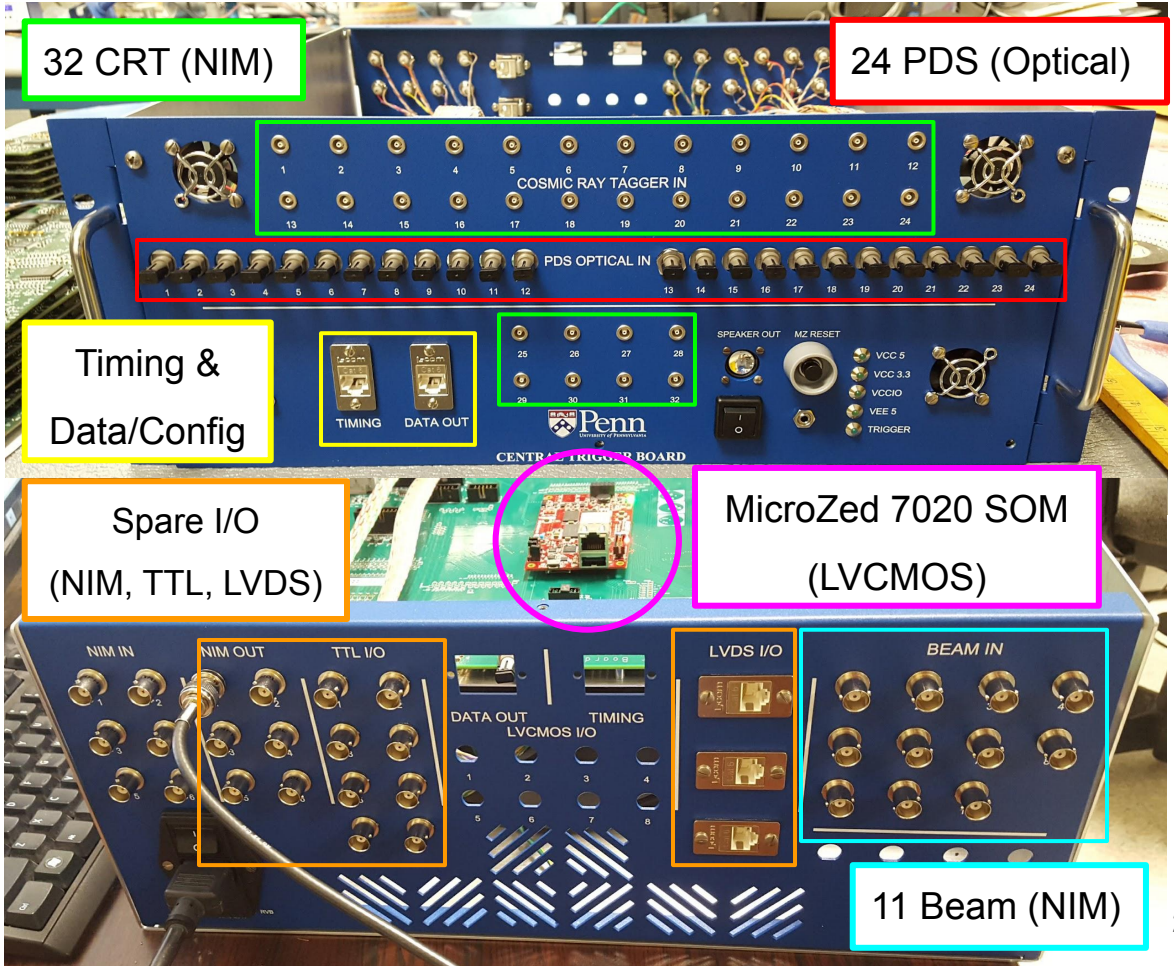
NP04 CTB Update

Ben Harris, James Shen,
Jon Sensenig (in spirit)
University of Pennsylvania
Jun 26, 2024

Central Trigger Board

- Interface between detector subsystems and a MicroZed 7020 SOM
  - Trigger logic
  - Data transmission

System	# I/O	Input Signals
CRT IN	1 - 32	CRT 1 - 32
PDS IN	1 - 24	PDS 1 - 24
BEAM IN	1 - 8	Beam Trig, Monitors, Cherenkov
Spare TTL	3	Beam Gate

