EUDAQ2 and AIDA TLU tutorial

Or: Using Upgrades for EUDET-type telescopes and other telescopes

17th January 2019 at BTTB7, CERN

Mengqing Wu, Yi Liu, David Cussans, Jan Dreyling-Eschweiler

Goals of this tutorial

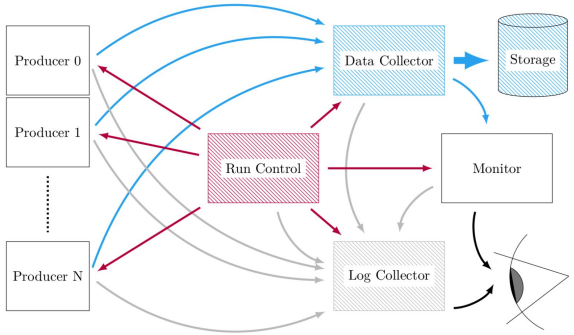
- for you: the integration strategy of EUDET-type telescopes
- for you: Learn the usage of EUDAQ2
- for you: Learn the usage of AIDA TLU
- for you: Learn the new potential at test beam setups
- from you: Feedback for further optimizations
  → also later possible: https://github.com/eudaq/eudaq/issues

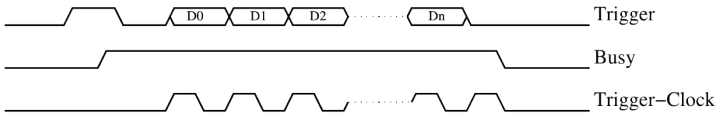
Track #0: EUDAQ2 Installation and Example
Track #1: Moving from 1 to 2
Track #2: Python Interface
Track #3: AIDA TLU

DAQ system: EUDAQ1 and EUDET TLU

Historically developed for EUDET-type telescopes
→ robust and flexible user device integration

- EUDAQ (Software level):
  - synchronisation at event-level
    → one central data collector
- TLU (Hardware level)
  - Trigger-Busy Logic (opt. Trigger ID)
    → trigger/event rate limited by the slowest device
    → limited time information for multiple tracks

