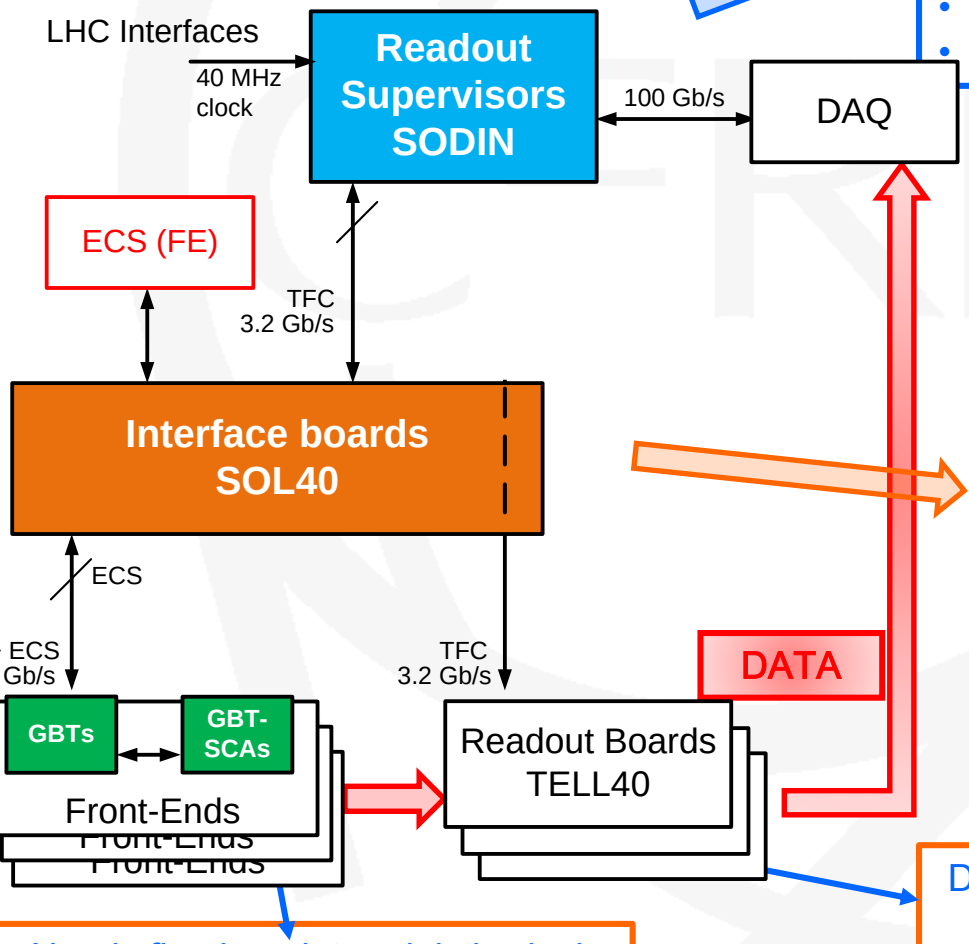
Controlling TFC

MiniDAQ workshop

31 July 2017

F. Alessio, CERN

The upgraded TFC system at a glance



SODIN responsible for controlling upgraded readout system

- Distributing timing and synchronous commands
- Manages the dispatching of events to the EFF
- Rate regulates the system

SOL40 responsible for interfacing FE and TELL40

- Fan-out TFC information to TELL40
- ~~Fan-in THROTTLE information from TELL40~~
  - do not distribute throttle back, but local throtting in TELL40
- Distribute TFC information to FE
  + clock and timing
- Distribute ECS configuration data to FE
- Receive ECS monitoring data from FE

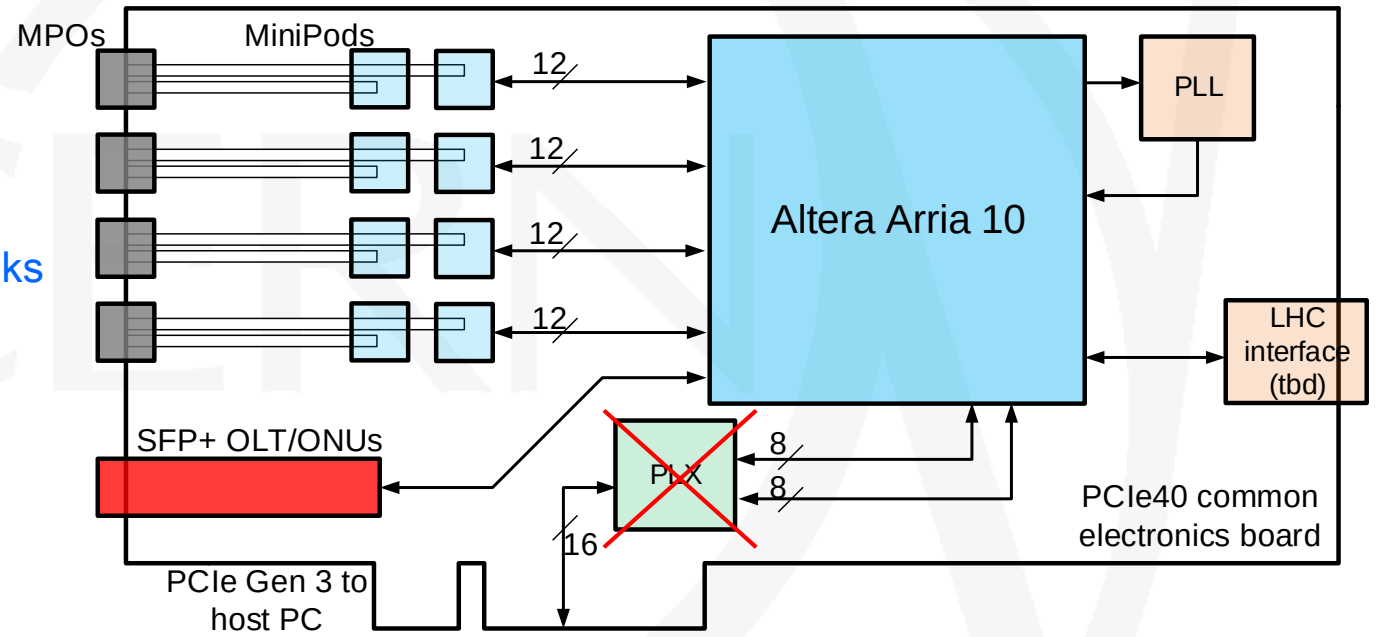
Does not need fixed and deterministic clock nor latency + readout control

Needs fixed an deterministic clock phase and fixed latency + configuration + monitoring + readout control

TFC on common hardware backbone

In principle:

- Firepower of 48 bidir links
- Dedicated TFC link on SFP+
- Deterministic on board external PLL for resynchronization



Same exact hardware (PCIe40), firmware defines the functionality:

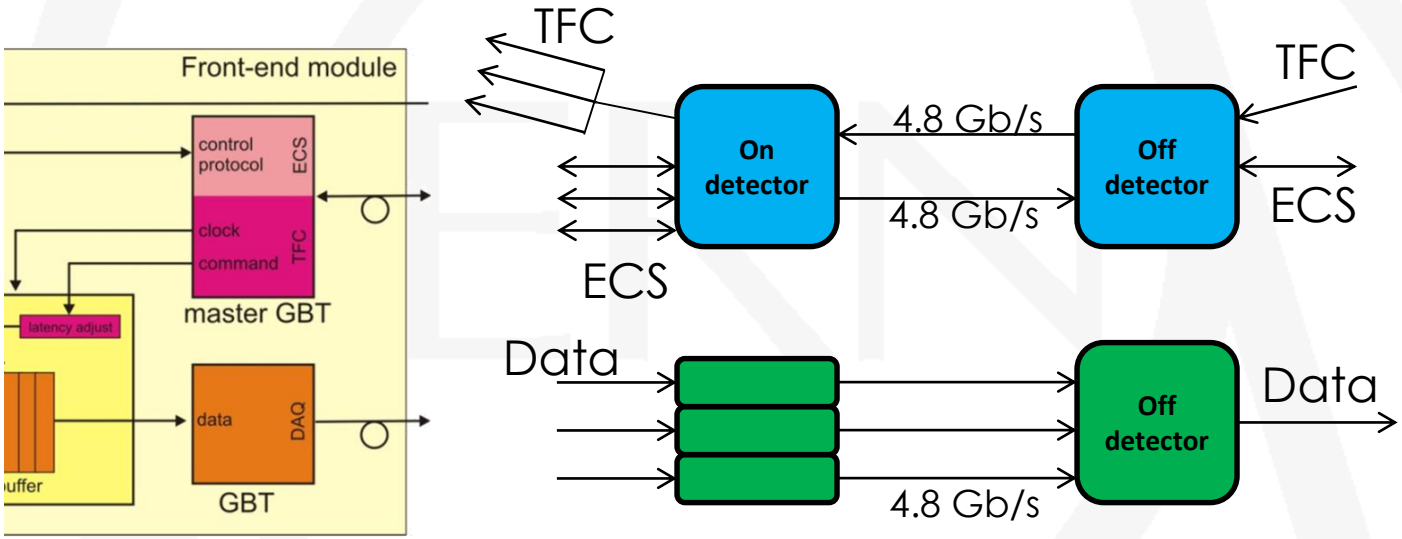
SODIN, SOL40 (TELL40)

- The Host PC is the PC controlling the board
- In principle ~100 Gbps of ECS access

Also, PCIe40 card has an interface to external electrical signals

- LHC or current TFC or external trigger....

