Run time structure and slice data building

Konstantin Gritsay on behalf of SPD DAQ team

Joint Institute for Nuclear Research

03.10.2022

Introduction

- SPD is expected to use a trigger less DAQ system. Another name for this is the free-running DAQ system. The front-end electronics must operate in self-trigger mode.
- A clock called the global clock is used for synchronization in the SPD DAQ system. A global clock frequency is 125 MHz (a period of 8 ns). The required jitter for the global clock is ≤ 50 ps.
- Data in the DAQ system is grouped by time into parts called slices. A slice length is 10–100 μs. A sequence of slices forms a frame. The frame length is 0.1–10 s.
- Time in the SPD detector is measured from the beginning of the frame and makes sense only inside that frame. No correlation is assumed between the time in two frames. Thus, one frame is the maximum possible time interval in SPD between time-correlated events.

Measuring time

- The time measurement must provide a continuous and uniform time inside one frame, in particular, there must be no breaks in the calculation of time within the frame. Each hit processed by the front-end electronics must have a time stamp that allows the time of the hit from the start of the frame to be calculated.
- Different electronics can use different time measurement techniques, while ensuring that the above requirements are met.
- Electronics can use internal clocks derived from global clocks to measure time. Two obvious ways to measure time are:
  - resetting the internal clock counter at the beginning of the frame;
  - measurement of the beginning of each slice by internal clock.
- In both cases, the slice number will give the rough time and the internal clock will give the fine time.
- In both cases, the required bit depth of the internal clock counter is determined by the length of the slice: the counter must at least overlap the length of the slice.

Readout control

- Two types of commands are used to control the readout process: synchronous with a global clock and asynchronous.
- The following synchronous commands are used:
  - Set Next Frame,
  - Start of Frame,
  - Start of Slice.
- Synchronous commands together with the global clock are sent to the L1 concentrator from a special system called Time Synchronization System (TSS). All synchronous commands are broadcast commands.
- The following asynchronous commands are used:
  - Disarm,
  - Arm,
  - other commands specific for various front-end electronics.
- Asynchronous commands are sent to the L1 concentrator from the L2 concentrator.

Synchronous commands

- Each slice is addressed by two numbers: the frame number in the run and the slice number in the frame.
- The Set Next Frame command loads the number for the next frame into the front-end electronics and may require more than one clock cycle to implement.
- The Start of Frame command terminates the current frame (if there was one) and simultaneously starts a new frame if before that a new number for the frame was loaded using the Set Next Frame command. The command starts the first slice of the frame simultaneously with the start of the frame, and simultaneously with the stop of the frame, the command stops the last slice in the frame.
- The Start Slice command completes the current slice and starts a new one inside the frame. The front-end electronics automatically numbers the slices, the slice number is reset to 0 by the Start of Frame command.

Asynchronous commands

- Two standard asynchronous commands Disarm and Arm are used to control the reset signal line that comes from the L1 concentrator to the front-end electronics.
- These commands are executed by the L1 concentrator and are not directly visible to the front-end modules.
- The Disarm command sets the active level of the reset signal, in response to which the front-end module enters the reset process.
- The Arm command removes the active level of the reset signal.
- All the time when the reset signal is active, the front-end module must ignore any synchronous commands. Thus, the Arm command enables and Disarm disables synchronous commands for the front-end electronics.
- The Disarm and Arm commands can be addressed to a single front-end module or to all modules connected to a single L1 concentrator.

