Hawaiian Muon Beam Line: Milestones

- Readout Telescope
  - Ethernet Firmware
- Triggering System (TLU)
- DAQ Integration (EUDAQ)
- DQM Plots
  (Correlation plot / Online Monitor)
- Track Fitting (EUTelescope)
  - Residual Plots ➔ Pointing Resolution
- DUT Integration
  - Trigger Integration
  - DAQ Software Integration
  - DQM (Correlation plots etc.)
- Data Reconstruction
  - EUTelescope
- Data Analysis
  - EUTelescope/Root/BASF2/Python/Matlab

Status of Ethernet SCROD Firmware:

- Can Set Registers for Target X
- Stores Trigger Bits in FIFO
- Gets read out from a PC
- PC has to request data from SCROD

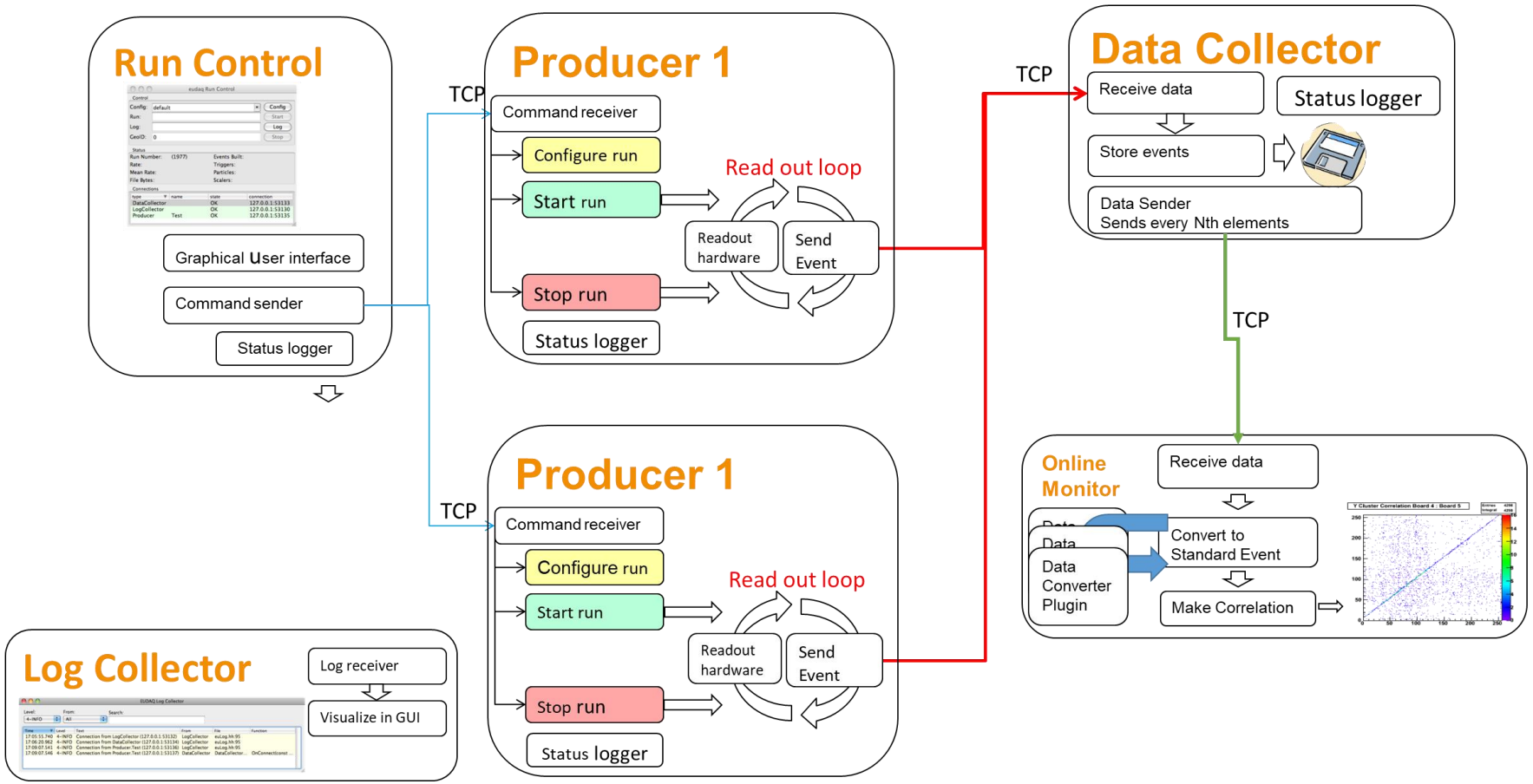
Status EDUAQ Integration

- Python Based SCROD producer
- Can set register values “onConfigure”
- Can Readout Data
- Data Converter Plugin general module, needs some tuning after the detector geometry etc. is decided

Open Question:
Where should we put our source code?

