts, are used for the tasks mentioned above. In this case, one socket is dedicated to the data acquisition path while the second one handles other types of commands.

10.1 COMMANDS THAT APPLY TO THE TDCM

Important note: Almost all the commands that are directed to the TDCM start with the prefix "be". This prefix can optionally be followed by the placeholder character ".", surrounded by one space character before and after, or by the index of the TDCM to which the command shall apply. This optional argument after the prefix "be" is intended to facilitate the control of multiple TDCMs from the same client program. This feature is not implemented yet and the TDCM will execute all the commands prefixed with "be" that it receives, independently of the placeholder character or index that is supplied in the command.

10.1.1 GENERAL CONTROL OF THE TDCM

The general control commands that act on the TDCM are listed in Table 27.

Table 27. TDCM general control commands.

Command	Argument	Action
be version		Returns major/minor software version and compilation date of the TDCM embedded software

version		Same as "be version" but also prints the version of the client software running on the control PC
be cmd	clr	Clears command sent/replies counters
be cmd	stat	Shows command statistics counters
be reg	<r>	Reads TDCM register <r>
be reg	<r> <0xdata>	Writes TDCM register <r> with hexadecimal data <0xdata>
be fe_workset	<0xSet>	Defines the work set of FEs
be fe_workset		Shows the current work set of FEs
be sel_fe		Shows to which FE(s) the interpreter commands currently apply
be sel_fe	<*|fe_workset| 0xSet|index>	Sets to which FE(s) subsequent relevant interpreter commands will be directed.

The command "be fe_workset <0xSet>" is normally used at startup to define which FEs are involved in the system. The argument 0xSet is a 32-bit unsigned integer where each bit indicates that the corresponding FE is part of the current system. The work set of FEs is normally set once and is not modified during runtime.

The command "be sel_fe …" defines to which FE(s) subsequent interpreter commands may apply. The "*" wildcard can be used to selects all FEs (even non-existing ones), but the user will normally use the argument "fe_workset" which corresponds to all FEs in the system. Alternatively, a subset of FEs can be selected using an argument with the hexadecimal prefix "0x": multiple FEs are selected according to which bit is set in binary representation of this 32-bit argument. To select only one particular FE, it is also possible to supply the argument in decimal format, in the interval [0; 31].

10.1.2 CONTROL OF COMMUNICATION LINKS WITH FRONT-END

The commands in Table 28 are used to configure and monitor the communication links with the front-end cards.

Table 28. Commands related to communication links with FEs.

Command	Argument	Action
be tx_reset		Shows the state of the reset signal of the TDCM to FE fanout link
be tx_reset	<0 | 1>	Sets or releases the reset signal of the TDCM to FE fanout link. This applies to all TDCM to FE links.
be rx_reset	<fe_workset|port_id> <0 | 1 | cycle>	Sets or releases the reset signal for the RX side of the desired FE to TDCM link port. When the keyword fe_workset is supplied, this command is looped for every active port. When the keyword cycle is supplied, a '1' followed by a '0'

		are written. Note that the value set cannot be read-back.
be inv_tdcm_clk		Shows if the primary clock sent by the TDCM is send with inversion or not to the FEs
be inv_tdcm_clk	<0 | 1>	Sets the optional inversion of the primary clock before it is fanout to all FEs. This command is only relevant when copper links are used between the TDCM and FEs.
be inv_tdcm_mosi		Shows if the serial data sent by the TDCM to all FEs is inverted or not before being fanout.
be inv_tdcm_mosi	<0 | 1>	Sets the optional inversion of the serial data sent by the TDCM to all FEs. This command is relevant whether the links between the TDCM and the FEs are copper or optical.
be dcbal_enc		Shows if DC-balanced encoding is enabled for the TDCM to FE fanout links.
be dcbal_enc	<0 | 1>	Sets DC-balanced encoding for the TDCM to FE fanout links. This must be enabled when optical media is used and is optional with copper media.
be tx_bert_ena		Shows if the bit error rate tester for the TDCM to FE link direction is enabled or disabled in the TDCM
be tx_bert_ena	<0 | 1>	Enables or disables in the TDCM side the bit error rate tester in the TDCM to FE link direction.
be tx_bert_pat		Shows the pattern used by the TDCM for the bit error rate tester in the TDCM to FE link direction

be tx_bert_pat	<0, 1, 2, 3>	Sets the pattern used by the TDCM for the bit error rate tester in the TDCM to FE link direction. Argument:
		    0: PRBS7
		    1: PRBS15
		    2: PRBS23
		    3: PRBS31

be tx_bert_doerr		Force the generation of a single bit error on the bit error rate tester in the TDCM to FE link direction.
be tx_rate		Shows the speed of the transmission in the TDCM to FE link direction

be tx_rate		Sets the speed of the transmission in the TDCM to FE link direction. Possible values: 0: 100 Mbps (normal operation) 1: 200 Mbps (for BER test only) 2: 400 Mbps (for BER test only) 3: not supported (defaults to 400 Mbps)
be rx_bert_ena		Shows if the bit error rate tester for the FE to TDCM link direction is enabled or disabled in the TDCM
be rx_bert_ena	<0 | 1>	Enables or disables in the TDCM the bit error rate tester for the FE to TDCM link direction
be rx_bert_pat		Shows the pattern expected by the TDCM for the bit error rate tester in the FE to TDCM link direction.
be rx_bert_pat	<0, 1, 2, 3>	Sets the pattern expected by the TDCM for the bit error rate tester in the FE to TDCM link direction. Argument: 0: PRBS7 1: PRBS15 2: PRBS23 3: PRBS31
be rx_bert_stat	<port>	Shows the statistics of the bit error rate tester in the FE to TDCM link direction for the selected RX port.
be crc32_check_ena		Shows if CRC32 verification is enabled for data packets received from the FEs by the TDCM over Virtual Channel C.
be crc32_check_ena	<0 | 1>	Enables or disables CRC32 verification for data packets received from the FEs by the TDCM over Virtual Channel C.
be crc32_inject_err		Force the generation of a CRC32 error on a data packet received from a FE.

10.1.3 COMMANDS FOR THE ASSIGNMENT OF INDEXES TO THE FES

In order to communicate with FEs after all communication links have been synchronized in both directions, the TDCM must assign to each FE the index that corresponds to the physical port where it is connected. In a first step, the TDCM reads the serial DNA number of each FE and builds the table that makes the correspondence between DNA numbers and port indexes. In a second step, the TDCM sends this correspondence table to all FEs so that each FE can determine its own index when the table entry that matches its DNA number is encountered. The commands related to that identification and enumeration process are listed in Table 29.

Table 29. Commands for FEC identification and enumeration.

Command	Argument	Action
be dna	get	Retrieves the DNA number of all the FEs detected by the TDCM
be dna	push	Multicast to all FEs all the DNA numbers received by the TDCM and the card index associated to each of them

At system startup, after all links have been synchronized, the TDCM must perform a "dna get" followed by a "dna push" to enumerate all FEs and assign them the index that correspond to the physical port of the TDCM where each of them is connected. It is also recommended to perform a second "dna get" after "dna push" to verify that each FE is assigned the correct index. Whenever a FE is power-cycled, the complete enumeration must be re-done to detect the new FE and assign it the correct index.

10.1.4 MESSAGE COUNTERS

The commands in Table 30 are used to show the counters of message received/transmitted from/to the FEs. These message counters are maintained on the TDCM side.

Table 30. Commands related to message counters.

Command	Argument	Action
be port	<* | first:last | id> <* | a | b | c> clr	Clears the message counter(s) of the selected port(s) and virtual channel(s). The first argument can specify all ports, a range of ports (it must be in [0;31]), or a specific port. The second argument can specify all 3 virtual channels or only one of them. The third argument is fixed and mandatory
be port	<id> <a | b | c>	Shows the message counter of the selected port and virtual channel. Contrary to the counter clear command, the counter read command only accepts one port and one virtual channel at a time.

10.1.5 FRONT-END PRESENCE AND STATE

The commands in Table 31 are used to set which FE are present and display their state as it is memorized in the TDCM. If the system loses synchronization, the actual state of a FE may be different from that currently read in the TDCM.

Table 31. Commands related to FEC presence and status.

Command	Argument	Action

be fe	active	Displays which FEs the TDCM supposedly controls.
be fe	active <fe_workset | 0xVal>	Sets which FEs are supposedly active in the system. The argument can be "fe_workset" or a 32-bit unsigned integer where each bit set to 1 corresponds to a FE with that index being present.
be fe	linkup	Displays which FEs are effectively detected, i.e. communication is currently established.
be fe	sampling	Displays which FEs are thought by the TDCM to be in the data sampling state
be fe	busy	Displays which FEs are thought by the TDCM to be in the busy state

Note that the internal variable that is set by the command "be fe active" is different from that set by the commands "be fe_workset" and "be sel_fe". The command "be fe_workset" should be used once at startup to define the complete set of FEs that compose the current system, while the command "fe active" and "be sel_fe" may select some subset of all the FEs comprised in the system. In practice, "be fe active" will normally take the reserved keyword "fe_workset" as an argument to specify that all the FEs of the system are active. Similarly, "be sel_fe" may take as an argument the keyword "fe_workset" if all FEs require identical configuration. Otherwise, the command "be sel_fe 0xSet" can be used to successively select different subsets of FEs and apply different configuration commands to each subset.

10.1.6   SYNCHRONOUS SIGNAL FANOUT

The commands in Table 32 can be used to send synchronously to all FEs a message over Virtual Channel A. These commands are for test only and some values must not be used during normal operation because these messages are normally automatically generated by the firmware part of the TDCM without the intervention of the user.

Table 32. Commands to send synchronous messages to all FECs.

Command	Argument	Action
be isobus	<0xVal>	Sends the message <0xVal> to all FEs synchronously. Argument <0xVal> is a 8-bit hexadecimal value obtained by a combination of: Bit 7 (MSB): reserved Bit 6: WCK_SYNCH (synchronize write clock phase) Bit 5: SCA_START (acquisition start or restart) Bit 4: SCA_STOP (acquisition stop, a.k.a. trigger) Bit 3: CLR_EVCNT (clear event counter) Bit 2: CLR_TSTAMP (clear timestamp counter) BIT 1-0: EV_TYPE (event type)

10.1.7   TDCM FINITE STATE MACHINES

The commands in Table 33 are used to display the state of several of the state machines driving the TDCM.

Table 33. Commands related to TDCM finite state machines.

Command	Argument	Action
be state	tg	Displays the current state of the trigger generator of the TDCM
be state	eb	Displays the current state of the local event builder
be state	pm	Displays the current state of the data packet mover
be restart		Clears all pending error on the trigger generator and restarts its operation
be max_readout_time		Shows the maximum allowable event readout time.
be max_readout_time	\<time>	Sets the maximum allowable event readout time. The value \<time> is an integer between 0 and 16 and represents an actual time of 1 s to 17 s.

The maximum event readout time is the allowable time from the reception of the trigger acknowledge from every FEC until all FEs have returned their own clear busy message. If a FE is still presumably in the busy state for readout and the allowable maximum readout time has elapsed, the TDCM enters an error state and stops data acquisition until this error is cleared.

10.1.8 TRIGGER GENERATOR AND TRIGGER CONTROL

The commands in Table 34 are used to control the trigger and trigger generator of the TDCM.

Table 34. Trigger generator and trigger control.

Command	Argument	Action
be trig_rate		Shows the current trigger rate of the embedded constant interval trigger generator
be trig_rate	\<range> \<rate>	Sets the rate of the embedded trigger generator. Four ranges are available:
		0: 0.1 Hz to 10 Hz in steps of 0.1 Hz
		1: 10 Hz to 1000 Hz in steps of 10 Hz
		2: 100 Hz to 10 kHz in steps of 100 Hz
		3: 1 kHz to 100 kHz in steps of 1 kHz
		The argument \<rate> is from 1 to 100 for each possible range.
be event_limit		Shows the current event limit of the embedded event generator

be event_limit	<limit>	Sets the event limit of the embedded event generator. Eight settings are supported:
		    0: infinity
		    1: 1 event
		    2: 10 events
		    3: 100 events
		    4: 1000 events
		    5: 10.000 events
		    6: 100.000 events
		    7: 1.000.000 events.
		The embedded event generator will stop producing new triggers when the allowable limit is reached.

be event	rx	Shows the number of valid triggers received by the TDCM
be event	tx	Shows the number of triggers sent by the TDCM to the FECs. The value can be less than the number of trigger received if system dead-time is excessive.
be trig_delay	<type>	Displays the current value of the trigger latency setting for the type of event specified. Four types of event are supported (0, 1, 2, 3).
be trig_delay	<type> <delay>	Sets the trigger latency to <delay> for the type of event specified. Trigger delay is an unsigned 16-bit integer value that specifies the delay in units of 10 ns. The maximum trigger delay is 655.35 µs.
be trig_ena		Shows which trigger sources are enabled in the TDCM

be trig_ena	<0xTrig_mask>	Sets which trigger sources are enabled in the TDCM. Parameter <0xTrig_mask> is an hexadecimal 4 –bit combination of:
		    Bit 0: embedded periodic generator
		    Bit 1: NIM level trigger input
		    Bit 2: MTCM serial trigger input
		    Bit 3: TTL level trigger input

be ss_trig_ena		Shows if the Single Shot trigger mode is enabled or disabled
be ss_trig_ena	<0 | 1>	Disables or enables the single shot trigger mode
be ss_trig_delay		Shows the trigger delay for the Single Shot Trigger mode
be ss_trig_delay	<Delay>	Sets the trigger delay for the Single Shot Trigger mode. <Delay> must be in [1; 255] corresponding to a delay of 10 µs to 25.5 ms in steps of 10 µs.

be mult_trig_ena		Shows if the trigger based on multiplicity is enabled or not
be mult_trig_ena	<0 |1>	Enables or disables the trigger based on multiplicity.
be mult_trig_dst		Shows the target of the trigger based on multiplicity
be mult_trig_dst	<0 | 1>	Sets the target of the trigger based on multiplicity. 0: send to FECs directly; 1: send to external device
be mult_more_than		Shows the lower bound of the window comparator that elaborates the trigger based on multiplicity signals
be mult_more_than	<low_mult>	Sets the lower bound of the window comparator that elaborates the trigger based on multiplicity signals. Parameter <low_mult> is an integer in [0, 127]
be mult_less_than		Shows the upper bound of the window comparator that elaborates the trigger based on multiplicity signals
be mult_less_than	<high_mult>	Sets the upper bound of the window comparator that elaborates the trigger based on multiplicity signals. Parameter <high_mult> is an integer in [0, 127]
be extra_dead_time		Shows the value of the extra dead-time
be extra_dead_time	<dead_time>	Sets the extra dead-time. The argument <dead-time> is in [0; 1023] and is expressed in µs.

