Status of ITk Pixel DAQ SW developments

(End of qualification task)

Angira Rastogi,

Simone Pagan Griso, Timon Heim

LBL Weekly Instrumentation Meeting

July 21st, 2023

Outline

- Various DAQ SW developments
  - Self-trigger source scan
  - Optoboard-FELIX test stand ← Start of QT
  - Digital scans with software-triggering
  - Digital scans with firmware-triggering
  - Reading register frames
  - Parallel masking
  - Data processor feedback

  } New!

- Summary, future plans

Various DAQ SW developments

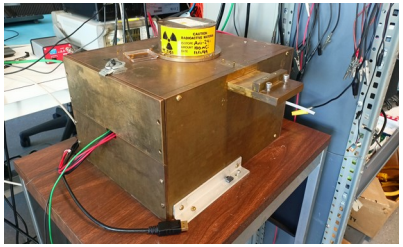
June-July, 2022

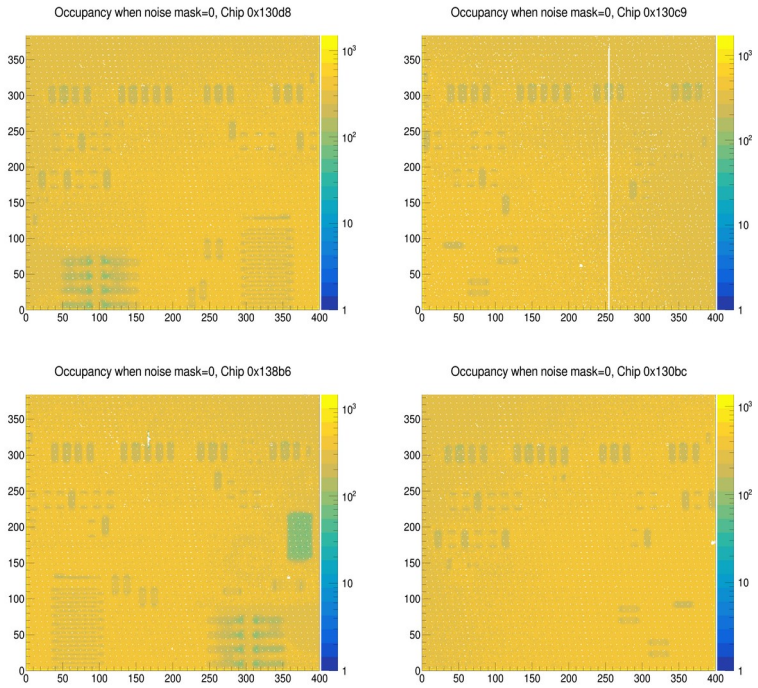
Self-trigger source scan

Implemented:

- Self-trigger digital scan
- Self-trigger analog scan
- Self-trigger source scan
- Tuning the various self-triggering parameters (delay, trigger multiplier, digital threshold).
- Fixed memory leak issue of the histogrammer clipboard.

Merge request





Masking sequence

- Standard digital scan (-m=1)
- Standard analog scan
- Standard noise scan
- Self-trigger digital scan
- ~~Self-trigger analog scan: MaskLoop(64,0,1)~~ Segmentation fault at random mask stages (communication errors).
- pToT digital scan
- pToT analog scan
- Self-trigger source scan (createMask = False in Noise analysis)
- Latency = 57, SelfTrigDelay = 45, SelfTrigDigiThr = 1, SelfTrigMulti = 4, Scan time = 3600s.

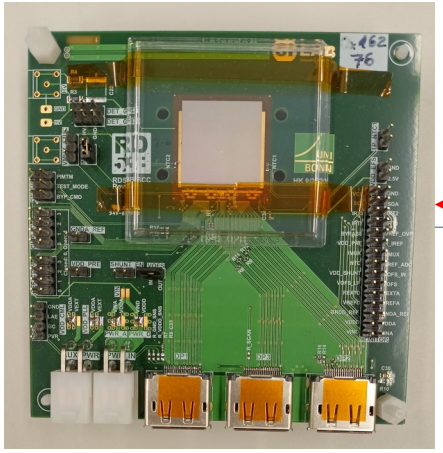
Weekly Instr. meeting
RD53B tesing meeting

Various DAQ SW developments

Start of QT: July 15, 2023

June-July, 2022 | Aug-Oct, 2022

To set up a ITk Pixel Optoboard-FELIX test stand at LBNL



ITkPix SCC, V1.1

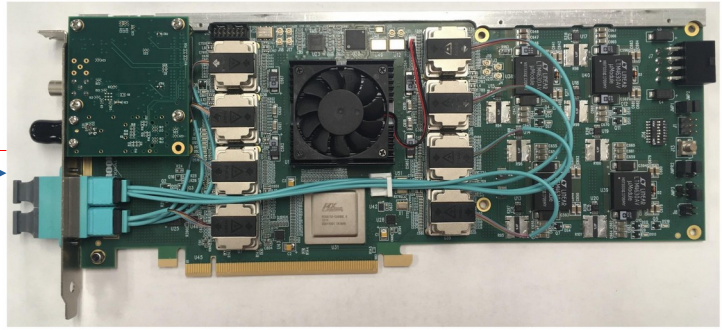
CMD

DATA

Optoboard, V2.1

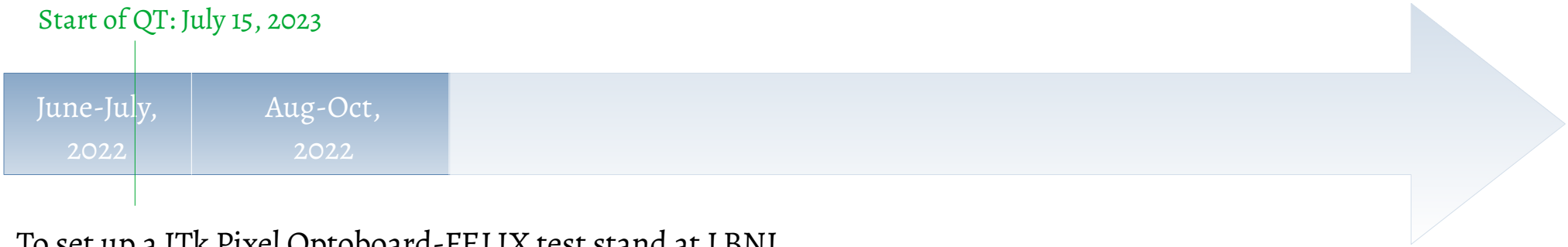
CMD

DATA



FELIX, FLX-712

Various DAQ SW developments

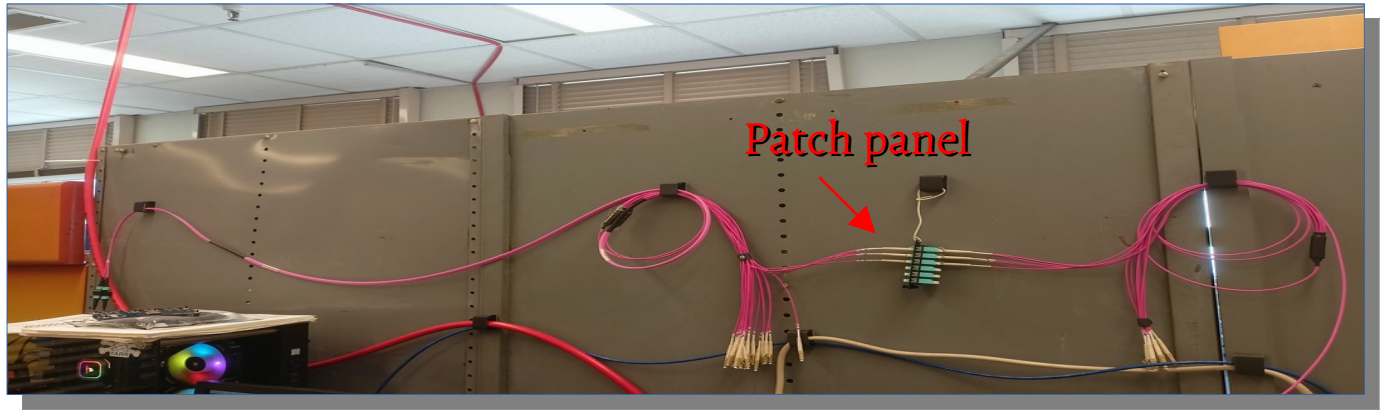


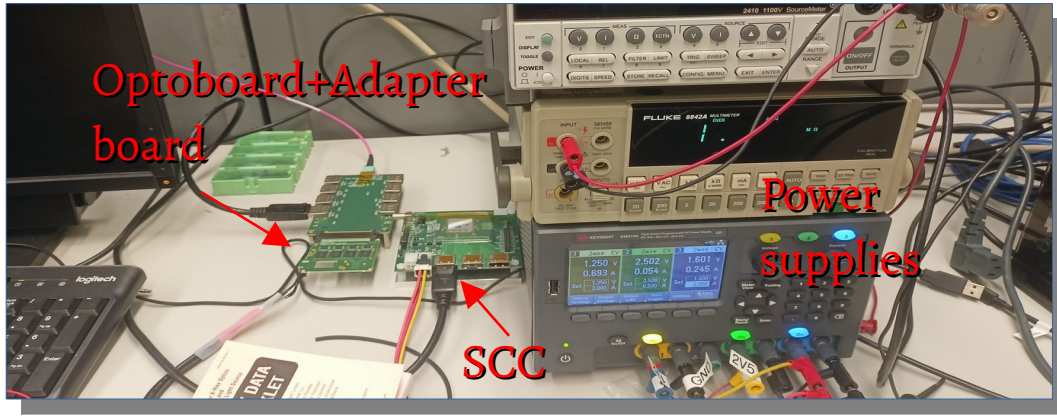
<u>To set up a ITk Pixel Optoboard-FELIX test stand at LBNL</u>

- For ATLAS TDAQ software-related developments.
- For a more realistic system testing or system-test like infrastructure e.g. for the Pixel Luminosity Ring (PLR).
- Validation of the base framework can be done for pixels and strips simultaneously through FELIX at LBNL.
- To test the scalability of the current setup like a realistic detector
  - reading out the full 8-quad module serial power chain
  - reading out the 5-triplet module serial power chain as well.