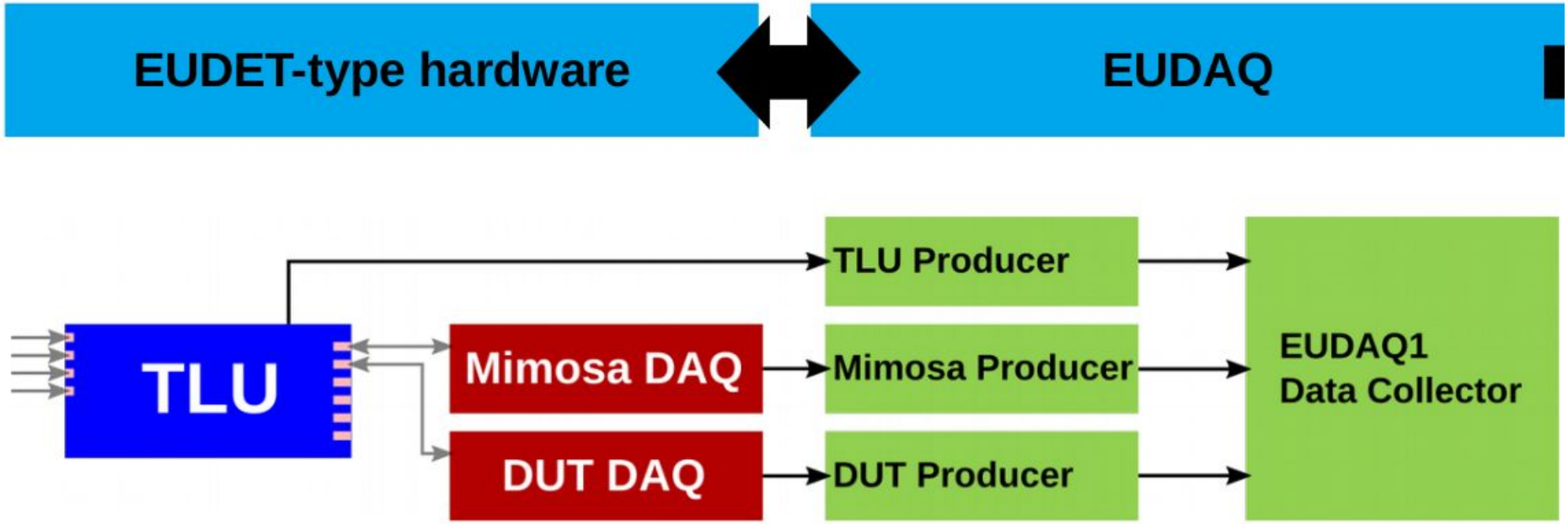




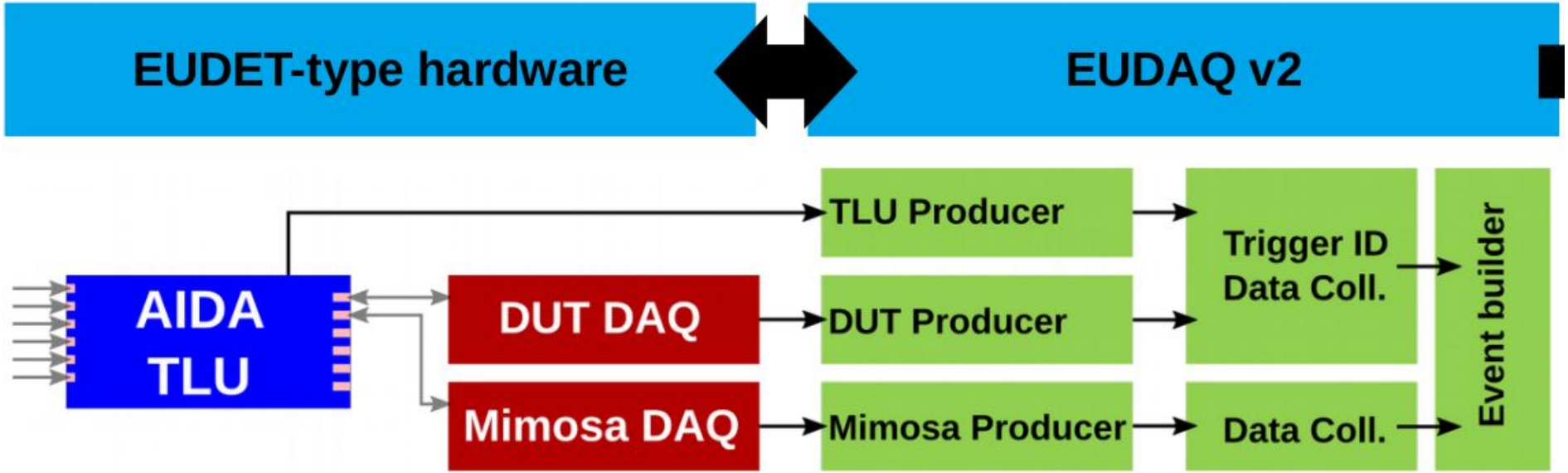
Ressources:

- https://eudaq.github.io/
- https://telescopes.desy.de/User_manual#Running_with_EUDAQ_1

EUDAQ1 and EUDET TLU



EUDAQ2 and AIDA TLU



EUDAQ comparison

EUDAQ 1 – robust

- No AIDA TLU implementation
- Centralized Data Taking with EUDET TLU
  - One Data Collector
  - Synchronisation by (sub-) event number
- Versions
  - Latest release v1.8.0, April 2018
  - Development Branch: v1.x-dev
- Code structure
  - One library
  - Each producer an executable
  - Component-based Structure

EUDAQ 2 – more flexible

- EUDET TLU is implemented
- Decentralized Data Taking with AIDA TLU
  - Multiple Data Collector (and connections)
  - Online or offline synchronsiation by event number, Trigger ID or timestamps
- Versions
  - Latest release v2.1.0, Nov. 2017
  - Development branch: master
- Code improvements
  - Core Library, Converter Library, ...
  - Producer abstraction (modules)
  - User-based file/folder structure

