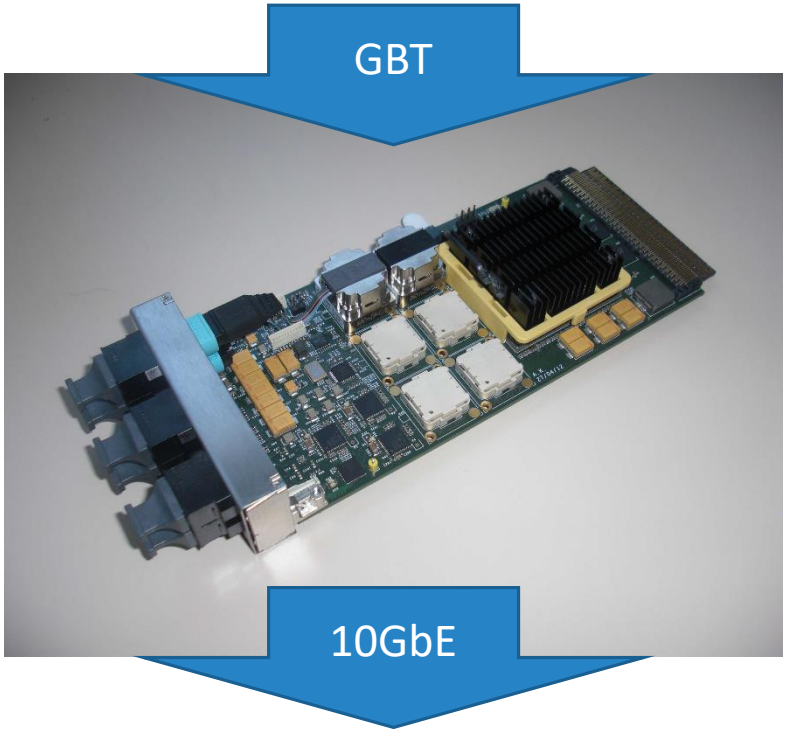
# PCIe40 output interface

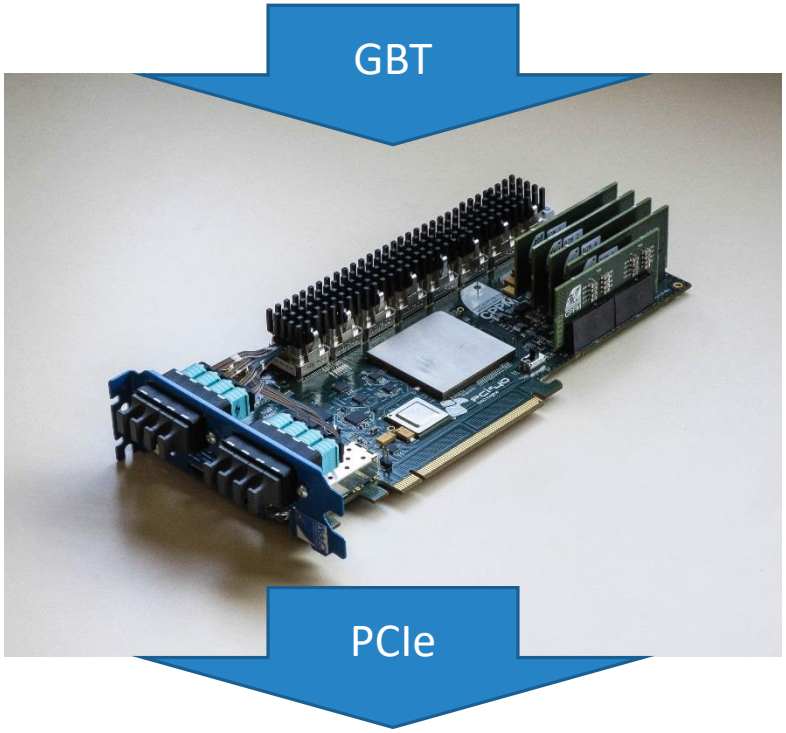
LHCB MINIDAQ2 WORKSHOP

# First of all…

MINIDAQ1 (AMC40)




MINIDAQ2 (PCIE40)



# Outline

Terminology

Firmware side

- Firmware structure and data flow
- Data processing interface and bandwidth
- Multichannel DMA and data streams

Driver side

- Driver structure
- Driver usage

Software side

- Data management
- Performance
- Performance
- Performance…

Reminder: all software releases are tagged by release date and are available here:
http://lbyum.cern.ch/daq40
The list of released tags can be accessed here:
https://gitlab.cern.ch/lhcb-daq40/lhcb-daq40-software/tags

# Terminology

*Interface ID*: unique (within a server) PCIe40 driver identifier
- eg: 2 PCIe40 boards in a server -> 4 interfaces (0, 1, 2, 3)

*Link ID*: local identifier (within a board) of a given interface
- eg: optical links 0..23 -> pcie link 0, optical links 24..47 -> pcie link 1

*Interface address*: topological address (bus/device/function) of an interface
- eg: 05:00.0 Communication controller: CERN/ECP/EDU Device ce40 (rev 01)

