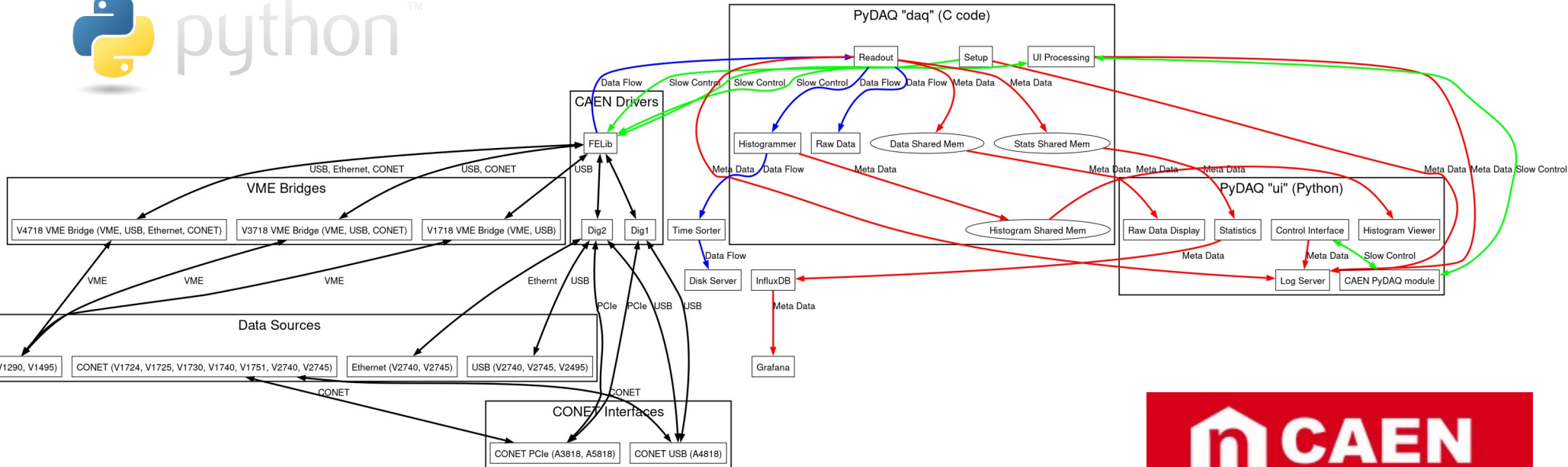


PyDAQ for FAUST

Status update – April 2025



Science and
Technology
Facilities Council



STFC Nuclear Physics Group CAEN Readout
Software Architecture (planned)



PyDAQ Overview

- PyDAQ is the new software framework being developed in Daresbury Lab Nuclear Physics Group.
- Aim is to provide reusable software components to support current and future detector and electronics developments.
- Core design principles will be familiar to users of “MIDAS”:
 - Separate programs for UI, data acquisition, writing to disk.
 - C used for data acquisition code and disk server. Chosen because of availability on all hardware platforms from micro-controllers to rack mount servers, and speed.
 - Browser based UI generated from code running in an interpreted programming language.

NPG Python DAQ: [Home](#) [Logs](#) [Help](#) [Histograms](#) [Statistics](#) Components: [RPIcamera](#) [Timepix3](#) [ZyDF](#) [CAEN](#)



Welcome to the NPG Python DAQ User Interface

Status

Time is 2025-04-13 17:08:13.592165!

Software version is: 0.1.4 ()



← Note: not the same “MIDAS” as mentioned in Dan’s SHARC talk yesterday.

PyDAQ Overview



- Design changes include:
 - Move from TCL to Python for UI development.
 - Modern source code control practices (Git/Gitlab)
 - Support for multiple firmware versions planned from beginning to reduce long term support workload.
 - Software testing to facilitate long term support and avoid regressions (this part is aspirational at this point).
- Projects currently in development
 - New readout system for CERN's Timepix3 ASIC.
 - Readout of new "27xx" generation of CAEN digitisers.
 - Continued support for legacy CAEN digitisers in use within UK nuclear physics community,

Current Status for CAEN Devices

- Existing capabilities:
 - Connect to modules with any interface supported by CAEN “FELib”.
 - Read/write firmware parameters for any modules supported by either generation of CAEN “DigitizerLib”.
 - Read data buffers from modules and write to disk.
 - Online tracking of statistics and histograms (In progress).
- Supported interfaces:
 - A4818 USB3 → CONET optical link
 - A3818 and A5818 PCIe → CONET optical link
 - Direct USB
 - Ethernet



Current Status for CAEN Devices

- CAEN digitiser libraries provide semantic information on parameters. This allows automatic generation of required UI elements.
- Online statistics and histograms need custom code for each module/firmware combination.
- Works with V, VX or DT versions of modules.

