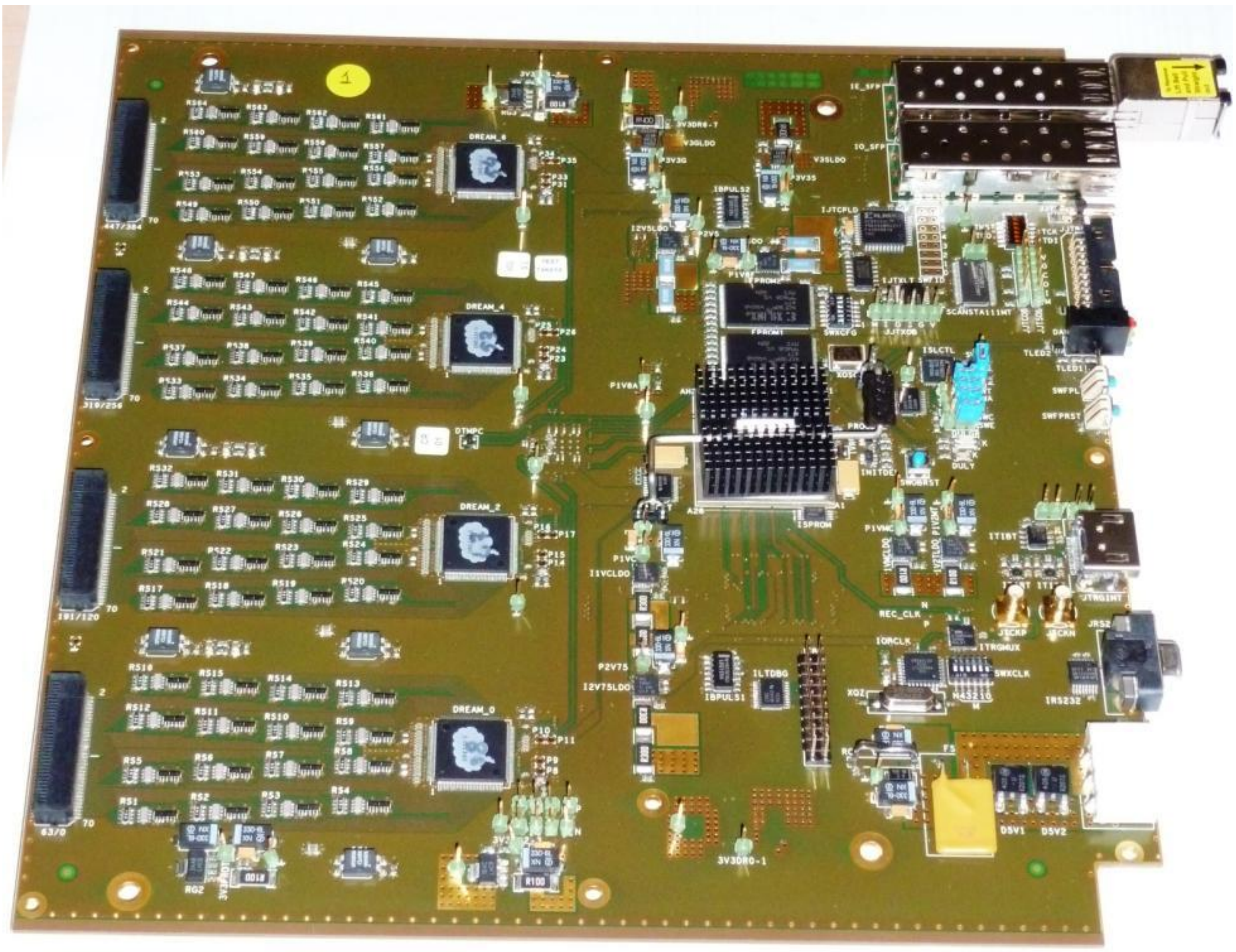
**DSM / Irfu / SEDI**
CEA Saclay
91191 Gif-sur-Yvette CEDEX
Tel: (33).1.69.08.17.96
Fax: (33).1.69.08.31.47
Email: Irakli.MANDJAVIDZE@cea.fr
Web: http://irfu.cea.fr

# FEU
# a frontend unit for Micromegas trackers

## User's Manual



# Revision History

| Date | Revision | Changes |
|---|---|---|
| 18-Mar-16 | 1.0 V0.0 | Section 3.8 on Self-Trigger module updated: a new veto register added, parameters added to distinguish combined hit propagation over the rear feu bus and over the optical link. Corresponding parameters added and updated to Appendix 1 and Appendix 2 |
| 14-Mar-16 | 1.0 V0.0 | Fig 9 and Fig 10 updated<br>Section 3.8 on Self-Trigger module updated.<br>Table 37 updated with `TrigVetoLen` parameter.<br>Table 5 updated with `SparseRd` parameter.<br>Corresponding parameters added and updated to Appendix 1 and Appendix 2 |
| 09-Feb-15 | 1.0 V0.0 | Table 35 (Section 3.9 - Dream clock module) and the first step in the initialisation sequence in Appendix 2 corrected. |
| 16-Jan-15 | 1.0 V0.0 | Changes concern Dream clock generator module. Table 2 has been changed to accommodate the new module type. Section 3.9 describing the module has been added. The corresponding parameters added to the Appendix 1. The initialisation sequence modified accordingly in Appendix 2. Xilinx WAPP878 application note on the Virtex-6 MMCM dynamic reconfiguration added to the Reference section |
| 20-Nov-14 | 1.0 V0.0 | Changes concern extended event ID and timestamp capabilities. Fig 9 and Fig 10 changed to indicate new optional fields in the Feu data packets. Table 16 has been changed to accommodate the new `EvTstExt` parameter and the corresponding parameters added to the Appendix 1, Appendix 2 |
| 23-Sep-14 | 1.0 V0.0 | Table 27 updated adding to the UDP communication channel the SFP physical layer module states and reset signal<br>3.8 section on Self-Trigger module completed with Topological trigger and corresponding parameters added to the Appendix 1, Appendix 2 |
| 03-Sep-14 | 1.0 V0.0 | Fig 10 changed according to the new format of ZS packet<br>Table 30, Appendix 1, Appendix 2: Bert parameter added to Auxiliary TI |

| | | |
|---|---|---|
| 15-Jul-14 | 1.0 V0.0 | Tables Table 14 and Table 17 corrected<br>Optical SFP transceiver status bits added to Table 21 |

# Table of Contents

## 1 INTRODUCTION

The 512-channel front end unit (FEU) electronics card, as well as the Dream ASICs [Drm] hosted on it, has been primarily developed for the DAQ system of the Clas12 Micromegas Vertex Tracker (MVT) [Cla, Mvt]. However, a flexible design allows for its use in the DAQ systems of various types of particle detectors (*e.g*. gaseous, silicon) with moderate readout rates of about 20 kHz. The FEU is responsible for pre-amplification and shaping of detector signals, for their pipeline buffering during the trigger generation process, for digitization and compression of selected event data and their delivery to the upper stages of the DAQ systems (event building, archiving). For illustration purposes the Clas12 MVT readout system is shown on Fig 1.

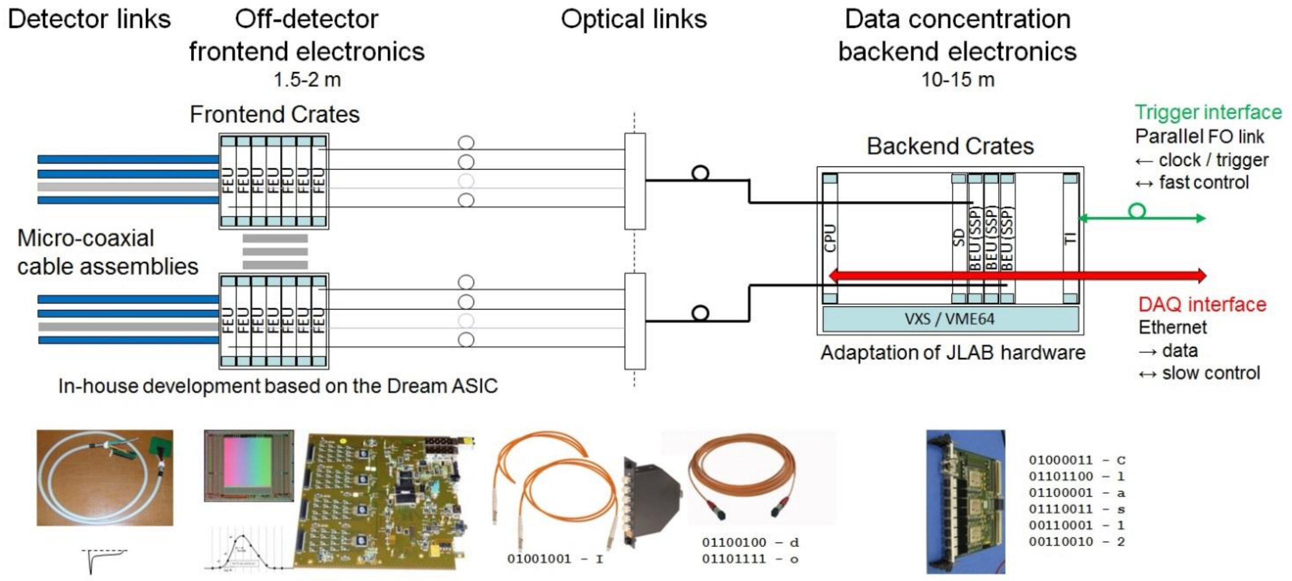


Fig 1. Clas12 MVT readout architecture

Due to very stringent space, the FEU-s are housed in the frontend crates about 1.5-2 m away from the Micromegas chambers. Detector links, implemented as 64-channel micro-coaxial cable assemblies, connect groups of chamber strips with the corresponding FEU-s. There are 8 64-channel input signal connectors on the front face of a FEU.

In the Clas12 MVT application a FEU communicates with the data concentration backend electronics over a bidirectional optical link running a proprietary communication protocol. In the upstream direction, from backend to frontend, the link encodes the Clas12 system clock and carries synchronous and asynchronous data. The synchronous data corresponds to the trigger signal and to various fast run-control commands (*e.g*. event counter reset, timestamp reset). The trigger and the fast commands are delivered to all FEU-s in a synchronous way with at least 1 ns precision. The upstream asynchronous channel carries run control commands and monitoring requests. In the downstream direction, from frontend to backend, the communication is asynchronous. The links carry event data, responses to the run control commands and monitoring information. The Clas12 MVT readout system accounts for 6 9-slot frontend crates with up to 54 FEU-s: 27 648 channels.

Alternatively, FEU-s can receive the system clock, trigger and fast command signals over an onboard RJ45 connector, implemented on its rear side and called Trigger Interface (TI). The low voltage differential signaling (LVDS) is used. The TI can also be used by FEU to output its state (*e.g*. busy, overflow warning) and/or locally generated trigger signal. Next, one of the two small form-factor pluggable (SFP) [Sfp] transceiver cages on FEU-s can be populated with an RJ45 electrical Ethernet module or an optical Gigabit Interface Converter (GBIC) module. Thus a FEU can be connected to a PC directly or via an Ethernet switch (electrical or optical). These interfaces allow the use of FEU-s in moderate size DAQ systems. An example is given on Fig 2.

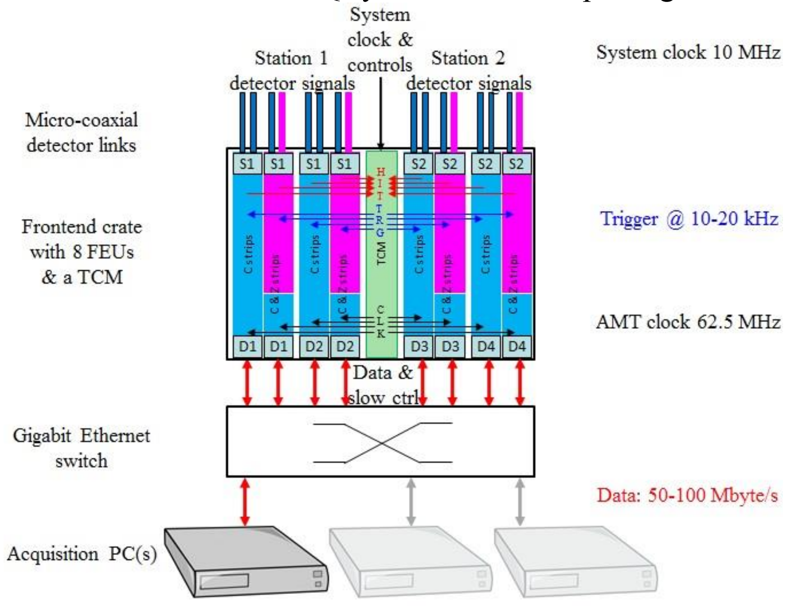


Fig 2. AMT readout architecture

The DAQ architecture is proposed for the Micromegas Tracker (AMT) of the Asacusa experiment at CERN [Asa]. As in the Clas12 MVT case, an off-detector frontend electronics must be used. The 4096 strips of the AMT detector are readout by 8 FEU-s housed in a single crate. The trigger and control module (TCM) developed for the MINOS experiment [Tcm, Min] receives the Asacusa system clock, derives a high quality 62.5 MHz clock and delivers it to FEU-s via category 5 or 6 RJ45 cables connected to their TI connectors. Over the same cables the TCM gathers trigger signals generated locally by FEU-s. The TCM can build a global trigger applying certain multiplicity criteria and send it to FEU-s. Selected event data are sent to acquisition PC(s) over a Gigabit Ethernet switch. The Gigabit Ethernet links are used for run control and monitoring of the FEU and TCM modules. The UDP/IP suite is used for the event building and control purposes. If necessary, this architecture can be extended up to 24 FEU-s (12 288 channels) – the maximum number of FEU-s a single TCM can handle.

A low level remote slow control of FEU-s is based on the 4-wire JTAG standard. Each FEU has a 24-pin 2 mm connector to access one of its 3 JTAG chains at a time. One of the chains groups a Xilinx XC6VLX75T Virtex-6 FPGA and two associated XCF32P configuration Platform flash PROM-s. This chain allows for remote programming of the FPGA and/or flash PROM-s, as well as for reading FPGA core and IO voltages and its temperature values. The second JTAG chain comprises a single MAX16031 system monitor chip. The power consumption of the module, various onboard generated voltages, the values of three temperature sensors are monitored by this chain. Yet a third JTAG chain includes an auxiliary Xilinx XC9572XL CPLD giving access to the board hardware ID, the FPGA boot status and some other additional information.

The Clas12 MVT frontend crates are placed within stringent area of restricted access. Heavy maintenance manipulations when the FEU-s will be physically accessible are foreseen only during rear (annual) shutdown periods. The nearest adequate place where a slow control PC can be installed is some 3 to 5 m away from the frontend crates. The organization of the slow control network proposed for the frontend electronics is sketched on Fig 3. A Xilinx USB programmer [Prg] or its equivalent (*e.g*. Digilent HS1, [Dig]) is attached to each frontend crate. A flat ribbon cable jumper carrying the four JTAG signals connects the programmer to all FEU-s in the crate. An USB extender is needed to control the programmer from a remote PC if the distance exceeds 3 m. An optical M2-110-10 extender from Opticis [Opt] could be a good candidate as it provides complete electrical isolation between the control PC and frontend electronics excluding ground loops. Otherwise, if the distance is less than 3 m, a galvanic isolator (like for example KXUSB-150-R2 from Keterex, [Ket]) can be deployed nearby the PC where the residual magnetic field from the Clas12 solenoid is negligible. The remote PC controls several USB programmers each connected to a frontend crate. If the number of USB ports on the PC is less than the number of crates, an USB hub can be used.

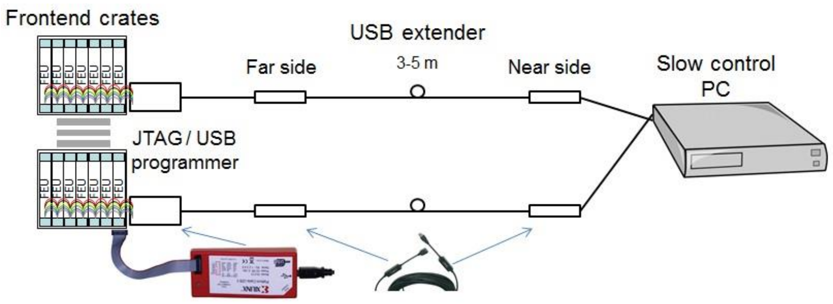


Fig 3. Slow control for the Clas12 MVT frontend electronics

The JTAB circuitry implemented on FEU-s allow for accesses to a single FEU within the crate and to the desired JTAG chain on the selected FEU. For remote firmware update operations, the control PC addresses the Virtex-6 JTAG chain of the desired FEU and executes the Xilinx iMPACT programming software. For monitoring operations, a standalone program can periodically or on user request scan a group of FEU-s establishing sequentially connections with all 3 JTAG chains, reading available information, populating specific databases with it and eventually presenting it in a user readable form.