*Event ID*: unique identifier (within an entire experiment/run) of a collision

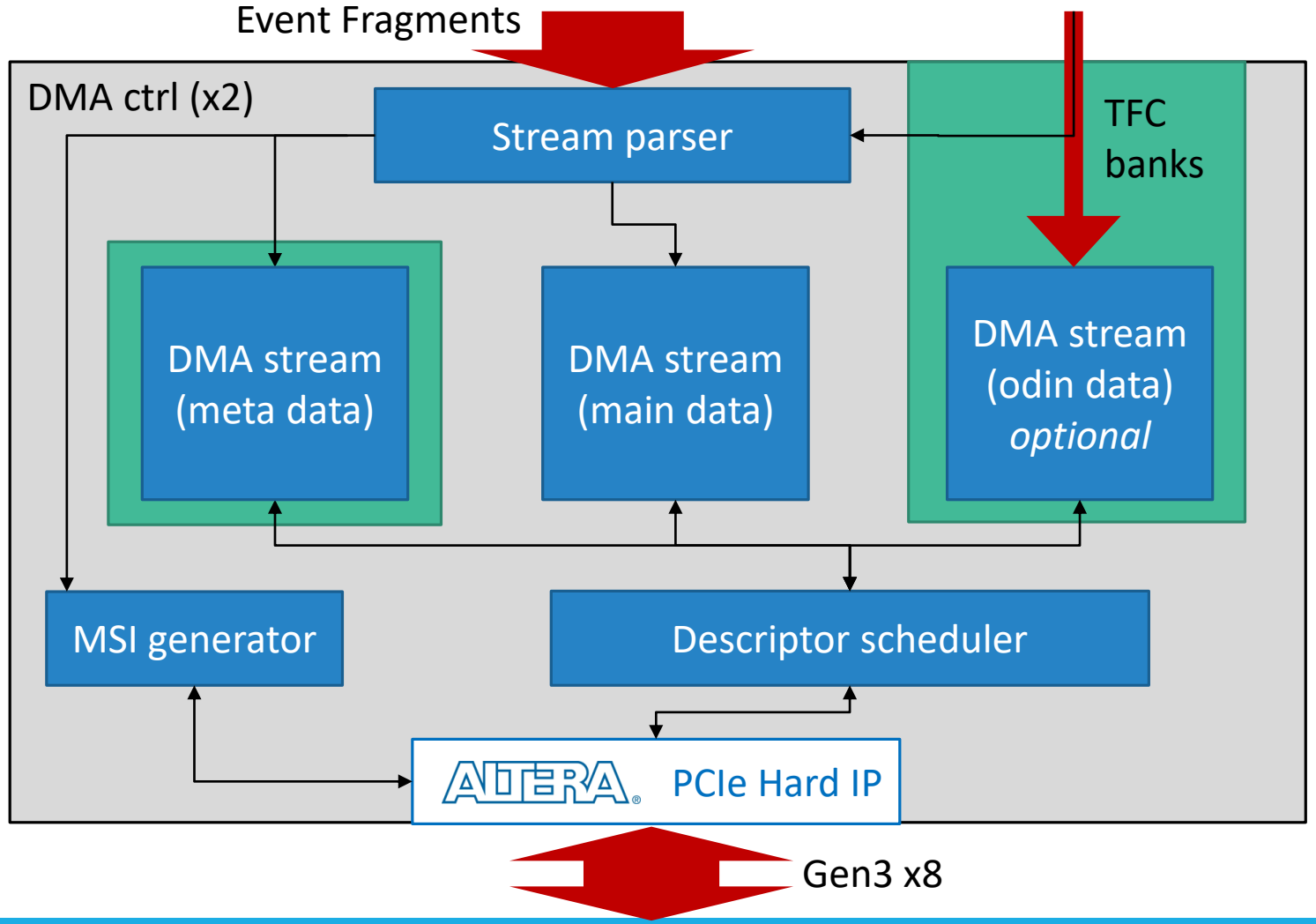
*Fragment*: the output data from a single TELL40 instance for a given Event ID
- eg: 1 PCIe40 board -> 2 fragments per Event ID

