Evolving the DAQ and Analysis Software of the AIDA Telescope: Toward high rates and *one-trigger-per-particle* Operation

Ulf Behrens, Alan Campbell, Francesco Crescioli, David Cussans, Hanno Perrey, Richard Peschke, Igor Rubinskiy

July 30, 2014

Abstract

A high resolution ($\sigma \approx 2\,\mu\text{m}$) beam telescope based on monolithic active pixel sensors was developed within the EUDET collaboration. It has become the primary in-beam tool for many high-energy physics groups, largely due to its precise spatial resolution, reliable operation and DAQ integration capabilities. For the telescope to deliver this excellent performance, two software packages play a central role: EUDAQ, a multi-platform data acquisition system that allows easy integration of the device-under-test, and EUTelescope, a group of processors running in ILCSoft's Marlin framework that allows the spatial reconstruction of particle tracks and the final data analysis.

In parallel to their successful operation in test beams for many years, both software packages are under constant development: from the integration of new device types and use-cases, to improvements to usability and flexibility, and the support of new features such as the high-rate capabilities of the next-generation pixel beam telescope developed within the European detector infrastructure project AIDA.

In this contribution, we present the features of the current releases of both EUDAQ and EUTelescope and discuss the plans for further development toward an easy-to-use software stack with the capability for high particle and data rates within the AIDA work package 9.3.

1. Introduction

Beam tests of future tracking devices are crucial to determine their characteristics under realistic operating conditions. By determining the track of a charged particle in a test beam to high precision using a beam telescope, one can perform detailed studies of newly developed detectors.

Originally built at DESY within the EUDET JRA1 project for detector R&D toward the International Linear Colider (ILC), the EUDET beam telescope [1] was designed as an easy-to-use system with well-defined interfaces allowing test beam studies on a short time scale.

Since the first EUDET telescope has been used in beam tests in 2007, it has become the primary beam tool for many groups, largely due to its precise pointing resolution of $\sim 2\,\mu\text{m}$, reliable operation and DAQ integration capabilities.

For the telescope to deliver this excellent performance, two software packages play a central role:

- EUDAQ [2], a multi-platform data acquisition (DAQ) system that allows easy integration of the device-under-test, and
- EUTelescope [3], a library that provides tools for the spatial reconstruction of particle tracks and the final data analysis.

Based on the EUDET telescope, a next generation telescope is being developed within work package 9.3 of the European AIDA project to better fulfill the evolving requirements of the user community. It will provide more than two orders of magnitude higher trigger rates of up to 1 MHz, precise time-stamping in the sub-nanosecond range, and large-area sensor planes of $4 \times 4\,\text{cm}^2$. This is made possible by a new trigger logic unit (TLU) with both faster discriminators and a simplified, shared-clock interface to the connected devices and by replacing one of the two telescope arms consisting of a triplet of MIMOSA26 sensors with three MIMOSA28 quad-planes. Both of these hardware developments are described in detail in [4] and [4], respectively.

To fully exploit the new hardware capabilities, both the DAQ framework and the analysis framework have to be extended to be able to deal with a significant increase in data rates, to support time stamping and multiple triggers per device readout, and to allow for a more challenging (offline) alignment of sensor planes composed of several individual devices.

An ongoing effort is made to incorporate these features and which are foreseen to be released as versions EUDAQ 2.0 and EUTelescope 1.0 later this year.

2. EUDAQ 2.0 – A Flexible High-Rate Capable DAQ Framework

EUDAQ is a generic DAQ framework that has been successfully used in testbeams with the EUDET-family of beam telescopes since 2007. It allows the easy (but optional) integration of the device-under-test and its DAQ into the telescope data stream and gives the user a central interface to control and monitor the operation of the full system. EUDAQ features a very modular design in which individual components communicate via TCP/IP and can run on different networked machines.

