

Contribution to run control discussion: excerpt from internal Salsa discussions

Irakli Mandjavidze

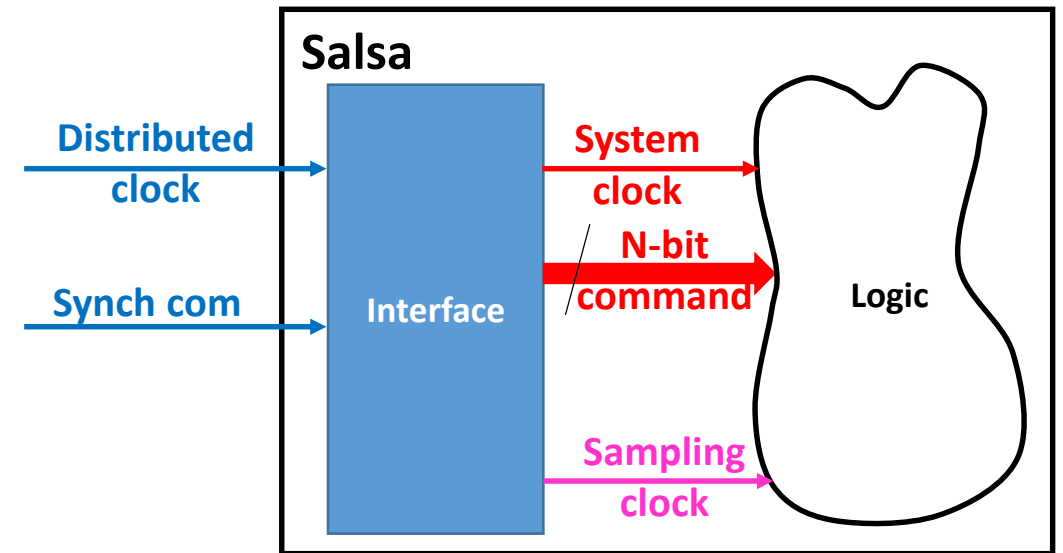
*Irfu, CEA Saclay
Gif-sur-Yvette, 91191
France*

Internal

21/Jan/2024

- Collider experiments with repeated beam structure
 - LHC : 40 MHz Bx clock, 3654 bunches per Orbit, well defined beam gaps
 - EIC : 100 MHz Bx clock, 1200 bunches per Revolution, well defined beam gaps
 - Distribution of system clock and synchronous commands to orchestrate data taking over heterogeneous sub-systems
 - Regular or on demand sync commands like bunch crossing 0, start / stop, sync, test trigger ...
 - Synchronous machines : strong relationship between the system clock and physics data
- Continuous uniform beam experiments
 - Clas12 : 250 MHz clock
 - Fermilab test facility : 53 MHz
 - Expect a system clock and some synchronous commands to orchestrate data taking over heterogeneous sub-systems
 - Regular or on demand sync commands like start / stop, sync, start of spill, end of spill ...
 - Synchronous or asynchronous experiments : relationship may or may not exist between the system clock and physics data
- Small standalone setups
 - Muon tomography telescopes, cosmic ray test benches, detector validation stands
 - Local clock with few locally generated synchronous commands to acquire data mostly from same type of detectors
 - Typically start / stop, sync
 - Asynchronous setups : no relationship between the system clock and physics data
- Triggered or streaming

- **System (aka bunch crossing) clock** : common to all sub-detectors in an experiment
 - Data is coarsely tagged by a time stamp from this clock domain
 - Salsa maintains only N (e.g. 12) LSB-s of the time stamp
 - There is a fixed relationship between the time stamp in the Salsa and in the complete time stamp in aggregator
- **Sampling clock** : used by Salsa to acquire the signal shape
 - Derived from system clock and is its (sub)multiples (e.g. 2, $\frac{1}{2}$) or simple (M / N) fractional
 - Known frequency and phase relationship exists between the sampling clock and system clock
 - Coarse time stamp is complemented by a fine time stamp indicating the sample phase with respect to the system clock
- **Distributed clock** : delivered to Salsa
 - System clock is recovered from distributed clock
 - Frequency and phase
 - Sampling clock is produced from distributed clock
 - Frequency and phase