Current setup

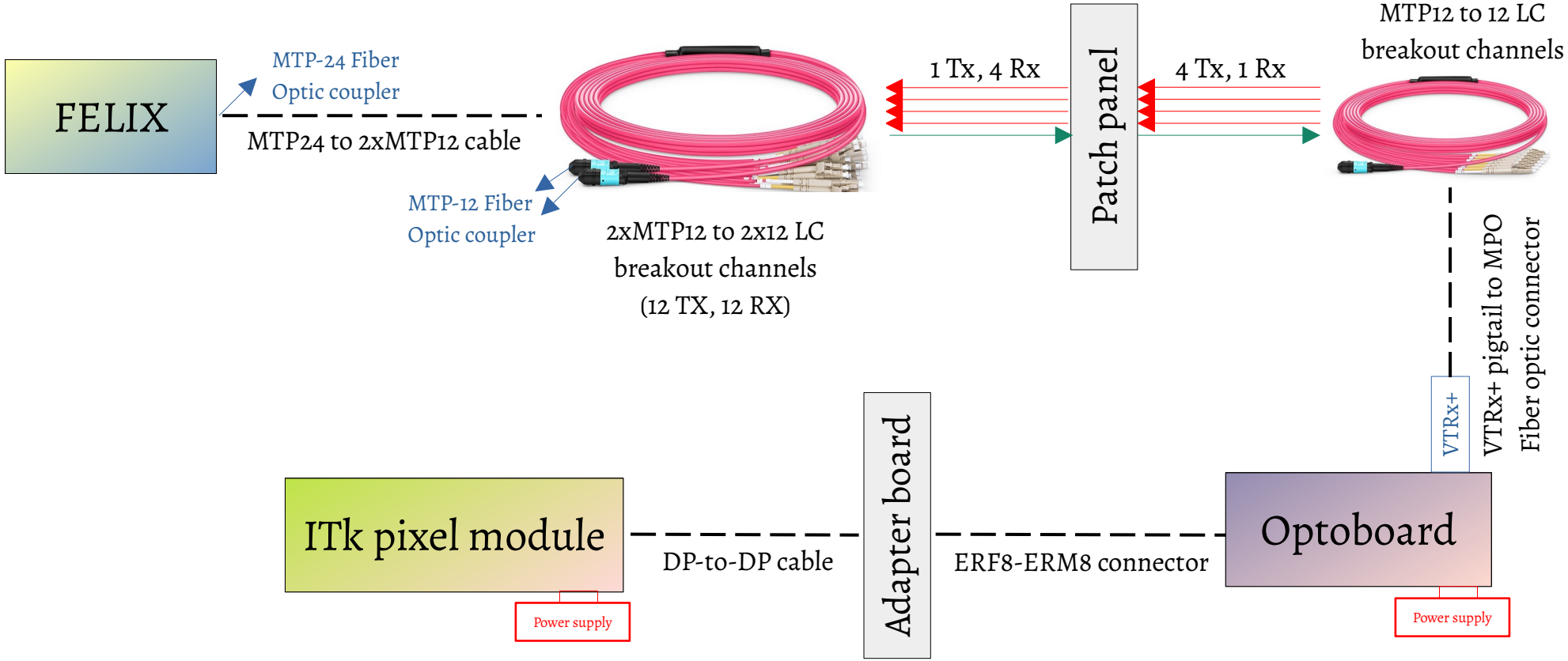
50B-6038



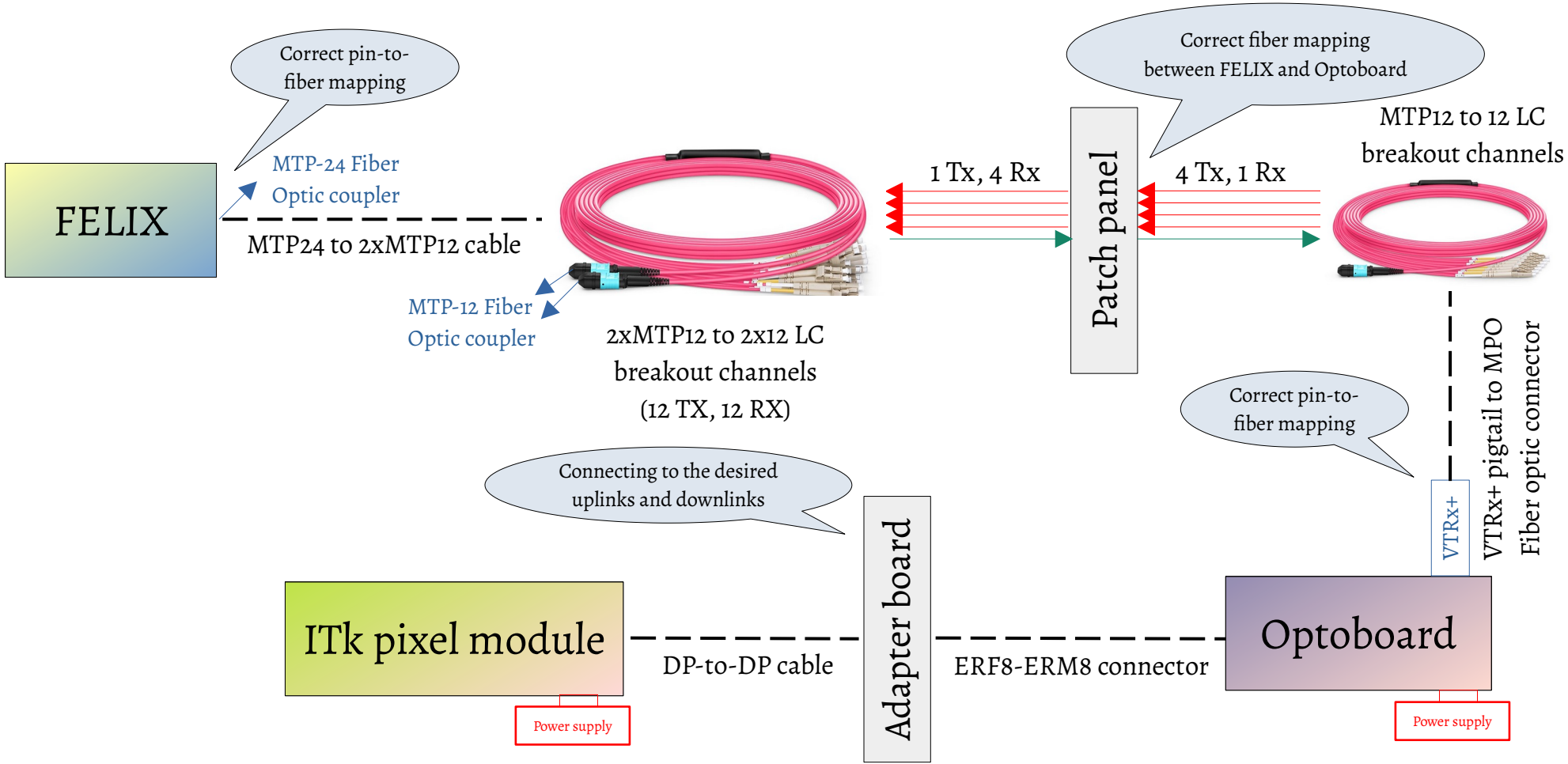


Setting up the DAQ chain

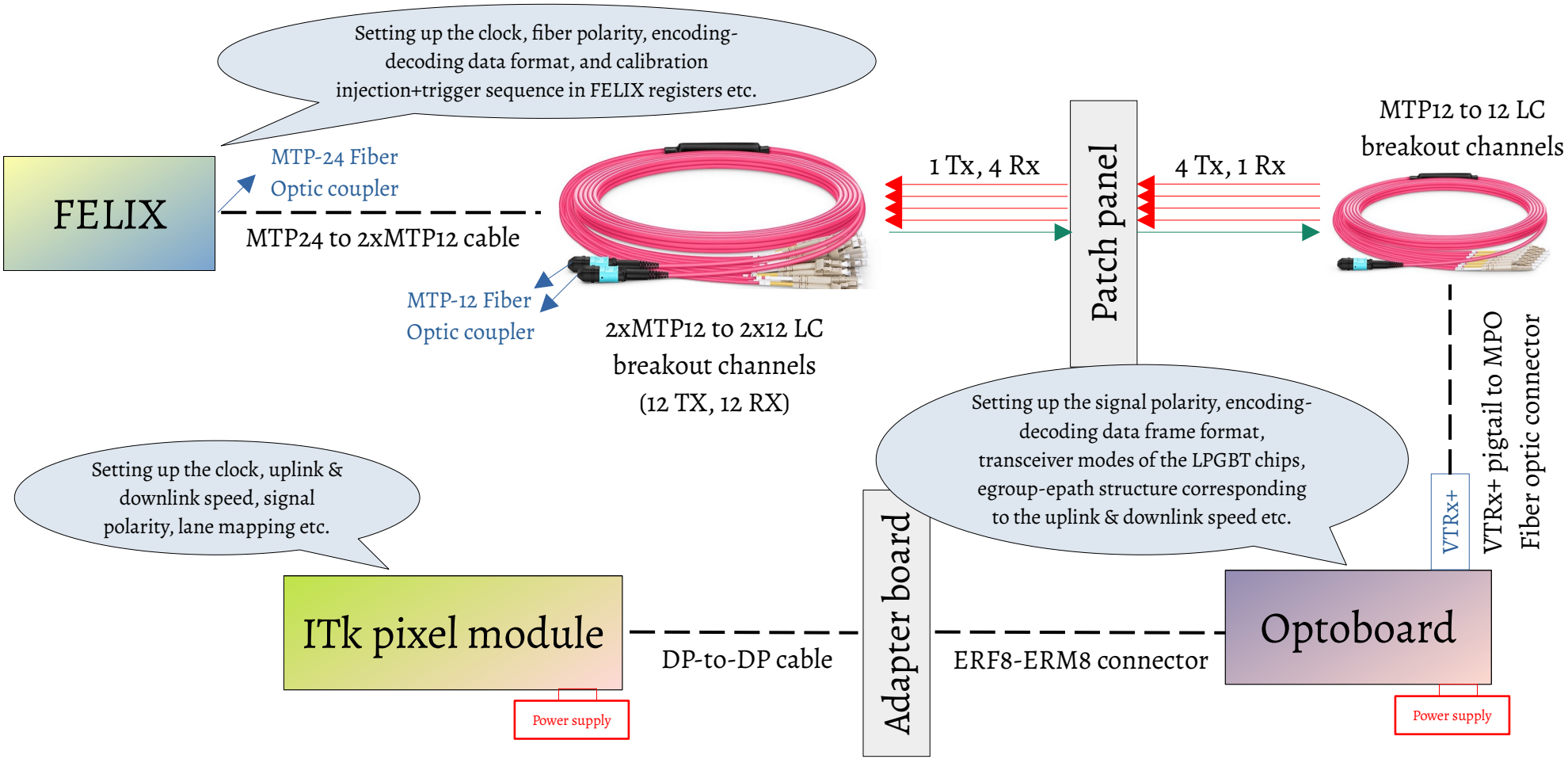
https://atlaswiki.lbl.gov/pixels/felixdaq