The central authority is called *RunControl*. It provides the user with a choice of either a graphical or console-based interface with status information on the current operation and allows to configure, start or stop the data taking. The actual device-hardware interaction with the telescope or the device-under-test is performed by independent *Producers*. Well-documented examples are provided to make the integration as easy and straightforward as possible. All Producers send their raw or already-decoded data to a *DataCollector* which stores it to disk.

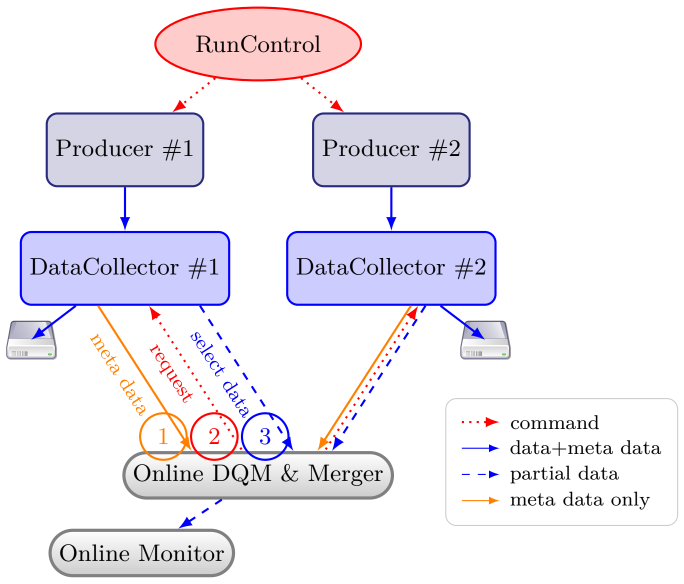

Figure 2.1: A schematic overview of the components of EUDAQ and the foreseen data flow for version 2.0. Shown is the communication between components when writing data from multiple data sources (devices/Producers) to local data sinks (DataCollectors) and monitoring the system online based on a partial sub-set of the stored data merged by a central data quality monitoring process (Online DQM & Merger).

EUDAQ will also be the DAQ framework of choice for the AIDA telescope. A new extended data format will allow for storage of additional meta data such as time stamps and multiple triggers per device readout to make use of the capabilities of the newly developed AIDA TLU.

In order to cope with the high data rates expected from the increased trigger rates and the large-area sensor planes, EUDAQ 2.0 will allow to run multiple DataCollectors simultaneously as illustrated in figure 2.1. Each DataCollector can be assigned to a single Producer and can run locally on the same physical machine as the Producer itself. This makes the setup more flexible and avoids bandwidth bottlenecks such as slow network connections.

The additional meta data available will make more thorough consistency checks and online verification possible: as devices obtain their clock directly from the TLU, synchronization loss and missed triggers can be reliably detected online. For this purpose, the meta data only will be collected from each DataCollector and verified by a central data quality monitoring process. Additionally, the data can be (partially) merged online by requesting packets containing specific trigger ranges – this allows to generate e.g. correlation plots online or even run the data analysis on a sub-set of the data while the recording is ongoing.

Even with these fundamental changes to the data-handling concept of EUDAQ, a lot of effort has been invested to make the modifications backward-compatible in order to preserve existing device integrations by the users.

The current release branch of EUDAQ (v1.X) offers a flexible and proven framework for data taking with multiple devices in a test beam. However, it is based on the concept of one-trigger-per-integration-cycle[1] and it suffers from data-rate limitations when going to high trigger rates. Some fundamental changes going toward a v2.0 (or AIDAQ) are needed.

To keep the amount of work needed small (for us and others) and to allow the project to be completed in small chunks (i.e. by different people), the changes should generally/where possible be kept

- backward compatible[2] to existing DUT integrations,
- minimally invasive, and
- modular.

3. DAQ

High trigger rates will require the ability to store large amounts of data in short time. Therefore, potentially slow links such as Ethernet should be avoided. Allowing each device's producer to connect to its own (local) data collector makes it possible to spread the load onto different machines.