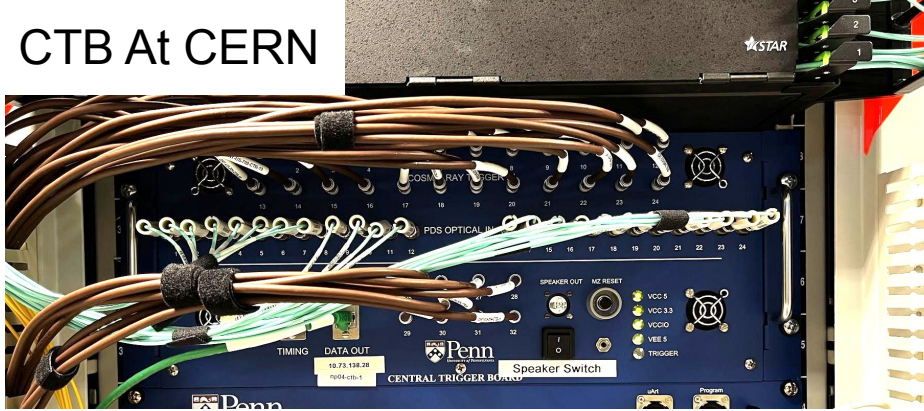


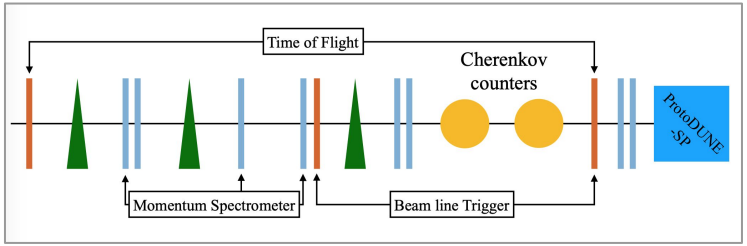
Central Trigger Board

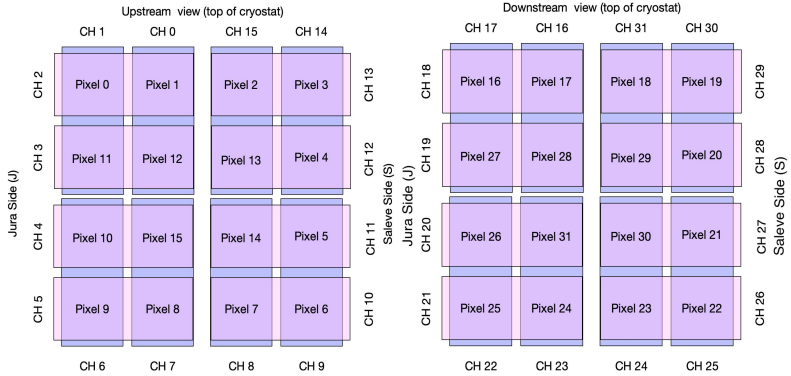
RUN 1:

- Form triggers based on:
  - Beam Instrumentation
  - Cosmic Ray Tagger (CRT)
  - Photon Detection System (PDS)
- Triggers sent to and then distributed by the timing system

