

# Implementation of the free running format to the DAQ software



Martin Zemko

COMPASS Front-end, Trigger and DAQ Workshop

3rd March 2020



**CTU**

CZECH TECHNICAL  
UNIVERSITY  
IN PRAGUE



# Overview

- Description of the free running format
- Design of the trigger framework
- High level trigger implementation
- Integration with the DAQ system
- Additional tools

# Description of the new DAQ format

- New data format includes a multi-layer structure of data
- Each layer has its purpose and encapsulates the lower layer (its children)

Epoch	• Encapsulates all the slices in a given time period
Slice	• Wraps all the images in a given time period
Image	• Wraps all the groups and hits with the same sampling frequency
Group	• Contains hits coming from one Source ID (can be nested in itself)
Data	• Carries information about a single hit
Additional Data	• Contains extended information about associated hit

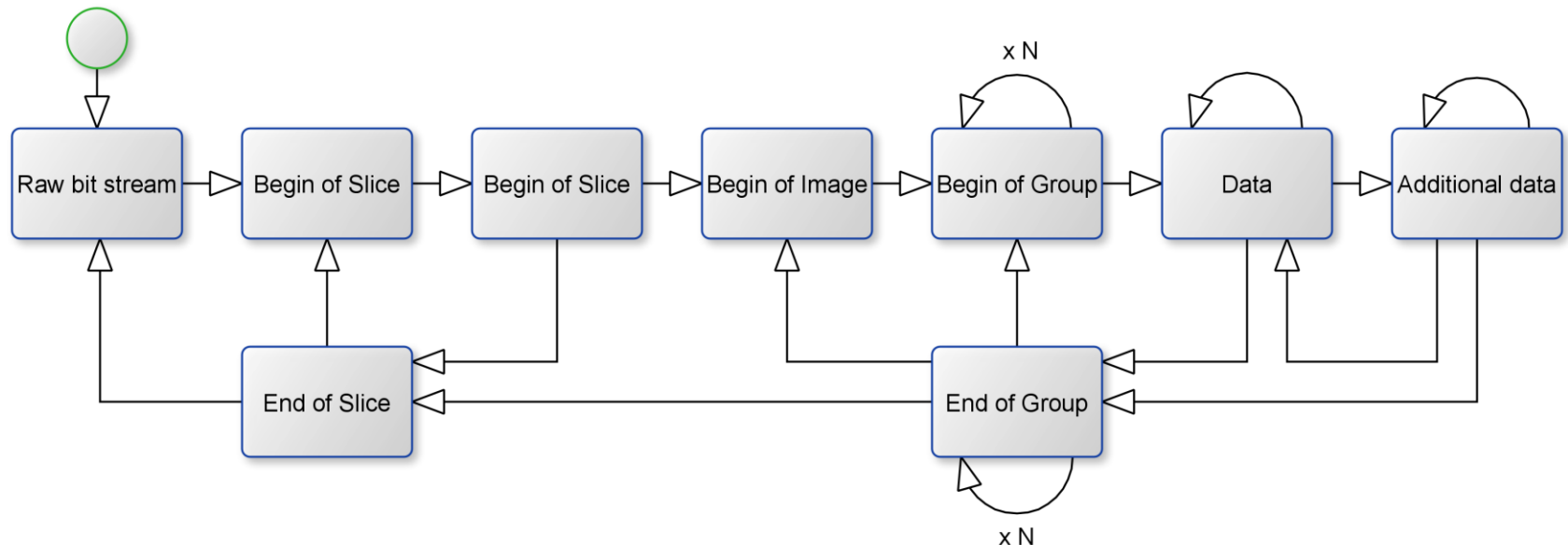
# Structure of the free running format

- Data are aligned in atomic packages of the 32-bit size called „Data words“
- Every data word contains a control word indicating the type of data

