dsadc analyzer

Andrea
TRIUMF
29 September 2021

5 February 2025 note

- These slides were prepared to train students in the context of DarkSide-20k SiPM development.
- Hence the names DS-*something*
- Slides for general consumptions are the ones between 6 and 11
- A git repo of a working code based on `manalyzer` can be found in the last slide

What is this?

- MIDAS data format analyzer
- Tailored to CAEN ADCs (DS use)
- Fast analysis tool thanks to
  - Modularity
  - Multithreading
- Suitable for online monitoring
- Easily integrated into MIDAS thanks to JSROOT

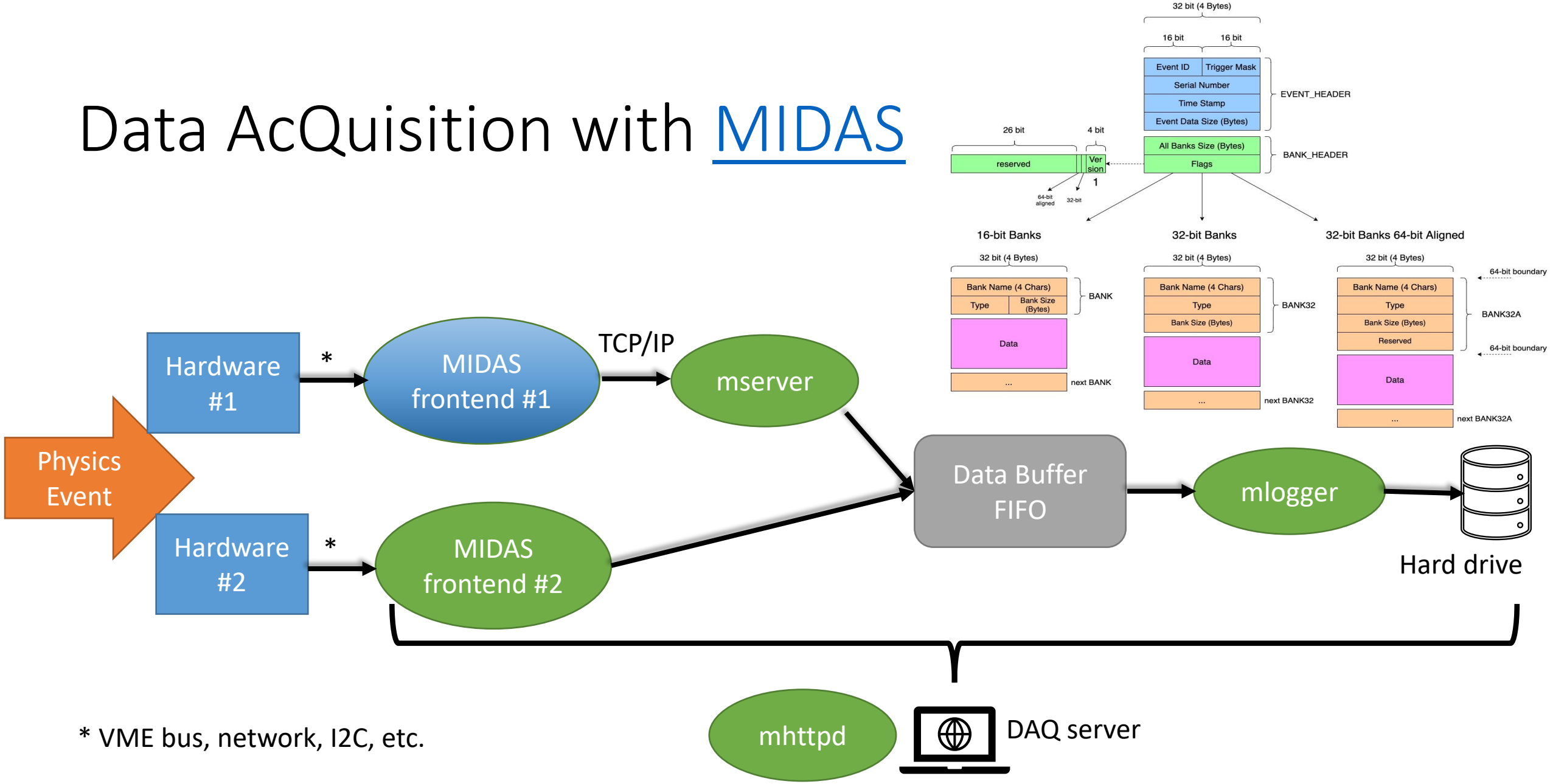
What is not?

- Competition/replacement pyreco*
- Complete and fully-featured analyzer
- Software that underwent code review