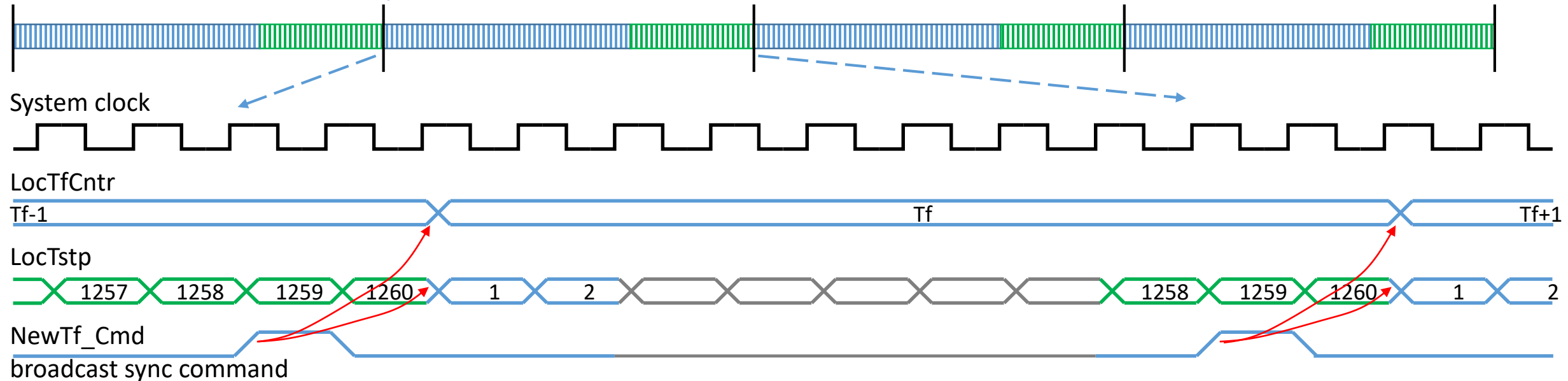
Synchronization for data aggregation:
time frames, data collection...
... and synchronous commands

- A state machine that brings FE (ASIC) from “on” state to “running” state through several intermediate states
 - For “lower” level states, transition requires asynchronous commands (e.g. configuration)
 - For “higher” level states, transition requires synchronous commands (e.g. data taking)
- An example of states
 - On : after power-up
 - Attempts to validate clocks, may require collaboration with the distributed clock source
 - ClockOK : input and derived clocks are validated
 - Attempts to validate rest of the logic
 - Initialized : all logic is up and running with its default configuration
 - Waits for successive asynchronous commands for configuration
 - Ready : configuration done, ready to take data
 - Waits for successive **synchronous** commands
 - Running : data taking in progress
 - Data taking in progress; re-configuration is forbidden
 - Waits for **synchronous** commands either to return to Ready state or to go to “Pause” state
 - Pause : data taking is suspended
 - Waits for **synchronous** commands
 - To recover from errors and resynchronize
 - To resume data taking
 - To go to stop data taking
 - Error : for whatever reasons
 - May requires a “heavy” intervention, but an attempt can be done to recover quickly