Module	UI	Readout	PHA	PSD	WO
V1724	✓	✓	✗	✗	✗
V1725	✓	✓	✓	✗	✗
V1730	✓	✓	✓	✗	✗
V1740	✓	✓	✗	✗	✗
V1742	✓	✓	✗	✗	✗
V1751	✓	✓	✗	✗	✗
V2740/45	✓	✓	✓	✗	✗

✓ Tested

✓ Expected working

✗ Not working yet

✗ Not available



Current Status for CAEN Devices

- CAEN digitiser libraries provide semantic information on parameters. This allows automatic generation of required UI elements.
- Online statistics and monitoring are available for most devices.
- Works with V, VX or VME modules.

- Modules not supported by CAEN FELib will not work “out of the box”.
- It is possible to build in support for such devices by constructing our own protocol from low level VME access.
- So far this is only planned for V1290 TDC

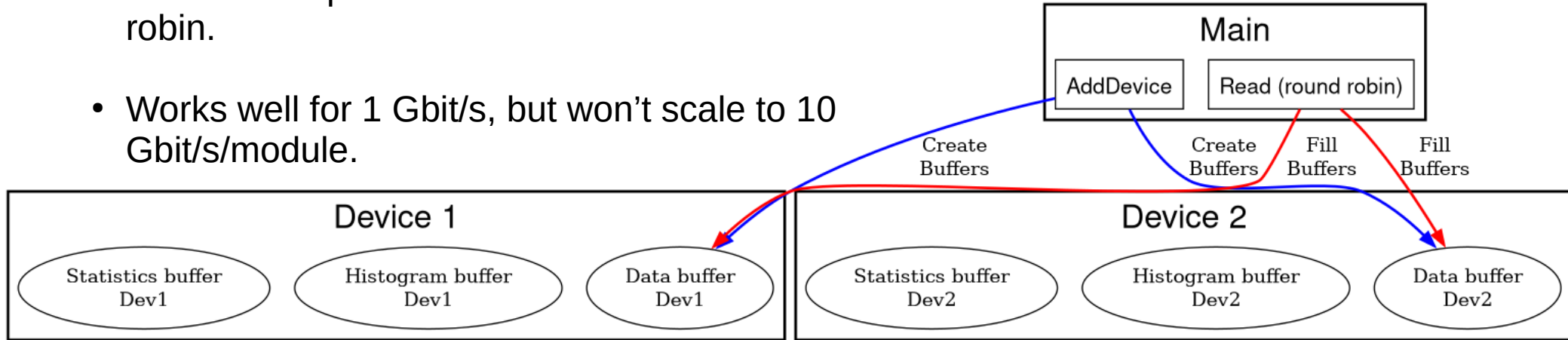
Module	U	U	U	U	WO
V1724	✓				✗
V1725	✓				✗
V1730	✓				✗
V1740	✓				✗
V1742	✓	✓	✗	✗	✗
V1751	✓	✓	✗	✗	✗
V2740/45	✓	✓	✓	✗	✗

