AIDA mini-TLU manual

F. Crescioli[1], D. Cussans[2], A. Dosil Suárez[3]

[1]*Laboratoire de physique nucléaire et des hautes énergies, Paris, France*

[2]*University of Bristol, Bristol, United Kingdom*

[3]*University of Santiago de Compostela, Spain*

15th April 2013

The AIDA mini Trigger Logic Unit (AIDA mini-TLU) has been conceived as a device to replace tens of VME modules and lemo cables dangling around in every test-beam setup. It permits an easier and remote configuration of the trigger logic through the Ethernet with added functionalities such as test readout elements without beam. The generic design of the mini-TLU permits to connect it to a very wide range of devices, even LHC like readout systems. This has the benefit of sharing hardware, firmware and DAQ-Software effort. Also, it implements a 1 ns accuracy TDC.

1 Introduction

The AIDA mini-TLU provides timing and synchronization signals to test-beam readout hardware. It can either provide or accept a system clock. It accepts the asynchronous trigger signals from an external source, such as beam-scintillators and generate synchronous signals to send to the devices, triggers or other control signals. It also accepts busy signals or other veto signals from DUTs. Also, the mini-TLU records a time-stamp of incoming signals.
The configuration parameters and data are sent and received via the IPbus. IPbus is a simple way to control and communicate TCA-based hardware via the IPbus protocol.

Clocking	Internal 200 MHz oscillator Socket for a single-ended user oscillator SMA connectors
Communication	10/100/1000 Tri-Speed Ethernet PHY
Expansion connectors	FMC-LPC connector (68 se. or 34 diff. user defined signals) 8 User I/O (Digilent 2x6 Header)
Display	4X LEDs
Control	4X Push Buttons 4X DIP Switches

Table 1: Key features of the Xilinx Spartan 601 evaluation kit

2 Hardware design

The AIDA mini-TLU consist of a Xilinx Spartan 601 evaluation kit, attached to a connection card by its FMC-LPC connector. The evaluation kit is a cheap solution for the mini-TLU. It provides all the features needed with at cheaper cost than a built-in design. Some of the key features of the sp601 evaluation kit are showed in Table 1.
The second part of the AIDA mini-TLU is the connection card. This card has been designed specifically for this purpose. The mini-TLU have the next connections plugs, which are also indicated schematically in Figure 1:

- 4 Trigger input: Lemo single-pole size-00
- Clock I/O: Lemo two-pole size-00
- 1 DUT: RJ45(4 differential signals)
- 2 DUT: HDMI (5 differential signals)
- 1 Ethernet connection (copper or optical)

The device under test signals are connected to the TLU via one RJ45 and two HDMI connectors. The two connections have the same signals, but only the RJ45 is backward compatible with the JRA1 Trigger Logic Unit.

2.1 RJ45 connector

The AIDA mini-TLU has one RJ45 connector, compatible with the JRA1 TLU. This connection provides 4 LVDS signals. A detailed description of the signals is indicated in Table 2. Pin out at the mini-TLU:

1. DUT_CLOCK-
2. DUT_CLOCK+
3. BUSY-

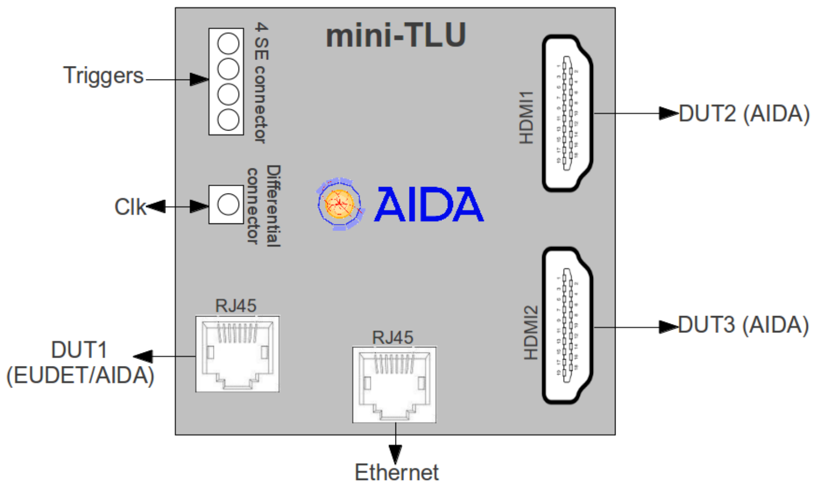


Figure 1: AIDA mini-TLU connections.

