Tutorial: AIDA-TLU/EUDAQ2

Lennart Huth
DESY

Thanks to P. Schütze and F. Diegritz for their help preparing the tutorial
And Tamar for helping with the shipping

28 Jan 2020

That's what you signed up for

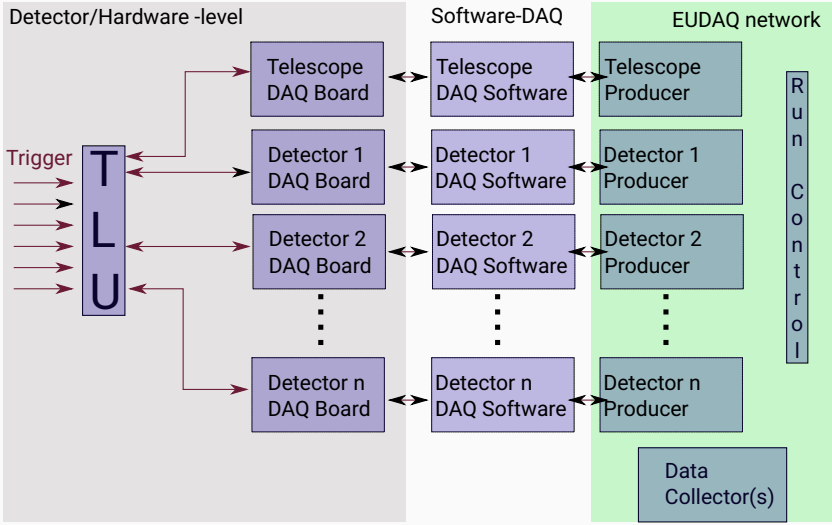
- Listening to my (short) introduction
- Getting and installing all the software pieces
- Starting, initializing and configuring with EUDAQ2
- Using the TLU
- How to trigger with the AIDA-TLU
- Testing the different DUT interfaces
- A short coffee break :)

Introduction

- Successful test beams requires best possible knowledge about the DAQ system that will be used.
- Last years tutorial from which I took quite some stuff
- I think (hope) I've prepared too much material - if we will not finish within the time, that's nothing to worry about. If we finish faster - we can discuss more details.
- The tutorial relies on you: Ask whatever you want to ask!

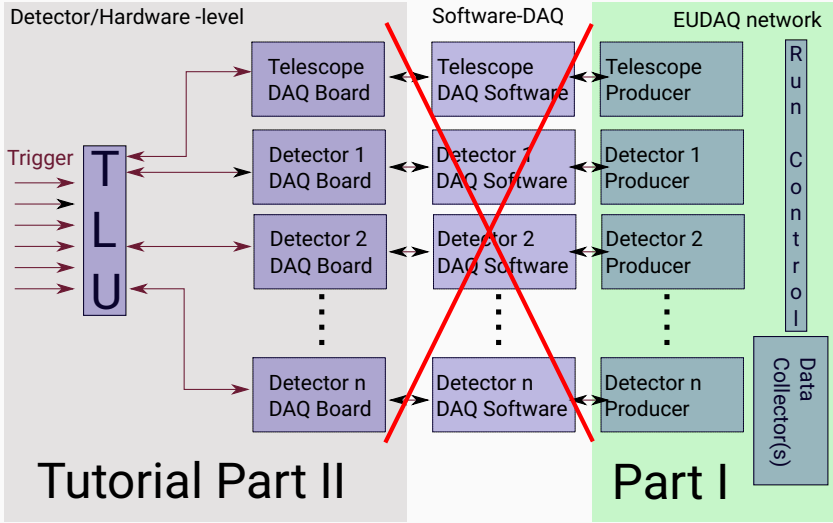
A TYPICAL TEST BEAM DAQ LAYOUT

- Reference telescope
- Triggering logic
- DAQ network
- N-DUTs



A TYPICAL TEST BEAM DAQ LAYOUT

- Reference telescope
- Triggering logic
- DAQ network
- N-DUTs
- We will ignore all device specific DAQ software
- We will assume that the DAQ Boards have an interface to the TLU