The TDCM comprises an embedded periodic trigger generator with programmable frequency and maximum event count. This generator is useful for system debugging and performance evaluation. Others sources of trigger are: the TTL level trigger input, the NIM level trigger input, the serial trigger from the MTCM, the self-trigger based on multiplicity. Multiple sources of trigger may be enabled when the TDCM acts as a master device, but the TDCM shall only enable triggers from the MTCM when it is a slave device. When a trigger occurs, the latency programmed in TRIG_LAT_x is applied, according to the 4 possible types of event. Currently, the periodic generator, the TTL and NIM trigger inputs, the multiplicity self-trigger and the manual push button trigger generate events of type "11" – this may be changed in the future. When the trigger comes from the MTCM, the type of event is provided by the MTCM.

The TDCM has also a special mode of self-trigger called the "Single Shot Trigger". This mode is intended to be used when the TDCM is a master device and the calibration pulser is enabled on the front-end side. When SCA_START occurs (manual push button START, or "isobus" software command with the appropriate bit set), the TDCM elapses

the latency programmed in SS_TRIG_LAT and finally generates a trigger of type "10". This trigger is further delayed by the value set in TRIG_LAT_2. After this Single Shot trigger has been generated, SCA_START must be generated by the user to produce a new event. On the contrary, when the TDCM is used in standalone master mode with the TTL, NIM or periodic trigger input, SCA_START need only be generated once at the start of run by the user. When a trigger occurs, the TDCM generates automatically the required SCA_START to resume operation after all front-ends have signaled that they are ready. When the TDCM is a slave device, all SCA_START commands come from the MTCM.

The TDCM can add some additional dead-time after SCA write in all FE have been restarted and before its BUSY pin is released (and before it sends CLEAR_BUSY to the M-TCM when it is used). This additional dead-time can be set from 0 to 1.023 ms.

10.1.9 DATA PUMP

The commands in Table 35 are used to control the "Data Pump" embedded in the TDCM.

Table 35. Commands related to the Data Pump.

Command	Argument	Action
be pump	ena	Displays from which subset of FEs the Data Pump tries to gather event data
be pump	ena <fe_workset |0xPat>	Sets from which FEs the Data Pump have to gather event data. The argument can be the keyword "fe_workset" to select all declared FEs or a 32-bit hexadecimal value <0xPat> where a 1 is set to every position that correspond to an active FE.
be pump	stalled	Shows for which subset of FEs the Data Pump is in the stalled state
be pump	running	Shows for which subset of FEs the Data Pump is in the running state
be pump	timed	Shows if the Data Pump operates in time-out mode or not.
be pump	timed <0 | 1>	Sets the Data Pump for operation in time-out mode or not
be pump	timeout	Shows the value of the time-out for the Data Pump
be pump	timeout <Val>	Sets the value of the time-out for the Data Pump. Argument: 　　0: 1 ms 　　1: 10 ms 　　2: 100 ms 　　3: 1 s

10.1.10 EVENT BUILDER AND PACKET FRAMER

The commands in Table 36 are used to control the Event Builder and Packet Framer embedded in the TDCM.

Table 36. Commands related to the Event Builder.

Command	Argument	Action
be eb	run	Shows if the Event Builder is enabled or disabled
be eb	run <0 | 1>	Enables or disables the Event Builder.
be eb	keep_fem_soe	Shows if the Event Builder keeps or discards the Start Of Event packet sent by each FE
be eb	keep_fem_soe <0 | 1>	Sets whether the Event Builder keeps or discards the Start Of Event packet sent by each FE
be eb	check_ev_nb	Shows if the Event Builder verifies or ignores if the event number matches across all FEs at each event
be eb	check_ev_nb <0 | 1>	Sets whether the Event Builder verifies or ignores if the event number matches across all FEs at each event
be eb	check_ev_ts	Shows if the Event Builder verifies or ignores if the event timestamp matches across all FEs at each event
be eb	check_ev_ts <0 | 1>	Sets whether the Event Builder verifies or ignores if the event timestamp matches across all FEs at each event
be eb	ts_tolerance	Shows what mismatch is allowable between event timestamps when verification is enabled
be eb	ts_tolerance <Tol>	Sets the allowable mismatch between event timestamps when verification is enabled. Parameter: 0: exact match required 1: match required within -1 to +1 units 2: match required within -2 to +2 units 3: match required within -4 to +4 units
be serve_target		Shows to which target the output of the Packet Framer is directed
be serve_target	<Target>	Sets to which target the output of the Packet Framer is directed. Parameter: 0: null device (drop data) 1: remote DAQ PC (normal operation) 2: locally accumulated pedestal histograms 3: locally accumulated hit count histograms

10.1.11   COMMANDS FOR DATA ACQUISITION

The commands listed in Table 37 are used by the DAQ PC to request acquired event data from the TDCM. Event data transfer from the TDCM to the DAQ PC uses a pull protocol with a credit mechanism that provides flow control. This mechanism is mandatory to avoid data losses because the TDCM and the DAQ PC communicates over

UDP/IP, which is not a reliable protocol. These commands are not typed by the user when the "pclient" program is used for data acquisition but they must be used by any other client program that performs the same function. In fact, the "pclient" program converts transparently the "DAQ …" commands used at the user level into series of "daq …" commands at the lower level of communication between the TDCM and the DAQ PC.

In order for the TDCM to send the event data gathered by its event builder and Ethernet packet framer, it must receive some send credits from the DAQ PC. This is accomplished by the command "daq <0xCredit F>" where the argument 0xCredit specifies the number of send credits. This argument is followed by the unit "F" which means Ethernet frames. Early versions of the TDCM embedded software also supported the unit "B" meaning bytes, but this is now deprecated and only credits expressed in a number of Ethernet frames are acceptable. The credit argument must be specified in hexadecimal with exactly 6 digits. For example, the command "daq 0x000008 F" instructs the TDCM that it is allowed to send up to 8 Ethernet data frames (of any size up to the MTU) to the DAQ PC. Every time the TDCM sends one data frame to the DAQ PC, it decrements its local credit count. When this local credit count reaches zero, the TDCM is not allowed to send any more data to the DAQ PC until it receives back some credits via the appropriate "daq …" command. The DAQ PC should not wait until the allowable number of frames has been received to send back some credits to the TDCM because starvation on the TDCM side may occur, and this reduces data acquisition throughput. On the opposite, the DAQ PC need not return immediately every single credit to the TDCM because it may be inefficient to send one "daq …" command for every received data frame. It is recommended that the DAQ PC accumulates a certain of credits (e.g. 4) before they are returned at once to the TDCM. The initial number of credits determines the total maximum number of Ethernet data frames that are allowed to flow at any time in the system. The optimal number depends on the buffering capability of the DAQ PC (network interface card and device driver) and network latency. It is assumed that the DAQ PC and the TDCM are connected directly by a point-to-point Ethernet cable, but in case multiple TDCMs are controlled from the same Ethernet adapter of the DAQ PC, an Ethernet switch will be needed in between. In this case, the buffering capacity of the Ethernet switch and its latency also play some role. If the total frame credit count is too high, some element in the chain may overflow, causing data losses. If the initial number of credits is too low, data acquisition throughput will be reduced because the TDCM will have to wait for returned credits before it can transmit more data. From experience, a total credit count of 8 Ethernet frames is a good tradeoff that allows to reach the highest transfer efficiency without any data loss in network elements caused by capacity overflow.

Table 37. Commands for data acquisition.

Command	Argument	Action
(be)* daq		Shows the current credit count of the TDCM for sending data to the DAQ PC
(be)* daq	<0xCredit> F <Seq_nb>	Gives some amount of send credits to the TDCM. An optional sequence number may be added
(be)* daq	0xFFFFFF F	Clears the send credit count of the TDCM
*note: the prefix "be" must not be present, but this may be changed in future software releases		

When the "daq" command is supplied without any argument, the TDCM returns a message that contains a copy of the current value of its credit counter. If data acquisition is in progress, this number will vary. If data acquisition is currently stopped, the returned value can be added to the current credit counter of the DAQ PC to verify that the total number of credits corresponds to the value injected at startup. If it is lower, some losses have occurred. Note that lost frames are currently not retransmitted, but in stable system, such losses are extremely rare.

In some cases, it is desired to clear the send credit count of the TDCM. The command "daq 0xFFFFFF F" is used for this purpose. In fact, it is recommended to clear the frame send credit count of the TDCM at the beginning of each data taking run and re-initialize it to the desired value. Care must be taken not to inject new credits in the system without purging properly all the credits that may remain from the previous run, because an excessive number of credits will most likely cause some frame losses at a later time. For increased configurability and easier performance tuning, it is recommended that the total number of send credits and the threshold for returning freed credits to the TDCM are made configurable at run-time without the need to recompile any software.

In order to detect the loss of data frames and returned credits, the TDCM and the DAQ PC may optionally place a sequence number in each data frame and daq command respectively. If this functionality is used, the first daq command sent by the DAQ PC to the TDCM should not include any sequence number. This instructs the TDCM to clear locally the value expected for the sequence number that will be contained in the next daq request. On the second daq request sent by the DAQ PC to the TDCM, the sequence number argument shall be present, and it must be 0x00 in this case. Then, the sequence number must be incremented by one on every sub-sequent daq command. When reaching 0xFF, the index wraps around and the next daq command will have the sequence number 0x00. Sending a daq command without adding the sequence number argument clears to 0x00 the value expected by the TDCM for the sequence number of the next daq command. When a daq command contains a sequence number, the TDCM compares the received sequence number to the expected value fetched from its own

local counter. In case of mismatch, the TDCM computes how many daq command frames were presumably lost. It updates a local error counter but it does not attempt to recover lost commands.

Each data frame sent the TDCM to the DAQ PC may also carry a sequence number. It is placed in the first 16-bit payload word of a data frame, just after the UDP/IP header. This protocol word is set to 0x0000 when the sequence number functionality is not used, see Fig. 77 (b). The first data frame sent after receiving a daq command that does not contain the sequence number argument always contains 0x0100 in the first payload word. When receiving this value, the DAQ PC shall set to 0x01 the value it expects for the next response frame from the TDCM. Subsequent data frames sent by the TDCM have in the first 16-bit word a value 0x00 in the 8-MSBs and an incrementing sequence number in the 8-LSBs. When the sequence number reaches 0xFF, it wraps around and the next sequence number will be 0x00. By comparing the expected sequence number to the actual sequence number received, the DAQ PC can determine if some of the data frames sent by the TDCM were lost. This operation is only for error monitoring. It does not attempt to recover lost data. Note that the sequence number in the DAQ PC to TDCM direction is different from that used in the opposite direction. Each end must maintain a separate counter for the two types of sequence numbers. The synchronization procedure is however identical for both counters: it is accomplished by sending to the TDCM a daq command without the sequence number argument.

10.1.12   CONTROL OF THE RING BUFFER OF THE TDCM

The commands that control the operation of the ring buffer of the TDCM are listed in Table 38.

Table 38. TDCM ring buffer commands.

Command	Argument	Action
be rbf	reset	Resets the ring buffer
be rbf	resume	Starts or resumes the operation of the ring buffer
be rbf	suspend	Suspends the operation of the ring buffer
be rbf	getpnd	Returns any pending buffer immediately, even if it partially filled with data
be rbf	timed	Shows if the ring buffer runs in time-out mode or not
be rbf	timed <0 |1>	Sets the time-out mode for the ring buffer
be rbf	timeval	Shows the value of the time-out of the ring buffer
be rbf	timeval <val>	Sets the value of the time-out for the ring buffer. Possible settings: 0:1 ms; 1:10 ms; 2:100 ms; 3:1 s.
be rbf	config	Reads the current configuration parameters of the ring buffer

10.1.13   COMMANDS RELATED TO COMMUNICATION WITH THE MASTER TCM

The commands in Table 39 are used to control the interface of the TDCM to the M-TCM.

The commands inv_mtcm_mosi and inv_mtcm_miso are used to optionally invert the serial data sent to / received from the M-TCM. These can correct differential signal pair inversion.

Table 39. Commands related to communication with the Master TCM.