Fast & Slow Control to FE



Separate links between controls and data

- A lot of data to collect
- Controls can be fanned-out (especially fast control)

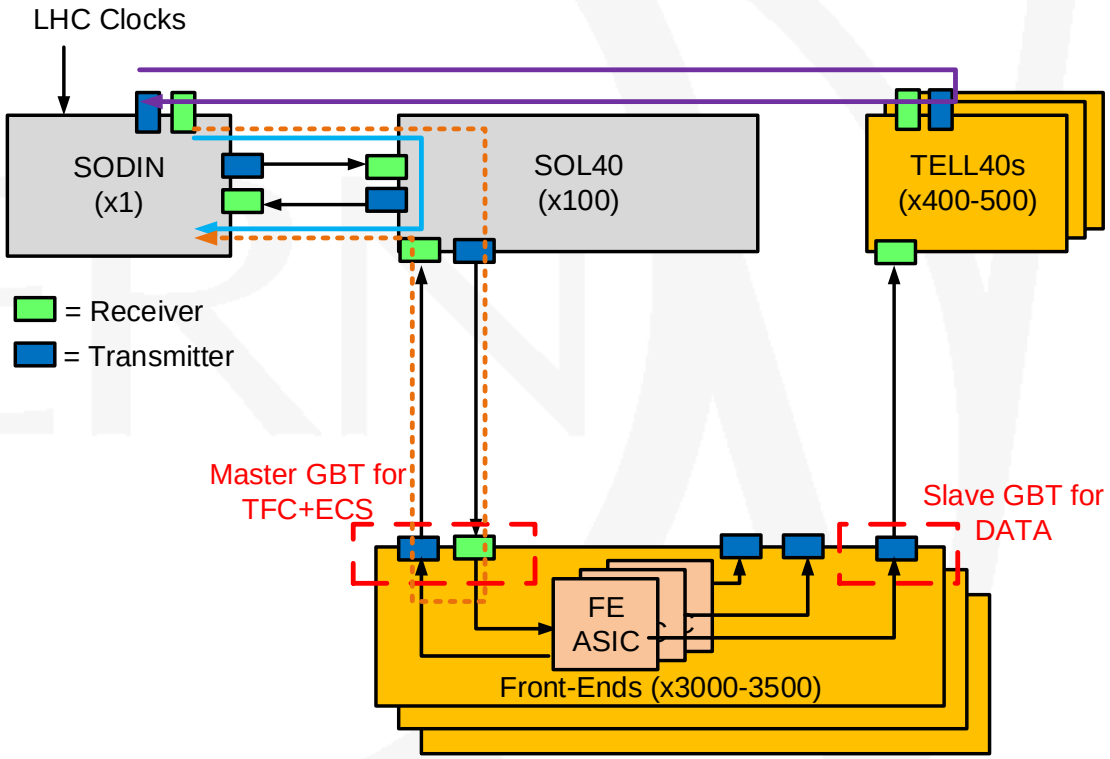
Compact links merging Timing, Fast and Clock (TFC) and Slow Control (ECS).

- Extensive use of GBT as Master GBT to drive Data GBT (especially for clock)
- Extensive use of GBT-SCA for FE configuration and monitoring

Timing and command distribution

Master GBT @ FE controls the FE ASIC + Slave GBTs

- Clock from Master GBT
- TFC commands on e-links
- ECS configuration and monitoring through SCA
- Slave GBTs controlled through SCA from Master GBT



Fixed latency and deterministic phase recovery of TFC commands is ensured by combination of FPGA and GBT features

→ Customization is needed to synchronize links and make sure that TELL40s decodes data properly

→ Customization is needed to properly decode TFC commands

→ ECS to FE through TFC links via SOL40 (Joao)

General philosophy for flow control

Flow control is centralized and synchronous

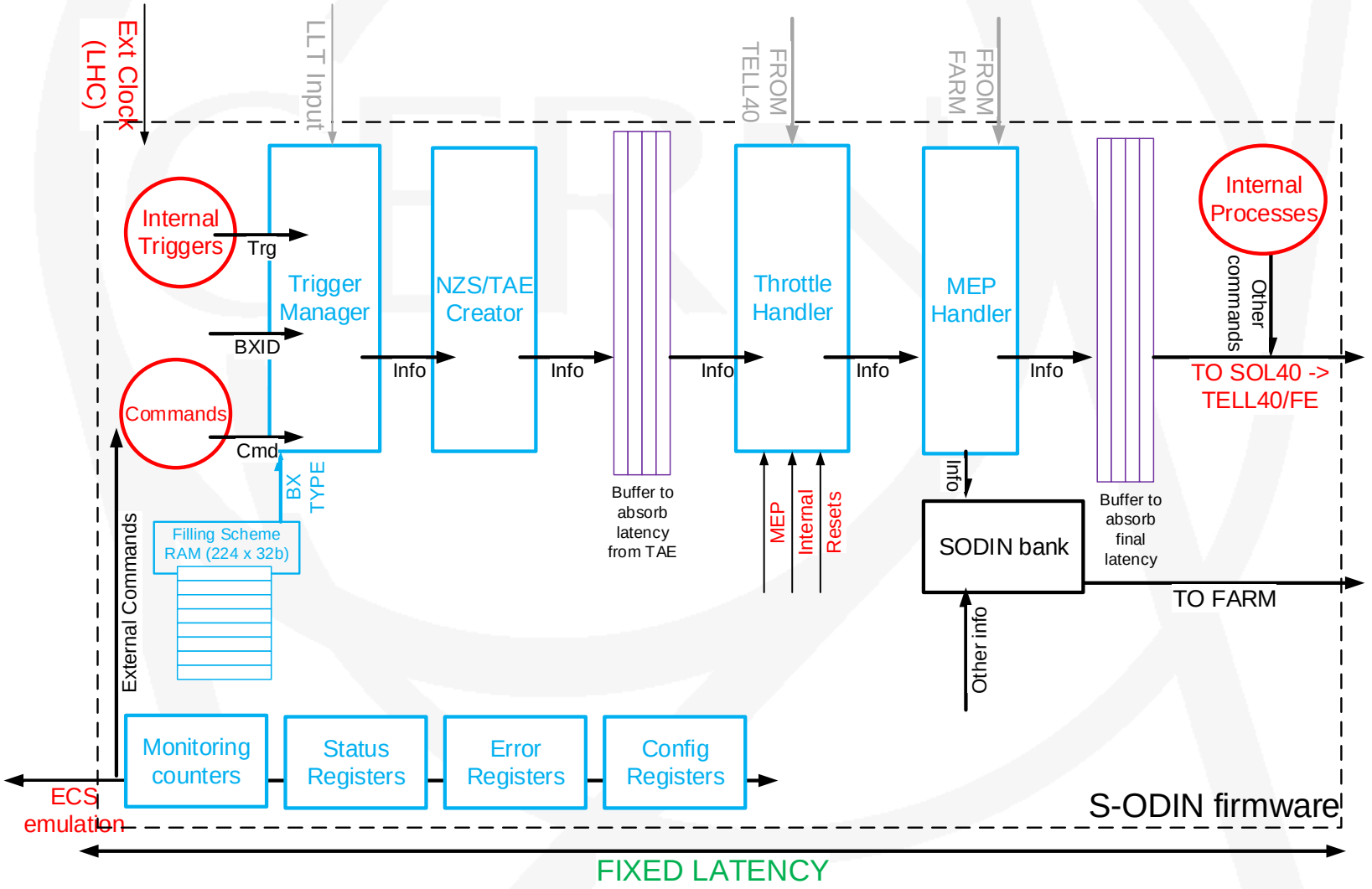
- One set of TFC commands per BXID, at 40 MHz
- TFC will make sure that the system is synchonized or that the usage of commands is consistent. Programmability of the firmware allows to create procedures.
  - ✓ Resiliant to create «ad-hoc» procedures
  - ✓ Generalization as a mean to cover for all scenarios
    - (increase complexity, but covers more cases)

Data readout is distributed and asynchronous

- No need to distribute a trigger to FE so data links can be either out of synch or offset in time
  - ✓ TELL40 has logic to align data from all input links
- Data across links should be consistent
  - ✓ If it's a calibration command at a BXID, it should be calibration data for all fragments from the whole detector on the same BXID.
- TELL40 data decoding at input should be independent from TFC
  - ✓ FE runs anyway at 40 MHz and too much memory would be needed to pipe TFC
  - ✓ Only applies a trigger and open/close MEPs (other information is for redundancy)

→ Need at least a minimum set of common solutions to ensure synchronicity, consistency and that bandwidth matches

