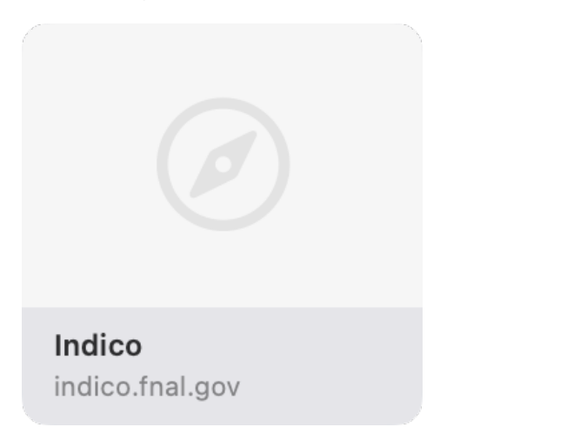
#ldmx

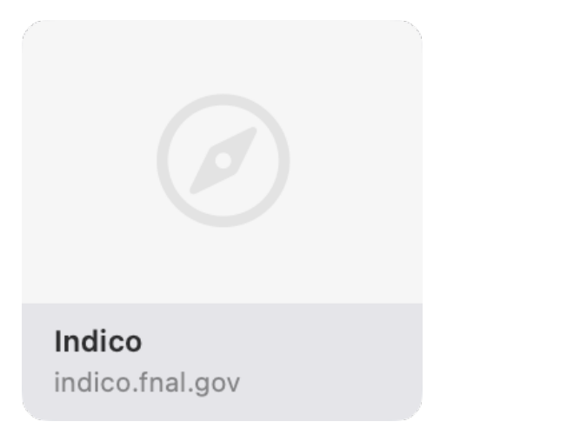
Backplane-less readout for HGCROC board quality control

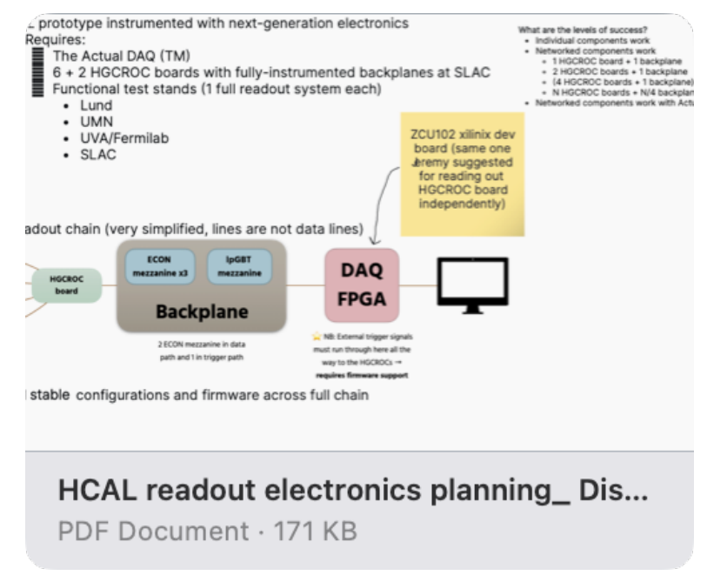
ESA: All roughly the same configuration (e beam with set energy)