Command	Argument	Action
be mtcm inv_mosi		Displays the current state of the inversion for the serial data received by the TDCM from the M-TCM
be mtcm inv_mosi	<0|1>	Sets the inversion of the serial data received by the TDCM from the M-TCM. 0: no data inversion. 1: invert data.
be mtcm inv_miso		Displays the current state of the inversion for the serial data sent by the TDCM to the M-TCM
be mtcm inv_miso	<0|1>	Sets the inversion of the serial data sent by the TDCM to the M-TCM. 0: no data inversion. 1: invert data.
be mtcm bert_pat		Displays the PRBS pattern used in bit error rate tester mode
be mtcm bert_pat	<0|1|2|3>	Sets the PRBS pattern used in bit error rate tester mode. 0: PRBS7; 1: PRBS15; 2: PRBS23; 3: PRBS31.
be mtcm_bert_ena		Shows if the bit error rate function is engaged or not
be mtcm_bert_ena	<0|1>	Enables or disables the bit error rate tester mode. 0: disabled; 1: enabled.
be mtcm_bert_rxen		Shows if the receive side of the bit error rate tester is running or not
be mtcm_bert_rxen	<0|1>	Starts or stops the receive side of the bit error rate tester. 0: stopped; 1: started (does not mean that synchronization is gained)
be mtcm_bert_txen		Shows if the transmit side of the bit error rate tester is running or not
be mtcm_bert_txen	<0|1>	Starts or stops the transmit side of the bit error rate tester. 0: stopped; 1: started
be mtcm bert_doerr		Generate a single bit error on the transmitter side
be mtcm mosi_sel		Shows which sample is used to deserialize the signal MTCM_MOSI
be mtcm mosi_sel	<0|1|2|3>	Sets which sample is used to deserialize the signal MTCM_MOSI. 0: t+0 ns; 1: t+2.5 ns; 2: t+5 ns; 3: t+7.5 ns
be mtcm cnt_clr		Clear the RX, RX error and TX counters of the MTCM interface

	be mtcm cnt_get	Displays the RX, RX error and TX counters of the MTCM interface

10.1.14 DEVICES CONTROLLED BY THE TDCM OVER I2C

The commands in Table 40 are used to drive the devices controlled by the TDCM over I2C. The TDCM uses the two I2C controllers of the ZYNQ. The devices to be controlled over I2C are: 1) on the FPGA module itself (real-time clock, secure EEPROM), 2) on the TDCM carrier or the Enclustra PE1 carrier (clock generators, SFP transceivers), 3) on the SFP extension mezzanine cards (parallel I/O ports for LED control and SFP transceivers).

Table 40. Commands for peripheral control over I2C.

Command	Argument	Action
be iic	mod info	Reads the information available on the FPGA module. Depends on which FPGA module is used.
be iic	mod power	Reads the current monitor of the FPGA module (if available)
be iic	mod clock	Reads the real time clock of the FPGA module (if available)
be iic	mod set clock <day month year hour min sec>	Sets the real time clock of the FPGA module (if available)
be iic	syscon	Reads the information on the system controller of the FPGA module host platform. Only available on Enclustra Mercury PE1 platform
be iic	msfp <ix> info	Retrieves various information from SFP transceiver plugged in port <ix> of a SFP Mezzanine Card. The index of the port can be in 0 to 31. The selection of the appropriate SFP Mezzanine Card and setting of the I2C multiplexer to address the appropriate transceiver is handled internally
be iic	msfp <ix> moni	Same as above, but retrieves various monitoring information from the selected SFP transceiver instead of fixed data.
be iic	csfp <ix> info	Retrieves various information from SFP transceiver plugged in port <ix> of the TDCM carrier. The index of the port can be in 0 to 3.
be iic	csfp <ix> moni	Same as above, but retrieves various monitoring information from the selected SFP transceiver instead of fixed data.
be iic	si5338 <reg>	Reads register <reg> of the on-board Si5338 PLL. This is only available on Enclustra Mercury PE1 platform
be iic	si5338 <reg> <val>	Writes register <reg> with value <val> in the on-board Si5338 PLL. This is only available on the

		Enclustra Mercury PE1 platform. Usage is reserved to expert users.
be fmc_iic		Shows the state (enabled or disabled) of the I2C interface of the FMC extension slot on Enclustra Mercury PE1 carrier card.
be fmc_iic	<0 | 1>	Enable or disable the I2C port of the FMC slot on the Enclustra Mercury PE1 carrier card
be sfp_mezz	<id> led	Shows the pattern of currently active LEDs on the front panel of the selected SFP Mezzanine Card. Parameter <id> selects SFP Mezzanine card 0 or 1.
be sfp_mezz	<id> led <0xPat>	Sets the front panel LEDs of SFP Mezzanine Card <id> to the value <0xPat>. Parameter <0xPat> is a 32-bit unsigned integer where each bit controls one LED on the SFP Mezzanine Card. When a bit is set to 1, the associated LED illuminates.
be sfp_mezz	<id> fault	Shows the state of the TX FAULT and/or RX_LOS pins of the optical transceivers of the selected SFP Mezzanine Card.
be sfp_mezz	<id> disable	Shows the state of the TX DISABLE pins of the optical transceivers of the selected SFP Mezzanine Card.
be sfp_mezz	<id> disable <0xDis>	Sets the state of the TX DISABLE pins of the optical transceivers of the selected SFP Mezzanine Card according to pattern <0xDis>. Argument <0xDis> is a 16-bit unsigned integer where setting a bit to 1 disables the TX part of the corresponding optical transceiver port.
be sfp_mezz	<id> enable <fe_workset | 0xEna>	Sets the state of the TX DISABLE pins of the optical transceivers of the selected SFP Mezzanine Card to the inverse of the current FE work set (i.e. do an enable for the current FE work set), or use the supply argument for the enable pattern. Argument <0xEna> is a 16-bit unsigned integer where setting a bit to 1 enables the TX part of the corresponding optical transceiver port
be sfp_mezz	<id> mux	Shows to which port the I2C multiplexer of selected SFP Mezzanine Card is set.
be sfp_mezz	<id> mux <port>	Sets the I2C multiplexer of selected SFP Mezzanine Card to the supplied port index. Argument <port> is within 0 to 15. This command is normally not used directly.

10.1.15   COMMANDS FOR THE EMBEDDED EVENT DATA GENERATOR

The commands in Table 41 are used control the event data generator embedded in the TDCM. This data generator is used to exercise the Event Builder and Packet Framer parts of the TDCM in the absence of a sufficient number of FEs, or without any FE at all.

Table 41. Commands for the embedded event data generator.

Command	Argument	Action
be eg	samp	Shows the number of data samples generated for each detector channel.
be eg	samp <Cnt>	Sets the number of data samples generated for each detector channel. This number corresponds to the number of SCA cells when using the AFTER, AGET or ASTRE chips. It can be set from 0 to 512 inclusive.
be eg	chan	Shows the number of channel per ASIC for the simulated FECs
be eg	chan <Cnt>	Sets the number of channel per ASIC for the simulated FEs. The allowable value is from 0 to 127 inclusive. The value 76 or 79 corresponds to the AFTER chip, the value 70 or 72 correspond to AGET or ASTRE in full readout mode, values from 0 to 72 correspond to AGET or ASTRE in partial readout mode.
be eg	chip	Shows the number of chips per simulated FE
be eg	chip <Cnt>	Sets the number of chips per simulated FE. Allowable values are 0 to 4, with 4 being the actual number of ASIC chips per physical FE
be eg	ena	Shows the state, enabled or disabled, of the event data generator
be eg	ena <0 | 1>	Enables or disables the event generator
be eg	emit_last	Shows whether the optional last cell read value is emitted or not
be eg	emit_last <0 | 1>	Sets whether the optional last cell read value information is emitted or not for each simulated front-end ASIC.
be eg	emit_hit	Shows whether the optional channel hit count value is emitted or not for the simulated front-end ASICs
be eg	emit_hit <0 | 1>	Sets whether the optional channel hit count value is emitted or not for the simulated front-end ASICs.
be eg	mode	Shows the mode of operation of the event data generator
be eg	mode <Val>	Sets the mode of operation of the event data generator. Argument: 0: constant event size 1: random size for the number of samples per simulated channel, but all channels have the same size within each event 2: constant event size 3: random size for the number of samples per simulated channel, the data of each

channel has a different size within the same event

10.1.16   DEAD-TIME AND INTER-EVENT TIME HISTOGRAMS

The TDCM measures the dead-time of the system for each event. This measurement is made from the time a valid trigger is received by the TDCM until all the FEs are ready to assert their respective SCA write signal, i.e. when they are ready to start the acquisition of the next event. The TDCM also measures the time interval between every two consecutive event. This measurement reflect the instantaneous trigger rate.

Dead-time and inter event time measurements are accumulated in two separates 1024-bin x 32-bit amplitude count histograms. Four resolutions settings are available for the dead-time histogram: 1 µs, 10 µs, 100 µs and 1 ms. These values correspond to measurement ranges of [0; 1.022 ms], [0; 10.22 ms], [0; 102.2 ms] and [0; 1.022 s]. The resolution of the inter-event time histogram is fixed to 100 µs leading to measurement range of [0; 102.2 ms]. When a measurement overflows the limit, the last bin of the relevant histogram is incremented. Each measurement is rounded to the closest resolution unit before it is added to its histogram. Saturation occurs when any bin reaches the maximum count. Statistics computed from these histograms are only accurate when the overflow bin is empty and no bin reached saturation. Note that for a given number of events N, the number of entries in the dead time histogram is N, while it is N-1 in the inter-event time histogram because the time interval between the first event and the precedent event is not taken into account. The commands related to dead-time and inter-even time measurements are listed in Table 42.

Table 42. Dead time and inter event time measurement related commands.

Command	Argument	Action
be busy_resol		Gets the resolution of the dead-time histogram
be busy_resol	<resol>	Sets the resolution of the dead-time histogram. Parameter: 0: 1 µs 1: 10 µs 2: 100 µs 3: 1 ms.
be hbusy	clr	Clears the histogram of dead time
be hbusy	get	Reads the histogram of dead time
be hevper	clr	Clears the histogram of inter event time
be hevper	get	Reads the histogram of inter event time

10.1.17   BOARD SERIAL ID AND MONITORING

The TDCM has a Maxim Semiconductor DS2438 silicon ID and monitor chip. This device provides a unique 48-bit serial identification number, and allows to measure local temperature, current, and two external voltages. The voltages that can be measured on the TDCM are the 3.3V and 2.5V power supplies. The command listed in Table 43 are used to access the DS2438 of the TDCM. The current measured is the total current drawn from the 12 V power input. It includes the current drawn by the FPGA module, the TDCM carrier and the mezzanine cards.

Table 43. Serial ID and monitoring commands.

Command	Argument	Action
be moni	<T|V|A|I|S>	Reads from the siicon ID chip of the TDCM, T: the local temperature, V: the 2.5V power supply, A: the 3.3V power supply, I: the current drawn by the TDCM, S: the 48-bit serial number of the TDCM.

10.1.18   COMMANDS FOR THE EMBEDDED FLASH MEMORY

The commands in Table 44 are used to read or write to the flash memory embedded on the FPGA module of the TDCM. Theses commands are not intended to be used directly, but they are intended to experts users. The content of the flash memory of the FPGA module must not be altered under normal circumstances.

Table 44. Commands for the embedded flash memory.

Command	Argument	Action
be flash	read 0xAdr 0xCount	Reads the content of the on-board flash memory starting at address 0xAdr and read 0xCount bytes. The maximum number of bytes that can be read with this command is 32.
be flash	<write | write_verify | erase_write | erase_write_verify> 0xAdr 0xCount data0data1data2…	Writes, writes and verify, erase then write, or erase write and verify, the supplied data starting at address 0xAdr. The argument count must not exceed 32. Each data byte is supplied with two hexadecimal digits without the usual "0x" prefix and must not be separated by space characters.

10.1.19   SD MEMORY CARD

The TDCM has a removable SD Memory Card that is essentially used to store the boot file that contains the FPGA firmware, the first stage boot loader program for the ARM processor of the ZYNQ, and also the main application software. The SD Card is formatted in FAT. If the SD Card is removed from the TDCM, files can be read and write using a PC with the appropriate card reader. When the SD Card is inserted in the TDCM, minimal

functions are provided to read and write files. Table 45 shows the available commands for file operations on the SD Card of the TDCM.

Table 45. Commands related to SD Card operations.

Command	Argument	Action
be sd wena		Shows if write protection is enabled or not for the media.
be sd wena	<0 | 1>	Enables or disable write operations to the media.
be sd mount		Mount the file system of the media
be sd umount		Unmount the file system of the media
be sd dir	<path>	Displays the content of the root directory or the content of the supplied path (the content displayed is limited to 256 characters)
be sd mv	<src> <dst>	Rename file <src> into <dst>
be sd rm	<file_name>	Delete file named <file_name>
be sd fopen	<r | w | c | f> <file_name>	Open a file for read and/or write. If the flag "c" is passed, the file is created if it does not exist. But if it already exists, an error is returned. If the flag "f" is passed, the file is created if it does not exist, and overwritten if it already exists.
be sd fclose		Close the currently opened file
be sd fread	<byte_count>	Reads <byte_count> from a currently opened file and prints them (in hexadecimal). When the end of file is reached, an empty string is printed. Argument <byte_count> is limited to 96.
be sd fwrite	<byte list…>	Write to the currently opened file. The byte list contains the data to be written. It must be composed of an even number of hexadecimal ASCII characters, where each group of two characters represents the hexadecimal value of the binary byte to be written. The byte list may contain up to 1024 characters, which represents 512 bytes of binary data.
be sd fdate	<file_name> <y/m/d h:m:s>	Sets the date and time of <file_name> to the supplied arguments. Note that the format must be respected: year/month/day hour:minutes:seconds.

Note that only short file names up to 8 characters are allowed. Note also that there is no distinction between upper case and lower case letters in file names. All file and directory names will be converted to upper case. The user will normally not need to use these functions directly. These commands are mostly used to upload new versions of the boot file of the TDCM remotely to the SD Card. See the section on boot file upload (client program command reference).

10.2 COMMANDS THAT APPLY TO THE FE

This sections contains the commands interpreted by the TDCM that perform some action on one or several of the FEs. **Important note**: the commands that are directed by the TDCM to the FE start with the prefix "fe". This prefix can optionally be followed by the placeholder character ".", surrounded by one space character before and after, or by the individual index of the FE to which the command applies. This optional argument after the prefix "fe" is intended to make each command self-contained. This is required when the TDCM receives commands from multiple clients that communicate with the same FEs. Although multiple clients are allowed, there must not be interference between them because the TDCM has no mechanism to provide exclusive access to FE hardware resources to a particular client. It is recommended that only one client performs system configuration and DAQ. There may be one or several distinct clients for slow control monitoring, non-intrusive register read-back and debugging. When no argument, or the placeholder character "." is placed after the prefix "fe", the command is directed to the FE, or the set of FEs, determined by the last command "be sel_fe …" or the last command "fe <index> …". For deployments that may use several clients, it is mandatory that every client only generates commands that contain the index of the FE to which the command applies. In a multi-client environment, a client program must not assume that the selection of FE(s) that it made with the command "be sel_fe …" at an earlier time may not be changed at any time by another client.