A FEU can be powered either from either 4.3 V or 5 V supply trough a 3-pin 643488-1 power connector on its rear side, which belongs to the TE Connectivity MATE-N-LOK product line [Pcn]. Each pin of the connector is rated for 12 A. One pin is for ground. One pin is for 4.3 V power source that is intended for standard operation with an elaborated power supply like Wiener's PL506 [Win]. The onboard very low dropout (VLDO) linear regulators use this power supply to produce various voltages (*i.e*. 3.3 V, 2.5 V, 1.8 V, 1.2 V and 1 V). Alternatively, for debugging purposes, 5 Volts can be applied over the third pin from a relatively simple power module, like a PC power supply, that cannot be adjusted. Internally two rectifier Schottky diodes drop the 5 Volts to approximately 4.3 V, avoiding excessive power dissipation on the further stages of the VLDO regulators.

A FEU with its Virtex-6 FPGA running the target data acquisition firmware, with all 8 Dream-s operating in most power hungry mode and with the optical and Ethernet transceivers inserted in the two SFP cages consumes 4.8 A. A 9-slot fully populated frontend crate needs a power supply with at least 60 A capabilities capable to stand transient consumption increases at power up while FPGA-s boot their firmware.

The Clas12 MVT frontend electronics is powered from a remote supply placed some 3 to 5 m away from the crates. Voltage drop is ineluctable on the over the long power cables connecting the crates with the power source. Single PL506 modular power supply system from Wiener [Win] is used. It has 6 independent programmable power supply modules with 100 A capabilities in the output range from 2 to 7 Volts. One module powers one frontend crate. Two large cross-section power cables and two thin sense cables are used to connect a module and a crate. The power cables are chosen with the cross-section exhibiting 1 V drop over the cable length. The sense cables are needed for power regulation. The powering scheme is shown on Fig 4.

A 2-pole power distribution bus bar from Schroff is attached to each crate [Pbb]. Each of its rails is rated from 60 to100 A, depending on ambient temperature. The power and the sense cables are connected to the rails. Short two-wire jumpers deliver the 4.3 V power from the bust bar to FEU-s. The voltage drop on the jumpers is negligible.

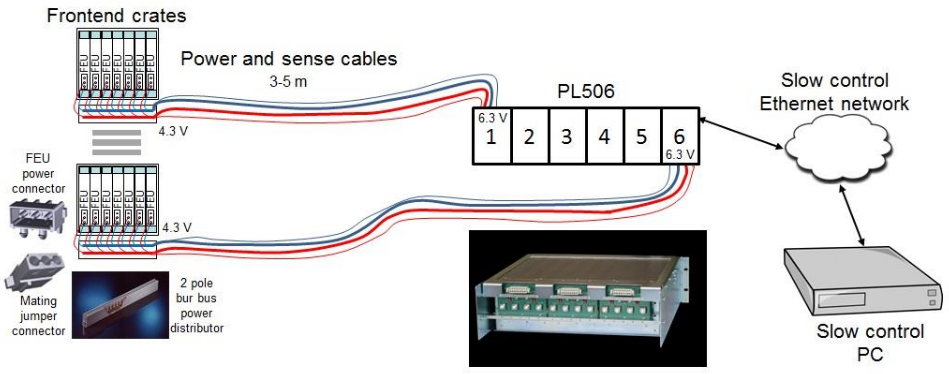


Fig 4. Powering of the Clas12 MVT frontend electronics

The desired output voltage values of the PL506 power supply is programmed by a slow control PC via an Ethernet network. The PC also monitors the power supply operation. Each power supply channel is interlocked with the cooling system of the corresponding crate.

A FEU consumes ~21 Watts. It has many sensible components such as the Dream ASICs, the ADC, the FPGA and the VLDO regulators that need to be cooled. In the environment without magnetic field cooling fans can be deployed beneath the frontend crates. This is the case for test bench systems. The fans are however prohibited for the Clas12 MVT frontend electronics as the residual magnetic field of ~1 T is too high. Cooling is achieved by pressurized air flow. The principles are shown on Fig 5.

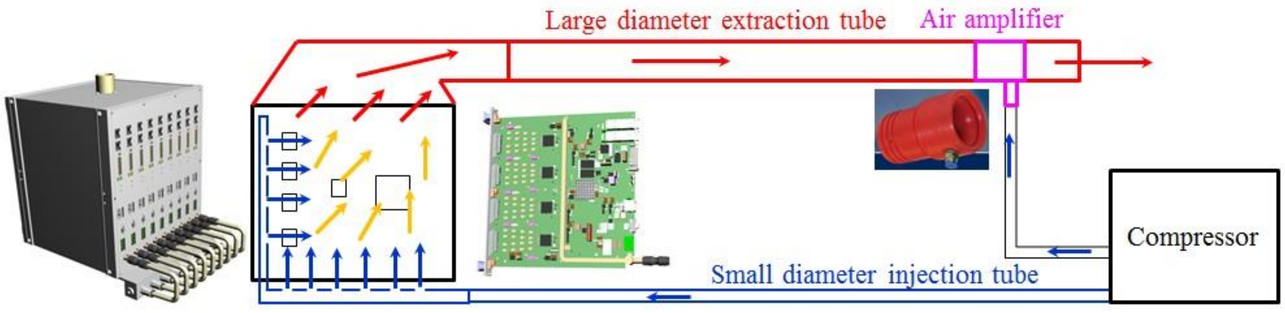


Fig 5. Cooling infrastructure for the Clas12 MVT frontend electronics

The FEU-s are equipped with a tube on each side. The tubes run nearby the sensitive components and have small diameter holes just in front of them. Pressurized air generated by a remote compressor is brought to the frontend crate by a small 8 mm$^2$ diameter flexible tube. An air distributor of the crate delivers the air to each FEU through an inlet common to its two tubes. The pressurized air flows through the tubes and escapes from the holes at a high speed. This air flux cools down the components.

Heated air must be extracted from the crates and released few meters away in the experimental cavern. This prevents increase of temperature of surrounding detectors. The heated air is pumped out of the crate through a large 30-50 mm$^2$ diameter tube. To create forced flow, a passive Ringjet air amplifier from Backair [Raa] is installed in each extraction tube. The same compressor delivers low volume of pressurized air to the amplifier's compressed air inlet. Due to Coandă effect, the air amplifier creates a high volume air movement from its large diameter inlet to its outlet.

The following sections give detailed description of the implementation and functionality of the FEU and of its interfaces.

## 2 FEU DESCRIPTION

The 512-channel FEU is a mixed analog-digital electronics board. Its main components are presented on Fig 6. The analog section comprises 8 input connectors [Mec], protection circuits, Dream ASICs and an 8-channel flash ADC [Ad9]. Four 64-channel connector-Dream pairs are placed at each side of the PCB. The inputs are connected to the Dream-s through optional protection circuits. The resistive Micromegas detectors operate without developing sparks and their strips can be directly connected to the Dream input pins without the risk of damaging the ASIC. The protection components are not soldered on the FEU-s destined to work with these detectors. On contrary, standard Micromegas detectors produce sparks that usually are harmful for the frontend electronics. During their production phase the FEU-s for standard Micromegas are equipped with protections.

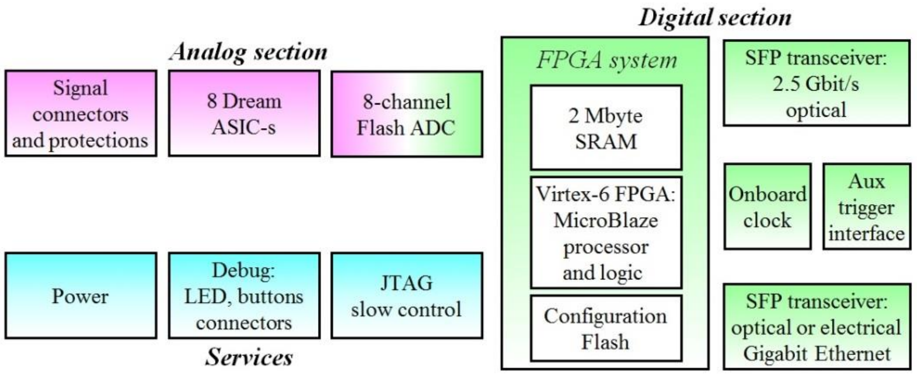

Digital section
Analog section
Signal connectors and protections
8 Dream ASIC-s
8-channel Flash ADC
FPGA system
2 Mbyte SRAM
Virtex-6 FPGA: MicroBlaze processor and logic
Configuration Flash
SFP transceiver: 2.5 Gbit/s optical
Onboard clock
Aux trigger interface
SFP transceiver: optical or electrical Gigabit Ethernet
Power
Debug: LED, buttons connectors
JTAG slow control
Services

Fig 6. FEU functional diagram

The Dream-s pre-amplify and shape the input analog signals. They continuously sample the signals and store them in a 512-cell circular memories build out of Switched Capacitor Arrays (SCA). There is a circular memory per input channel. When an active signal is applied to the Dream trigger input a desired number of cells within a programmable time period are removed from the SCA pools and do not participate anymore in the sampling process. The analog values frozen within the cells are next readout and presented to the ADC in a FIFO order: the oldest sample first and the most recent samples last. At the end of cell readout they are returned to the circular memory pool and are reused for sampling. Thus the Dream circular memories have dual functionality: a pipeline to account for trigger latency and de-randomization buffer to absorb trigger bursts during readout.

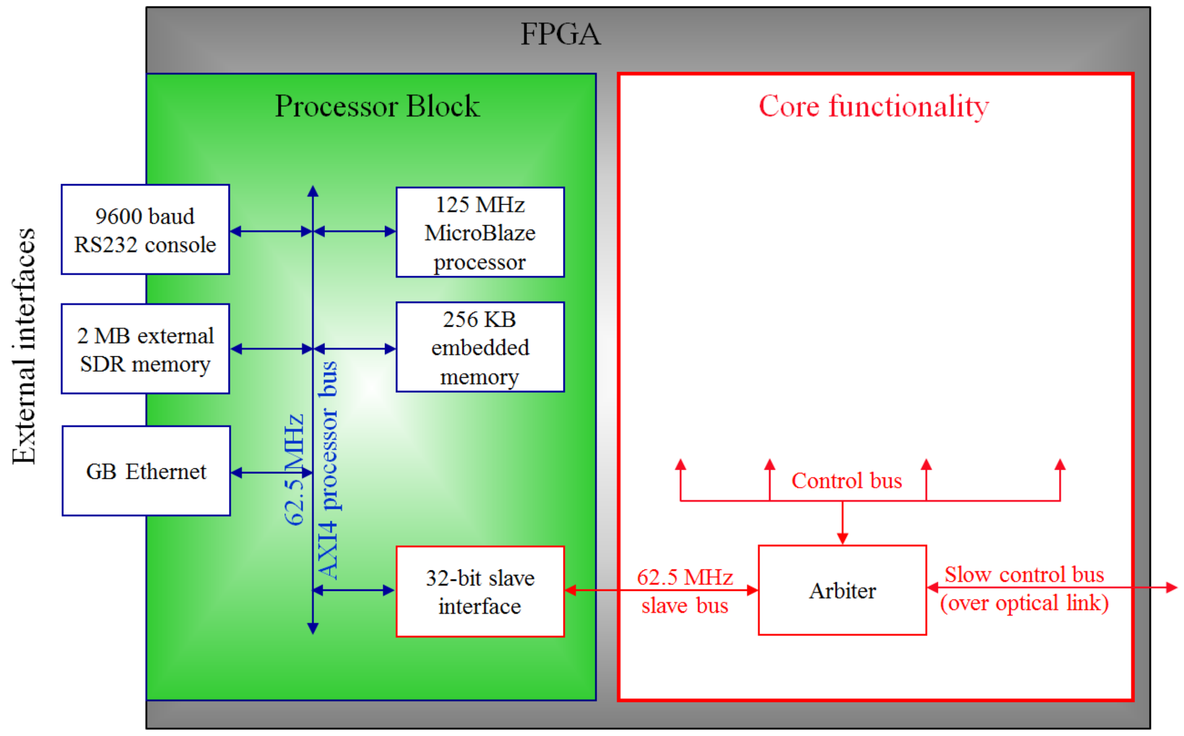


Fig 7. FEU firmware structure

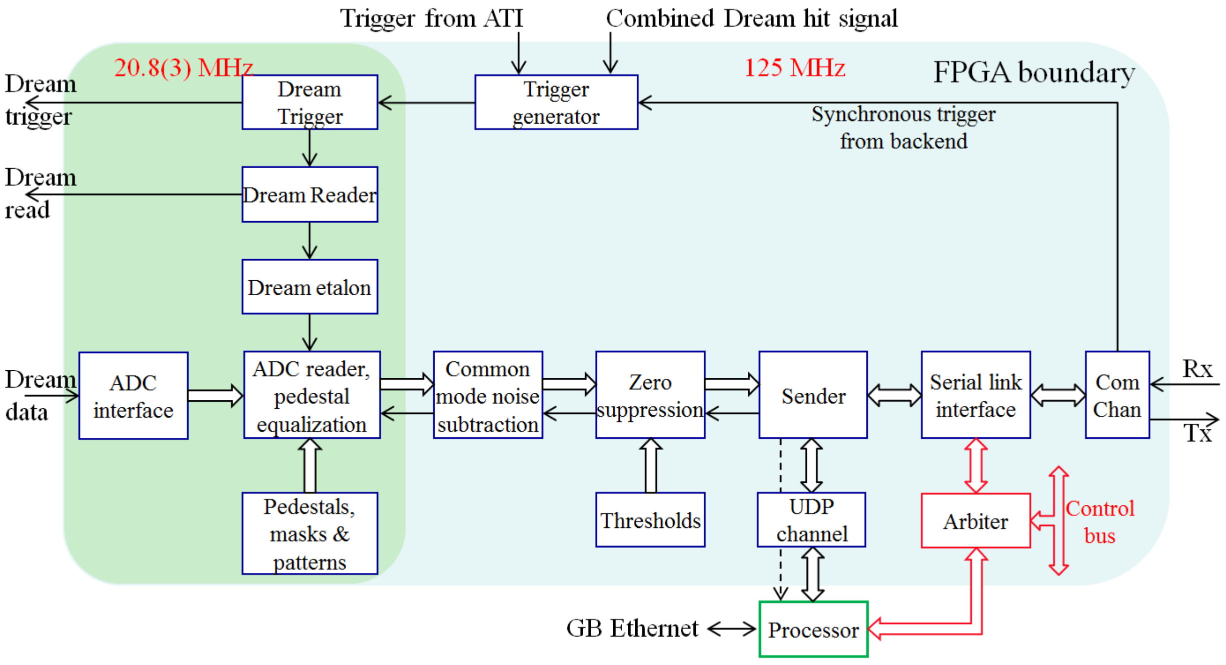
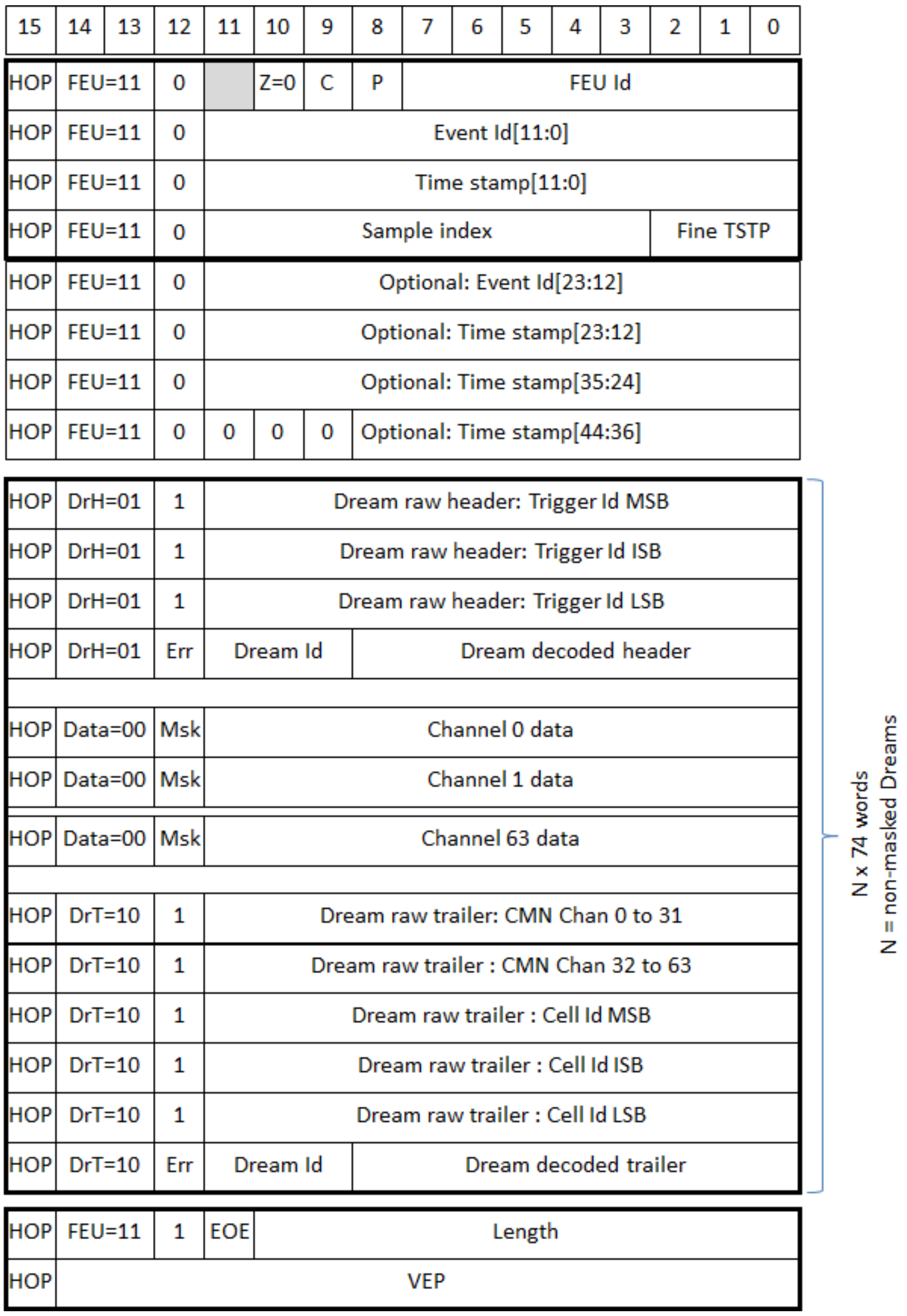
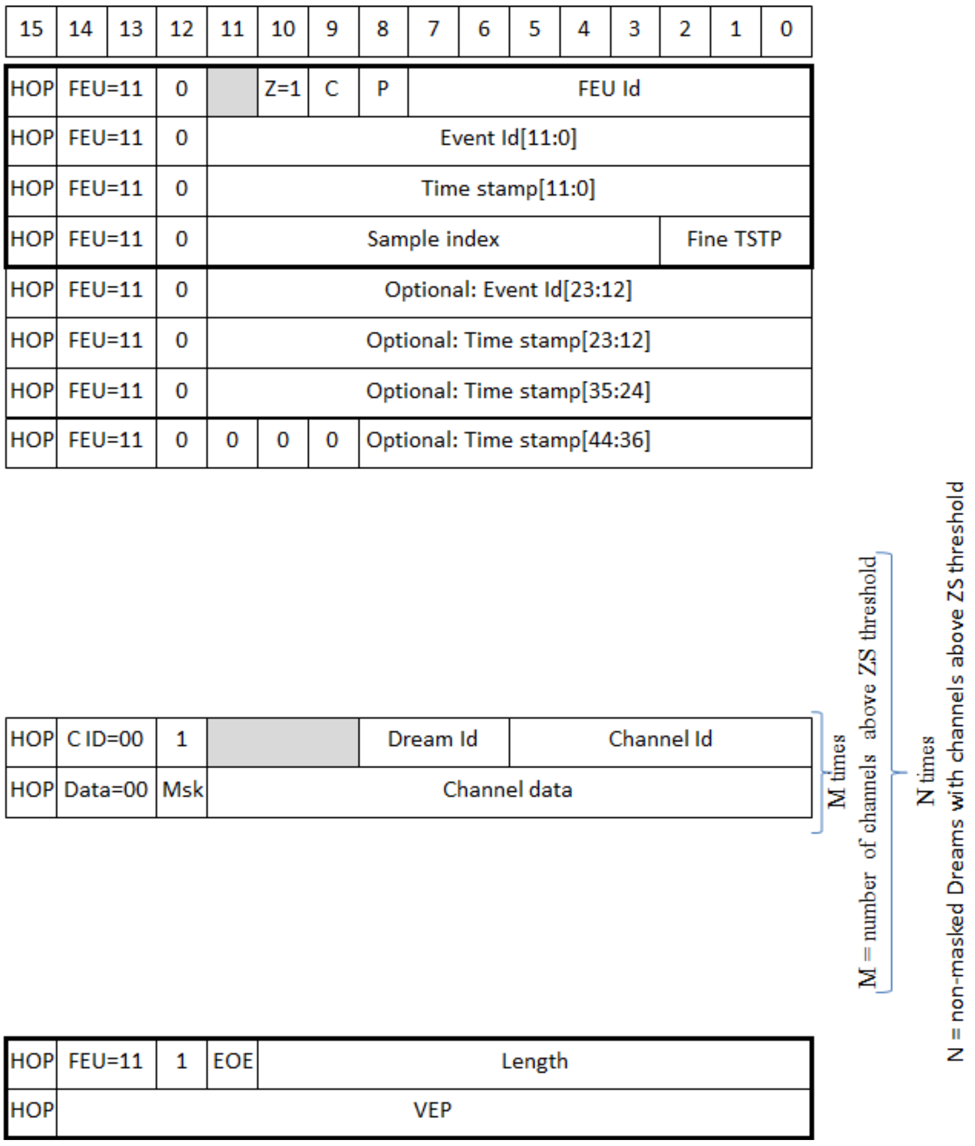