*Event*: the output data from the full event builder for a given Event ID

*Host*: the server housing a given PCIe40 board

*Stream*: an independent, simplex (FPGA->HOST) data communication channel

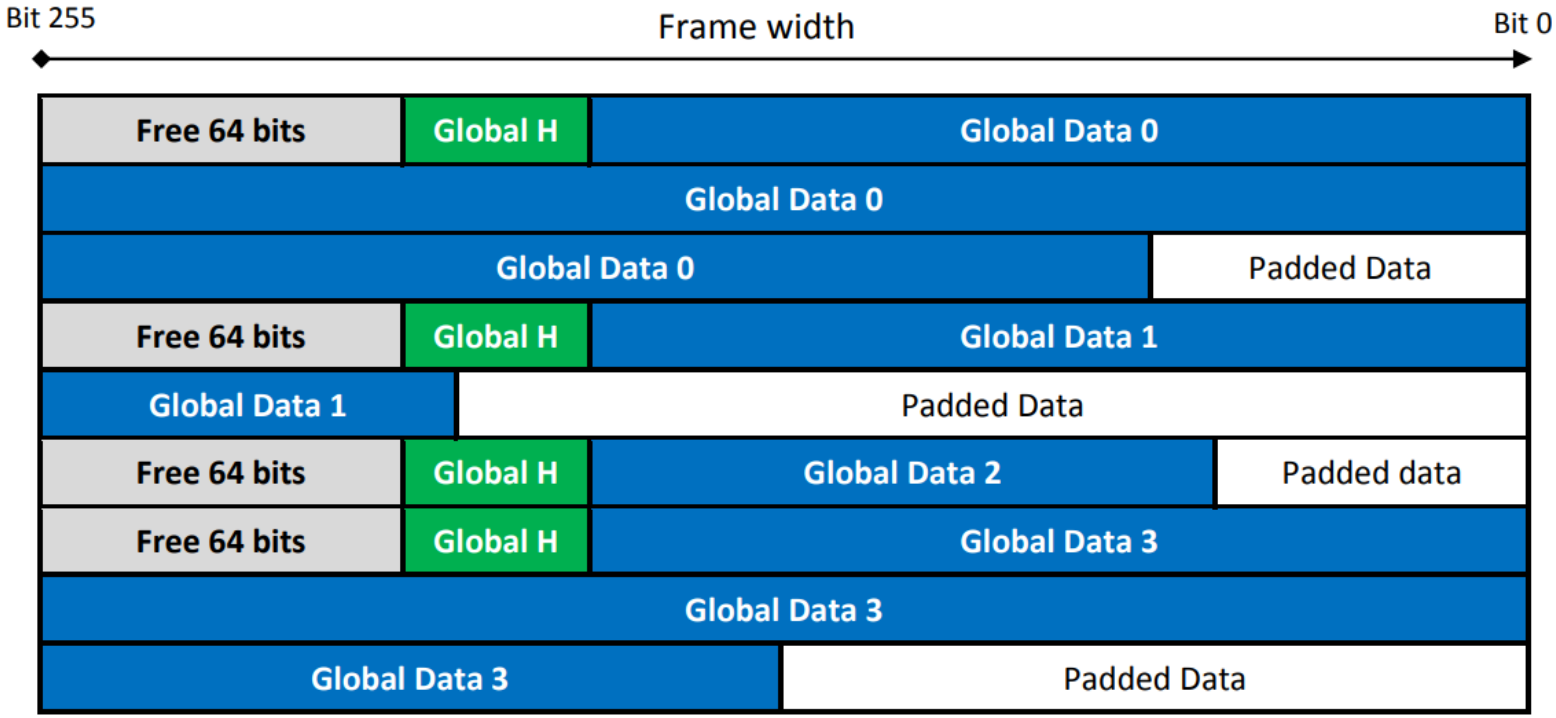
# Firmware dataflow

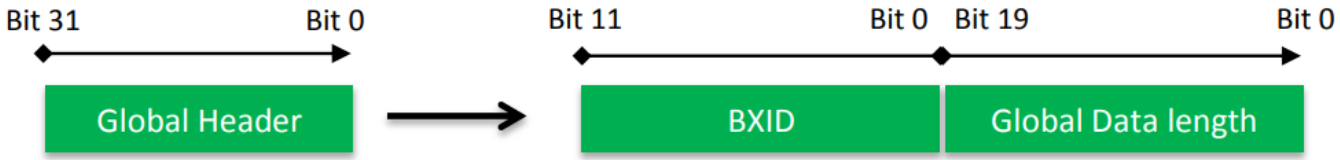



250 MHz
40 MHz

$$\frac{main}{meta} = \frac{\sim 100}{1}$$

# Data processing interface



# Data streams

Generic FPGA->PC communication mechanism implemented for the LHCb upgrade

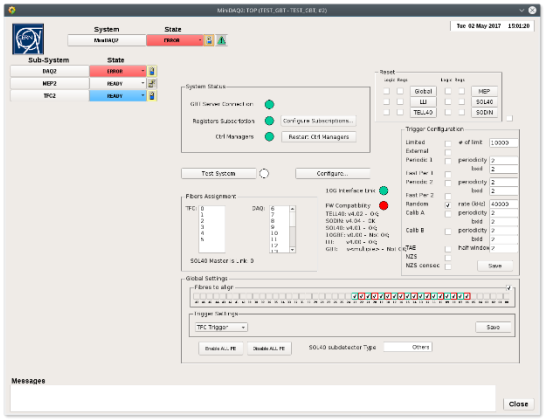
- FPGA: custom DMA controller
- Host: custom Linux kernel driver and userspace software

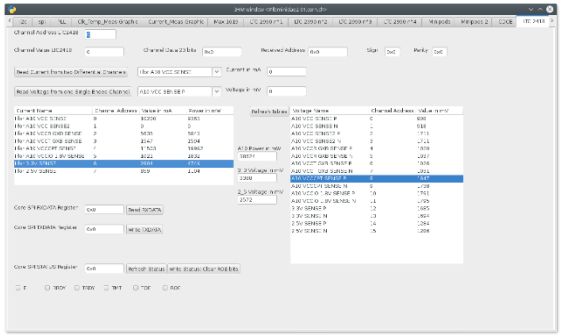
Three stream types in current design, more could be added if you have ideas:

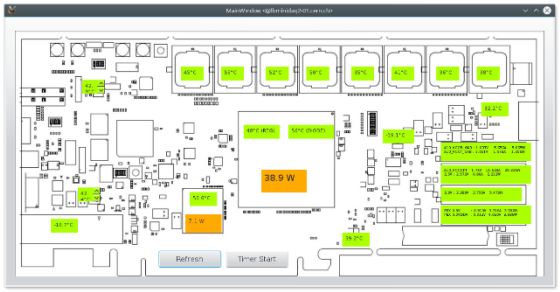
- MAIN data stream
  - Fully implemented (firmware and software)
  - Transports all the frontend data aggregated by the TELL40 firmware
- META data stream
  - Implemented in firmware, not fully integrated in current software
  - Transports “blocks” of event metadata into the network event builder
- ODIN data stream
  - Not yet instantiated, will be used to transport ODIN banks out of the SODIN boards