Furthermore, one-trigger-per-particle operation requires to take different device's data delivery concepts and speeds into account. For example, devices integrating over longer periods of time will send data blocks that cover a range of triggers. This breaks the current concept of a synchronous event number across devices.

3.1. New data format: *packets* instead of *events*

Instead of sending the data in *events* corresponding to single triggers, producers send the data in *packets* which can cover arbitrary number of triggers and/or time ranges.

[1]The duration of this integration cycle is determined by the longest-integrating device.

[2]Backward compatible meaning no changes to the *producer's* source code on user's side needed; if any changes are required, these need to be well motivated, kept minor and have to be clearly documented. Binary compatibility with old producers or data files on the other hand is not being foreseen.

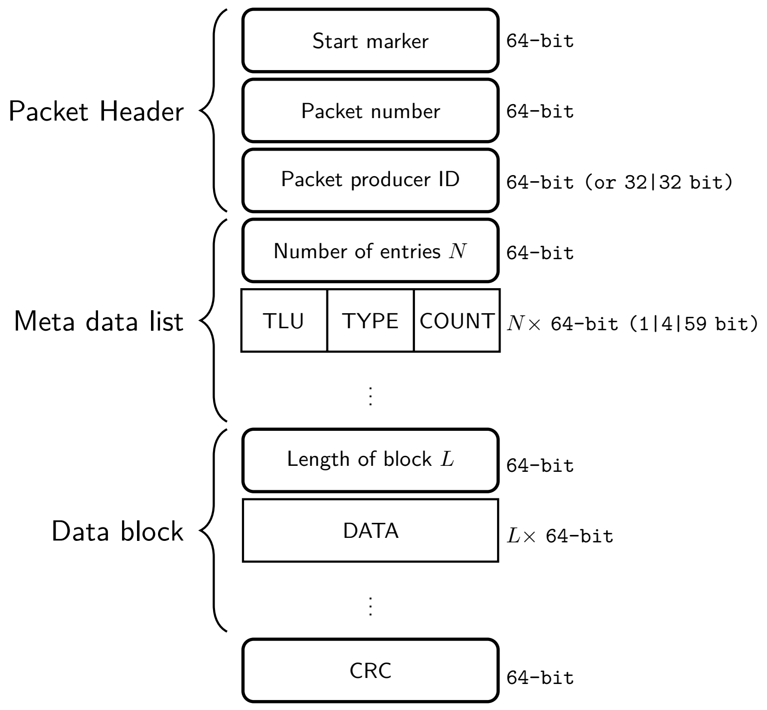

Figure 3.1: The new data format: packets structured into list of meta data (timestamps/counter values) which describes trigger and/or time ranges and the corresponding block of data.

This allows to integrate different DAQ concepts into EUDAQ, such as untriggered or data-driven devices.

The meta data format is designed to allow partial data merging on request via a pull mechanism (see section 3.3).

- each device records and sends its data in arbitrary units "natural" to its operation (e.g. a set of frames for Mimosas)
- the **packet header** consists of:
  - a start marker
  - the *packet number* is set by each producer for every data block send to the data collector (starting at 1 for every run)
  - the producer ID field describes the detector/producer from which the data originate and which decoder to use; possibly to be divided into major and sub fields.
- the **meta data** to each packet consists of a length field and a list of counter entries with three fields:
  - *TLU* bit to indicate whether or not this counter corresponds to a TLU signal,
  - a *TYPE* ID with a width of 4-bit which indicates the nature of the counter, e.g. trigger number, clock count (i.e. timestamp) of a TLU-event, or timestamp of begin/end of packet, and
  - the 59-bit wide counter value.

  This allows to store as meta data:
  - a list of trigger numbers,
  - a range in time during which the data was recorded, and/or
  - a list of trigger timestamps.
- **Trailer**: A 64-bit field for storing a *check sum* (optional)

Implications:

- as long as the old data format (one packet per trigger) is still supported, all existing DUT producers should work as expected after recompiling with the new underlying packet design.
- the packet number does not (necessarily) match the trigger number and is not synchronous between devices. Specifically, the size and delivery rate of events is different for each device.
- run control, data collectors and online monitors need to be adjusted so support the new format.