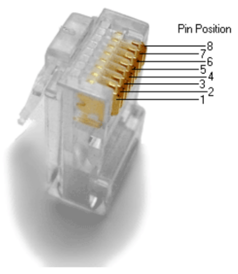


Figure 2: 8P8C (RJ45) connector pin-out

4. CONTROL-
5. CONTROL+
6. BUSY+
7. TRIGGER-
8. TRIGGER+

Connector pins numbering is illustrated in Figure 2.

2.2 HDMI connectors

The AIDA mini-TLU has two HDMI connectors, providing 1 LVTTL and 5 LVDS signals each, as indicated in 3. The pin out is the following:

1. CLOCK-
2. GND
3. CLOCK+

Name	Description	Signal type
Clock	Input/Output: Clock from/to DUT	LVDS
Busy	Input: Busy signal from DUT	LVDS
Control	Output: Shutter, spill-on, power-cycling	LVDS
Trigger	Output: Trigger signal to DUT	LVDS

Table 2: RJ45 signal details

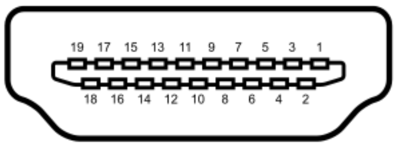


Figure 3: Type A receptacle HDMI pin-out

4. CONTROL+
5. GND
6. CONTROL-
7. BUSY+
8. GND
9. BUSY-
10. SPARE+
11. GND
12. SPARE-
13. UNCONNECTED
14. HDMI_POWER_ENABLE
15. TRIGER+
16. TRIGER-
17. GND
18. UNCONNECTED
19. UNCONNECTED

Connector pins numbering is illustrated in Figure 3.
Each of this cables can control one device and, since the mini-TLU only provides two HDMI and one RJ45 connectors, we will need a method to spread the signals. We will use an HDMI fannout. This gadget is a 1:8 fanout, providing three differential pairs having 1:8 fanout and two differential pair having a 8:1 fan-in. The fanout schematic is indicated in Figure 4.

Name	Description	Signal type
Clock	Output: Clock to DUT	LVDS
Busy	Input: Busy signal from device	LVDS
Control	Output: Shutter, spill-on, power-cycling	LVDS
Trigger	Output: Trigger signal to DUT	LVDS
Spare	Input: General purpose Input	LVDS
HDMI_POWER_ENABLE	Output: HDMI power enable	LVTTL