Fig 8. Modular structure of the FEU core logic



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HOP | FEU=11 | | 0 | ▨ | Z=0 | C | P | FEU Id | | | | | | | |
| HOP | FEU=11 | | 0 | Event Id[11:0] | | | | | | | | | | | |
| HOP | FEU=11 | | 0 | Time stamp[11:0] | | | | | | | | | | | |
| HOP | FEU=11 | | 0 | Sample index | | | | | | | | | Fine TSTP | | |
| HOP | FEU=11 | | 0 | Optional: Event Id[23:12] | | | | | | | | | | | |
| HOP | FEU=11 | | 0 | Optional: Time stamp[23:12] | | | | | | | | | | | |
| HOP | FEU=11 | | 0 | Optional: Time stamp[35:24] | | | | | | | | | | | |
| HOP | FEU=11 | | 0 | 0 | 0 | 0 | Optional: Time stamp[44:36] | | | | | | | | |
| HOP | DrH=01 | | 1 | Dream raw header: Trigger Id MSB | | | | | | | | | | | |
| HOP | DrH=01 | | 1 | Dream raw header: Trigger Id ISB | | | | | | | | | | | |
| HOP | DrH=01 | | 1 | Dream raw header: Trigger Id LSB | | | | | | | | | | | |
| HOP | DrH=01 | | Err | Dream Id | | | Dream decoded header | | | | | | | | |
| | | | | | | | | | | | | | | | |
| HOP | Data=00 | | Msk | Channel 0 data | | | | | | | | | | | |
| HOP | Data=00 | | Msk | Channel 1 data | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| HOP | Data=00 | | Msk | Channel 63 data | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| HOP | DrT=10 | | 1 | Dream raw trailer: CMN Chan 0 to 31 | | | | | | | | | | | |
| HOP | DrT=10 | | 1 | Dream raw trailer : CMN Chan 32 to 63 | | | | | | | | | | | |
| HOP | DrT=10 | | 1 | Dream raw trailer : Cell Id MSB | | | | | | | | | | | |
| HOP | DrT=10 | | 1 | Dream raw trailer : Cell Id ISB | | | | | | | | | | | |
| HOP | DrT=10 | | 1 | Dream raw trailer : Cell Id LSB | | | | | | | | | | | |
| HOP | DrT=10 | | Err | Dream Id | | | Dream decoded trailer | | | | | | | | |
| HOP | FEU=11 | | 1 | EOE | Length | | | | | | | | | | |
| HOP | VEP | | | | | | | | | | | | | | |

Fig 9. Non-zero-suppressed data packet



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HOP | FEU=11 | | 0 | ▨ | Z=1 | C | P | FEU Id | | | | | | | |
| HOP | FEU=11 | | 0 | Event Id[11:0] | | | | | | | | | | | |
| HOP | FEU=11 | | 0 | Time stamp[11:0] | | | | | | | | | | | |
| HOP | FEU=11 | | 0 | Sample index | | | | | | | | | Fine TSTP | | |
| HOP | FEU=11 | | 0 | Optional: Event Id[23:12] | | | | | | | | | | | |
| HOP | FEU=11 | | 0 | Optional: Time stamp[23:12] | | | | | | | | | | | |
| HOP | FEU=11 | | 0 | Optional: Time stamp[35:24] | | | | | | | | | | | |
| HOP | FEU=11 | | 0 | 0 | 0 | 0 | Optional: Time stamp[44:36] | | | | | | | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HOP | C ID=00 | | 1 | ▨ | | | Dream Id | | | Channel Id | | | | | |
| HOP | Data=00 | | Msk | Channel data | | | | | | | | | | | |

M times
M = number of channels above ZS threshold
N times
N = non-masked Dreams with channels above ZS threshold

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HOP | FEU=11 | | 1 | EOE | Length | | | | | | | | | | |
| HOP | VEP | | | | | | | | | | | | | | |

Fig 10. Zero-suppressed data packet

## 3 THE CONTROL BUS

VHDL entities that implement a given functionality of the FEU firmware form logical modules. Examples of modules are optical serial or UDP communication channels, pedestal or zero suppression threshold memories, etc. The control bus provides slave accesses to the modules. The bus is A24/D32 *i.e*. its address is 24-bit long and only 32-bit data read-write accesses are supported.

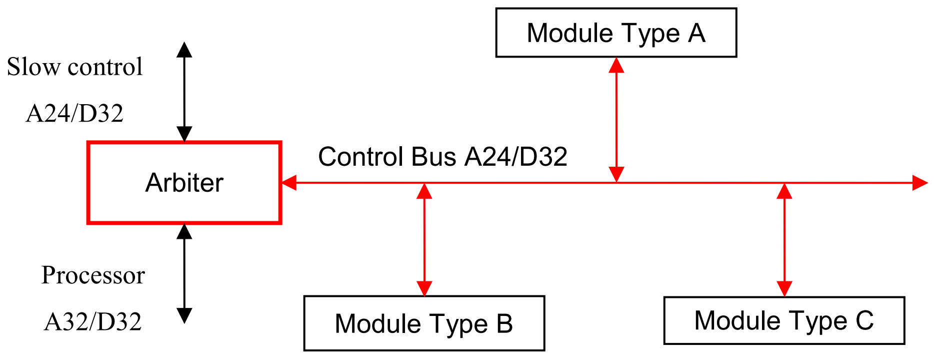


Fig 11. Modules on the Control Bus

The address space is divided in the fields shown in Table 1. As there are D32 accesses only, the two least significant bits of the addresses (0 and 1) must always be set to 0. To address the modules on the local bus their type is used in the 4 most significant address bits.

Table 1. Control Bus addresses

| Module Type | | | | Module Address Space | | | | | | | | | | | | | | | | | | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |

The CBus.h header file defines currently supported module types. Only the modules types supported by the FEU firmware are listed in Table 2.

Table 2. Module types instantiated by the FEU firmware

| Module Type | Value | Comments |
|---|---|---|
| Main | 1 | Common to several applications |
| Feu | 2 | FEU specific |
| ComChan | 3 | Optical serial link |
| Pedestal | 4 | Pedestal memories |
| Threshold | 5 | ZS threshold memories |
| UdpChan | 6 | UDP communication link |
| TrigInt | 9 | Auxiliary trigger interface |
| SelfTrig | 10 (0xA) | Self-trigger based on Dream Hit |
| DreamClock | 13 (0xD) | Sampling & read clock generator |
| TrigGen | 14 (0xE) | Trigger Generator module |

The following example shows the local bus address space of the Trigger Generator module: `0xE00000-0xEFFFFC`. Note that this address has to be complemented by the base address assigned to the core functionality block itself in the embedded processor system (the base address assigned to the 32-bit slave interface – Fig 7). If, for example, the base address is `0x80000000`, the address space of the Trigger Generator seen by the embedded processor is `0x80E00000-0x80EFFFFC`.

### 3.1 MAIN MODULE

During the R&D and prototyping phases, several versions of firmware with different functionalities have been written. Despite the differences, they had many common parts that needed configuration and monitoring. The configuration and status spaces of these parts have been grouped in a common (main) module. The main module address space is 6-bit wide with the two least significant bits set always to 0. The registers of the common main module are listed in Table 3.

Table 3. Main module registers: Control bus address space 0x100000-0x10003C

| Register | Offset |
|---|---|
| `Command` | 0x00 |
| `Config` | 0x04 |
| `TrigConfig` | 0x08 |
| `Status` | 0x0C |
| `FwRev` | 0x10 |
| `SwRev` | 0x14 |
| `TrigAcptCntr` | 0x18 |
| `TrigDropCntr` | 0x1C |
| `Error` | 0x20 |
| `LastEvent` | 0x24 |

### 3.1.1 COMMAND REGISTER

To force an action through the command register, the corresponding bit must be first set to "1", next cleared to "0".

Table 4. Command register (Control bus address: 0x100000)

| Register | Bit | Comment |
|---|---|---|
| Reset | 0 | Resets hardware and configuration parameters |
| Configure | 1 | Forces hardware to be configured (Level) |
| Run | 2 | Enables data taking (Level) |
| Pause | 3 | Suspends data taking (Level) |
| LatchStat | 4 | Latches statistics |
| ClearStat | 5 | Clears statistics |
| ClearEvtCntr | 6 | Clears event counter: next event will have ID 1 |
| ClearTimeStp | 7 | Sets timestamp counter to preprogrammed value |
| ReSync | 8 | Resets hardware preserving configuration parameters |

Usually, all configuration parameters are set by software; next the `Configure` bit is set in the command register. The software then waits until the `Configured` bit is set in the Status register.

To get coherent statistics from the FEU firmware, software must first issue the `LatchStat` command (writing 1, than 0). Once the statistics is latched, it can be read and analyzed.

### 3.1.2 CONFIGURATION REGISTER

Table 5. Config register (Control bus address: 0x100004)

| Register | Bit | Comment | Default |
|---|---|---|---|
| NbOfSamples | 7:0 | Number of samples to be read per trigger | 128 |
| DreamMask | 15:8 | 1 bit per Dream: 0-active; 1-masked | 0x00 |
| SparseRd | 18:16 | Sparse readout factor: 0 read all samples, n=[1,7] skip n samples | 0 |
| Reserved | 25-19 | Must be set to 0 | 0 |
| ClkSel | 27:26 | 0-OnBoard; 1-External; 2-Recovered clock | 0 |

The `ClkSel` parameter selects the source for the trigger interface clock. For standalone operation the 125 MHz clock produced by the FEU onboard clock synthesizer can be used (value 0 – `OnBoard`). The clock can be provided by the Auxiliary Trigger Interface (value 1 – `External`). Yet another possibility that will be ultimately used in the Clas12 Micromegas tracker readout is to use the clock recovered from the optical synchronous link that transports fast commands and trigger from the backend units to the frontend units (value 2 – `Recovered clock`).

### 3.1.3 TRIGGER LOGIC CONFIGURATION REGISTER

Table 6. TrigConfig register (Control bus address: 0x100008)

| Register | Bit | Comment | Default |
|---|---|---|---|
| TimeStamp | 11:0 | Timestamp offset | 0 |
| OvrWrnLwm | 17:12 | Overflow warning low water mark | 16 |
| OvrWrnHwm | 23:18 | Overflow warning high water mark | 24 |
| OvrThersh | 29:24 | Overflow threshold | 48 |
| LocThrot | 30 | Allows local trigger throttle | 0 |

The FEU firmware monitors the number of triggers queued in the de-randomization FIFO. The FIFO is 64 trigger deep. If the FIFO occupancy exceeds the `OvrWrnHwm` value, the firmware will raise the `OverflowWarning` flag. This can be used by the system trigger supervisor to slow down the trigger pace. The `OverflowWarning` flag is set to 0, when the trigger FIFO occupancy becomes lower than the `OvrWrnLwm` value. If the FIFO occupancy exceeds the `OvrThersh` value, the firmware raises the Overflow flag. To remove the Overflow flag and to return to normal operation state, a reset or a resynchronization procedure has to be performed.

The FEU firmware is also capable to perform local throttling of the trigger signals. If the `LocThrot` bit is set to '1', the triggers will not be accepted in the `OverflowWarning` condition. (Update)

### 3.1.4 STATUS REGISTER

This is a read-only register.

Table 7. Status register (Control bus address: 0x10000C)

| Register | Bit | Comment |
|---|---|---|
| NbOfDreams | 7:0 | Max number of Dreams supported |
| Status | 11:8 | Run control status |
| AcqFSM | 15:12 | Acquisition state machine |
| RcFSM | 19:16 | Run control state machine |
| ClocksValid | 20 | Clocks valid indicator: 1-OK, 0-Error |
| Configured | 21 | 1-All parameters set; 0-not configured |

The `NbOfDreams` field indicates how many Dreams are supported by the firmware. Normally, its value should be 8.

The `Status` is a summary that can be, in principle, communicated to the run control facilities in order to indicate the readiness of the FEU. The field can take the following values: `0x0 (Unknown0); 0x1 (OverflowWarn); 0x2 (OutOfSync); 0x4 (Busy); 0x8 (Ready); 0xC (Error); 0xF (Unknown15)`. All other codes are forbidden and should be considered as an error.

The `AcqFSM` field indicates the state of the acquisition state machine. It can take the following values: `0x0 (WaitForAcqEnable); 0x1 (WaitForTrig); 0x2 (ReadDream); 0xD (WaitEoE); 0xE (Error); 0xF (Unknown)`. Other codes may be added in future, but currently are forbidden and should be considered as an error.

The `RcFSM` field indicates the state of the run control state machine. It can take the following values: `0x0 (On); 0x1 (Init); 0x4 (InitTrigGen); 0x5 (EnbAcq); 0x6 (Idle); 0x7 (EnbTrigGen); 0x8 (Running); 0x9 (OutOfSync); 0xE (Error); 0xF (Unknown)`. All other codes are forbidden and should be considered as an error.

### 3.1.5 FIRMWARE REVISION REGISTER

This is a read-only register.