CTB At CERN





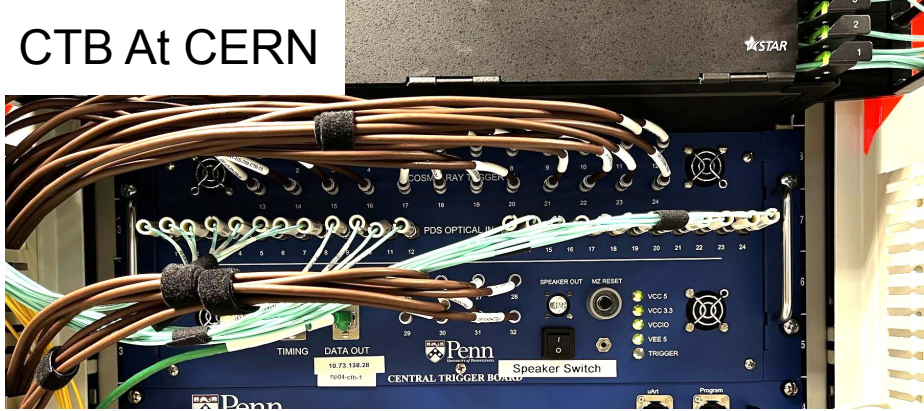


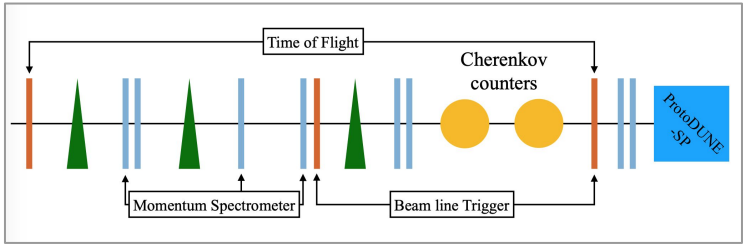
Central Trigger Board

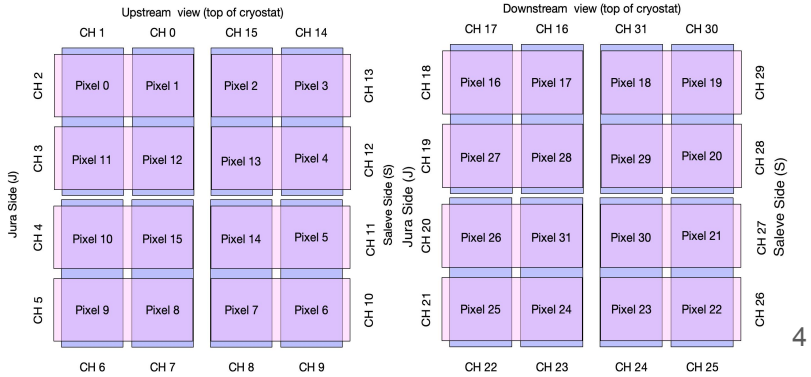
RUN 2:

- Form triggers based on:
  - Beam Instrumentation
  - Cosmic Ray Tagger (CRT)
  - ~~Photon Detection System (PDS)~~
- Act as a Hardware Signal Interface (HSI) for the software trigger.
- Trigger words sent to DUNE-DAQ readout (ctbmodule) over the network.