HCAL readout electronics timeline and work planning, March 7


Indico
indico.fnal.gov

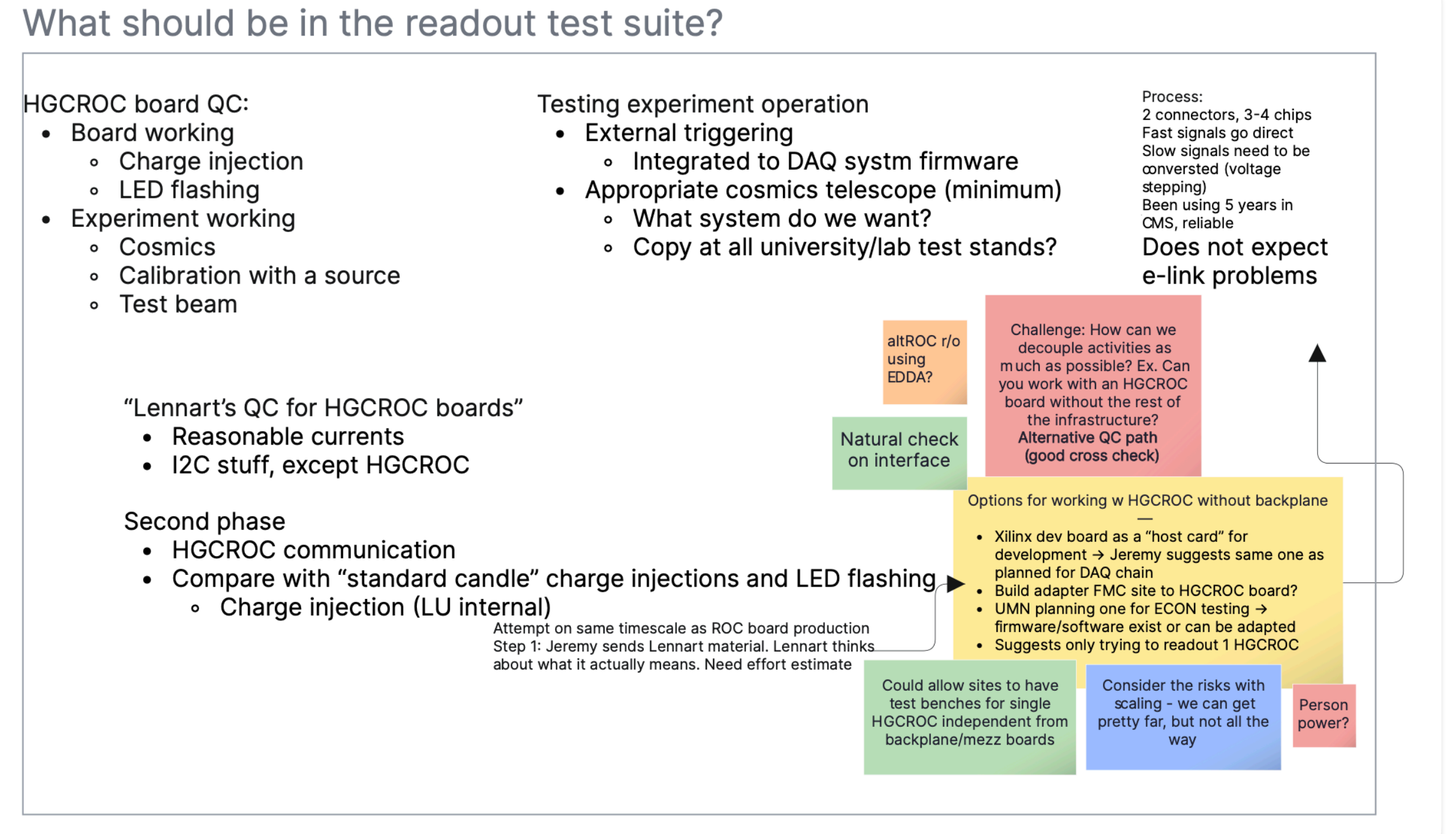
Design review, March 21


Indico
indico.fnal.gov


HCAL readout electronics planning_ Dis...
PDF Document · 171 KB

What firmware blocks is UMN/ Jeremy prepared/comfortable supporting? If somebody else wants to maintain the firmware from CMS, we need permission to maintain it. If we use the CMS stackup, we need to support it ourselves.

What should be in the readout test suite?



HGCROC board QC:
- Board working
  - Charge injection
  - LED flashing
- Experiment working
  - Cosmics
  - Calibration with a source
  - Test beam

Testing experiment operation
- External triggering
  - Integrated to DAQ systm firmware
- Appropriate cosmics telescope (minimum)
  - What system do we want?
  - Copy at all university/lab test stands?

Process:
2 connectors, 3-4 chips
Fast signals go direct
Slow signals need to be conversted (voltage stepping)
Been using 5 years in CMS, reliable
Does not expect e-link problems

"Lennart's QC for HGCROC boards"
- Reasonable currents
- I2C stuff, except HGCROC

Second phase
- HGCROC communication
- Compare with "standard candle" charge injections and LED flashing
  - Charge injection (LU internal)

Attempt on same timescale as ROC board production
Step 1: Jeremy sends Lennart material. Lennart thinks about what it actually means. Need effort estimate

altROC r/o using EDDA?