The SODIN fw


Ext Clock (LHC)
LLT Input
FROM TELL40
FROM FARM
Internal Triggers
Trg
BXID
Commands
Cmd
Trigger Manager
BX TYPE
Filling Scheme RAM (224 x 32b)
External Commands
NZS/TAE Creator
Info
Buffer to absorb latency from TAE
Throttle Handler
MEP
Internal
Resets
MEP Handler
Info
SODIN bank
Other info
Buffer to absorb final latency
Internal Processes
Other commands
TO SOL40 -> TELL40/FE
TO FARM
Monitoring counters
Status Registers
Error Registers
Config Registers
ECS emulation
S-ODIN firmware
FIXED LATENCY

TFC commands to TELL40, list

Protocol on information to tell the TELL40 what each event is:

63 .. 52	51	50	49 .. 18	17 .. 14	13 .. 10
BXID(11..0)	**Reserve**	**MEP Accept**	**MEP Dest(31..0)**	**Trigger Type(3..0)**	**Calibration Type(3..0)**

9	8	7	6	5	4	3	2	1	0
Synch	**Snapshot**	**Trigger**	**BX Veto**	**NZS Mode**	**Header Only**	**BE Reset**	**FE Reset**	**EID Reset**	**BXID Reset**

Trigger command to accept events for that BXID:

→ Trigger acting as a VETO to rate regulate the system

- Not based on physics decision (NO LLT)
- Unbiased rate regulation of the system by reducing the rate at which trigger is set

MEP accept command when MEP ready:

- Take MEP address and send all fragments to that address
- Dynamic mechanisms based on well-oiled mechanism used today
- Investigations ongoing to see if a different mechanism is more suited for upgraded system

Rate regulation can also help to reduce output bandwith

→ Rate regulation in SODIN was also considered as the safest mechanism to reduce output bandwidth should your FE send too much data wrt to the TELL40 output bandwidth

→ Out of the full 40 MHz, only ~28 MHz contains beam-beam, ~4 MHz contains beam-empty/empty-beam, ~8 MHz contains emtpy-empty

- ✓ SODIN can have a programmable mask that defines the rate of crossing types to be kept
- ✓ 100% for bb, 25% for eb/be and 10% for ee?
- ✓ Effectively the full input rate is 30 MHz.

→ This is described in EDMS 160693.

→ Usage of a TRIGGER sent to TELL40 to discard events which are not useful.

TFC commands to FE, list

Protocol on information to FE for flow control:

23 .. 12	11 .. 10	9	8 .. 5	4	3	2	1	0
BXID(11..0)	Reserve	Snapshot	Calibration Type(3..0)	BX Veto	NZS Mode	Header Only	FE Reset	BXID Reset

Each command should have a local configurable delay

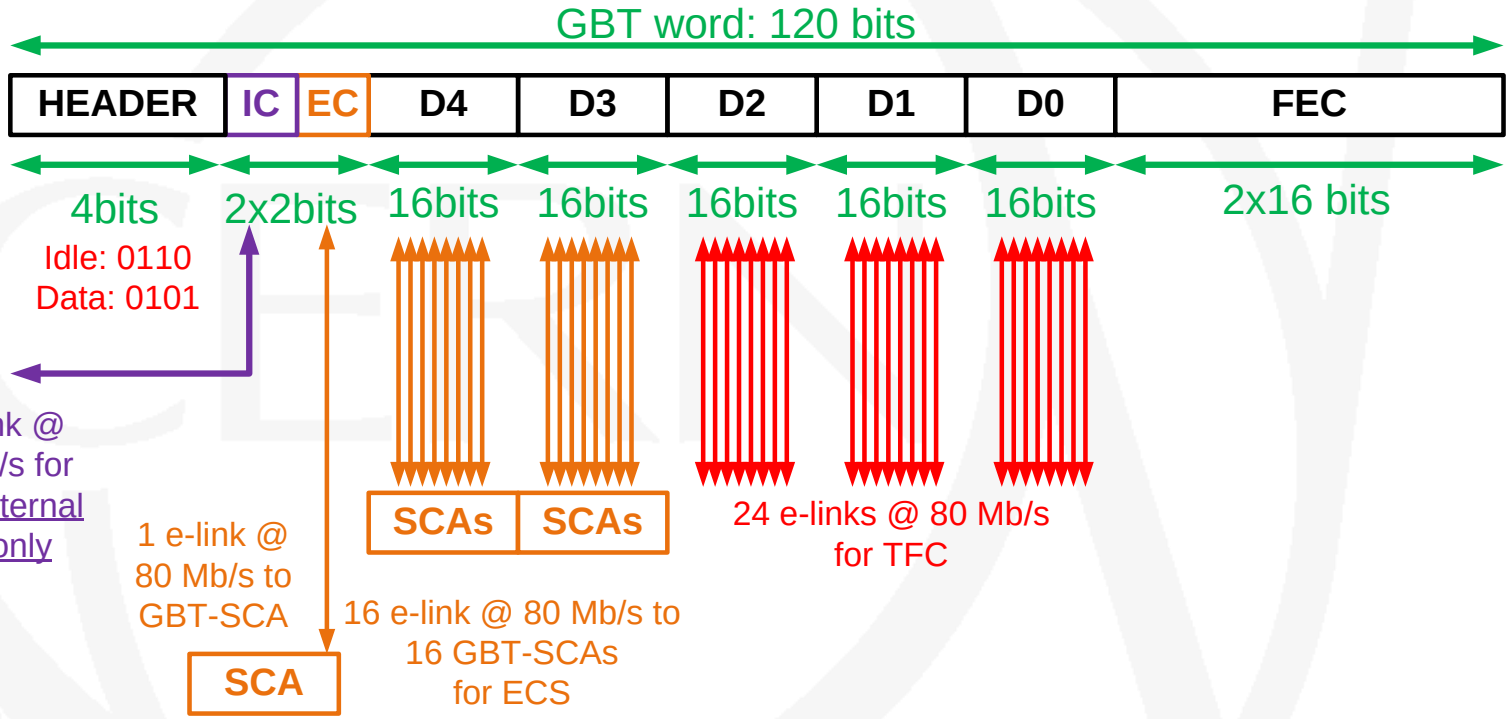
- → GBT does not support individual delays (only global delay)
- → Need for «local» pipelining: detector delays+cables+operational logic (i.e. laser pulse? Calibration?)

To allow use of commands/resets for particular BXID, TFC word will arrive before the actual event takes place

- → Accounting of delays in SODIN: for now, 16 clock cycles earlier + time to receive
- → Aligned to the furthest FE

Each sub-detector can choose whichever set of TFC commands they need and in which specific position (e-link) → configurable mapping done in SOL40

The TFC+ECS GBT protocol to FE



→ Mapping of TFC commands on GBT bits can be customized to your needs
- Number of SCAs, speed of e-link and position
- Also which TFC commands and their position and if they need to be copied many times
- Make you sure you get your mapping right

→ Come to me and let's add it in the SOL40 firmware, this is needed to compile a firmware and see how many FPGA resources it may take

→ It may have an impact on how many SOL40 you need

TFC commands to FE explained

"BXID" and "BXID Reset"

→ Every TFC word carries a BXID for synchronicity checks of the system

→ A BXID Reset is sent at every turn of the LHC (orbit pulse)

- ✓ Only reset the internal bunch counter of the FE
- ✓ This must be in your FE chip

→ BXID can be ignored by sub-detectors if this is compensated by a mean to check the synchronicity of the system

"FE RESET"

→ Bit set for one clock cycle in TFC word sent to FE and TELL40

→ Reset of FE operational logic for data processing, formatting, transmission…

- ✓ Should not touch the internal bunch counter
- ✓ FE electronics should be back as soon as possible: SODIN will ensure no data is being accepted during the FE reset process (by setting Header Only and veto trigger).
- ✓ Reset TELL40 data input logic: the same bit is sent to TELL40 for same BXID.
- ✓ Followed by a SYNCH command.

"BE RESET"

→ Bit set for one clock cycle in TFC word, only to TELL40

→ Reset of TELL40 operational logic for data processing, formatting, transmission…

- ✓ TELL40 should be back as soon as possible: SODIN will ensure no data is being accepted during the BE reset process (by setting Header Only and veto trigger).
- ✓ May be or may not be followed by a SYNCH command. In principle is not needed if TELL40 can still monitor the BXID at the input.