Data Taking Modes for EUDET-type telescopes

#	Mode	Sync.	TLU	EUDAQ	Streams	DataCollector	Event building	Realizations/User
1	EUDET	global busy	EUDET	1	1	DataCollector	Online by DC	EUDAQ1
2	EUDET	global busy	both	2	1	**EventIDSync DataCollector**	Online by DC	ATLAS ITK and EUDET telescope
3	EUDET	global busy	both	2	>1	**DirectSave DataCollector**	Offline by **euCliMerger StandardEvtID**	TORCH and EUDET telescope
4	mixed	Trigger ID	AIDA	2	1	**TriggerIDSync DataCollector**	Online by DC	EUDET telescope
5	mixed	Trigger ID	AIDA	2	>1	**DirectSave DataCollector**	Offline by **euCliMerger StandardTrigID**	EUDET telescope
6	AIDA	timestamp	AIDA	2	1	**TimestampSync DataCollector**	Online by DC	CALICE, BIF and CaliceTelDataCollector
7	AIDA	timestamp	AIDA	2	>1	**DirectSave DataCollector**	Offline by ***TimestampSync EventBuilder***	na

For newcomers: What to do for integration?

- Hardware
  - TLU communication with your Sensor-DAQ
- Software
  - EUDAQ producer on your DAQ-PC for your Sensor-DAQ
    - Init/Conf/Start/Stop/Reset/Terminate commands and data flow
  - EUDAQ data converter, if needed for your analysis
    - Raw to Standard Event for OnlineMonitor/StdMonitor
    - Raw to LCIO Event for EUtelescope
    - ...

Track #0: EUDAQ2 Installation

First installation:

- Get `git`, `cmake`
- Get further prerequisites: https://telescopes.desy.de/EUDAQ#Out_of_box_on_Ubuntu_18
- Follow the instructions: https://github.com/eudaq/eudaq/blob/master/README.md#quick-installation-for-unix

(Try to change files and Re-Compiling or Reset ‘make_clean.sh’ and compiling).

Track #0: EUDAQ2 example execution

Follow the instructions:

https://github.com/eudaq/eudaq/blob/master/README.md#execution

Check the data file with (in `bin`, `-h` option for man page)

```
euCliReader -i FILE
euCliReader -i FILE -e 0 -E 2
```

and similar with

```
euCliTeleReader
```

Change ini/conf file for saving data/log in a different location.

Track #1: Moving from EUDAQ 1 to 2

- Change the folder structure
  → from component-based to user-based folder structure
- Copy/Adjust CMake files
  → have a look at `user/eudet/...` for example
- Update your Producer
  - Apply the module structure
  - Rename functions: `OnInitialise` → `DoInitialise`, etc.
- Update your Converter
  - Apply the module structure and the `id`-name
- Try (and error) it

Track #2: Integrate a Python based-DAQ

- Condition:
  - you have checked out and successfully compile the usual EUDAQ2
  - Python3
- Re-compile with extra option:
  - `EUDAQ_BUILD_PYTHON=ON`
  - or use `$cmake-gui ..`
- Setup python run environment:
  - `cd ${EUDAQ2}/user/example/python`
  - `source setup_eudaq.sh`

Hands-On: Run the python example

- Run the example with Ex0.ini/Ex0.conf:
  - `${EUDAQ2}/bin/euRun &`
  - `python3 ExamplePyProducer.py`
  - `python3 ExamplePyDataCollector.py`
- Exercises:
  - Change the Producer/DataCollector to change/add configurable variables;
  - Change the Producer to change the output event structure;
    - Check with command line tools in `${EUDAQ2}/bin`, see Slide 10: Track #0: EUDAQ2 example execution;
  - NB: yes you do not need re-compilation if you modify your user py modules!
- Bonus: how to integrate a python-based DAQ to EUDAQ?

Track #3: AIDA TLU

- Communication with TLU over Gbit/s Ethernet (UDP/IP)
  - Uses IPBus (developed for CMS, now used by Atlas and many other HEP experiments)
- Installation:
  - Instruction: https://github.com/eudaq/eudaq/blob/master/user/tlu/README.md
  - IPBus/Cactus: https://ipbus.web.cern.ch/ipbus/doc/user/html/software/install/compile.html#instructions
  - cmake .. and Re-compile

Hands-On: Trigger and Data Collection options

Perform a ...