Natural check on interface

Challenge: How can we decouple activities as much as possible? Ex. Can you work with an HGCROC board without the rest of the infrastructure? **Alternative QC path (good cross check)**

Options for working w HGCROC without backplane
—
- Xilinx dev board as a "host card" for development → Jeremy suggests same one as planned for DAQ chain
- Build adapter FMC site to HGCROC board?
- UMN planning one for ECON testing → firmware/software exist or can be adapted
- Suggests only trying to readout 1 HGCROC

Could allow sites to have test benches for single HGCROC independent from backplane/mezz boards

Consider the risks with scaling - we can get pretty far, but not all the way

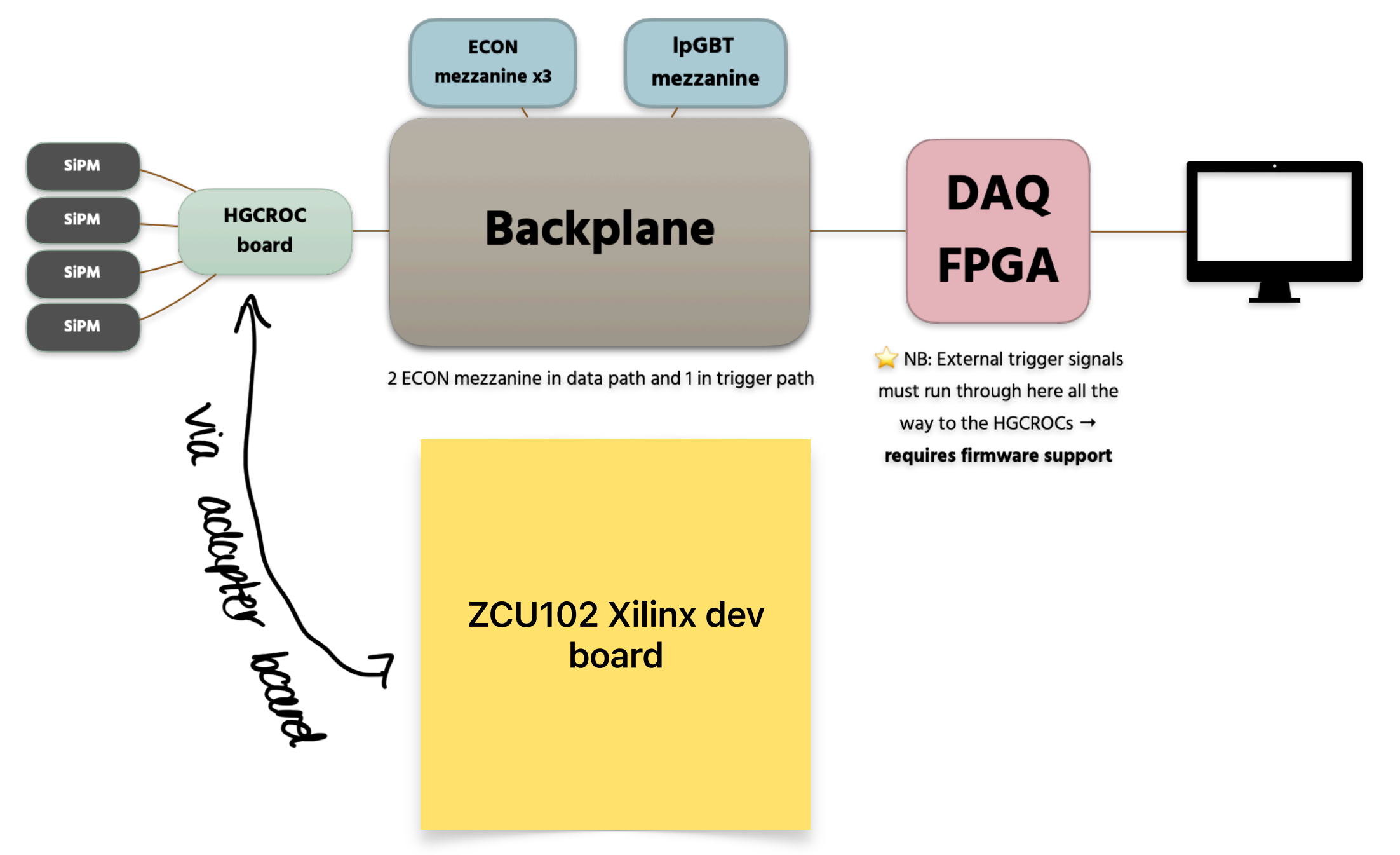
Person power?