Type of data word	Information included
Begin of Slice I	Number of Images in Slice, Slice number in Epoch
Begin of Slice II	Start time of Slice in Epoch
Begin of Image	Start time of Image in Epoch
Begin of Group	Source ID, View ID, First Hit time in Group
Data Word	Frontend ID, Channel ID, Hit time
Additional data	Any kind of data (29 bits)
End of Group	CRC checksum, 3 x bit flag
End of Image	Currently not used
End of Slice	CRC checksum, 3 x bit flag

# Structure of the free running format

- State machine for validation of raw data streams
- Sequence of data words must follow the paths indicated below
- If any data stream has a different sequence, data are corrupted



# Requirements for the trigger framework

## Functional requirements

- Support of various triggering algorithms and triggering patterns (triggering rules)
- Adjustable trigger sensitivity
- Identification of regions of interest based on hit correlations
- Generating event candidates (list of images that comply with the triggering rules)
- Emulation of the hardware trigger
- Output for the CORAL (CSDigits)

## Technical requirements

- Stand-alone framework – as few dependencies as possible
- Distributed computing for high performance
- Support for multithreading and NUMA balancing
- Reviewed code base (changes committed via pull requests)

# Interface of the trigger framework

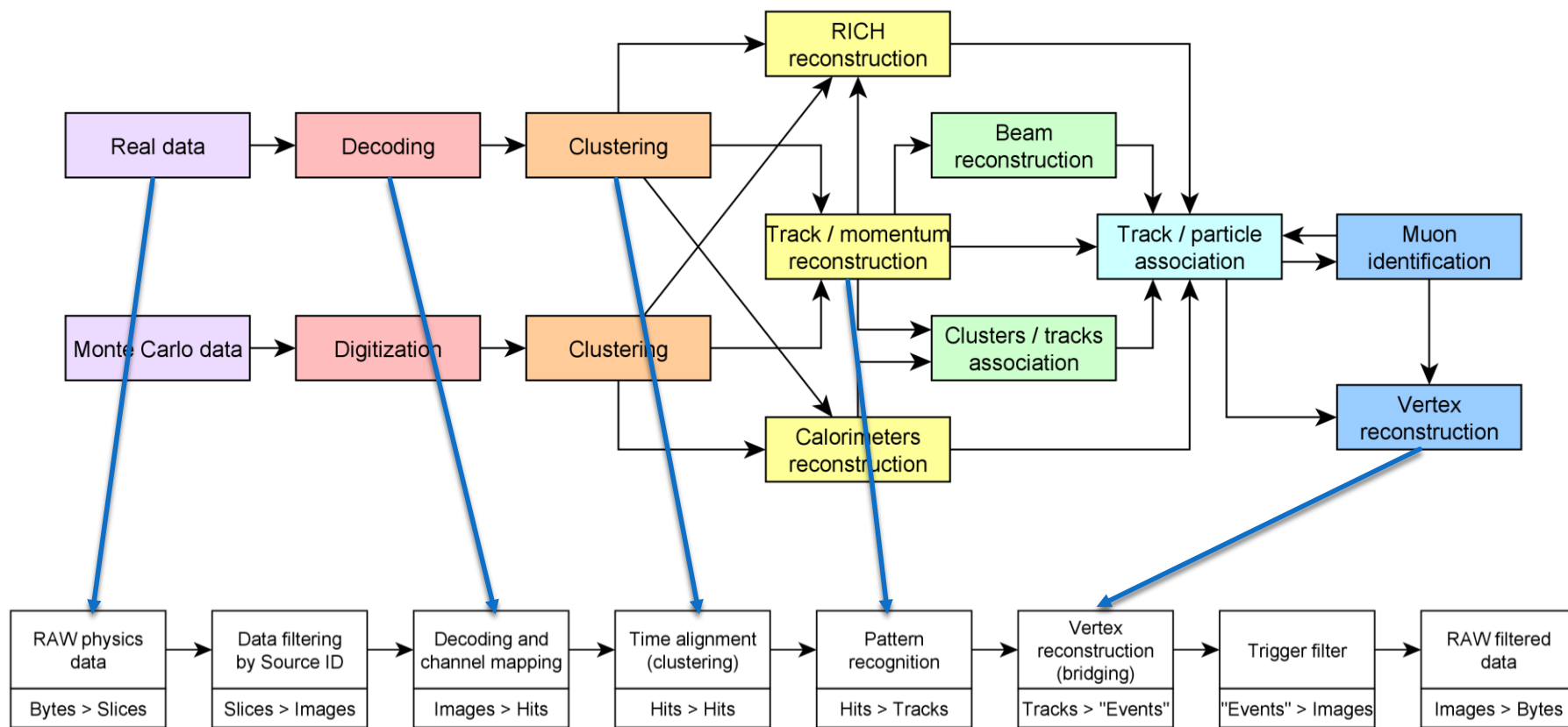
## Inputs

- RAW data files from the DAQ
- Detector response simulations
- Detector descriptions (detectors.dat)
- Mapping files
- Triggering options

## Outputs

- Event candidates (filtered RAW files)
- CORAL files (CSDigits)
- Monitoring information for monitoring tools

# Reconstruction chain

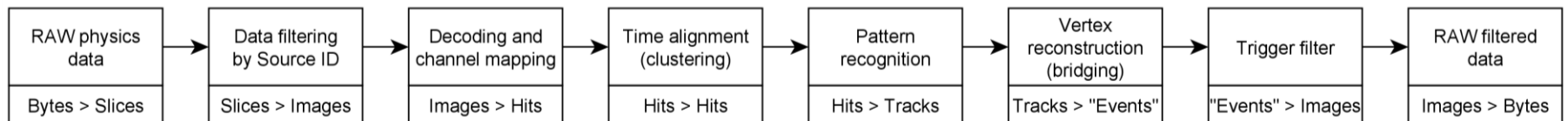




# Reconstruction chain

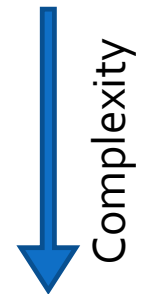
Steps that have to be taken in the reconstruction procedure:

1. Reading RAW data – convert byte stream into slices
2. Data filtering by Source ID – extract trigger information from the slices
3. Decoding and channel mapping – change the transfer mapping into the physical mapping
4. Time alignment – find correlations between hits and justify hits in time dimension
5. Pattern recognition – convert hits into partial tracks (se)
6. Vertex reconstruction – join projections into the full particle tracks
7. Filtering – data containing event candidates are passed through

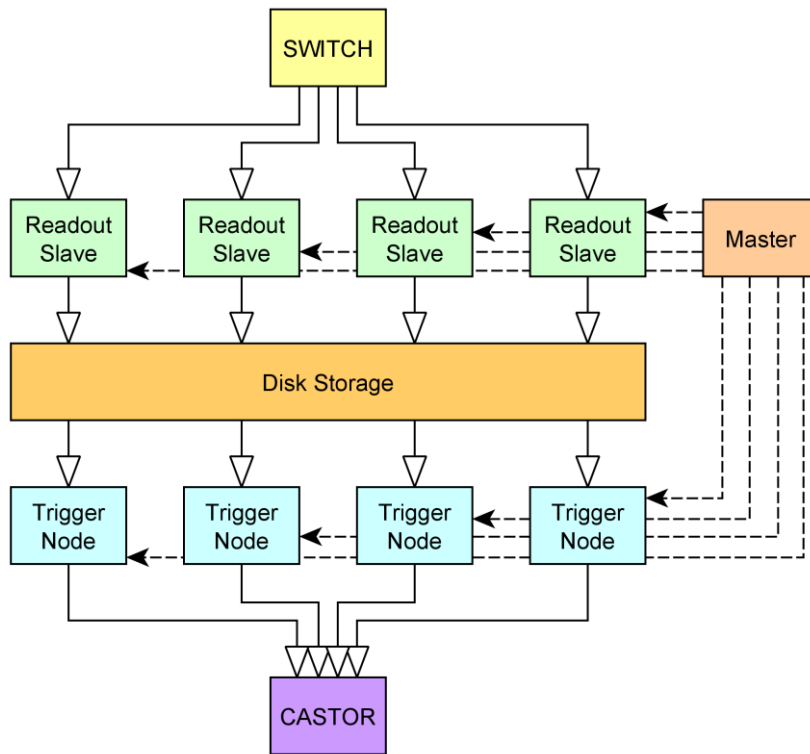


# Possible integration scenarios

- Scenario 1 : Common master
- Scenario 2 : Independent trigger supervisor
- Scenario 3 : Common master interfacing through the database
- Scenario 4 : Independent trigger supervisor using the database