10.2.1 FE FIRMWARE VERSION, FE INDEX AND DNA

A project family number (optional), firmware major revision number and minor revision number are assigned at to the FPGA firmware of the FE at compilation time. This information can be read using the command "fe fw_version".

As it was explained earlier, each FEC is assigned a card index in [0;31] by the TDCM which corresponds to the ID of the physical port where it is connected. The correspondence between the serial DNA number of a FE and the card index is determined by the appropriate enumeration procedure at system initialization. The command "fe dna" given in Table 46 is used to request to a FE its ID and DNA.

Table 46. Commands for FE firmware version, index and DNA display.

Command	Argument	Action
fe fw_version		Retrieves the project family, and the major and minor firmware revision numbers
fe dna	show	Retrieves the DNA number and index of the currently selected FE

The "fe dna" command is normally called after the TDCM has enumerated all FEs and assigned an ID to each of them to verify that each FE has correctly received its own ID.

10.2.2 DIRECT ACCESS TO THE VIRTUAL MEMORY SPACE OF THE FE

After all FEs have established communication with the TDCM and have been properly enumerated, configuration and monitoring commands can be executed by series of read and write operations in the virtual register and memory space of the FEC that is accessible by the TDCM over Virtual Channel B. The commands listed in Table 47.

Table 47. Commands for direct access to the virtual Register/Memory space of a FEC.

Command	Argument	Action
fe bus	read <0xAdr> <0xByteEn>	Read the 32-bit data at address 0xAdr in the currently active FE. The address must be aligned in 4-byte boundaries. The optional argument <0xByteEn> is a 4-bit hexadecimal value that gives which bytes to read among the 4 bytes that are addressed. If this argument is omitted, the 4 bytes are read
fe bus	write <0xAdr> <0xByteEn> <0xData>	Writes the supplied 32-bit word at the specified address in the currently active FE. The address must be aligned on 4-byte boundaries. If the argument <0xByteEn> is omitted, all 4 bytes are altered. Otherwise only the bytes corresponding to the bits set to 1 in this argument are written.
fe reg	<Reg_index>	Reads the content of the configuration register of the specified index. This allow reading one the 16 configuration registers listed in Table 22.
fe reg	<Reg_Index> <0xValue>	Writes the specified configuration register with the supplied data. This allow writing one the 16 configuration registers listed in Table 22.

In principle, all the commands for configuring and monitoring the FE could be performed only by using the above commands. In practice, it can be simpler to use the more specific commands that are given in the following sub-sections. Note that the above commands do not allow to alter groups that are smaller than 8 bits (i.e. one byte). Hence, to alter a 3-bit field stored within a 32-bit wide register, the appropriate bits need to be read and preserved while those corresponding to the field to be programmed have to be set or cleared according to the value to be programmed. All these operations are done transparently by the commands dedicated to each particular function while the "fe bus" family of commands is more adequate for programming and reading back complete registers in each FE. For the emulation of serial protocols, SPI or I2C, the dedicated commands must be used because it would be far too complex for the user to provide the series of "fe bus" commands that make the appropriate sequence on the corresponding serial I/O pins of the FPGA of the FE.

10.2.3 COMMANDS RELATED TO THE COMMUNICATION LINK WITH THE TDCM

The FEC communicates with the TDCM over a custom link. The commands related to operation of this link are listed in Table 48.

Table 48. Commands related to the operation of the link to the TDCM.

Command	Argument	Action
fe crc32_insert_ena		Shows if the FE computes and inserts a CRC32 at the end of data packets sent to the TDCM over Virtual Channel C
fe crc32_insert_ena	<0 | 1>	Sets CRC32 computation and insertion in the data packets sent by the FE to the TDCM
fe rx_bert_ena		Shows whether the bit error rate tester in the TDCM to FEC direction is enabled or disabled in the currently active FE
fe rx_bert_ena	<0 |1>	Enables or disables the bit error rate tester in the TDCM to FEC direction on the currently active FE
fe rx_bert_pat		Shows the pattern expected by the FE for the bit error rate tester in the TDCM to FE direction
fe rx_bert_pat	<Pattern>	Sets the pattern expected by the FE for the bit error rate tester in the TDCM to FEC direction. Argument: 0: PRBS7 1: PRBS15 2: PRBS23 3: PRBS31
fe tx_bert_ena		Shows if the bit error rate tester is enabled in the currently active FE for the link in the FE to TDCM direction
fe tx_bert_ena	<0 | 1>	Enables or disables the bit error rate tester of the currently active FE in the FE to TDCM direction
fe tx_bert_pat		Shows the pattern generated by the currently active FE for the bit error rate tester in the FE to TDCM direction
fe tx_bert_pat	<Pat>	Sets the pattern generated by the currently active FE for the bit error rate tester in the FE to TDCM direction. Argument: 0: PRBS7 1: PRBS15 2: PRBS23 3: PRBS31

fe port	<id | * | p_first:p_last> <a | b | c | *> clr	Clears the message counters of the specified FE(s) and Virtual Channel(s)
fe port	<id> <a | b | c>	Shows the message counters of the specified FE for the specified Virtual Channel.

10.2.4   GENERAL FE CONTROL COMMANDS

General control commands for the FE are listed in Table 49.

Table 49. General control commands.

Command	Argument	Action
fe mode		Prints the mode of operation of the FE currently selected, AGET or AFTER
fe mode	<after | aget>	Sets the operating mode of the currently active FE to AGET or AFTER
fe fec_enable		Shows the state of FE enable bit
fe fec_enable	<0xVal>	Enable/disables power on the FEC (if supported). 0: both FECs OFF; 1: FEC #0 ON; 2: FEC #1 ON, 3: FEC#0 and FEC#1 ON.
fe asic_mask		Shows which ASICs are active/masked
fe asic_mask	<0xMask>	Sets which ASICs are active/masked (16-bit mask)
fe emit_hit_cnt		Shows if per ASIC total channel hit count is sent in the data stream or not
fe emit_hit_cnt	<0 | 1>	Disable/enable the option to send the per ASIC total channel hit count
fe rst_len		Shows the current setting of AGET reset channel count (2 or 4)
fe rst_len	<0 | 1>	Selects the setting for AGET reset channel count (0 leads to 2; 1 leads to 4)
fe forceon_all		Shows current setting to skip reading the Channel Hit Register and force readout of all channels
fe forceon_all	<0 | 1>	Clear or sets the option to force the readout of all channels in AGET mode.
fe keep_rst		Shows if reset channels are removed or kept in the data stream sent to the TDCM
fe keep_rst	<0 | 1>	Disable/enable the option to keep some of the reset channels in the data stream sent to the TDCM
fe skip_rst		Shows the current setting that determines which reset channels are dropped
fe skip_rst	<0,1,2,3>	Sets how many reset channels should be dropped
fe emit_lst_cell_rd		Shows the setting for the option to send in the data stream to the TDCM the value of the last cell read pointer of each ASIC
fe emit_lst_cell_rd	<0 | 1>	Disable or enable the option to send the last cell read pointer of each ASIC

fe mmpol		Shows the state of the Photomos relays controlling the polarization circuits of the currently active FE
fe mmpol	<Val>	Sets the state of the Photomos relays controlling the polarization circuits of the currently active FE. Argument values: 0: both Photomos in the open state 1: Photomos #0 closed, #1 opened 2: Photomos #0 open, #1 closed 3: both Photomos closed (normal mode)
fe state		Shows the current internal state of the FE

Depending on front-end implementation, it may be possible to power OFF some parts of a front-end. If the front-end supports this feature, the user must enable the FEC side using the command "fe fec_enable 1" before the card can be configured and used. On the T2K-II FEM, the hardware supports two FECs that can be powered ON or OFF independently. Other hardware implementations of the front-end do not use these settings.

By default, all ASICs are active and all the bits of ASIC_MASK are cleared. One or several ASICs may be disabled by setting the appropriate mask. All values are valid and ASIC_MASK=0xFFFF disables all ASICs. For hardware implementations where only 4 ASICs are controlled by a front-end node, only the 4 LSB's of this field are used. When all ASICs are masked, empty events containing only header and trailer information can be collected.

10.2.5 HIT CHANNEL REGISTER COMMANDS

In the AGET mode when the forceon_all option is disabled, the FE reads the Hit Channel Register to determine the set of channels to readout. The content of this register can optionally be modified before SCA digitization. The commands listed in Table 50 determine the corresponding alteration rules.

Table 50. Commands to act on Channel Hit Register.

Command	Argument	Action
fe modify_hit_reg		Shows the current setting for the modify hit register option
fe modify_hit_reg	<0 | 1>	Disable/enable Channel Hit Register alteration before SCA digitization
fe forceon	<A> <C>	Shows the setting to force the readout of channel <C> of Asic <A>
fe forceon	<A> <C> <0 | 1>	Disable/enable a forced readout for channel <C> of Asic <A>
fe forceoff	<A> <C>	Shows the setting to force skipping readout of channel <C> of Asic <A>

fe forceoff	<A> <C> <0 | 1>	Disable/enable a forced skip of channel <C> of Asic <A>
fe erase_hit_ena		Shows the setting to clear hit channel registers when the channel hit count is too high
fe erase_hit_ena	<0 | 1>	Disable/enable the function to clear hit channel registers when the channel hit count is too high
fe erase_hit_thr	<A>	Shows the hit count limit threshold of Asic <A>
fe erase_hit_thr	<A> <0xThr>	Sets the hit count limit threshold of Asic <A> to <0xThr>

When both forceon and forceoff options are disabled for a given channel, the value read from the Channel Hit Register is preserved. Both forceon and forceoff options must not be set simultaneously. Note that for programming rules, the forceon and forceoff commands accept scalar arguments, ranges or wildcard characters for arguments <A> and <C>. For example, the command "forceon * 3:78 1" sets to 1 the forceon flag for channel 3 to 78 of all ASICs.

The Hit Channel Register of each chip can be cleared automatically before SCA digitization when the number of hit channels passes a programmable limit. This feature is intended to minimize the dead-time caused by excessively busy events or noise events where almost all channels fire. This feature is only active when ERASE_HIT_ENA and MODIFY_HIT_REG are set. The threshold limit is programmable from 0x0 to 0x7 and corresponds to a maximum acceptable channel hit count of 4 to 32 channels. For example, if the threshold of a chip is set to 0, the Channel Hit Register of that chip will not be altered for events that have from 1 to 4 channels hit, and it will be cleared for events that have 5 channels hit or more. The number of channels hit in each chip is computed by the FPGA logic when reading the Hit Channel Register of each chip before SCA digitization and independently of the ForceOn and Force_Off rules. Note that the ForceOn and ForceOff rules are still applied to determine the final value of each Channel Hit Register when ERASE_HIT_ENA is set.

10.2.6   COMMANDS FOR PEDESTAL EQUALIZATION AND ZERO-SUPPRESSION

The FE can optionally perform pedestal subtraction and apply a threshold to zero-suppress data. A per channel constant pedestal value and threshold value can be programmed. The commands to set or read-back pedestal values and threshold values are listed in Table 51.

Table 51. Commands for pedestals and thresholds.

Command	Argument	Action
fe ped	<A> <C>	Shows the pedestal equalization constant of channel <C> of Asic <A>

fe ped	<A> <0xped>	<C>	Sets the pedestal equalization constant of channel <C> of Asic <A>
fe subtract_ped			Show the current setting for pedestal subtraction
fe subtract_ped	<0 | 1>		Disable/Enable pedestal subtraction
fe thr	<A> <C>		Shows the threshold of channel <C> of Asic <A>
fe thr	<A> <C> <0xthr>		Sets the threshold of channel <C> of Asic <A>
fe zero_suppress			Shows if zero-suppression is enabled or not
fe ero_suppress	<0 | 1>		Disable/enable data zero-suppression
fe polarity	<A>		Shows the polarity of the zero-suppressor of Asic <A>
fe polarity	<A> <0 | 1>		Sets the polarity of the zero-suppressor for Asic <A>. 0 to keep data above threshold; 1 to keep data below threshold
fe zs_pre_post			Shows the number of samples to keep before/after threshold is passed in zero-suppressed readout mode
fe zs_pre_post	<pre> <post>		Sets the number of samples to keep before/after threshold is passed in zero-suppressed readout mode
fe emit_empty_ch			Shows the current policy for sending or skipping empty channels in the data stream sent to the TDCM
fe emit_empty_ch	<0 | 1>		Sets the policy for handling empty channels: drop silently or send channel index and one null value

Acceptable values for the pedestal are 0xF000 (-4096) to 0x0FFF (+4095). Threshold values are from 0x0000 (0) to 0x0FFF (4095). When programming pedestal equalization constants and thresholds, commands arguments <A> and <C> can be scalars, ranges or "*" wildcard characters. For reading a pedestal or threshold value, the commands arguments must correspond to a single ASIC and Channel.

The zero-suppressor engine not only keeps data samples that are above the programmable threshold but it also keeps a programmable number of samples before the threshold is passed and after the threshold is no longer passed. From 0 to 15 precursor samples and 0 to 16 trailer samples can be kept on zero-suppressed waveforms. When working with detectors that produce positive signals, the zero-suppressor must be programmed to keep pulses that are below threshold rather than above.

When no data remains for a given channel after the zero suppression, either no data at all is sent for this channel, or the index of that channel followed by a null sample can

be sent. The command "emit_empty_ch" is used to set the desired option. Suppressing empty channels leads to more compact events while keeping empty channel can be useful for debugging.

10.2.7 COMMANDS FOR THE CONFIGURATION OF FRONT-END ASIC REGISTERS

Depending on implementation, the FE may support the AFTER, AGET or the ASTRE chip. Mixing different types of chips on the same FE is normally not supported. The AFTER and AGET/ASTRE chips have 3 and 13 programmable configuration registers respectively. These registers can be programmed and read-back via the commands listed in Table 52 and Table 53 for the AGET and AFTER/ASTRE chips respectively. Because registers have a different size (from 16-bit to 128-bit), the correct number of 16-bit arguments has to be supplied. Refer to the documentation of the AFTER and AGET/ASTRE chips for a detailed description of internal registers.