*standardized analysis toolkit for DS

Midas analyzer
the Basics

Data AcQuisition with MIDAS

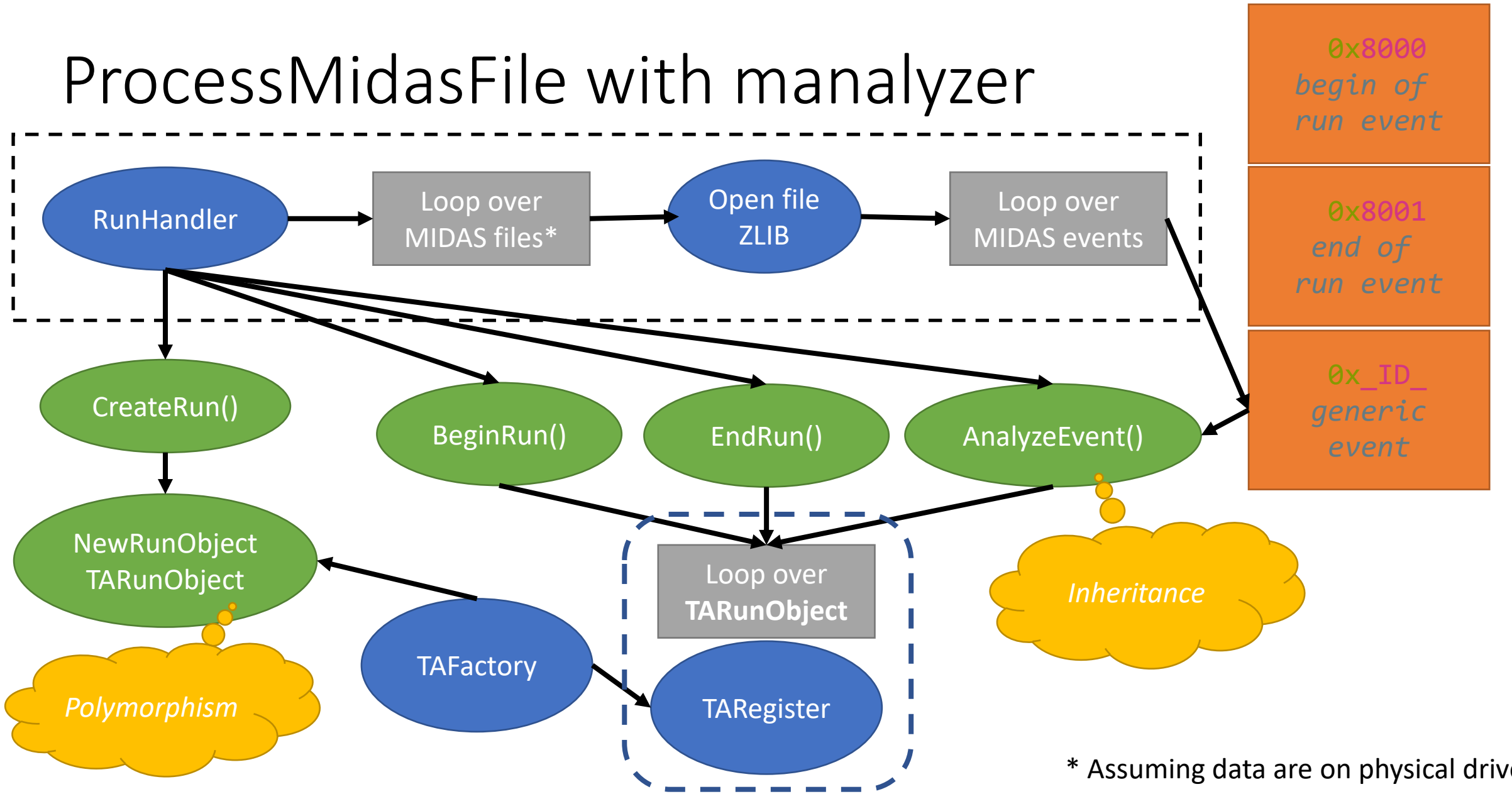


* VME bus, network, I2C, etc.

Manalyzer

- ROOT based data analysis package for MIDAS
- manalyzer is originally developed by K. Olchanski
- manalyzer provides an analysis framework through:
  - inheritance and
  - polymorphism

ProcessMidasFile with manalyzer



* Assuming data are on physical drive

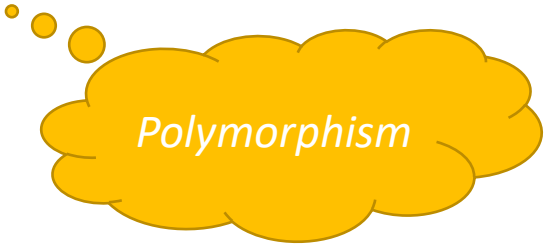
A Modular Approach #1

```
class MyAnalysis: public TARunObject
{
private:
   int fPar;
...
public:
   MyAnalysis(TARunInfo* runinfo, MyAnalysisFlags* flags): // ctor
                                                    TARunObject(runinfo), fPar(flags->newpar)

   void BeginRun(TARunInfo* runinfo)
   void EndRun(TARunInfo* runinfo)

   TAFlowEvent* Analyze(TARunInfo* runinfo, TMEvent* event, TAFlags* flags, TAFlowEvent* flow)

   // your analysis generally goes in here:
   TAFlowEvent* AnalyzeFlowEvent(TARunInfo* runinfo, TAFlags* flags, TAFlowEvent* flow)
...
};
```

Public Interface

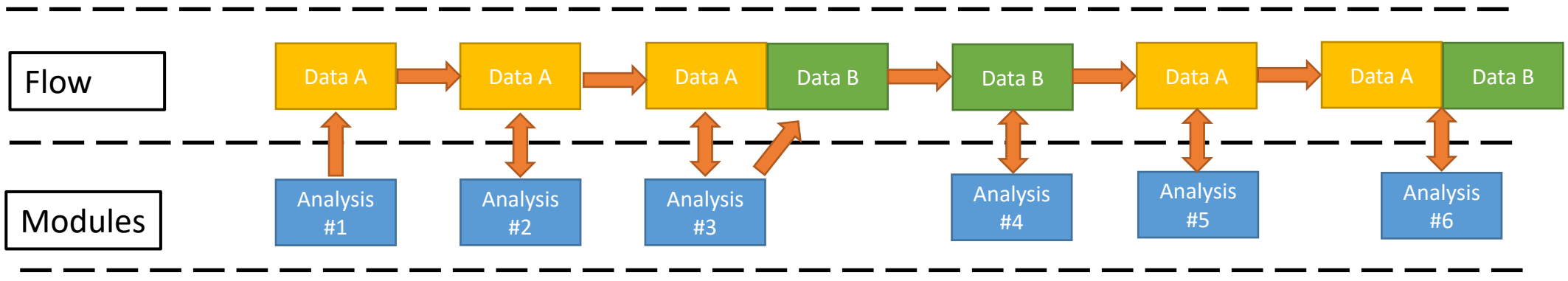
A Modular Approach #2

```
class MyAnalysisFactory: public TAFactory
{
   MyAnalysisFlags fFlags;
...
public:
   void Init(const std::vector<std::string> &args) // read command line arguments after '--'
                                                   // and pass them to module
      {   if(args[i] == "--myparameter")  fFlags.newpar = atoi(args[++i].c_str());  }
 ...
   TARunObject* NewRunObject(TARunInfo* runinfo)
   {
      return new MyAnalysis(runinfo, &fFlags);
   }
};

static TARegister tar(new MyAnalysisFactory);
```