Asynchronous commands

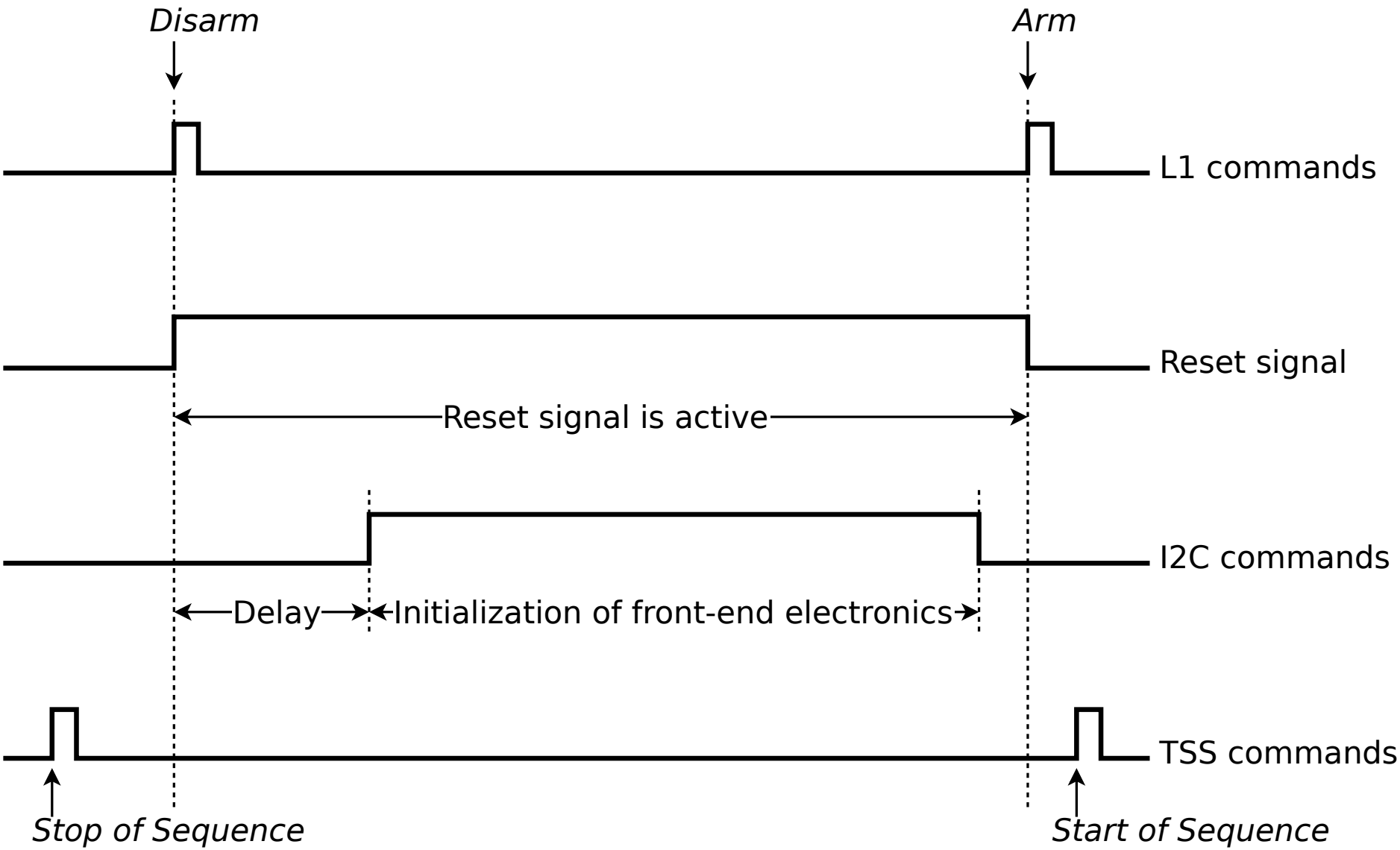
- In addition to the standard asynchronous commands, various front-end electronics may have different sets of specific asynchronous commands for their initialization and monitoring.
- Monitoring commands that do not change the state of the front-end electronics can be used at any time, even in parallel with the synchronous commands.
- These asynchronous commands are transmitted to the front-end module from L1 concentrator in the form of $\mathrm{I^2C}$ commands, and are address commands.
- In addition to commands transmitted over the $\mathrm{I^2C}$ bus, front-end electronics can use fast buses connecting modules and the L1 concentrator and normally used for data and synchronous commands transmission for transmitting large amounts of data at the moment when the response to synchronous commands is disabled.
- This function can be used, for example, to update the firmware in front-end electronics.

Start of the run procedure

- The TSS is controlled using the appropriate commands, two of which are the most important:
  - Start of Sequence,
  - Stop of Sequence.
- Start of Sequence — upon receiving this command, TSS starts generating a sequence of synchronous commands according to the specified parameters;
- Stop of Sequence — upon receipt of this command, TSS stops the generation of synchronous commands.