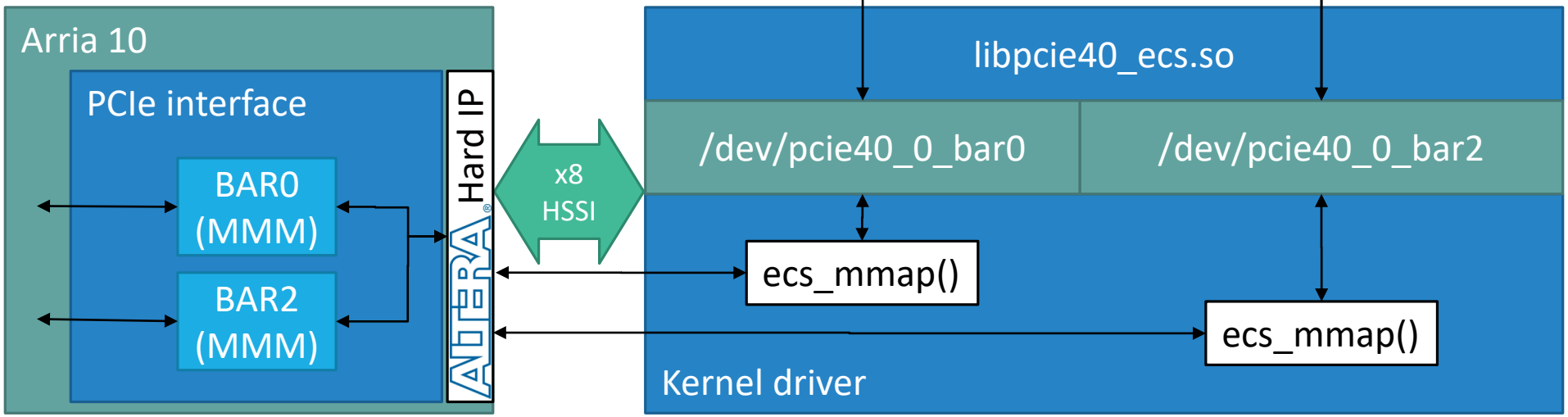
Today we will be using the first stream only

# ECS kernel driver







# Demo time!

Read / write registers (eg: 0x700004) from the command line…

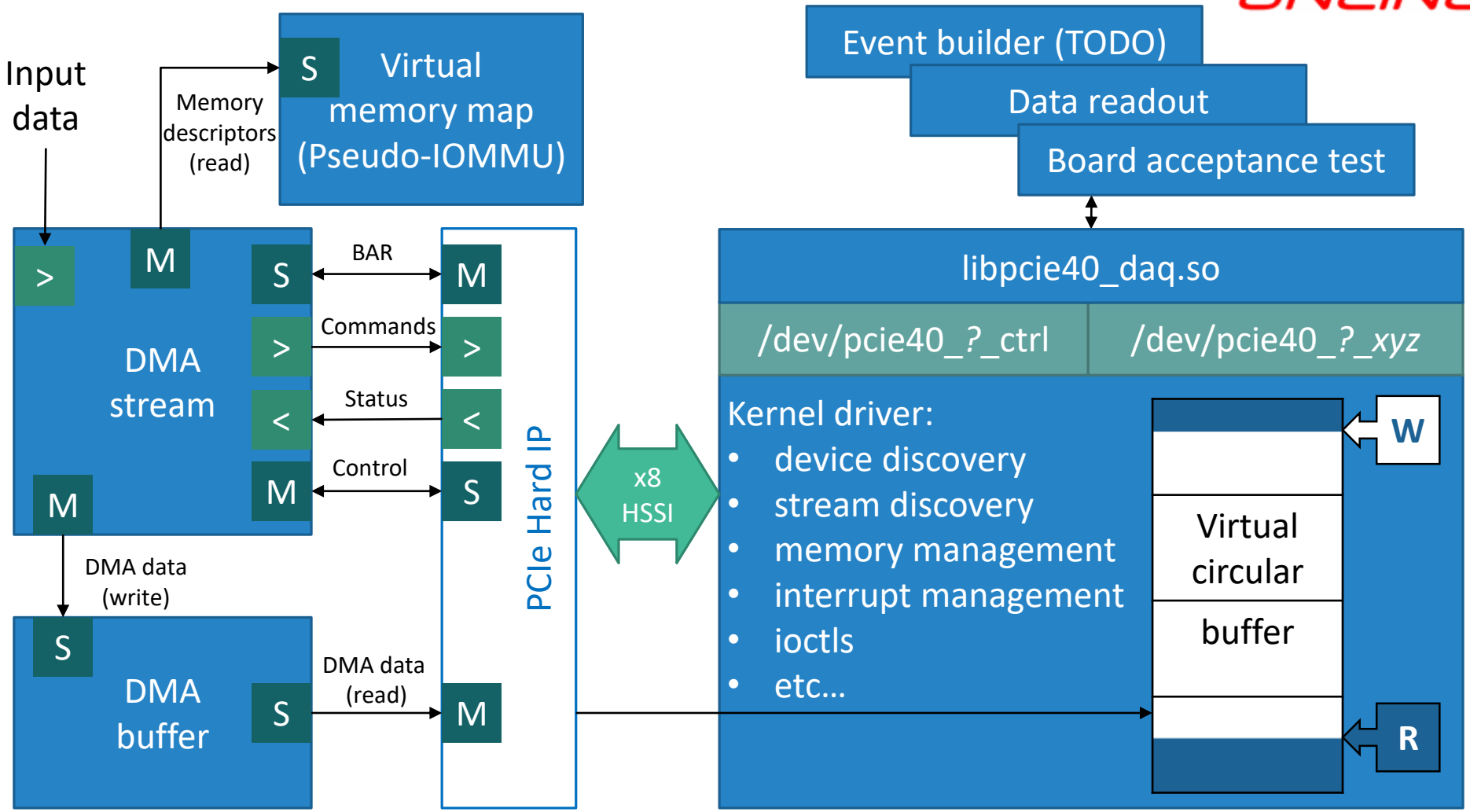
```
pcie40_ecs –b {BAR} –a {ADDRESS} –r
pcie40_ecs –b {BAR} –a {ADDRESS} –w {VALUE}
```