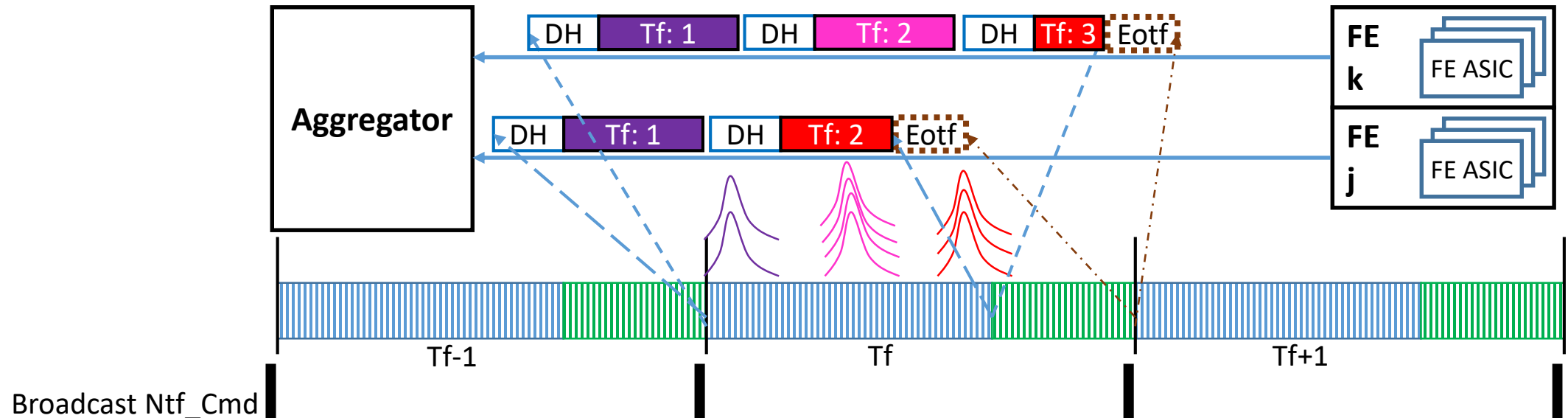
Discussed in what follows

- A versatile ASIC may be deployed in a system with periodic time frames
 - Like LHC or EIC
- It can be used in systems without periodic timing structure
 - Test beams, some fixed target experiments, cosmic ray setups ...
- Therefore, operating with time frames is optional
- But can be handy especially in streaming readout
- Time frames represent a certain number of system clock cycles
 - 3564 ~25 ns bunch crossing orbit at LHC : ~89.1 μ s
 - 1260 ~10 ns bunch crossing revolution at EIC : ~12.8 μ s
 - 4096 8 ns system clocks for Clas12 MPGD readout : 32.8 μ s
- Time frames can be delimited by a dedicated synchronous command...
 - BC0 @ LHC
 - RevTick @ EIC
- ... or can be in absence of such can be generated locally
 - 12-bit counter roll-over in Clas12 MPGD readout

- Assume there is a periodic synchronous command : `Ntf_Cmd` (new time frame)
 - It always arrives during the last system clock cycle of the timer frame
 - BC0 @ LHC
 - RevTic @ EIC
- FE (ASIC) maintains N-bit counter of system clocks within a timeframe - a local timestamp : `LocTstp`
 - N is small – enough to avoid roll over within the time frame
 - 12 @ LHC; 11 @ EIC
- FE (ASIC) maintains M-bit counter of time frames – a local time frame counter: `LocTfCntr`
 - M is even smaller – enough to avoid roll over during the data collection in aggregator
 - 4 bits should be enough, but can be any reasonably small number of bits



- Both aggregator (chain) and FE (ASIC) perform some actions on reception of Ntf_Cmd
 - Verify that Ntf_Cmd arrives always at the expected LocTstp - synchronization
 - Reset LocTstp : possible convention forcing the timestamp of very first clock cycle to 1
 - Increment LocTfCntr
- Reminder : FE (ASIC) tags data by LocTstp
 - And if needed by LocTfCntr – to be decided
- Optionally FE (ASIC) can be instructed to send a short end of time frame packet to aggregator : Eotf_Pack
 - It includes elapsed local time frame counter LocTfCntr-1 and whatever additional handy information
 - The Eotf_Pack is sent after all data packets belonging to the elapsed time frame has been sent
 - Assist aggregator in data collection : no data expected after Eotf_Pack