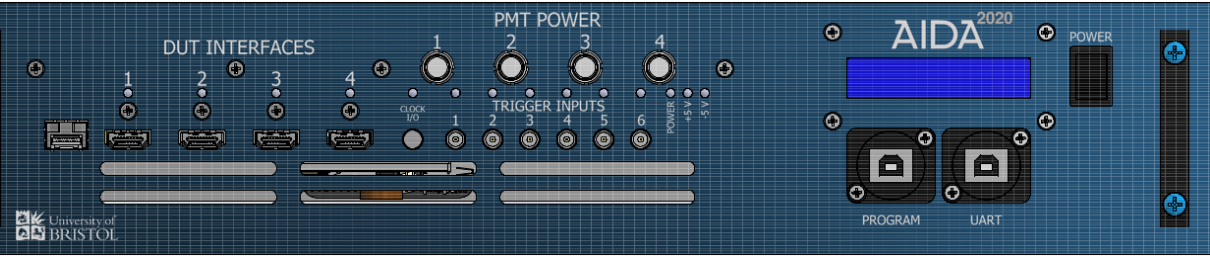


AIDA-TLU - Synchronizing different detectors

- TLU = TriggerLogicUnit
- 6 configurable inputs to create a trigger
- 4 HDMI-differntial DUT interfaces

DUT interface modes

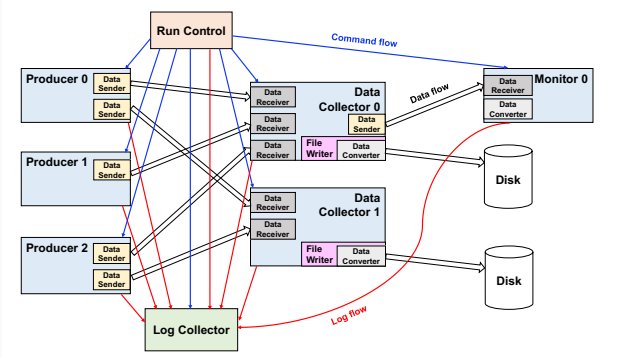
- Trigger, ID (and veto)
- Trigger, clock and T_O
- Trigger (and veto)



EUDAQ2 - The telescope DAQ software

Modular DAQ framework with components running in the same network

- One RunControl
- One (optional) LogCollector
- One Producer per hardware device
- One to many DataCollectors to store data
  - → Event-id sorted
  - → Trigger-id sorted
  - → Unsorted
- Optional Monitor



EUDAQ2 paper

User: MyFancySensorProducer and MyFancySensorRaw2StdEventConverter

Part I: EUDAQ2 in a nutshell

Installing the software

```
# if not yet done clone the repository for the tutorial
git clone https://github.com/lhuth/bttb8_daq_tutorial.git
cd bttb8_daq_tutorial

# driver to talk to the TLU
git clone https://github.com/ipbus/ipbus-software.git
#Follow instructions on https://ipbus.web.cern.ch/ipbus/doc/user/html/software/install/compile.html
# source the download_eudaq script
source download_eudaq.sh
cd eudaq && mkdir build && cd build
cmake -DUSER_TLU_BUILD=on ..
make install -j4
cd ..
source setup.sh
```

If more than two people manage to install it all we are good for the rest of our tutorial

Getting Started

```
cd $PATH_to_EUDAQ/bin
./euRun &
./euCliProducer -n AidaTluProducer -t aida_tlu &
# Any other device can be added later on via
# ./euCliProducer -n <ModuleName> -t <givenName> -r <IP>:<port>
```