Table 3: HDMI signals details

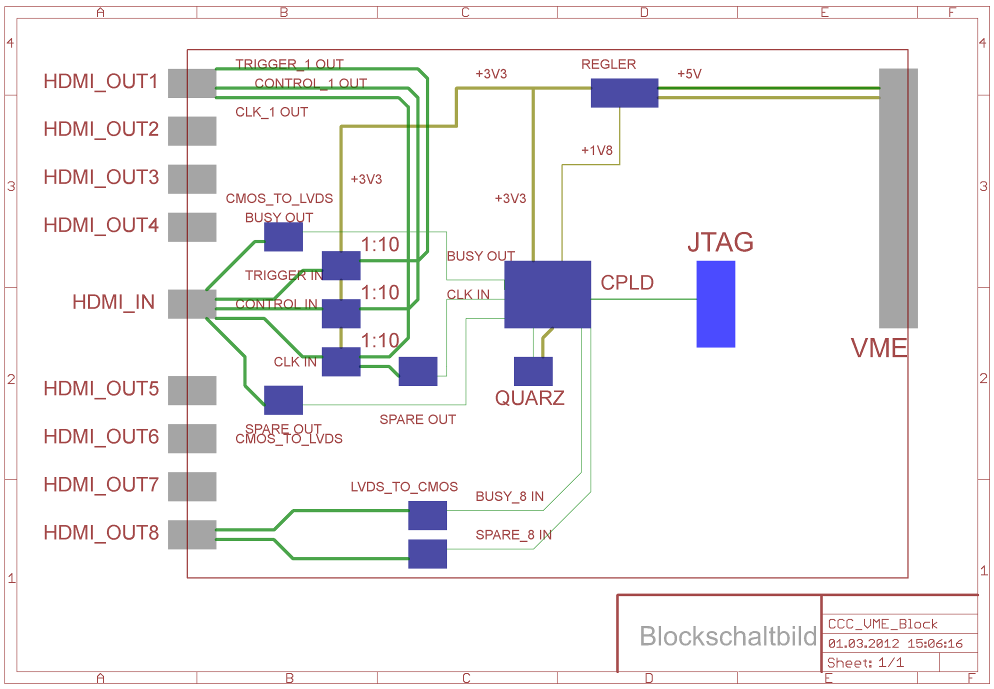


Figure 4: Fanout to spread the signals from the mini-TLU to many devices.

3 Firmware

The AIDA mini-TLU firmware is composed basically by the next main elements: the Ethernet communication driver (IPbus), the time to digital converter (TDC), the handshake controller, the controler for discriminator threshold DAC and the controller for programmable shutter/spill-on/power-cycling signal.

3.1 IPbus

IPbus is a simple way to control and communicate with your Ethernet-attached xTCA hardware via the IPbus protocol. IPbus is the communication protocol used for the data interchange between the mini-TLU and the DAQ pc. Its source code is available to download in [?], it is written in VHDL and it has a small footprint. Data is sent directly

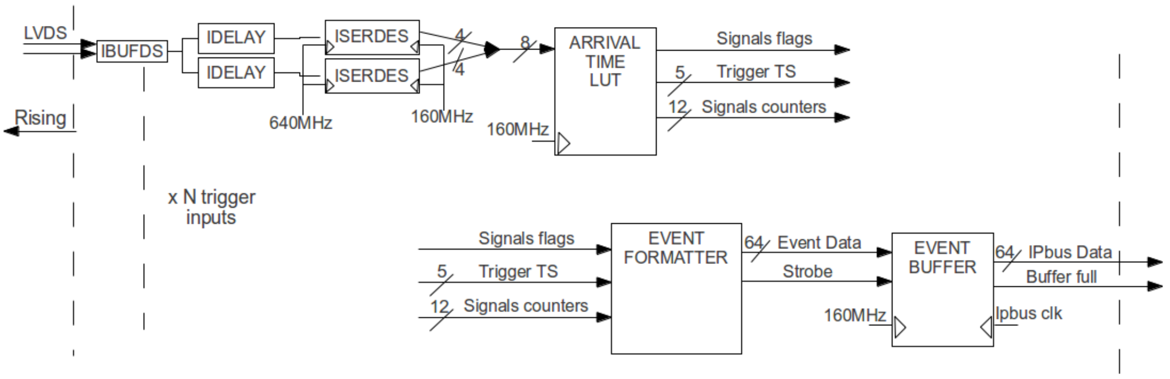


Figure 5: TDC global schematic view

to the FPGA and you do not need a microblaze or other CPU. It is not needed any proprietary hardware, firmware or software to operate it as well. The transport protocol is UDP and the data is addressable in 32b packets.

3.2 Time to digital converter (TDC)

The TDC is the component in charge of time stamping every interesting signal with an accuracy of 0.725 ns. To accomplish this precision the TDC uses a 640 MHz input clock as reference and splits the input, delaying one of the split signals one half cycle. Each branch goes then to the deserializer module IOSERDES (Input/Output SERializar/DESerializer). This module generates a 4 bits parallel word with the input data deserialized, then this data is merged and we obtain an 8 bits parallel word every 1/(160 MHz) ns. The data obtained is analysed and the system generates a rising or falling signal, depending on the input. The rising and falling edges are sent then to the event formatter, which generates the final information word to be sent out. Finally, the data arrives to the event buffer, where it is saved in a FIFO while it waits to be sent to the DAQ pc. The schematic of this process can be seen in Figure 5.
The IODELAY module needs to be recalibrated every once in a while. For the moment the calibration is started externally from IPbus. The finite state machine of the calibration process is indicated in Figure 6.