(reminder: BAR 0 = user code (you!), BAR 1 = hidden, BAR 2 = low level interface)

…or using the library:

```
#include <lhcb/pcie40/ecs.h>
-lpcie40_ecs
p40_ecs_open(interface_id, bar_number, handle)
p40_ecs_w32(handle, address, value)
p40_ecs_r32(handle, address)
```

# DMA kernel driver



# Demo time!

Locate the device files
- ls /dev/pcie40_*

Understand the reload procedure
- pcie40_reload

Understand the output of the low-level DAQ command line
- pcie40_daq

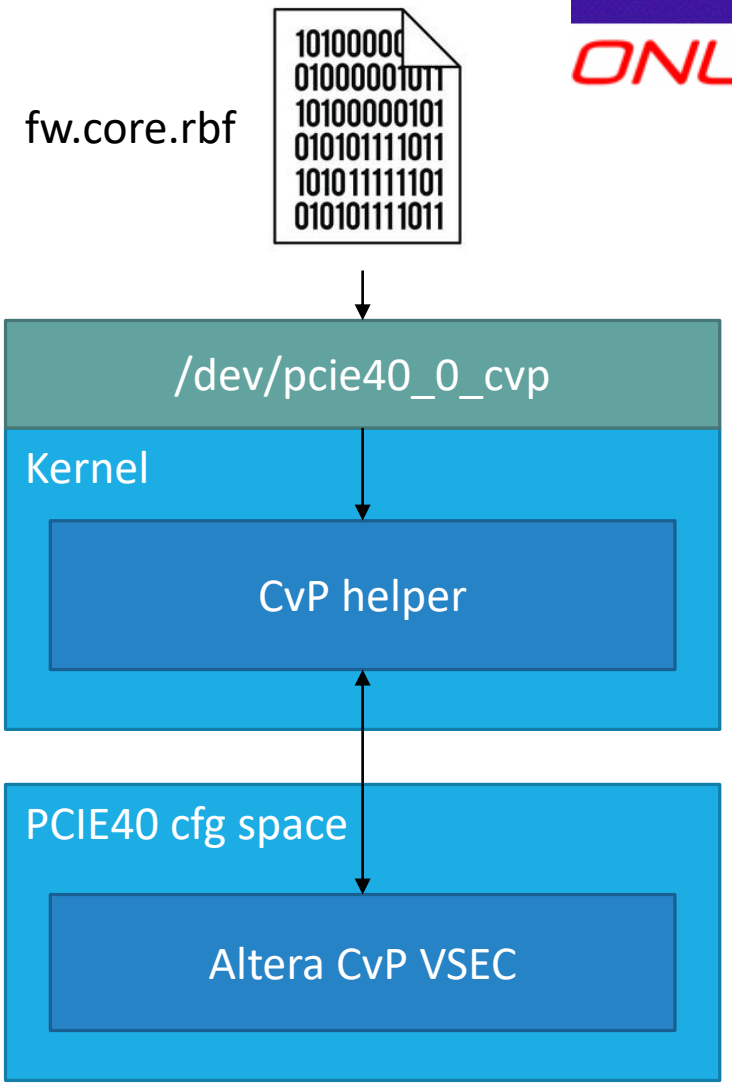
Run a data integrity test
- pcie40_daq -vRFegc

Run a performance test
- pcie40_daq –vRegO

Understand how to reset an interface
- pcie40_daq –rt
- pcie40_daq –Rt

# CVP kernel driver

- Allows reprogramming the FPGA over PCIe bus
- Based on Altera reference implementation
- Will be necessary to upgrade firmware “in the field” when we scale out
- Partial success on Nallatech FPGA
- Future work: create flash bootstrap image for PCIe40 and integrate support in driver



# Demo time!

Not yet, the CvP driver is still in development…

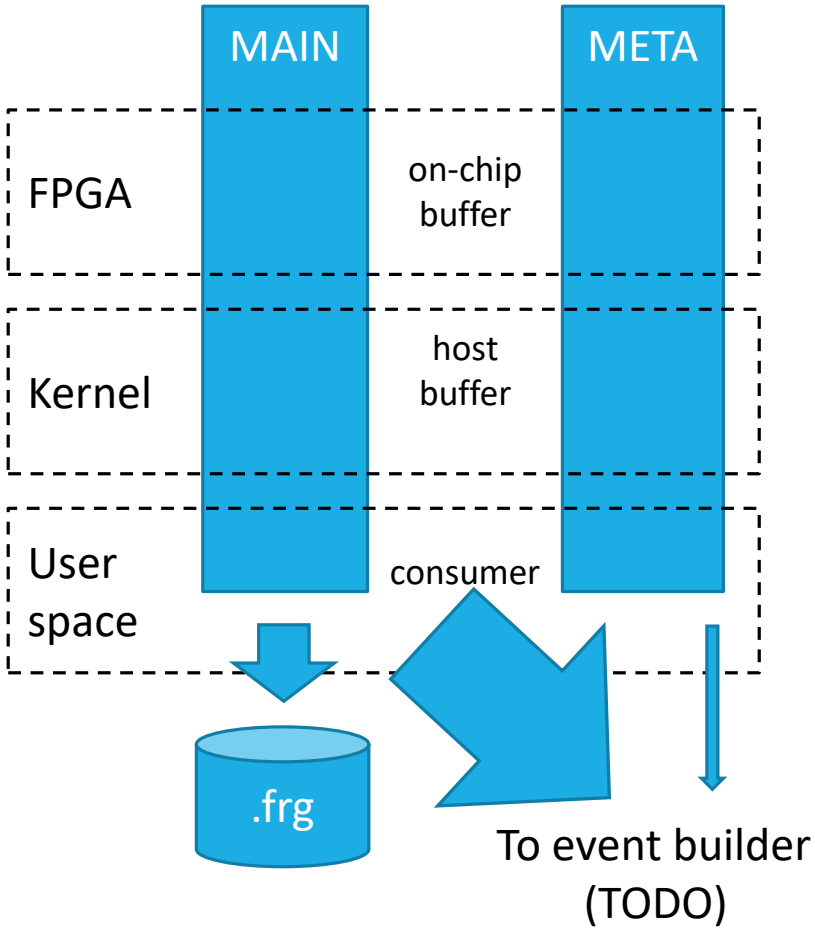
However, time for a few words about driver updates:

- Driver releases are managed in parallel with DAQ40 software releases
  - eg: lhcb-pcie40-tools-**20170801** is always released together with lhcb-pcie40-driver-**20170801**
- There are a "package version" (↑), a "core version" (FPGA only, you don't care) and a "register map version" (FPGA & Driver)
- PCIe firmware and driver are versioned together
  - **register map version compatibility is checked at runtime**
- Linux drivers are packaged with DKMS, this means:
  - Kernel upgrades/downgrades are transparent
  - Driver upgrades/downgrades are (mostly) transparent
  - yum erase lhcb-pcie40-driver; yum install lhcb-pcie40-driver-**YYYYMMDD**
- Driver releases in the *stable* channel track the *master* firmware branch

# MiniDAQ2 readout

- Similar philosophy between MiniDAQ1 and MiniDAQ2
- Common output file format (.frg)
  - Contains fragments, not events!
  - It's not .mdf, but serves similar purpose
  - Comes with a C++ API
- MiniDAQ2 status
  - Stable firmware interface
  - Kernel drivers
  - Command line tools
  - Storage layer
  - DIM interface
  - WinCC integration
- TODO
  - Emulation driver
  - Event builder integration (longer term)

DAQ streams



# Demo time!

Capture some data using only the command line, check the FSM transitions

- pcie40_frgwriter –D

Check the data file:

- pcie40_frgreader –V file.frg

Examine the data file contents:

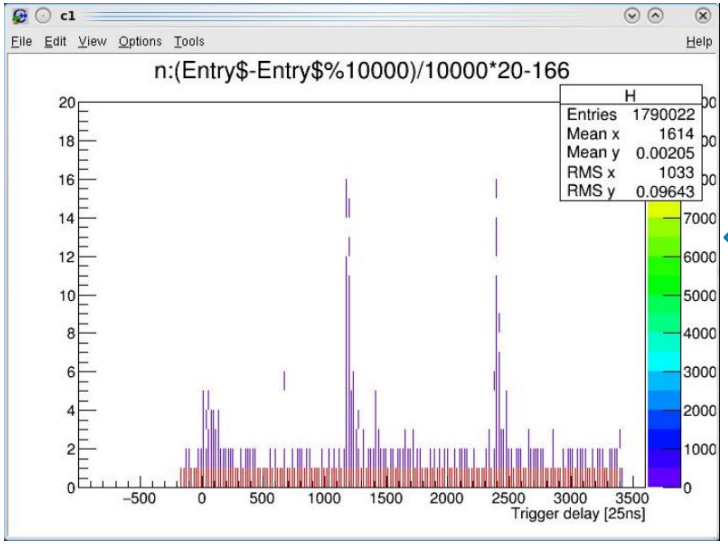
- pcie40_frgreader file.frg | less

Do the same using the C++ library:

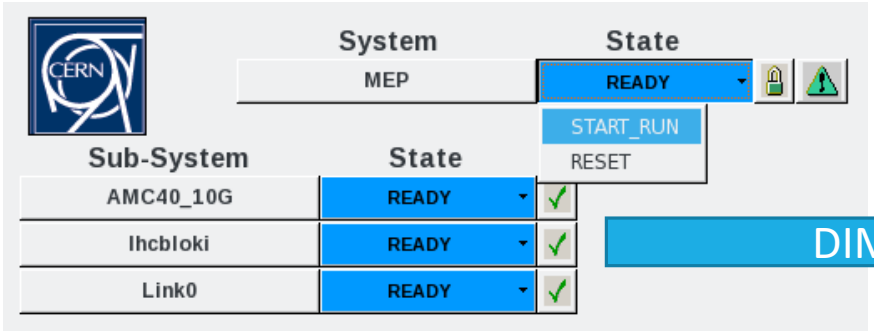
- lhcb-daq40-software/daq40_frgreader/example