- IDLAB
- Stash
- GitHub

EUDAQ 2.0 Software Layout



- **Run Control:**
  - Central authority
  - Starts, stops and configures runs
  - Assigns Data Collectors to Producers
  - Main user interface
- **Producer:**
  - Interface with user's readout system
  - Receives command from Run Control
  - Sends its data to the Data Collector
- **Data Collector:**
  - Receives data from multiple producers
  - Producer can have their own Data Collector on local machine
- **Log Collector:**
  - Stores and displays log information
  - Start time, stop time
  - Errors and warnings

Configuration File

```
# example config file: Ex0.conf
[RunControl]
EX0_STOP_RUN_AFTER_N_SECONDS = 600
# from the base RunControl.cc
EUDAQ_CTRL_PRODUCER_LAST_START = my_pd0
EUDAQ_CTRL_PRODUCER_FIRST_STOP = my_pd0
# Steer which values to display in the GUI:
# producerName and displayed value are
# seperated by a ",".
ADDITIONAL_DISPLAY_NUMBERS = "log,_SERVER"

[Producer.my_pd0]
# connection to the data collector
EUDAQ_DC = my_dc
# config-parameters of the example producer
EX0_PLANE_ID = 0
EX0_DURATION_BUSY_MS = 10
# EX0_ENABLE_TIMESTAMP = 0
EX0_ENABLE_TRIGERNUMBER = 1
EX0_DEV_LOCK_PATH = mylock0
```

- Standard .INI format
- Comments start with a “#” symbol
- Sections start with [<sectionName>]
  `[RunControl]`
- Key value pairs:
  `EUDAQ_DC = my_dc`
- Key: EUDAQ_DC
- Value: my_dc
- Is stored in the data file
- Since the sections and keys don’t have to be used by the program it is a very convenient place to store additional information for the run
- For example:
  - Voltages
  - Detector Setups
  - Firmware versions

Do Configuration

```
def DoConfigure(self):
    print ('DoConfigure')

    self.config_sender.IP      =  self.GetConfigItem("config_sender.IP)
    self.config_sender.Port    =  self.GetConfigItem("config_sender.Port")
    self.config_sender.path    =  self.GetConfigItem("config_sender.path")
    self.config_sender.header  =  self.GetConfigItem( "config_sender.header")


    self.data_sender.IP      =  self.GetConfigItem( "data_sender.IP")
    self.data_sender.Port    =  self.GetConfigItem( "data_sender.Port")
    self.data_sender.path    =  self.GetConfigItem( "data_sender.path")
    self.data_sender.header  =  self.GetConfigItem( "data_sender.header")


    allAsics = []
    for i in range(11):
        allAsics.append(TargetXRegisters(i, self))

    allAsics.append(SCROD_registers(self))

    saveConfigFile(configCSV, allAsics)
    self.config_sender(configCSV,configCSV_HWout )
```

- On Configuration the “Producer” the configuration as key value pairs.
- Only the configuration for this producer are used
- User can retrieve the values by using the “GetConfigItem” function provided by the Producer class

Run Loop (Readout Loop)

```
def RunLoop(self):
    try:
        print ("Start of RunLoop in KLMPyProducer")
        trigger_n = 0;
        while(self.is_running):
            self.data_sender(
                tx_triggerbitmonitor_top,
                tx_triggerbitmonitor_top_out_HW
            )
            data = load_data(
                tx_triggerbitmonitor_top_out_HW
            )
            ev = pyeudaq.Event(
                "RawEvent",
                "KLM_TriggerBits"
            )
            ev.SetTriggerN(trigger_n)
            block = bytes(data, 'utf-8')
            ev.AddBlock(0, block)

            self.SendEvent(ev)
            trigger_n += 1
            time.sleep(1)
        print ("End of RunLoop in  KLMPyProducer")
```

- On starting the run the RunLoop function of every “Producer” is called
- This function Runs on an separate thread to not interfere with the command receiver
- In general EUDAQs design idea was that the devices receive a hardware trigger and send the data to the readout (Producer)
- In this case the device is constantly collecting data and is periodically readout out by the Producer
- EUDAQ 2.x Is fine with both ways of readout