3.3 Data Format

The AIDA mini-TLU records the time stamp of input and internally generated signals with an accuracy better than 1 ns. The time stamp format is different depending on the signal recorded and the number of input signals configured in the measuring moment. There are three possible signal types: trigger, internally generated and edge signals. The internally generated and edge signals have the properties of rising and falling edges. The

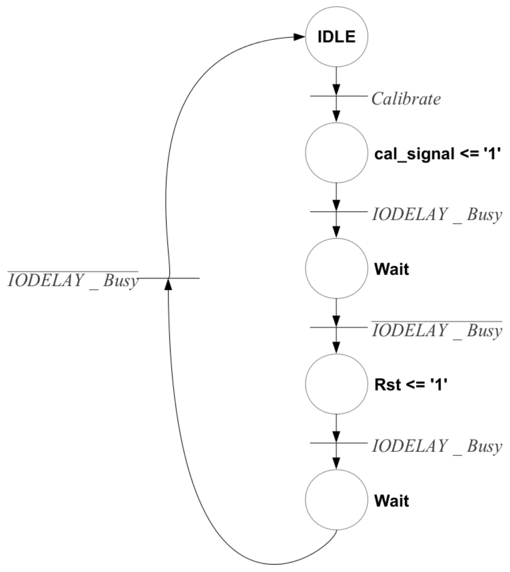


Figure 6: Finite state machine for the IODELAY calibration

0000	internal trigger
0001	external trigger
0010	shutter falling
0011	shutter rising
0100	edge falling
0101	edge rising
0110	spill off
0111	spill on

Table 4: Event type codification

coding of these names is indicated in Table 4. Data is sent to the DAQ pc in packages from one to three words of 64 bits.

3.3.1 Trigger data

The AIDA TLU will have 12 dedicated trigger inputs. The mapping for this type of signal is indicated in Figure 7. The information of this signal is sent in two or three 64 bits word depending on whether any of the 4 to 11 dedicated trigger inputs are enabled. The AIDA mini-TLU only has four trigger inputs, so it will always send only two 64 bits words.
The 48 lowest significant bits of the first word indicate the 25 ns time stamp. This time stamp is synchronized with the 40 MHz main system clock. Bits from 48 to 59 indicate which of the trigger inputs have been fired and the 4 most significant bits indicate the event type. In this particular case the possibilities are only internal or external trigger. The 32 lowest significant bits of the second word indicate the event number. The 32

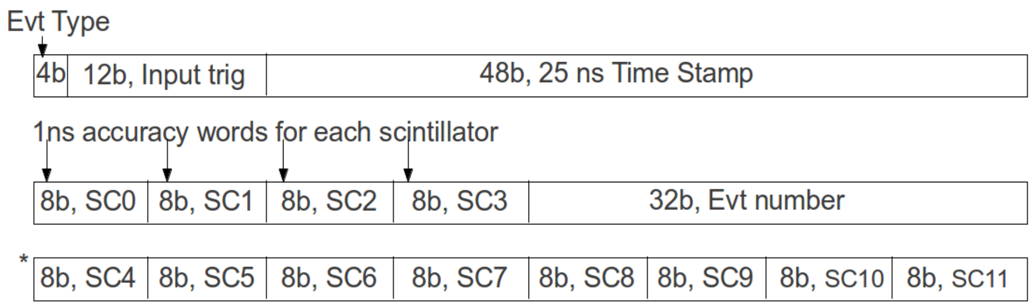


Figure 7: Mapping of the trigger signal information. The third word, marked with a star, is only sent when at least one of the 4 to 11 dedicated trigger inputs is enabled.

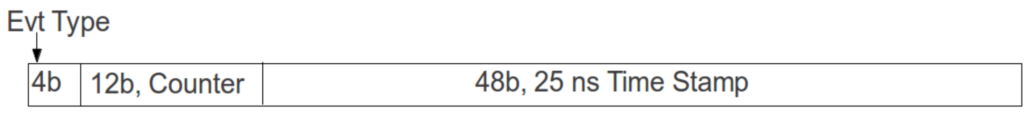


Figure 8: Mapping of the internally generated signals information.

highest significant bits are divided in four words of 8 bits. Each word indicates the 1 ns accuracy time stamp of the signal received in each dedicated input. Only in the case that at least one of the 4 to 11 dedicated trigger inputs is enabled, the third word is created and sent out. This third word contains eight words of eight bits, each indicating the 1 ns time stamp of the trigger in each input.