9 October 2024: https://indico.fnal.gov/event/66391/
Erik, Lene Kristian, Lennart, Cristina, Jeremy, Hannah

HGCROC board manufacturing
In production at CERN. Torsten will go to the workshop and talk with them 10 October.
4 boards in production first and then 16 more (total 20) [HGCROC board + adapter card]
Planning to mail out 18 October (subject to confirmation tomorrow)
- 2 at Lund (1 → Fermilab?)
- 1 UMN
- 1 UVA (or 2 to UMN and 1 at LU)
(First round)

Firmware stackup:
- Leverage CMS readout as much as possible in the short term to readout HGCROC board directly with Xilinx dev board
- Worry about SURF compliance next
- If we use the CMS path, we have all the blocks we need:
  - Fast control
  - I2C
    - Standard IPs from Xilinx
  - Data capture side / Link capture [needs work]
    - CMS implementation complex...
    - Options available
    - Not super complicated to hit a reasonable data rate
      - 100 Hz to kHz achievable
    - Rate-dependent; we'd need to update this later
      - **TODO: Work out a data rate progression plan for after first QC**
- Workload if board + adapter card in-hand (and no powering-type issues) → few days for low-data rate (100Hz) firmware

We'd previously identified that the backplane-less readout path for reading out a single HGCROC needed:
- ZCU development board (Xilinx FPGA)
- An adapter to connect the development board and HGCROC board (FMC site)
- HGCROC board 😎
- Firmware for the development board to read directly the HGCROC board
- HGCROC configurations appropriate for HGCROCv3b



ECON mezzanine x3
lpGBT mezzanine
SiPM
SiPM
SiPM
SiPM
HGCROC board
Backplane
DAQ FPGA
2 ECON mezzanine in data path and 1 in trigger path
⭐ NB: External trigger signals must run through here all the way to the HGCROCs → requires firmware support
via adapter board
ZCU102 Xilinx dev board

Quickly becomes a software question...
- CMS: Microhall (similar to Rogue)
  - Built-in protection layer
- Testbeam 2022: We had our own custom software
  - Pflib/pftool was C++ tool for configuration/setting registers
  - Not a plug-and-play solution to recycle this...
  - Could have a mismatch between how these libraries were intended and what they are supposed to do now...
  - pftool interface sensible; try to keep
  - pflib needs to be rewritten
    - Based on information passing to/from the polarfire, which doesn't work like that
    - We don't want to build polarfire emulator
    - ZCU also stronger processor than polarfire's was :) we can give it some work
- Rogue: Python-Rogue good for configuration
  - Slow through Program io
  - DMA required if we want to go fast
  - Harder on the firmware side
    - At that point, makes more sense to flop over to Stanford firmware suite
  - Could get decent performance with Program io (but have to dump Python)
- **TODO: Consider where we want to invest our sw dev time**
  What's a 1-time deal?
  Creating an appropriate bit/byte stream for the ROCs definitely lives; probably good to develop expertise outside UMN
  Definitely need a new register map
  Then send your register map to the ROC and start up a couple other operations
  Keep core level simple as possible
  Build clear map of steps
  - Setting biases
  - Deciding order registers get tuned (Jeremy has some presentations lying around)
  - Parameter extractions from couple of runs
  Testing firmware requires building ~90% low-level software as well as the firmware itself. Other 10% PLUS next-level wrapper (take a run; take sequence of runs) should definitely be joint UMN/LU/UVA

New HGCROC boards arrive Lund

HGCROC board | **Adapter between ROC and ZCU**

Includes quality control on voltages around the board; checking adapter works

+1 week

Shipping. +1 week

Firmware for the ZCU to readout the ROC

Borrow from CMS, need to be prepared to support it

Some things (eg some I2C comms) could be prepared before the +2 weeks from board manufacture