Run start procedure.

- The generation of all synchronous commands is disabled by issuing the Stop of Sequence command to TSS.
- The Disarm command is issued for all front-end modules, in response to which the modules enter the reset process.
- After some delay necessary to bring the electronics into a state of readiness to receive incoming commands, the DAQ system begins initializing the front-end electronics using specific $\mathrm{I^2C}$ commands.
- The Arm command is issued for all front-end modules. Now the front-end electronics are ready to receive synchronous commands. It should be noted that the Arm command, as well as the Disarm command, is not atomic for the entire installation: it is a sequence of Arm (Disarm) commands addressed to various front-end modules.
- The generation of a synchronous command is started by issuing the Start of Sequence command to TSS.


Disarm
Arm
L1 commands
Reset signal
Reset signal is active
I2C commands
Delay
Initialization of front-end electronics
TSS commands
Stop of Sequence
Start of Sequence

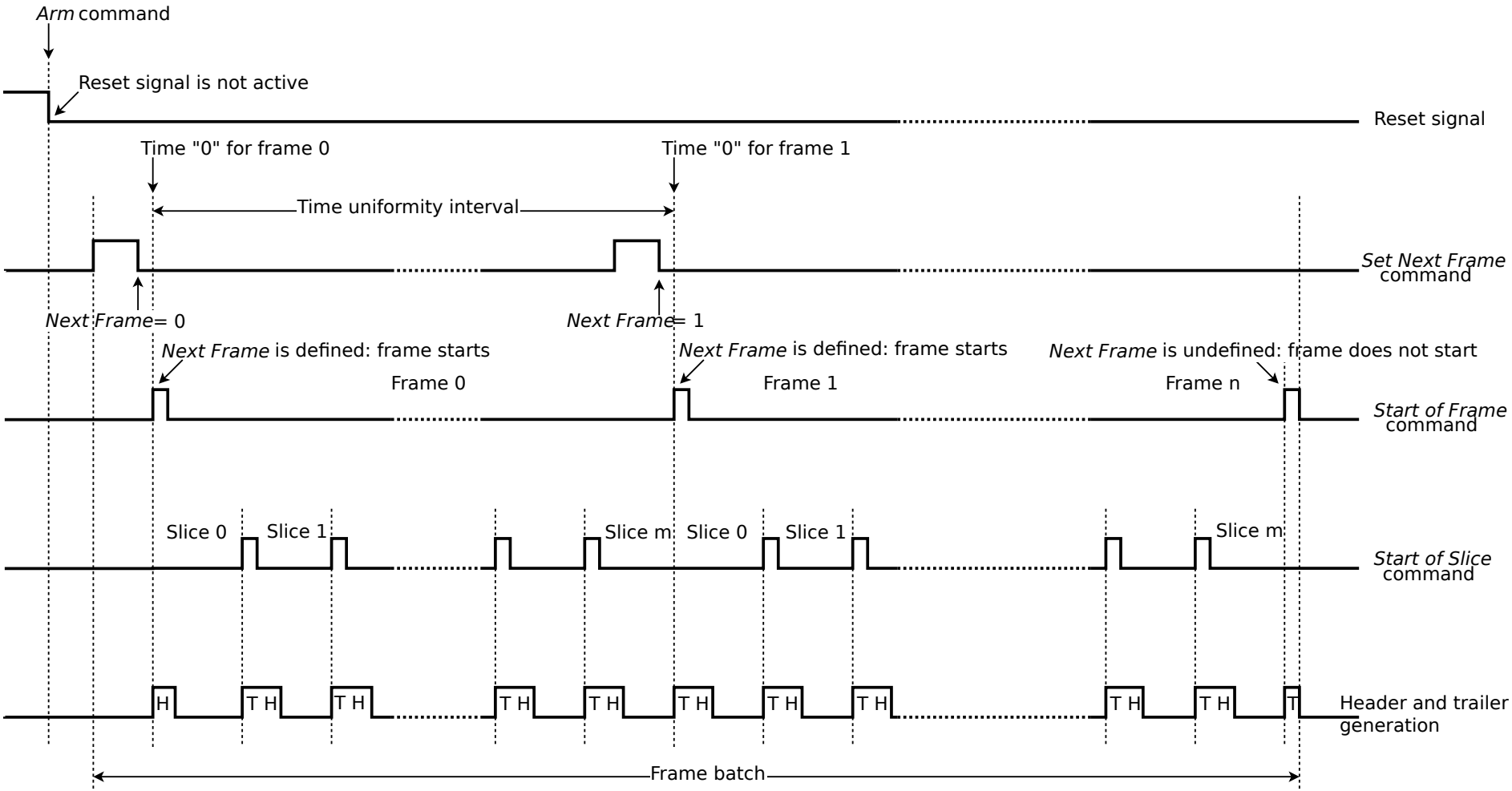
Time structure of the run

- The structural unit of the run is a package of frames, called a frame batch.
- A frame batch contains a continuous sequence of frames following each other, without time intervals between frames.
- On the other hand, there are time intervals between frame batches that can be used by the front-end electronics to perform the necessary periodic actions, such as resetting.
- In the absence of such a need, one frame batch can be stretched for the entire run. In addition, the frame batch is interrupted when the run is put into a suspended state.
- Frames in the run have continuous numbering, independent of the grouping of frames into batches. There is no direct limit on the number of frame batches per run or on the number of frames per batch. There is only a general limit on the number of frames per run, resulting from the size of the corresponding field in the data format.

Time structure of the run

- Before the first frame in the batch, the Set Next Frame command is executed, which loads the number for the first frame in the batch into the front-end electronics.
- All frames in the batch, excluding the last frame, contain the Set Next Frame command closer to their end.
- There is no Set Next Frame command in the last frame of the batch, and therefore the last Start of Frame command ends the frame without starting a new one.
- The frame consists of slices.
- The Start of Frame command simultaneously with the start of a new frame starts the first slice of this frame and simultaneously with the completion of the frame completes the last slice of this frame.
- Inside the frame, the Start of Slice command has completed one slice and is starting a new one.