Table 52. Commands for operations on AGET/ASTRE internal registers.

Cmd	Argument	Action
fe aget	<A> read <reg>	Reads the content of register <reg> of AGET chip <A>
fe aget	<A> write <reg> <0xVal0> …	Writes register <reg> of AGET <A> with the specified value (16-bit per argument, supply the required number of values to match the actual size of the register)
fe aget	<A> wrchk <reg> <0xVal> …	Performs a write to register <reg> followed by a read and comparison
fe aget	<A> icsa	Reads the setting for the bias current of charge sense preamplifiers of AGET <A>
fe aget	<A> icsa <0 | 1>	Sets the bias current of the charge sense preamplifiers of AGET <A>: 0 normal bias; 1: normal bias x 2
fe aget	<A> time	Reads the shaping time of AGET <A>
fe aget	<A> time <0xVal>	Sets the shaping time of AGET <A> to <0xVal>
fe aget	<A> test	Shows Test setting of AGET <A>
fe aget	<A> test <0xVal>	Sets Test of AGET <A> to <0xVal>
fe aget	<A> mode	Shows Readout mode setting of AGET <A>
fe aget	<A> mode <0xVal>	Sets Readout mode of AGET <A> to <0xVal>
fe aget	<A> polarity	Reads the polarity setting of AGET <A>
fe aget	<A> polarity <0 | 1>	Sets the polarity of AGET <A>: 0 for detectors with negative signals, 1 for positive
fe aget	<A> fpn	Reads the FPN setting for AGET <A>
fe aget	<A> fpn <0xVal>	Sets the FPN readout of AGET <A>
fe aget	<A> vicm	Shows Vicm setting of AGET <A>

fe aget	<A> vicm <0xVal>	Sets Vicm of AGET <A> to <0xVal>
fe aget	<A> dac	Reads the DAC threshold of AGET <A>
fe aget	<A> dac <0xVal>	Sets the DAC threshold of AGET <A>
fe aget	<A> trigger_veto	Reads the trigger veto of AGET <A>
fe aget	<A> trigger_veto <0xVal>	Sets the trigger veto of AGET <A>
fe aget	<A> synchro_discri	Reads the synchronization discriminator bit of AGET <A>
fe aget	<A> synchro_discri <0 |1>	Clears or sets the synchronization discriminator bit of AGET <A>
fe aget	<A> tot	Reads the time-over-threshold bit of AGET <A>
fe aget	<A> tot <0 | 1>	Clears or sets the time-over-threshold bit of AGET <A>
fe aget	<A> range_tw	Reads the range bit of trigger width of AGET <A>
fe aget	<A> range_tw <0 | 1>	Clears or sets the range bit of trigger width of AGET <A>
fe aget	<A> trig_width	Reads the trigger width of AGET <A>
fe aget	<A> trig_width <0xVal>	Sets the trigger width of AGET <A>
fe aget	<A> rd_from_0	Reads the bit read_from_0 of AGET <A>
fe aget	<A> rd_from_0 <0 |1>	Clears or sets the bit read_from_0 of AGET <A>
fe aget	<A> tst_digout	Reads the bit tst_digout of AGET <A>
fe aget	<A> tst_digout <0 | 1>	Encode the index of the last SCA cell read or a fixed pattern (0x159) at the end of the SCA read phase of AGET <A>
fe aget	<A> en_mkr_rst	Reads en_mkr_rst setting of AGET <A>
fe aget	<A> en_mkr_rst <0 | 1>	Clears or sets en_mkr_rst of AGET <A>
fe aget	<A> rst_level	Reads rst_level setting of AGET <A>
fe aget	<A> rst_level <0 |1>	Clears or sets rst_level of AGET <A>
fe aget	<A> cur_ra	Reads the SCA line buffer current setting of AGET <A>
fe aget	<A> cur_ra <0xCur>	Sets the SCA line buffer current of AGET <A>
fe aget	<A> cur_buf	Reads the buffer current setting of AGET <A>
fe aget	<A> cur_buf <0xCur>	Sets the buffer current setting of AGET <A>
fe aget	<A> short_read	Reads short_read setting of AGET <A>
fe aget	<A> short_read <0 | 1>	Clears or sets short_read of AGET <A>
fe aget	<A> dis_multiplicity_out	Reads dis_multiplicity_out setting of AGET <A>
fe aget	<A> dis_multiplicity_out <0 | 1>	Clears or sets dis_multiplicity_out of AGET <A>
fe aget	<A> autoreset_bank	Reads autoreset_bank setting of AGET <A>

fe aget	<A> autoreset_bank <0 | 1>	Clears or sets autoreset_bank of AGET <A>
fe aget	<A> in_dyn_range	Reads in_dyn_range setting of AGET <A>
fe aget	<A> in_dyn_range <0 | 1>	Clears or sets in_dyn_range of AGET <A>
fe aget	<A> gain <c>	Reads the gain setting of AGET <A> Channel <c>
fe aget	<A> gain <C> <0xVal>	Sets the gain of channel(s) <C> of AGET <A> to <0xVal>
fe aget	<A> inhibit <c>	Reads the inhibit setting of AGET <A> Channel <c>
fe aget	<A> inhibit <C> <0xVal>	Sets the inhibit of channel(s) <C> of AGET <A> to <0xVal>
fe aget	<A> threshold <c>	Reads the threshold setting of AGET <A> Channel <c>
fe aget	<A> threshold <C> <0xVal>	Sets the threshold of channel(s) <C> of AGET <A> to <0xVal>
fe aget	<A> threshold ++ | --	Increments / decrements the threshold of AGET <A>
fe aget	<A> hitprob <C> <Val>	Sets the threshold of chanel(s) <C> of AGET <A> so that the probability that this channel appears as hit is less than <Val> (assumes that histograms of hit probability have been accumulated)

All commands that perform a write operation support argument <A> in the following format: the "*" wildcard to designate all AGET chips, a range (two integers separated by the ":" character) or a single index. Read operations can only apply to a single AGET and eventually a single channel. The majority of the commands that modify the content of AGET registers perform internally a write followed by a read and a comparison. If the command did not return an error, it is reasonably safe to consider that the register was modified as intended.

Table 53. Commands for operations on AFTER internal registers.

Cmd	Argument	Action
fe after	<A> read <reg>	Reads the content of register <reg> of AFTER <A>
fe after	<A> write <reg> <0xVal0> …	Writes register <reg> of AFTER <A> with the specified value (16-bit per argument, supply the required number of values to match the actual size of the register)
fe after	<A> wrchk <reg> <0xVal0> …	Writes register <reg> of AFTER <A> with the specified value, performs read-back and verification
fe after	<A> gain	Shows the current gain of AFTER <A>

fe after	<A> gain <120,240,360,600>	Sets the gain of AFTER <A> to a dynamic range of 120 fC, 240 fC, 360 fC or 600 fC
fe after	<A> time	Shows the shaping time of AFTER <A>
fe after	<A> time <stime_ns>	Sets the shaping time of AFTER <A> to <stime_ns> (16 possible values)
fe after	<A> test_mode	Shows the test mode of AFTER <A>
fe after	<A> test_mode <0 xMode>	Sets test mode of AFTER <A> to <0xMode> (4 possible values)
fe after	<A> en_mkr_rst	Reads the value of the enable marker reset flag of AFTER <A>
fe after	<A> en_mkr_rst <0 | 1>	Disable/enable the reset marker of AFTER <A>
fe after	<A> rst_level	Reads the level of the reset marker of AFTER <A>
fe after	<A> rst_level <0 |1>	Sets to low or high the level of the reset marker of AFTER <A>
fe after	<A> rd_from_0	Reads the bit read_from_0 of AFTER <A>
fe after	<A> rd_from_0 <0 | 1>	Disable/enable the option to force SCA read from cell 0
fe after	<A> test_digout	Reads the bit test_digout of AFTER <A>
fe after	<A> test_digout <0 | 1>	Disable/enable the option to send a fixed pattern (0x159) instead of the index of the last cell read at the end of the SCA read phase

For AFTER register modification commands, argument <A> also accepts wildcards, ranges and scalar values. For read-back commands, only a scalar value for <A> is accepted. Contrary to AGET related commands, no verification of register content is made after a modification command (except for the "wrchk" command). The verification may be added in future embedded software releases.

10.2.8   COMMANDS RELATED TO THE CONFIGURATION OF THE TRIGGER

The FE includes a flexible trigger and a programmable rate on-board event generator. The corresponding control commands are listed in Table 54.

Table 54. Trigger and event generator control commands.

Command	Argument	Action
fe trig_ena		Reads the current setting of trigger mask in the active FE
fe trig_ena	<0xVal>	Enables the trigger sources specified by <0xVal>
fe trig_delay		Reads the current value of trigger delay
fe trig_delay	<0xVal>	Sets the trigger delay to <0xVal>

The argument of "fe trig_ena" is a 4-bit hexadecimal value where each bit enables the following sources of trigger: bit 0 (AUTO_TRIG_ENABLE): on-board generator; bit 1 (EXT_TRIG_ENABLE): external trigger pin EXT_TRIG; bit 2 (PUL_TRIG_ENABLE): local trigger from pulser; bit 3 (TDCM_TRIG_ENABLE): remote trigger received from the serial link to the TDCM.

The argument of the "fe trig_delay" command is an unsigned integer for 0 to 131068 that expresses in 10 ns units the delay to apply to all types of triggers. The maximum delay is 1.31068 ms. However, if the supplied argument is above 32767, the resolution of the delay is changed from 10 ns to 40 ns.

10.2.9   SCA RELATED COMMANDS

The operation of the internal SCAs of the front-end ASICs is governed by the commands listed in Table 55.

Table 55 . SCA control commands.

Command	Argument	Action
fe sca	cnt	Reads the current setting for the number of SCA cells to read out
fe sca	cnt <0xVal>	Sets the number of SCA cells to digitize (from 8 to 511 or 512)
fe sca	wckdiv	Reads the current value of SCA Write clock divisor
fe sca	wckdiv <0xDiv>	Sets the SCA write clock divisor to <0xDiv> (from 1 to 255)
fe sca	enable	Reads the current value of the SCA_ENABLE bit
fe sca	enable <0	1>	Clears or sets the SCA_ENABLE bit
fe sca	autostart	Shows the current value of the SCA_AUTO_START bit
fe sca	autostart <0	1>	Clears or sets the SCA_AUTO_START bit
fe sca	start	Starts SCA write operation
fe sca	stop	Generates a software trigger event

After system configuration, the SCA_ENABLE bit must be set to 1 to be ready to take data. Clearing the SCA_ENABLE bit stops operation as soon as the readout of the current event is finished. The write operation in front-end SCAs starts immediately if the SCA_AUTO_START bit was set to 1. If not, writing in the SCAs will only start after the corresponding order has been received from the TDCM. If the FE is operated in standalone mode without the TDCM, SCA_AUTO_START is normally set to 1. If the FE is used as a slave device controlled by the TDCM, SCA_AUTO_START must be 0.

The sca start command is used when SCA_AUTO_START is inactive and the TDCM is not present or disable. This command is normally only used in standalone mode when pulse generator of the FE is being used. In this particular mode, the SCA start order is given in software and the stop order is automatically generated by the pulse generator control logic.

The sca stop command is used to issue a trigger in software for test purposes in standalone mode. It should not be used when running with the TDCM.

10.2.10   SYNCHRONIZATION COMMANDS

In a system with multiple FEs, synchronization signals are distributed by the TDCM. It is also possible to perform some of the functions of the TDCM asynchronously for each FE with the commands listed in Table 56.

Table 56. Synchronization commands.

Command	Argument	Action
fe clr	tstamp	Clears or presets (if an initial value different from 0 was pre-loaded) the time stamp counter
fe clr	evcnt	Clears the event counter
fe tstamp_init	0xVal_msb 0xVal 0xval_lsb	Loads a 48-bit preset value for the timestamp counter.
fe tstamp_isset		Shows if timestamp preset was received since last clear
fe tstamp_isset	clr	Clears the timestamp preset received indicator flag

If trigger generation is disabled during the operation, it is an acceptable method to clear the event counter sequentially in each FE. Event numbers will match across the whole system after the trigger is restarted (synchronously on all FEs). On the other hand, event time stamps will only match after a synchronous clear/preset through the TDCM has been made prior to data taking.

By default, the timestamp counter will be set to 0 upon clear. But any other initial value can be defined using the "fe tstamp_init" command. A clear of the timestamp counter will in fact cause a preset of this counter to the initial value programmed. This feature can be used to synchronize to the same value the timestamps across multiple systems. The initial value for the timestamp counter should be set to the value required to compensate for the propagation delay of the synchronous clear signal through the TDCM and the FEs.

The command "fe tstamp_isset" is used to check if the FE received a synchronous message from the TDCM with the bit CLR_TSTAMP set. This flag can be cleared with the command "fe tstamp_isset clr".

10.2.11   PULSE GENERATOR CONTROL COMMANDS

The FEC controls an embedded pulse generator that can inject calibrated charges to the calibration or functional test inputs of the front-end ASICs at a well-defined time. Depending on the version of the FE, the calibration pulser is realized with different circuitry and components. The main component of the pulser is the programmable DAC that controls the amplitude of the injected pulse. The commands that control the pulse generator of the FE are listed in Table 57.

Table 57. Pulse Generator control commands.