Hardware-level complexities

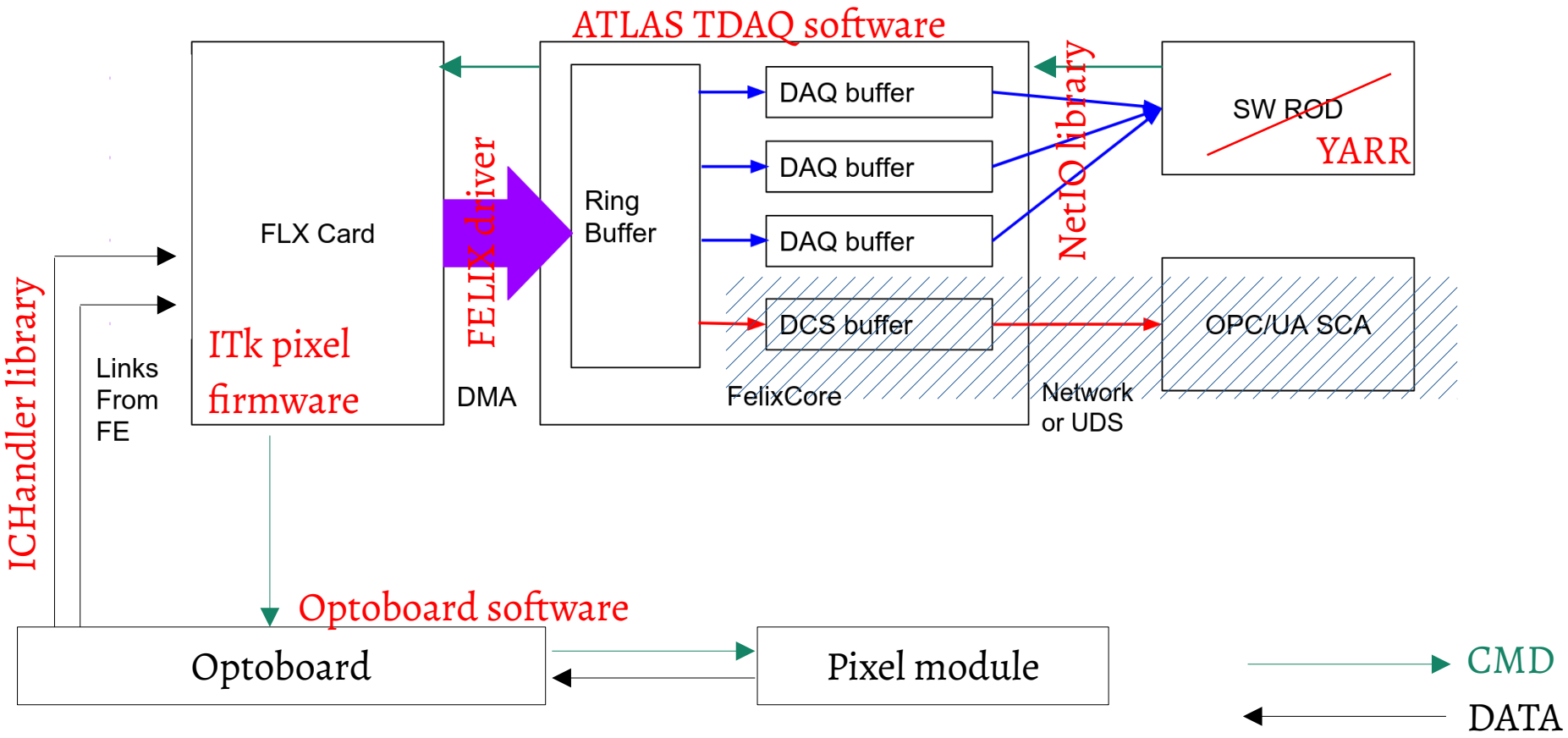


Configuration-level complexities



Driver, firmware & software

https://atlaswiki.lbl.gov/pixels/felixdaq

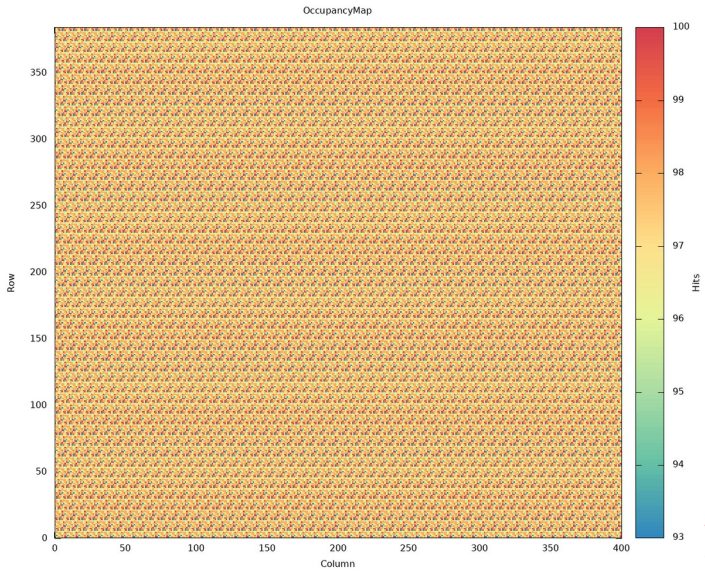


Various DAQ SW developments

June-July, 2022	Aug-Oct, 2022	Nov-Dec, 2022

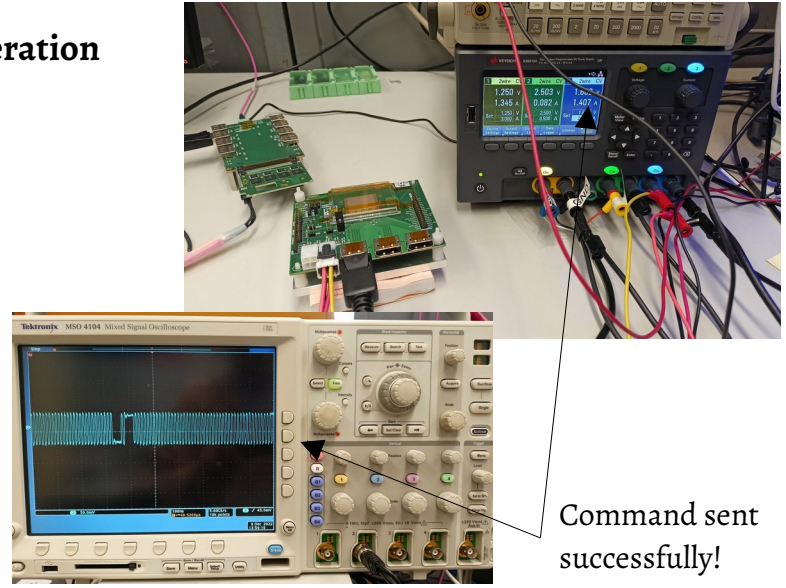
Weekly Instr. meeting, Dec 9th

<u>Running unsuccessful digital scans</u>



- **Software-based trigger generation**
- Trigger frequency = 100 Hz
- Injections = 100
- Number of mask stages = 64
- rxWaitTime = 1 sec.

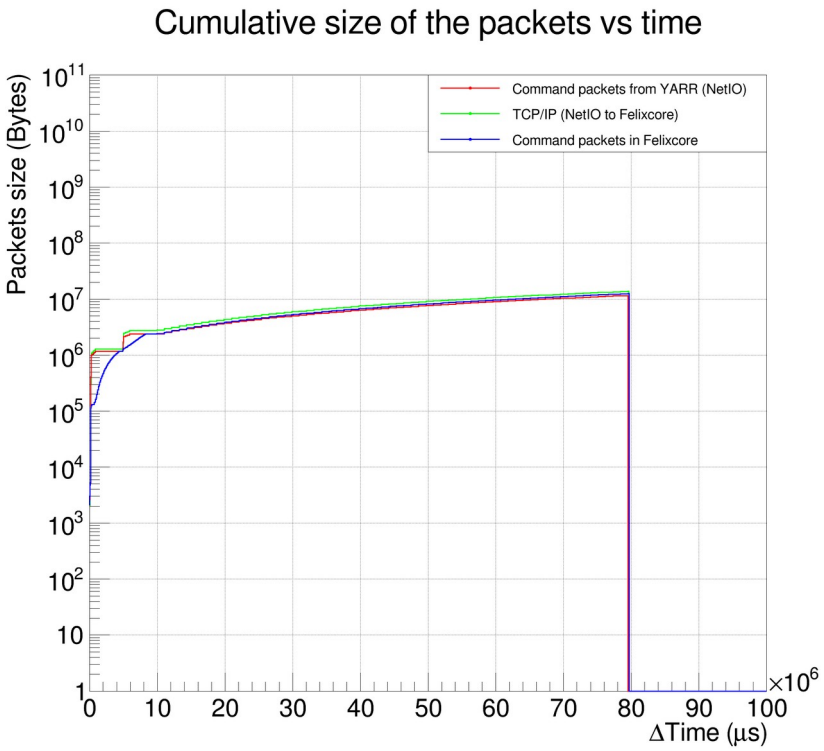
Data losses observed.
Also, FELIX software very unstable, would crash unexpectedly.



Command sent successfully!

Debugging software-triggering

- SW triggers from the YARR (NetIO) are delivered unevenly to felixcore because of the network buffering – also seen within the strips community here.
- Unlike firmware triggers, full sequence of SW trigger command is packed together by NetIO and sent out for every trigger-generation.
- As a result, extra latency introduced in the network before reaching FELIX.