Initialize the system

Take a look at the `configs/example.init`, load it and click init

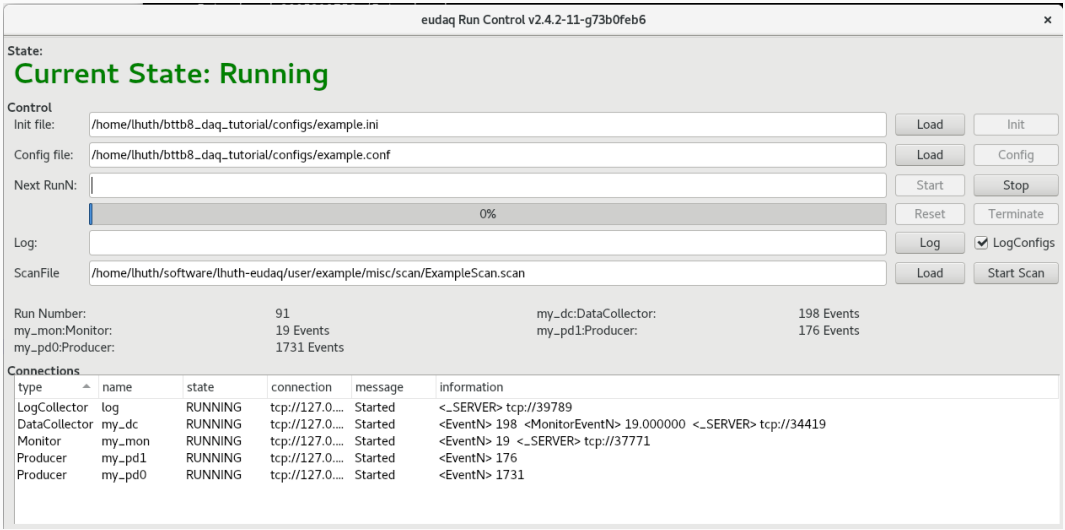
```
[RunControl]
#Nothing to be done here

[LogCollector.log]
EULOG_GUI_LOG_FILE_PATTERN = myexample_$12D.log

[Producer.my_pd0]
EX0_DEV_LOCK_PATH = /tmp/mydev0.lock

[Producer.my_pd1]
EX0_DEV_LOCK_PATH = /tmp/mydev1.lock
```

Configure the system

Take a look at the `configs/example.conf`, load it and click conf

```
[RunControl]
EX0_STOP_RUN_AFTER_N_SECONDS = 60
EUDAQ_CTRL_PRODUCER_LAST_START = my_pd0
EUDAQ_CTRL_PRODUCER_FIRST_STOP = my_pd0

[Producer.my_pd0]
EUDAQ_DC = my_dc # used data collector
EX0_PLANE_ID = 0
EX0_DURATION_BUSY_MS = 10
EX0_ENABLE_TRIGERNUMBER = 1
EX0_DEV_LOCK_PATH = mylock0

[DataCollector.my_dc]
EUDAQ_MN = my_mon #monitor that receives data
EUDAQ_FW = native
EUDAQ_FW_PATTERN = run$3R_$12D$X
EUDAQ_DATACOL_SEND_MONITOR_FRACTION = 10

[Monitor.my_mon]
```

Start/Stop a run



Scanning - Automated data taking

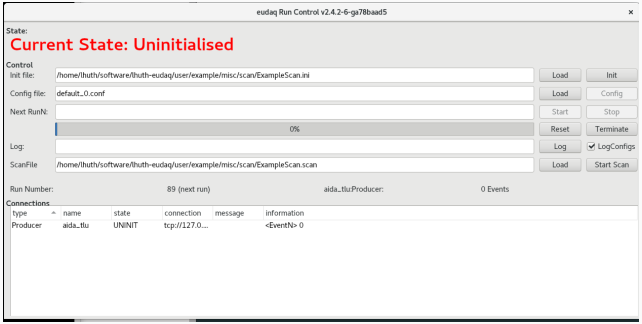
```
[global]
repeatScans = 1
allowNested = 0 # overwrites local nested arguments
#configPrefix = "path/to/folder/scanned" // optional
timeBasedScan = 1 # 1 = true :)
timePerStep = 10 #in second
nEventsPerStep   = 200

[0]
default = 0
start = 1
stop = 2
step = 1
name = Producer.my_pd0
#eventCounter = Producer.my_pd0
parameter = EX0_PLANE_ID

[4]
nested = 1
default = 0
```

Part II: The AIDA Trigger Logic Unit

Getting started

- We need be two groups from now on
- And we need one laptop that connects to the TLU network
- Each team will have one TLU
- TLU-IP: 192.168.200.30
- Laptop-IP: 192.168.200.1
- Try to ping the TLU
- Start EUDAQ+TLU

```
cd $PATH_to_EUDAQ
./startup_tlu.sh
```