3.3.2 Internally generated data

The internally generated signal information includes the event type, an event number counter and a 25 ns time stamp, as can be seen in Figure 8. In this case only one 64 bits word. These signals are synchronous with the 40 MHz system clock, therefore only a 25 ns time stamp is needed.
The 48 lowest significant bits indicate the 25 ns time stamp. The next 12 bits are the event number and the four most significant bits indicate the event type.

3.3.3 Edge data

There are some external signals that can be interesting to record. We call them edge data because we only want to know their rising or falling edge time stamps. The data format can be seen in Figure 9. As in the shutter and spill case, only one 64 bits word is sent. The 48 lowest significant bits indicate the 25 ns time stamp and the 4 highest significant bits indicate the event type, as in every other signals. Bits from 49 to 55 indicate the 1 ns time stamp and the next 4 bits code which input detected the event.

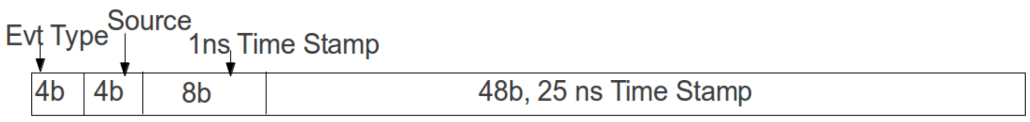


Figure 9: Mapping of the edge signal information.

3.4 Handshake between TLU and DUT

The AIDA TLU provides two different handshake modes. One EUDET handshake, compatible with the old JRA1 TLU, and a new synchronous mode.

3.4.1 EUDET handshake (Trigger/busy with trigger number)

1. TLU receives trigger from beam scintillators
2. TLU asserts TRIGGER
3. On receipt of TRIGGER going high, the detector asserts BUSY
4. On receipt of BUSY going high, TLU de-asserts TRIGGER and switches the TRIGGER line to the output of a shift register holding the trigger number/data.
5. The DUT clocks data out of the shift register by toggling TRIGGER CLOCK. Data changes on the rising edge of TRIGGER CLOCK3 . The least significant bit of the trigger data is shifted out first. Only the bottom 15-bits of the 32-bit trigger counter are clocked out. If more than 15 clock pulses are issued on the TRIGGER CLOCK line the TRIGGER output is set to zero. The DUT should issue 16 clock pulses which will clock out the bottom 15-bits of the trigger number and return the TRIGGER line to logical low. This will avoid glitches on the TRIGGER line when the DUT returns the BUSY line to logical low.
6. After clocking out the trigger number (and the detector being ready to take more data, the DUT de-asserts BUSY)
7. System is ready for triggers again.

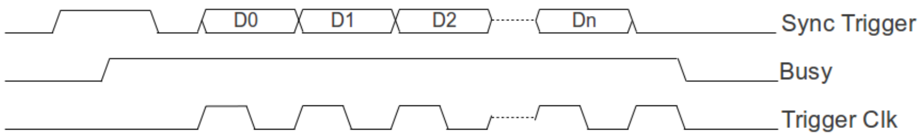


Figure 10: Timing of signals in “EUDET Handshake”

3.4.2 Synchronous mode

This mode is new for the AIDA TLU and it is incompatible with old handshake modes.

1. TLU receives trigger from beam scintillators
2. TLU asserts TRIGGER synchronous with its internal clock
3. After one clock cycle the system is ready for triggers again

- If at any time the TLU receives a busy signal from DUT, it will veto all incoming triggers while the signal is active