Table 8. FwRev register (Control bus address: 0x100010)

| Register | Bit | Comment |
|---|---|---|
| RevMinId | 3:0 | Minor Id |

| | | |
|---|---:|---|
| RevMajId | 7:4 | Major Id |
| Date | 25:8 | Integer in the form of "YYMMDD" |
| FwType | 27:26 | 0-Clas12; 1-Asacusa; 2-Proto; 3-TbDream |
| SerNumLsb | 31:28 | 4 LS bits of FEU serial number |

### 3.1.6 SOFTWARE REVISION REGISTER

Table 9. SwRev register (Control bus address: 0x100014)

| Register | Bit | Comment |
|---|---:|---|
| RevMinId | 3:0 | Minor Id |
| RevMajId | 7:4 | Major Id |
| Date | 25:8 | Integer in the form of "YYMMDD" |
| SwType | 27:26 | 0-Clas12; 1-Asacusa; 2-LowLevelTests; 3-TbDream |
| SerNumMsb | 31:28 | RO: 4 MS bits of FEU serial number |

This first four bit-fields of this register is filled by the application running on the embedded processor. The remote controller can read the register in order to determine the embedded software version and type. The field `SerNumMsb` is read-only.

### 3.1.7 TRIGGER ACCEPT REGISTER

This register is a 32-bit counter mapped to the control bus address 0x100018.

### 3.1.8 TRIGGER DROP REGISTER

Table 10. TrigDrop register (Control bus address: 0x10001C)

| Register | Bit | Comment |
|---|---:|---|
| TrigDropCntr | 7:0 | Drops due to close triggers |
| FifoDropCntr | 15:8 | Drops due to trigger FIFO overflow |
| TrigFifoMaxOcc | 21:16 | Trigger FIFO max occupancy |

### 3.1.9 ERROR REGISTER

Table 11. Error register (Control bus address: 0x100020)

| Register | Bit | Comment |
|---|---:|---|
| DreamRdErr | 7:0 | 1 bit per Dream: 0-OK; 1-Error |
| Reserved | 15:8 | Must be 0 |
| RdErrCntr | 23:16 | Readout error counter |
| CombErr | 24 | Set if any error occurs |

The `DreamRdErr` flags are raised if Dream de-synchronization is detected: either the Dream header or the Dream trailer does not match expected values.

### 3.1.10 LAST EVENT REGISTER

Table 12. LastEvent register (Control bus address: 0x100024)

| Register | Bit | Comment |
|---|---:|---|
| EventId | 11:0 | ID of the last event |
| TimeStamp | 23:12 | Its timestamp |
| FineTimeStamp | 26:24 | Its fine timestamp |

## 3.2 FEU MODULE

The FEU module groups configuration and monitoring registers that are proper to the frontend unit operation. Its address space is 5-bit wide with the two least significant bits set always to 0. The registers of the FEU module are listed in Table 13.

Table 13. FEU module registers: Control bus address space 0x200000-0x200018

| Register | Offset |
|---|---|
| Power | 0x00 |

| | |
|---|---|
| SlowControl | 0x04 |
| RunControl | 0x08 |
| RunStat | 0x0C |
| TrgStat | 0x10 |
| Pulser | 0x14 |
| Prescale | 0x18 |

### 3.2.1 POWER REGISTER

Table 14. Power register (Control bus address: 0x200000)

| Register | Bit | RW | Default | Comment |
|---|---|---|---|---|
| Dream | 3:0 | RW | 0 | 1 bit per a pair of Dreams: 0 -off; 1 - on |
| ProtFlt | 19:4 | RW | 0 | 1 bit per 32 channels of Dream; 0-flt; 1-gnd |
| Reserved | 23:20 | RW | 0 | |
| AdcDataValid | 24 | RO | 0 | ADC synchronization: 0 – not ready; 1 – done |
| EmbDreamRdy | 25 | RO | 0 | Embedded Dream: 0 – not ready; 1 – ready |

### 3.2.2 SLOW CONTROL REGISTER

The slow control register is used to configure various FEU onboard devices through their serial link interfaces. These are the ADC, the Dream ASICs and the MAX16031 device. Dedicated software libraries have been written in C in order to communicate with these devices.

Table 15. SlowControl register (Control bus address: 0x200004)

| Register | | RW | Bit | Default | Comment |
|---|---|---|---|---|---|
| ADC | CS | RW | 0 | 0 | |
| | Clk | RW | 1 | 0 | |
| | WrD | RW | 2 | 0 | |
| | TsD | RW | 3 | 0 | Tristate buffer control |
| Dream | CS | RW | 11:4 | 0 | |
| | Clk | RW | 12 | 0 | |
| | WrD | RW | 13 | 0 | |
| | En | RW | 14 | 0 | |
| Max16301 | Clk | RW | 15 | 0 | |
| EeProm | CS | RW | 16 | 0 | |
| | Clk | RW | 17 | 0 | |
| | WrD | RW | 18 | 0 | |
| | TsD | RW | 19 | 0 | Tristate buffer control |
| Max16301 | WrD | RW | 20 | 0 | |
| | TsD | RW | 21 | 0 | |
| Semaphore | EmbProcSem | RW | 22 | 0 | |
| | RemProcSem | RW | 23 | 0 | |
| ADC | RdD | RO | 24 | 0 | |
| Dream | RdD | RO | 25 | 0 | |
| EeProm | RdD | RO | 26 | 0 | |
| Max16301 | RdD | RO | 27 | 0 | |

The slow control register allows also for mutually excluded accesses for the embedded processor and for the remote controller acting via the optical serial link. The semaphore bits are used for this purpose. Prior to any access, the embedded processor first checks that the `RemProcSem` bit is not set, meaning the remote controller is not accessing the hardware. The embedded processor then sets the `EmbProcSem` bit and rereads again the `RemProcSem` bit. If the `RemProcSem` bit is set (remote controller has meanwhile gained access), the embedded processor clears its `EmbProcSem` bit and tries to gain the access later. Otherwise, it performs a series of requested operations and resets the `EmbProcSem` bit freeing the hardware for the remote controller accesses. The remote controller follows the same algorithm to gain an access to the hardware.

### 3.2.3 RUN CONTROL REGISTER

Table 16. RunControl register (Control bus address: 0x200008)

| Register | Bit | Default | Comment |
|---|---|---|---|
| PedSub | 0 | 0 | 0 – No; 1 - Yes |
| ComModSub | 1 | 0 | 0 – No; 1 - Yes |
| ZS | 2 | 0 | 0 – No; 1 - Yes |
| DrRawOvh | 3 | 0 | 0 – No; 1 - Yes |
| ZsChkSmp | 6:4 | 0 | Samples to be compared to threshold; < 5<br>0 – samples 1, 2<br>1 – samples 1, 2, 3<br>2 – samples 1, 2, 3, 4<br>3 – samples 1, 2, 3, 4, 5<br>4 – samples 1, 2, 3, 4, 5, 6 |
| SmpClkDbl | 7 | 0 | 0 – SmpClk=RdClk; 1 – SmpClk=2*RdClk |
| FeuId | 15-8 | 0 | |
| Rd2AdcDataDel | 20:16 | 0 | See text |
| EvTstExt | 21 | 0 | 0 – 12-bit EvId & Tstp; 1 – extended EvId & Tstp |
| DreamRdDel | 22 | 0 | 0 – no delay; 1 – Dream Read delayed by 1536 trigger clock cycles |
| CmnPedOffset | 31:23 | 0 | Pedestal values after Cmn subtraction |

The field `ZsChkSmp` determines number of samples to be compared to the zero suppression thresholds. In the current implementation, the 0th sample of channels is never compared to thresholds. If zero suppression is allowed, this parameter allows comparison of at least two and at most 6 consecutive samples, as indicated in the table.

The `Rd2AdcDataDel` field configures the delay that the logic has to wait between the Dream read signal generation and the corresponding valid ADC data. The delay is given in the Dream read clock cycles. For the 20.8(3) MHz read clock this value is usually set to 8.

The `DreamRdDel` bit is intended for tests. Usually, if no events are queued in the system, the readout of Dream ASICs starts immediately after the trigger. This is default operation with the `DreamRdDel` equal 0. For test purposes, setting the bit to 1, the very first Dream Read signal in the train can be delayed by a fixed latency. The delay is hardcoded and equals 1536 core clock cycles. For 125 MHz clock this corresponds to 12 288 ns. Similarly, for the 20.8(3) MHz sampling clock this delay corresponds to 256 samples, *i.e.* to the currently supported maximal duration of the trigger signal. At a low trigger rate, this guaranties that the Dream Read signals will never overlap with the corresponding Dream Trigger signal.

The `EvTstExt` field sets the event Id and timestamp filed width in the FEU data packets (see Fig 9 and Fig 10). The default value of the parameter is "0" and the event ID and timestamp fields are 12-bit wide. The optional FEU header words are not present. If the bit is set to "1", the optional FEU header words are included in FEU data packets. The event ID is extended to 24 bits. The event timestamp is extended to 45 bits.

### 3.2.4 RUN STATISTICS REGISTER

This is a read-only register.

Table 17. RunStat register (Control bus address: 0x20000C)

| Register | Bit | Initial | Comment |
|---|---|---|---|
| ReSync | 7:0 | 0 | ReSync command counter |
| RstEvtCntr | 15:8 | 0 | Reset event counter command counter |
| RstTstp | 23:16 | 0 | Reset timestamp command counter |

### 3.2.5 TRIGGER STATISTICS REGISTER

This is a read-only 32-bit counter that counts all received triggers. It is mapped to the control bus address 0x200010.

### 3.2.6 PULSER REGISTER

Table 18. Pulser register (Control bus address: 0x200014)

| Register | Bit | Default | Comment |
|---|---|---|---|
| Dream | 7:0 | 0 | A bit per Dream: 0 – no pulse, 1 - pulse |
| Width | 23:8 | 0 | Pulse width in trigger clock cycles |
| Enable | 24 | 0 | 0 - disable, 1 - enable |
| Go | 25 | 0 | Toggle 0-1-0 to fire pulses |

Note, that the pulser circuitry is used for test purposes. The trigger generator has a possibility to output a so called raw trigger. The raw trigger is applied to the pulser, while its delayed (pipelined copy) is used as a trigger signal for the FEU logic. The Dream-s sample the pulses generated by the raw triggers. The pulse samples are read out once the pipelined delayed triggers are received. Alternatively, the pulses can be fired by software: toggling the `Go` bit.

### 3.2.7 PRESCALE REGISTER

Table 19. Prescale register (Control bus address: 0x200018)

| Register | Bit | Default | Comment |
|---|---|---|---|
| EventPrescale | 11:0 | 1 | 12-bit prescale value |
| InterPacketDelay | 29:12 | 0 | In core clock cycles |

It is possible to limit the event data sent to the backend electronics via the optical link and/or to the control PC via the Ethernet link. Data of every $\mathtt{N}^{\mathtt{th}}$ event is sent to these destinations, where `N` is the 12-bit `EventPrescale` value.

By default, a FEU outputs the data of every event (the register is set to 1). In case of long test runs, this may result in an overwhelmingly large amount of data. Setting the `EventPrescale` to a value other than 1 decreases amount of produced data and allows for longer runs. Still, examining the collected data, it is possible to verify correct functionality of the system, as well as to analyze detector functionality over a long period of time.

Another use case is a high trigger rate run. At 20 kHz trigger rate, the throughput of the UDP link might not be enough to save all event data on the control PC. Packet losses occur at the PC side as it cannot follow the pace. The event rate can be limited to only 20 Hz setting the `EventPrescale` to 1000. In this configuration the incoming triggers are processed at 20 kHz, but events are sent to the control PC only at 20 Hz which is well within the UDP capability of modern PCs. Analysis of the stored data allows, at some extent, to verify high rate functionality of the system (*e.g*. Dream internal memory management, ADC, synchronization, data processing in FPGA, etc.).

## 3.3 OPTICAL COMMUNICATION MODULE

The ComChan module groups configuration and monitoring registers that are proper to the optical serial link communication channel operation. Its address space is 5-bit wide with the two least significant bits always set to 0. The registers of the common main module are listed in Table 20.

Table 20. ComChan module registers: CBus address space 0x300000-0x300014

| Register | Offset |
|---|---|
| CSR | 0x00 |
| TxPacketLsb | 0x04 |
| TxError | 0x08 |
| RxPacketLsb | 0x0C |
| RxSyncDataLsb | 0x10 |
| RxError | 0x14 |

### 3.3.1 CONFIGURATION AND STATUS REGISTER

Table 21. CSR register (Control bus address: 0x300000)

| Register | Bit | RW | Expected | Comment |
|---|---|---|---|---|
| PllKDet | 0 | RO | 1 | 0 - PLL is not locked; 1 - PLL locked |
| ResetDone | 1 | RO | 1 | |
| RxReSync | 2 | RO | 0 | 0 - synchronized; 1 - in ReSync state |
| RxLossOfSync | 3 | RO | 0 | 0 - synchronized; 1 - synchronization lost |
| RxSyncDone | 4 | RO | 1 | |
| RxByteAllign | 5 | RO | 0 | |
| RxAllignDone | 6 | RO | 1 | |
| RxReady | 7 | RO | 1 | 0 - not ready; 1 - ready |
| TxBufError | 8 | RO | 0 | 0 - no errors; 1 - error; reset needed |
| TxReady | 9 | RO | 1 | 0 - not ready; 1 - ready |
| PhyAbs | 10 | RO | 0 | 0 - optical transceiver present; 1 - absent |
| PhyLoS | 11 | RO | 0 | 0 - valid optical signal; 1 - no signal |
| Rsvd | 14:12 | | 0 | |
| Enable | 15 | RW | | 0 - disabled; 1 - enabled |

To enable the communication channel, the `Enable` bit must be set. For proper operation, the read-only status bits must have the expected values. Otherwise, an error should have been occurred and reset is necessary.

### 3.3.2 TX PACKET COUNTER LSB REGISTER

The transmitted packet counter is 48-bit wide. This register holds the 32 least significant bits of the counter. It is mapped to the control bus address 0x300004. This is a read-only register.

### 3.3.3 TX ERROR COUNTER REGISTER

This is a read-only register.

Table 22. TxError register (Control bus address: 0x300008)

| Register | Bit | Initial | Comment |
|---|---|---|---|
| TxPacket | 15:0 | 0 | 16 most significant bits of the Tx packet counter |
| TxError | 15:8 | 0 | 8-bit Tx error counter; 128-Overflow |

### 3.3.4 RX PACKET COUNTER LSB REGISTER

The received slow control commands packet counter is 40-bit wide. This register holds the 32 least significant bits of the counter. It is mapped to the control bus address 0x30000C. This is a read-only register.

### 3.3.5 RX SYNCHRONOUS DATA COUNTER LSB REGISTER

The received synchronous commands counter is 40-bit wide. This register holds the 32 least significant bits of the counter. It is mapped to the control bus address 0x300010. This is a read-only register.

### 3.3.6 RX ERROR COUNTER REGISTER

This is a read-only register.

Table 23. RxError register (Control bus address: 0x300014)

| Register | Bit | Initial | Comment |
|---|---|---|---|
| RxPacketMsb | 7:0 | 0 | 8 most significant bits of the Rx packet counter |
| RxSyncDataMsb | 15:8 | 0 | 8 most significant bits of the Rx sync data counter |
| RxError | 23:16 | 0 | 8-bit Rx error counter; 128-Overflow |

| | | | |
|---|---|---|---|
| RxParError | 31:24 | 0 | 8-bit Rx parity error counter; 128-Overflow |

## 3.4 PEDESTAL MEMORY MODULE

The Pedestal memory module serves three functions: program channel pedestals, mask channels, generate preprogrammed data patterns. A pedestal memory is organized as four 32-bit memory blocks. Each memory address contains pedestal values for the same channel of 2 Dreams. A pedestal value entry is 16-bit word. The pedestal value itself is 12-bit wide. 16th bit is a channel mask bit. If the bit is set, the channel is masked. The corresponding ADC value will be substituted by the 12-bit pedestal value – programmed pattern. If the 16th bit is 0, the 12-bit pedestal value will be added as a two's complement to the corresponding ADC value, thus performing pedestal correction operation.

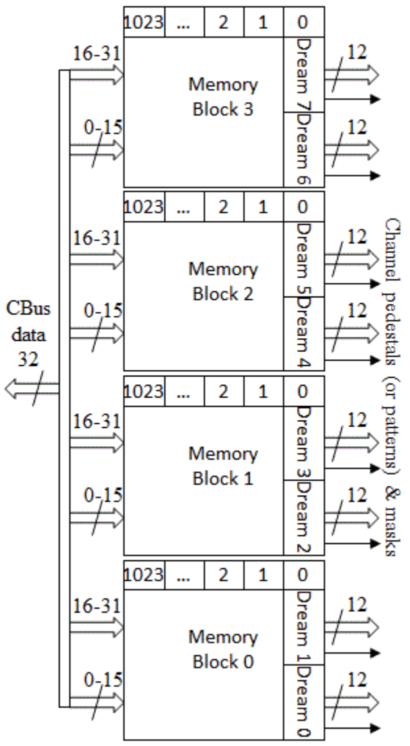


Fig 12. Pedestal memory structure

The control bus is 32-bit data wide. A single access is needed to fill an address location of the 32-bit wide memory blocks. The data bits from 0 to 15 configure pedestals, masks and patterns for Dreams with even ids: 0, 2, 4 and 6. The data bits from 16 to 31 configure pedestals, masks and patterns for Dreams with odd ids: 1, 3, 5 and 7.