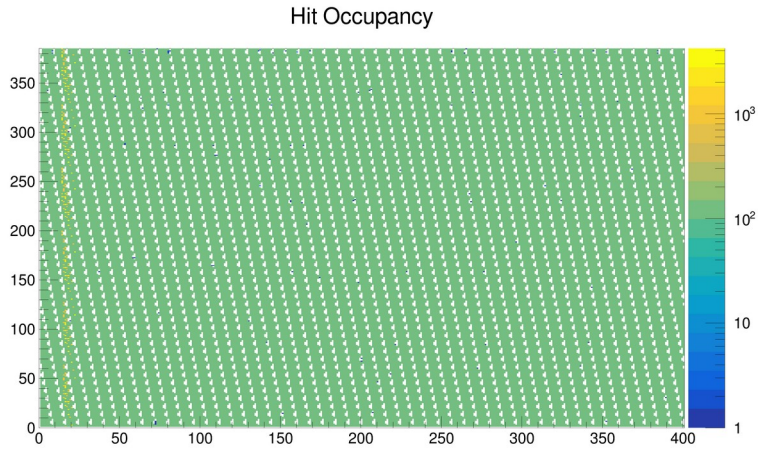


- No easy fix for this in that moment.
- Also, other people seemed to be using firmware-triggering, which is more desired for. So, we also switched to that…

Various DAQ SW developments

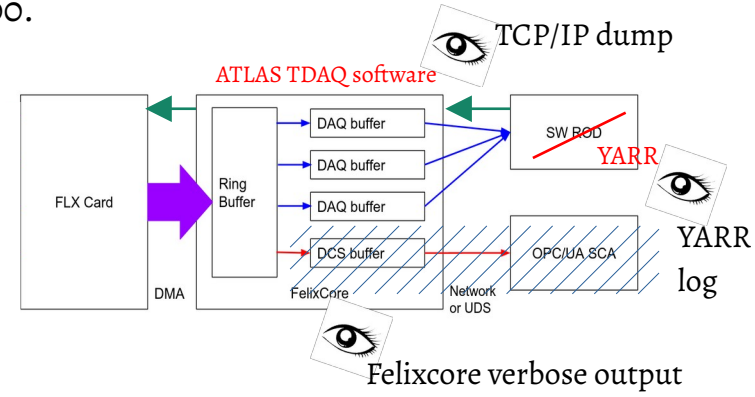
June-July, 2022	Aug-Oct, 2022	Nov-Dec, 2022	Jan-Mar, 2023

Switched to firmware-based triggering



Missing chip readout data. Incorrect digital scan results.

- **Firmware-based trigger generation.**
- rxWaitTime = 1 s.
- Trigger frequency = 5000 Hz.
- Trigger count (injections) = 100.
- Number of mask stages = 64.



Debugging firmware-triggering

Fig. 1: Cumulative command packets size from YARR (red), in the network from TCP/IP dump (green) and in felixcore (blue) vs time.

- All the packets from YARR are not seen by felixcore, around 5% are lost. Also, last packet seen in felixcore ~4 seconds later after YARR finished sending all commands due to network buffering. Hence, many empty mask stages (StdDataLoop wait time ~ 1 sec).
- Many smaller-sized TCP/IP packets are created by the network before sending to felixcore, hence downstream buffer flushes at a slower rate.

Fig. 2: Example behavior of a few mask stages from the digital scan, with data from chip seen in NetioHandler (yellow).

- Data received around 0.5 seconds later in every mask stage initially, but then becomes asynchronous due to the network latencies (especially towards last few mask stages).
- No command packets seen in felixcore occasionally due to network buffering, hence no readout data from chip (for e.g. at the 59th second).

Fig. 1

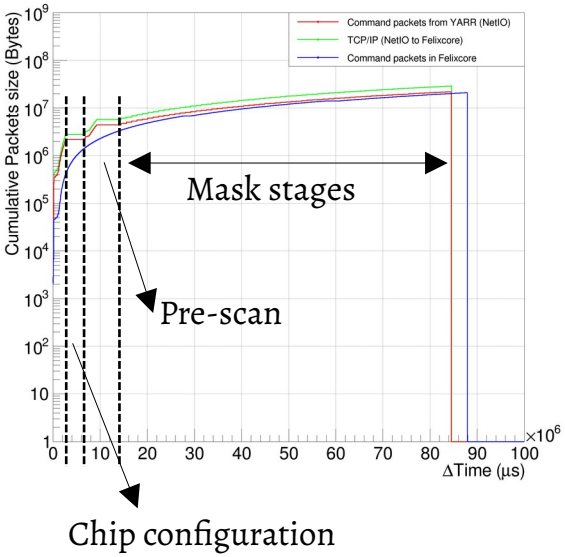
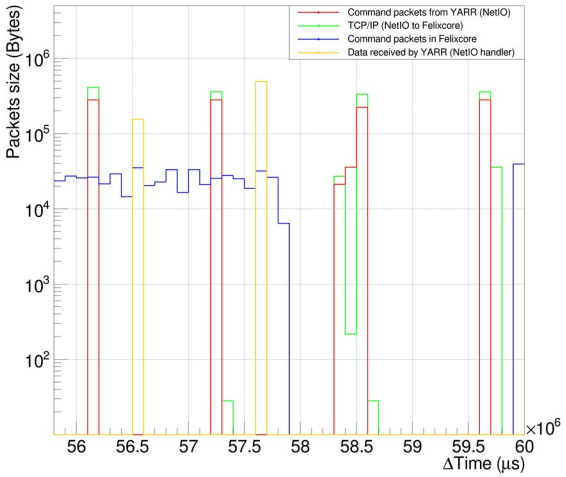


Fig. 2



Weekly Instr. meeting, Feb 24th

Fix firmware-triggering

Solution:

Aggregate the command packets from YARR (NetioTxCore) to send longer messages, flushing the NetIO FiFo regularly using IsCmdEmpty() call.

- Advantages: Commands reach the chip faster, and in its entirety. Less number of packets overall, hence less crowding of command packets in FELIX.

Results:

- Successful digital scan .
- All the command packets from YARR reach felixcore almost instantly (Fig. 3). Also, less number of extra packets created by the network while in buffer.
- Scan time faster by 10%, due to aggregation of command packets.
- Data received within 0.2 to 0.3 seconds later in every mask stage which is roughly a factor of 2 faster than before (Fig. 4).

Fig. 3

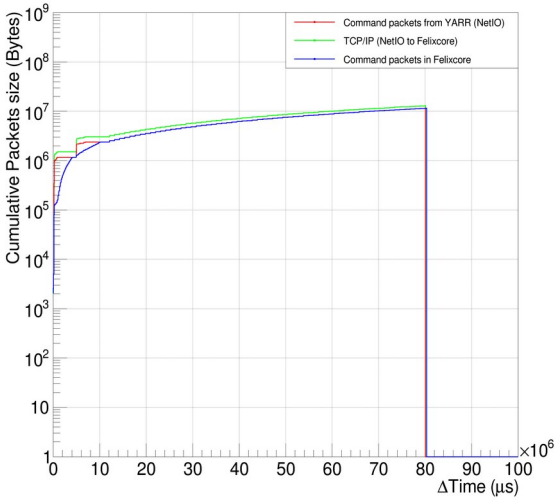
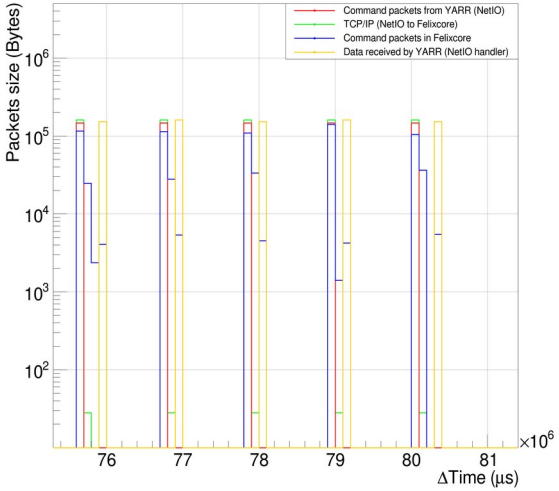
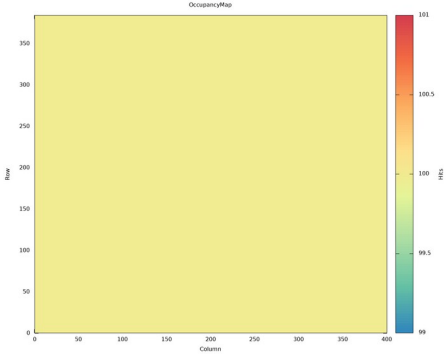


Fig. 4



MR !615



Various DAQ SW developments

June-July, 2022	Aug-Oct, 2022	Nov-Dec, 2022	Jan-Mar, 2023	Apr-May, 2023

<u>Reading chip configuration registers</u> New!

- Currently, two separate elinks in the firmware for the service and readout data frames.
- They need to be merged (or at least copying the ones with useful K-words in the data elink) for full interaction with chip functionalities (configuration, DCS monitoring, special commands etc.) during the normal scans.

- Until then, we can do this from the software-level.
- A simple hack in the NetIO client for YARR.

Adding two elinks in NetIO-YARR

- In addition to the original data channel (elink), we also enable channel (elink+offset), where "offset" determines the shift in elink number for the service frames.

NetioRxCore for YARR

```
void NetioRxCore::enableChannel(uint64_t elink){
  nlog->info("Enable RX elink: 0x{:x}", elink);
  if(m_elinks.find(elink)==m_elinks.end()){
    m_nioh.addChannel(elink);
    std::this_thread::sleep_for(std::chrono::milliseconds(5000)); // Note: is this needed; can it be shortened?
    m_elinks[elink]=true;
    if(m_regOffset>0){
      //spdlog::info("Adding register frames elink: 0x{:x}",elink+m_regOffset);
      m_nioh.addChannel(elink+m_regOffset);
      std::this_thread::sleep_for(std::chrono::milliseconds(5000)); // Note: is this needed; can it be shortened?
      m_elinks[elink+m_regOffset]=true;
    }
  }
}
```

m_regOffset=1 (in controller config)

```
"SerSelOut2": 1,
"SerSelOut3": 1,
"ServiceBlockEn": 1,
"ServiceBlockPeriod": 50,
"SkippedTrigCnt": 0,
"SldoEnUndershuntA": 0,
```

Setting to 1 in the chip configuration

- Played a lot with firmware building at this point, but the ITk-pixel FW git repository is not up-to-date.
- Wanted to add command ILA signals to trace the commands from YARR in Felixcore.