Remaining questions/issues/details yet to be decided:

- Can the packet format be introduced without having to adjust any code in the producers?
- Can the old "event" be renamed "packet" for clarity and consistency without having to make significant code changes in existing producers?

- What meta data types (*TYPE* field) need to be supported? For current list of suggestions, see table 3.1.
- What/how many detector/packet types (producer ID field in packet header) need to be supported and where should they be defined? For a list of suggestions, see A. Should the field be separated into two fields (e.g. each with 32-bit width), one "major" and one "sub-id" field? Should the values be fixed and centrally defined or should they be freely assignable by each producer?

TYPE	**description**
0000	internal trigger – *reserved for internal TLU usuage*
0001	external trigger
0010	shutter falling
0011	shutter rising
0100	edge falling
0101	edge rising
0110	spill off
0111	spill on
1011	start/end of packet data recording
1111	first/last trigger *number* contained in packet

Table 3.1: Codification of the meta data *TYPE* field in the new packet format based on TLU signals[4]. The specific value defines how the *COUNT* field of the meta data entry is interpreted. An additional *TLU* bit indicates whether or not the counter value is based on the central clock provided by the AIDA TLU.

3.2. Spreading the load: multiple data collectors

In order to avoid slow network links and to be able to store data on local machines, EUDAQ should support multiple data collectors, i.e. one per (device) processor as illustrated in figure 3.2.

Implications:

- Producers need to be associated with specific data collectors; possible approaches:
  - run control assigns producers to data collectors; the appropriate data collector could be determined by the configuration file or by identical processor and data collector names.
  - every producer is informed of every available data collector by run control and chooses one according e.g. to its name; this also allows for example the TLU producer to send its data to *every* data collector. **Note:** Need to verify whether or not the producer-side modifications are limited to the `Producer::OnData()` implementation or if changes to actual producer's code would be necessary.

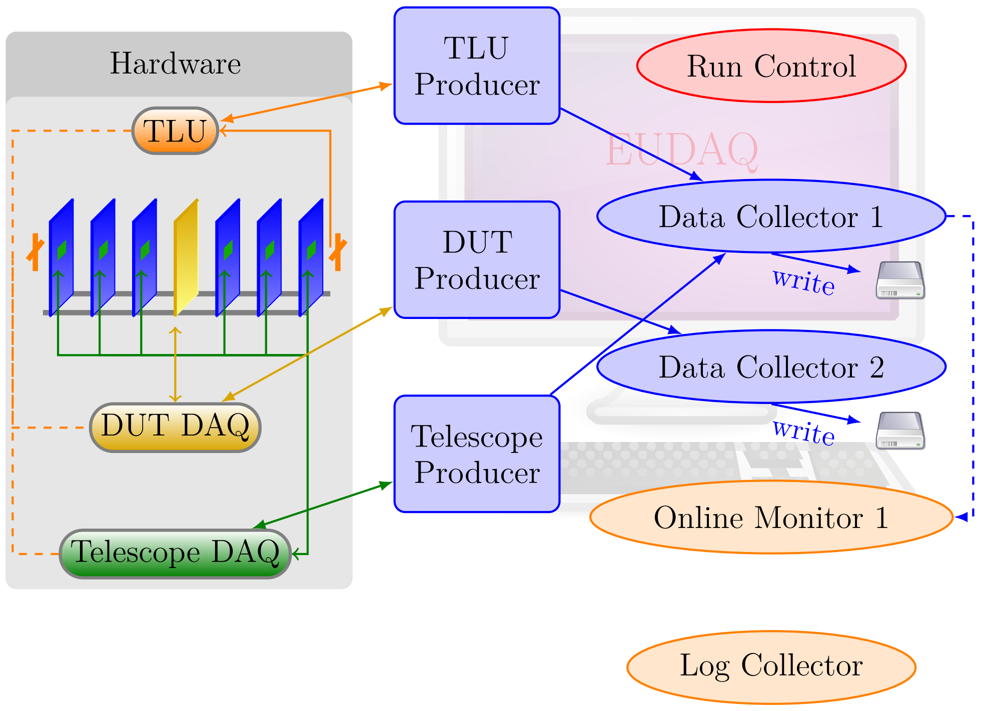

Figure 3.2: Schematic layout of the proposed system with multiple data collectors each writing onto a separate disk.

