Judgement System: Qualifying a signal set against a limit mask

Peter Milne
D-TACQ Solutions
www.d-tacq.com

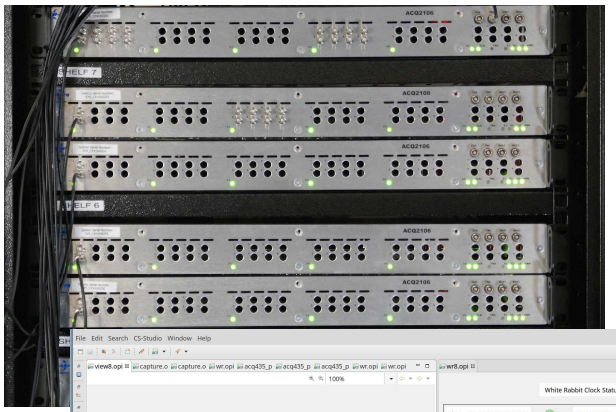
High Performance Simultaneous Data Acquisition

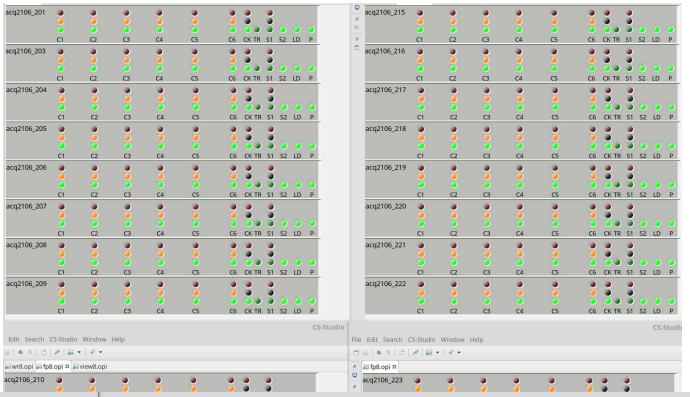
D-TACQ Solutions

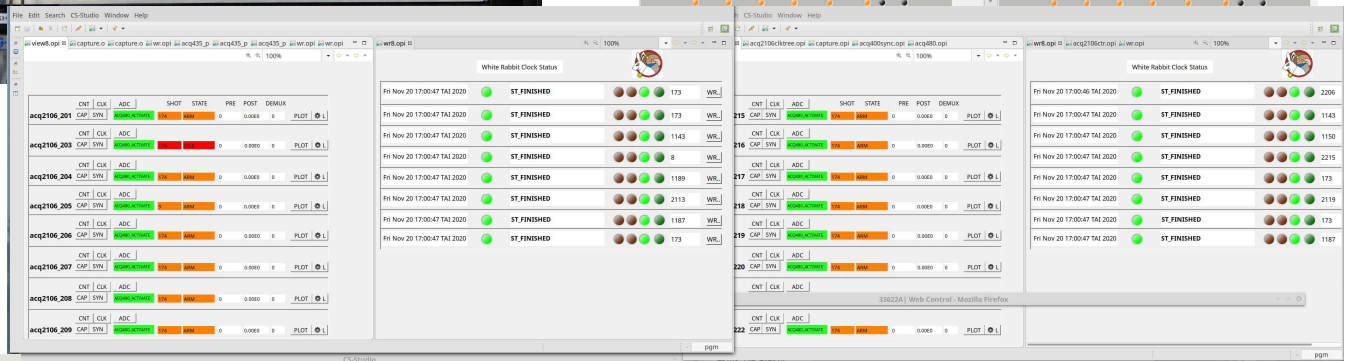
Networked DAQ, 4-192 channels, 1k-80MSPS