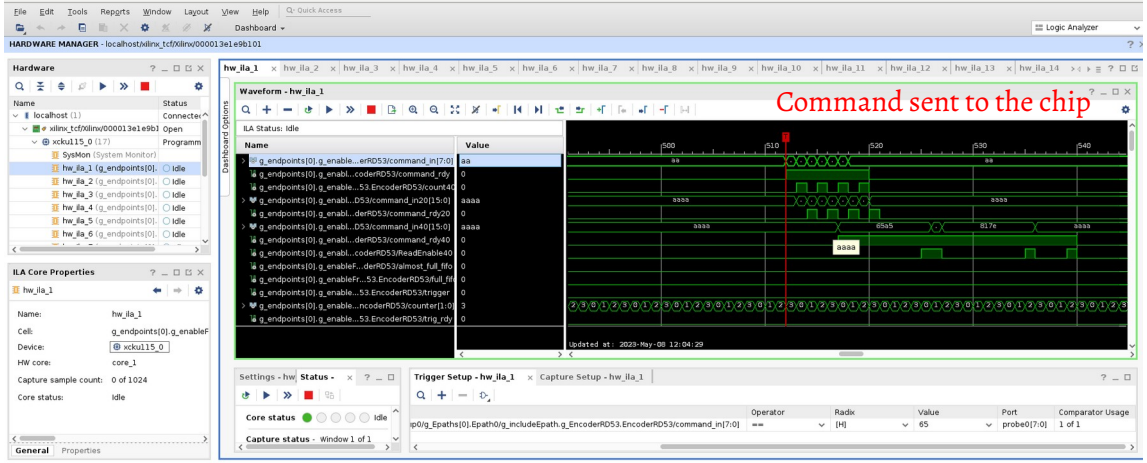
- Opened a new JIRA ticket to request for adding command ILA probes in the firmware: FLXUSERS-606

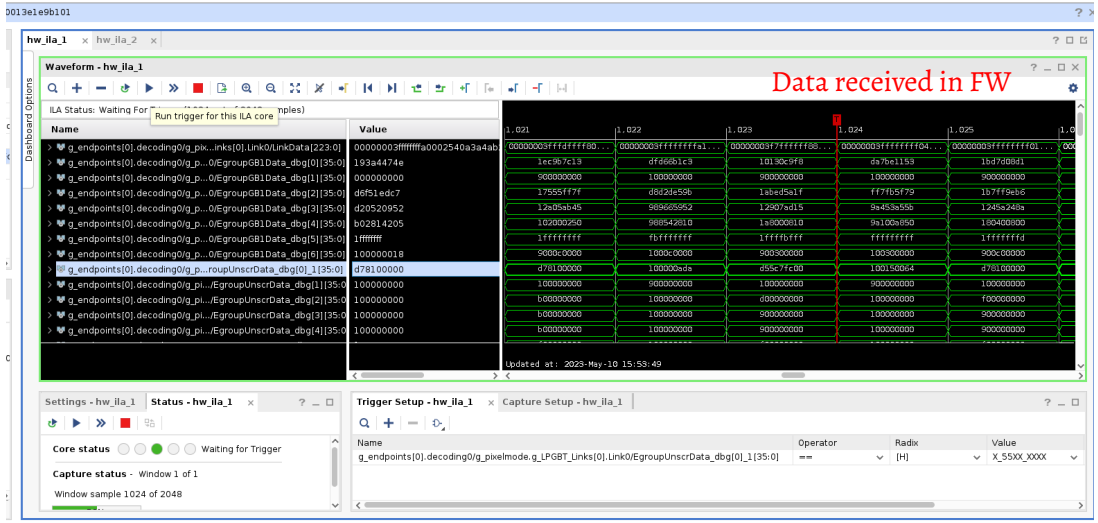
Felixcore verbose

```
[2023-05-09 14:26:01.331] [debug] [21715] one-shot: toflx card 0, dma transfer: size = 128
[2023-05-09 14:26:01.331] [debug] [21715] toflx card 0, transferring fragment: ptr = 0xe74800000, size = 128
[2023-05-09 14:26:01.332] [debug] [21715] toflx card 0, waiting for dma transfer to finish...
[2023-05-09 14:26:01.332] [debug] [21715] toflx card 0, fragment dma transfer finished
[2023-05-09 14:26:01.332] [debug] [21715] toflx card 0, dma transfer finished
[2023-05-09 14:26:01.332] [debug] [21715] Received message from 127.0.0.1:52620, size=28, parts=1
[2023-05-09 14:26:01.332] [debug] [21715] MSG received for elink 0
[2023-05-09 14:26:01.332] [debug] [21715] MSG encoded size=32
[2023-05-09 14:26:01.332] [debug] [21715] one-shot: toflx card 0, dma transfer: size = 64
[2023-05-09 14:26:01.332] [debug] [21715] toflx card 0, transferring fragment: ptr = 0xe74800000, size = 64
[2023-05-09 14:26:01.332] [debug] [21715] toflx card 0, waiting for dma transfer to finish...
[2023-05-09 14:26:01.332] [debug] [21715] toflx card 0, fragment dma transfer finished
[2023-05-09 14:26:01.332] [debug] [21715] toflx card 0, dma transfer finished
[2023-05-09 14:26:03.479] [info] [21721] Subscription from endpoint 127.0.0.1:40863 for elink 0
[2023-05-09 14:26:08.479] [info] [21721] Subscription from endpoint 127.0.0.1:38727 for elink 1
```

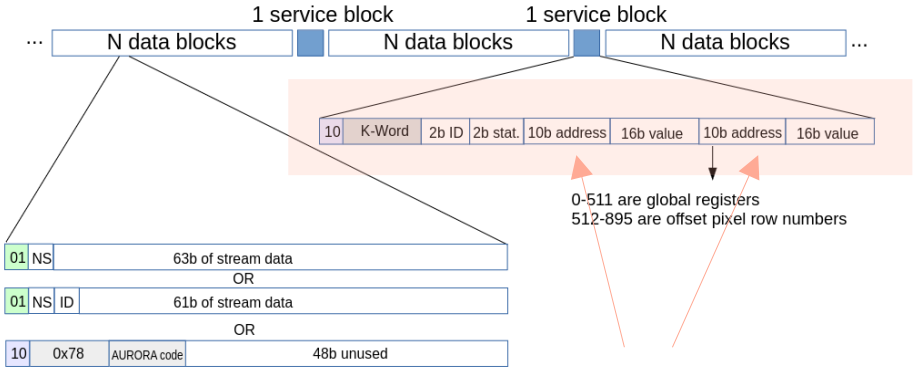
- Felixcore verbose shows that we have subscribed to both the elinks.
- Hence, we should be able to see the register frames now.

Reading register frames



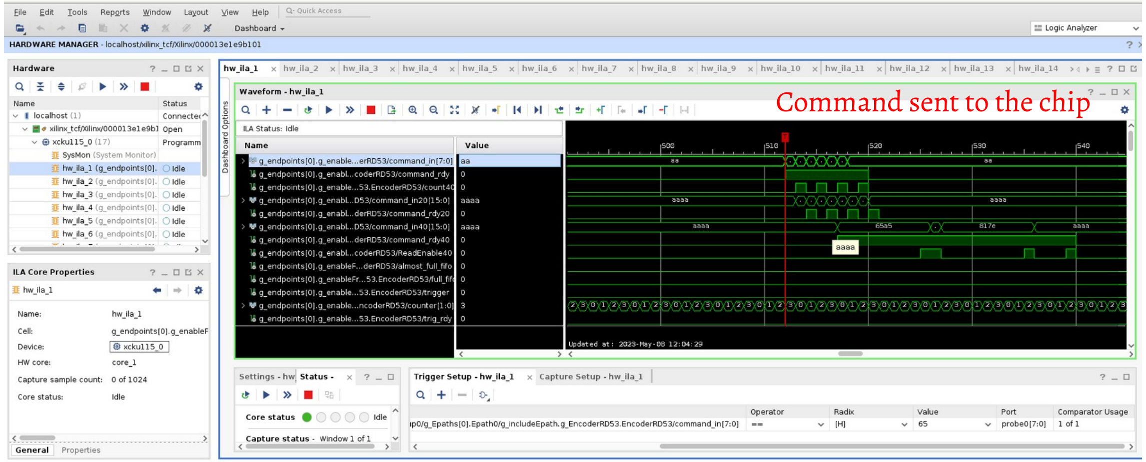


Aurora frames



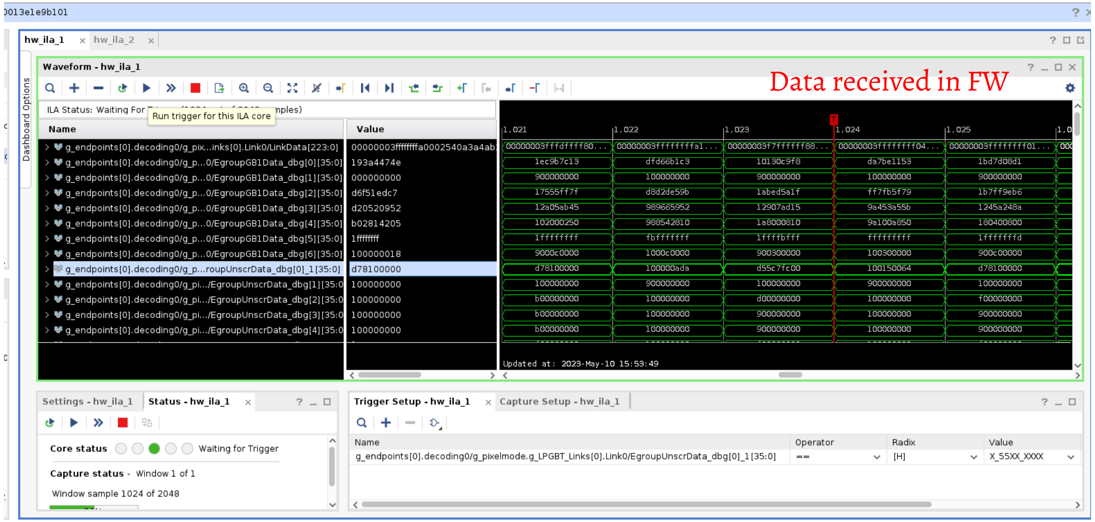
Aurora K-Word code (hex)	Meaning
0xB4	both register fields are of type AutoRead
0x55	first frame is AutoRead, second is from a read register command
0x99	first is from a read register command, second frame is AutoRead
0xD2	both register fields are from read register commands
0xCC	Indicates an error. Fields are meaningless