Initialize the TLU

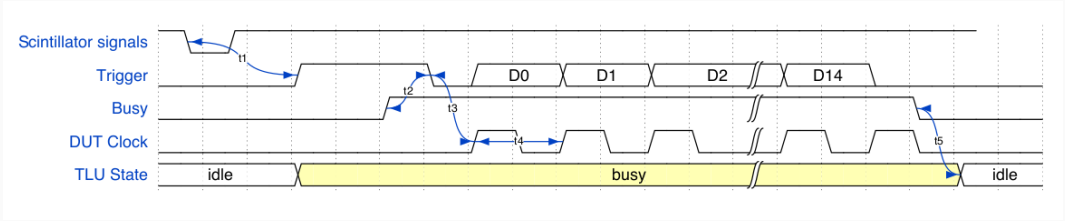
```
[Producer.aida_tlu]
# you can use this to track your changes, e.g. using the date
initid = 20180925
TLUmod= "1e"
# Path on the PC with TLU Producer and relative path is starting path euRun!
ConnectionFile = "file:///home/lhuth/bttb8_daq_tutorial/eudaq/user/eudet/misc/hw_conf/aida_tlu/aida_tlu_connection.xml"
# ControlHub is recommended for Ubuntu, the name is the name in the conncetion file
DeviceName = "aida_tlu.controlhub"
#DeviceName = "aida_tlu.udp"


# Set CONFCLOCK to 1 to configure clock, which is necessary after a power cycle
CONFCLOCK = 1
# Path to clock file
CLOCK_CFG_FILE = "/home/lhuth/bttb8_daq_tutorial/hw_conf/aida_tlu/aida_tlu_clk_config.txt"
skipini = 0 # Set skipini to 1, if you want to skip the init-step
```

Configure the TLU

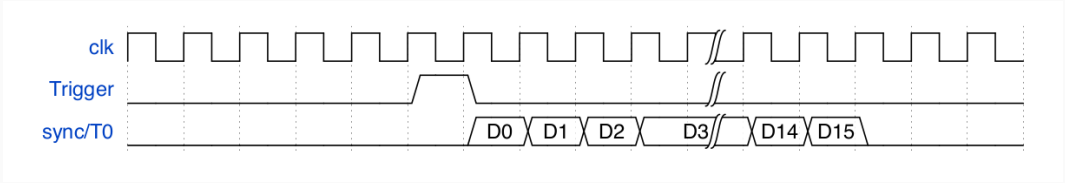
```
1
2  [Producer.aida_tlu]
3  verbose = 0
4  confid = 20181002
5  skipconf = 0
6
7  # delay start in ms
8  delayStart = 0
9
10 ######################################################
11 # DUT IN/OUTPUT
12
13 # Mask: 0 CONT, 1 SPARE, 2 TRIG, 3 BUSY (1 = driven by TLU, 0 = driven by DUT)
14 # EUDET mode: 7
15 HDMI1_set = 0x7
16 HDMI2_set = 0x7
17 HDMI3_set = 0x7
18 HDMI4_set = 0x7
19
20 # same as above for the clock line, 1 = AIDA mode, 2 = FPGA
```

DUT-Inferface modes

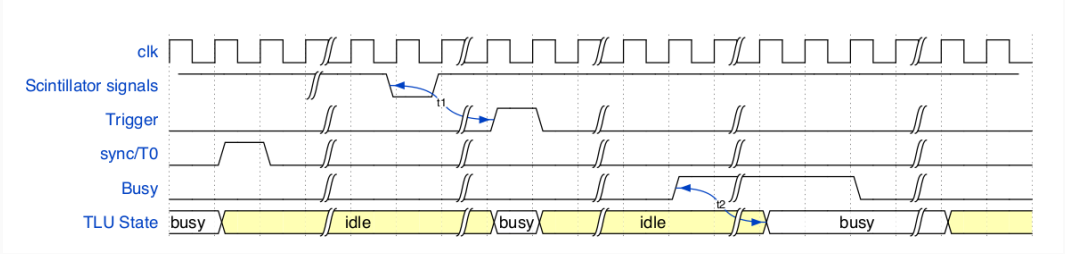
Classical EUDET-mode



Mixed-mode



Full AIDA-mode



Testing the trigger modes I

”Simply” connect the TLU to the signal, configure auto-triggers and study the impact on the oscilloscope. We have two little HDMI to LEMO converters that you can use