For normal operation, only first 64 32-bit locations are needed in the memory blocks, one location per Dream channel. And to fill a memory block the software has to perform 64 32-bit write accesses. Once one memory block is configured, the software can configure the second block, and so on.

But the pedestal memory is organized as a circular memory of up to 1024 entries that can hold pre-calculated pattern values. Control software may set the channel mask bits to 1 and fill the pedestal fields with desired pattern values. This allows to program the pedestal memory with up to 16 (1024 / 64) predefined patterns per channel. This functionality allows testing operation of the hardware with known data values. It is important to note, that the control software must always program first N memory locations, where N is multiple of 64.

The pattern memory address space is 15-bit wide with the two least significant bits set always to 0. The bit 14 must be set to 1 indicating memory accesses. The bits 12 through 13 select memory blocks for read or write operation. The bits 2 through 11 select one of the 1024 locations to be accessed.

Table 24. Pedestal memory address space: 0x404000-0x407FFC

| **Module Type** | | | | **Module Address Space** | | | | | | | | | | | | | | | | | | **0** | **0** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| PedMem | | | | Reserved | | | | | M | Block | | Location | | | | | | | | | | 0 | 0 |
| 4 | | | | 0 | | | | | 1 | 3-0 | | 1023-0 | | | | | | | | | | 0 | |

## 3.5 THRESHOLD MEMORY MODULE

The Threshold memory module includes memory blocks that hold zero suppression threshold values. The memories are 32-bit wide. Each threshold value is 12-bit wide. Each memory address contains threshold values for the same channel of 2 Dreams. Four 32-bit memory blocks are needed to keep threshold values of 8 Dreams.

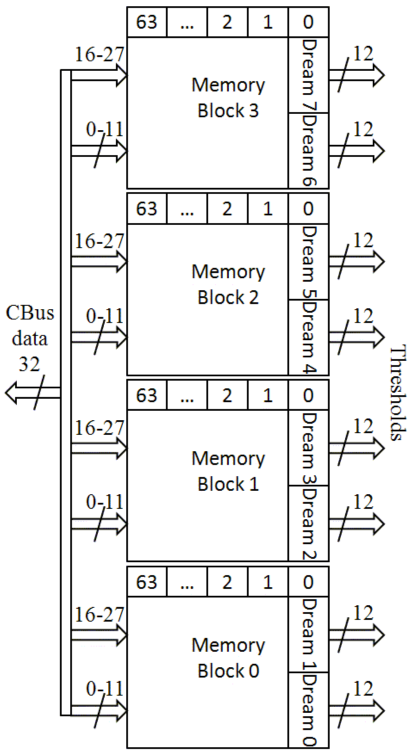


Fig 13. Threshold memory structure

The control bus is 32-bit data wide. A single access is needed to fill an address location of the 32-bit wide memory blocks. The data bits from 0 to 11 configure thresholds for Dreams with even ids: 0, 2, 4 and 6. The data bits from 16 to 27 configure thresholds for Dreams with odd ids: 1, 3, 5 and 7.

Only first 64 locations are used in the memory blocks, one location per Dream channel. To fill a memory block the software has to perform 64 accesses. Once one memory block is configured, the software can configure the second block, etc. The Threshold Memory address space is 12-bit wide with the two least significant bits set always to 0. The bit 11 must be set to 1 indicating memory accesses. The bits 8 through 10 select memory blocks for read or write operation. The bits 2 through 7 select one of the 64 locations to be accessed.

Table 25. Threshold memory address space: 0x500800-0x500FFC

| **Module Type** | | | | **Module Address Space** | | | | | | | | | | | | | | | | | | **0** | **0** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| ThreshMem | | | | Reserved | | | | | | | | M | Block | | | Location | | | | | | 0 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| 5 | 0 | 1 | 7-0 | 64-0 | 0 |

## 3.6 UDP COMMUNICATION MODULE

The UdpChan module groups configuration and monitoring registers that are proper to the Ethernet UDP link communication channel operation. Its address space is 3-bit wide with the two least significant bits set always to 0. There are only two registers that are listed in Table 26

Table 26. UdpChan module registers: CBus address space 0x600000-0x600004

| Register | Offset |
|---|---|
| CSR | 0x0 |
| TxPacketLsb | 0x4 |

### 3.6.1 CONFIGURATION AND STATUS REGISTER

Table 27. CSR register (Control bus address: 0x600000)

| Register | Bit | RW | Comment |
|---|---|---|---|
| CurBuf | 3:0 | RO | Currently active buffer out of 16 buffer pool |
| PhyAbs | 4 | RO | 0 – SFP module present; 1 – SFP module absent |
| PhyLoS | 5 | RO | 0 – SFP synchronized; 1 – SFP loss of synch |
| RdyNotify | 6 | RO | 0 – buffer is not ready; 1 – buffer can be sent |
| Enable | 7 | RW | 0 – disabled; 1 – enabled |
| State | 11:8 | RO | See Table 28 |
| PhyReset | 12 | RW | 0-1-0 transaction to reset SFP module |
| TxPacketMsb | 17:13 | RO | Most significant bits of the TxPacket counter |
| MultiPackThresh | 18-28 | RW | 0 – disabled; 1 – enabled |
| MultiPackEnable | 29 | RW | 0 – disabled; 1 – enabled |
| EnableInt | 30 | RW | 0 – disabled; 1 – enabled |
| PendingInt | 31 | RO | Pending interrupt flag |

To activate the UDO communication link, the `Enable` bit must be set. The `State` field indicates the state of the UDP channel state machine. It can take the following values:

Table 28. UDP channel states

| State | Value |
|---|---|
| WaitForUdpEnable | 0 |
| SetFreeUdpBufferAdr | 1 |
| WaitForFreeUdpBuffer | 2 |
| WaitForData | 3 |
| WriteData | 4 |
| BufferData | 5 |
| ReadCsumWord | 6 |
| ReadUdpLenWord | 7 |
| WriteUdpLen | 8 |
| WriteBufLen | 9 |
| WriteIpLen | 10 |
| WriteIpCsum | 11 |
| UdpPacketReady | 12 |
| UdpState_Unknown | 15 |

There are 16 8 Kbyte buffers shared between the firmware logic and the embedded processor. The `CurBuf` field indicates the buffer currently used. When at least one buffer contains data ready to be transmitted the bit `RdyNotify` is set. The embedded processor may poll for the bit and initiate data transmission cycles until the bit is set. Another preferred possibility is to use interrupts. To allow interrupts the `EnableInt` bit must be set. The `PendingInt` bit indicates that a ready buffer has been signaled to the embedded processor via an interrupt. This bit is cleared by the processor from the interrupt service routine (see next section).

In order to improve network performance, the UDP communication module can be instructed to use Jumbo frames. If the `MultiPackThresh*4` value exceeds the standard Ethernet MTU of 1500 bytes, the use of Jumbo frames is allowed for all communication types (slow control and data). In addition, if the `MultiPackEnable` parameter is set to 1, several data sample packets are packed in a single UDP datagram before being sent to the control PC. The discussion about the right choice of the `MultiPackThresh*4` value is given in Section 5.1.7.

The PhyAbs and PhyLoS read-only bits indicate the absence of the physical layer SFP module and it state. The SFP can be initialized and force to acquire link synchronization by toggling the PhyReset bit. The timing of the reset procedure (the reset duration and initialization latency is SFP module dependent and usually requires several hundreds of milliseconds.

### 3.6.2 TX PACKET COUNTER LSB REGISTER

The transmitted packet counter is 38-bit wide. This register holds the 32 least significant bits of the counter. It is mapped to the control bus address 0x600004.

This register is also used to clear pending interrupts. An interrupt service routine writes any value to the register to clear the pending interrupt. The counter value itself is not altered.

## 3.7 TRIGGER INTERFACE MODULE

The trigger interface module allows for communications with the Minos TCM board through the RJ45 ATI connector. The physical and application layers of the interface are described in [Tcm] and [Fem]. One LVDS line is used to receive system level clock from TCM. The second LVDS line encodes trigger and various synchronous commands. The third LVDS carries to TCM encoded information about FEU status and trigger primitives. The LVDS lines can be either AC or DC coupled, though the current FEU and TCM hardware implement DC coupling.

The FEU logic can be instructed to receive system clock from the ATI connector (see 3.1.2 and Table 5). In the current implementation, TCM distributes 100 MHz clock (the Clas12 system clock distributed to FEUs is 125 MHz). Similarly, the FEU can be programmed to process the triggers received from the ATI connector (see 3.10 and Table 36).

In the current implementation the Trigger Interface address space is 3-bit wide with all three bits set to 0. It contains single configuration and status register.

Table 29. Trigger Interface module register: CBus address space 0x900000-0x900000

| Register | Offset |
|---|---|
| CSR | 0x0 |

### 3.7.1 CONFIGURATION AND STATUS REGISTER

Table 30. CSR register (Control bus address: 0x900000)

| Register | Bit | RW | Comment |
|---|---|---|---|
| DcBalEnc | 0 | RW | When set to 1 DC balanced encoding is used on Tx |
| DcBalDec | 1 | RW | When set to 1 DC balanced encoding is expected on Rx |
| TcmIgnore | 2 | RW | When set to 1 TCM is ignored |
| Bert | 3 | RW | When set to 1 TCM link is in bit error rate test mode |
| Reserved | 4:6 | RW | |
| TcmOn | 7 | RO | When set to 1, TCM is detected |
| RxCntr | 15-8 | RO | 8-bit roll-over counters |
| RxErr | 23-16 | RO | |
| TxCntr | 31-24 | RO | |

## 3.8 SELF-TRIGGER MODULE

The self-trigger module receives the Hit signals from the 8 Dream ASICs in order to produce a trigger primitive signal if certain programmable criteria are met. The trigger primitive signal can be sent to external trigger builder logic via the RJ45 ATI connector such as for example TCM. The

FEU logic can be instructed to use the trigger primitive signal as a local trigger (see 3.10 and Table 38). The trigger primitive signal can also be propagated to other FEUs in a crate through one of the two bussed signals (ref here to Slow Control cable).

The module implements either simple coincidence logic with programmable multiplicity and coincidence window width or more elaborated topological coincidence logic. In the current implementation the self-trigger address space is 4-bit wide with the two least significant bits set always to 0. There are three registers that are listed in Table 31.

Table 31. Self-trigger module registers: CBus address space 0xA00000-0xA00008

| Register | Offset |
|---|---|
| CSR | 0x0 |
| CoincCntrLSB | 0x4 |
| Veto | 0x8 |

### 3.8.1 CONFIGURATION AND STATUS REGISTER

Table 32. CSR register (Control bus address: 0xA00000)

| Register | Bit | RW | Comment |
|---|---|---|---|
| DreamMask | 7:0 | RW | 1 bit per Dream: 0-active; 1-masked |
| Multiplicity | 10:8 | RW | Number of simultaneously active hits > than value |
| CmbHitPropagateFb | 11 | RW | Propagate active combined hit over a bussed signal |
| DreamHitWidth | 17:12 | RW | Hit width to be considered active (trig clock cycles) |
| CmbHitWidth | 23:18 | RW | Combined hit signal width to be considered active |
| TopoTrig | 24 | **RW** | 0-simple coincidence trigger; 1-topological trigger |
| CmbHitPropagateOl | 24 | **RW** | Propagate active combined hit over the optical link |
| CoinCntrMsb | 31-26 | RO | 7 MSBs of coincidence counter |

### 3.8.2 COINCIDENCE COUNTER LSB REGISTER

This is a read-only 32-bit counter that counts coincidences. It is mapped to the control bus address 0xA00004.

### 3.8.3 SELF-TRIGGER VETO DELAY REGISTER

The veto register is mapped to the control bus address 0xA00008. The 24 LS bits set the veto delay after a valid combined hit has been detected. The delay is given in the trigger clock cycles (e.g. 8 ns when working with the on board 125 MHz clock or with the 125 MHz recovered optical link clock; or 10 ns when working with the auxiliary trigger interface clock connected to the TCM).

Table 33. Veto register (Control bus address: 0xA00008)

| Register | Bit | RW | Comment |
|---|---|---|---|
| Veto | 23:0 | RW | Veto between to active combined hits (trig clock cycles) |
| RSVD | **31:24** | **RW** | For future use (Set 0) |

### 3.8.4 TOPOLOGICAL TRIGGER MEMORY

The topological trigger is implemented by means of an asymmetric dual port memory that holds pre-calculated trigger patterns. For the trigger logic the memory is 4-bit wide and 256-location deep. The topological trigger logic addresses the pattern memory with the Hit signals of the eight Dreams. The trigger is generated if any bit in the addressed location is set to 1.

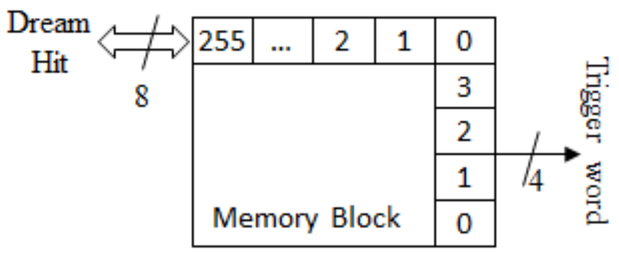


Fig 14. Topological trigger logic

Consider an example when the coincidence of hit signals is required from at least 4 Dream-s. When the topological trigger is not needed (the TopoTrig parameter set to 0), the simple coincidence logic will generate the trigger for any four Dream Hit coincidences. If however the trigger should be generated when two of the four Dreams must belong to the group 0 to 3 and the other two to the group from 4 to 7, the topological trigger has to be used. To satisfy the requirement, the trigger pattern memory must be filled accordingly. Clearly, the hit pattern 00001111 does not satisfy to the requested topology as all four active hit signals originate from the Dream in group 0 to 3. Therefore, the memory location accessed by the trigger logic port at the 0x0F address must contain 0. On the other hand, the hit pattern 11000011 satisfies the criteria and the coincidence trigger must be generated. The memory location at address 0xC3 must be preprogrammed to hold 1 in any bit of the 4-bit wide trigger word.

The Control Bus sees the topological trigger pattern buffer as a 32-bit wide 32-word deep memory. To fill the memory the software has to perform 32 accesses.

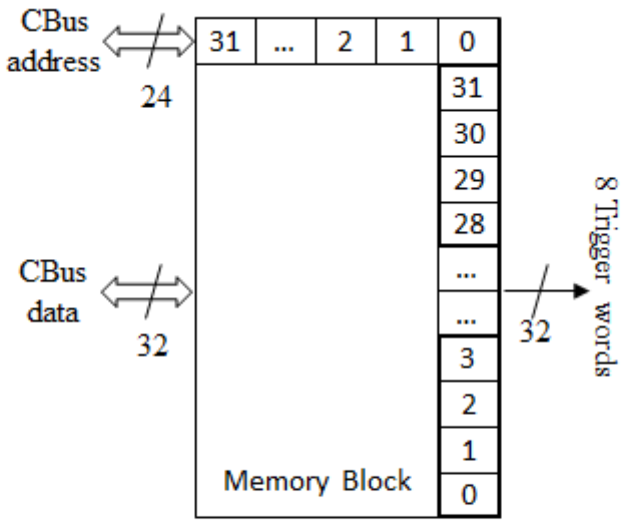


Fig 15. Topological trigger pattern memory structure

The pattern memory address space is 8-bit wide with the two least significant bits set always to 0. The bit 7 must be set to 1 indicating memory accesses. The bits 2 through 6 select one of the 32 memory locations to be accessed.

Table 34. Topological trigger pattern memory address space: 0xA00080-0xA000FC

| Module Type | | | | Module Address Space | | | | | | | | | | | | | | | | | | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| SelfTrig | | | | Reserved | | | | | | | | | | | | | | M | Location | | | 0 | 0 |
| 0xA (10) | | | | 0 | | | | | | | | | | | | **1** | 0-31 | | | | | 0 | |

As mentioned above, the 256-deep 4-bit pattern memory is programmed with thirty-two 32-bit write operations over the Control Bus. Each 32-bit word holds eight 4-bit trigger words.

Considering the above examples, all four bits of the trigger word 15 (starting from 0) must be set to 0 in order to exclude the hit coincidence 00001111 from the topological trigger generation. The word 15 is most significant nibble of in the memory location addressed 0xA00084, so the most significant bit of the addressed 32-bit word must be set to 0. Similarly, to generate the topological trigger for the 11000011 coincidence pattern, at least one bit of the trigger word 195 must be set to 1: (most significant nibble of the 32-bit word addressed as 0xA000E4).

## 3.9 DREAM CLOCK MODULE

The Dream clock generator module address space is 9-bit wide with the two least significant bits set always to 0. Six registers are needed to configure clock frequencies and phases. The register mapping is shown in Table 35. All registers are 16-bit wide.

Table 35. Dream clock module 16-bit registers: CBus address space 0xD00000-0xD001FC

| Register | Offset |
|---|---|
| RdClk_Freq | 0x20 |
| WrClk_Freq | 0x28 |
| WrClk_Phase | 0x2C |
| AdcClk_Freq | 0x30 |
| AdcClk_Phase | 0x34 |
| PwrBits | 0xA0 |

The Dream clock generator module maps the dynamic reconfiguration port registers of the Xilinx Virtex-6 FPGA MMCM core [Mmc].