Command	Argument	Action
fe pulser	delay	Reads the delay of the pulse generator
fe pulser	delay <0xVal>	Writes pulse generator delay with <0xVal>
fe pulser	model	Shows the model of DAC used for the pulser of the currently active FE that is assumed by the TDCM

fe pulser	model <Mod>	Sets the model of DAC used by the pulser of the FE that the TDCM has to take. Argument:
		    0: unknown
		    1: AD9744
		    2: AD5667
		    3: others

fe pulser	base <0xVal>	Presets the baseline of the pulser DAC to <0xVal>
fe pulser	ampl <0xVal>	Presets the amplitude of the pulser DAC to <0xVal>
fe pulser	load	Loads the DAC with the preset value (implementation dependent)
fe pulser	enable	Reads the state of the pulser enable bit
fe pulser	enable <0 | 1>	Enable / disable pulser injection
fe pulser	ft_enable	Shows the state of the FT_ENABLE analog switch
fe pulser	ft_enable <0 | 1>	Sets the state of the FT_ENABLE analog switch

The command "fe pulser delay" sets the delay between the active edge of SCA_WRITE and the injection of the charge by the pulse generator. The delay is expressed in units of 10 ns and the acceptable range is 0 to 131,072 units. However, the resolution of the delay of the pulser is 10 ns in the [0; 327.67 μs] range and 40 ns in the [327.68 μs; 1.31068 ms] range.

The command "fe pulser model" is used to instruct the TDCM on which model of pulser is presumably implemented in the currently active FE. The appropriate value must

be set before the pulser can be used. Currently, the FE does not have some means that would allow the TDCM determine by itself which type of pulser is implemented on the FE. So far, this information must be supplied by the user to the TDCM.

The "fe pulser base" and "fe pulser ampl" commands are used to program the DAC of the pulse generator with the baseline and amplitude levels respectively. When the model of DAC used is the AD5667, the expected value is a 16-bit unsigned integer. In the implementation based on the AD9744, arguments are a 14-bit unsigned integer. For both implementations, the programmed values cannot be read-back.

The "fe pulser load" command is only used for the implementation based on the AD9744 (single channel DAC). This command performs the actual update of the output of the DAC to the baseline previously programmed. This command must be called to set (or restore from the previous value) the baseline output level of the DAC for every pulse the user wants to generate. In the implementation based on the AD5667, two DAC channel are used: one for the baseline and one for the amplitude. An analog switch is used to switch between the two analog outputs to perform the actual pulse generation. Hence the baseline and amplitude can be left unchanged between the injection of several pulses.

The "fe pulser enable" command is used to check, enable or disable the operation of the pulser. The pulser must be enabled when it is in use. The trigger associated to the pulser should be enabled for test or calibration with the pulser but should be disabled when testing triggers based on AGET multiplicity outputs.

The command "fe pulser ft_enable" is only relevant for the pulser based on the AD5667. This bit controls the state of the external switch that connects the functional test input pin of the four ASICs of the FEC to the output of the pulser or to ground.

10.2.12   MULTIPLICITY PROCESSING

The multiplicity output of the AGET chips can be used to elaborate a local self-trigger (if the FEC supports standalone operation) or the multiplicity information be sent to the TDCM for global processing. The multiplicity output of each AGET chip is fed to two comparators with a different 8-bit programmable threshold. The multiplicity signal must be greater than the first threshold and less than the second threshold to fire the corresponding logic. The commands related to multiplicity processing are listed in Table 58.

Table 58. Multiplicity related commands.

Command	Argument	Action
fe mult_thr	<aget>	Reads the current multiplicity threshold of AGET <aget>

fe mult_thr	<*|aget> <0xVal>	Sets the multiplicity threshold of one or multiple AGET to the specified value
fe mult_limit	<aget>	Reads the current multiplicity threshold limit of AGET <aget>
fe mult_limit	<*|aget> <0xVal>	Sets the multiplicity threshold limit of one or multiple AGET to the specified value
fe mult_trig_ena		Shows if self-trigger based on multiplicity are enabled or not
fe mult_trig_ena	<0 | 1>	Disables or enables self-triggers based on multiplicity
fe snd_mult_ena		Shows if sending multiplicity-over-threshold bits to the TDCM is enabled or disabled
fe snd_mult_ena	<0 | 1>	Disables or enables sending the multiplicity-over-threshold bits to the TDCM

The command "fe mult_trig_ena" is used to check, enable or disable self-triggers based on AGET multiplicity. When active, a local self-trigger is generated whenever one or more of the multiplicity output of the AGET is above the programmed threshold and below the programmed limit. Enabling the local self-trigger is only possible for FEC models that support standalone operation without the TDCM.

The command "fe snd_mult_ena" is used to control if the multiplicity-over-threshold bits generated by the FEC are forwarded to the TDCM or not.

10.2.13 TEST DATA PATTERN GENERATOR

The FE features a test mode and has a fully programmable ADC data pattern memory. The data pattern memory simulates the data-stream of hit channels. The test memory contains 4096 12-bit entries. It can store the data of up to 8 channels for a 512-time bin configuration. The same data pattern is reproduced cyclically when more data are needed. The commands related to the test mode and patterns are listed in Table 59.

Table 59. Test mode and pattern related commands.

Command	Argument	Action
fe test_enable		Reads the current setting of the TEST_ENABLE bit
fe test_enable	<0 | 1>	Disable/enable TEST_ENABLE
fe test_mode		Reads the current value of the TEST_MODE bit
fe test_mode	<0 | 1>	Clears or sets the TEST_MODE bit
fe test_zbt		Reads the current setting of the TEST_ZBT bit
fe test_zbt	<0 | 1>	Disable/enable TEST_ZBT
fe tdata	<0xAdr>	Reads test data sample at address <0xAdr>
fe tdata	<0xAdr> <0xData>	Writes <0xData> in pattern memory at address <0xAdr>
fe tdata	<A, B, C> <0xVal>	Fill test memory with a pre-defined pattern

fe keep_fco		Reads the state of the KEEP_FCO option
fe keep_fco	<0 | 1>	Clears/Sets the KEEP_FCO option (drops data of ASIC#3 to keep FCO data when set)

When test mode is disabled, the data fed at the input of the zero-suppression block are the data digitized from the SCAs of the front-end chips. When test mode is enabled, the zero-suppression block is fed with the address counter where SCA data would be read when TEST_MODE is 0 and test data fetched from the pattern memory when TEST_MODE is 1. The pattern memory may be filled with an arbitrary pattern loaded word by word, or can be filled with one of several pre-defined patterns via a single command. Pattern A is a periodic increasing ramp starting from 0 and incremented by one unit until the specified value minus one is reached. Pattern B is similar except that the ramp is decreasing. Pattern C is a single sample wide pulse of fixed amplitude (2048 ADC counts) with a repetition period specified in the supplied argument. Different pre-defined patterns may be added in the future.

When TEST_ZBT is 0, the source of data for writing to the event buffer memory is the stream of data received from the external ADC. When TEST_ZBT is 1, the 12-LBS's of the write address to the event buffer memory is taken as data. This mode is meant for tests.

The serial outputs of the ADC of the FE produce a constant 12-bit binary pattern used to delineate channel data samples. For debugging purposes, this framing pattern can be kept in the stream passed to the DAQ. The data of ASIC #3 (or #7 and #15) are dropped when this option is enabled. The constant value that replaces ASIC #3 (or ASIC #7 and #15) data is 0xFCO. Pedestal equalization and zero-suppression (if enabled) are still applied to this stream of constant data.

10.2.14   FRONT END ADC CONFIGURATION

Depending on the implementation, the FE may be equipped with different model of ADCs for the digitization of the front-end ASICs. Some models can be controlled via I2C. The commands related to configuration of the ADC on-board the FE are listed in Table 60. Note that the actual operation performed depends on the type of ADC that is declared for each individual FE. At present, the FE does not includes means for the TDCM to determine which model of ADC is on-board. This information has to be set in the TDCM by the appropriate user command.

Table 60. Commands related to the ADC of the FE.

Command	Argument	Action
fe adc	model	Shows which model of ADC the TDCM assumes is present on the currently active FE
fe adc	model <Type>	Sets the model of ADC that the TDCM have to consider for the currently active FE. Argument:

		0: unknown
		1: AD9229
		2: AD9228
		3: others
fe adc	read <Reg>	Reads register <Reg> from the ADC of the currently active FE.
fe adc	write <Reg> <Val>	Writes register <Reg> of the ADC of the currently active FE with the value <Val>.

10.2.15   PEDESTAL HISTOGRAMS, EQUALIZATION AND THRESHOLD SETTING

The TDCM can accumulate the pedestal histograms of each channel of every FE during specific runs. When this mode of operation is selected, the data received from the hardware is not transferred to the remote DAQ computer, but it is processed locally by the TDCM instead. After the completion of the pedestal accumulation run, the user can acquire pedestal histograms in various different formats. The user can also instruct the TDCM to program in the FE the internal pedestal equalization table to adjust the mean pedestal to the desired level. The TDCM can also program in each FE the threshold table used for the real time digital zero-suppression of channel data. The threshold of each channel can be set to some value above the equalized pedestal plus the desired number of sigmas above noise. The commands to control pedestal histogram functions are listed in Table 61.

Table 61. Pedestal Histogram commands.

Command	Argument	Action
fe hped	<A> <C> clr	Clears the pedestal histogram(s) of specified Asic(s) and Channel(s)
fe hped	<A> <C> offset <off>	Sets the pedestal histogram offset of specified Asic(s) and Channel(s) to <off>
fe hped	<a> <c> getbins	Gets the full list of non-null bins of the pedestal histogram of channel <c> of Asic <a>
fe hped	<a> <c> getmath	Gets detailed statistics of the pedestal histogram of channel <c> of Asic <a>
fe hped	<a> <C> getsummary	Gets a summary of the pedestal histogram(s) statistics for channel(s) of Asic <a>
fe hped	<A> <C> centermean <M>	Program equalization constants in the current FE so that pedestal of specified Channel(s) of Asic(s) is <M>
fe hped	<A> <C> setthr <M> <S>	Program thresholds of specified Channel(s) of Asic(s) in the current FE to <M> + <S>*_channel_noise
fe list	<a> ped	Lists the pedestal adjustment constant for all channels of Asic <a> of the current FEC

fe list	<a> thr	Lists the programmed threshold for all channels of Asic <a> of the current FE

Pedestal histograms have 512 bins. When working with detectors producing negative signals, the baseline is typically ~250 ADC counts and the recommended range for pedestal histograms is [0; 511]. When detectors producing positive signals are used, the baseline is ~3840 ADC counts and the range of the pedestal histogram should be set to [3584; 4095]. The offset of pedestal histograms should be set to 0 or 3584 according to the polarity of detector signals.

At the end of a pedestal accumulation run, the user must flush any data that may be left in the ring buffer interface to the hardware and enable normal data taking mode with the "serve_local 0" command.

The list of pedestal adjustment constants and threshold settings can be retrieved with the "list" commands. These will be displayed in the client program as a list of interpretable commands so that they can be saved and re-loaded at a later time in an easy way.

The arguments <A> and <C> can be scalar, a range of the type <Begin:End>, or the "*" wildcard character to match all possible values. The arguments <f>, <a> and <c> specify only one ASIC, or channel at a time.

Note that all histograms are accumulated in the SDRAM of the TDCM while all pedestal equalization constants and thresholds for zero-suppression are stored on the front-end side.

10.2.16 Channel Hit Count Histograms

The FE computes for each event the count of channels hit in the 4 ASICs it controls and accumulates this information in histograms. In AFTER mode, the count of channels hit is constant and equal to 79 for each event (72 physical channels + 4 FPN + 3 reset channels). In AGET mode, it ranges from 2 to 72 depending on the number of channels hits obviously, but also on various settings: 2 or 4 reset channels, read all channels or only hit channels, alterations made at run-time to the content of the hit channel register, etc. The channel hit count per event are accumulated in four 80-bin x 32-bit amplitude count histograms. These histograms are useful for on-line monitoring and system performance evaluation. The commands related to channel hit count histograms are listed in Table 62.

Table 62. Commands related to channel hit count histograms.

Command	Argument	Action

fe hhit	\<A\> clr	Clears the channel hit count histogram of the specified ASIC in the FE currently selected
fe hhit	\<a\> get	Reads the channel hit count histogram of the specified ASIC in the FE currently selected

10.2.17   DEAD-TIME HISTOGRAMS

The FE also measures the dead-time for the acquisition of each event. This measurement is made from the time a valid trigger is received by the TDCM until all the FEC is ready to assert its SCA write signal. This histogram is normally only useful when the FEC operates in standalone mode without the TDCM. In a system with multiple FEs, the histogram accumulated by the TDCM should be used because it combines the dead-time all FEs.

The FE accumulates its own dead-time in a 1024-bin x 32-bit amplitude count histogram. Four resolutions settings are available: 1 µs, 10 µs, 100 µs and 1 ms. These values correspond to measurement ranges of [0; 1.022 ms], [0; 10.22 ms], [0; 102.2 ms] and [0; 1.022 s]. When a measurement overflows the limit, the last bin of the histogram is incremented. Each measurement is rounded to the closest resolution unit before it is added to the histogram. Saturation occurs when any bin reaches the maximum count. Statistics computed from this histogram are only accurate when the overflow bin is empty and no bin reached saturation. The commands related to dead-time measurements are listed in Table 63.

Table 63. Front-end dead time measurement related commands.

Command	Argument	Action
fe busy_resol		Gets the resolution of the dead-time histogram

fe busy_resol	\<resol\>	Sets the resolution of the dead-time histogram. Parameter:
		    0: 1 µs
		    1: 10 µs
		    2: 100 µs
		    3: 1 ms.

fe hbusy	clr	Clears the histogram of dead time
fe hbusy	get	Reads the histogram of dead time

10.2.18   MISCELLANEOUS MONITORING COMMANDS

The FPGA logic of the FE has controllers for the OneWire bus and the I2C bus. A silicon identifier chip (Maxim DS2438) is mounted on some models of FEC. For FE models that use an embedded FPGA module from Enclustra (Mars MX2 and some versions of Marx AX3), this module has a secure serial EEPROM and a real time clock. The FE normally contains a SFP optical transceiver. This device has a monitoring interface accessible via

I2C. All these various devices can be readout with the commands listed in Table 64. Other implementations of FE may have some other devices for the purpose of slow control / monitoring.

Table 64. Slow control monitoring commands.