- online monitoring becomes more complicated as the data has to be collected from several sources (discussed in section 3.4) but a single instance of the online monitor could run on each data collector as before.

Issues/to be decided:

- by what method should producers be associated with specific data collectors?
- should the TLU producer's data be treated special in any way? It could be sent to a separate data collector or even to all that are available.
- will one data collector per processor be required or do we have a (central) default one as fallback?
- if one data collector has more than one producer assigned, how is the storage of the received events managed? We could foresee an index file containing a copy of the meta data and a seek position for the main binary file, allowing faster access to specific packets.

3.3. Online data integrity checks and selected data packet reading

Depending on how the devices are integrated (i.e. whether they are running synchronous or not), quite thorough consistency checks are possible using the trigger and clock information from the TLU:

- correct trigger IDs with matching time stamp
- correct trigger IDs
- device active during all triggers (from time stamp ranges)
- total number of triggers (difficult during run time if the trigger rates are constantly high, assessable with certainty only after run stop)

This task will be performed by a central *run monitor* processor while the retrieval of the packets will be performed by *reader* processors as shown in figure 3.3. These collect packets (or optionally their meta-data only) by accessing the files written by the data collectors and retrieving specific trigger and clock ranges. The run monitor can write these packets to disk, creating a sub-set of the available data corresponding to specific triggers that would be suited for further processing by an online monitor or an offline analysis.

To be decided/implications/notes:

- the run monitor could start by requesting meta-data only by the readers; verifying consistency and finding common ranges suitable e.g. for determining correlations it could then proceed to request specific packets for each device from the readers (correlation logic only inside run monitor).
- alternatively, the readers can find the packet(s) corresponding to a TLU trigger or TLU clock range on this information alone; this would allow e.g. the configuration of a specific reader with additional device-specific information (e.g. clock speed) but would probably require TLU information to be present inside each data collector's file.

- the reader processor will use file-system access to the data (either on cache or on RAM disc); an index file containing all the packets' meta information and seek position for the main binary file could speed up the time needed to retrieve the desired packet.
- if a device's DAQ does not include fully-qualified identifiers for the packet meta-data to associate it to a trigger (i.e. TLU trigger number and/or TLU clock-based timestamp or range), the information of the TLU might be needed to determine the correct packet(s); it could therefore be useful to have the TLU producer send its data to all data collectors
- the association reader processor and data collector could be handled as discussed in section 3.2 (e.g. by name)
- the reader processors should be separate from the full system in such a way, that their crashing does not impact the data taking; furthermore, reader processors and run monitor should be able to reconnect to a running DAQ. This could be accomplished by having run control announce all available data collectors and log collectors to reconnecting processors that are neither device data-sources or device data-sinks and sending the currently active configuration file (in-run configure and start).
- alternatively, the reader and run monitor constitute a separate software that does only minimally or not at all interact with the core EUDAQ system; this could provide benefits for the stability but might make integration and configuration more complicated.

This run-monitor triggered data retrieval mechanism can be modified to include TLU-issued markers that are signaled to all connected DAQ systems, allowing hardware-aided pre-selection of packets: this can be performed in two steps. First, a hardware signal from the TLU to all DAQs is issued on request to allow very fast devices to cache packets (e.g. into a separate data collector) for later retrieval by the *run monitor*. This TLU signal is timestamped and included in the next packet to mark it. Then, the *reader processors* are requested to send all packets containing such TLU marker timestamp. If a DAQ does not include a timestamp for this signal in its packets, either the run monitor or the *reader* determines the device's packet(s) that would correspond to the time/trigger range containing the marker signal and retrieves it.