This is a delicate part and detailed discussion will be added later with admissible ratios between trigger, Dream read and sampling clocks frequencies and Dream read and sampling phases. Note that the ADC and Dream readout clock frequencies must be equal.

## 3.10 TRIGGER GENERATOR MODULE

The trigger generator module produces a trigger signal either from an external trigger signal or from a signal produced by the trigger generation logic proper to the module. There can be up to 8 trigger sources: trigger generated based on Dream hit signals, external asynchronous signals, external signals brought to the firmware trigger clock domain, trigger signals detected by any other logic blocks of the firmware (typically, the trigger received over the synchronous fixed latency optical link), triggers generated by software, constant rate triggers generated at various frequencies, triggers produced by the internal trigger pattern memory, or a negative exponential random trigger generator.

The trigger generator address space is 13-bit wide with the two least significant bits set always to 0. The address space is divided into memory and register spaces. The latter has two register shown in Table 36.

Table 36. Trigger Generator module registers: CBus address space 0xE00000-0xE00004

| Register | Offset |
|---|---|
| CSR | 0x0 |
| Cntr | 0x4 |

### 3.10.1 CONFIGURATION AND STATUS REGISTER

Table 37. CSR register (Control bus address: 0xE00000)

| Register | Bit | RW | Comment |
|---|---|---|---|
| Rate | 1:0 | RW | Const: 0-0Hz, 1-1Hz; 2-10Hz; 3-100Hz<br>NegExp: 0-20Hz, 1-200Hz, 2-2kHz, 3-20kHz |
| Src | 4:2 | RW | See text |
| TrigPipeLen | 16:5 | RW | See text |
| TrigVetoLen | 26:17 | RW | Veto delay in trigger clock periods |

| | | | |
|---|---|---|---|
| Init | 27 | RW | 0-1-0: initialize with the parameters |
| Enable | 28 | RW | 0-disable; 1-enable |
| Trig | 29 | RW | 0-1-0 generates trigger if Src = Tg_Src_Soft |
| Ready | 30 | RO | 0-not ready; 1-ready |
| Enabled | 31 | RO | 0-not enabled; 1-enabled |

The `Src` filed can take one of the following values:

Table 38. Trigger sources

| Register | value | Comment |
|---|---|---|
| Tg_Src_Int | 0 | Input from the FPGA logic |
| Tg_Src_ExtAsyn | 1 | External asynchronous signal |
| Tg_Src_ExtSyn | 2 | External signal brought to the core clock domain |
| Tg_Src_SelfTrig | 3 | Dream hit based trigger |
| Tg_Src_Soft | 4 | Software generated trigger |
| Tg_Src_Constant | 5 | Internally generated constant rate trigger |
| Tg_Src_Memory | 6 | Internally generated trigger from memory pattern |
| Tg_Src_NegExp | 7 | Internally generated random trigger |

The trigger generator produces two trigger signals: a so called raw trigger and its delayed copy. The time interval between the two signals is determined by the `TrigPipeLen` field. The trigger clock of 125 MHz or 100 MHz) is used to produce the delay. In normal operation, this value has to be set to 1: the system trigger propagates directly to the readout logic. For test purposes, when pulse generation circuitry is used, the `TrigPipeLen` field can be set to any value from 1 to 4095. In case of 125 MHz trigger clock this will emulate the system trigger pipeline from 8 ns to about 32 µs. For 100 MHz trigger clock the depth of the emulated pipeline range from 10 ns to 40 µs.

The trigger generator module is usually initialized and enabled by en external run control state machine. But re-initialization can also be done by writing 0-1-0 sequence to the `Init` bit. Similarly, the trigger generation can be enabled by software setting the `Enable` bit to 1.

Setting the `Src` parameter to the `Tg_Src_Soft`, `Tg_Src_Constant`, `Tg_Src_Memory` or `Tg_Src_NegExp` value makes internal trigger generation logic active. When the `Src` parameter equals `Tg_Src_Soft`, software can generate triggers writing 0-1-0 sequences to the `Trig` bit. When `Src` parameter equals `Tg_Src_Constant`, equidistant triggers are generated with the rate determined by the `Rate` parameter: 1, 10 or 100 Hz. When `Src` parameter equals `Tg_Src_NegExp`, random triggers with negative exponential intervals are generated with the rate determined by the `Rate` parameter: 0.02, 0.2, 2 and 20 kHz. When `Src` parameter equals `Tg_Src_Memory`, the triggers are generated from the trigger pattern memory. This allows for emulation of trigger bursts.

### 3.10.2 COUNTER REGISTER

This is a read-only 32-bit counter that counts all generated triggers. It is mapped to the control bus address 0xE00004.

### 3.10.3 TRIGGER PATTERN MEMORY

The trigger generator module includes circular memory block that holds pre-calculated trigger patterns. The memory is 32 Kbit deep. The trigger generation logic circles through the 32768 locations of the memory and generates a trigger signal any time the location value is set to 1.

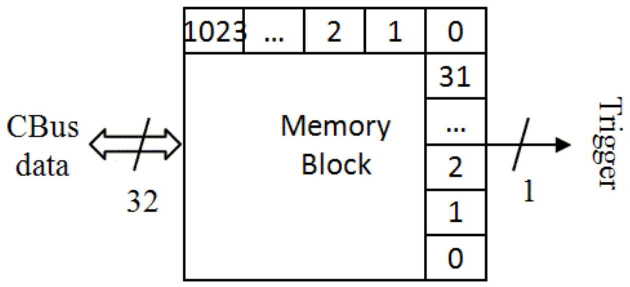


Fig 16. Trigger pattern memory structure

The Control Bus sees the circular memory as a 32-bit 1024-word deep memory. To fill the memory the software has to perform 1024 accesses. The trigger pattern memory address space is 13-bit wide with the two least significant bits set always to 0. The bit 12 must be set to 1 indicating memory accesses. The bits 2 through 11 select one of the 1024 locations to be accessed.

Table 39. Trigger pattern memory address space: 0xE01000-0xE01FFC

| **Module Type** | | | | **Module Address Space** | | | | | | | | | | | | | | | | | | **0** | **0** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| TrigGen | | | | Reserved | | | | | | | M | Location | | | | | | | | | | 0 | 0 |
| 0xE (14) | | | | 0 | | | | | | | 1 | 0-1023 | | | | | | | | | | 0 | |

# 4 Optical Communication Channel

In the readout system of the CLAS12 Micromegas tracker, serial optical communication links are established between the backend data concentrator units and the frontend electronics. A backend unit communicates with up to 32 front end units. A FEU has a connection with only one BEU.

In the downstream direction, from BEU to FEU, the link is used to send synchronous commands such as trigger, event counter reset, timestamp reset, resynchronization, etc. The transmission latency in this direction is kept constant at least within 1 ns accuracy. In the upstream direction, from FEU to BEU, the serial links transports event data.

The serial link is also used for slow control and monitoring of the front end units. A BEU can send asynchronously slow control packets requesting read/write transaction from/to FEU registers. The FEU acknowledges completion of the operation and if needed responds with its results.

Thus, in the downstream direction from BEU to FEU, the serial link transports slow control commands and synchronous commands. The format adopted is similar to the one used for the TPC readout of the T2K experiment [T2K]. It is shown on Fig 17.

| Slow Control Path | Parity bit | Synchronous Path |
|---|---|---|
| 15 … 8 | 7 | 6 … 0 |
| Comma (0x3C-K28.1) | | D0.0 |
| Comma (0x3C-K28.1) | | D0.0 |
| **Slow control header** | **P** | 0 |
| **Address (23..16)** | **P** | 0 |
| **Address (15..8)** | **P** | 0 |
| **Address (7..0)** | **P** | 0 |
| **Data (31..24)** | **P** | **Synchronous command** |
| **Data (23..16)** | **P** | 0 |
| **Data (15..8)** | **P** | 0 |
| **Data (7..0)** | **P** | 0 |
| Comma (0x3C-K28.1) | | D0.0 |
| Comma (0x3C-K28.1) | | D0.0 |
| Comma (0x3C-K28.1) | **P** | **Synchronous command** |
| Comma (0x3C-K28.1) | | D0.0 |

Fig 17. Multiplexing of slow control and synchronous commands

The data path is 16-bit wide. In absence of data, the sender (BEU) continuously transmits a particular code (hex 3C00) keeping the link synchronized. The synchronous command is transmitted in bits 0 through 6. The format of synchronous commands is shown on Fig 18. Currently, only four out of seven bits are defined. This will most probably be changed in future.

| 6 | 5 | 4 | 3 2 | 1 | 0 |
|---|---|---|---|---|---|
| **ReSync** | Reserved | **Trigger** | Reserved | **RstTstp** | **RstEvtCntr** |

Fig 18. Synchronous command

The slow control commands occupy bits from 8 through 15. A slow control command consists of a succession of 8 bytes (Fig 17). It starts with a header byte, followed by 3 address bytes and 4 data bytes. For read operations the data bytes are ignored. The slow control header is shown on Fig 19. The command index is a 5-bit code (counter) that command initiator BEU sends to the command destination FEU and expects to receive it back unchanged with acknowledgement. Currently, only point-to-point communications are supported (bit 14 set to 0).

| 15 | 14 | 13 | 12 … 8 |
|---|---|---|---|
| Reserved | **P2P=0 / Broadcast=1** | **Action: Rd=1 / Wr=0** | **Command index** |

Fig 19. Slow control command header

The point-to-point commands are always acknowledged by FEUs via the upstream channel. The slow control response / acknowledgement packet format is shown on Fig 20. As for data packets, the words are 16-bit wide. In absence of data, the upstream link kept synchronized by continuous transmission of the synchronization code hex 3C00. The start of packet code in the least significant

8 bits indicates beginning of the FEU data. The slow control response composed of a header word, two address words and two data words, followed with 8 padding words. The FEU data packets are terminated by two CRC32 words and an end of packet code placed in the most significant 8 bits.

| 15 8 | 7 0 |
|---|---|
| Comma (0x3C-K28.1) | D0.0 |
| Comma (0x3C-K28.1) | D0.0 |
| Comma (0x3C-K28.1) | **Start Of Packet (SOP: 0xFE-K30.7)** |
| **Slow control header** | |
| **0x00 & Address (23..16) (echo)** | |
| **Address (15..0) (echo)** | |
| **Data (31..16)** | |
| **Data (15..0)** | |
| **Padding #0** | |
| … | |
| **Padding #7** | |
| **CRC32** | |
| **CRC32** | |
| **End Of Packet (EOP: 0xF7-K23.7)** | D0.0 |
| Comma (0x3C-K28.1) | D0.0 |
| Comma (0x3C-K28.1) | D0.0 |

Fig 20. Slow control command response

The slow control response header echoes the command index and the action fields and places the operation completion flag in the bit 6 of the word (figure 10).

| 15 7 | 6 | 5 | 4 0 |
|---|---|---|---|
| 0 | **Ack: OK=0 / Err=1** | **Action: Rd=1 / Wr=0 (echo)** | **Command index (echo)** |

Fig 21. Slow control command response header

Note that as for the slow control response, the FEU event data packets are delimited by the SOP and EOP codes and protected by the CRC32 code. The leading zeroes in the bits 15 through 7 of the response header word allow distinguishing FEU slow control response packets from the FEU event data packets. The latter have bits 14, 13 and 12 of their first word set to the FEU header identification code 6 (Fig 9 and Fig 10).

## 5 UDP/IP COMMUNICATION WITH THE FEU EMBEDDED PROCESSOR

The Microblaze processor embedded within the FEU Virtex6 FPGA runs a standalone "C" application with an IP networking support based on the LwIp library [Lwi].

For run control parameters settings and monitoring, a remote PC establishes a UDP/IP socket connection with FEU and communicates with the Microblaze processor based on the client-server paradigm. The remote controller acts as a client sending various requests to the FEU embedded processor that behaves as the server, processes the requests and responds with the results of the processing. A remote PC may control several FEU-s. A pair of an IP address and a UDP port identifies uniquely the FEU server. The FEU embedded processor creates a UDP/IP socket on the UDP port and listens for incoming requests.

For data acquisition, a remote PC establishes yet another UDP/IP socket connection with the FEU. This UDP/IP data channel between the remote PC and the FEU is established upon a special request from the PC coming via the UDP/IP slow control channel. During the data acquisition process, the FEU pushes produced data packets (see Fig 9 and Fig 10) towards the control PC without expecting any confirmation of the data reception. The data loss is therefore possible.

To establish the UDP/IP data channel the remote PC must communicate its MAC and IP addresses and the data UDP port. The remote PC may receive data from several FEU-s. As in the case of the slow control UDP/IP connections, a pair of an IP address and a UDP port identifies uniquely the FEU data producer. Upon the request, the FEU embedded processor creates a UDP/IP data socket on the UDP port that will be used for data transfers.

To guarantee unique sets of IP addresses and UDP ports, the address 0 of the FEU EEPROM contains a unique 8-bit FEU ID value. The FEU IP address is a sum of the base IP address and the FEU ID. The UDP slow control port is fixed to be UDP slow control base port (1300) + FEU ID. Similarly, the UDP data port is determined as UDP data base port (1200) + FEU ID.

Example below illustrates UDP slow control and data connections between a control PC and a set of four FEU modules. The base IP address is set to 192.168.10.12. The four FEU-s have their IDs set to 1, 2, 3 and 4 respectively. The control PC has its local IP address set to 192.168.10.11.

Table 40. Example of multiple FEU-s controlled by single PC

| | | IP | Slow control UDP Port | Data UDP Port |
|---|---|---|---|---|
| FEU ID | 1 | 192.168.10.13 | 1301 | 1201 |
| | 2 | 192.168.10.14 | 1302 | 1202 |
| | 3 | 192.168.10.15 | 1303 | 1203 |
| | 4 | 192.168.10.16 | 1304 | 1204 |
| Control PC | | 192.168.10.11 | 1300 | 1200 |

### 5.1 UDP/IP SLOW CONTROL REQUEST/RESPONSES

A remote controller PC and a FEU exchange character strings that code request messages and corresponding responses. The response message contains a copy of the request followed by either the request completion status information (*e.g.* success/fail for write requests) or the retrieved data (*e.g.* for read requests). The response message can be quite large (*e.g.* reading of memory or statistics). The buffers to hold the response messages should be large enough. The list of slow control commands recognized by the embedded processor is listed in Table 41.

Table 41. Slow control commands

| Command | Comment |
|---|---|
| help | List recognized commands |
| peek | Read register |

| | |
|---|---|
| peekm | Read N registers or a memory block |
| poke | Write register |
| pokem | Write N registers or a memory block |
| pokef | Write a bit filed to a register |
| poket | toggle a bit filed in a register |
| UdpConnect | Establish fast UDP connection for data transfer |
| GetStat | Get statistics |
| DreamRead | Read a Dream(s) register |
| DreamWrite | Write to a Dream(s) register |
| AdcRead | Read an ADC register |
| AdcWrite | Write to an ADC register |
| EePromRead | Read from an EeProm address |
| EePromWrite | Write to an EeProm addess |
| MaxRead | Read a register from the Max16031 slow control device |
| MaxWrite | Write a register from the Max16031 slow control device |

#### 5.1.1 PEEK COMMAND

The format of the peek command is: `peek 0xAdd`, where 0xAdd is a hexadecimal control bus address. The format of the response message is `peek 0xAdd = 0xVal`, where the read value is given in the hexadecimal notation.

#### 5.1.2 PEEKM COMMAND

The format of the peek command is: `peek 0xAdd N`, where 0xAdd is a hexadecimal control bus base address and N is a decimal value of number of consecutive 32-bit address locations to read. The format of the response message is `peek 0xAdd 0xVal1 0xVal2 … 0xValN`, where the read values are given in the hexadecimal notation.

#### 5.1.3 POKE COMMAND

The format of the poke command is: `poke 0xAdd 0xWrVal`, where 0xAdd is a hexadecimal control bus address and 0xWrVal is a hexadecimal value to be written in the register. The format of the response message is `poke 0xAdd 0xWrVal = 0xRdVal`, where the 0xRdVal is the value read from the register after the write operation. It is given in the hexadecimal notation.

#### 5.1.4 POKEM COMMAND

The format of the poke command is: `poke 0xAdd HexWrVal1 HexWrVal2 … HexWrValn`, where 0xAdd is a control bus base address in the format 0x12345678 (eight digits after leading 0x) and HexWrVal parameters are values to be written in the consecutive 32-bit register given in 8-digit hexadecimal notation but without the leading 0x symbols (*e.g.* `pokem 0x00E01000 fedcba90 fedcba91 fedcba92 fedcba93`). The format of the response message is `poke 0xAdd HexWrVal1 HexWrVal2 … 0xRdVal1 0xRdVal2 … 0xRdValn`, where the 0xRdVal are the values read from the registers after the write operations. They are given in the hexadecimal notation with leading 0x symbols.

#### 5.1.5 POKEF COMMAND

The format of the pokef command is: `"poke 0xAdd 0xFld 0xWrVal"` , where 0xAdd is a hexadecimal control bus address, 0xFld is a hexadecimal bit filed value and 0xWrVal is a hexadecimal value to be written in the register bits indicated by the bit field parameter. The command is executed as follows: first the register is read, next the read value is modified according the bit field and the write value, the modified value is written to the register, and at last, the register is re-read to return the new value. The format of the response message is `"pokef 0xAdd 0xFld`

`0xWrVal = 0xRdVal"`, where the 0xRdVal is the value read from the register after the write operation. It is given in the hexadecimal notation.