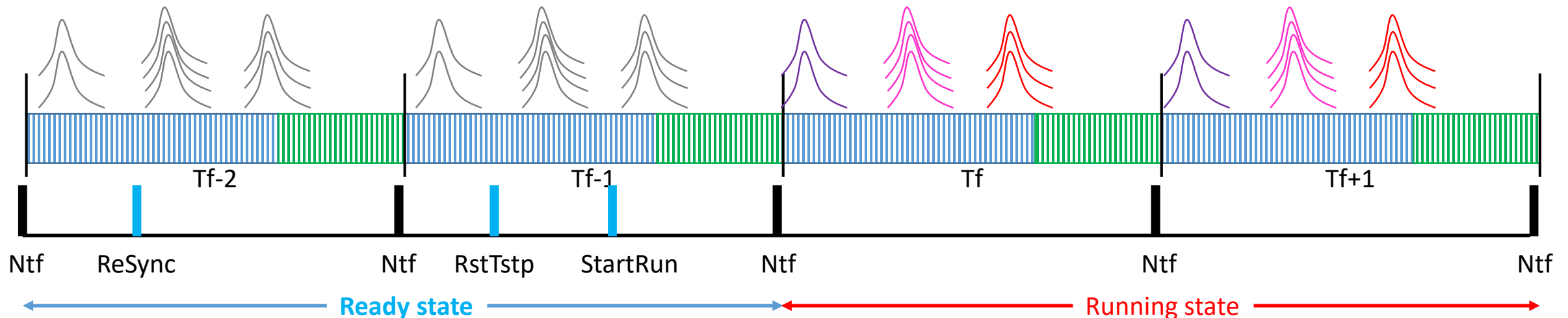


- The time frame mechanism works for all enumerated use cases
 - Repeated beam structure, uniform beam structure, small standalone setups, triggered, streaming
 - Previous discussion covered systems with centralized synchronous command distribution
 - In uniform beam setups, central supervisor can be programmed to send periodic time frame delimiting commands
 - In standalone setups FE (ASIC) can be programmed to generate Eotf_Pack
 - Upon roll-over of the local time stamp counter LocTstp or when it reaches a preprogrammed value
- In streaming readout systems time frames are handy to preform data aggregation
 - The new time frame synchronous command Ntf_Cmd can be considered as periodic, constant rate trigger
 - Local time frame counter LocTfCntr can be used as a sort of “event ID”
 - The end of time frame packet Eotf_Pack can assist in data aggregation closure at the aggregator level
 - Aggregator can compare M LSBs of its LocTfCntr counter with the M-bit LocTfCntr received from FEs (ASICs)
- In triggered readout data packets are tagged by a timestamp and event id
 - Synchronization mechanism assures that LocTstp counters and Event IDs are synchronized throughout the system
 - Aggregator compares N LSBs of its LocTstp counter with the N-bit LocTfCntr received from FEs (ASICs)
 - The end of time frame packet EotfPack can assist in data aggregation closure at the aggregator level
- Finally, end of time frame packet reinforce system synchronization checks
 - If Eotf_Pack has not been received following the Ntf_Cmd or periodically, packet loss occurred

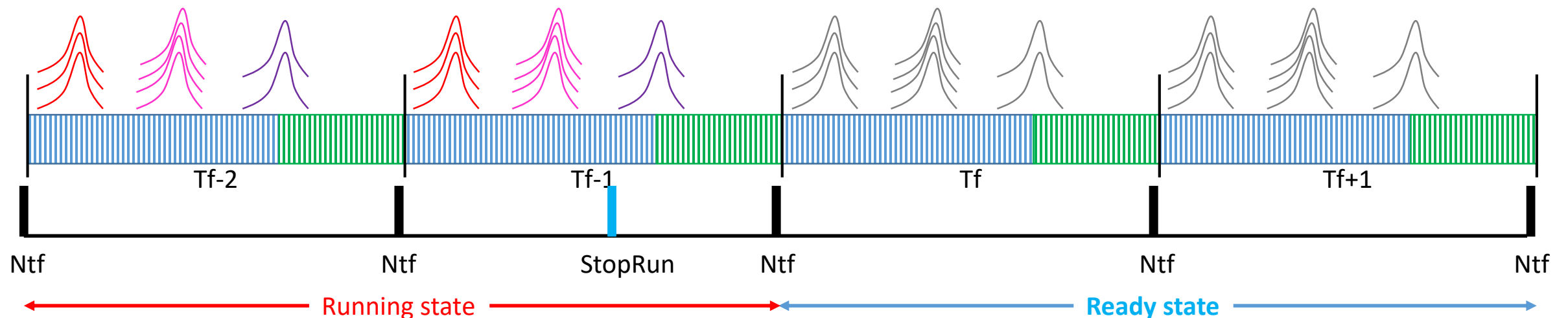
Some proposals : **run control** synchronous commands

- The aim is to ensure synchronous transition to/from running (data taking) state of entire acquisition system
- Also quick re-synchronization or error recovery
- The run control commands have mostly broadcast nature
- A list of few possible run control commands
 - StartRun_Cmd
 - StopRun_Cmd
 - Pause_Cmd
 - Resume_Cmd
 - ReSync_Cmd
 - RstTstp_Cmd
 - RstEvld_Cmd
 - Ntf_Cmd
- Some possible actions for these commands are discussed in the following
- Certain synchronous commands may require an acknowledgment packet to be sent to a system supervisor
 - Example can be commands that result to the state change, such as Start, Stop, Pause, Resume