CTB At CERN



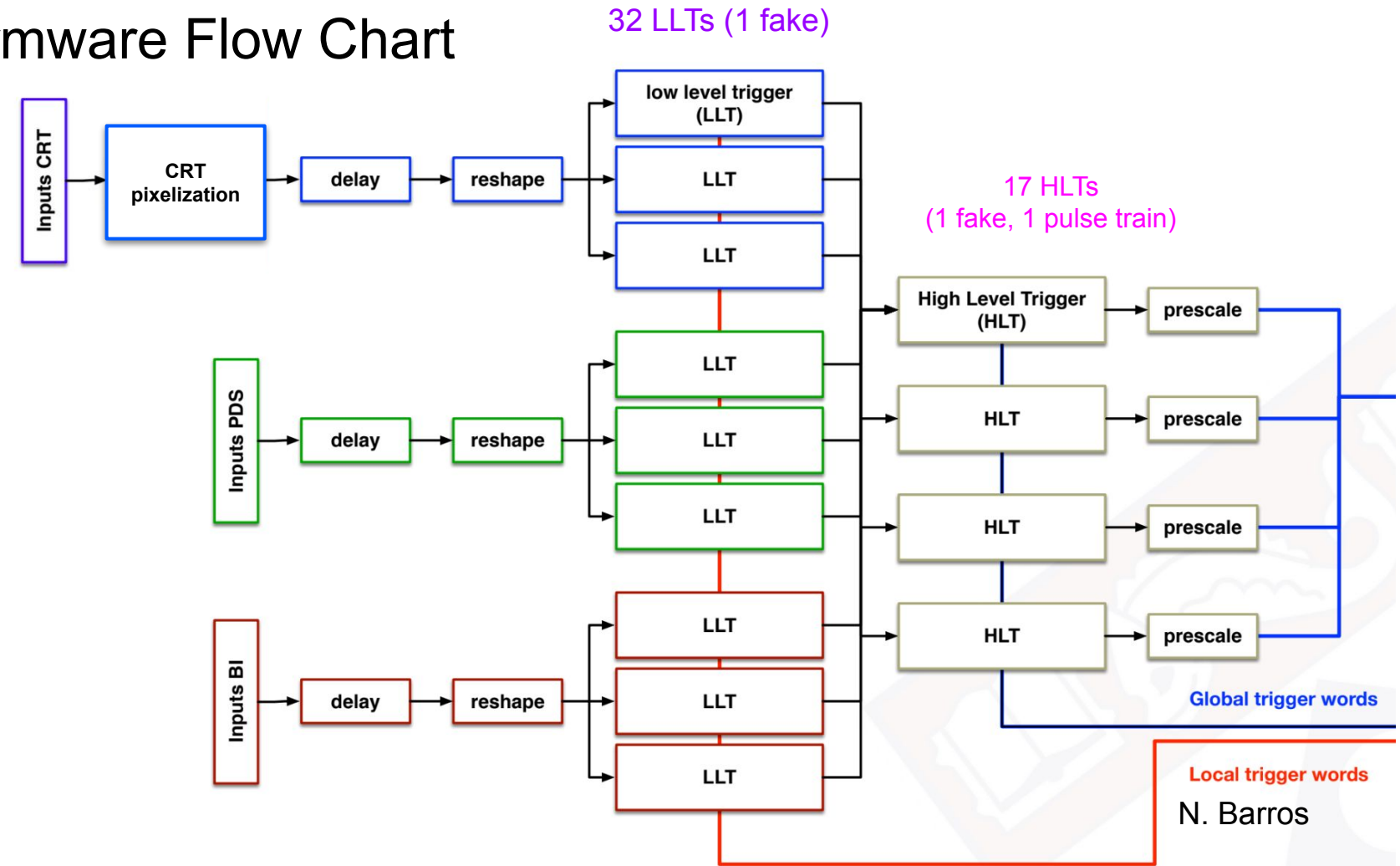




Central Trigger Board

- MicroZed 7020 with Zynq 7000 SoC: FPGA + Arm CPU
- FPGA performs synchronous trigger logic
  - **Masking, delaying,** and **reshaping** of inputs from detector subsystems
  - 32 **Low level triggers** (LLTs) perform logic on inputs from specific detector subsystems
  - 17 **High level triggers** (HLTs) perform logic on LLTs, and directly map to **trigger candidates**
- CPU performs configuration and data transmission
  - Running driver daemon on Ubuntu 16.04 (embedded)
  - Configures the FPGA based on an internal json configuration from DUNE-DAQ ctbmodule
  - Sends **LLT trigger words**, **HLT trigger words** and **channel status words** back to DUNE-DAQ ctbmodule
- Also have ability to do standalone runs where we directly talk to the driver without going through DUNE-DAQ

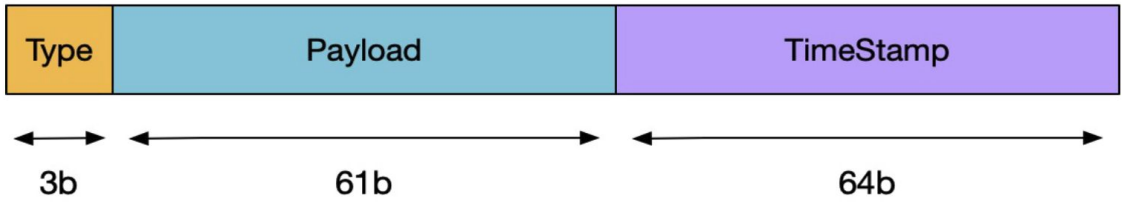
Firmware Flow Chart



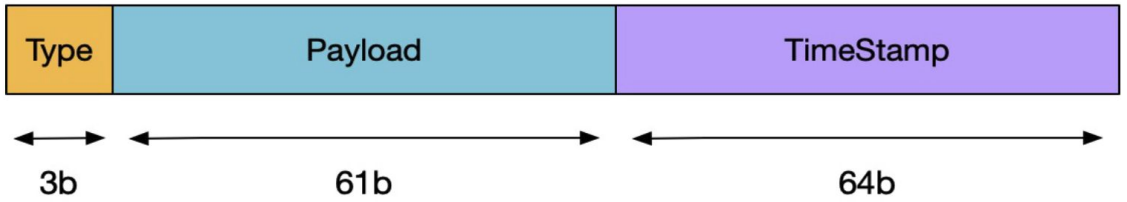
CTB Firmware Data Format

- Respective words created every time an LLT or HLT asserts.
- Channel status words created for every LLT and HLT word.
  - Used to match with and double check triggers offline
- Also issue periodic timestamp words (heartbeats) and feedback words (when there is an error)