### 5.1.6 POKET COMMAND

The format of the poket command is: `"poke 0xAdd 0xFld 0xWrVal"` , where 0xAdd is a hexadecimal control bus address, 0xFld is a hexadecimal bit filed value and 0xWrVal is a hexadecimal value to toggle the register bits indicated by the bit field parameter. The command is executed as follows: first the register is read, next the read value is modified according the bit field and the write value, the modified value is written to the register, then the initial value of the register is written again and at last, the register is re-read to return the new value. The format of the response message is `"poket 0xAdd 0xFld 0xWrVal = 0xRdVal"`, where the 0xRdVal is the value read from the register after the last write operation. It is given in the hexadecimal notation.

### 5.1.7 UDPCONNECT COMMAND

The format of the UdpConnect command is: `"UdpConnect MacAdr UdpPort IpAdr MultiPackEnb MultiPackThresh"`, where the `MacAdr`, `UdpPort` and `IpAd` are the parameters proper to the remote controller PC. The `MacAdr` is given in the following hexadecimal notation: "h1:h2:h3:h4:h5:h6". The `UdpPort` is given in decimal notation. The `IpAd` is given in the following decimal notation "d1.d2.d3.d4".

The `MultiPackEnb` parameter takes 0 or 1 value and instructs the FEU hardware to pack multiple event data packets (see Fig 9 and Fig 10) in a single UDP buffer to be sent to the data acquisition PC. The `MultiPackThresh` parameter, expressed in bytes, limits the UDP datagram size. The FEU logic fills a UDP buffer with complete data packets and when its size exeeds the `MultiPackThresh` value, the buffer is sent to the PC. These last two parameters allow to profit from the Jumbo frame capabilities of the Ethernet network interfaces. The network performance is especially improved for zero-suppressed data. Many small size data packets are packed in a single large size Jumbo frame before being sent out.

In case of success the format of the response message is `"UdpConnect MacAdr UdpPort IpAdr MultiPackEnb MultiPackThresh: D_RetCode_Sucsess"`. In case of failure the response message is of the form `"UdpConnect MacAdr UdpPort IpAdr MultiPackEnb MultiPackThresh: 'cause of failure message'"`.

The `MultiPackThresh` parameter value can be calculated as follows: "MultiPackThresh" = "JumboFrameSize" - "18-byte EthernetOverhead" - "28-byte Udp/IpHeader" - "2-byte DataAlignment" - "MaxDataPacketSize". Or,

"MultiPackThresh" = "JumboFrameSize" - 48 - "MaxDataPacketSize".

The "JumboFrameSize" value is the minimum between the Jumbo frame size supported by the network and 8 kbyte supported by the FEU firmware (see 3.6).

The size of longest data packet depends on various FEU configuration parameters set by the Config register of the Main module and the RunControl register (Table 16) of the FEU module mapped respectively to the 0x100004 and 0x200008 Control Bus addresses. The number of active Dreams is determined by the `DreamMask` parameter in the Config register (Table 5). The `ZS` parameter determines if zero-suppression has to be applied on channel data. The `DrRawOvh` parameter determines if the Dream raw headers and trailers have to be included in the data packets.

The maximum number of bytes, which a non-zero-suppressed packet can contain, is equal to 2*[4+NbOfActiveDreams*(3*`DrRawOvh`+1+64+5*`DrRawOvh`+1)+2]. The longest zero-suppressed packet can be obtained when all channels are above the threshold. Its size can be calculated as 2*[4+NbOfActiveDreams*(3*`DrRawOvh`+1+2*64+5*`DrRawOvh`+1)+2]. Assuming that none of the 8 Dreams are masked and that their raw headers and trailers are not required in data the biggest non-zero-suppressed and zero-suppressed packets are 1068 and 2092 bytes long. If the Dream overhead is required the figures are 1196 bytes and 2220 bytes respectively.

The following is an example of the command: "`UdpConnect 00:00:CA:FE:FA:DE 1200 192.168.10.10 1 4888`".

### 5.1.8 GETSTAT COMMAND

The format of the GetStat command is simply: "`GetStat`". Upon reception of the message the embedded processor executes a statistics acquisition sequence, executes a function transforming the register values to a string of a predefined format and returns the string as a response. The response message is of the form "`GetStat 'Statistics string'`". The remote controller PC needs to extract the statistics string and dump it either on the screen or in a file.

### 5.1.9 DREAMREAD COMMAND

The format of the DreamRead command is: "`DreamRead DreamId RegAdr`". The DreamId can take values from 0 to 7 or a wild character "*". In this latter case the action will be performed for all active, non-masked Dream ASICs. The Dream ASICs are masked through the configuration register of the Main module mapped to the control bus address 0x100004 (see Table 5). The RegAdr is a decimal address of the Dream register that can take values from 1 to 12 (see [Drm]).

The format of the response message depends if the Dream is masked or not. For the non-masked Dream the response is "`DreamRead DreamId RegAdr : D(EffectiveId) 0xRdVal1 0xRdVal2 0xRdVal3 0xRdVal4`". The `EffectiveId` is ID of the Dream. The command always returns four 16-bit hexadecimal words. For the 16-bit registers only the first word is significant. For the 32-bit registers the first and the second words have to be considered with the most significant bits placed in the first word. For the 64-bit registers all four values are meaningful with the most significant bits placed in the very first word.

For the masked Dream ASICs the response message has the following format: "`DreamRead DreamId RegAdr : D(EffectiveId) : masked`". In case of failure the response message has the following format: "`DreamRead DreamId RegAdr : Spi_DreamRegRead failed with 'cause of failure message'`".

### 5.1.10 DREAMWRITE COMMAND

The format of the DreamRead command is: "`DreamWrite DreamId RegAdr 0xWrVal1 0xWrVal2 0xWrVal3 0xWrVal4`". The DreamId can take values from 0 to 7 or a wild character "*". In this latter case the action will be performed for all active, non-masked Dream ASICs. The Dream ASICs are masked through the configuration register of the Main module mapped to the control bus address 0x100004 (see Table 5). The RegAdr is a decimal address of the Dream register that can take values from 1 to 12 (see [Drm]). The command always takes four 16-bit hexadecimal WrVal values. For the 16-bit registers only the first word is significant. For the 32-bit registers the first and the second words are considered with the most significant bits placed in the first word. For the 64-bit registers all four values are meaningful with the most significant bits placed in the very first word. The unused words should be set to 0.

The format of the response message depends if the Dream is masked or not. For the non-masked Dream the response is "`DreamWrite DreamId RegAdr : D(EffectiveId) : D_RetCode_Sucsess`". The `EffectiveId` is ID of the Dream. For the masked Dream ASICs the response message has the following format: "`DreamWrite DreamId RegAdr :`

`D(EffectiveId) : masked"`. In case of failure the response message has the following format: `"DreamRead DreamId RegAdr : Spi_DreamRegWrite failed with 'cause of failure message'"`.

### 5.1.11 ADCREAD COMMAND

The format of the AdcRead command is: `"AdcRead 0xRegAdr"`. The 0xRegAdr is an 8-bit hexadecimal address of the ADC register (see [Ad9]). In case of success the format of the response message is `"AdcRead 0xRegAdr = 0xRdVal"`, where the 0xRdVal is an 8-bit hexadecimal value. In case of failure the response message has the following format: `"AdcRead 0xRegAdr : Spi_AdcRegRead failed with 'cause of failure message'"`.

### 5.1.12 ADCWRITE COMMAND

The format of the AdcWrite command is: `"AdcWrite 0xRegAdr 0xWrVal"`. The 0xRegAdr is an 8-bit hexadecimal address of the ADC register (see [Ad9]) and the 0xWrVal is an 8-bit hexadecimal value to be written in the register. In case of success the format of the response message is `"AdcRead 0xRegAdr 0xWrVal : D_RetCode_Sucsess"`. In case of failure the response message has the following format: `"AdcWrite 0xRegAdr 0xWrVal : Spi_AdcRegWrite failed with 'cause of failure message'"`.

### 5.1.13 EEPROMREAD COMMAND

The format of the EePromRead command is: `"EePromRead 0xRegAdr"`. The 0xRegAdr is an 8-bit hexadecimal address of the serial EEPROM (see [M24]). In case of success the format of the response message is `"EePromRead 0xRegAdr = 0xRdVal"`, where the 0xRdVal is an 8-bit hexadecimal value. In case of failure the response message has the following format: `"EePromRead 0xRegAdr : I2C_Eeprom_ReadByte failed with 'cause of failure message'"`.

### 5.1.14 EEPROMWRITE COMMAND

The format of the EePromWrite command is: `"EePromWrite 0xRegAdr 0xWrVal"`. The 0xRegAdr is an 8-bit hexadecimal address of the serial EEPROM (see [M24]) and the 0xWrVal is an 8-bit hexadecimal value to be written in the register. In case of success the format of the response message is `"EePromWrite 0xRegAdr 0xWrVal : D_RetCode_Sucsess`. In case of failure the response message has the following format: `"EePromWrite 0xRegAdr 0xWrVal : I2C_Eeprom_WriteByte failed with 'cause of failure message'"`.

### 5.1.15 MAXREAD COMMAND

The format of the MaxRead command is: `"MaxRead 0xRegAdr"`. The 0xRegAdr is an 8-bit hexadecimal address of the Max16031 system monitor device (see [Max]). In case of success the format of the response message is `"MaxRead 0xRegAdr = 0xRdVal"`, where the 0xRdVal is an 8-bit hexadecimal value. In case of failure the response message has the following format: `"MaxRead 0xRegAdr : I2C_Max16031_ReadByte failed with 'cause of failure message'"`.

### 5.1.16 MAXWRITE COMMAND

The format of the MaxWrite command is: `"MaxWrite 0xRegAdr 0xWrVal"`. The 0xRegAdr is an 8-bit hexadecimal address of the Max16031 system monitor (see [Max]) and the 0xWrVal is an 8-bit hexadecimal value to be written in the register. In case of success the format of the response message is ″`MaxWrite 0xRegAdr 0xWrVal : D_RetCode_Sucsess`. In case of failure the response message has the following format: ″`MaxWrite 0xRegAdr 0xWrVal: I2C_Max16031_WriteByte failed with ‘cause of failure message’`″.

## 5.2 UDP/IP SLOW CONTROL SINGLE-CHARACTER COMMANDS

Apart from the request / respond messages described in the previous section, a control PC can communicate to a FEU single-character commands. The FEU performs the actions corresponding to the command and echoes back the character. The following single-character commands are recognized:

Table 42. Single character commands

| **Command** | **Comment** |
|---|---|
| Q | Embedded software stops to run (debugging, should not be used) |
| S | Embedded software sleep mode: stop periodic printing of statistics on RS232 console (debugging, should not be used) |
| s | Embedded software awaken mode: do periodic printing of statistics on RS232 console (debugging, should not be used) |
| **R** | Reset the hardware |
| I | Initialize with local parameters (debugging, should not be used) |
| **i** | Back to run control state "Init": reconfiguration possible |
| **G** | Start: accept and process triggers |
| **g** | Stop: stop trigger processing |
| **P** | Pause: suspend trigger processing |
| **p** | Resume: resume trigger processing |
| **C** | Clear statistics |
| **T** | Generate software trigger |
| L | Print Ethernet link level statistics on RS232 console: debugging - should not be used |
| M | Print Monitoring information on RS232 console: debugging - should not be used |