TFC commands to FE explained

"HEADER ONLY"

→ Idling the system: only header (or few bits) in data word if this bit is set

- ✓ Multiple purposes: set it during reset sequence, during NZS transmission, during TAE mode…
- ✓ Header Only also can be set in case of back-pressure. If rate regulation is applied it could be chosen to sent also Header Only to FE

"BX VETO"

→ Based exclusively on filling scheme

- ✓ Only header (or few bits) in data word if this bit is set
- ✓ Allows "recuperating" buffer space in a LHC-synchronous way
- ✓ Load filling scheme in SODIN, then apply recipe (which BX Type to keep, can also define a specific rate for it)

✓ BX Veto and Header Only commands are identical from FE point of view → *ORed*

"EID RESET"

→ Reset the Event Counter ID (64 bits), only sent to TELL40

- ✓ This is the only unambiguous identifier of an event and its fragments
- ✓ EID monotonically increases every time an event is accepted
- ✓ Reset only if Run number changes or if triggered via ECS

A word on the Event ID

The Event ID (64 bits) is the unique identifier of an event for the “time it is sitting in the system”

- → Events are recorded asynchronously, so event fragments (MEPs) can come out of the TELL40 at very different times → this is the only way an event is uniquely identified
- → Event ID shall increase monotonically for each run.
  - ✓ Only the Reset issued by SODIN resets it.
  - ✓ And the process is triggered by either ECS (asynchronously) or by a change run (STOP Trigger/START Trigger)
- → First Event ID is 0

The Event ID should be in the header of the MEP packet sent out to a destination

- → All other events in the MEP will have Event ID reconstructed by knowing its position in the MEP → can reduce overhead by packing a lot of fragments in a MEP (O(1000))
- → All other events in the MEP carry a BXID which wraps around every orbit
- → The TELL40 simply relays the BXID received from FE
  - ✓ If it’s wrong at the FE, it is wrong at the TELL40 and will be wrong in the DAQ!
  - ✓ It’s the role of the FE to maintain synchronicity, TELL40 can only monitor and possibly recuperate few clock cycles (16), but not more.
- → In current system its size is reduced in TELL1 packets to cover for the time it sits in the system
  - → A simple calculation shows that keeping it to 64 bits would be advisable (even @ 28 MHz, it needs at least 35 bits..)
- → Only SODIN transmits the full 64 bits Event ID together with all the information of that event (trigger - event - calibration type, UTC timestamp, orbit number, run number, etc).

TFC commands to FE explained

"CALIBRATION TYPE"

→ Used to take data with special trigger pulses (periodic, calibration)
- ✓ Dedicated 4 bits: i.e. 4 different calibration commands possible
- ✓ Dynamic association to be used for calibration and monitoring
  - Absolute need of delays to account for each individual delay in the detectors
- ✓ SODIN overrides internal trigger decision at TELL40
  - Periodic or calibration higher priority, can also have a fast rate, programmable

"NZS MODE"

→ Read out (all) FE channels non-zero suppressed
- ✓ Packing of full set of bits in many consecutive GBT frames: needs buffering

→ Possible to have also multi-NZS readout: *consecutive NZS events*
- ✓ SODIN will take care of sending Header Only for a defined set of clock cycles later to allow recuperating buffer space (programmable as well, to the slowest of the detector)
- ✓ And reject events thereafter to avoid creating bottlenecks

"SNAPSHOT"

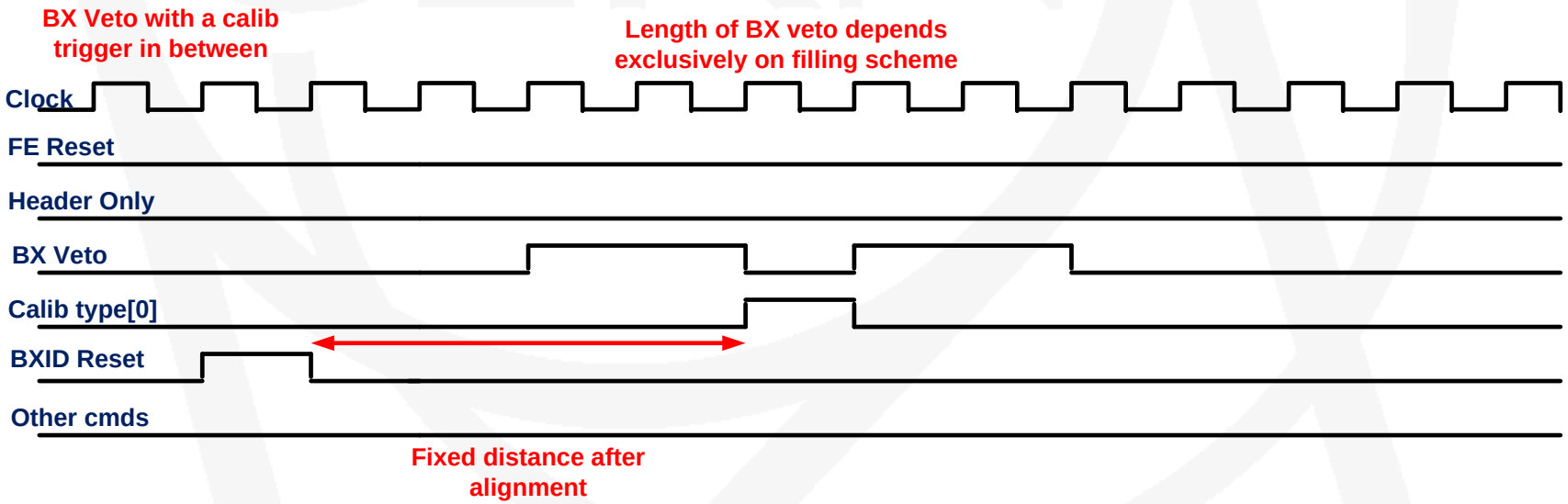
→ Read out all status and counter registers in a "latched" way
- ✓ Latch monitoring registers on snapshot bit, which is set periodically (programmable) and also single shot
- ✓ When snapshot bit is received, send all data via ECS field in TFC

"SYNCH"

→ See after ☺

Data flow control scheme

- TFC commands are synchronous wrt to BXID Reset
  - → once we align BXID Reset with beam, TFC commands come ALWAYS at the same latency
    - (wrt to BXID Reset, hence BXID)!
  - → Compression/suppression logic should act accordingly to TFC command
    - (why would you want to compress/suppress if that crossing is rejected a priori? Especially if your pre-processing is dynamic…)

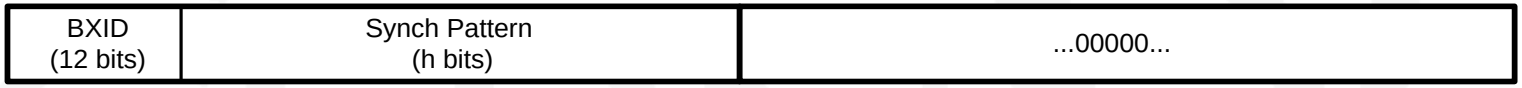


- Data is filtered according to TFC commands and the FE buffer status
- Data is packed onto the GBT link in a continuous fashion

What to do on SYNCH command?

When a Synch command is received

→ replace data packet by a specific Synch Pattern with full BXID (12 bits)



→ Synch Pattern should be programmable via ECS (in length and content)

Synch command is meant to be sure that system is synchronized… in a synchronous way!

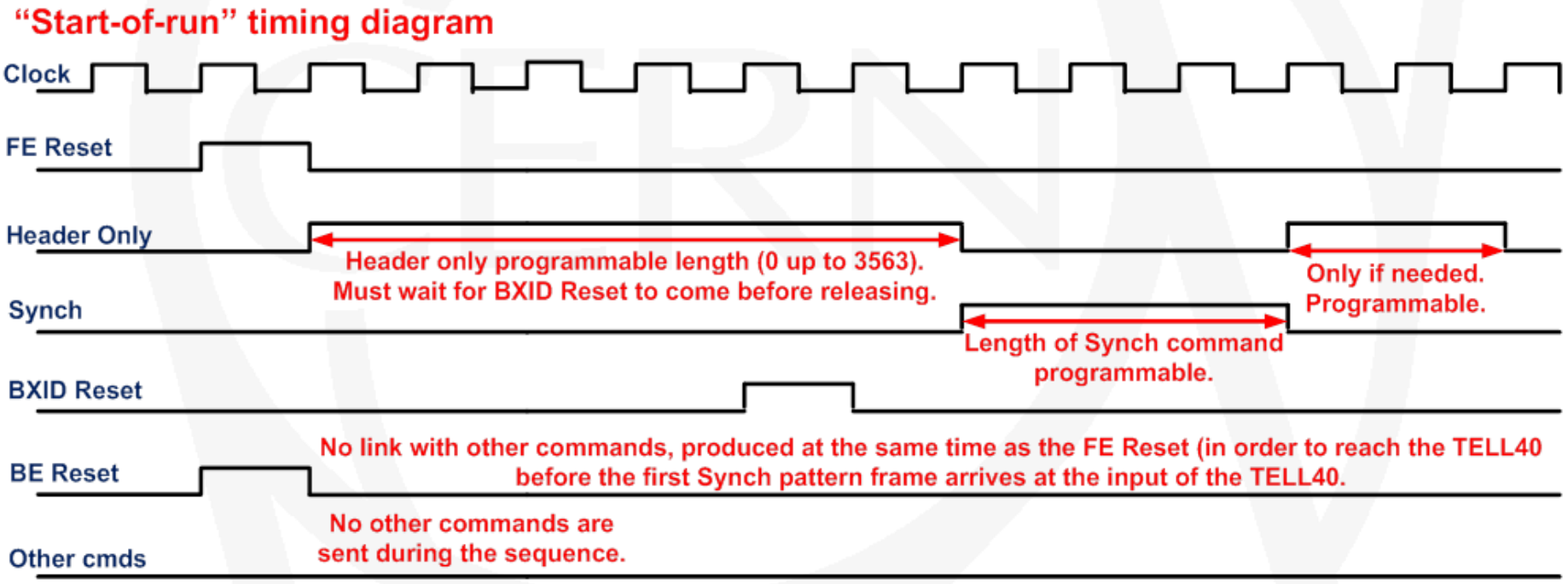
Double usage (in AND or in OR):

1. *Periodically:* i.e., SYNCH command sent every *n* Hz
   → this is intended as a safe synchronicity check!