At first, only the latter mechanism (purely run-monitor-driven request for specific trigger number or clock range) will be implemented.

To be decided/discussed:

- Should the reader and run monitor be implemented in the EUDAQ framwork or work as separate tools?
- How could the algorithm for identifying correct data packets from meta data work in the various use cases and levels of TLU integration?
- Specifically, how to treat DAQs running without TLU-synchronization?
- How does the *run monitor* access the information collected by the TLU producer?

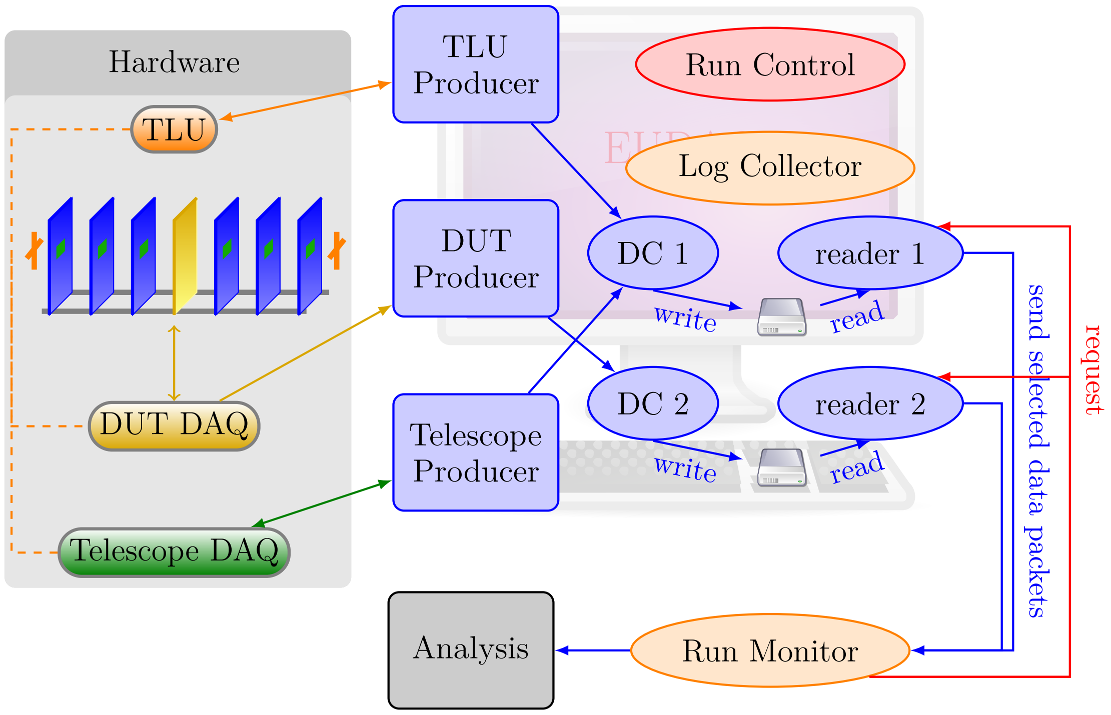

Figure 3.3: Schematic layout of the proposed collection mechanism for reading selected data packets corresponding to specific trigger timestamp/number: (1) the run monitor requests meta-data from all readers; (2) the run monitor makes consistency checks on the meta data and requests data packets corresponding to a set of triggers/time stamps; (3) the full data packets are read by the readers either from disk or cache and sent to the run monitor, where they can be further processed e.g. by an offline analysis framework.

3.4. Online monitoring

Online monitoring (i.e. correlation plots) potentially very difficult:

- data stream are now asynchronous between devices
- data is stored at several locations
- data blocks are potentially large i.e. spanning many triggers

However, online monitoring of a device class (e.g. all Mimosa planes) is still possible with only minor adjustments to the current online monitor.

Correlations, even of tracks fitted to aligned hits, could be studied in immediate offline analysis.