- ✓ Tested
- ✓ Expected working
- ✗ Not working yet
- ✗ Not available



PyDAQ- CAEN Readout Architecture

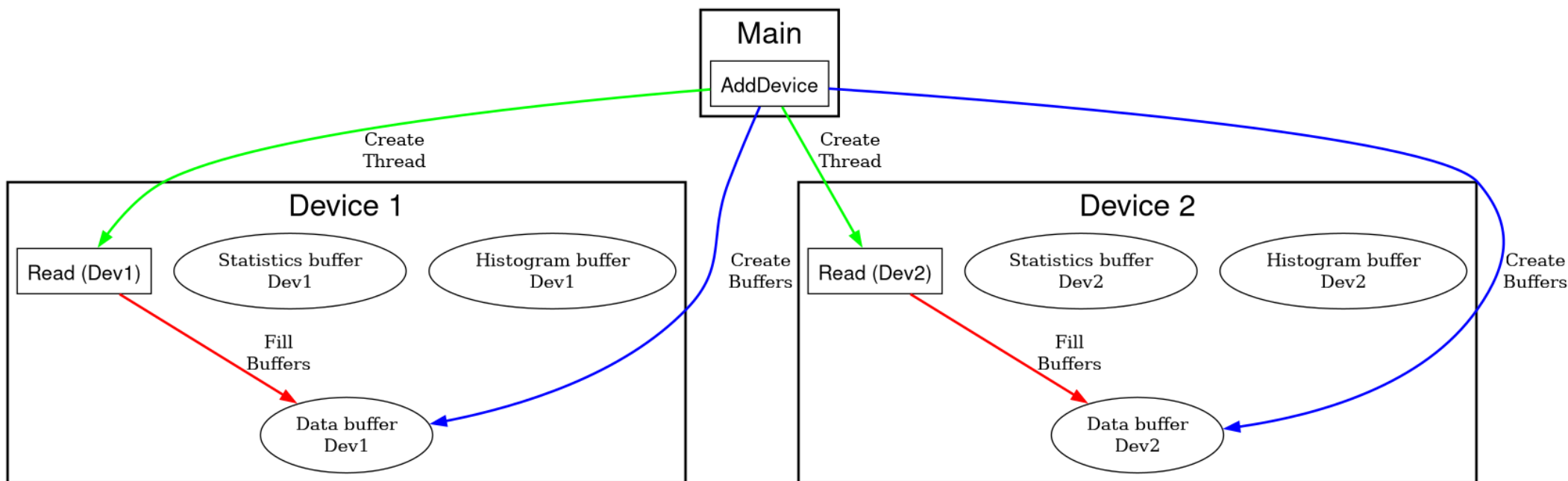
- Currently readout runs in a single thread on the DAQ PC.
- Each device added has unique buffers for data, histograms and statistics.
- Main thread polls devices for data in a round robin.
- Works well for 1 Gbit/s, but won't scale to 10 Gbit/s/module.



CAEN Readout Logic

PyDAQ- CAEN Readout Architecture

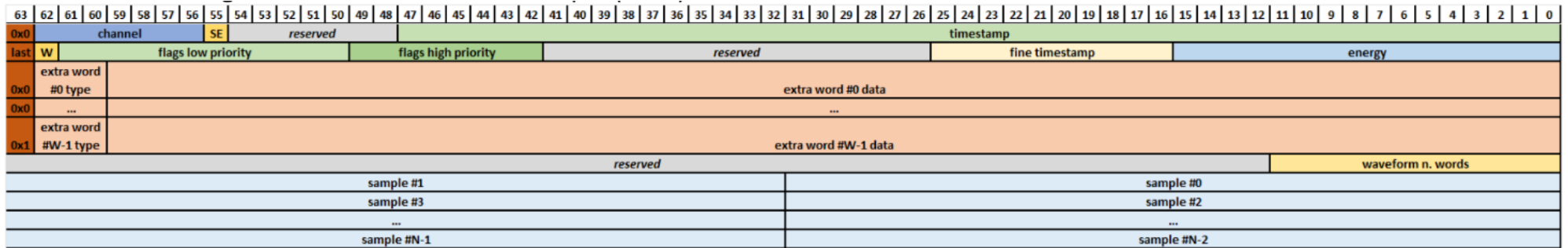
- Separation of buffers facilitates a model where a new readout thread is launched for each device.
- This approach should scale to whatever speed the hardware is capable of.
- But we aim to get everything else working first...



CAEN Readout Logic (Threaded)

Future Plans

- Custom data format:
 - (Currently just writing raw data buffers from modules to disk from readout code).
 - Captures all information from all modules firmwares.
 - Flexible to support data from future devices/projects.
 - 64bit 1ns contiguous time stamp included with all data items to facilitate ordering and event building.
 - Real “wall clock” time and date included in data block headers.



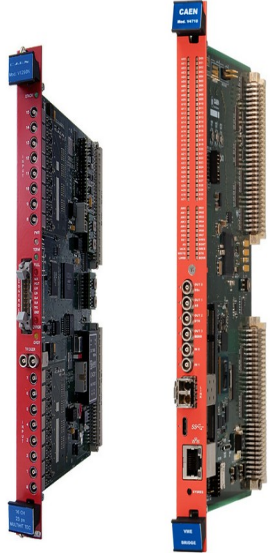
Future Plans

- Custom data format:
 - (Currently just writing raw data buffers from modules to disk from readout code).
 - Captures all information from all modules firmwares.
 - Flexible to support data from future devices/projects.
 - 64bit 1ns contiguous time stamp included with all data items to facilitate ordering and event building.
 - Real “wall clock” time and date included in data block headers.
- Time ordering program supporting multiple sources.
- Disk server supporting multiple sources.
- Statistics logging to InfluxDB/Grafana or similar.
- Multi-threaded readout to allow maximum data rate to be achieved in all modules.