2. *Asynchronously*: i.e. when a desynch is detected
   → TELL40 detects wrong frames, wrong packing, fast diagnostics in TELL40 specific sub-detectors' codes.
   → makes sense to clear the FE buffer
   → could be sent only for a local sub-detector from SOL40
   - i.e. could be fast triggered either by ECS or by TELL40 via SODIN through SOL40….

→ FEs send Synch Pattern for the same BXIDs everywhere
- TELL40 aligns to corresponding frame and BXID
- FE frees its memory : delete its content, read and write pointers back to empty
- FE sends Synch Pattern
  → TELL40 naturally goes on packing the preceding events in the buffers

TFC *start-of-run* fast sequence

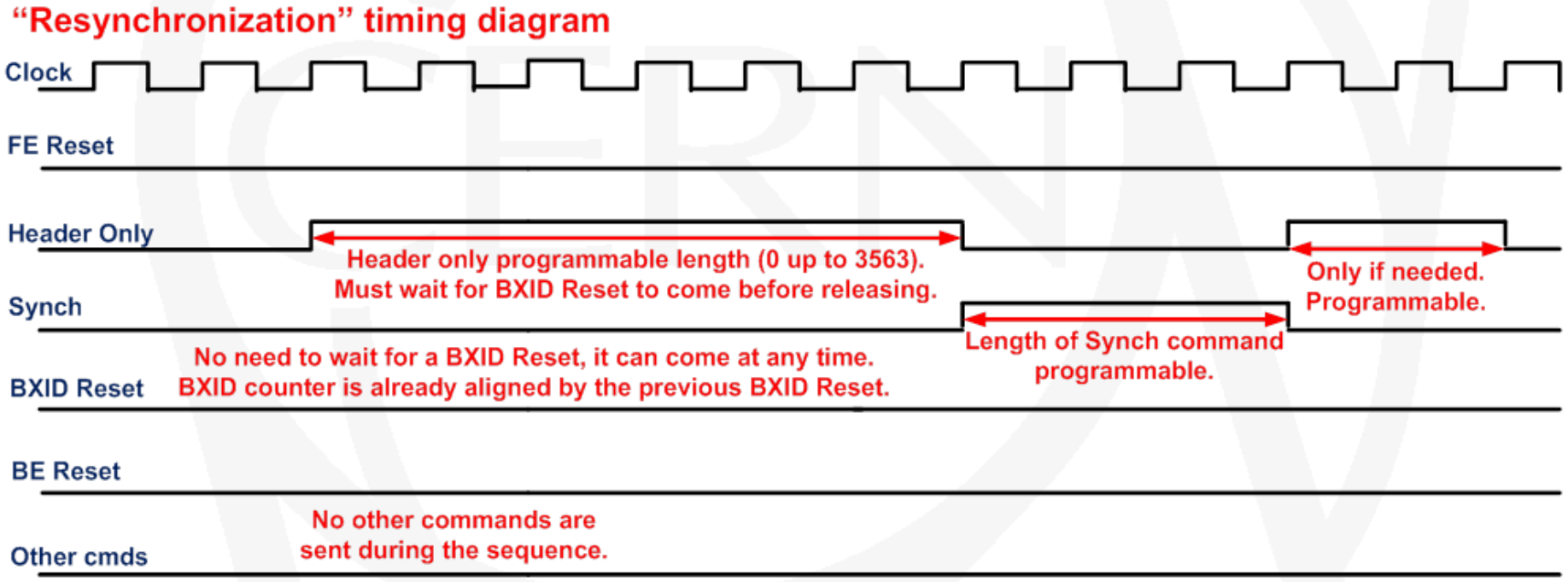


FE Reset is by definition completely asynchronous:

- Can come from control system (ECS) at change Run
- Can be periodic in the firmware
- Can be enabled by another processes (if programmed)

This mechanism will be generated everytime a FE Reset is issued.

Re-synchronization sequence



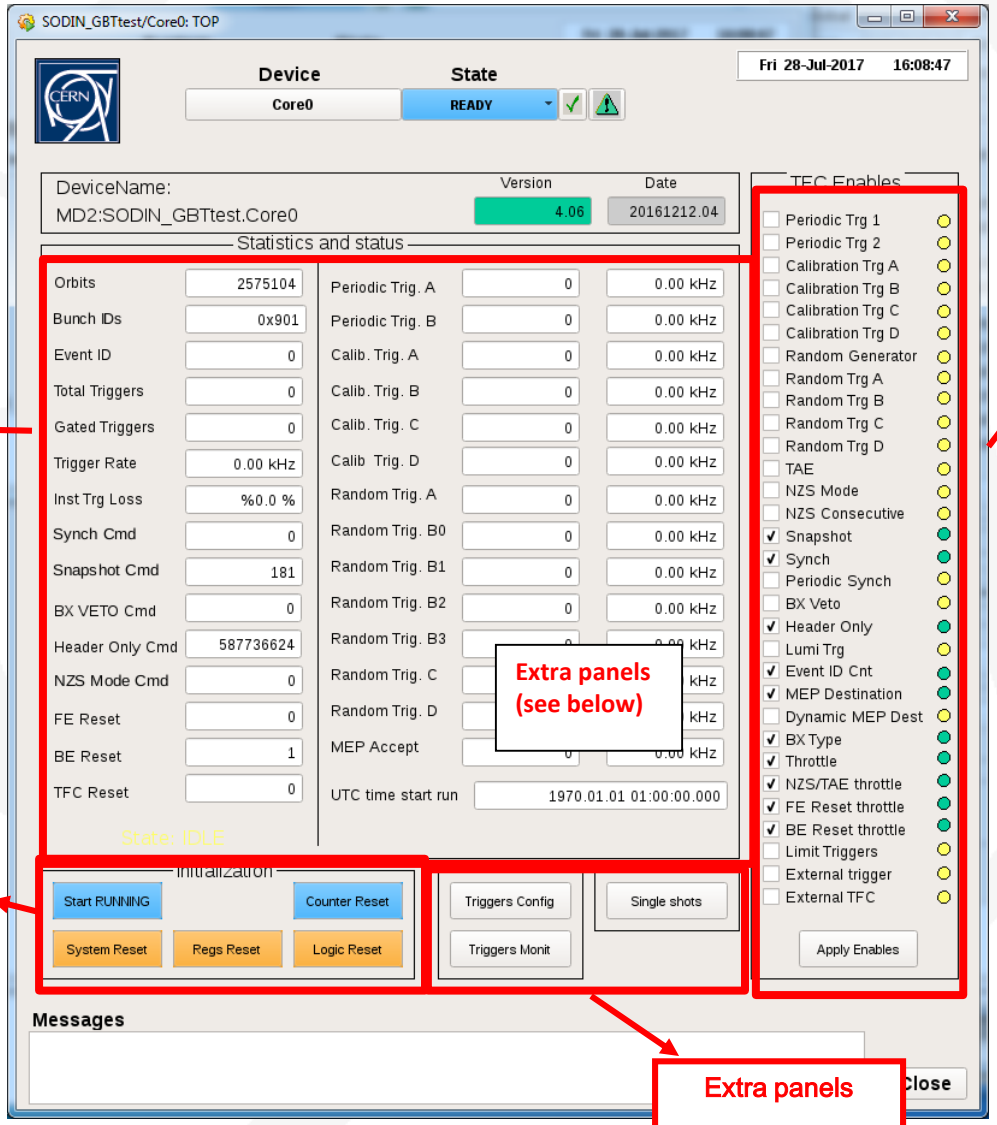
Example of a synch command being requested

→ From ECS or fast via TELL40.

→ Synch command is entirely programmable in frequency and length, while its position can be completely asynchronous or programmable.

→ Header Only is entirely programmable in length.