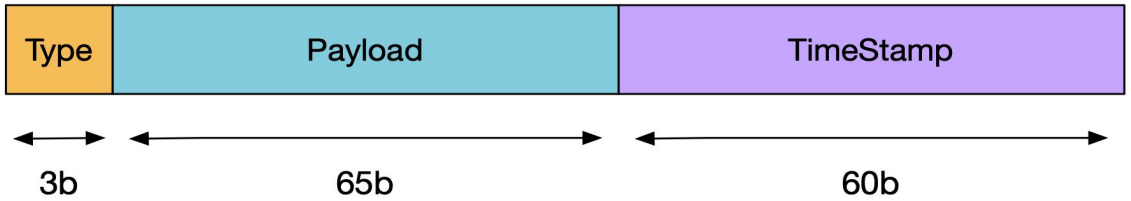
High Level Trigger (HLT)



Low Level Trigger (LLT)



Channel Status (the inputs)



Firmware Flow Summary

- Inputs can be delayed and reshaped
- Trigger words issued when corresponding input conditions are met
- LLTs are **state-like**: assert for as long as the input conditions are met
  - Have mask inclusions and counters
  - LLT trigger word only sent out when it first asserts
- HLTs are **pulse-like**: assert for one clock tick after LLT conditions are met
  - Have mask inclusions, mask exclusions, and prescales
  - Can only reassert once input conditions are not met for at least a clock tick
- Also send out channel status words for every trigger word, periodic timestamp words, and feedback words for debugging

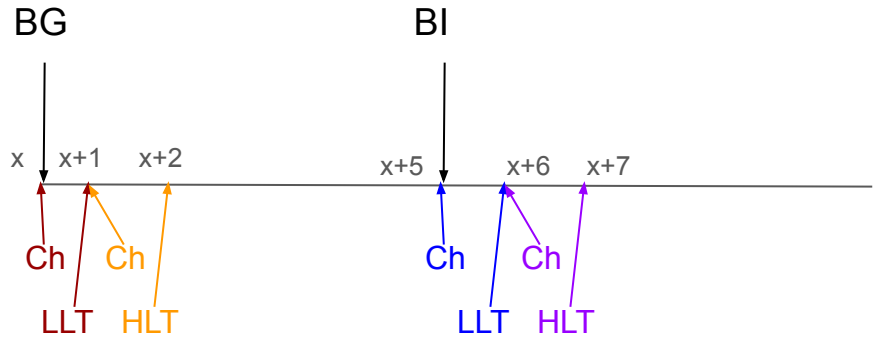
Toy beam event and trigger issuing

- LLT1: Beam instrument tigger (BI), LLT4: Beam Gate (BG).
- HLT1: LLT1 AND LLT4. HLT13: LLT4

Assume beam gate is on at ts=x (and stays on). Assume BI Fires at x+5

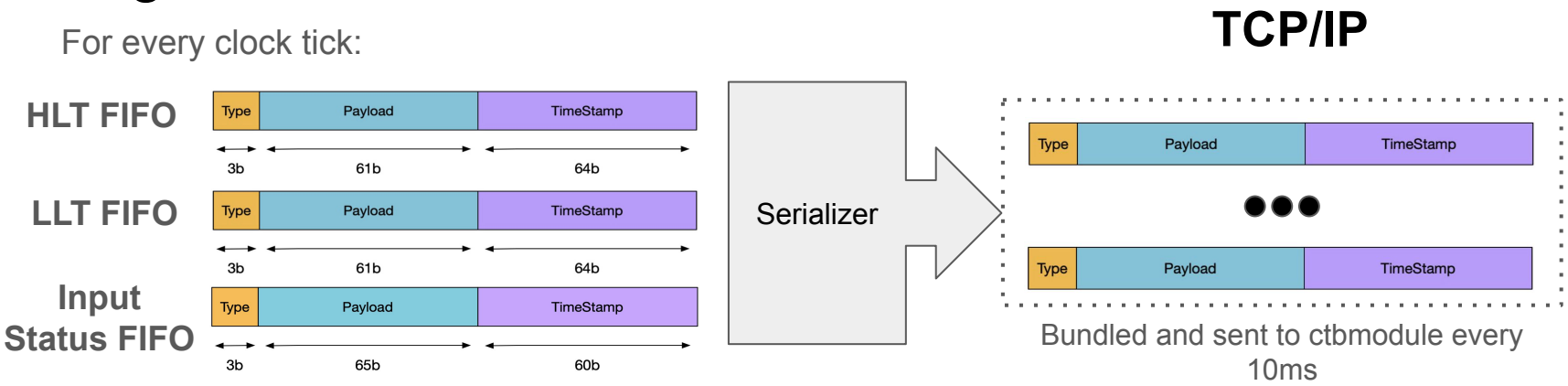
Words generated:

- ts=x, channel status word (BG)
- ts=x+1, LLT trigger word (LLT4)
- ts=x+1, channel status word (BG)
- ts=x+2, HLT trigger word (HLT13)
- ts=x+5, channel status word (BI + BG)
- ts=x+6, LLT trigger word (LLT1+LLT4)
- ts=x+6, channel status word (BI + BG)
- ts=x+7, HLT trigger word (HLT1)



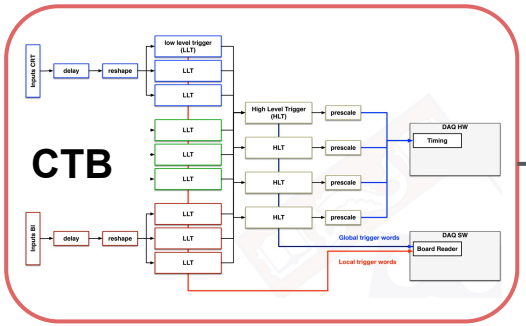
Word generation and serialization

For every clock tick:



- Serialization is done to create a software-friendly interface.
- The serializer prioritizes fifos: HLT -> LLT -> Channel Status. This guarantees that HLTs are always sent out as soon as they are issued.
- Trigger words are delayed by 1 tick compared to their channel status word.
- HLTs are issued one tick after LLT conditions are met.
- We assume all words are time-ordered, but lower-priority words can actually arrive after higher priority words when trigger words are issued at a high rate (matching errors!).
  - *Planned fix is to add a look-ahead for no-match trigger words.*

Data Path



CTB Module

IF **HLT**

HSI Raw Data

HSI Event Data

To Data Selection as TCs

IF **LLT**

IF **HLT**

LLT Latency Buffer

HLT Latency Buffer

HSI: Hardware Signal Interface

HSI **Raw** Data Format (Saved in Data stream)

- HSI data format shared by timing and CTB.
- Format reflects input → trigger processing step.
  - e.g. input channel → LLT or LLT → HLT
  - Two HSI Fragments, one for LLT and one for HLT.

```
typedef uint32_t word_t;

word_t version : 6, detector_id : 6, crate : 10, slot : 4, link : 0;

word_t timestamp_low : 32;

word_t timestamp_high : 32;

word_t raw_input_low : 32;

word_t raw_input_high : 32;

word_t trigger : 32;

word_t sequence : 32;
```

Link=1 for HLT
Link=0 for LLT

Why are we triggering?

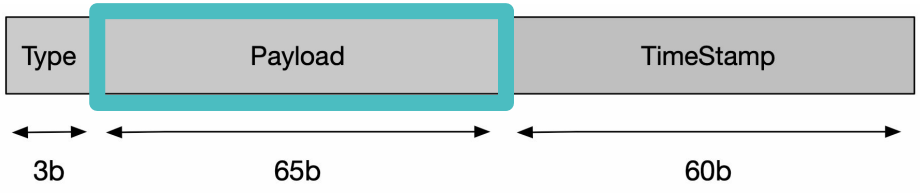
Types of trigger emitted

HSI **Raw** Data: LLT

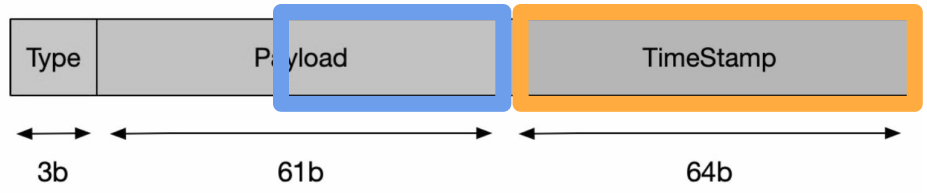
1.

Virtual Link: LLT = 0

Channel Status Word 2.