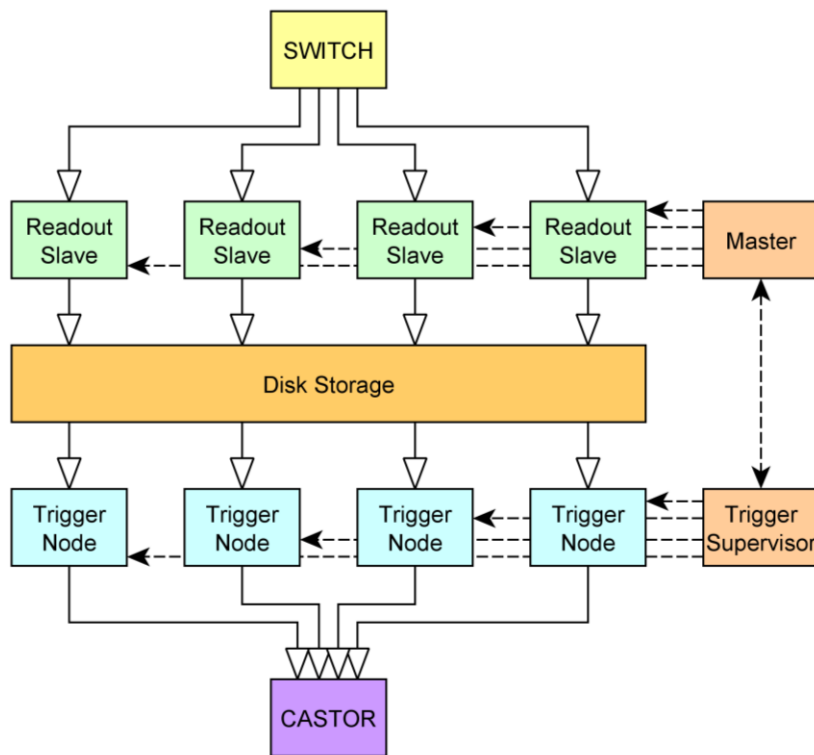


# Scenario 1 : Common master



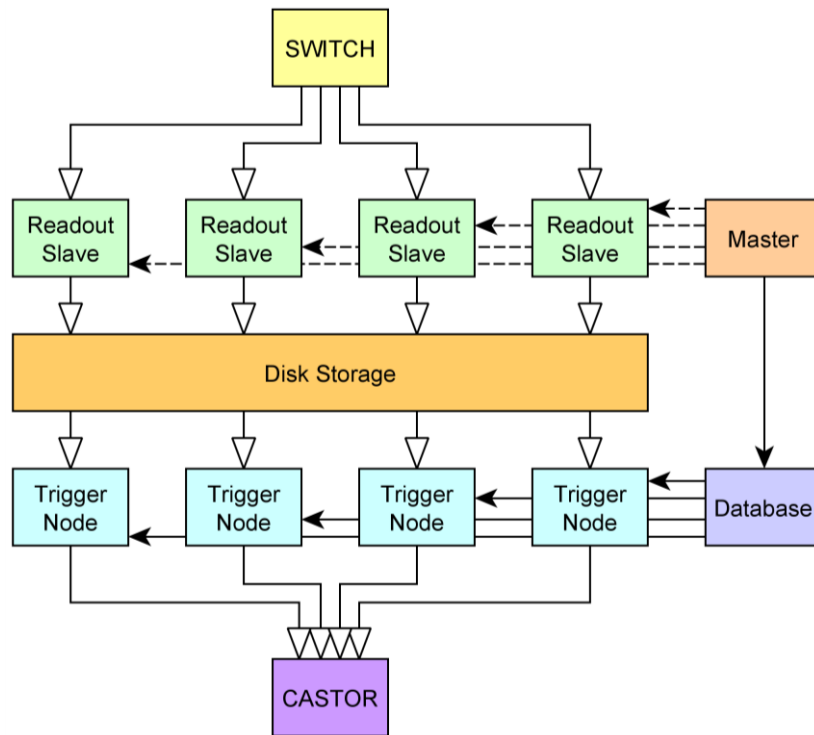
- After successful readout, the Master notifies trigger nodes about RAW data waiting for the triggering
- Trigger Nodes directly fetch RAW data from the Disk storage and initiate the triggering procedure
- Notification about successful completion is sent back to the Master