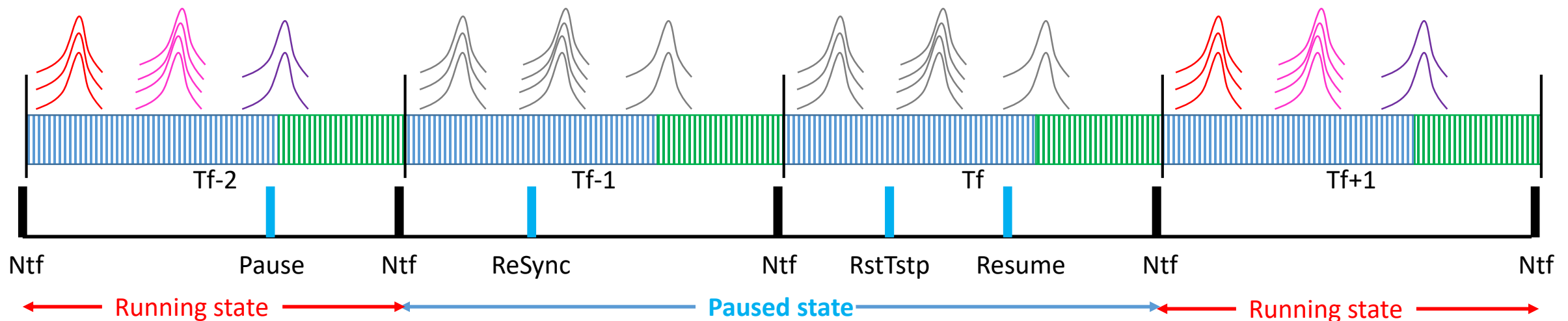
- The frontend (ASIC) is in an Ready state
 - Configuration done, ready to take data
 - Waits for a sequence of successive synchronous commands
- Assume synchronization of sub-systems has not been done yet, the sequence can be
 - ReSync_Cmd : a kind of “warm” reset, configuration registers are not affected
 - Makes sure all current data in pipelines and in FIFOs are processed and sent out
 - Performs necessary actions on resynchronization if any
 - RstTstp_Cmd : resets Loc_TstpCntr and possibly Loc_TfCntr to predefined values
 - (In triggered mode) RstEvld_Cmd : resets Loc_Evld to 0
 - StartRun_Cmd : indicates that data taking must start after next Ntf_Cmd
 - Ntf_Cmd : frontend (ASIC) goes to Running state and starts to deliver acquired data