“To Do” ahead of FAUST Tests

- Testing in UK labs.
- Clock sync with FRIB
 - Agreed clock frequency (50 MHz?) and source.
 - Appropriate module PLL configuration.
 - Sync source (CAEN modules only reset to $t=0$, no absolute time reference).
 - Cables.
- DSP parameter optimisation.
- Trigger scheme. Easy if all channels run independently but need to test if:
 - Using Veto/Enable from accelerator.
 - Propagating triggers between modules via GPIO/LVDS.
- Data sorting
 - Currently data is written in “raw” buffers received from the modules.
 - Code used for online histogramming is reusable, I can provide an example code which reads files and prints event data to screen.
 - Possibly better if someone from the physics side handles later processing e.g. ROOT trees.



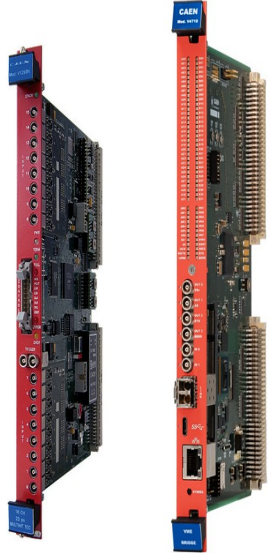
“To Do” ahead of FAUST Tests

- Testing in UK labs.
- Clock sync with FRIB
 - Agreed clock frequency
 - Appropriate modules
 - Sync source (CAEN)
 - Cables.
- DSP parameter optimization
- Trigger scheme. Easy to implement
 - Using Veto/Enable
 - Propagating triggers
- Data sorting
 - Currently data is written to files
 - Code used for online histogramming is reusable, I can provide an example code which reads files and prints event data to screen.
 - Possibly better if someone from the physics side handles later processing e.g. ROOT trees.

Good News for FAUST!

We are planning some in-beam tests at RIKEN RIBF in June using both 1730 and 2740 modules.

Hopefully some bugs will be ironed out in advance of FAUST tests.



Thank You

- Special thanks to University of Liverpool and University of York for recent help testing the software on their systems.
- We are keen to get feedback so please get in touch if you'd like to try it out.

Questions?

Save Restore Functions

- System configuration and module settings are saved to JSON formatted text file.
- These are easily manipulated using a variety of languages and GUI tools.

```
{
  "Config": {
    "Module1": "dig1://caen.internal/usb_a4818?link_num=22077&conet_node=0",
    "Module2": "dig1://caen.internal/usb_a4818?link_num=22077&conet_node=1",
    "Module3": "dig1://caen.internal/usb_a4818?link_num=22077&conet_node=2",
    "Module4": "dig1://caen.internal/usb_a4818?link_num=22077&conet_node=3"
  },
  "Settings": {
    "Module1": {
      "ch": {
        "0": {
```

```
      "description": {
        "handle": 1063,
        "value": "Threshold"
      },
      "expuom": {
        "handle": 1072,
        "value": "0"
      },
      "handle": 1062,
      "increment": {
        "handle": 1070,
        "value": "1"
      },
      "maxvalue": {
        "handle": 1069,
        "value": "16383"
      },
      "minvalue": {
        "handle": 1068,
        "value": "0"
      },
      "setinrun": {
        "handle": 1066,
        "value": "true"
      },
      "uom": {
        "handle": 1071,
        "value": "LSB"
      },
      "value": "100"
```

GUI Features

Device List

Name	Model	FW Type	ROC FW Version	AMC FW Version	FPGA FW Version	Is Active?
------	-------	---------	----------------	----------------	-----------------	------------

Send Command to Control Server

Control Device List

Add devices by name and path. Reset will stop any ongoing run and close all devices