Part II-a: Setting trigger thresholds and control voltages

Setting up scintillators

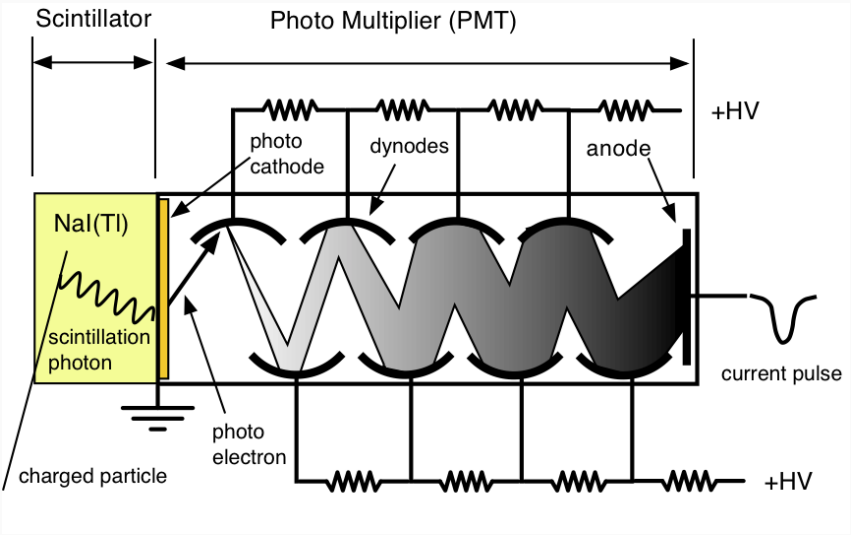
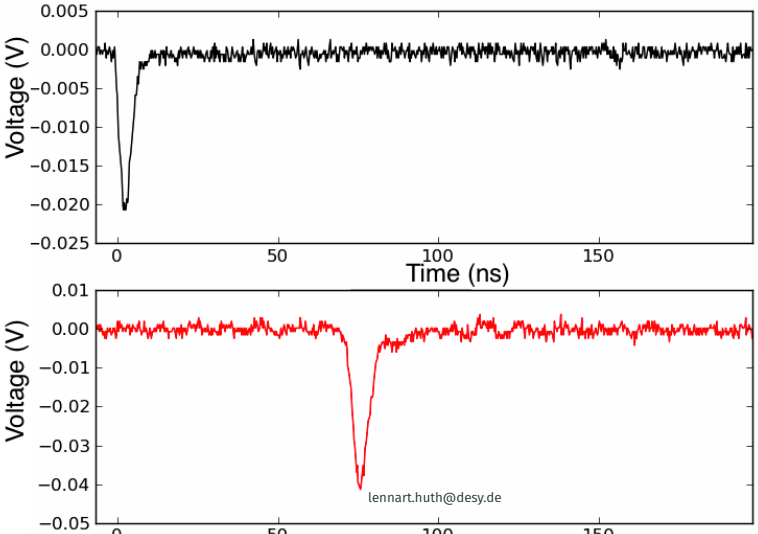


Image credit: http://wanda.fiu.edu/teaching/courses/Modern_lab_manual/_images/PMT.png

Setting up scintillators



lennart.huth@desy.de

Configuring the PMTs

- `PMTX_V` to define the control voltage - ranges from 0-1V and should be 0.8V as a default
- `DACThresholdX` to set the threshold in V, typically 0.04V

```
PMT1_V =0.8
PMT2_V =0.8
PMT3_V =0.8
PMT4_V =0.8

DACThreshold0 = -0.04
DACThreshold1 = -0.04
DACThreshold2 = -0.04
DACThreshold3 = -0.04
DACThreshold4 = -0.20
DACThreshold5 = -0.20
```

Setting valid trigger inputs

The TLU has 6 inputs, resulting in $2^6 = 64$ potential trigger combinations $\rightarrow$ 2 32 bit words for configuration.

Find the configuration to trigger on:

- Only $I_0 + I_1$
- $I_0 + I_3 + I_5$ OR $I_1 + I_4$
- $\bar{I_0} + I_1 + I_3$
- $I_0 + I_3 + I_5$ AND $I_1 + I_4$

Now let's see what the following combinations will trigger on (only LSBs):

- 0x105
- 0xC

Setting valid trigger inputs - solutions

- Only $I_0 + I_1$
- $I_0 + I_3 + I_5$ OR $I_1 + I_4$
- $\bar{I_0} + I_1 + I_3$
- $I_0 + I_3 + I_5$ AND $I_1 + I_4$

- 0x105
- 0xC

Part II-b: Raspberry Pi as DUT

Using the auto trigger

- connect via: `ssh pi@192.168.200.111(113)`
- execute `sudo ./DutDummy`

Check the rates for different modes with different delays on the raspberry pi