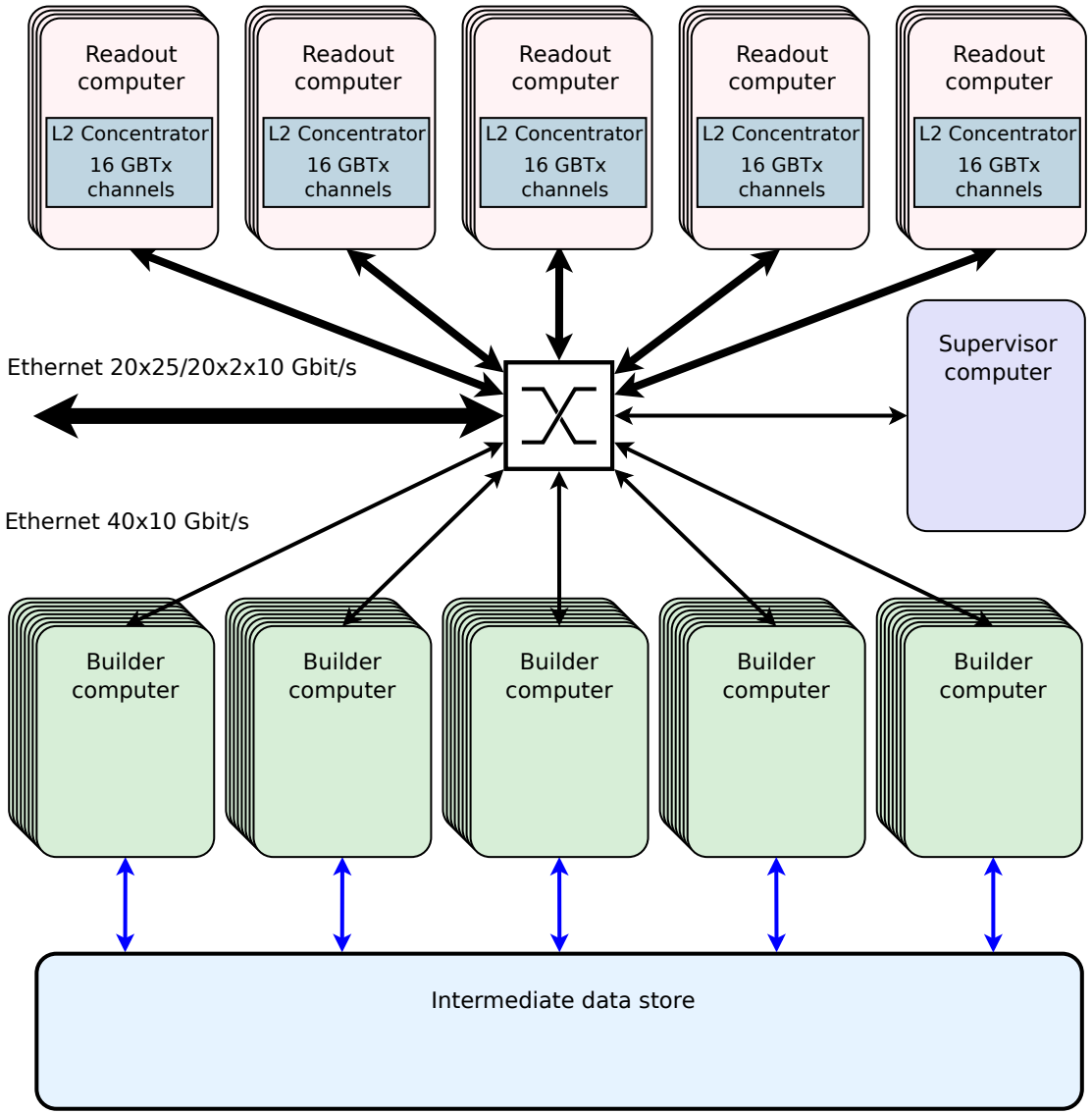
Time structure of the run



Slice building

- The main task of the slice building subsystem is to receive data from the L2 concentrators, combine them into fragments of sufficient duration in terms of astronomical time and acceptable size, and write this data to an intermediate data storage.
- The natural choice of the length of these data fragments is the frame length (0.1–10 s). In some cases the volume of data in a frame may happen to be too large for convenient processing. Therefore, the frame is divided into parts called chunks.
- With an expected total data flow up 20 GB/s, a reasonable chunk length is 0.1–1 s, depending on the actual data flow.
- The splitting of the frame into chunks is performed the slice boundary and is transparent to the software located below in the data processing chain. Thus, the chunk is a data processing unit in the slice-building subsystem.

Slice building



Slice building

- The slice building hardware consists of readout computers with L2 concentrator installed, a network switch, a supervisor computer, and builders computers.
- From the software point of view, slice building includes readout processes, supervisor process, and builder processes.
- The slice building works under control of readiness messages from the builder processes, in response to which the supervisor process assigns chunks for processing to particular builder processes. Each builder process works with one chunk at a time.
- Data buffering in the readout computer provides a transition from the synchronous part of the DAQ chain, working on commands from TSS, to the asynchronous part of the DAQ chain, working on readiness messages from builder computers.

Readout process

- A separate readout computer is used for each L2 concentrator. The expected data flow through one readout computer is about 1 GB/s, which gives a number of readout computers of about 20. The readout computer has 2×10 Gbit/s or 25 Gbit/s network interface(s).