Example path: dig1://caen.internal/usb_a4818?link_num=22077&conet_node=0

Device Name

Device Path

GUI Features

Device List

Name	Model	FW Type	ROC FW Version	AMC FW Version	FPGA FW Version	Is Active?
Module1	VX2740B	DPP_PHA	unknown	unknown	1.0.102	True

Send Command to Control Server

Control Device List

Add devices by name and path. Reset will stop any ongoing run and close all devices

Example path: dig1://caen.internal/usb_a4818?link_num=22077&conet_node=0

Device Name

Device Path

Add Device

Reset DAQ

GUI Features

Parameter path /par

Parameter	Value	Access	Data Type	Update
acquisitionstatus	8	READ_ONLY	STRING	
adc_nbit	16	READ_ONLY	NUMBER	
adc_samplerate	125	READ_ONLY	NUMBER	
boardready	True	READ_ONLY	STRING	
boardvetopolarity	ActiveHigh	READ_WRITE	STRING	ActiveHigh ▾
boardvetosource	Disabled	READ_WRITE	STRING	Disabled ▾
boardvetowidth	24	READ_WRITE	NUMBER	24 ▾
busyinsource	SIN	READ_WRITE	STRING	SIN ▾
clocksource	Internal	READ_WRITE	STRING	Internal ▾
cupver	2025012205	READ_ONLY	STRING	
dacoutchselect	0	READ_WRITE	NUMBER	0 ▾
dacoutmode	Static	READ_WRITE	STRING	Static ▾
dacoutstaticlevel	8192	READ_WRITE	NUMBER	8192 ▾
dutycyclesensdc	14.8	READ_ONLY	NUMBER	
enautodisarmacq	False	READ_WRITE	STRING	False ▾
enclockoutfp	False	READ_WRITE	STRING	False ▾
endatareduction	False	READ_WRITE	STRING	False ▾

GUI Features

Parameter path /par

Parameter	Value	Access	Data Type	Update
acquisitionstatus	8	READ_ONLY	STRING	
adc_nbit	16	READ_ONLY	NUMBER	
adc_samplerate	125	READ_ONLY	NUMBER	
boardready	True	READ_ONLY	STRING	
boardvetopolarity	ActiveHigh	READ_WRITE	STRING	ActiveHigh ▾
boardvetosource	Disabled	READ_WRITE	STRING	Disabled ▾
boardvetowidth	24	READ_WRITE	NUMBER	24 ▾
busyinsource	SIN	READ_WRITE	STRING	SIN ▾
clocksource	Internal	READ_WRITE	STRING	Internal ▾
cupver	2025012205	READ_ONLY	STRING	
dacoutchselect	0	READ_WRITE	NUMBER	0 ▾
dacoutmode	Static	READ_WRITE	STRING	Static ▾
dacoutstaticlevel	8192	READ_WRITE	NUMBER	8192 ▾
dutycyclesensdc	14.8	READ_ONLY	NUMBER	
enautodisarmacq	False	READ_WRITE	STRING	False ▾
enclockoutfp	False	READ_WRITE	STRING	False ▾
endatareduction	False	READ_WRITE	STRING	False ▾

GUI Features

Parameter path /par

Parameter	Value	Access	Data Type	Update
acquisitionstatus	8	READ_ONLY	STRING	
adc_nbit	16	READ_ONLY	NUMBER	
adc_samplerate	125	READ_ONLY	NUMBER	
boardready	True	READ_ONLY	STRING	
boardvetopolarity	ActiveHigh	READ_WRITE	STRING	ActiveHigh ▾
boardvetosource	Disabled	READ_WRITE	STRING	Disabled ▾
boardvetowidth	24	READ_WRITE	NUMBER	24 ▾
busyinsource	SIN	READ_WRITE	STRING	SIN ▾
clocksource	Internal	READ_WRITE	STRING	Internal ▾
cupver	2025012205	READ_ONLY	STRING	
dacoutchselect	0	READ_WRITE	NUMBER	0 ▾
dacoutmode	Static	READ_WRITE	STRING	Static ▾
dacoutstaticlevel	8192	READ_WRITE	NUMBER	8192 ▾
dutycyclesensdcdc	14.8	READ_ONLY	NUMBER	
enautodisarmacq	False	READ_WRITE	STRING	False ▾
enclockoutfp	False	READ_WRITE	STRING	False ▾
endatareduction	False	READ_WRITE	STRING	False ▾