Note: {pcie40,daq40,amc40}_frgreader are the *same* program

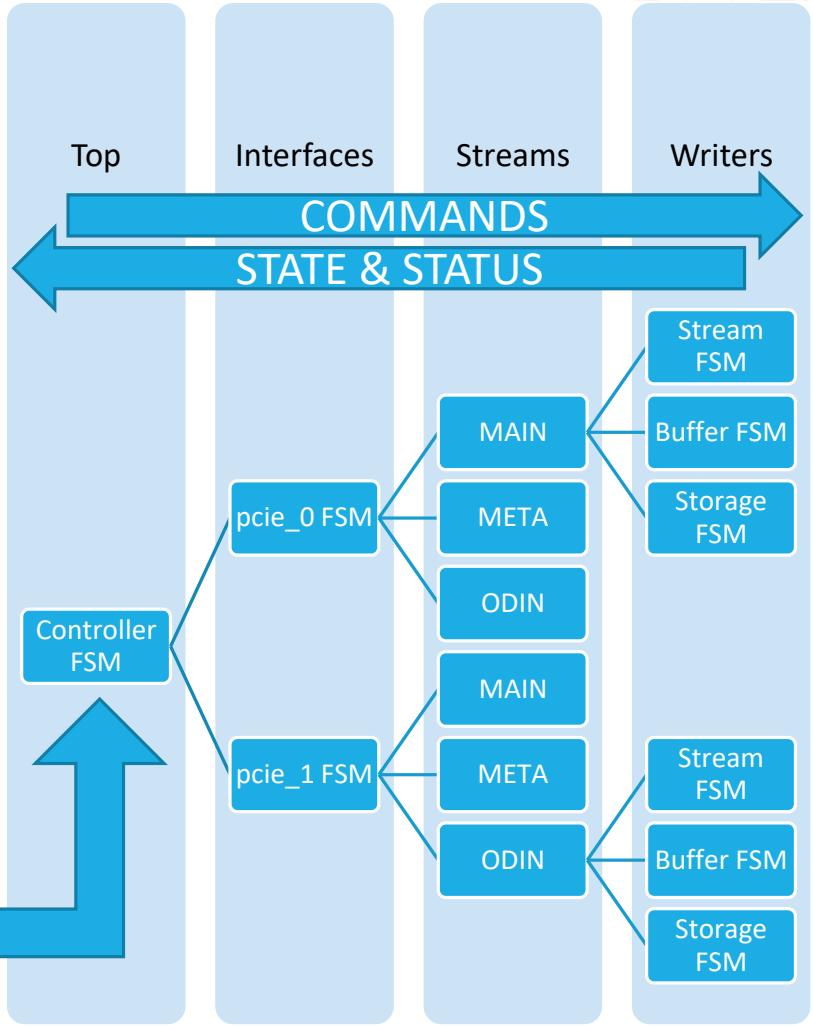
# PCIe DIM interface




disk




DIM



# Demo time!

Use the PCIe WinCC panel to understand PCIe40 readout performance:

Observe buffer occupancy in performance mode

Observe buffer occupancy in data integrity mode

Observe buffer occupancy while going through the DAQ software

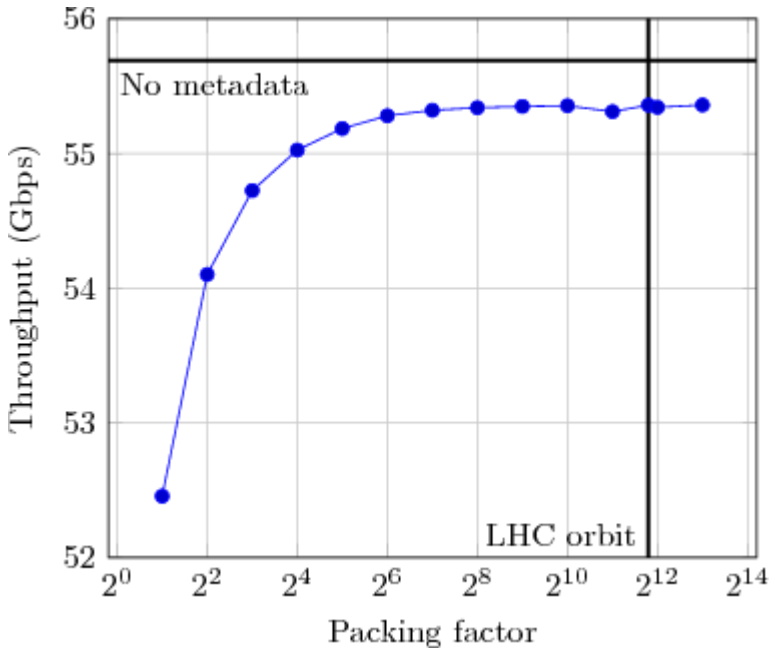
Metadata acceleration to the rescue!

# Metadata acceleration

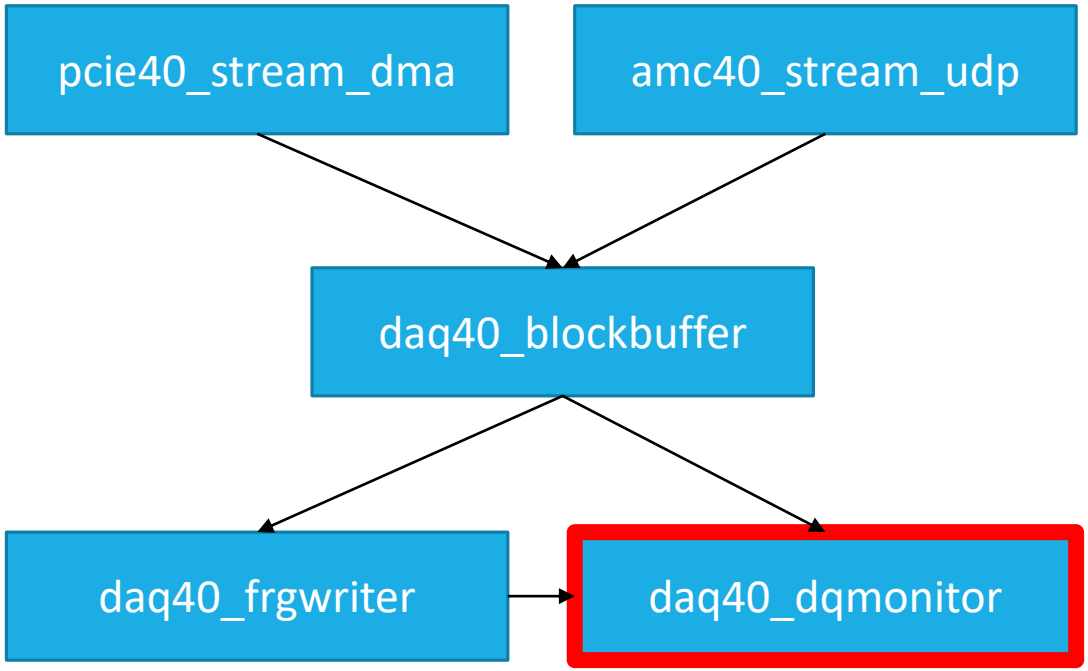
We see the software spends a lot of CPU and memory bandwidth to parse the fragment sequence and split it into “blocks”.