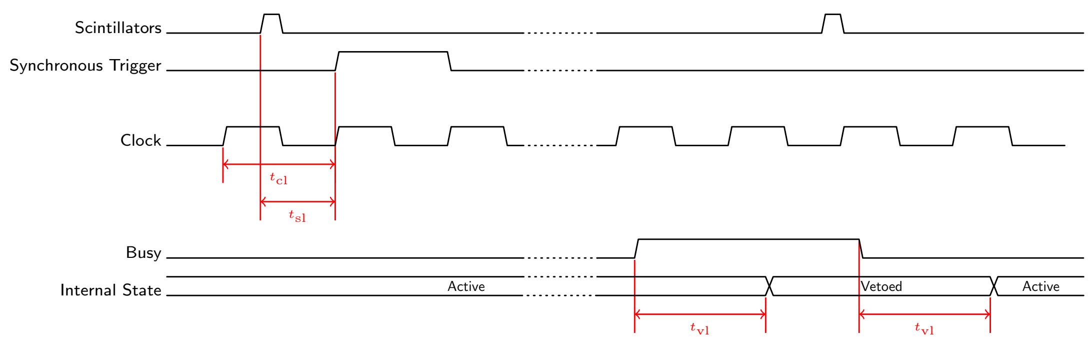


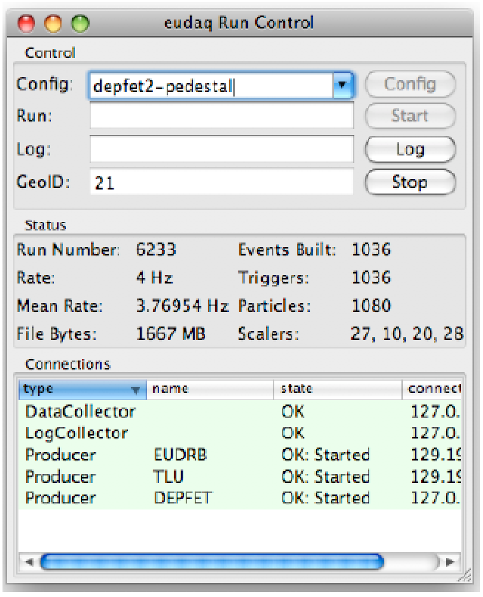
Figure 11: Timing of signals in “Synchronous mode”

4 Software

The TLU control software is based on two main frameworks: EUDAQ[?] and μHAL[?] framework.

4.1 μHAL

The μHAL hardware abstraction framework is a component of the wider CACTUS framework developed for the upgrades of the CMS Level-1 Trigger. It can be compiled and used separate from the rest of the CACTUS framework.
The μHAL library provides a simple and flexible way to describe the hardware registers and other memory elements. The hardware is described in XML files and it reflects also the internal hierarchy of the firmware.
The tree structure described in XML is made accessible with robust C++ classes with methods for I/O operations. On the other side of the interface the communication with the hardware is done using the UDP-based IPBus protocol. This abstraction layer provides an efficient separation between the firmware implementation and the C++ access to hardware memory elements.



Control

Config: depfet2-pedestal
Run:
Log:
GeoID: 21

Status

Run Number:	6233	Events Built:	1036
Rate:	4 Hz	Triggers:	1036
Mean Rate:	3.76954 Hz	Particles:	1080
File Bytes:	1667 MB	Scalers:	27, 10, 20, 28

Connections

type	name	state	connect
DataCollector		OK	127.0.
LogCollector		OK	127.0.
Producer	EUDRB	OK: Started	129.1
Producer	TLU	OK: Started	129.1
Producer	DEPFET	OK: Started	127.0.

Figure 12: EUDAQ Run Controller

4.2 EUDAQ

EUDAQ is a simple and easy to use data aquisition framework, written in C++. It was originally designed to be used for the EUDET JRA1 beam telescope.
It provides a simple and flexible structure to build a DAQ system.
The framework provides several GUI interfaces. For example the run controller is shown in figure 12 and the histogram viewer for online monitoring is shown in figure 13.

The other main components of the framework are networked deamons. The “producers” have the task to communicate with the hardware and produce the data stream. The “data collector” has the task to receive all data streams from the producers and collect them in a structured binary file for offline analysis.
The deamons communicate to the run controller and each other via TCP/IP. The run controller provides commands to coordinate the state transitions of all connected daemons (ie. to configure or to start a data taking run).

4.2.1 TLU Producer

The EUDAQ component for the mini-TLU is the TLU_IPbusProducer.
This producer can access all TLU registers via the abstraction provided by μHAL.