Controlling the TFC – top panel

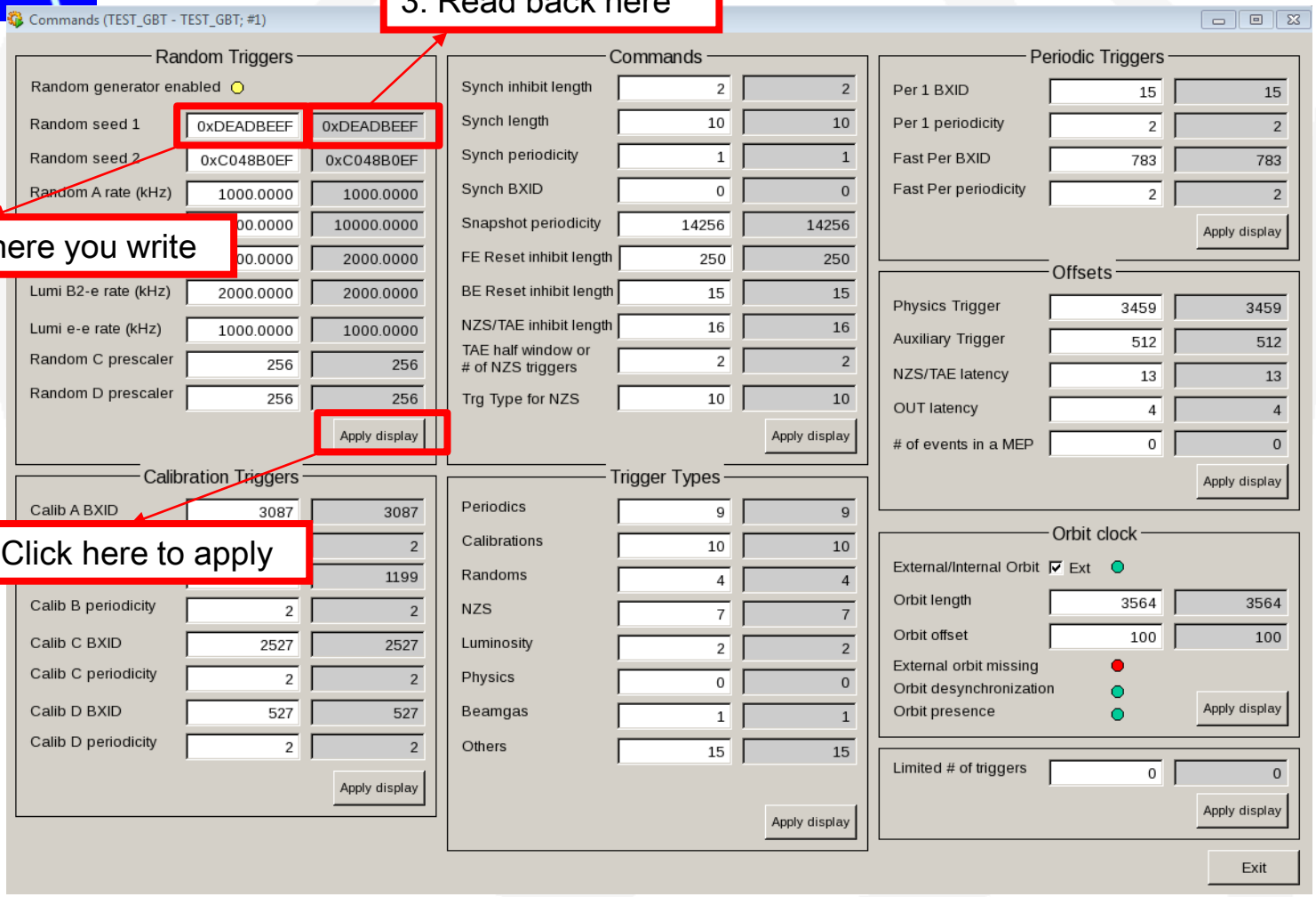


Counters and rates: when you open the panel, make sure that the counter for Orbits and Bunch IDs are counting. These counters should always be counting

Extra functionalities (explained below)

Enables for triggers and commands: tick the enable for whichever set of triggers and commands you would like to have and click on "Apply Enables". After you click the corresponding LED should change color (**yellow = disabled, green = enabled**).

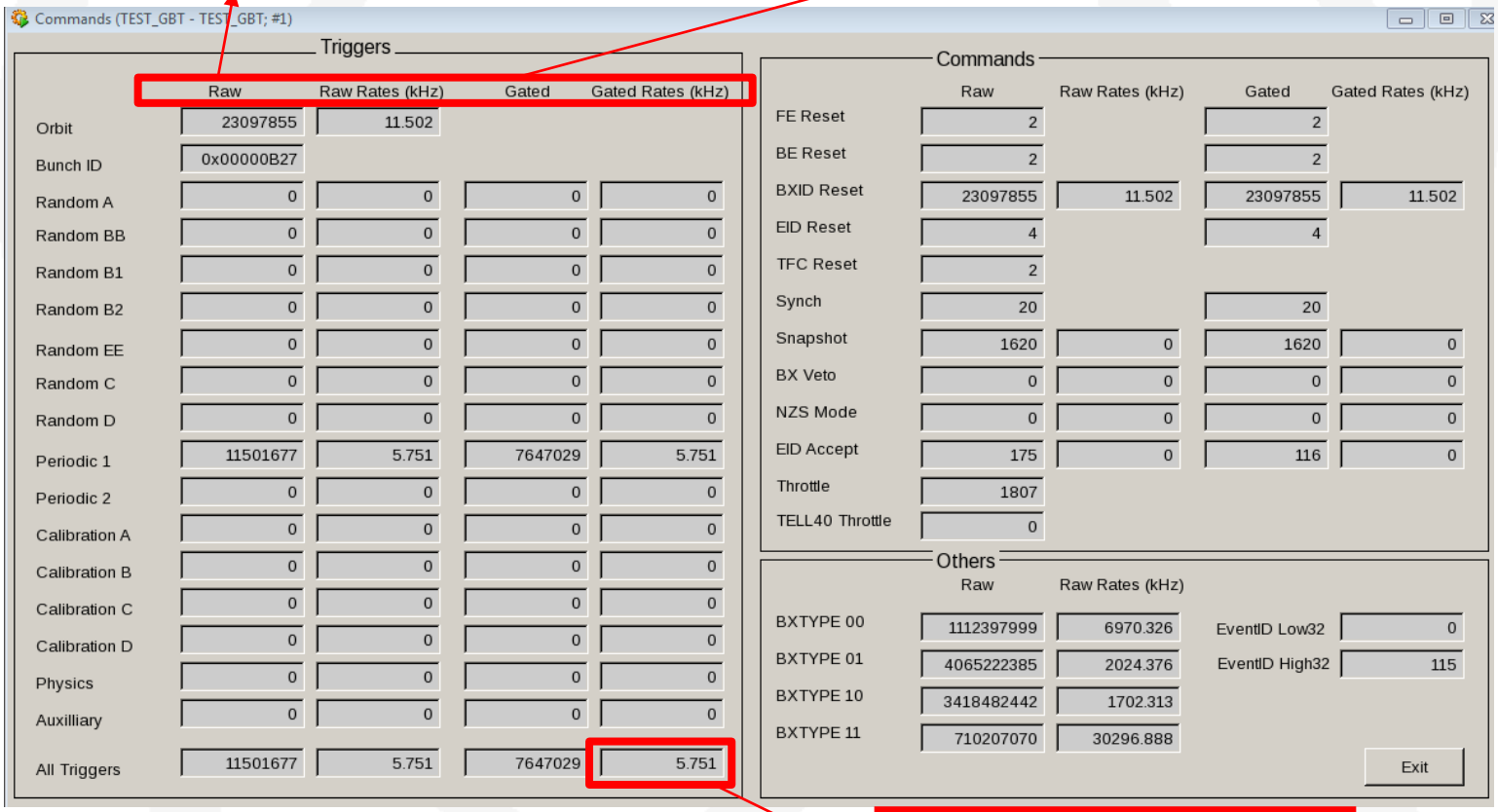
Controlling the TFC



Monitoring the TFC

RAW = internally generated by SODIN by independent processes

GATED = effectively sent by SODIN



Actual trigger rate ....

Why would a trigger generated by SODIN not be sent? Because of priority scheme or throttle!

TFC triggers explained

"PERIODIC" & "CALIBRATION"

→ a trigger every X orbit (periodicity X)

- Difference between periodic and calibration trigger? With the calibration trigger there is a calibration command associated to it. I.e., your FE receives a command associated to that trigger.
- Why? Because we can read out the event as a "calibration event" and the FE can generate a pulse or a calibration procedure. In this way the event the go to a "different stream" of event.

"FAST PERIODIC" & "FAST CALIBRATION"

→ a trigger every X bunch clock (periodicity X)

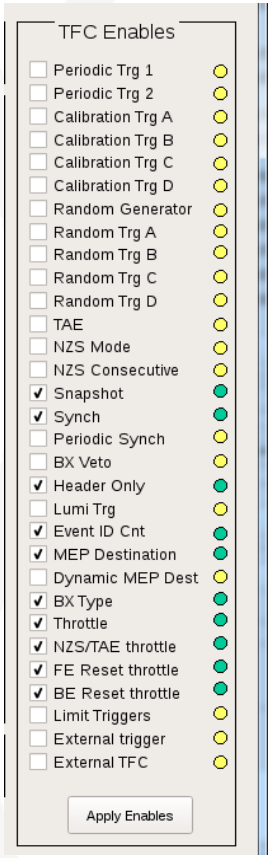
- Same as before, but faster (instead of doing a periodicity every X orbits, the periodicity is eveyr x bunch clocks)

"RANDOM"

→ fully random trigger at XX Hz.