Reading register frames



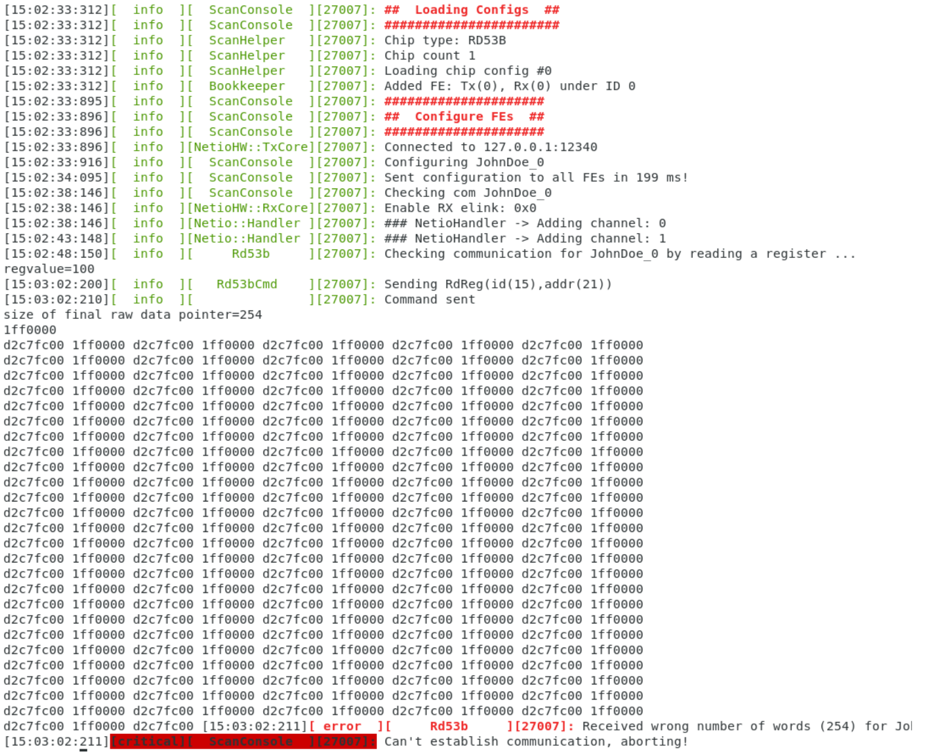
Name	Value
g_endpoints[0].g_enable...erRD53/command_in[7:0]	aa
g_endpoints[0].g_enabl...coderRD53/command_rdy	0
g_endpoints[0].g_enable...53.EncoderRD53/count40	0
g_endpoints[0].g_enabl...D53/command_in20[15:0]	aaaa
g_endpoints[0].g_enabl...derRD53/command_rdy20	0
g_endpoints[0].g_enabl...D53/command_in40[15:0]	aaaa
g_endpoints[0].g_enabl...derRD53/command_rdy40	0
g_endpoints[0].g_enabl...coderRD53/ReadEnable40	0
g_endpoints[0].g_enableF...derRD53/almost_full_fifo	0
g_endpoints[0].g_enableFr...53.EncoderRD53/full_fifo	0
g_endpoints[0].g_enable...53.EncoderRD53/trigger	0
g_endpoints[0].g_enable...ncoderRD53/counter[1:0]	3
g_endpoints[0].g_enable...53.EncoderRD53/trig_rdy	0

Name	Operator	Radix	Value	Port	Comparator Usage
...p0/g_Epaths[0].Epath0/g_includeEpath.g_EncoderRD53.EncoderRD53/command_in[7:0]	==	[H]	65	probe0[7:0]	1 of 1



Name	Value
g_endpoints[0].decoding0/g_pix...inks[0].Link0/LinkData[223:0]	00000003ffffffffa0002540a3a4ab
g_endpoints[0].decoding0/g_p...0/EgroupGB1Data_dbg[0][35:0]	193a4474e
g_endpoints[0].decoding0/g_p...0/EgroupGB1Data_dbg[1][35:0]	000000000
g_endpoints[0].decoding0/g_p...0/EgroupGB1Data_dbg[2][35:0]	d6f51edc7
g_endpoints[0].decoding0/g_p...0/EgroupGB1Data_dbg[3][35:0]	d20520952
g_endpoints[0].decoding0/g_p...0/EgroupGB1Data_dbg[4][35:0]	b02814205
g_endpoints[0].decoding0/g_p...0/EgroupGB1Data_dbg[5][35:0]	1ffffffff
g_endpoints[0].decoding0/g_p...0/EgroupGB1Data_dbg[6][35:0]	100000018
g_endpoints[0].decoding0/g_p...roupUnscrData_dbg[0]_1[35:0]	d78100000
g_endpoints[0].decoding0/g_pi.../EgroupUnscrData_dbg[1][35:0	100000000
g_endpoints[0].decoding0/g_pi.../EgroupUnscrData_dbg[2][35:0	100000000
g_endpoints[0].decoding0/g_pi.../EgroupUnscrData_dbg[3][35:0	100000000
g_endpoints[0].decoding0/g_pi.../EgroupUnscrData_dbg[4][35:0	100000000

Name	Operator	Radix	Value
g_endpoints[0].decoding0/g_pixelmode.g_LPGBT_Links[0].Link0/EgroupUnscrData_dbg[0]_1[35:0]	==	[H]	X_55XX_XXXX



```
[15:02:33:312][  info  ][  ScanConsole  ][27007]: ##  Loading Configs  ##
[15:02:33:312][  info  ][  ScanConsole  ][27007]: #######################
[15:02:33:312][  info  ][  ScanHelper   ][27007]: Chip type: RD53B
[15:02:33:312][  info  ][  ScanHelper   ][27007]: Chip count 1
[15:02:33:312][  info  ][  ScanHelper   ][27007]: Loading chip config #0
[15:02:33:312][  info  ][  Bookkeeper   ][27007]: Added FE: Tx(0), Rx(0) under ID 0
[15:02:33:895][  info  ][  ScanConsole  ][27007]: #####################
[15:02:33:896][  info  ][  ScanConsole  ][27007]: ##  Configure FEs  ##
[15:02:33:896][  info  ][  ScanConsole  ][27007]: #####################
[15:02:33:896][  info  ][NetioHW::TxCore][27007]: Connected to 127.0.0.1:12340
[15:02:33:916][  info  ][  ScanConsole  ][27007]: Configuring JohnDoe_0
[15:02:34:095][  info  ][  ScanConsole  ][27007]: Sent configuration to all FEs in 199 ms!
[15:02:38:146][  info  ][  ScanConsole  ][27007]: Checking com JohnDoe_0
[15:02:38:146][  info  ][NetioHW::RxCore][27007]: Enable RX elink: 0x0
[15:02:38:146][  info  ][Netio::Handler ][27007]: ### NetioHandler -> Adding channel: 0
[15:02:43:148][  info  ][Netio::Handler ][27007]: ### NetioHandler -> Adding channel: 1
[15:02:48:150][  info  ][     Rd53b     ][27007]: Checking communication for JohnDoe_0 by reading a register ...
regvalue=100
[15:03:02:200][  info  ][   Rd53bCmd    ][27007]: Sending RdReg(id(15),addr(21))
[15:03:02:210][  info  ][               ][27007]: Command sent
size of final raw data pointer=254
1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000 d2c7fc00 1ff0000
d2c7fc00 1ff0000 d2c7fc00 [15:03:02:211][ error  ][     Rd53b     ][27007]: Received wrong number of words (254) for Jol
[15:03:02:211][critical][  ScanConsole  ][27007]: Can't establish communication, aborting!
```

- This time packets are received by YARR, they are **auto reads** in both the register fields (0xD2 instead of 0xB4 due to a bug in ItkPix).
- We need to filter out these auto reads otherwise the FelixCore buffer gets full very quickly.

Removing auto-reads

- To remove the several auto reads, we **build a new firmware** ourselves.
- We defined "is_good_userk" flag which is used to copy over the DCS frames with K-word "0x55" or "0x99", in the readout data elink.
- Hence, removing the auto reads while reading out from elink 0.

```
--DEMUX to separate data (hits) from k characters (service)
is_good_userk <= '1' when (rx_pe_data_i(63 downto 56) = x"55" and rx_user_k_i = '1') else
                 '1' when (rx_pe_data_i(63 downto 56) = x"99" and rx_user_k_i = '1') else
                                          '0';

  datadcs_out(63 downto 0)  <= rx_pe_data_i when (rx_user_k_i='1') else (others => '0');
  datadcs_out(64)           <= (rx_user_k_i and en40_i) and mask_k;
  data_out(63 downto 0)     <= rx_pe_data_i when (rx_user_k_i='0' or is_good_userk = '1') else (others => '0');
  data_out(64)              <= (rx_pe_data_v_i or is_good_userk) and en40_i;
  sep_out                   <= rx_sep_i;
```

64b66b_top.vhd

Alternatively,

- DECODING_MASK64B66BKBLOCK[3:0] register in FELIX firmware also helps in removing the service frames from Felixcore to network clients.
- Setting "DECODING_MASK64B66BKBLOCK=0x0" removes all the auto reads.

```
[15:57:06:207][  info  ][  ScanHelper   ][1339]: ~~~ }
[15:57:06:207][  info  ][  ScanConsole  ][1339]: ########################
[15:57:06:207][  info  ][  ScanConsole  ][1339]: ##  Loading Configs  ##
[15:57:06:207][  info  ][  ScanConsole  ][1339]: ########################
[15:57:06:207][  info  ][  ScanHelper   ][1339]: Chip type: RD53B
[15:57:06:207][  info  ][  ScanHelper   ][1339]: Chip count 1
[15:57:06:207][  info  ][  ScanHelper   ][1339]: Loading chip config #0
[15:57:06:207][  info  ][  Bookkeeper   ][1339]: Added FE: Tx(0), Rx(0) under ID 0
[15:57:06:796][  info  ][  ScanConsole  ][1339]: #####################
[15:57:06:796][  info  ][  ScanConsole  ][1339]: ##  Configure FEs  ##
[15:57:06:796][  info  ][  ScanConsole  ][1339]: #####################
[15:57:06:796][  info  ][NetioHW::TxCore][1339]: Connected to 127.0.0.1:12340
[15:57:06:816][  info  ][  ScanConsole  ][1339]: Configuring JohnDoe_0
[15:57:06:978][  info  ][  ScanConsole  ][1339]: Sent configuration to all FEs in 182 ms!
[15:57:11:032][  info  ][  ScanConsole  ][1339]: Checking com JohnDoe_0
[15:57:11:032][  info  ][NetioHW::RxCore][1339]: Enable RX elink: 0x0
[15:57:11:032][  info  ][Netio::Handler ][1339]: ### NetioHandler -> Adding channel: 0
[15:57:16:033][  info  ][     Rd53b     ][1339]: Checking communication for JohnDoe_0 by reading a register .
regvalue=100
[15:57:30:084][  info  ][   Rd53bCmd    ][1339]: Sending RdReg(id(15),addr(21))
[15:57:30:094][  info  ][               ][1339]: Command sent
Time elapsed since reading the buffer=30
Number of RDPs looked at in this time interval=0
[15:58:00:094][ error  ][     Rd53b     ][1339]: Did not receive any data for JohnDoe_0
[15:58:00:094][critical][  ScanConsole  ][1339]: Can't establish communication, aborting!
```

- No raw data pointer was found.
- The auto reads are clearly removed.