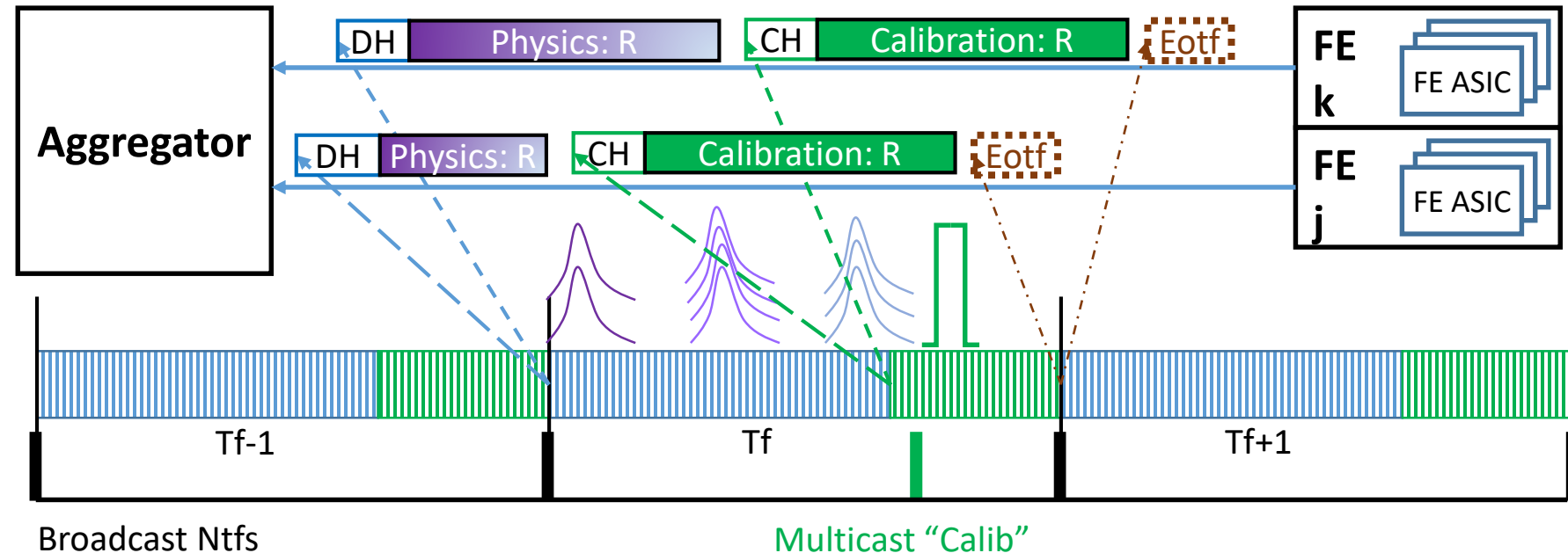
- The frontend (ASIC) is in Running state
 - Needs a sequence of successive synchronous commands to stop data taking
- The sequence can be
 - StopRun_Cmd : indicates that data taking must stop after next Ntf_Cmd
 - Ntf_Cmd :
 - Frontend (ASIC) does not accept new data
 - Finishes data delivery of the elapsed time frame
 - Goes to Ready state



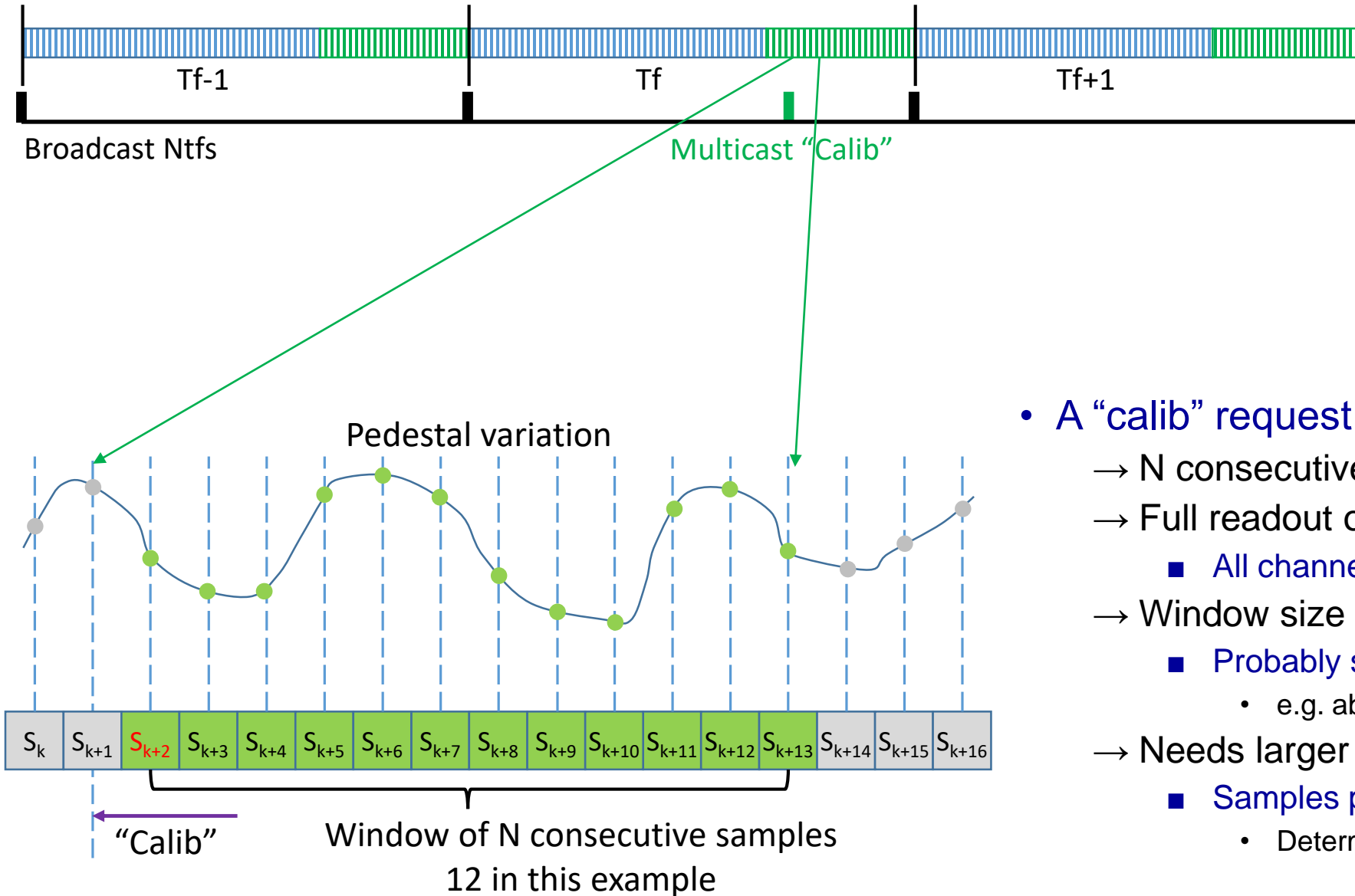
- An attempt of **quick recovery from some run-time errors** can be envisaged with **Pause** and **Resume** commands
 - Upon reception of **Pause_Cmd** the system goes to Paused state following the next Ntf_Cmd
 - Data from elapsed time frame delivered
 - Healthy sub-systems are kept ready to restart data taking
 - Faulty sub-system attempts to recover
 - Warm reset, link synchronization, whatever
 - If faulty sub-system becomes healthy, restart data taking sending a sequence of synchronous commands
 - ReSync_Cmd, RstTstp_Cmd, (RstEvld_Cmd)
 - Upon reception of **Resume_Cmd** the system goes to Running state following the next Ntf_Cmd
- **Pause / resume commands can be handy for system-wide monitoring inducing very little dead-time**