## Scenario 2 : Independent trigger supervisor



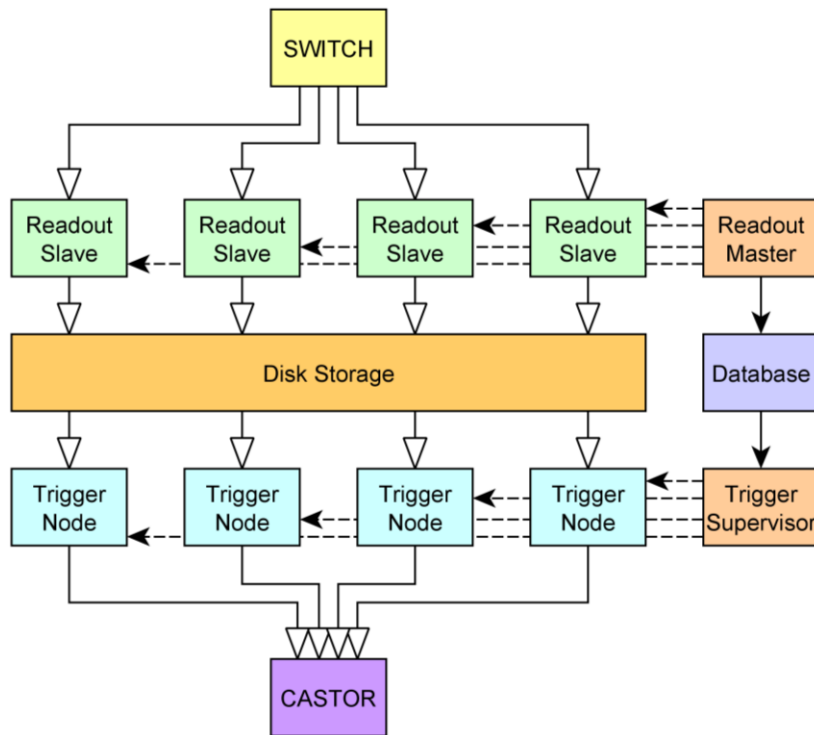
- After successful readout, the Master sends meta information about RAW files to the Trigger Supervisor
- Trigger Supervisor distributes this information to Trigger Nodes
- Trigger Nodes fetch RAW data directly from the Disk storage and initiates the triggering process

# Scenario 3 : Common master interfacing through the database



- Master saves meta information about RAW files to the database
- Trigger Nodes regularly check the database for new data
- Trigger Node directly pull RAW data from the Disk storage and immediately initiates the triggering procedure

## Scenario 4 : Independent trigger supervisor using the database



- Master saves meta information about RAW files to the database
- Trigger Supervisor accesses the database checking for new data and notifies Trigger Nodes about new data
- Data are directly transferred from the Disk storage to Trigger Nodes
- Trigger Supervisor initiates the triggering procedure on Trigger Nodes

# Common features for all scenarios

- At first, RAW data are stored on the disk storage
- Then, they are directly transferred to the trigger nodes and „filtered“
- Filtered data are sent to the CASTOR storage
  