Data/link capture most delicate and will take longer

100 Hz first draft
+1 month

***Low-level software:** Successor to pflib*

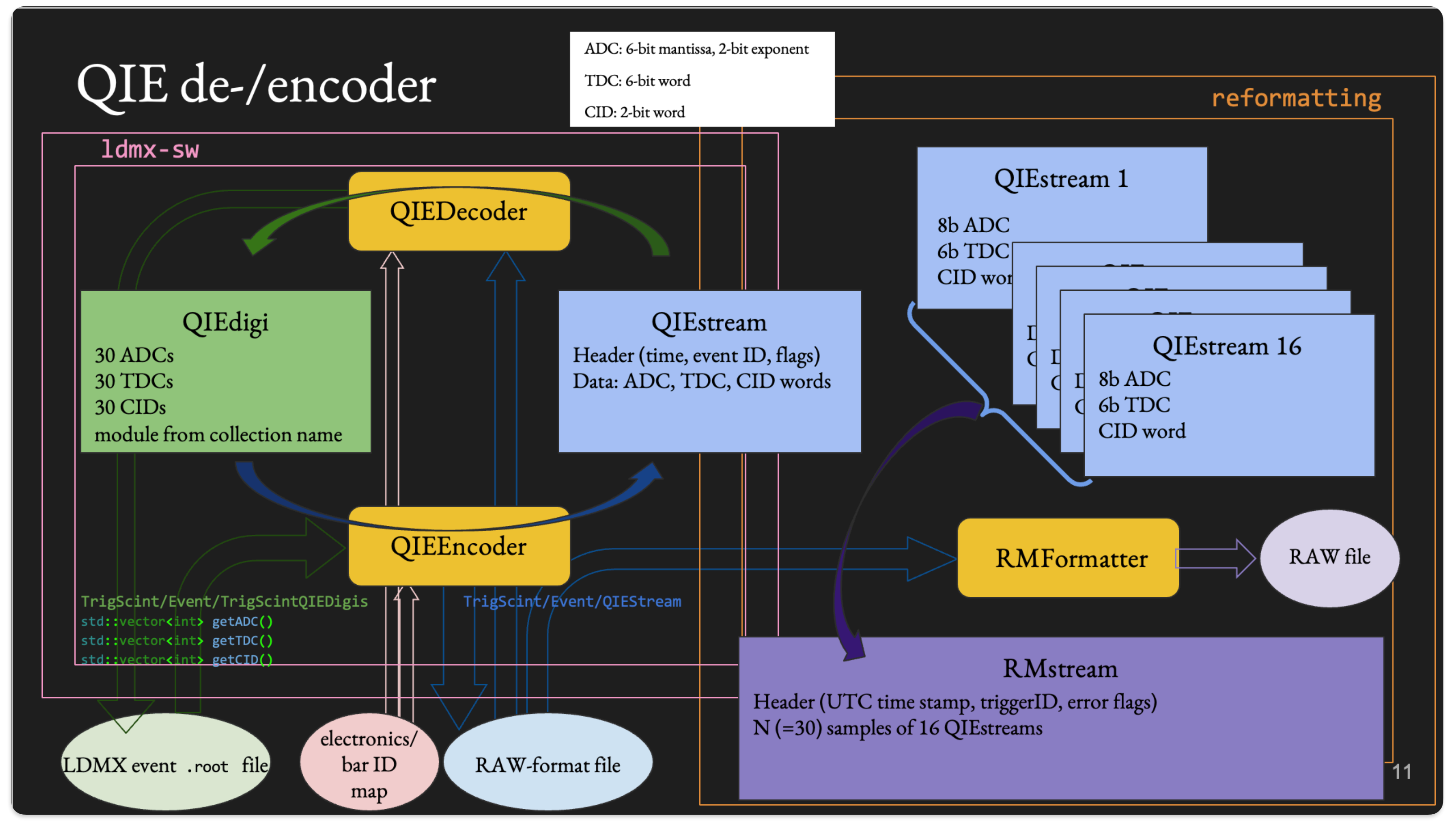
Control fast control
Set up capture blocks (phases 0.1 ns for getting data from electrical link (e-link) to get bits into system; timing wrt L1A)
Pulling data from buffers so it can go to disk
Housekeeping (resets)

100 Hz first draft
Evolving

Wrapper software (onion/nested doll)

I2C communication
  Full register map of the ROCs
  Establish reliable communication with ROC
  Issue resets
Communicate with firmware already developed
µHal to communicate with data capture side (outputing byte streams or ...)
Setting up a run; setting up a sequence of runs

TODO: Consider how we can modularize this box and split work accordingly ; map the pieces ; pftool as inspiration



QIE de-/encoder
ADC: 6-bit mantissa, 2-bit exponent
TDC: 6-bit word
CID: 2-bit word
reformatting
ldmx-sw
QIEDecoder
QIEdigi
30 ADCs
30 TDCs
30 CIDs
module from collection name
QIEstream
Header (time, event ID, flags)
Data: ADC, TDC, CID words
QIEstream 1
8b ADC
6b TDC
CID word
QIEstream 16
8b ADC
6b TDC
CID word
QIEEncoder
RMFormatter
RAW file
TrigScint/Event/TrigScintQIEDigis
std::vector<int> getADC()
std::vector<int> getTDC()
std::vector<int> getCID()
TrigScint/Event/QIEStream
RMstream
Header (UTC time stamp, triggerID, error flags)
N (=30) samples of 16 QIEstreams
LDMX event .root file
electronics/ bar ID map
RAW-format file
11