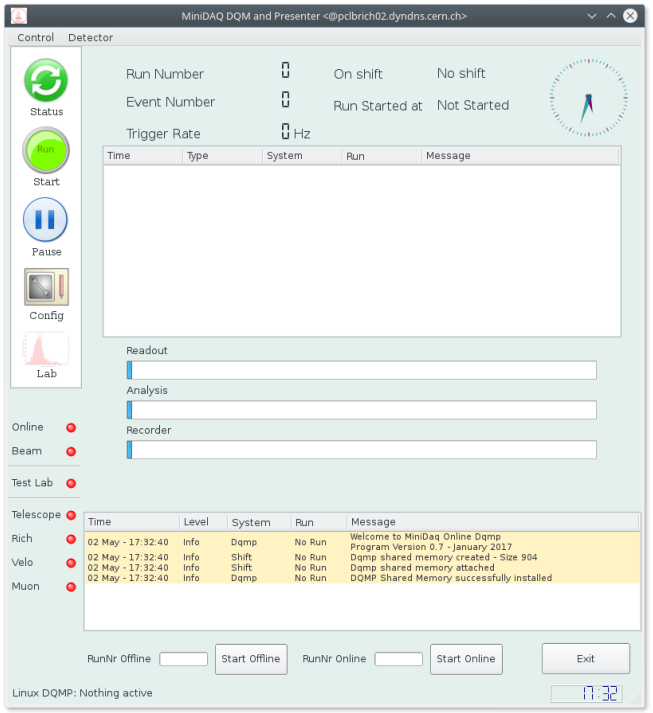
Using metadata acceleration, all of this work is done in the FPGA, and the ready “blocks” are sent to the event builder using a dedicated stream.

This function is already implemented in the firmware, but not yet fully in the software, will be necessary in order to run the event builder efficiently.



# DQMP integration



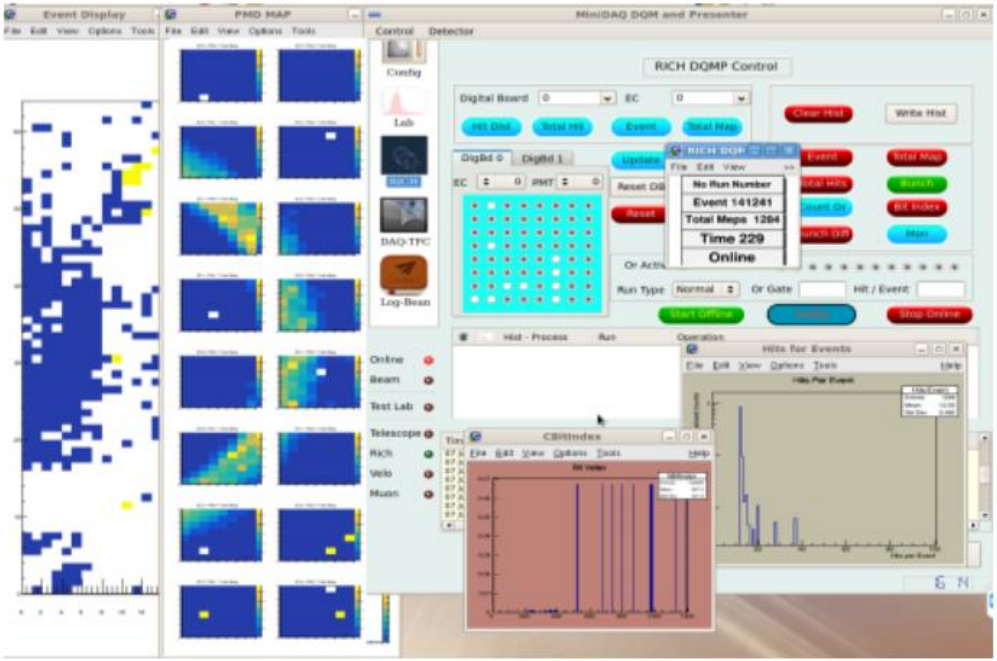

- So far used successfully by RICH test beam with MiniDAQ1

SHMEM IPC

# DQMP at work (F. Cindolo)

## Online monitoring: presenter

- Online team provided real time data quality presenter to monitor data based on Qt-FW and shared memory processes, thanks!




Roberta Cardinale on behalf of RICH TestBeam group Tuesday meeting, 11/07/2017 11

# Questions? Comments?

THANK YOU