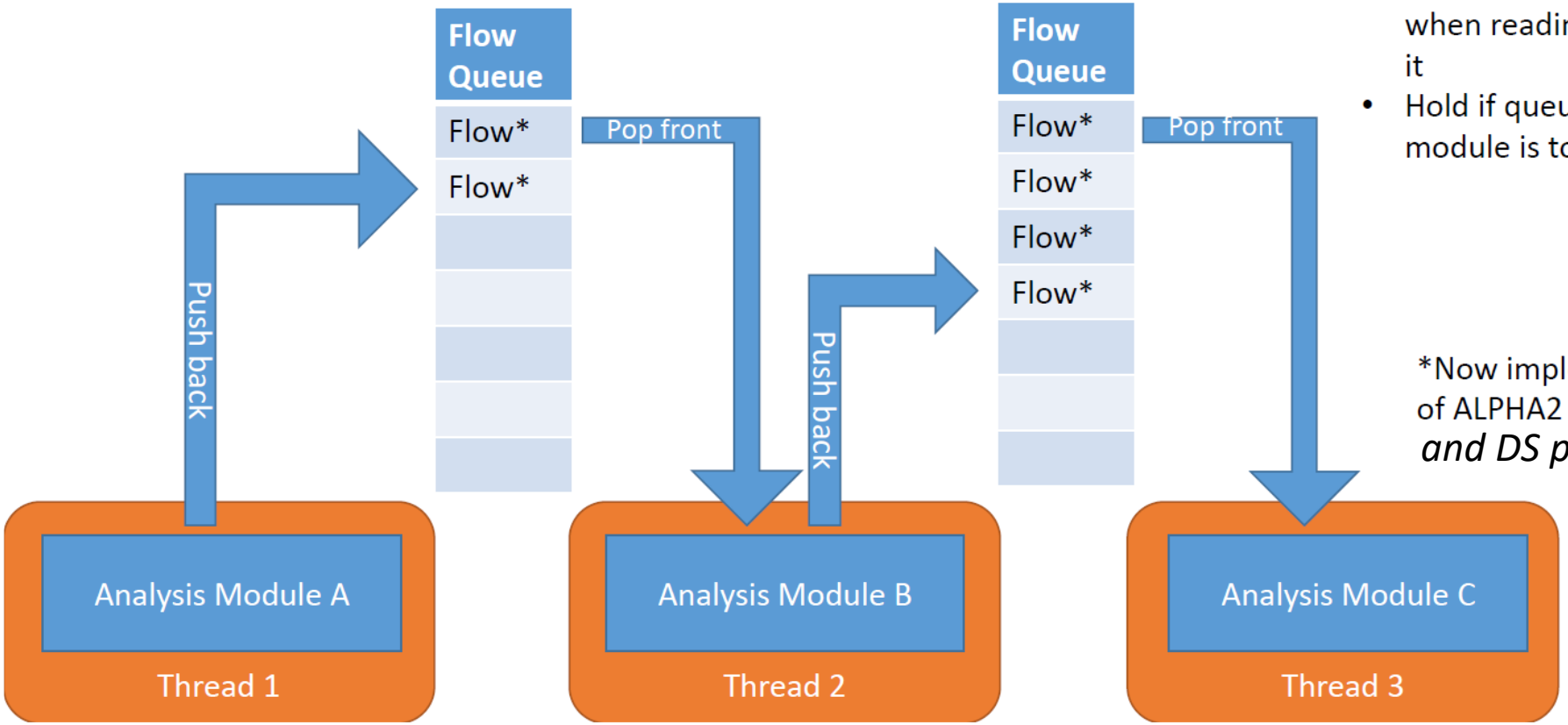
The concept of Flow

- Pass refined data between modules
- Created or modified inside TARunObject:: AnalyzeFlowEvent(…)
- Create your own flow object through the public interface of TAFlowEvent
- Use template method Find() to retrieve the flow object that contains the data that you need



Concept of multithreaded flow

Credit JTK McKenna



Key steps:

- Modules A,B,C all are their own threads, queues are communication
- Thread lock Flow queue when reading and writing to it
- Hold if queue for next module is too long