→ random Generator must be active to generate random triggers

- Four different types of Random triggers: A,B,C,D
  - A is fully randomized at the selected rate
  - B is used to generated luminosity triggers (see after).
  - C is a pre-scaled random trigger. I.e., after setting the rate, the C generates a trigger only every X random triggers.
  - D is a random trigger generated only in beam-beam crossing type



TFC triggers explained

"TAE" (Timing alignment Event)

→ a set of consecutive triggers are generated as a window around the central trigger

- Programmable in length (half window) → (half_window) *2 + 1
- If TAE is enabled, every trigger will be enlarged to a TAE event of that type
- TAE maintains the type of the central trigger
  - → If it was generated around a random trigger, ALL trigger in the TAE will be made random.

"NZS" (Non-Zero Suppressed)

→ An NZS command is generated for a specific trigger

- To define for which trigger the NZS command is desired
- In the Trigger config panel according to the tooltip in WinCC (or ask me)

"NZS consecutive" (Non-Zero Suppressed)

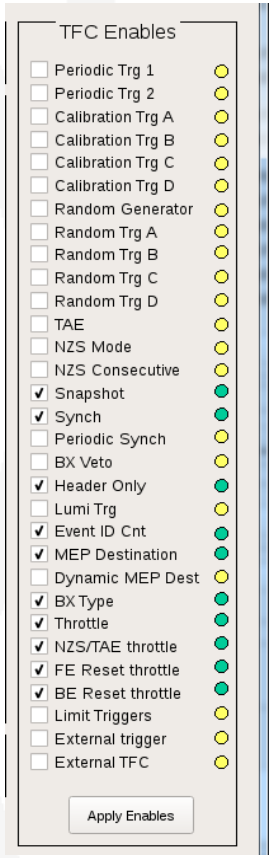
→ Generate a set of consecutive NZS events starting from the initial NZS trigger

- Same half window as used for the TAE → NZS consecutive = half_window + 1

"LIMIT TRIGGERS"

→ Limits the number of triggers to a maximum value

- Programmable number in Trigger config panel



TFC triggers explained

"LUMINOSITY"

→ A random trigger generated in each of the four crossing type
- Beam-beam, beam1-empty, empty-beam2, empty-empty
  - Select individual rates for each crossing type
    - Programmable in Trigger Config
  - Used today to normalize luminosity

"EXTERNAL TRIGGER"

→ Enable SODIN to accept an external trigger
- Usually referred to as auxiliary trigger

"EXTERNAL TFC"

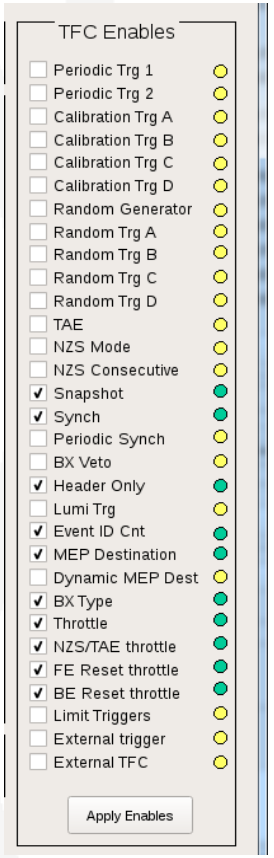
→ In MiniDAQ2, oblige SODIN to take the TFC word from an external SODIN rather than from the core itself
- In case of need of fanning out or for a Master/Slave relationship

"# of events in a MEP"

→ Choses how many events are contained in a MEP

"TRIGGER TYPES"

→ This is a code that identifies what type a trigger was in the HLT farm.
- Not really used in MiniDAQ2, but may become important later...

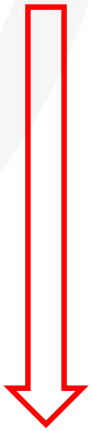


Trigger priority scheme

SODIN can be configured to generate many triggers → they can be concurrent!

To help SODIN decide on which trigger actually select/send, there is a priority scheme applied: the trigger with highest priority gets chosen. In this order:

EXTERNAL TRIGGER (AUXILIARY)
CALIBRATION
PERIODIC
RANDOM TRIGGER C
RANDOM TRIGGER D
RANDOM TRIGGER A
LUMINOSITY (RANDOM TRIGGER B)
BEAMGAS
L0 (in case used with current LHCb...)



Throttles

A trigger may be generated internally by SODIN but throttled away because of another process

→ Throttle in this sense means «rejection»

→ For example: if FE RESET throttle is enabled, this means that no triggers are accepted after a FE RESET for a programmable period (in Trigger Config panel).

- Same for BE RESET throttle and NZS/TAE throttle.

→ Why do you care? Well, if your FE is busy doing something after a FE RESET you probaby don't want to put that data to disk. Or maybe you do... ? You choose and set the corresponding «Inhibit length».

Offsets (or latencies)

SODIN generates commands and words synchronously, i.e. always with the same delay/latency

→ In short, trigger xx is generated at time t0, the corresponding command will always be generated at time t0+offset

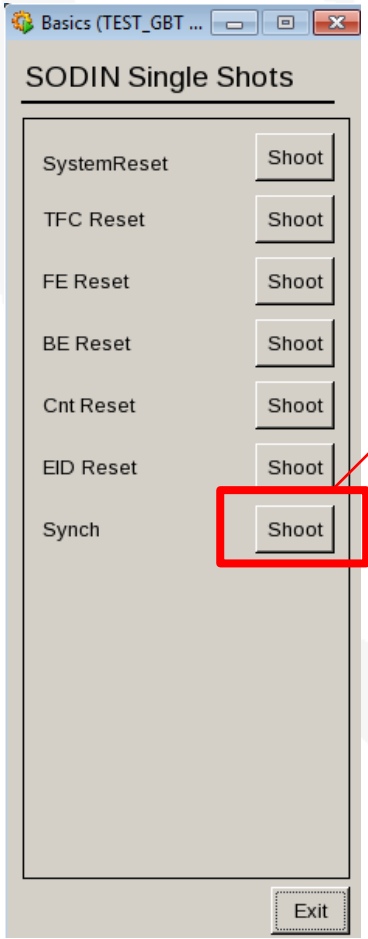
→ Offset is in fact a combination of other various offsets (see Trigger config)

- You can play with the offset to find the perfect latency for your system
- For example: if you are using an external trigger, then you probably want to align this trigger to the corresponding BXID (maybe) and then you want that trigger and specific TFC command comes out after a specific latency so that your FE is ready to send data out.
- It depends case by case, contact me if you need help in figuring out the best latency for your system.

→ The same concept applies to the delays in the SOL40.

→ NB: once you change a delay or an offset, always click on Logic Reset (either in SODIN or in SOL40) to have it effectively executed!!

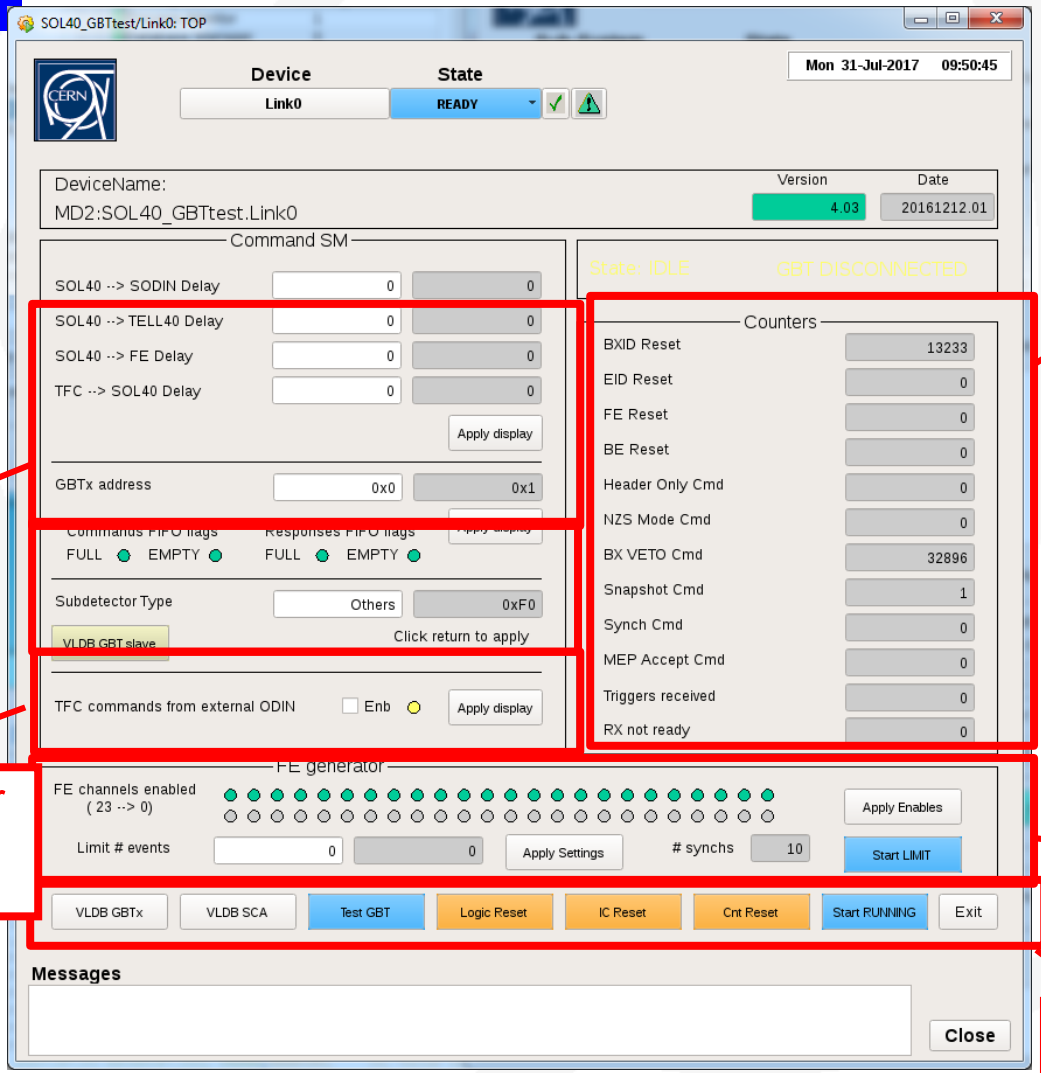
Controlling TFC



Hit it and it will shoot one

NB: if you need a specific procedure, you can implement it in WinCC. Just look behind the button and you can steal the function...