Command	Argument	Action
fe moni	<T|V| A|I|S>	Reads information provided by the DS2438 chip of the FE. T: measured local temperature; V: supply voltage; A: supply voltage; I: measured supply current; S: 48-bit unique serial number.
fe sfp	<info | moni>	Reads the information (manufacturer, device number) from the SFP transceiver over I2C or some of the monitored operational parameters (supply voltage, current, optical power, temperature)

.

10.2.19 COMMANDS RELATED TO SPI FLASH MEMORY OF THE FRONT-END

The TDCM can have read, write and program access to the SPI flash memory of each FE. This is useful for downloading new revisions of the firmware of the FE without physically accessing each FE with a JTAG cable. The commands shown in Table 65 are used to read or write to the SPI flash memory of a target FE. At the present time, only Spansion S25FL512S SPI flash is supported. Other devices may be supported if needed. Theses commands are not intended to be used directly by the normal user. The content of the SPI flash memory of a FE must not be altered under normal circumstances.

Table 65. Commands to control the SPI flash memory of a FE.

Command	Argument	Action
fe flash	open	Get ready to access the SPI flash. This command must be called before accessing the device. It can apply to only one FE.
fe flash	close	This should be called when access to the SPI flash of the FE is no longer needed
fe flash	id	Shows the first 8 bytes of the Manufacturer and Device IP field of the SPI flash memory.
fe flash	read 0xAdr 0xCount	Reads the content of the SPI flash memory of the target FE starting at address 0xAdr and read 0xCount bytes. The maximum number of bytes that can be read with this command is 32.
fe flash	<write | write_verify | erase_write | erase_write_verify>	Writes, writes and verify, erase then write, or erase write and verify, the supplied data starting at address 0xAdr. The argument count must not exceed 512 bytes (i.e. one page). Each data byte is supplied with two hexadecimal digits without the

0xAdr	0xCount	usual "0x" prefix and must not be separated by
data0data1data2…		space characters.

10.2.20   FRONT-END XADC CONFIGURATION AND READ-BACK COMMANDS

Assuming the FE uses a Xilinx FPGA, the embedded ADC, "XADC", may be used for monitoring of internal and external variables (voltage, temperature, etc). The commands to configure and read-back the XADC in the FE are shown in Table 66.

Table 66 . Commands to control XADC in the FE.

Command	Argument	Action
fe xadc	read 0xAdr	Read XADC Register at address 0xAdr.
fe xadc	write 0xAdr 0xData	Write XADC Register at address 0xAdr with data 0xData.

Refer to [5] for details on how to configure and read-back XADC.

11 CLIENT PROGRAM

Pclient is a minimal, console-based, multi-platform, client program for controlling one or several TDCMs from a PC in a simple way. This application is mostly intended for the development and test of the TDCM itself, for new users to become familiar with the TDCM, and for debugging. *Pclient* is not intended to be used for real data taking in a physics experiment and it cannot replace a more elaborated configuration and DAQ software with some user-friendly graphical interface. Nonetheless, *pclient* gives access to the full functionality of the TDCM and the associated front-end. It can configure the system, perform data acquisition, and save the acquired data to local storage.

11.1 INSTALLATION

The *pclient* program can be installed on Windows and Linux platforms. Dependencies on external libraries are minimal and the program should compile on the majority of platforms. The code is entirely written in C and requires Microsoft Visual Studio Express for compilation under Windows, and a standard C compiler under Linux. To install a release of the TDCM client software, the procedure is the following:

- Save the archive file to the local disk and unzip it.

- Under Windows, compile the util library. The project file is located in:

```
<release_name>/projects/pandax/util/winnt/util.sln
```

- Compile *pclient*, *preader* and *fdecoder*. For Windows and Linux respectively, the project file and makefile are located in:

```
<release_name>/projects/pandax/mclient/winnt/mclient.sln
<release_name>/projects/pandax/mclient/linux/makefile
```

- For Windows and Linux respectively, the executable files are located in:

```
<release_name>/projects/bin/pandax/winnt/pclient.exe
<release_name>/projects/bin/pandax/linux/pclient
```

11.2 USING PCLIENT

Pclient is started from a DOS command window or a Linux terminal. The available command line options are the following:

```
pclient <options>
-h                 : print this message help
-arc               : server is an ARC (by default: TDCM)
-s <xx.yy.zz.ww> : base IP address of remote server(s) in dotted decimal
-p <port>          : remote UDP target port
-S <0xServer_set>: hexadecimal pattern to tell which server(s) to connect
-c <xx.yy.zz.ww> : IP address of the local interface in dotted decimal
-f <file>          : read commands from file specified
-o <file>          : save results in file specified
-v <level>         : verbose
```

By default, *pclient* expect to interact with one TDCM only, at a default IP address 192.168.10.1. The "-s" option is used to specify a different IP address. The "-arc" option is used to specify that the controlled hardware is an ARC (standalone version) instead of a TDCM. The "-p" option can be used to specify the UDP port number. Multiple TDCMs or ARCs, called "servers" can be specified with an hexadecimal pattern after the option "-S". Up to 32 servers can be interrogated from the same instance of *pclient*. For example, specifying the option "-S 0xB" means that *pclient* opens a connection with servers #0, #1 and #3. The IP address of each server is derived from the base IP address of the first server, in incremental order. For example: server #0 is at IP address 192.168.10.1, server #1 at 192.168.10.2, etc. The "-c" option is used to select the IP address of the client interface. The client PC must be on the same subnetwork as the server(s), i.e. 192.168.10.xx by default. If the control PC has only one network interface card on the required subnetwork, the "-c" option need not be specified. However, if the control PC has a NIC with several Ethernet interfaces, the appropriate port must be selected with the "-c" option.

For robust operation, the TDCM (and ARC) require a private Ethernet sub-network. No other traffic that that for communication with the TDCM-ARC should be routed to that private network. It is recommended to use Gigabit Ethernet, although Fast Ethernet should also work. When using Gigabit Ethernet, Jumbo frames up to 8 kByte should be enabled (this is even mandatory for systems of a certain size, e.g. more than 4 front-end cards).

Multiple instances of *pclient* that control the same TDCM can be started simultaneously provided that: only one instance does data acquisition and different UDP port numbers are used for each instance. Note however, that the actions initiated by each instance of *pclient* must remain coherent because the TDCM does not prevent concurrent access by several clients to common hardware resources. For example, one instance of *pclient* can be used to perform the periodic slow control monitoring task only, while a second instance of *pclient* can be used for run-time system configuration and data acquisition.

11.3 COMMAND REFERENCE

The majority of commands entered in *pclient* are forwarded to the server without any modifications. Some commands are interpreted locally, and some are altered by *pclient* before they are forwarded to the server.

The general control commands of *pclient* are listed in Table 67.

Table 67. *pclient* general control commands.

Command	Argument	Action

version		Returns major/minor software version and compilation date
quit		Quits the client program (server still running)
exit		Similar to command quit
verbose		Shows the level of verboseness
verbose	<level>	Sets the level of verboseness
vflags	<0xFlags>	Selects the type of debug information printout. Each bit of <0xFlags> corresponds to some particular type of information.
srv		Shows to which server commands apply
srv	* <0xSrvBitField> <srv_id>	Commands will apply to all servers Commands will apply to subset of servers Commands will apply to server of ID srv_id
sleep	<duration>	Sleeps for <duration> seconds
exec	<script_name.txt>	Execute the command script <script_name.txt>
LOOP	<nb_iterations>	Repeat the block of commands until NEXT <nb_iteration times>
LOOP	<begin> TO <end>	Similar to LOOP except that the boundaries of the index of the loop are specified
NEXT		Sets the end of the block of commands to repeat
END		Recommended at the end of a script file
	$loop	When this special string is found in some few specific commands, it is replaced on the fly by the current value of the index of the loop.
rcp	<source_bin> <boot.bin>	Copies the firmware file <source.bin> to the micro-SD memory card of the TDCM. Caution! If an incorrect file is copied, the TDCM will no longer boot!
fe program	flash Spansion <fe_firmware.mcs>	Transfers the firmware file <fe_firmware.mcs> to the SPI flash memory of the selected FE. Caution! Make sure a valid firmware file is downloaded! The transfer must not be interrupted until it is complete! The FPGA of the FE will no longer be configured successfully in case of file or transfer corruption!

The command server of the TDCM and the *pclient* program maintain the count of commands sent/received; the number of "daq" requests and replies are counted separately. The command "be cmd clr" should be sent when both the server and client side programs have been restarted, and when DAQ is inactive, to allow the direct comparison of counters at each end. The commands to clear and to get the statistics are also counted. Hence, if the user clears message count and get message statistics, the number of command message sent/received at each end will be 2.

The *pclient* program can execute scripts contained in text files. However, scripts cannot be nested, i.e. the exec command cannot be called within a script. The LOOP and NEXT commands are only available within a script and multiple loops cannot be nested.

The command "rcp" is used to copy a local file to the micro-SD card of the TDCM. Any type of file can be copied, but this command is primarily intended to update the firmware and embedded software of the TDCM with a new release. The supplied file must be a complete valid bitstream that includes the FPGA configuration, the first stage bootloader code and the application code for the embedded processor of the on-board Zynq of the TDCM. The board must be power-cycled or the POR push button must be pressed to boot the new firmware and software. If the file copy fails, or an inappropriate file is supplied, the TDCM will no longer boot properly at power-up. The micro-SD memory card will need to be removed and a valid bitstream file will need to be copied to restore operation.

The command "fe program flash" is used to download a new revision of the FPGA firmware of the front-end in its local SPI flash memory. This command only supports Spansion S25FL512S SPI flash memory. Addressing uses 24-bit, hence only the lower 16 MB are used although the device has a capacity of 64 MB. The supplied file must be in .mcs format. If the file has been prepared on a Windows machine, it must be converted to Unix format prior to download with the command "dos2unix". The user must issue the command "fe flash open" before the transfer, and should type the command "fe flash close" after the transfer has been completed. Only one FE can be selected at a time for the operations that involve the SPI flash of the FE. After a successful download, the FE must be restarted. If the download fails, incorrect file supplied, wrong format, transmission error, power cut, or other reasons, the FPGA of the FE will probably no longer configure at power-up. If this situation occurs, the SPI flash of the FE must be reprogrammed with a proven firmware file using the appropriate JTAG cable.

11.4 DATA ACQUISITION COMMANDS

The commands that control data acquisition using *pclient* are listed in Table 68.

Table 68. *pclient* data acquisition commands.

Command	Argument	Action
DAQ		Shows how much data still need to be collected to complete the current DAQ request
DAQ	<size>	User level command to request a large amount of data at once (<size> is a 64-bit integer and is expressed in bytes)
DAQ	0	Stop the current data acquisition
credits	show	Shows how many data bytes (or frames) pclient can request to each TDCM

credits	restore	Restore the initial credit count in pclient with default values
credits	restore <cnt> <thr> <unit>	Sets the credit count of pclient to <cnt>. Credits are posted to the TDCM when <thr> is reached. The unit for credits may be bytes (B) of frames (F).

The "DAQ" family of commands is used at the level of the human user of the system or within a script. The size argument specifies the total amount of data (in bytes) to collect from all active TDCMs. This amount of data requested can be up to a several tens of TBytes. The *pclient* command interpreter automatically post series of "daq" commands to the different TDCMs until the size specified by the "DAQ" command is globally reached (and is slightly exceeded).

The "credits" commands are used to diagnose and restore operation after some error communications between the TDCM and *pclient*. If *pclient* has posted to a TDCM all its credits and these have been lost for any reason, a potential dead-lock can occur: the *pclient* program cannot request data from the TDCM because it does not have any credits but the TDCM cannot send data for the same reason. Restoring credits at one end can solve this problem. Credits can also be adjusted to optimize transfers depending in different setups. System throughput can be increased up to some saturation level when more credits circulate in the system. But buffer overflow and communication errors will occur if some element in the chain cannot absorb the traffic that results.

11.5 FILE I/O COMMANDS

Pclient offers to the end-user a minimal set of commands to save the data received from the TDCMs to files. These commands are listed in Table 69.

Table 69. File I/O commands.

Command	Argument	Action
path		Shows the path for saving result files
path	<path_str>	Sets the path for saving result files to <path_str>
file_chunk		Shows the current maximum file size (1 GByte by default)
file_chunk	<size>	Sets the maximum file size to <size> (in Mbytes)
fopen		Create a file to store event data frames in binary
fopen	asc	Create a file to store event data frames in ASCII
fclose		Close the currently opened data file
LIST	ped <A>	Lists the pedestal equalization constants of all the channels of Asic <A> and save them in an interpretable script file

LIST	thr <A>	Lists the programmed threshold of all the channels of Asic <A> and save them in an interpretable script file
event_builder		Shows the mode of operation of the event builder
event_builder	<0xMode>	Sets the mode of operation of the event builder

The path command is used to define the root directory where data are saved.

When the user enters the "fopen" command, the system automatically creates a file named as follows:

```
<path_str>/Ryyyy_mm_dd-hh-mm-ss_bbb.xxx
```

where <path_str> is the relative or absolute path set by the "path" command, yyyy is the current year, mm is the month, dd is the day, hh-mm-ss is the current time, bbb is the file number within this run and xxx is "aqs" if the selected format is binary and "txt" if it is ASCII.

For basic manual data acquisition (after system configuration), the user should first do "fopen", then type "DAQ <size>" with the amount of data he would like to collect for this run. The user should then manually issue "DAQ" commands until the system indicates that there are no more data to collect (this step will be automated). If the amount of data requested exceeds the maximum file size (defined by the "file_chunk" command), *pclient* will automatically close the current file and create sub-sequent files as needed. The last file is closed by the user with the "fclose" command.

The format of the data files that are recorded by *pclient* on the DAQ PC uses the same encoding rules that are used for the communication between the TDCM and the DAQ PC. The data that is found in the file is however not the exact copy of the messages exchanged between the TDCMs and the DAQ PC. Notably, responses to configuration commands and frames that contain monitoring information are skipped and only event data frames are recorded. Some unnecessary words (e.g. the empty word at the beginning of each UDP/IP payload) are also skipped. At the beginning of a file, the run date and time is encoded in ASCII format using the required prefix. The content of that string is the following:

```
Ryyyy_mm_dd-hh_mm_ss-rrr
```

Where yyyy is the year, mm is the month, dd is the day, hh-mm-ss is the time when the file was created, and rrr is a sub-run number starting from 000 and is incremented by one for each file produced.