Low Level Trigger Word 3. 4.




```
word_t version : 6,
       detector_id : 6,
       crate : 10,
       slot : 4,
       link : 0;
word_t timestamp_low : 32;
word_t timestamp_high : 32;
word_t raw_input_low : 32;
word_t raw_input_high : 32;
word_t trigger : 32;
word_t sequence : 32;
```

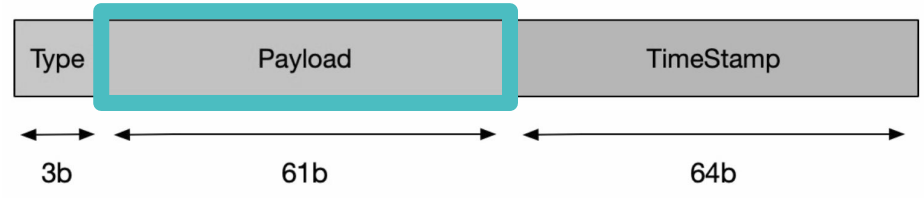
1. 4. 2. 3.

HSI **Raw** Data: HLT

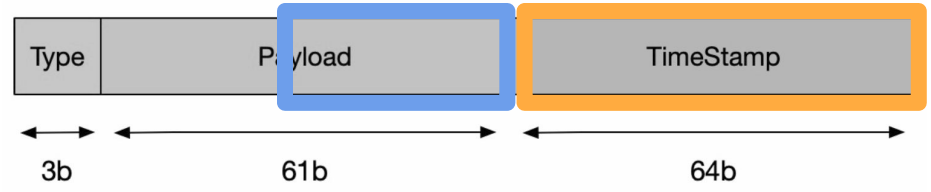
1.

Virtual Link: HLT = 1

Low Level Trigger Word 2.



High Level Trigger Word 3. 4.




```
word_t version : 6,
       detector_id : 6,
       crate : 10,
       slot : 4,
       link : 1;
word_t timestamp_low : 32;
word_t timestamp_high : 32;
word_t raw_input_low : 32;
word_t raw_input_high : 32;
word_t trigger : 32;
word_t sequence : 32;
```

1. 4. 2. 3.

HSI **Event** Data Format

- HSI trigger passed to data selection (shared between timing and CTB).
- Shared between timing and CTB.
- `signal_map` bitmap of the triggers.
- `timestamp` for CTB HSI is from the causal HLT.

```
uint32_t header{ 0 }; // Header field

uint32_t signal_map{ 0 }; // Bit map of signals. 1 bit, 1 signal

daqdataformats::timestamp_t timestamp{dfmessages::TypeDefaults::s_invalid_timestamp};

uint32_t sequence_counter{ 0 }; // Event sequence number

run_number_t run_number{ 0 };
```

HLT -> TC Maker

- Each HLT corresponds to a TC Word in the software trigger:
- HLT 0: Fake fixed-frequency trigger
- HLT 1-9: Beam triggers with fully enumerated PID selection using the Cherenkov detectors
  - E.g. **kCTBBeamChkvHL**: Beam trigger, requiring coincidence with both Cherenkov detectors.
  - **kCTBBeamChkvLx:** Beam trigger, vetoed by low pressure Cherenkov detector.
  - *This was silly of us… there are only 4 mutually exclusive Cherenkov states, and 16 possible trigger conditions. We will only emit words HL, HxL, HLx, and HxLx. Other words will be recycled for other purposes in the next patch release.*
- HLT 10-12: CRT Triggers:
  - HLT 10: Offspill cosmics
  - HLT 11: Offspill cosmics, Jura (beam) side only
  - HLT 12: all cosmics
- HLT 13-16: Custom Triggers:
  - Customizable triggers for any other scenarios. E.g. PDS tests with specific CRT panels, diagnostic runs, etc.
  - HLT 16 is connected to a signal repeater, emitting 10 HLTs at 83Hz.
- Full mapping at: https://twiki.cern.ch/twiki/bin/view/CENF/TriggerBasicOp
- Documentation on all existing configurations: https://github.com/pennneutrinos/protodune-ctb-configurations (private repo. Ask us for access!)

Integration with software trigger

- Multiple HLTs can be emitted with the same timestamp and within the same event window. One HSI Frame is emitted for each LLT/HLT word.
- Trigger Decision can be made based on specific TC words, but *all* TC words emitted during readout will be saved in the TC and HSI Fragments.
- ***Prescaling*** can currently be done either on the CTB or via the software trigger. *However, sw trigger* can only apply a global pre-scale on all CTB-emitted triggers, not per TC Type.
    - *Test beam run uses CTB Pre-scale.* Pre-scaling at the TC Level can be beneficial in vetoing pile-ups off-line. Maybe?