- The DIALOG interface is used for communication between Master <—> Slaves and Supervisor <—> Nodes
- Access to the database is always direct, i.e. using the SQL client

# High level trigger dependencies

- Qt framework – the base framework of the DAQ
- DAQ structure file – taken from the RCCARS, must be converted into the library
- DIALOG – library providing communication with the DAQ
- ROOT – may be needed for advanced calculations in some modules



# Modules of the trigger framework

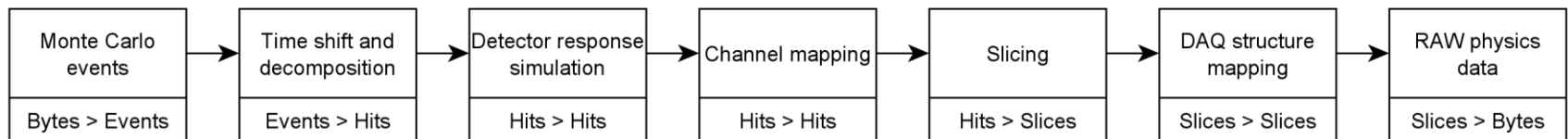
- HLT Node process – main triggering software running on Trigger Nodes
- HLT Supervisor process – manages Trigger Nodes and distributes the load
- **Data Generator** – tool producing artificial data in the new format
- CLI Data Generator – command line interface to the Data Generator
- **Data Browser** – tool for browsing chunks of data
- CLI Data Browser – command line version of the Data Browser
- Monitoring tools – not yet defined
- Configuration tools – not yet defined

## Other tasks related to the new DAQ format

- Design and implementation of trigger configuration tools
- Design and implementation of trigger monitoring tools
- Modification of the Slave Readout process (work in progress)
- Modification of user interfaces, e.g. GUI, Logbook, etc.
- Adaptation of monitoring tools, e.g. COOOL, MurphyTV, etc.
- New database design
- Installation of new hardware

# Data generator

- Data Generator is already implemented and **partially** operational
- Utilizes a straight pipelined structure (see picture below)
- The only missing part is the Detector Response Simulation, which is also the most important part
- It must be created in a cooperation with detector experts (hits -> signals)
- This part will be exploited also in the triggering procedure
- Inversed relation is required also for triggering (signals -> hits)



# Data generator GUI

The screenshot displays the 'RAW file generator' application window. It is divided into two main sections: 'General properties' and 'Slice properties'. The 'General properties' section includes fields for 'Configuration file', 'Output file', and 'Data format'. The 'Slice properties' section includes fields for 'Slice duration', 'Image duration', 'Sample size', 'Flux value', 'Mapping file', 'Detector file', and 'Sample event file'. At the bottom, there are buttons for 'Start', 'Load default values', and 'Exit'.

**General properties**

- Configuration file:
- Output file:
- Data format:

**Slice properties**

- Slice duration:  ns
- Image duration:  ns
- Sample size:  events
- Flux value:  $10^6$  hits per second
- Mapping file:
- Detector file:
- Sample event file:

# Data browser

- Tool for browsing RAW data files
- Supports files up to several GBs
- Provides visualization of data words
- Additional useful features can be added – structure validation, statistics, histograms, etc.

# Data browser

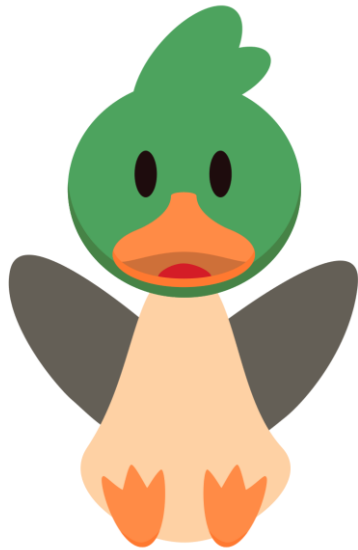
Data Browser: /online/RCCARS/compass-rccars-daq/compass-rccars-daq-rawfile-generator/resources/generated\_data.raw