Data Converter Plugin

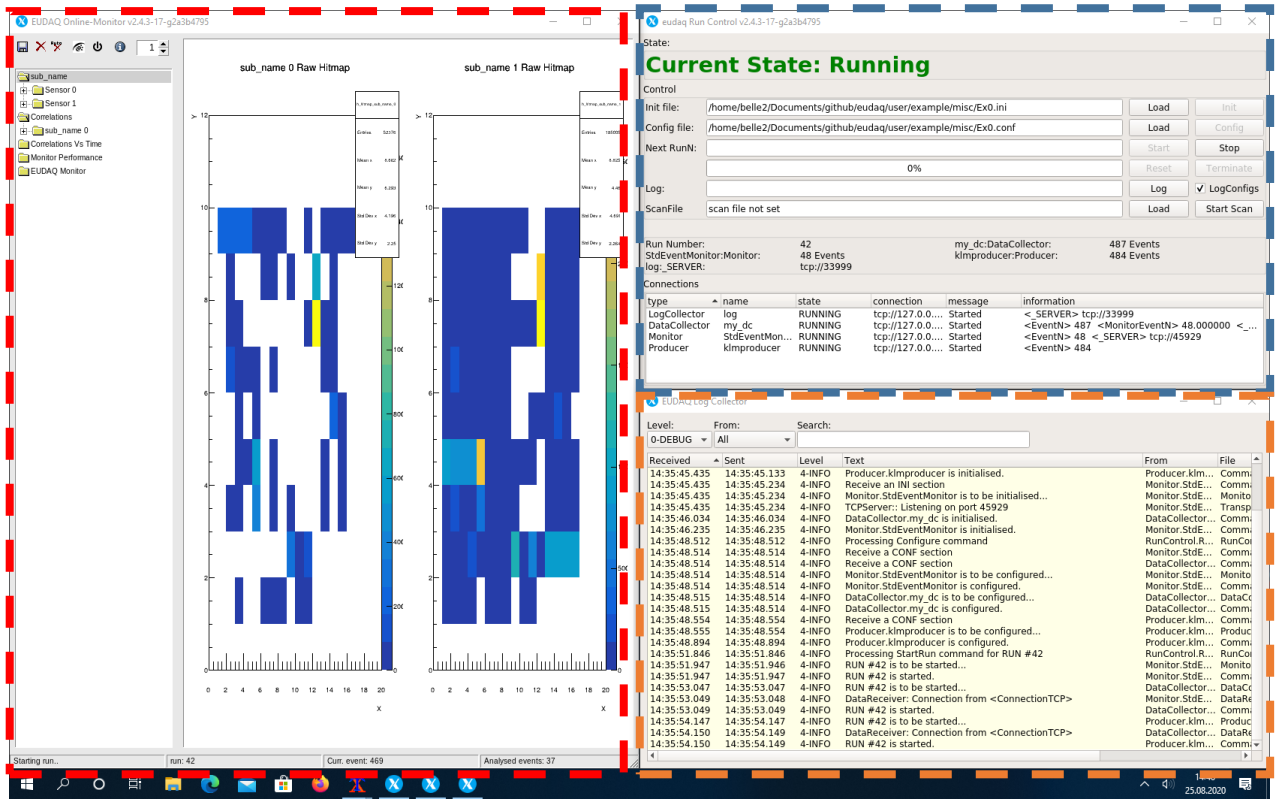
```
bool Ex0RawEvent2StdEventConverter::Converting(
          eudaq::EventSPC ev,
          eudaq::StdEventSP d2,
          eudaq::ConfigSPC conf) const{


 …
  auto b = ev->GetBlock(0);
  std::string block (b.begin() ,b.end() );

  eudaq::StandardPlane plane(0, “KLM", “KLM");
   plane.SetSizeZS(20, 12, 0);
  …
     for (const auto& e : data){
      ++y;
      //std::cout << e<< "; ";
      if(e < 16 && e >0 ) {
      safe_push_on_plane(plane, e , y , 1.0);
      }
  …

  d2->AddPlane(plane);
  return true;
}
```

- Data Converter Plugins are colled to convert RawDataEvents to Planes in an “StandardEvent”
- These plane consists of XY Coordinates and a charge value
- Plane ID needs to be unique for all Event types in the whole system
- Each Detector can push multiple planes to the StandardEvent
- Standard events are used for the Converter (to other data formats such as root, csv, LCIO) and the Online Monitor