- Update to latest software version (FLX-1994)
- Check AXI stream output of the encoder in FW.
- Check display-stats in FELIX software, if the data chunks are set for timeout.
- Switch to FELIX-client (new FELIX SW).

Various DAQ SW developments

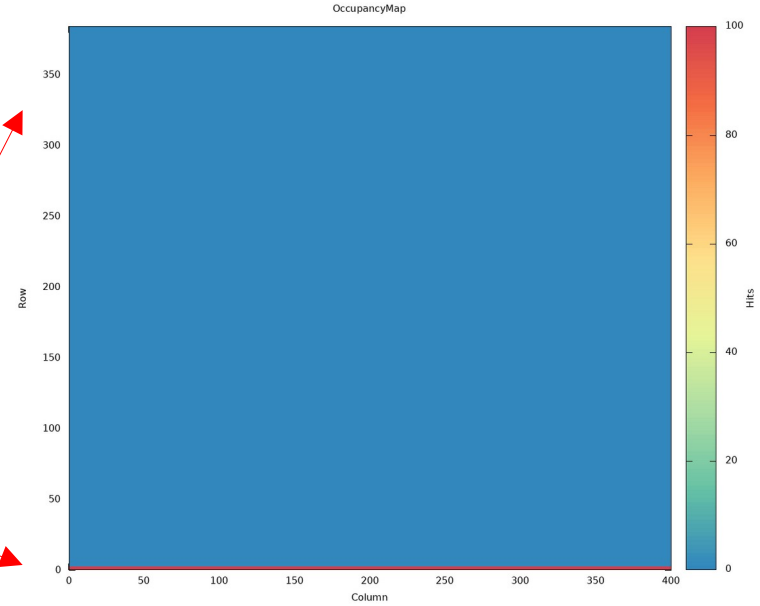
June-July, 2022	Aug-Oct, 2022	Nov-Dec, 2022	Jan-Mar, 2023	Apr-May, 2023

<u>Enable parallel pixel masking</u> New!

- The latest "devel" branch of YARR has a new class Rd53bParMaskLoop to set the pixel masking.
- This works by sending very long commands for one core column only, in each mask stage (different from the previous Rd53bMaskLoop).
- The chip then broadcasts the long command to all other core columns, to enable the pixel matrix for injections in that mask stage.

- But, it is not working as expected through NetIO-FELIX.

The rows of the core column are not properly set in a normal digital scan.

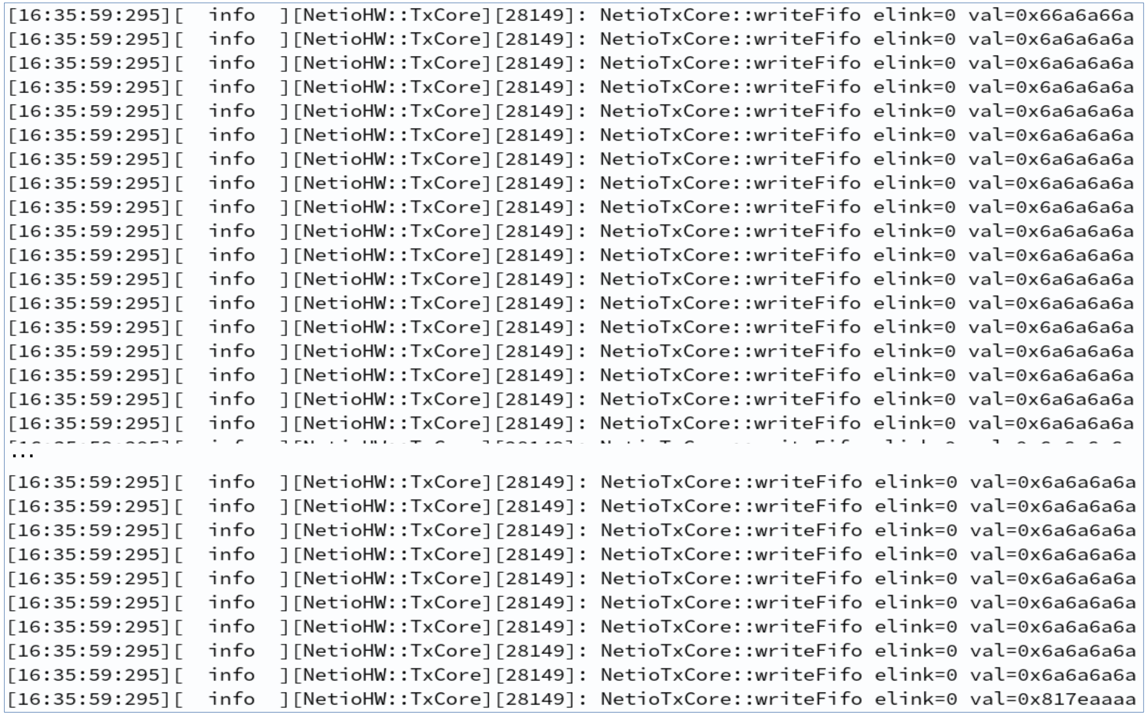


The “long” command

- This is WrReg(1) command, which starts with “0x66a6” (write Register + chip ID), followed by “0xa66a”.
- After that, to enable the masking for one core column, the FiFo is written out with **384 16-bit data words** (“0x6a6a”, but can be different as well).
- The software then writes out the sync “0x871e” and Pll lock “0xaaaa” words at the end of WrReg(1).

From NetIO-YARR, we can see the full command being written out to the FiFo.

YARR log



Command starts here

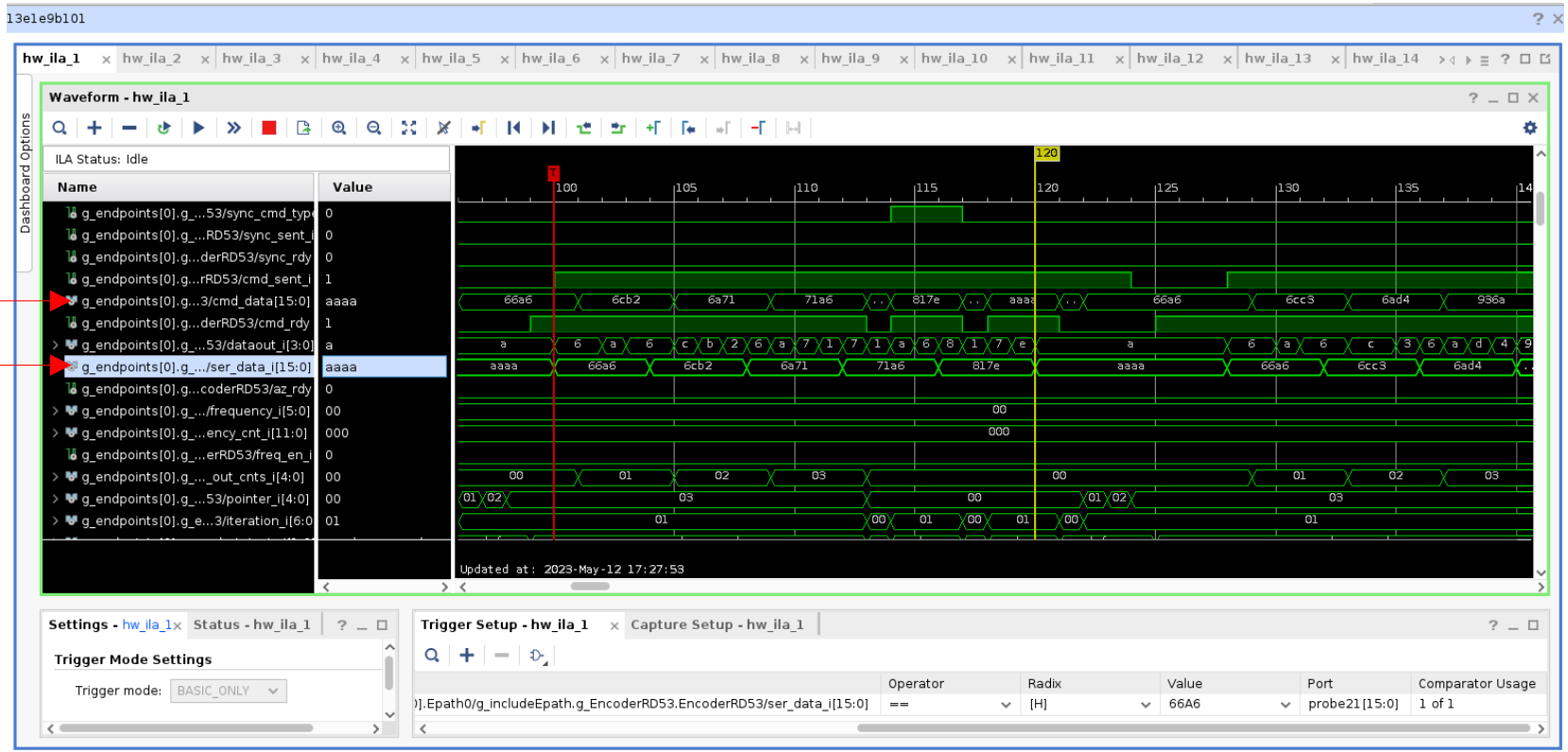
```
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x66a6a66a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
...
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x6a6a6a6a
[16:35:59:295][  info  ][NetioHW::TxCore][28149]: NetioTxCore::writeFifo elink=0 val=0x817eaaaa
```

Command ends here

Tracing with ILA

- Looking at two ILA signals from the ENCRD53 block: cmd_data[15:0] and ser_data[15:0].
- cmd_data[15:0] has the actual command from YARR-NetIO, while ser_data[15:0] is the processed one (with some delay) that gets routed out to the chip.