The "LIST" commands are used typically used after a pedestal run when the pedestal equalization constants and thresholds for zero-suppressed readout have been set and

the user want to save them to be able to re-load the same values at a later time. Pedestal and threshold settings are saved in text file which are called respectively:

```
ped_yyyy_mm_dd-hh-mm-ss.txt
thr_yyyy_mm_dd-hh-mm-ss.txt
```

Where yyyy is the year, mm is the month, dd is the day, hh-mm-ss is the current time. The "exec" command following by the pedestal or threshold file name can be used to re-load the corresponding values.

Pclient contains an experimental event builder intended for operation with multiple TDCMs. The event builder can be transparent (bit 0 of Mode = 0) or active (Bit 0 of Mode = 1). In transparent mode, the data received from several TDCMs are stored in a common file in arrival order. In the active mode, the event builder searches for event boundaries in the data received from each TDCM and groups the data of each event before they are stored to disk. At present, the event builder is very simplistic and is mostly intended for demonstration. For proper operation, it requires that the TDCM makes the end of event always appear at an end of frame. When the event builder is active, the prefix "PFX_START_OF_BUILT_EVENT" and "PFX_END_OF_BUILT_EVENT" are added to wrap the data gathered from all TDCMs for each event. The event builder can optionally check that event numbers and timestamps are identical across all TDCMs. Checking event numbers and timestamps are enabled independently by setting bit 1 and bit 2 of the event builder mode to 1 respectively. If timestamp checking is enabled, the event builder checks that all timestamps match exactly. In order to tolerate slight synchronization errors, the event builder can be set to accept events with timestamps that differ from +1 or -1 unit. This policy is enabled by setting bit 3 of event builder Mode to 1.

11.6 DECODING THE BINARY FILES RECORDED BY PCLIENT

The appropriate method for decoding a binary file acquired with *pclient* is to read this file in elementary datum of 16-bits (i.e. one unsigned short integer), and interpret each datum successively according to the format rules detailed in section 9 until the end of file is reached. Little-endian byte ordering is assumed. The interpretation of each datum may be obtained from the matching prefix header of this datum, or it may be determined by one or several of the previous data. For example, when the datum PFX_START_OF_EVENT is first encountered, the next datum is implicitly the 16-LSBs of the event time stamp, the next one is implicitly the 16 middle bits of the event time stamp, the ones that follow are the 16-MSBs of the event time stamp, the 16-LSBs of the event counter, and finally the 16-MSBs of the event counter. The type of the datum that follows the 16-MSBs of the event counter is not implicitly known. It must be determined from the datum itself by identifying a unique matching prefix. If the datum does not match any of the existing prefixes, subsequent data cannot be decoded. Ignoring this

error and attempting to interpret the next datum will most likely fail or lead to incorrect data because the non-interpreted datum may be followed by one or several data that have an implicit signification which is different from that of a possibly matching prefix. Only a subset of all the possible different types of messages defined in section 9 are stored in data files by *pclient*. Obviously, the event header, trailer, and ADC samples are recorded. On the other hand, configuration frames, monitoring frames, pedestal histograms, and dead-time histograms are not recorded in event data files by *pclient*. This may be changed in the future. Some datum have self-contained information, e.g. PFX_ADC_SAMPLE contains a complete 12-bit ADC sample, while some others provide only a piece of information that becomes complete when several data are grouped together. For example, the datum PFX_LONG_ASCII_MSG only forms an actual complete string after the following datum that contains the size of the string has been read and after the number of ASCII characters indicated by this size field have been read. Each piece of complete information is called an *item*. An *item* is built from the content of one or several elementary 16-bit datum. After interpreting a *datum*, it may be possible to retrieve a complete *item*. In other cases, one or several more data have to be processed before a new *item* is complete. Events are composed of a collection of *items* stored sequentially in the data files recorded by *pclient*.

Two example programs are provided to demonstrate the concepts of decoding these files: *preader* and *fdecoder*. It is recommended to use *fdecoder* which is more recent and clearer than *preader*. The central part of the decoder relies on the `DatumContext` structure and the associated functions `DatumContext_Init()`, `Datum_Decode()` and `Item_Print()`. The complete source code is provided and it is strongly recommended that these functions are re-used without modification in the development of tools for file format conversion (e.g. from the TDCM format to some ROOT format) or in the development of software for on-line monitoring. Several header files have to be included and only one source code file, `datum_decoder.c`, is needed. Refer to the source code for details. Dependencies are minimal and this code should be exploitable in C or C++ in any Windows or Linux environment.

To initialize the datum decoder, the function `DatumContext_Init()` is called with the required appropriate parameters. The parameter `sample_index_offset_zs` indicates how many pre-samples are recorded in zero-suppressed mode. The DAQ setting that was used when recording the file is needed to determine the appropriate offset to properly index the time buckets read from the SCA of the front-end chips. After context initialization, the function `Datum_Decode()` is called sequentially for every datum. After this function returns, the user must query the member `isItemComplete` of the structure `DatumContext` to determine if an *item* is available. If an error occurs, a negative value is returned and structure member `ErrorString` indicates the cause of the error. The function `Item_Print()` may optionally be called to output a string

representation of any complete *item* for debugging. If an *item* is complete, it must be copied elsewhere if it is desired to keep it because subsequent calls `Datum_Decode()` to will erase the current *item* to construct the following one. The above procedure is repeated for every datum found in the file to interpret until the end of file is reached. An event stored by *pclient* never spans across two files. Processing multiple files without calling `DatumContext_Init()` between them is allowed. Note that internal counters are currently limited to 32 bits unsigned integers. Although the data interpreted shall remain correct in all cases, some counters will roll over if the total amount of data read at once exceeds 4 Gbytes.

The current list of all the possible *items* that may appear in binary files produced by *pclient* is given in Table 70. Note that all types of *items* may not necessarily appear in every file. Other types of *items* may be added in the future. The hexadecimal value assigned for each *item* may be changed in future releases of this software. Hence, the user must not modify the corresponding header files to maintain compatibility and ease the integration of future extensions and improvements.

Table 70. List of *Items* that can be found in binary files produced by *pclient*.

Item	DatumContext fields	Signification
IT_UNKNOWN	none	Unknown type of item. Indicates a caveat in the code
IT_SHORT_MESSAGE	MessageString[]	Short ASCII string
IT_LONG_MESSAGE	MessageString[]	Long ASCII string
IT_DATA_FRAME	FramingVersion FrameSourceType FrameSourceId	Indicates the beginning of an Ethernet data frame that contains event data. This item is for DAQ diagnosis and can be ignored for physics data analysis.
IT_MONITORING_FRAME	FramingVersion FrameSourceType FrameSourceId	Would normally not appear in a binary data file. The code to interpreted subsequent content is currently not implemented.
IT_CONFIGURATION_FRAME	FramingVersion FrameSourceType FrameSourceId	Would normally not appear in a binary data file. The code to interpreted subsequent content is currently not implemented.
IT_END_OF_FRAME	none	Currently not used
IT_START_OF_EVENT	FramingVersion FrameSourceType FrameSourceId EventType SourceType SourceId EventNumber	Event header information. Only the header information from the BE should be kept. The event header from each FE may be present for verification debugging. Subsequent items

	EventTimeStampLsb EventTimeStampMid EventTimeStampMsb	belong to this event until the next IT_START_OF_EVENT is found.
IT_CHANNEL_HIT_COUNT	CardIndex ChipIndex ChannelHitCount	Indicates the number of channels hit in the corresponding chip. Only relevant using the AGET.
IT_LAST_CELL_READ	CardIndex ChipIndex LastCellRead	For advanced debugging. Can be safely ignored.
IT_END_OF_EVENT	SourceType SourceId EventSize	Indicates the end of the current event and its size. This must correspond to the amount of data found from the start of the event until its end.
IT_CHANNEL_HIT_HEADER	CardIndex ChipIndex ChannelIndex	Indicates to which card, chip and channel the data that follows pertains. This triplet shall be translated to the actual X and Y coordinates of the physical pad connected to that channel.
IT_TIME_BIN_INDEX	CardIndex ChipIndex ChannelIndex TimeBinIndex	In zero suppressed mode, indicates the index of the first ADC sample above threshold
IT_ADC_SAMPLE	CardIndex ChipIndex ChannelIndex RelativeSampleIndex AbsoluteSampleIndex AdcSample	Indicates the amplitude of one ADC sample. The absolute sample index indicates the actual time bucket in the SCA of the front-end ASIC. Beware that the absolute index may be negative and the relative sample index may exceed 511 depending on how many pre-samples and post-samples are recorded. Samples with a negative index are always null and should be dropped. The relative sample index is always restarted from 0 after IT_TIME_BIN_INDEX.
IT_NULL_DATUM	none	Empty datum used for padding some items to an integral number of 32-bit words.
IT_START_OF_BUILT_EVENT	none	Reserved for future use (event builder mode with multiple TDCMs)

IT_END_OF_BUILT_EVENT	none	Reserved for future use (event builder mode with multiple TDCMs)

12 Reference Documents

[1] P. Baron et al. "AFTER, an ASIC for the Readout of the Large T2K Time Projection Chambers", IEEE Transactions on Nuclear Science, volume N°55, issue 3, part 3, pp. 1744 – 1752, June 2008.

[2] S. Anvar et al., "AGET, the GET front-end ASIC, for the readout of the Time Projection Chambers used in nuclear physic experiments", in proc. IEEE Nucl. Sci. Symposium 2011, pp. 745-749.

[3] D. Baudin et al., "ASTRE: ASIC with switched capacitor array (SCA) and trigger for detector readout electronics hardened against Single Event Latchup (SEL)", in NIMA, November 2017. [Online]. Available: https://doi.org/10.1016/j.nima.2017.10.043

[4] C. Glattfelder, "Mercury ZX1 User Manual", Enclustra Gmbh, 2015. [Online]. Available: http://www.enclustra.com

[5] "7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter", Xilinx User Guide UG480, July 23, 2018.

13 DOCUMENT HISTORY

June 2016: (Version 0.0) initial release.

September 2016: (Version 0.1) documented message and data format.

October 2016: (Version 0.2) added section on FEC identification procedure.

November 2016: (Version 0.3) added section on clocking, line encoding/decoding and procedure to establish communication between the TDCM and FECs.

January 2017: (version 0.4) added TDCM register map and description of bit error rate tester.

April 2017: (version 0.5) revised FEC register map; added the list of command interpreter commands and their description.

August 2017: (version 0.6) added pictures of TDCM and detailed boot procedure. Changed TDCM Register #10 from FEM_MASKED to FE_ACTIVE (same function but all bits inverted).

September 2017: (version 0.7) changed the specification of the bit error rate tester of the S-TDCM to M-TCM link. Defined content of TDCM Register #13.

February 2018: (version 0.8) changed syntax of commands with prefix "be" and "fe" for those that apply to the back-end and front-end side respectively. Added "Single Shot Trigger" mode. Defined content of TDCM Register #14. Defined PFX_LONG_ASCII_MSG, updated Fig. 76 and described newly defined long ASCII messages.

March 2018: (version 0.9) corrected syntax of command "be rx_bert_stat". Added extended format for packet of channel data and pedestal histogram in order to support front-end cards that have up to 16 ASICs instead of 4. Added new format for FEC Register#1 to support read/write to the registers of up to 16 ASICs.

April 2018: (version 1.0) added description of the Ring Buffer Interface and dual ported memory blocks.

September 2018: (version 1.1) updated the definition of the content of TDCM Register#7. Added figure showing pedestal histogram detailed statistics frame in extended format. Defined prefix PFX_EXTD_CARD_CHIP_LAST_CELL_READ and added Fig. 79. Removed FEC_POW_INV control bit in FEC Register #1 and extended the field FEC_ENABLE to two bits. Moved field ASIC_MASK from FEC Register #1 to FEC Register #9; extended it to 16 bits instead of 4 bits.

October 2018: (version 1.1) defined prefix PFX_EXTD_PEDTHR_LIST and modified format of pedestal or threshold list accordingly to support up to 16 ASICs per front-end.

Wrote section 11. Revised document to replace "FEC" by "FE" where appropriate. Defined the command "be fe_workset" and introduced the use of the keyword "fe_workset" in several previously existing commands.

November 2018: (version 1.2) expanded section on TDC mock-up using Enclustra PE1 evaluation kit. Revised FE DPRAM map (Table 25) to include SPI Flash Controller I/O data buffers. Defined content of FE Register #12.

January 2019: (version 1.3) changed signification of TDCM to "Trigger & Data Concentrator Module" instead of "Trigger & Data Collection Module". Added Table 65 and the corresponding section. Added command "fe program flash" and the related description. Corrected the text that explains how the parity bit is computed for various messages (M-TCM to/from S-TDCM and FE to/from BE over Virtual Channel A). Added field FRA_TIMEOUT in FE Register #6 and the associated commands. Corrected Fig. 82. Added section 10.1.11 "Commands for Data Acquisition". Changed the syntax of commands to support an optional placeholder character or optional board index after the prefix "be" and "fe" and before the main body of the command.

January 2019: (version 1.4) added command "fe sfp". Updated definition of TDCM Register #14 and added command "be extra_dead_time". Defined field I2C_TARGET in FE Register #6.

April 2019: (version 1.5) added definition of FE Register #13 and #15; added definition of prefix PFX_EXTD_CARD_CHIP_CHAN_H_MD; added commands "fe fw_version" and "fe xadc".

April 2019: (version 1.6) corrected the file name of the firmware and embedded software placed on the SD memory card of the TDCM which must be called "BOOT.BIN". Revised section on *pclient*. Added section 11.6 on the decoding of binary data files acquired with *pclient*.