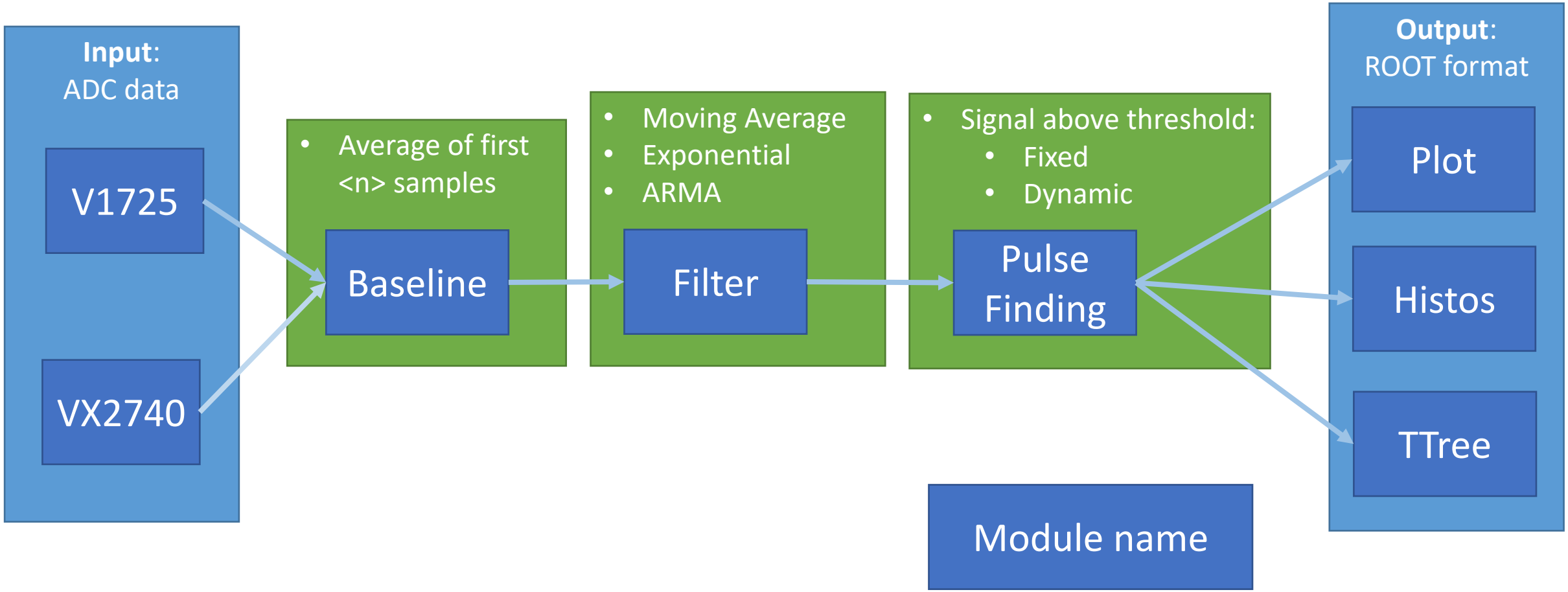
*Now implemented as part of ALPHA2 and ALPHAg *and DS proto-1*

`dsadc` analyzer

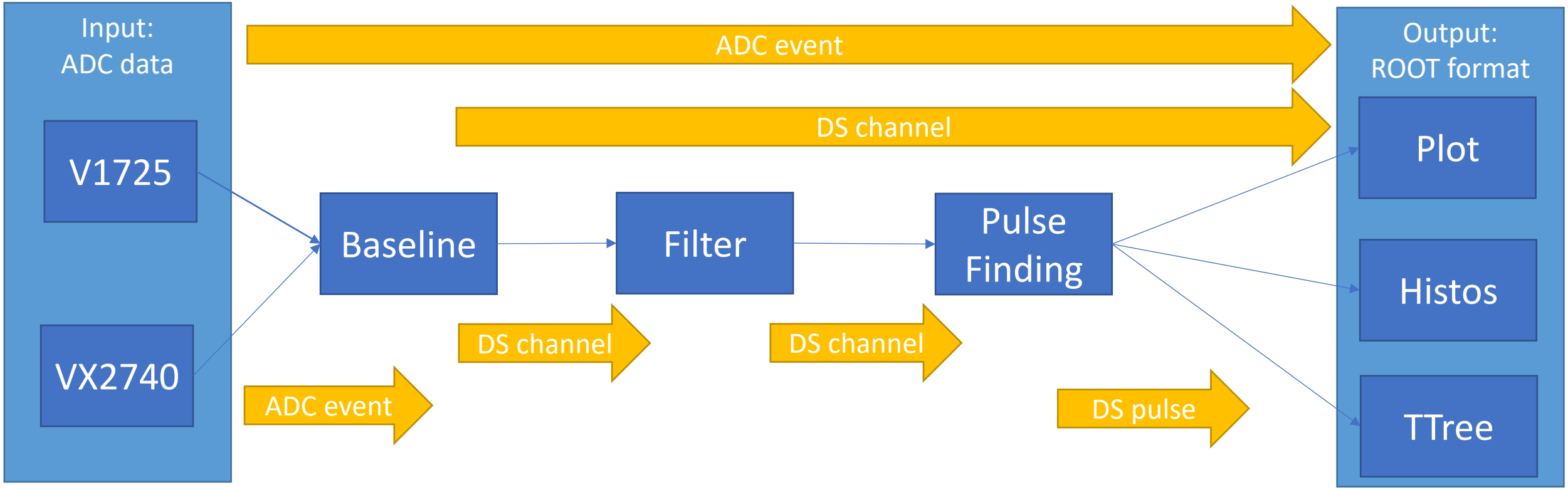
General information

- Based on dsproto_analyzer.
  - Maintains some of the original features, e.g., data structures
- Written in C++ with the manalyzer framework
- Each analysis task is performed by a module `TARunObject`
- Output format is ROOT: `TH1`, `TTree`, `TCanvas`, etc.
- Throughput (ballpark): 200 MB/second
- Embedded profiler
- Easy multithreading