The hierarchy of the various modules within the firmware is reproduced. The top level is described in table 5. At top level the only accessible register is the firmware version, all the other memory elements are contained in a subsystem.
The available subsystems are described in tables 6 to 13. The IPBus address of each memory element can be computed by adding the “leaf address” to the “base address” of

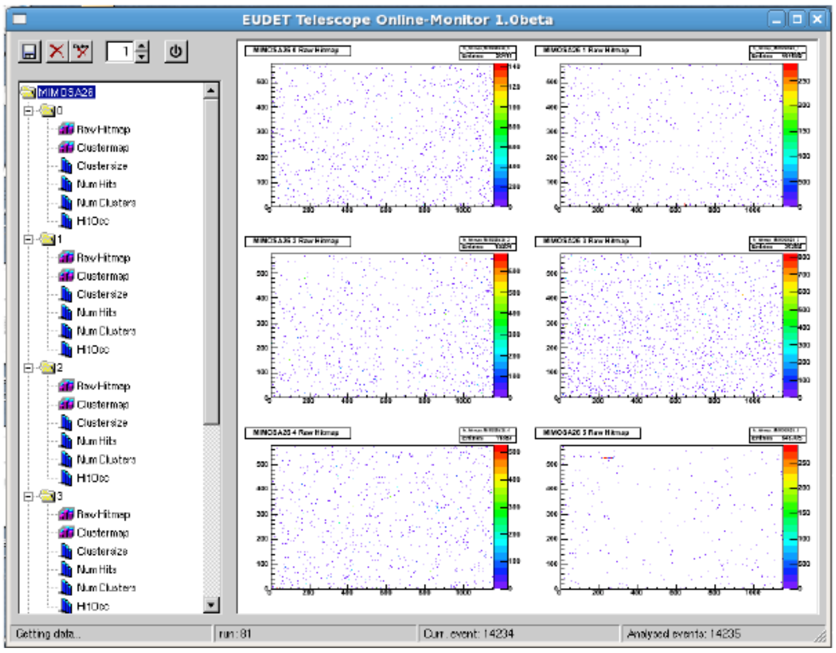


Figure 13: EUDAQ Online Monitor

the subsystem. On the C++ TLU Producer software all memory elements are accessed by the “Node-Id” name.

The TLU Producer software handles the commands from run control and provides high level configuration to the TLU unit. It also reads the Event Buffer FIFO in the TLU and send the corresponding stream of data to the EUDAQ data formatter.
All functions of the TLU Producer are available in a standalone command line program for debugging purposes.

4.3 Configurations

5 Hardware Interfaces

The TLU is normally under the control of the central DAQ software. It can be configured, controlled and the time-stamp information read-out by the central DAQ.

5.1 CALICE

The signals to/from the CALICE Clock and Control Card (CCC) are as follows: Clock(5MHz), Trigger , Busy, Master Clock(50MHz). The Trigger and Busy signals are Manchester(Phase) encoded accordinng to IEEE 802.3 to maintain DC-Balance. Event matching between Calice and beam-telescope verified by matching timestamps between TLU and CALICE readout within a combined event.

Node-Id	Base address	Type
FirmwareVersion	0x00000000	register
EMACHostbus	0x00000002	subsystem
DUTInterfaces	0x00000020	subsystem
TriggerInputs	0x00000040	subsystem
TriggerLogic	0x00000060	subsystem
EventBufferc	0x00000080	subsystem
I2C	0x000000c0	subsystem
TriggerGenerator	0x000000e0	subsystem
ShutterGenerator	0x00000100	subsystem
SpillGenerator	0x00000120	subsystem
EventFormatter	0x00000140	subsystem

Table 5: Root level μHAL nodes

Node-Id	Leaf address	Type
SerdesRst	0x00000000	register

Table 6: Trigger inputs subsystem μHAL nodes

It seems likely that the CALICE CCC system the CALICE Beam Information (BIF) system and the beam-telescope TLU could be implemented in the same physical hardware.

5.2 TimePix

The LHCb TimePix telescope is being offered as part of AIDA infrastructure and hence needs to be able to synchronize with a device under test. Currently this is done using NIM based electronics. If a central clock , a trigger signal and an optional synchronization signal is distributed then event synchronization between the DUT and the TimePix telescope can be done either by timestamp matching.
Event matching by comparing timestamps has the advantage that as long as there is no upset in the timestamp coiunters missing triggers are not an issue.

5.3 LHC

Either use EUDET-Style signal definitions or Calice-style interface (phase encoding is optional). Depending on requirements.

5.4 Existing EUDAQ Users

Keep compatibility with EUDAQ TLU.[?]