SOL40 panel



SOL40 counters: Check that what is being counted is compatible with what SODIN is generating and what you expect

Delays (see before)

Subdetector Type (see later)

FE controller: Enable/disable a FE channel and controls a limited number of trigger

Extra functionalities

Subdetector type

In the SOL40, a case statement identifies what subdetector is using that specific SOL40.

- The name is associated to a code and the code is set in a register in the SOL40 core
  - Remember? Different encoding for TFC commands and ECS connections to SCAs, GBTs etc...

→ Simply write in the test box the name of your sub-detector
  → VELO, UT, RICH, SCIFI, CALO, MUON, etc...
  → If a specific name/encoding or more than one encoding is needed per sub-detector we can generate a specific name and a specific code (once we have applied the specific encoding in the SOL40).
  → Every sub-detector has currently up to 16 available possibility.

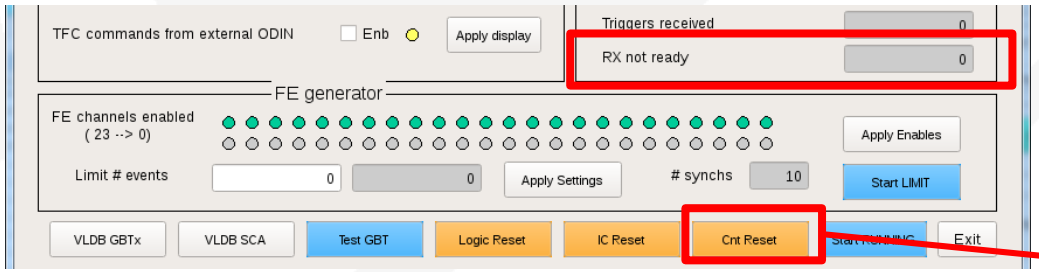
«Test GBT» button or check link sanity

In the SOL40, a button may help you identify if the SOL40 communication to a GBT is working or not

→ Hit the TEST GBT button and look in the Log Viewer in WinCC

- The button generates a read of a specific register of the master GBT (366, the last one) and this register should contain a known patter (0xAA).
- In the Log Viewer in WinCC you should see, the address of the register that has been read, the length of the read (1) and the pattern.
  - → If any of these values is wrong, either the link is not up or the GBT is configured to be talked to via I2C only

- Additionally, a counter in the SOL40 panel shows if the link is «ready» or not
  - → «Rx not ready». If the counter is counting, then the link is no good.



Hit counter reset to make sure it restarts counting

Last word on recipes

Normally, once you found your «golden» recipe, the TFC should be configured always in the same way.

→ This is normally handled by WinCC but we don't have that in place yet

→ You can find the SODIN and SOL40 configuration file here
- → <wincc_proj_path>/fwComponents_<nameofproject>/data
  - STFC_configSODIN.cfg
  - STFC_configSOL40.cfg
- → These two files can be edited by hand (ask me) or generated automatically through the simulation in modelsim

Still alive?
Any (more) questions?

A word on rate regulation

- ✓ Throttle mechanism is meant to «pull the brake» if driving too fast
  - → Simple idea: veto events by putting Trigger bit sent to TELL40 to 0 synchronously across all TELL40 for the same BXID (or same EvID)

- ✓ Today, throttle is completely asynchronous and effectively rejects «L0yes» decisions from hardware trigger
  - → It is applied as long as a TELL1 does not release the throttle
  - → The event is rejected at the FE (for all FEs) which simply do not store fragments of that particular BXID in the derandomizer buffer

- ✓ In the upgrade scenario, events fragments have already been recorded and sent to the TELL40 asynchronously
  - → TELL40 must not «lag» behind otherwise its buffer will get full
    - Its input stage must be able to cope with a fully trigger-less, throttle-less readout system
    - But if the TELL40 has trouble (see next), a *throttle* must be applied
    - See next slides

On the re-synchronization sequence ...

The Synch command is of outmost importance. Without it the TELL40 cannot decode your input data format!

Take good care in where you implement your Synch command logic!

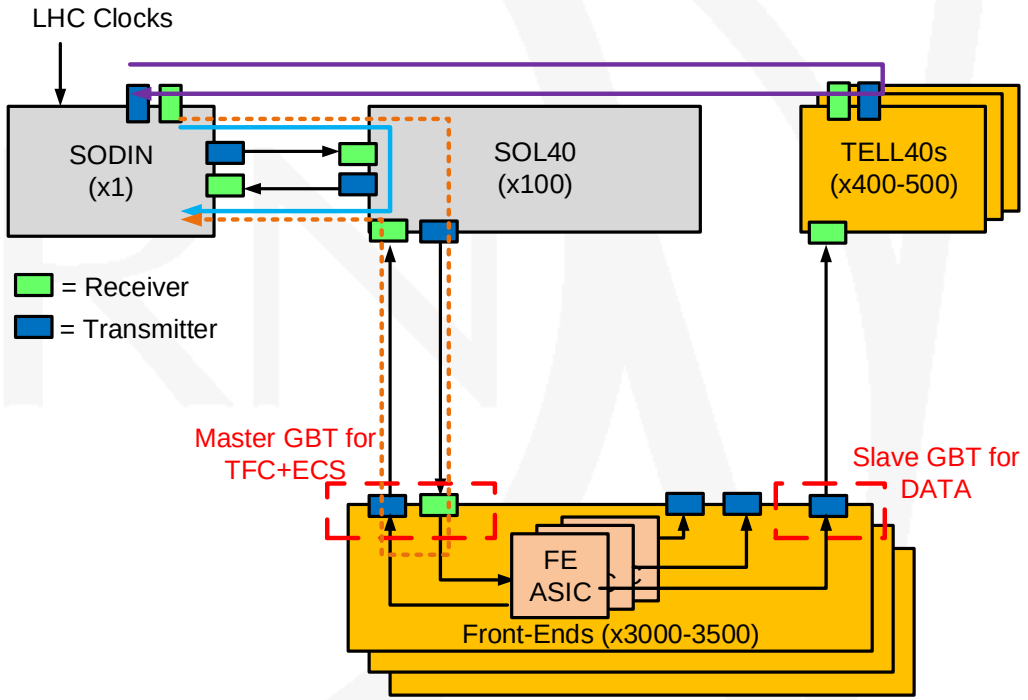
- If you implement it at the output of your buffer, you must clear the buffer, reset the read/write pointers and then start from a clean sheet (as of specs).
  - ✓ In this case the TELL40 must truncate all other events in between (it will anyway as it doesn't receive the deleted events).
  - ✓ TFC can minimize the losses by sending Header Only to FE, reject events with trigger at TELL40 before a Synch, in any case if this is the implementation than we have necessarily a loss of data if the Synch command is transmitted periodically.
  - ✓ On the other hand, the rate can be pretty low (O(10 Hz)), so the loss can be minimal with the payback of having a system which is regularly re-synchronized.
  - ✓ However: how are you going to ensure that the synch pattern is for the correct BXID if you apply it at the output of your FE buffer?

- If you implement it at the input of your buffer, then a regular synch command can help fast diagnose de-synchronization of the data link.
  - ✓ Data will be consecutive and the synch frame can just be a cross-check mechanism.
  - ✓ This feature would be desirable especially if you are sending a shortened BXID.

Do not forget the synchronicity checks

In the specs and at the reviews we asked you to implement few features to check synchronicity and latency/cable delays/sources

→ See the paths in the picture



There should be means to loopback input data to each link:

1. Loop back TFC downlink data onto the TFC uplink
   - ✓ Desirable if this goes through the FE ASIC/FPGA
   - ✓ If not, GBT has loopback capabilities
2. Loop back TFC downlink data onto the DATA uplink
   - ✓ Check for all cable delays
3. Same mechanisms to eb put in place between SODIN-SOL40 and SODIN-TELL40