- The readout processes are performed on the readout computers, one process for each L2 concentrator.
- Depending on the actual implementation of the L2 concentrator, the data coming from the L2 concentrator may already be formed as sub-slices or may be independent streams from each front-end card. In the latter case, the readout process must first reorganize data in the form of sub-slices.
- The data organized into sub-slices is buffered in RAM of the readout computer. The buffering depth should be sufficient, taking into account the architecture of the slice building subsystem and possible emergencies, such as delays in data transmission and recording. A reasonable buffering time is 30-60 s, which requires 64-128 GB of RAM.

Builder process

- The builder process is responsible for building the complete chunk data block. The builder process extracts all sub-slices of a single chunk from all readout processes, composes a complete chunk and stores it in an intermediate data store.
- The builder processes run on the dedicated computers that are part of the builder pool. The computer running the builder process can be attached to or detached from the builder pool at any time, including runtime. The number of computers in the builder pool must be sufficient to operate without data loss, but if there are no builder processes available, the data of one chunk is dropped by a command from the supervisor process.
- The builder computers have a 10 Gbit/s network interface for communication with the readout computers and a separate connection to the intermediate data store. Assuming that there are 20 readout computers, around 40 builder computers must run simultaneously to ensure operations with data flows from each readout computer at 1 GB/s.