- Autotrigger run
- External trigger run
- Multiple DirectDataCollectors
- (Synchronisation by Events)
- (Synchronisation by Trigger Number)

Check the data with `euCliReader` or `euCliTriggerReader`

AIDA TLU Configuration

- TLU “hardware manual” currently at https://github.com/PaoloGB/firmware_AIDA/raw/master/Documentation/Latex/Main_TLU.pdf (will be moved at some point)
- TLU configuration in INI and CONF files, in [Producer.aida_tlu] sections
  - INI file describes
    - TLU hardware parameters (different in miniTLU)
    - IPBus link to TLU over IPBus (connection file)
    - clock configuration (frequency, input/output, clock source).
      - Clock configuration only need to be done once after start up. Can be repeated if desired but clock will stop during reconfiguration.
  - CONF file describes triggering, PMT bias, DUT connection,

Triggering

- TLU has six Lemo 00 coaxial connectors that can be used as trigger inputs
  - Fixed 50Ω termination
  - Threshold variable between +/- 1.2V
    - Minimum reliable pulse size ~ 5mV
    - Maximum pulse size +/- 5V
  - Currently (version 0x14) triggers on falling edge (high voltage → low voltage transition) ie. leading edge of NIM, scintillator pulses. Will be made configurable in later firmware versions.
- Which combination of inputs produces a trigger is set by trigger mask.
  - See chapter-5 of TLU hardware manual
  - Each of the 64 (2^6) bits in the trigger mask controls if that combinations of inputs produces a trigger.
    - Divided into high and low trigger words
    - e.g. high , low triggers words 0xFFFFFFFF , 0xFFFFFFFE trigger on any combination of inputs except for 0,0,0,0,0,0

Triggering (continued)

- Triggering is edge sensitive (i.e. does not depend on width of input pulse)
  - Can adjust the internal width of triggers
    - units of 1/160MHz , up to 32
    - inX_STR parameter in CONF file
      - e.g in0_STR=5 produces pulse of 5/160MHz = 31.25ns for every transition on input 0
  - ... can also delay each input by up to 32 cycles of 160MHz
- Can also use a fixed frequency internal trigger

Data Recorded by TLU

- One data record written for each trigger
  - This is inefficient, likely to move to multiple triggers in each record.
  - Saves much of the information as text “tags” this is very inefficient.
- Data recorded:
  - Event number and timestamp (number of 40MHz clock cycles since start of run)
  - Which inputs fired
    - even the ones that aren’t demanded by trigger mask. Can be used for e.g. tagging events with threshold Cherenkov detector information
  - “Fine grained” timestamp of last hit on each of the trigger inputs
    - time bins currently 1.56ns wide (hope to improve to 781ps)
      - Fixed threshold discriminators limits time resolution to ~ 1ns
- Data can be decoded into CSV with the euCliTluReader command line program

TLU Information in Run Control

- TLU producer reports some status information to run control
  - Number of triggers processed
  - Number of possible triggers (particles).
    - Particles can be vetoed by
      - run status (i.e. run stopped)
      - DUT busy
      - shutter cycle
      -

Shutter

- One of DUT signals is a “shutter” which indicated when triggers are active
  - Will be used by AIDA-2020 Silicon Strip tracker to synchronize data taking with accelerator cycle
- Shutter sequence can be controlled by a signal on one of the trigger inputs
  - N.B. Remember to disable this input from the trigger mask .....
  - Can also be controlled by internal counter

Tutorial With TLU

- Connect TLU Ethernet link and check network configuration
  - Should respond to "`ping 192.168.200.30`"
- Execute the `01_aida_tlu_local` script in user/eudet/misc/starting_scripts
  - This version assumes that all processes are running on localhost, rather than distributed according to standard telescope network configuration
  - Make sure that `eudaq/bin` is in your `PATH`:
    `export PATH=$PATH:/path/to/eudaq/bin`
- Edit CONF file for desired trigger configuration. You can start from `user/tlu/misc/aida_tlu/aida_tlu_test.conf`
- (Re-)conf and re-start…