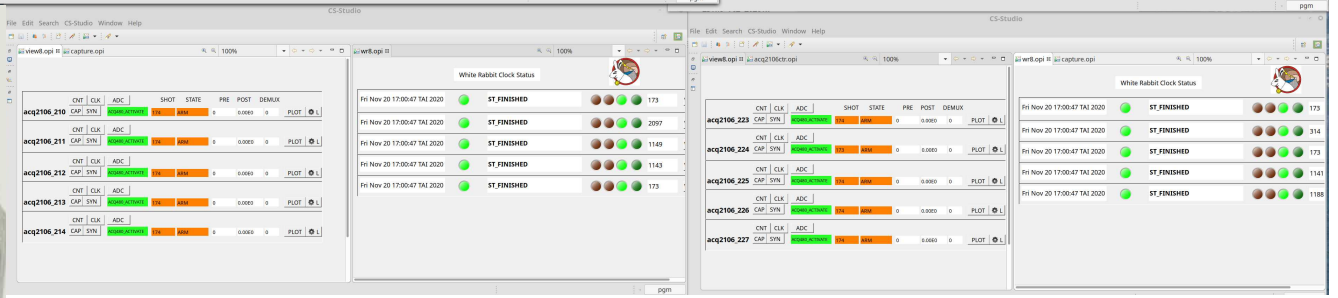
example: Distributed Oscilloscope 1248 channels, White Rabbit timing:

https://www.d-tacq.com/news52.shtml









High Performance Simultaneous Data Acquisition

D-TACQ Solutions

Example System:
CPSC2 : Corrector PS Controller
https://www.d-tacq.com/news48.shtml

Judgement System Requirement

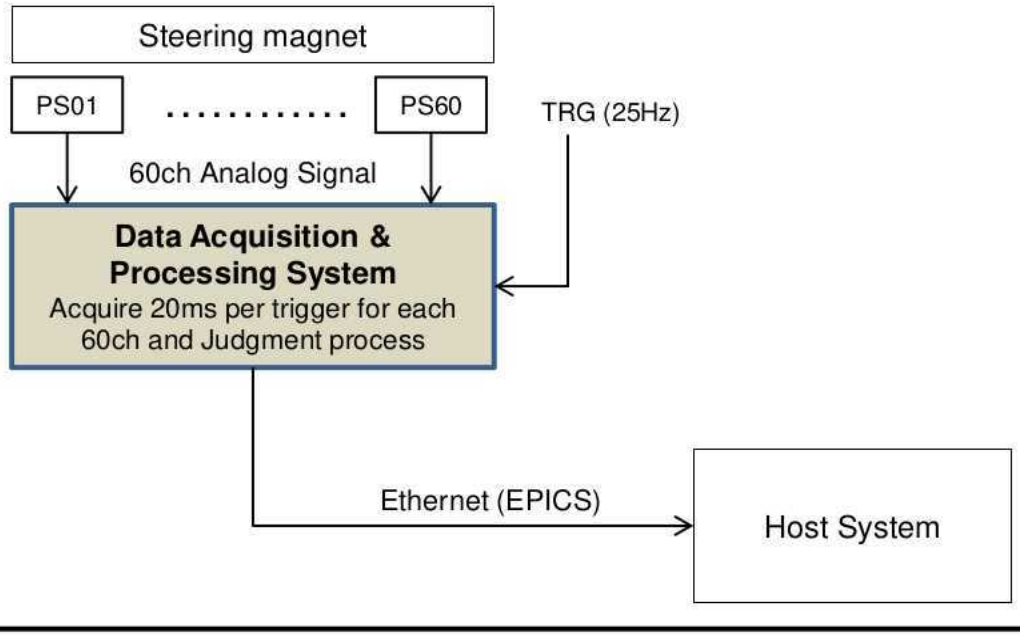
Specification

Data Acquisition System

- Sampling Rate: 50KHz
- Resolution: Not specified
- Input signal: Current or Voltage
- B/W: 25Hz
- Number of Ch: 60ch (Max.)
- Acquisition period: 20ms (1000 samples per Trigger)
- Trigger frequency: 25Hz
- Output Interface: Gigabit Ethernet
- Output format: EPICS