Status of EUDAQ Integration

Online Monitor

Run Control



Log Collector

- Python Based SCROD producer
- Can set register values “onConfigure”
- Can Readout Data
- Data Converter Plugin general module
- Integration into Online Monitor
- Converter to TTrees and CSV files

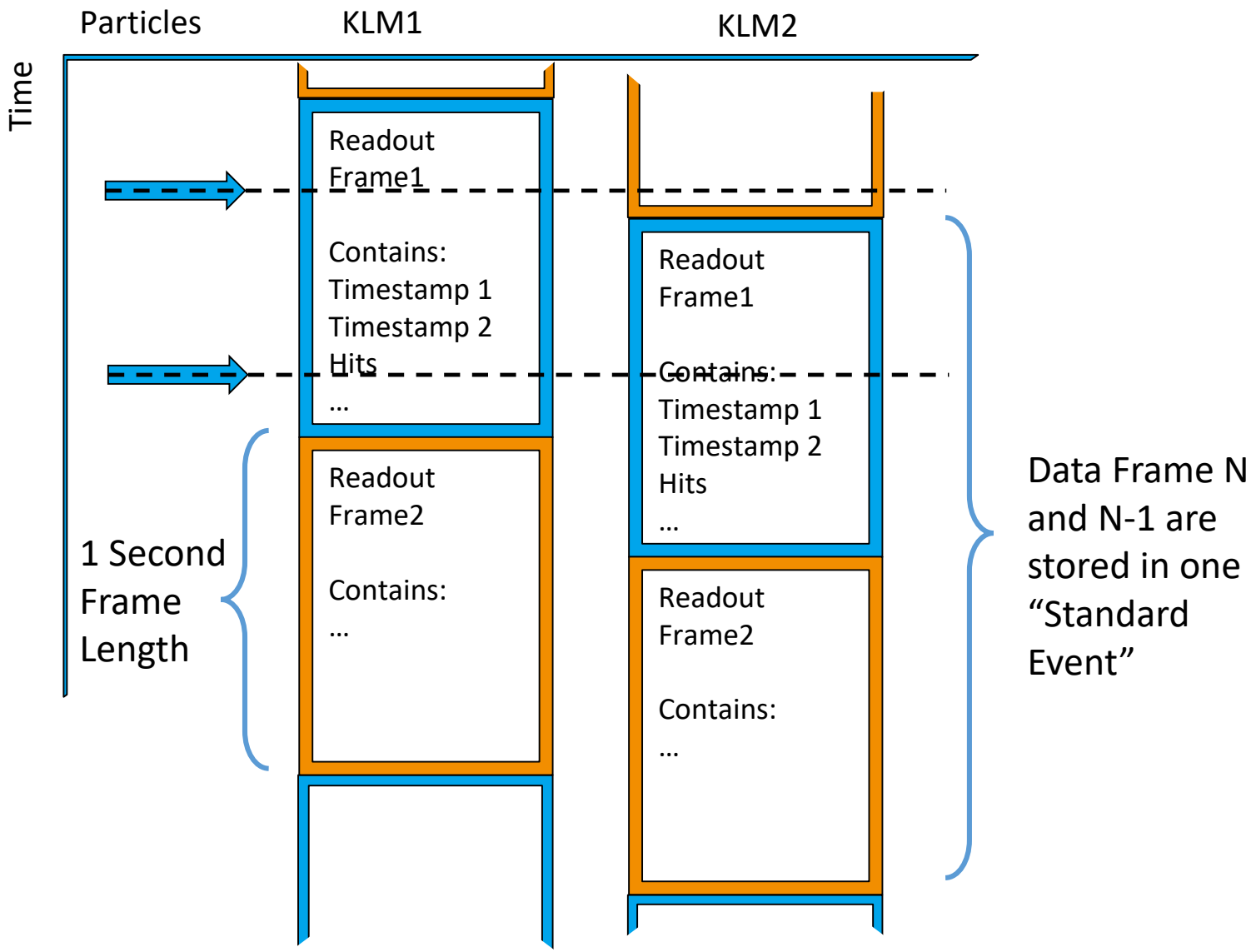
Hawaiian Muon Beam Line: Milestones

- Readout Telescope
  - Ethernet Firmware
- Triggering System (TLU)
- DAQ Integration (EUDAQ)
- DQM Plots
  (Correlation plot / Online Monitor)
- Track Fitting (EUTelescope)
  - Residual Plots ➔ Pointing Resolution
- DUT Integration
  - Trigger Integration
  - DAQ Software Integration
  - DQM (Correlation plots etc.)
- Data Reconstruction
  - EUTelescope
- Data Analysis
  - EUTelescope/Root/BASF2/Python/Matlab