GUI Features

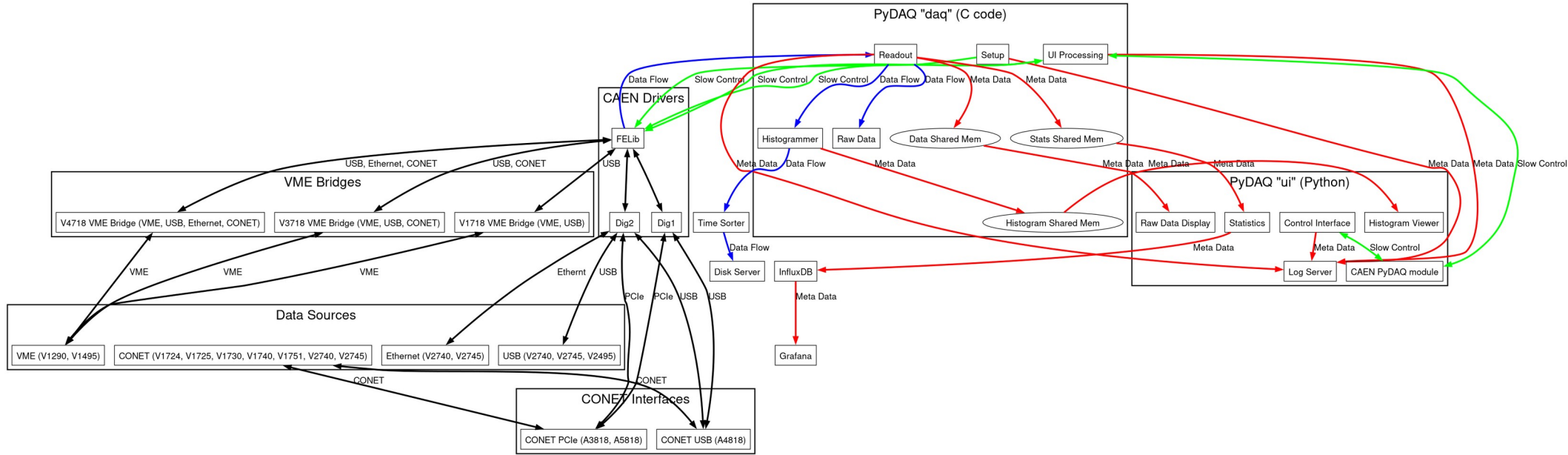
itlbmask 0 READ_WRITE STRING 0 Toggle Hex Tool

0x0000000000000000

63	<input type="checkbox"/>	62	<input type="checkbox"/>	61	<input type="checkbox"/>	60	<input type="checkbox"/>	59	<input type="checkbox"/>	58	<input type="checkbox"/>	57	<input type="checkbox"/>	56	<input type="checkbox"/>	55	<input type="checkbox"/>	54	<input type="checkbox"/>	53	<input type="checkbox"/>	52	<input type="checkbox"/>	51	<input type="checkbox"/>	50	<input type="checkbox"/>	49	<input type="checkbox"/>						
48	<input type="checkbox"/>	47	<input type="checkbox"/>	46	<input type="checkbox"/>	45	<input type="checkbox"/>	44	<input type="checkbox"/>	43	<input type="checkbox"/>	42	<input type="checkbox"/>	41	<input type="checkbox"/>	40	<input type="checkbox"/>	39	<input type="checkbox"/>	38	<input type="checkbox"/>	37	<input type="checkbox"/>	36	<input type="checkbox"/>	35	<input type="checkbox"/>	34	<input type="checkbox"/>						
33	<input type="checkbox"/>	32	<input type="checkbox"/>	31	<input type="checkbox"/>	30	<input type="checkbox"/>	29	<input type="checkbox"/>	28	<input type="checkbox"/>	27	<input type="checkbox"/>	26	<input type="checkbox"/>	25	<input type="checkbox"/>	24	<input type="checkbox"/>	23	<input type="checkbox"/>	22	<input type="checkbox"/>	21	<input type="checkbox"/>	20	<input type="checkbox"/>	19	<input type="checkbox"/>						
18	<input type="checkbox"/>	17	<input type="checkbox"/>	16	<input type="checkbox"/>	15	<input type="checkbox"/>	14	<input type="checkbox"/>	13	<input type="checkbox"/>	12	<input type="checkbox"/>	11	<input type="checkbox"/>	10	<input type="checkbox"/>	9	<input type="checkbox"/>	8	<input type="checkbox"/>	7	<input type="checkbox"/>	6	<input type="checkbox"/>	5	<input type="checkbox"/>	4	<input type="checkbox"/>						
3	<input type="checkbox"/>	2	<input type="checkbox"/>	1	<input type="checkbox"/>	0	<input type="checkbox"/>																												