Using the polling mechanism discussed before that flushes e.g. ranges of triggers to a central location, the integrity checks could be extended to a semi-online monitoring based on an offline analysis of the collected data.

Advantages of using the offline analysis framework:

- very modular approach
- all work spent on integration benefits the later analysis
- data is thoroughly and in more detail verified than (easily) possible with pure online monitoring

4. Analysis

The analysis chain will be based on the EUTelescope framework. The goal is to deliver time-tagged tracks from the telescope that can be matched to the DUT data.

The conversion to LCIO, clustering, and transformation of timing information to the TLU clock as global reference can be performed on each data sample separately. But for track finding, all data belonging to a given trigger needs to be present to allow spacial matching. The work flow for the telescope data (Mimosa26 and timing reference plane) could look as follows:

1. convert data to LCIO
2. use information from TLU to synchronize timing information to the TLU clock
3. verify that all triggers are present and have been seen at the correct time; correct if possible (optionally).
4. store hits in x, y, z and t_1, t_2 coordinates, where the latter determine the time bin in which this hit occurred.
5. do spacial track matching, associate the smallest time interval found in a fitted hit with the track.
6. match time-tagged telescope tracks with DUT for analysis.

Implications:

- after repeating this for all data streams, the resulting LCIO event corresponds to a specific trigger ID and will have data collections from all devices.

- still, the final event might not be assigned a single but several triggers if they were issued quicker than the integration time of the fastest device.

Requirements/changes needed/to be discussed:

- make TLU information centrally available and store it in LCIO format
- time stamps t_1 and t_2 in the LCIO event (what floating point precision is required?)
- write TLU clock sync for Mimosa26 data with multiple triggers per frame
- figure out how to efficiently handle multiple LCIO files and merge collections spanned across several events ($\rightarrow$ ask ILCSoft developers).
- decide whether to output time-tagged telescope tracks or to handle data of telescope and DUT simultaneously when merging
- produce monitoring/correlation histograms useful during data taking
- provide geometry information up front (e.g. within EUDAQ)
- optimize the analysis chain to work out-of-the-box in as many cases as possible (e.g. automatic iterative alignment)

5. Acknowledgments

The research leading to these results has received funding from the European Commission under the FP7 Research Infrastructures project AIDA, grant agreement no. 262025. The support is gratefully acknowledged. *Disclaimer*: The information herein only reflects the views of its authors and not those of the European Commission and no warranty expressed or implied is made with regard to such information or its use.

A. Packet header: *Packet type* field values

Values to be supported for the major field:

1. TLU
2. Mimosa26
3. ATLAS FEI-4
4. CMS Pixel
5. TimePix2
6. TimePix3
7. . . .?

B. Mimosa Readout

B.1. Firmware

- replace existing firmware with “official” one from IPHC Strasbourg
- request support for continuous readout
- use global clock from TLU?

B.2. Producer/connection to EUDAQ

- drop current NI producer and write Mimosa producer interfaced to Strasbourg DAQ software
- potentially needs to run on Windows; communication to Strasbourg DAQ through TCP/IP interface?
- Clocking out of the TLU trigger ID should only be performed once per frame maximum.
- when packaging data to be sent to data collector, “pad” buffered frames with one earlier and one later frame to guarantee that triggers received during the readout are contained in this data block.
- record position of pivot pixel at time of trigger *or* use global TLU clock and time-stamp beginning/end of frames

References

[1] *EUDET/AIDA Pixel Beam Telescopes References and Documentation.* URL: `http://beam-telescopes.desy.de`.

[2] *EUDAQ Documentation and Source Code.* URL: `http://eudaq.github.io`.

[3] *EUTelescope Documentation and Source Code.* URL: `http://eutelescope.desy.de`.

[4] F. Crescioli, D. Cussans, and A. Dosil Suárez. *AIDA mini-TLU manual.* 2014. URL: `http://www.ohwr.org/projects/fmc-mtlu`.