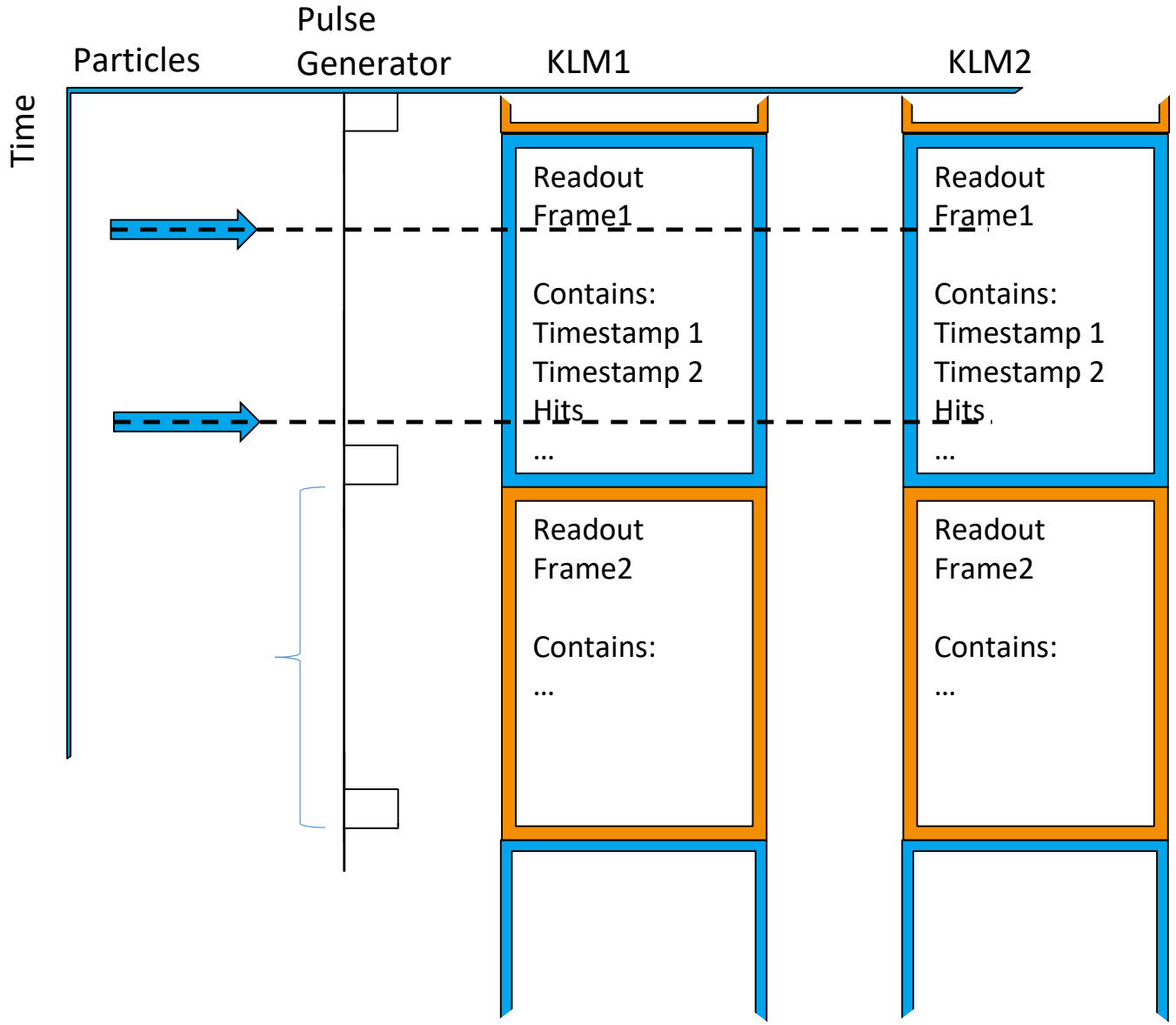
Triggering system
- No TLU
- Continues Readout ?

Open Question:
Where should we put our source code?
- IDLAB
- Stash
- GitHub

Continues Readout (Software Trigger)



Continues Readout (Hardware Trigger)



- Use Pulse Generator to synchronize readout
- Accumulate all hits until Trigger
- Readout on trigger from pulse generator

➔Synchronized Readout Frames

➔Shorter Readout Frames