Node-Id	Leaf address	Type
PostVetoTriggers	0x00000000	register
PreVetoTriggers	0x00000001	register
InternalTriggerInterval	0x00000002	register
TriggerMask	0x00000003	register
TriggerVeto	0x00000004	register
ExternalTriggerVeto	0x00000005	register
ResetCounters	0x00000006	register

Table 7: Trigger logic subsystem μHAL nodes

Node-Id	Leaf address	Type
EventFifoData	0x00000000	register
EventFifoFillLevel	0x00000001	register
EventFifoCSR	0x00000002	register
EventFifoFillLevelFlags	0x00000003	register

Table 8: Event buffer subsystem μHAL nodes

6 Integrating Different Detectors

6.1 CALICE

Integration between CALICE Calorimeter modules and other detectors, for example the pixel beam telescope is part of the AIDA programme. The purpose is mainly for system integration tests and proof of concept.
There are various integration options. Integration at the hardware level will be done by integration of the AIDA TLU with the CALICE CCC. Possibly by making them the same physical object. For software integration, one possibility[?] is to control CALICE run control from EUDAQ. Write separate files. Combine offline. Purpose: concept.

6.2 TPC

Integration of XXX , YYYY system almost completed under EUDAQ programme[?] , martin-killenberg.

6.3 Triggered Detectors (e.g. ATLAS, CMS Pixels)

As in EUDET. Beam-telescope and DUT synchronized by TLU hardware signals. DUT data written either via an EUDAQ producer, or by writing telescope and DUT data to separate files and comdining offline.

Node-Id	Leaf address	Type
LogicClocksCSR	0x00000000	register
LogicRst	0x00000001	register

Table 9: Logic clocks subsystem μHAL nodes

Node-Id	Leaf address	Type
TriggerLength	0x00000000	register
TrigStartupDeadTime	0x00000001	register
TrigInterpulseDeadTime	0x00000002	register
TriggerDelay	0x00000003	register
NMaxTriggers	0x00000004	register
TrigEvtNumber	0x00000005	register
RstTriggerCounter	0x00000006	register
TrigRearmDeadTime	0x00000007	register

Table 10: Trigger generator subsystem μHAL nodes

6.4 TimePix

Synchronize TimePix and TLU timestamp counter via common system clock. Provide DUT with triggers if needed. Write DUT and TimePix data to separate files and combine offline. One possible refinement[?] is that the DUT can be "EUDAQ" compatible. Then only one data combiner is needed - TimePix with EUDAQ.

7 Specifications

Parameter	Value
Maximum master clock frequency, F_{master}	80MHz
Master clock jitter	To Be Decided.[1]
Timestamp precision	1 ns
Minimum pulse width (time above threshold)	5 ns
Latency	< XXX cycles [2]
Maximum instantaneous trigger rate [3]	F_{master}
Maximum sustained trigger rate	1 MHz

8 High Rate Tests

LHC sensors need 400MHz/cm^2 for pile-up tests. The TLU will not be able to cope with this rate of triggers except by using very small area scintillator. In addition, even if the TLU could cope the MAPS telescope sensors would not. However, it has been pointed out[?] that it would still be possible to conduct efficiency studies by placing the telescope and DUT in a moderate rate beam and then illuminating the DUT with a high flux of

Node-Id	Leaf address	Type
ShutterLength	0x00000000	register
ShutStartupDeadTime	0x00000001	register
ShutInterpulseDeadTime	0x00000002	register
ShutterDelay	0x00000003	register
NMaxShutters	0x00000004	register
ShutEvtNumber	0x00000005	register
RstShutterCounter	0x00000006	register
ShutRearmDeadTime	0x00000007	register

Table 11: Shutter generator subsystem μHAL nodes

Node-Id	Leaf address	Type
SpillLength	0x00000000	register
SpillStartupDeadTime	0x00000001	register
SpillInterpulseDeadTime	0x00000002	register
SpillDelay	0x00000003	register
NMaxSpills	0x00000004	register
SpillEvtNumber	0x00000005	register
RstSpillCounter	0x00000006	register
SpillRearmDeadTime	0x00000007	register

Table 12: Spill generator subsystem μHAL nodes

radiation from either a radioactive source or an X-ray generator. That is to say, use the same approach as the Gamma Irradiation Facility[**?**] at CERN.

Node-Id	Leaf address	Type
EnableRecordData	0x00000000	register

Table 13: Event formatter subsystem μHAL nodes