Supervisor process

- The supervisor process collects information about the received chunks from the readout processes and data processing requests from the builder processes, and basing on this decides which builder process will handle a particular chunk.
- The supervisor process informs the selected builder process, it connects to all readout processes, and starts receiving the chunk data. When chunk data transfer is complete, the builder process closes the connections to the readout processes. The builder process informs the supervisor process when data transfer from the readout computers has finished and when it is ready to process the next chunk.
- The supervisor process and readout processes form the critical core of the DAQ system. A problem with this part of the DAQ system will abort the current run. At the same time, any problem with one of the builder processes can lead to data loss for one chunk, but should not stop data taking.

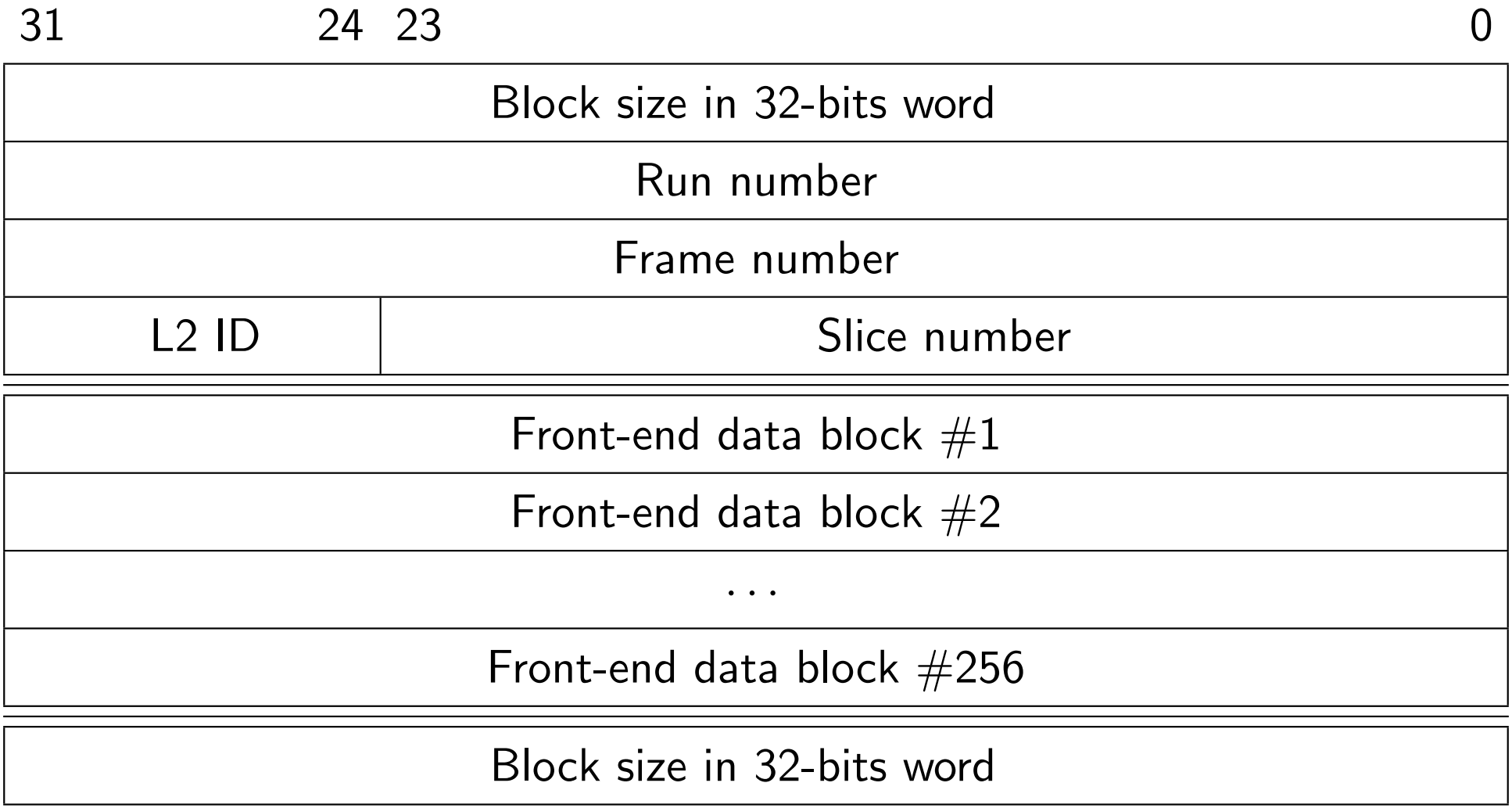
Data format