Tasks View


Input:
ADC data
V1725
VX2740
Average of first <n> samples
Baseline
Moving Average
Exponential
ARMA
Filter
Signal above threshold:
Fixed
Dynamic
Pulse Finding
Output:
ROOT format
Plot
Histos
TTree
Module name

Tasks View


Input:
ADC data
V1725
VX2740
ADC event
DS channel
Baseline
Filter
Pulse
Finding
DS channel
DS channel
ADC event
DS pulse
Output:
ROOT format
Plot
Histos
TTree

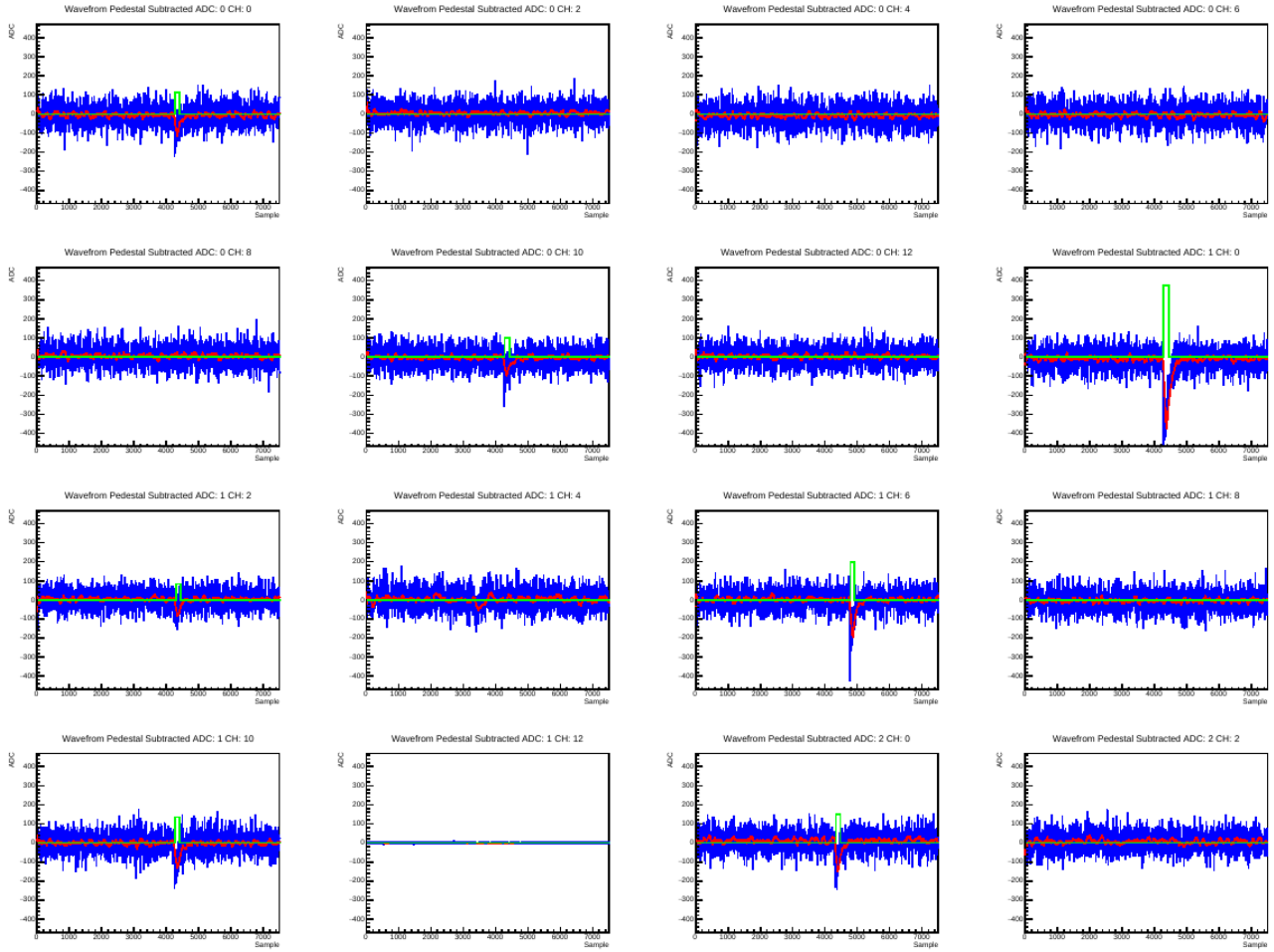
CMake

- Selectively compile the modules
  - `-DV1725=ON/OFF`
  - `-DVX2740=ON/OFF`
  - `-DFilter=ON/OFF`
  - `-DPlot=ON/OFF`
  - `-DHisto=ON/OFF`