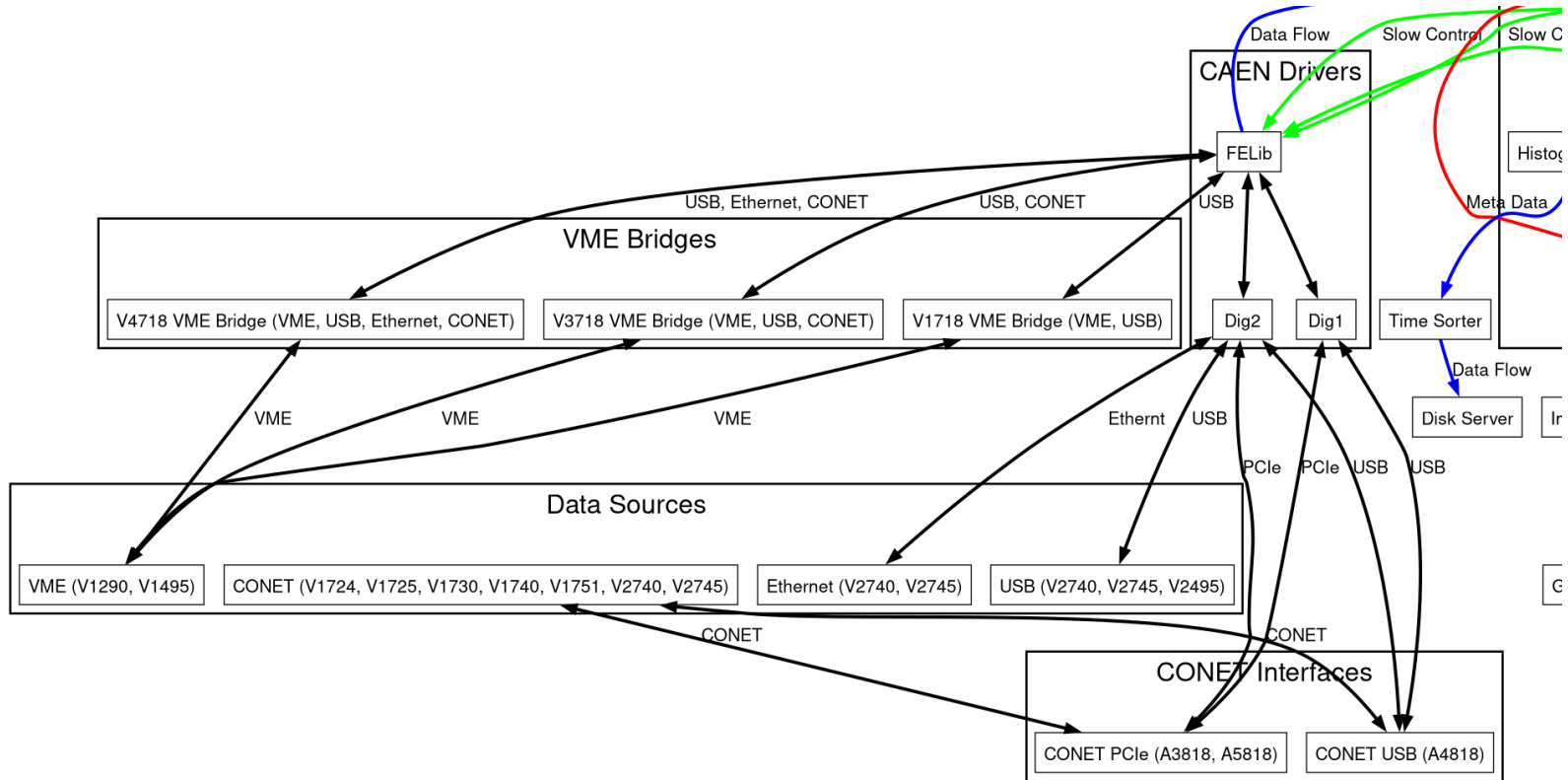
Load Value

Software Architecture



STFC Nuclear Physics Group CAEN Readout Software Architecture (planned)

Software Architecture



Software Architecture