# Appendix 1. Example of Configuration Parameters file

```
#/*
#--------------------------------------------------------------------------------
#-- Company:        IRFU / CEA Saclay
#-- Engineers:      Irakli.MANDJAVIDZE@cea.fr (IM)
#--
#-- Project Name:   Clas12 Micromegas Vertex Tracker
#-- Design Name:    Clas12 Dream testbench software
#--
#-- Module Name:    TbConfig.cfg
#-- Description:    Sample Test bench configuration file
#--
#-- Target Devices: ASCII file
#-- Tool versions:  To be read by test bench software
#--
#-- Create Date:    0.0 2011/10/14 IM
#-- Revision:
#--
#-- Comments:
#--
#--------------------------------------------------------------------------------
#*/
# Format
# Configuration entries are organized in lines
# # - is comment line
# * - is wildcard, "Feu *", means all FEUs; "Dream *" means all Dreams

########################################
# Feu Ids and IPs
########################################
# Tb Dream or to be used for the very first connection
Feu 0 Feu_RunCtrl_Id            0
Feu 0 NetChan_Ip                192.168.10.12

# Preseries FEU-s
Feu 1 Feu_RunCtrl_Id            1
Feu 1 NetChan_Ip                192.168.10.13

Feu 2 Feu_RunCtrl_Id            2
Feu 2 NetChan_Ip                192.168.10.14

Feu 3 Feu_RunCtrl_Id            3
Feu 3 NetChan_Ip                192.168.10.15

Feu 4 Feu_RunCtrl_Id            4
Feu 4 NetChan_Ip                192.168.10.16

Feu 5 Feu_RunCtrl_Id            5
Feu 5 NetChan_Ip                192.168.10.17

Feu 6 Feu_RunCtrl_Id            6
Feu 6 NetChan_Ip                192.168.10.18

Feu 7 Feu_RunCtrl_Id            7
Feu 7 NetChan_Ip                192.168.10.19

Feu 8 Feu_RunCtrl_Id            8
Feu 8 NetChan_Ip                192.168.10.20

Feu 9 Feu_RunCtrl_Id            9
Feu 9 NetChan_Ip                192.168.10.21

Feu 10 Feu_RunCtrl_Id           10
Feu 10 NetChan_Ip               192.168.10.22

Feu 11 Feu_RunCtrl_Id           11
Feu 11 NetChan_Ip               192.168.10.23
Feu 12 Feu_RunCtrl_Id          12
Feu 12 NetChan_Ip              192.168.10.24

#Prototype FEU-s
Feu 13 Feu_RunCtrl_Id          231
Feu 13 NetChan_Ip              192.168.10.243
Feu 14 Feu_RunCtrl_Id          232
Feu 14 NetChan_Ip              192.168.10.244

#Dream Testbench
Feu 15 Feu_RunCtrl_Id          230
Feu 15 NetChan_Ip              192.168.10.242

#######################################
# Trigger Generator configuration register
# CBus address: 0xE00000
#######################################
#TrigPipeLen | Src | Rate |
#    16-5    | 4-2 | 1-0  |
#-- Src: Tg_Src_Int=0;  Tg_Src_ExtAsyn=1;  Tg_Src_ExtSyn=2; Tg_Src_PushButton=3;
#        Tg_Src_Soft=4; Tg_Src_Constant=5; Tg_Src_Memory=6; Tg_Src_NegExp=7
#-- RateConst: 0-0Hz; 1-1Hz; 2-10Hz; 3-100Hz
#-- Rate NegExp: 0.02, 02, 2, 20 kHz
Feu * Trig_Conf_Rate          2
Feu * Trig_Conf_Src           Tg_Src_ExtSyn
Feu * Trig_Conf_TrigPipeLen   240
# Associated parameter: file to fill trigger pattern memory for Tg_Src_Memory
# Trigger pattern memory CBus address range 0xE01000-0xE01FFC
Feu * Trig_Conf_File          None

#######################################
# Main common configuration register
# CBus address: 0x100004
#######################################
#-- Trigger interface clock source
#--   OnBoardClk  = '00' - On-board clock
#--   TrgIfConClk = '01' - Clock from on-board trigger interface connector
#--   RecClk      = '1x' - Recovered Rocket IO clock
#-- Mask
#--    a bit per Dream, 1 - masked, 0 - active
#--    Up to 8 Dreams, hense 8 bits
#-- NbOfSamples
#--    Current absolute maximum for Dreams is 255 cells / channel, hense 8 bits
#--
#-- |27-26 |25-19|  18-16 |15-8|    7-0    |
#-- |ClkSel| RSVD|SparseRd|Mask|NbOfSamples|
#######################################
Feu * Main_Conf_ClkSel       TrgIfConClk
Feu * Main_Conf_DreamMask    0x00
Feu * Main_Conf_SparseRd       0
Feu * Main_Conf_Samples       16

#######################################
# Main Trigger logic register
# CBus address: 0x100008
#######################################
#-- Configuration register for Trigger logic
#-- |    29-24  |    23-18  |    17-12  |    11-0   |
#-- | OvrThersh | OvrWrnHwm | OvrWrnLwm | TimeStamp |
#######################################
Feu * Main_Trig_TimeStamp  0
Feu * Main_Trig_OvrWrnLwm  4
Feu * Main_Trig_OvrWrnHwm 10
Feu * Main_Trig_OvrThersh 13
Feu * Main_Trig_LocThrot   1

#######################################
#-- FEU PowerUp Register
#-- CBus address: 0x200000
```

```
#######################################
#--| Prot |     Dream     |
#--| Flt  |8-7|6-5|4-3|2-1|
#--| 19-4 | 3 | 2 | 1 | 0 |
#######################################
Feu * Feu_Pwr_Dream    0xF
Feu * Feu_Pwr_PrtFlt   0xFFFF

#######################################
#-- FEU Run Control Register
#-- CBus address: 0x200004
#######################################
#--| CMN  |Read |EvTst| ADC   |     |SmpClk|         Algo                       |
#--|Offset|Delay| Ext |DataRdy|FeuId|  Dbl |ZsChkSmp|DrOvh| ZS |ComModSub|PedSub|
#--|31-23 |  22 |  21 | 20-16 |15-8 |   7  |  6-4   |  3  |  2 |    1    |  0   |
#######################################
Feu * Feu_RunCtrl_Pd             0
Feu * Feu_RunCtrl_CM             0
Feu * Feu_RunCtrl_ZS             0
Feu * Feu_RunCtrl_DrOvr          0
Feu * Feu_RunCtrl_RdDel          1
Feu * Feu_RunCtrl_ZsChkSmp       3
Feu * Feu_RunCtrl_CmOffset     256
Feu * Feu_RunCtrl_Id            -1
Feu * Feu_RunCtrl_AdcDatRdyDel   8
Feu * Feu_RunCtrl_EvTstExt       0
Feu * Feu_RunCtrl_DrDblSmpClk    0
# Associated parameters to fill
# Pedestal memory 0x404000-0x407FFC
Feu * Feu_RunCtrl_PdFile         None
# Threshold memoriy 0x500800-0x500FFC
Feu * Feu_RunCtrl_ZsFile         None

#######################################
#-- FEU Pulser Register
#-- CBus address: 0x200014
#######################################
#--|TestFunc|Rsvd|DacTrigVal| Rsvd|DacBaseVal|
#--|   31   | 30 |  29-16   |15-14|   13-0   |
#######################################
Feu * Feu_Pulser_DreamTst     0x00
Feu * Feu_Pulser_PulseWid     512
Feu * Feu_Pulser_Enable       1

#######################################
#-- FEU PreScale Register
#-- CBus address: 0x200018
#######################################
#-- InterPacket |  Event |
#--   Delay     |Prescale|
#--   29-12     |  11-0  |
#-- Event prescale: Every Nth packet will be sent to backend
#-- InterPacketDelay: Deley between two packets sent to backend (in nanoseconds)
#--                   The register is configured in core clock cycles: 125 MHz (8 ns)
#--                   Software writes InterPacketDelay / 8 in the register
#######################################
Feu * Feu_PreScale_EvtData  1
Feu * Feu_InterPacket_Delay 0

#######################################
#-- Communication registers
#######################################
#-- UDP Channel configuration
#-- CBus address: 0x600000
#######################################
# UdpChan_MultiPackThr = Eth_MTU - MaxSmpData - 60
# MaxSmpData = 2220 bytes (ZS packet with all channels above threshold)
# Eth_Mtu depends on Eth NIC but is less 8kbytes
# e.g. for MTU 7152 the parameter is 4872
```

```
#######################################
Feu * UdpChan_Enable 1
Feu * UdpChan_MultiPackEnb 1
Feu * UdpChan_MultiPackThr 4872
#######################################
#-- Optical Communication Channel configuration
#-- CBus address: x300000
#######################################
Feu * ComChan_Enable 0

#######################################
#-- Auxiliary Trigger Interface configuration register
#-- CBus address: 0x900000
#######################################
#--   0  |  1  |   2  |  3
#-- DcBal|DcBal| TCM  |BERT
#--  ENC | DEC |Ignore|
#######################################
Feu * TI_DcBal_Enc 0
Feu * TI_DcBal_Dec 0
Feu * TI_Ignore    1
Feu * TI_Bert      0

#######################################
#-- Self Trigger parameters
#-- Configuration
#--  0-7 |  8-10 |   11    |12-17 |18-23 | 24
#-- Dream| Multi | CmbHit  |DrmHit|CmbHit|Trig
#-- Masks|plicity|Propagate|Width |Width |Topo
#######################################
Feu * SelfTrig_DreamMask  0xFF
Feu * SelfTrig_Mult         7
Feu * SelfTrig_CmbHitPropFb 0
Feu * SelfTrig_CmbHitPropOl 0
Feu * SelfTrig_DrmHitWid   63
Feu * SelfTrig_CmbHitWid   63
Feu * SelfTrig_TrigTopo     0
#######################################
#-- Self Trigger Veto parameter
#-- in Trig_Clk counts 8ns for Recovered
#--                   10ns for TCM
#######################################
#-- 0-23 |
#-- Veto |
#-- Cntr |
#######################################
Feu * SelfTrig_Veto      0
#######################################
# -- Self Trigger topology parameters
# -- MemAdrBase MemVal0 MemVal1 MemVal2 MemVal3
# --      MemAdr in range [0;28;4]
# --      MemVal 32-bit Hex value with leading 0x
Feu * SelfTrig_Topology    0 0x00000000 0x01000000 0x00200000 0x00030000
Feu * SelfTrig_Topology    4 0x00004000 0x00000500 0x00000060 0x00000007
Feu * SelfTrig_Topology    8 0x80000000 0x09000000 0x00A00000 0x000B0000
Feu * SelfTrig_Topology   12 0x0000C000 0x00000D00 0x000000E0 0x0000000F
Feu * SelfTrig_Topology   16 0x10000000 0x11000000 0x10200000 0x10030000
Feu * SelfTrig_Topology   20 0x10004000 0x10000500 0x10000060 0x10000007
Feu * SelfTrig_Topology   24 0x90000000 0x19000000 0x10A00000 0x100B0000
Feu * SelfTrig_Topology   28 0x1000C000 0x10000D00 0x100000E0 0x1000000F

#######################################
#-- Dream registers
#-- See Dream User Manual
#-- Programmed through FEU slow control register 0x200004
#######################################
#-- reg HexVal3 HexVal2 HexVal1 HexVal0
#-- the first HexVal carries MSB of the register
#-- All four Hex values must be present even if register has < 64 bits
#-- For 64-bit registers all 4 Hex values are significant
#-- For 32-bit registers first two values are significant
#-- For 16-bit register only first value is significant
#-- Non significant Hex values must be set to 0x0000
#######################################
Feu * Dream *  1 0x001F 0xC123 0x0000 0x0000
Feu * Dream *  2 0x0000 0x0000 0x0000 0x0000
Feu * Dream *  3 0x2000 0x0000 0x0000 0x0000
Feu * Dream *  4 0x0000 0x0000 0x0000 0x0000
Feu * Dream *  6 0xAAAA 0xAAAA 0xAAAA 0xAAAA
Feu * Dream *  7 0xAAAA 0xAAAA 0xAAAA 0xAAAA
Feu * Dream *  8 0xFFFF 0xFFFF 0x0000 0x0000
Feu * Dream *  9 0xFFFF 0xFFFF 0x0000 0x0000
Feu * Dream * 10 0xFFFF 0xFFFF 0x0000 0x0000
Feu * Dream * 11 0xFFFF 0xFFFF 0x0000 0x0000
Feu * Dream * 12 0x002B 0x0000 0x0000 0x0000

#######################################
#-- ADC registers
#-- See AD9222 device manual
#-- Programmed through FEU slow control register 0x200004
#######################################
#-- reg val flag
#-- flag: 0 - set only external copy of the register
#-- flag: 1 - set internal register by coping its external value
#--       and tell ADC to take all
#--       set registers into account
#######################################
#Feu * Adc 0x04 0x00 0
#Feu * Adc 0x05 0x01 0
#Feu * Adc 0x0d 0xC4 1
#Feu * Adc 0x19 0xDE 0
#Feu * Adc 0x1A 0xAD 0
#Feu * Adc 0x1B 0xFA 0
#Feu * Adc 0x1C 0xCE 1

#######################################
#-- EeProm values
#-- See M24C08 device manual
#-- Programmed through FEU slow control register 0x200004
#######################################
#-- adr val
#######################################
#Feu * EeProm 0x00 0xE7

#######################################
#-- Dream Clock Parameters
#-- Div: Float value TrigClock / RdClk = 3.0, 4.0, 5.0, 6.0
#-- Phase: from 0 to 360 in steps of 45
#######################################
Feu * DrmClk RdClk_Div      6.0
Feu * DrmClk WrClk_Div      6.0
Feu * DrmClk WrClk_Phase    0
Feu * DrmClk AdcClk_Phase   125
```

## Appendix 2. A typical configuration sequence

1) Configure Dream sampling & readout clocks in the DreamClock module (see 3.9)
    a. write 0xFFFF to PwrBit register 0xD000A0
    b. set Dream read clock frequency
        i. Calculate desired ratio RdClk_Div = TrgClk_Freq / RdClk_Freq: must be a real value in the range [3;6] in 0.5 steps
        ii. Calculate 6-bit value of desired half period: 4 * RdClk_Div
        iii. Read RdClk_Freq register 0xD00020
        iv. Write the half period in bits 5-0 and 11-6 in the RdClk_Freq register w/o modifying original values of bits 15-12
    c. set ADC clock in the AdcClk_Freq register 0xD00030 to the same value following exact the same procedure described in point b
    d. set ADC clock phase relative to the Dream read clock
        i. Calculate 6-bit value of desired phase: (AdcClk_Phase * 64) / 360
        ii. Read AdcClk_Phase register 0xD00034
        iii. Update bits 5-0 in the read value with calculated value
        iv. Force bits 9-8 to 0
        v. Write the value in the AdcClk_Phase register w/o modifying original values of bits 15-10 and 7-6
    e. set Dream sampling clock frequency in the WrClk_Freq register 0xD00028 to the desired value following exact the same procedure described in point b;
        i. Calculate desired ratio WrClk_Div = TrgClk_Freq / WrClk_Freq: must be a real value in the range [3;6] in 0.5 steps
        ii. Calculate 6-bit value of desired half period: 4 * RdClk_Div
        iii. Read WrClk_Freq register 0xD00028
        iv. Write the half period in bits 5-0 and 11-6 in the WrClk_Freq register w/o modifying original values of bits 15-12
    f. set Dream sampling clock phase in the WrClk_Phase register 0xD0002C to the desired value following exact the same procedure described in point d
2) Reset FEU
    a. toggle (0-1-0) bit "0" of the main module command register 0x100000 (see 3.1.1)
    b. make sure that the run control state filed of the main module status register 0x10000C takes "`Init`" value (see 3.1.4)
3) Set the main module configuration register 0x100004 (see 3.1.2)
    a. number of samples
    b. dream mask
    c. Sparse readout factor
    d. trigger clock source

4) Set the main module trigger logic configuration register 0x100004 (see 3.1.3)
    a. time stamp reset value
    b. overflow warning low and high water marks
    c. overflow threshold
    d. local throttling enable
5) Set FEU core module power register 0x200000 (see 3.2.1)
    a. dream pairs: activate Dream pairs one by one avoiding large rush currents with 50 ms delay
    b. protection grounding or floating for protected FEUs
6) Set the run control register 0x200008 of the FEU core module (see 3.2.3)
    a. Pedestal equalization
    b. Common mode noise subtraction
    c. Zero suppression
    d. Dream raw overhead enable in data
    e. Samples to be compared with thresholds for zero suppression
    f. Normal or doubled sampling clock
    g. FEU logical ID
    h. Dream read signal to digitized data ready delay: must be set to 8
    i. Set event ID and timestamp width to default 12-bit width or to extended width
    j. Use immediate or delayed Dream Read signal
    k. Pedestal equalization value after common mode noise subtraction
7) Configure pedestal memory in address range 0x404000-0x407FFC (see 3.4)
    a. Pedestals and/or masks and patterns
8) Configure threshold memory in address range 0x500800-0x500FFC (see 3.5)
9) Set the pulser register 0x200014 of the FEU core module (see 3.2.6)
    a. Dream to be tested
    b. Pulse width
    c. Enable: set to 0 for normal operation
10) Set the pre-scale register 0x200018 of the FEU core module (see 3.2.7)
    a. Event pre-scale: set to 1 for normal operation
    b. Inter packet delay
11) Set the CSR 0x300000 of the optical communication channel module (see 3.3.1)
    a. Enable
12) Set the CSR 0x600000 of the UDP communication channel module (see 3.6.1)
    a. Enable
    b. Multi-packet threshold

   c. Multi-packet enable
   d. Interrupt enable

13) Set the CSR 0x900000 of the auxiliary trigger interface module (see 3.7.1)
   a. DC balanced encoding of transmitted data
   b. DC balanced decoding of received data
   c. Enable or ignore commands received from TCM
   d. Make sure the TCM link is not in the BERT mode
14) Set the CSR of the self-trigger module 0xA00000 (see 3.8.1)
   a. Dream mask
   b. Multiplicity
   c. Enable or disable combined hit propagation through one of the two bussed signals
   d. Enable or disable combined hit propagation over the optical link
   e. Width of valid Dream hit signal
   f. Width of valid coincidence signal
   g. Simple coincidence or topological trigger
   h. Self-trigger veto delay
   i. Program 32 topological memory locations 0xA00080 through 0xA000EC with the required trigger patterns (see 3.8.2)
15) Set the CSR of the trigger generator module 0xE00000 (see 3.10.1)
   a. Trigger veto length
   b. Trigger pipeline length
   c. Trigger source
   d. Trigger rate for locally generated constant or random triggers
   e. Program trigger pattern memory 0xE01000-0xE01FFC if necessary (see 3.10.3)
16) Program the registers of active Dreams [Drm] through the slow control register 0x200004 of the FEU core module (see 3.2.2)
   a. Gain, peaking time, pipeline depth, etc.
17) Configure FEU
   a. Set bit "1" of the main module command register 0x100000 (see 3.1.1)
   b. make sure that the run control state filed of the main module status register 0x10000C takes "`Idle`" value (see 3.1.4)

## ABBREVIATIONS

| | |
|---|---|
| AMT | Asacusa Micromegas tracker |
| BEU | Backend unit |
| CPLD | Complex Programmable Logic Device |
| CSR | Configuration and Status Register |
| FEU | Frontend unit |
| FPGA | Field-Programmable Gate Array |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| GBIC | Gigabit interface converter |
| JTAG | Joint Test Action Group |
| LVDS | Low voltage differential signaling |
| MTU | Maximum Transmission Unit |
| MVT | Micromegas vertex tracker |
| PROM | Programmable Read-Only Memory |
| SCA | Switched capacitor array |
| SFP | Small form-factor pluggable [transceiver] |
| TCM | Trigger and Clock Module |
| TI | Trigger interface |
| VLDO | Very low dropout [regulators] |

## REFERENCES

[Ad9] Analog Devices, "AD9222: Octal, 12-Bit, 40/50/65 MSPS Serial LVDS 1.8 V A/D Converter", Datasheet, Rev. D

[Asa] http://asacusa.web.cern.ch/ASACUSA/asacusaweb/main/main.shtml

[Cla] http://www.jlab.org/Hall-B/clas12

[Cpl] Xilinx, "XC9572XL High Performance CPLD", DS057, v2.0, April 3, 2007

[Dig] Digilent, "JTAG HS1, Programming Cable for Xilinx FPGAs", June 10, 2011

[Drm] P. Baron; E. Delagnes; C. Flouzat; F. Guilloux, "DREAM, a Front End ASIC for CLAS12 detector", User Manual, Production Version, Document 1.0, Sept. 25, 2012

[Fem] D. Calvet, "Feminos Card User's Manual"; Version 2.3, March 27, 2014

[Lwi] A. Sarangi, S. MacMahon, "LightWeight IP (lwIP) Application Examples", XAPP1026 (v3.2), Xilinx, October 28, 2012

[M24] STMicroelectronics, "M24C08, 8 Kbit I²C bus EEPROM", Datasheet, Doc ID 5067, Rev 13, May 2009

[Max] MAXIM, "EEPROM-Based System Monitors with Nonvolatile Fault Memory", datasheet, Rev 4, Aug 2011

[Mec] Samtec, MEC8-RA 0.80 mm Mini Edge Card Connector, Right Angle, http://www.samtec.com/technical-specifications/Default.aspx?SeriesMaster=mec8-ra

[Min] A. Obertelli and T. Uesaka, "Hydrogen targets for exotic-nuclei studies developed over the past 10 years", The European Physical Journal A, vol. 47, issue 9, September 2011, p. 1 – 22

[Mmc] Karl Kurbjun and Carl Ribbing, Xilinx, "MMCM Dynamic Reconfiguration", XAPP878 (v1.1) June 9, 2010

[Mvt] S. Aune et al., "Micromegas Vertex Tracker Feasibility", Saclay, May 2009

[Opt] Opticis, "M2-100 / 210 / 10S / 21S Optical USB Extension Cable", Datasheet, Ver. 4.0, Mar. 31, 2010

[Pbb] Schroff, Power distribution bus bars, reference: 20100-008

[Pla] Xilinx, "Platform Flash In-System Programmable Configuration PROMs", DS123, v2.18, May 19, 2010

[Prg] Xilinx, "Platform Cable USB II", DS593, v1.2.1, March 17, 2011

[Pcn] http://www.te.com/catalog/pn/en/643488-1

[Raa] BECKAIR, "Ringjet Air Amplifiers", Datasheet, www.beck-air.com

[Sfp] Small Form-factor Pluggable (SFP) Transceiver MultiSource Agreement (MSA), September 14, 2000

[T2K] D. Calvet, "T2K TPC Read-out Electronics Digital Front-End Mezzanine Card", Design notes, February 18, 2009

[Tcm] D. Calvet, "TCM User's Manual", Version 1.0, June 6, 2013

[Vir] Xilinx, "Virtex-6 Family Overview", DS150, v2.4, January 19, 2012

[Win] W-Ie-Ne-R, "PL512/PL506, Modular Power Supply System", Technical Manual, June 6, 2011