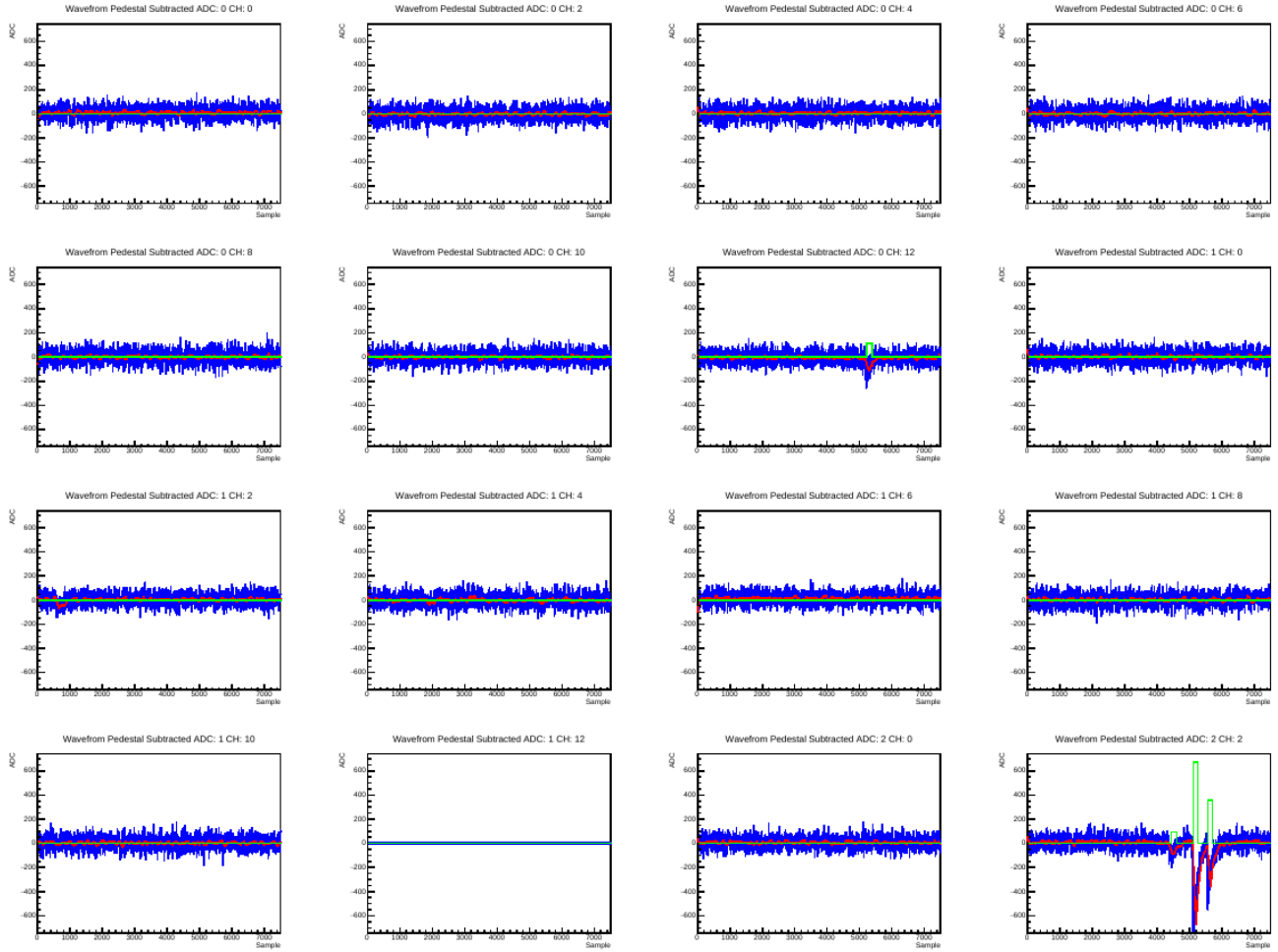
JSON configuration

```
{
    "Global": {
        "Comment" : "online analysis"
    },

    "ADC": {
        "Number of VX2740": 0,
        "Number of V1725": 4
    },

    "Baseline": {
        "ROI":{
            "min": 1200,
            "max": 1500
        },
        "Pedestal":150
    },

    "Filter": {
        "Name":"Moving Average",
        "Par0":100,
        "Par1":false
    },

    "Pulse": {
        "Fixed":false,
        "Sigma":1.5,
        "Amplitude":40,
        "Duration":100,
        "Charge":{
            "before":100,
            "total":400
        }
    },

    "Histo": {
        "Number of Channels":32,
        "Bins baseline":2000,
        "Lower bound baseline":14000,
        "Upper bound baseline":16000,
        "Bins baseline rms":500,
        "Lower bound baseline rms":0,
        "Upper bound baseline rms":500,
        "Bins pulse height":1000,
        "Upper bound pulse height":3000,
        "Bins charge":2000,
        "Upper bound charge":200000,
        "Bins time":2000,
        "Upper bound time":20000,
        "Bins ROI PH":1800,
        "Upper bound ROI PH":18000,
        "Bins ROI Q":50000,
        "Upper bound ROI Q":500000,
        "Bins Rate":6000,
        "Upper bound run time":30000
    },

    "Plot": {
        "Canvas":true,
        "Channel":8,
        "Persistency":false,
        "Persistency Channel":0,
        "Persistency Events Limit":2000,
        "Save PDF":false,
        "Save Events Limit":100,
        "Raw WF Minimum":0,
        "Raw WF Maximum":18000
    }
}
```