- The result of the slice building process is data files stored in an intermediate data storage.
- One chunk data is usually stored in a single data file, but can be split into multiple data files if necessary.
- The data file consists of slice data blocks.
- The data files do not have a header or other metadata — a database will be used to save the metadata.
- The total size of the data and all its logical units must be aligned to the size of a 32-bits words.

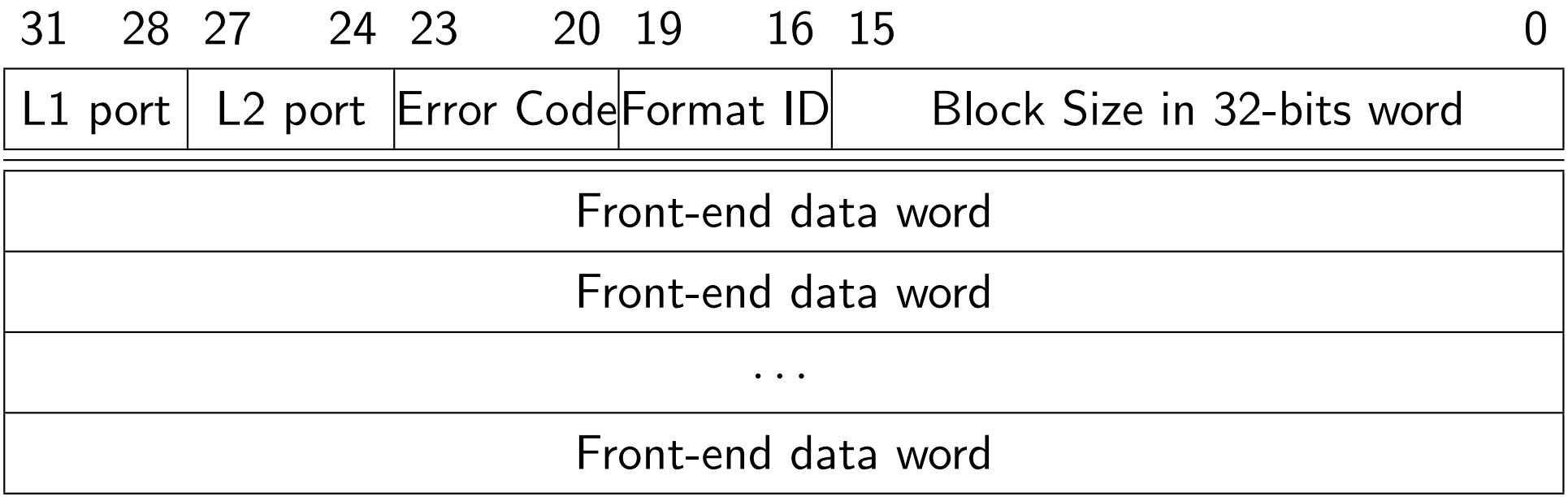
Slice data block

31 — 24	23 — 0
Data size in 32-bits word	
Run number	
Frame number	
Reserved	Slice number
L2 concentrator data block #1	
L2 concentrator data block #2	
...	
L2 concentrator data block #n	
Error code	
Data size in 32-bits word	

L2 concentrator data block



Front-end data block



Data file

- This data file format is preliminary.
- Data files of this format will have a limited 'lifetime': they will be processed by an online filter, and then deleted.
- A different data format will be used to record the result of processing the online filter.

Thanks for attention