Open file ... First slice Previous slice Next slice Last slice Jump to slice Jump Slice 8344 out of 10003 Reload file Close file Exit browser

# Images	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32						
99983	1	BOS I (0)	Number of images: 10											Slice number: 8343																								
9983	2	BOS II (0)	Start time of slice: 33372000																																			
10000	3	BOI (2)	Time of image: 33372000																																			
10000	4	BOG (4)	Source ID: 67											View ID: 0			First Hit Time: 5946																					
10000	5	DATA (6)	Frontend ID: 8			Channel ID: 3						Relative Hit Time: 5946																										
10000	6	ADATA (7)	Additional data: 0x0																																			
10000	7	EOG (0x5)	CRC: 0x241351c																											F1: 0			F2: 0			F3: 0		
10000	8	BOG (4)	Source ID: 68											View ID: 0			First Hit Time: 1958																					
1000	9	DATA (6)	Frontend ID: 8			Channel ID: 26						Relative Hit Time: 1958																										
1000	10	ADATA (7)	Additional data: 0x0																																			
1000	11	DATA (6)	Frontend ID: 8			Channel ID: 29						Relative Hit Time: 7014																										
100	12	ADATA (7)	Additional data: 0x0																																			
100	13	DATA (6)	Frontend ID: 8			Channel ID: 9						Relative Hit Time: 8085																										
100	14	ADATA (7)	Additional data: 0x0																																			
100	15	DATA (6)	Frontend ID: 8			Channel ID: 29						Relative Hit Time: 11815																										
10	16	ADATA (7)	Additional data: 0x0																																			
10	17	EOG (0x5)	CRC: 0x16361f8																											F1: 0			F2: 0			F3: 0		
10	18	BOG (4)	Source ID: 71											View ID: 0			First Hit Time: 4387																					
10	19	DATA (6)	Frontend ID: 8			Channel ID: 18						Relative Hit Time: 4387																										
10	20	ADATA (7)	Additional data: 0x0																																			
10	21	EOG (0x5)	CRC: 0x3390580																											F1: 0			F2: 0			F3: 0		
10	22	BOG (4)	Source ID: 72											View ID: 0			First Hit Time: 14785																					
100	23	DATA (6)	Frontend ID: 8			Channel ID: 3						Relative Hit Time: 14785																										
100	24	ADATA (7)	Additional data: 0x0																																			
100	25	EOG (0x5)	CRC: 0x2d40cc																											F1: 0			F2: 0			F3: 0		
100	26	BOI (2)	Time of image: 33372400																																			
1000	27	BOG (4)	Source ID: 68											View ID: 0			First Hit Time: 6744																					
1000	28	DATA (6)	Frontend ID: 8			Channel ID: 7						Relative Hit Time: 6744																										
1000	29	ADATA (7)	Additional data: 0x0																																			
1000	30	DATA (6)	Frontend ID: 8			Channel ID: 23						Relative Hit Time: 6886																										
1000	31	ADATA (7)	Additional data: 0x0																																			
1000	32	DATA (6)	Frontend ID: 8			Channel ID: 9						Relative Hit Time: 8560																										
1000	33	ADATA (7)	Additional data: 0x0																																			
1000	34	DATA (6)	Frontend ID: 8			Channel ID: 29						Relative Hit Time: 8593																										
1000	35	ADATA (7)	Additional data: 0x0																																			
1000	36	DATA (6)	Frontend ID: 8			Channel ID: 13						Relative Hit Time: 11225																										
10000	37	ADATA (7)	Additional data: 0x0																																			

# Summary

- Common goal is to implement the free running format and the high-level trigger system
- To accomplish such task, an advanced framework is required
- The trigger system will create a separate and independent ecosystem capable of emulation of the hardware trigger
- Interface with the current DAQ system must be also designed and implemented
- Many support tools (that are currently in use) must be modified as well
- Any available help would be highly appreciated
- It is a challenging task, but we are on our way



Thank you for your  
attention