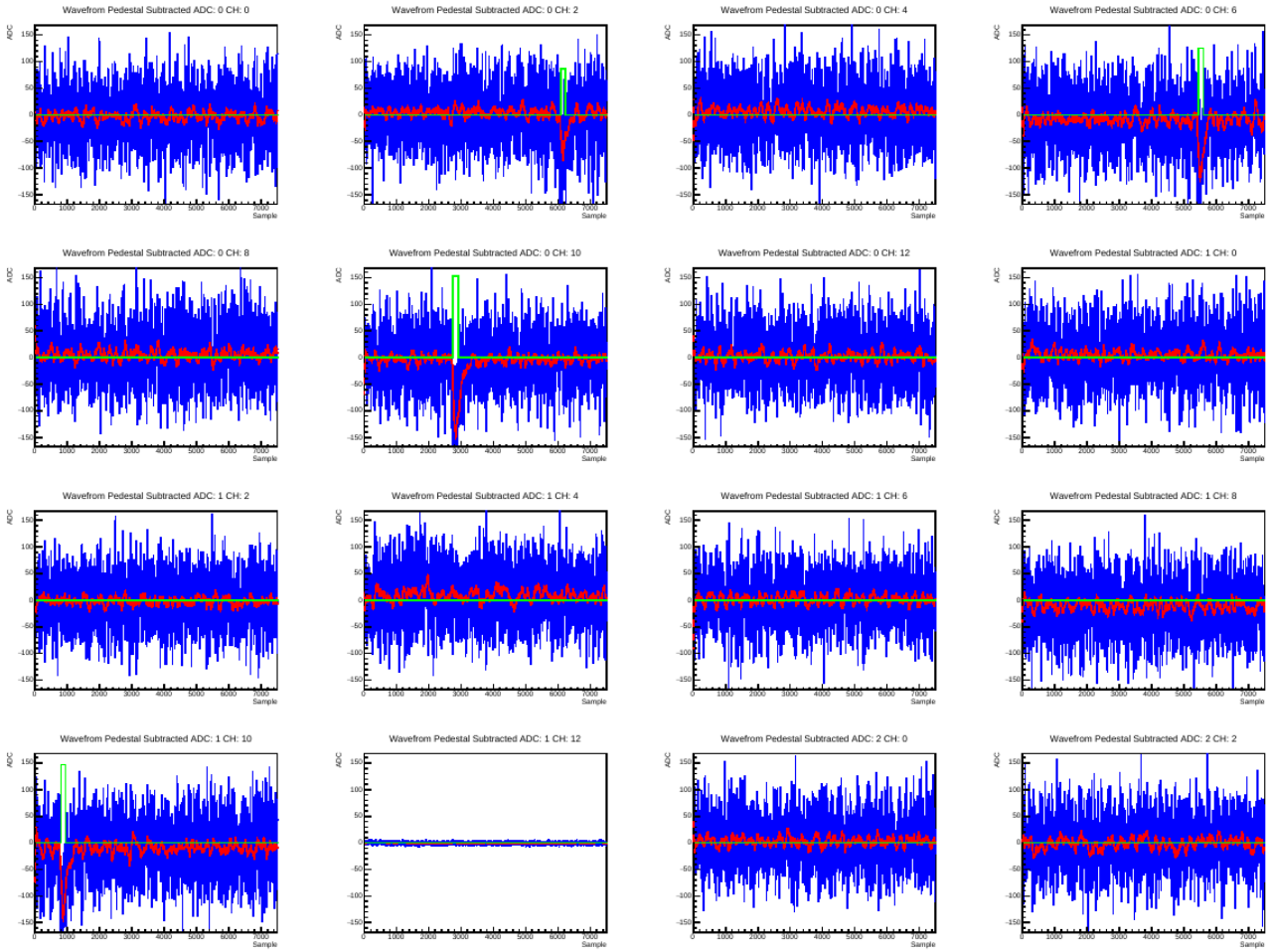
Sample Output



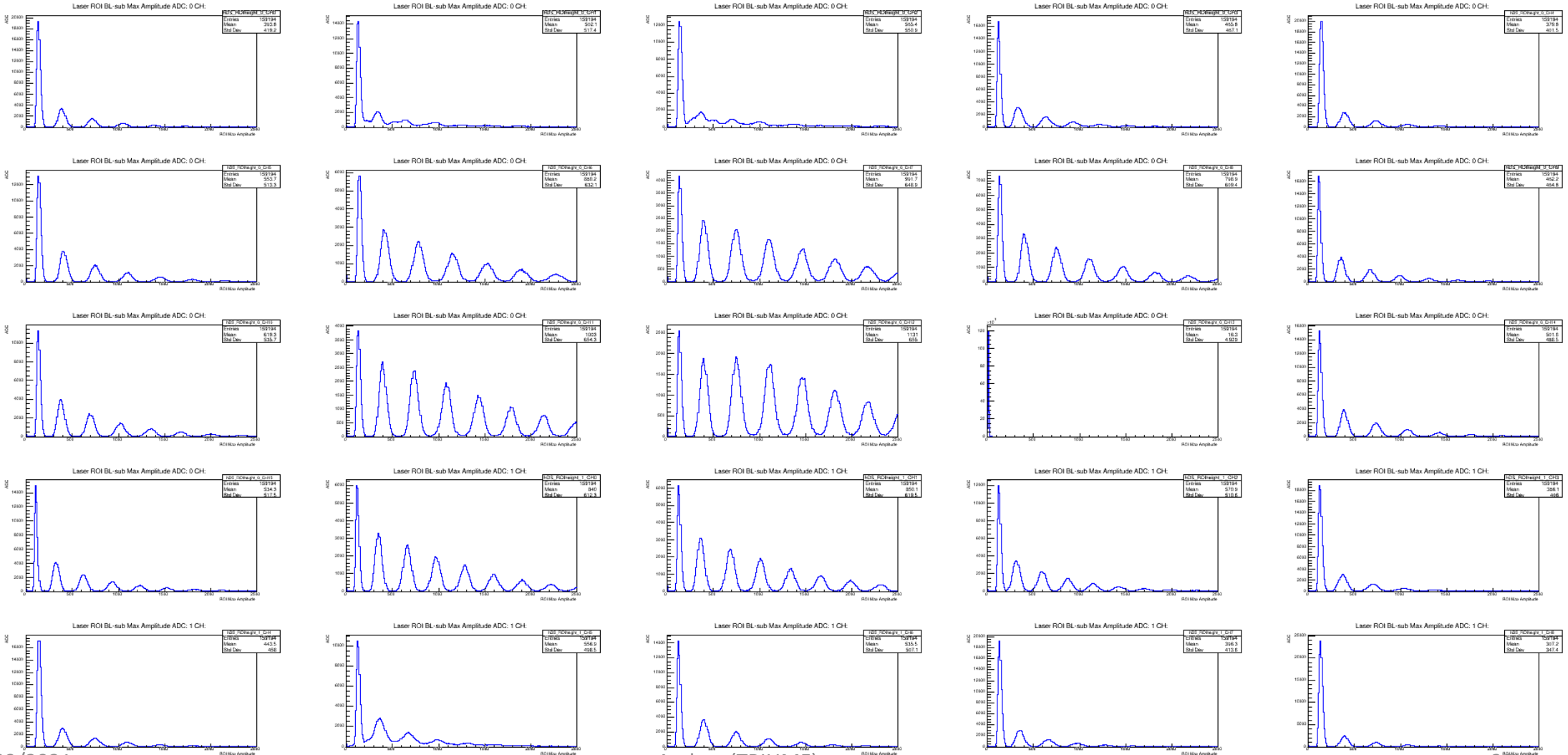
Sample Output



Sample Output



Sample Output



Sample Output

ROOT online server

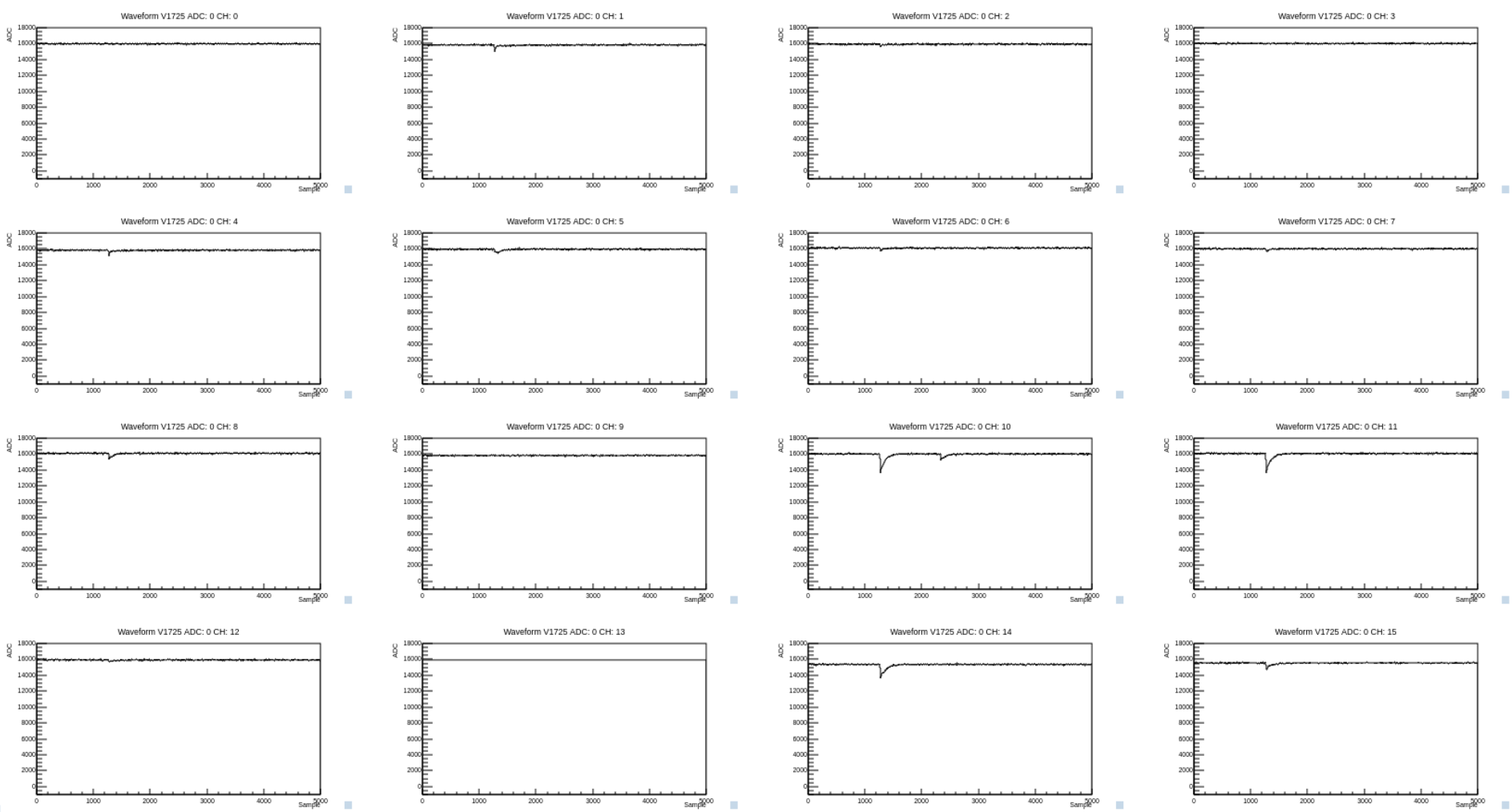
JSROOT version 6.2.0 13/07/2021

Hierarchy in json and xml format

Monitoring simple

open all | close all | reload | clear

- ROOT
  - manalyzer
  - Canvases
    - Plot_ADC_channel8_Run217
    - Plot_ADC_Run217_First16Channels
    - Plot_RawADC_Run217_section1
    - Plot_RawADC_Run217_section2
    - Plot_Q_ROI_Run217_section1
    - Plot_Q_ROI_Run217_section2
    - Plot_PH_ROI_Run217_section1
    - Plot_PH_ROI_Run217_section2
  - Files



git clone https://bitbucket.org/ttriumfdaq/dsadc.git