Shorter commands for setting some global & pixel registers while preparing for the scan.



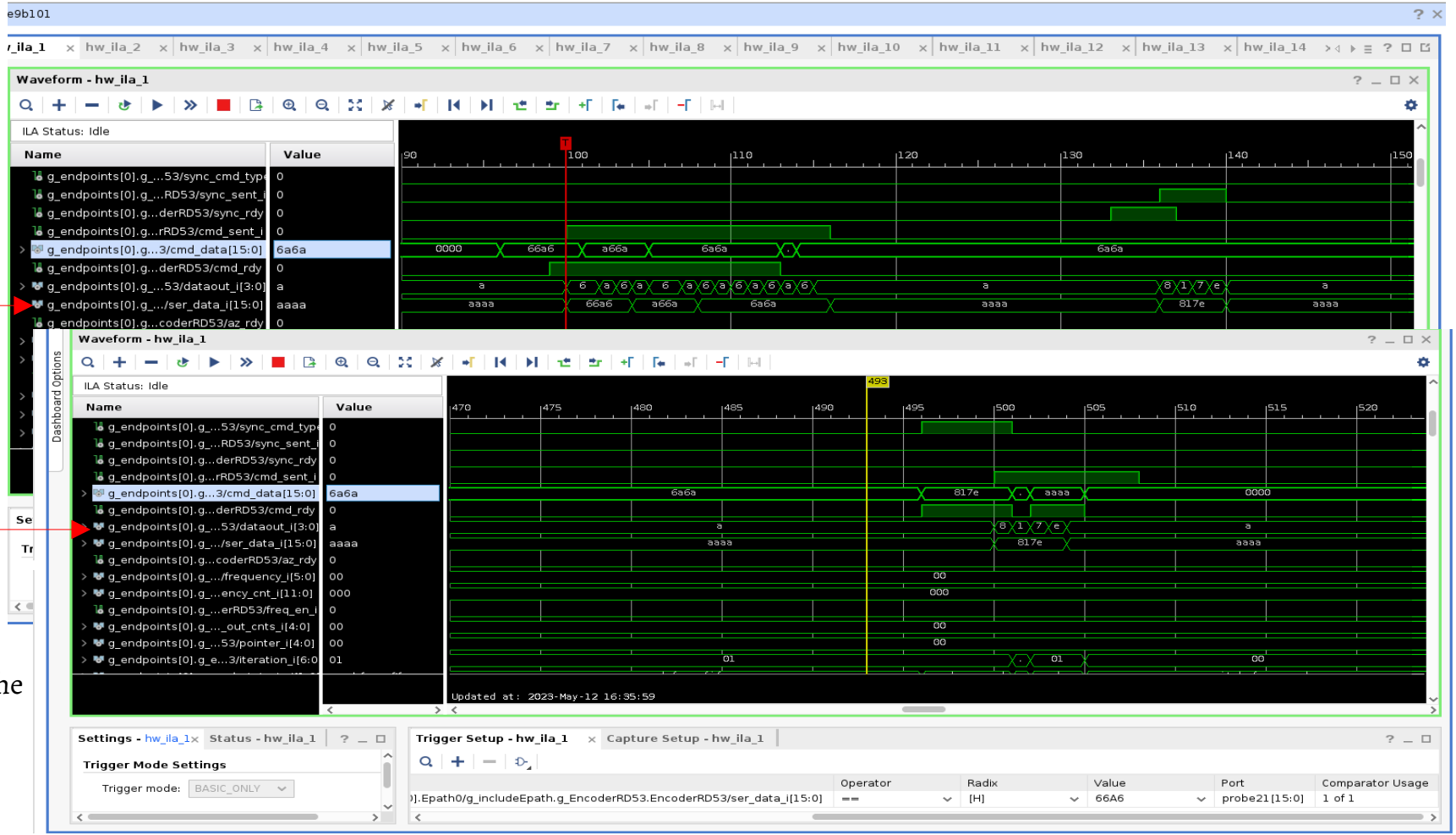
So far so good…

The “problem”

While cmd_data[15:0] is “0x6a6a” for the whole length, ser_data[15:0] is replaced with “0xaaaa”.

Command starts here

Command ends here



- Needed to be fixed in the firmware.
- Opened a JIRA ticket: FLXUSERS-615, fixed.

Various DAQ SW developments

Data processor feedback

New!

- Along with sending out larger commands, allow YARR to continue reading the received data in each iteration, while concurrently looking at the feedback from the DataProcessor.
- The reading cycle will continue until all the incoming data is received.

- MR - otoldaie/YARR:devel_DataFeedback exists for strips, needs to be implemented for pixels.

DataProcessor feedback

Edit | Code

Open **Alex Toldaiev** requested to merge `otoldaie/YARR:devel_DataFe…` into `devel` 9 months ago

Overview 23 | Commits 32 | Pipelines 25 | Changes 19 | All threads resolved!

The `DataProcessor`s get a `ClipBoard<FeedbackInfo> *fbStatus` to send their feedback info about data parsing-processing to `LoopAction`s. The `struct {integer trigger_tag} FeedbackInfo` and `Receiver` class are declared in FeedbackBase.h.

`LoopAction`s that inherit the class `RecieverOfRawDataProcessingFeedback` can connect the map to such Clipboards.

The `ScanHelper` keeps such a map, and sets it for the `ScanFactory : ScanBase` in `buildScan`, and in `buildRawDataProcs` for the FE processors.

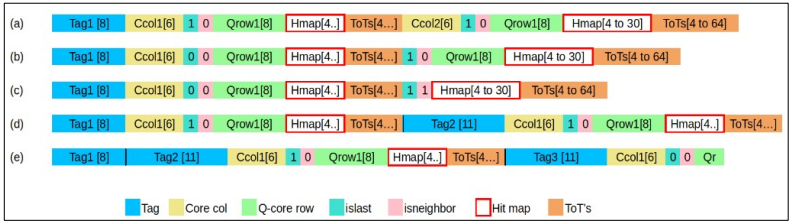
The `StdDataLoop` sends multiple `RawDataContainer`s with `LoopStatus::is_end_of_iteration = false` in one scan iteration (default is `true`). When it considers the iteration has finished, it sends an empty `RawDataContainer` with `is_end_of_iteration = true`, that marks the end of the scan iteration. `DataProcessor`s process this container as any other, create `FrontEndData` with the same `LoopStatus` and pass to `HistogramAlgorithm`. The `HistogramAlgorithm` pushes all incoming data to analysers. When it sees `is_end_of_iteration = true`, it calls `publish`, and resets analysers for new iteration.

The `StdDataLoop` considers the iteration has finished, when it runs out of the total time for iteration (`total_iteration_time_us` in json config, or 500us default), or it received the processing feedback for all expected triggers (`m_trigLoop->getTrigCnt() = trig_count` in the config of the trigger action). It checks for the feedback only if the feedback has been set up (`feedbackFromRawDataProcessing != nullptr`).

The preliminary (not tested yet) real usage of this feedback mechanism is implemented in `StarDataProcessorFeedback` and `StdDataLoopFeedback`.

Implementing feedback for pixels

- Adding a new clipboard to gather feedback, one per front-end.
- Pushing the data received from StdDataLoop class to the feedback clipboard for analyzing data.
- Counting the number of unique tags (or triggered-events) collected. Pushing this feedback in the event data clipboard.



whether the trigger loop has finished sending all the data

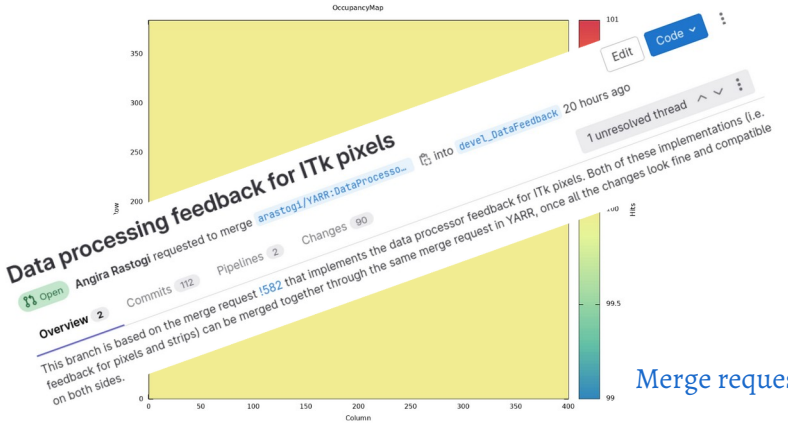
Maximum timeout for reading rx data

Received rx data corresponding to all the triggers

- Finally, StdDataLoop checks the feedback condition: "!trigger_is_done || (there_is_still_time && !received_all_triggers)"
  - If false, then moves onto next core column loop.
  - If true, performs another rx read cycle.

Digital scan, Rd53bParMaskLoop, Mask stages = 64, Injections = 100, firmware-triggerring

No feedback (latest devel branch)		With full feedback implementation (specCard, birdspider PC)		With full feedback implementation (FELIX-NetIO, littleoakhorn PC)	
real	0m10.460s	real	0m10.794s	real	0m34.570s
user	0m9.881s	user	0m7.334s	user	0m12.264s
sys	0m0.916s	sys	0m2.158s	sys	0m4.464s



Merge request

Summary & immediate next steps

- My qualification task has officially ended*, but I'll still continue to contribute on the DAQ SW developments actively.
- Many DAQ SW-related developments have been done over the past year at LBNL, yet so much more to do.

Some immediate action items are -

- Optimizing the timing and performance of the data processor feedback mechanism.
- Running all the standard QC scans, benchmark the timing in the latest & greatest setup.
- Reading service frames through FELIX-NetIO – still open but have some ideas now.
- Fix software-triggering in FELIX.

Longer-term plans

July 2023

- Test the data transmission using FELIX for the 8-ITk Pixel quad module serial power chain.
- Test the data transmission using FELIX for the 5-ITk Pixel triplet module serial power chain (potentially for luminosity measurements via cluster counting (Pixel Lumi Rings, general IS).
  Both of these above will require us to get extra FELIX card, new optoboard, adapter boards to connect with the modules more fiber optics etc.

- Medium-scale system-like test stand with FELIX for Pixels (& strips) to support DAQ development during commissioning and operation. Eventual goal would be to support nightly tests via CI.

- Continued support to low-cost lab system readout hardware to support online software development (lower threshold for contribution from local institutes to CERN operations).