Summary, 9 October 2024 meeting • https://indico.fnal.gov/event/66391/
Erik, Lene Kristian, Lennart, Cristina, Jeremy, Hannah

Backplaneless readout of HGCROC board requirements
- HGCROC board v2 ("board")
- ROC board ↔ ZCU development board adapter card ("card")
- Firmware installed on the dev board
- Low-level software (on a neighbouring computer) for key firmware controls/interfaces
- Wrapper-level softwares

Board + Card availability [LU]
- 4 boards in production at CERN electronics workshop now
  - Torsten will pass the workshop 10 October to get an update on the production
- 16 more boards will be produced when we approve these 4 (total 20 boards)
- Lennart will QC powering, voltage levels across the four boards before any distribution
  - Could approve further production at this point, or wait for experience with readout chain
- Lennart has also designed the adapter card
  - Need to actually test board + card works
- Special handling: Good idea to get a dissipative mat + use grounding wrist straps while working with the boards

Board + Card distribution [LU]
- 4 board + card sets in near future
  - 1 each to UMN, UVA, LU
  - 1 more available
    - Extra for UMN or LU?
    - FNAL?
- Board + Card set quality-controlled before distribution

Firmware [UMN]
- Fastest way to make firmware available: Crib off the CMS stackup
  - Most firmware blocks available
  - 3 main species: fast control, I2C, capture
- We cannot expect CMS to maintain our implementation or make adaptations for us.
  - If somebody else (not Jeremy/UMN) wants to maintain our "fork" of CMS's firmware, we need CMS' permission
  - Jeremy/UMN comfortable with CMS fast control and I2C blocks. Data capture (also called link capture) needs more attention → most delicate part and will take longer
- Capture side is readout-rate-dependent
  - 100 - 1000 Hz reasonably achievable to implement accurately and quickly
  - Readout computer attached to ZCU also has to keep up
  - **We agreed to start at 100 Hz**
    - Better to get a draft of the chain and start identifying how things work → "Start simple, then get fancy"
    - We'll need to converge with TDAQ's rates eventually
      - Elected to work that timeline out separately

Low-level software [UMN]
- Some low-level software required to test firmware
- Functions include
  - Controlling fast control
  - Setting up capture blocks
  - Pulling data from buffers so they can go to disk
  - Some housekeeping
- `pflib` library for 2022 testbeam not appropriate for this anymore
  - Polarfire != ZCU Xilinx
  - ZCU actually more powerful
  - We do not want to build a Polarfire emulator
- Consider this low-level software the successor to `pflib`

Wrapper softwares [LU, UVA with significant input from UMN]
- Successor to `pftool`
  - Use `pftool` as a model
- Responsible for higher-level tasks requiring eg more coordination
  - Setting up runs/sequences of runs
  - Setting biases
  - Full register map of ROCs, setting order registers get tuned
  - Parameter extractions
- Key opportunity to build distributed support model and knowledge base across HCal-LDMX

Timeline
- Setting t0 to "HGCROC boards back at LU from CERN workshop"
- [LU] Task 1 - 1 week: Board + adapter quality control (total time: t0 + 1 week)
- [LU] Task 2 - 1 week: Board + adapter distribution, waiting on shipments (total time: t0 + 2 weeks)
- [UMN] Task 3 - 2 weeks: Firmware preparation before board in-hand (parallel to Tasks 1, 2; total time: t0 + 2 weeks)
- [UMN] Task 4 - 1 month: First draft of firmware + low-level software for readout at 100 Hz distributed to LU and UVA (total time: t0 + 6 weeks)
- [UVA, LU, UMN] Task 5 - evolving; to be broken down!!: Wrapper software
  - [UVA, LU] Task 5.1 - 6 weeks: Map needed wrapper software functions into modules and begin designing implementation based on `pftool` (parallel to tasks 1-4; total time: t0 + 6 weeks)
    - Includes breaking Task 5 down into sub-tasks and assigning time planning to those sub tasks
    - Evolving product
      - Make a plan for tracking feature requests, bug reports
  - Cristina, Erik, Lene Kristian, Hannah, Gréta
- **Total time estimate to viable backplaneless readout path at 100 Hz draft: t0 + 6 weeks minimum**