- DAQ sends calibration request to frontends (ASICs)
 - Multicast calibration synchronous commands : Calib_Cmd
- Frontends (ASICs) respond with non-ZS data samples
 - Concerned aggregator performs calibration data collection from all FE / ASICs
 - As described previously for physics data
- This sequence repeats programmable number of times to form a complete calibration cycle
 - Need to gather sufficient per channel statistics



- Consider calibration type with readout of a programmable number of consecutive samples
- Consider calibration by coupling synchronous commands, like arming a pulser followed by calibration



- A “calib” request results in a Window readout
 - N consecutive samples
 - Full readout of non-ZS data
 - All channels of all ASICs
 - Window size programmable
 - Probably same as for ZS readout
 - e.g. able to contain a typical signal shape
 - Needs larger de-randomizer buffers in ASICs
 - Samples pile up awaiting their turn to be sent
 - Determined by ASIC output throughput

- A list of possible commands

- Run control

- StartRun_Cmd
- StopRun_Cmd
- Pause_Cmd
- Resume_Cmd
- ReSync_Cmd
- RstTstp_Cmd
- RstEvd_Cmd
- Ntf_Cmd

- Calibration and monitoring

- Calib_Cmd
- Pulser_Cmd
- Mon_Cmd

- Internal

- LinkSync_Cmd
- ...

- Trigger

- 16 synchronous commands is enough ?

Summary

- What was shown is (possible) example only
 - To trigger discussions
- Need to know ePIC DAQ state machine
 - Understand its influence on frontend design
 - Make sure ePIC DAQ takes into account the needs of the MPGD frontends and readout
- Need to know synchronous commands
 - Defined by ePIC DAQ an obligatory set of commands to obey to and a protocol to follow
 - e.g. broadcast commands like StartRun, EndRun, etc.
 - A set of commands needed by the MPGD readout
 - e.g. multicast commands like Calib
 - Support for coupled sync commands
 - e.g. ArmPulser followed by Calib
- Need to know command decoding strategy dictated by ePIC
 - A command per bunch crossing
 - Bit-by-bit aggregation of commands over several bunch crossings
- Need a document that sets the ePIC DAQ rules