Quantity: 2 sets

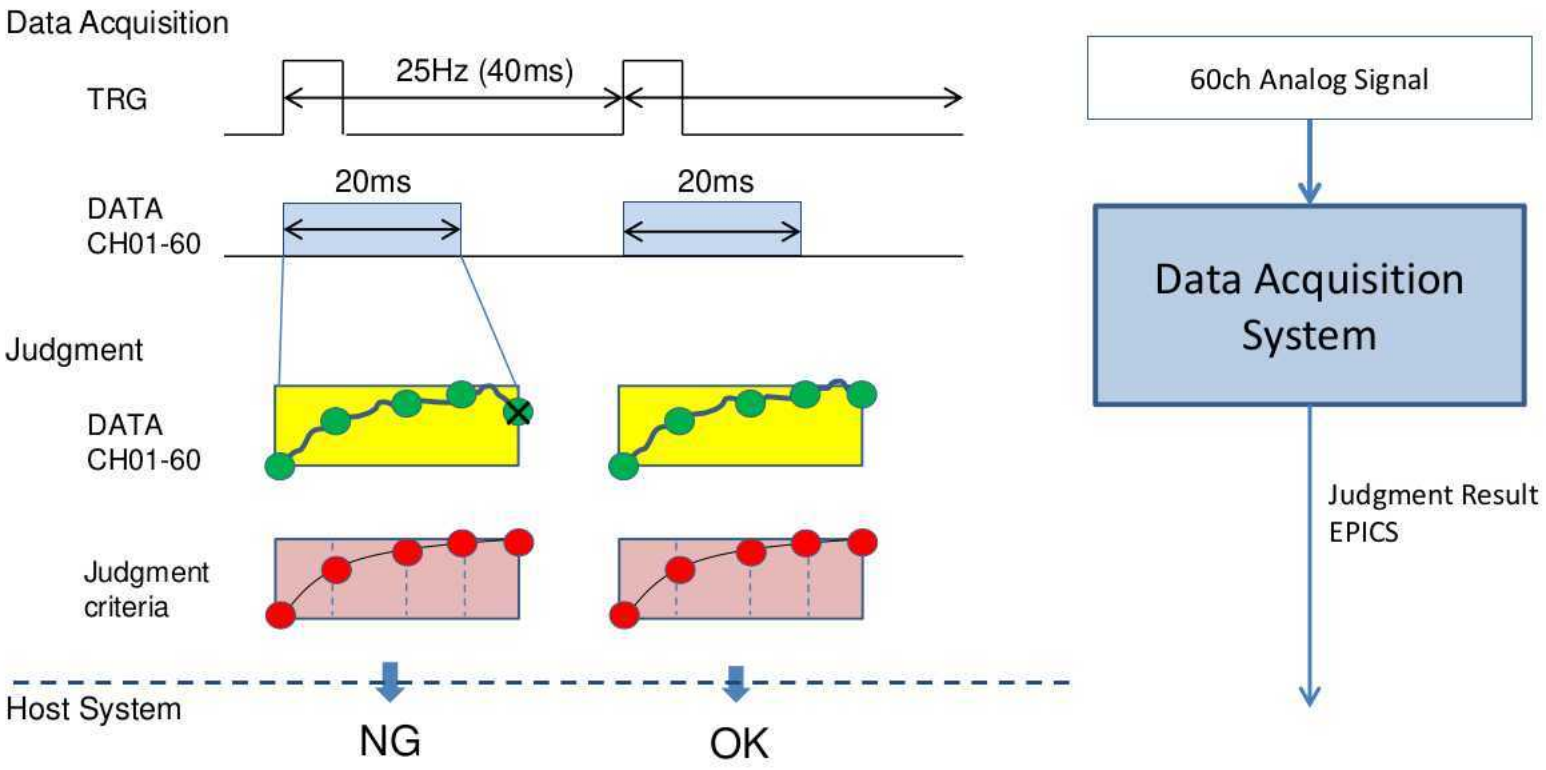
System Block Diagram



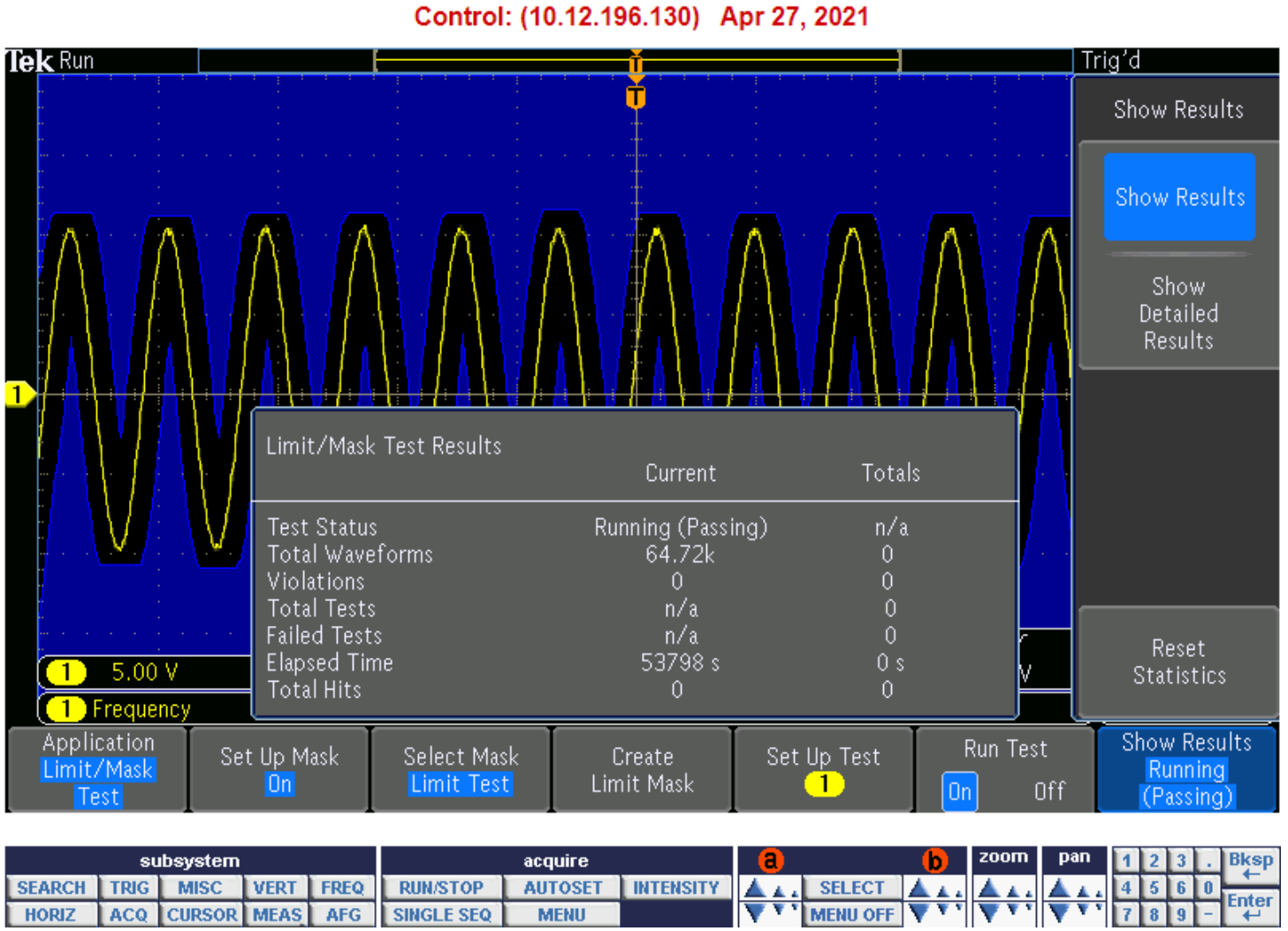
Processing Flow

1) Digitally sample and capture 60ch analog signal for each trigger.
2) Determines whether the captured signal satisfies the judgment criteria by the analog signal of 60ch by software.
3) Send the judgment result to the host system via Ethernet with EPICS.

Requirement #2



Using a mask to test a signal..



Judgement System .. key features

Magnetic Coil Current Monitor:

64 simultaneous Analog Inputs,
16 bit, 50kSPS/channel.
Burst Mode: 20ms on a 40ms (25Hz) trigger
Burst Length: 1024 samples with timestamp

Hardware:

ACQ1002+2xACQ423

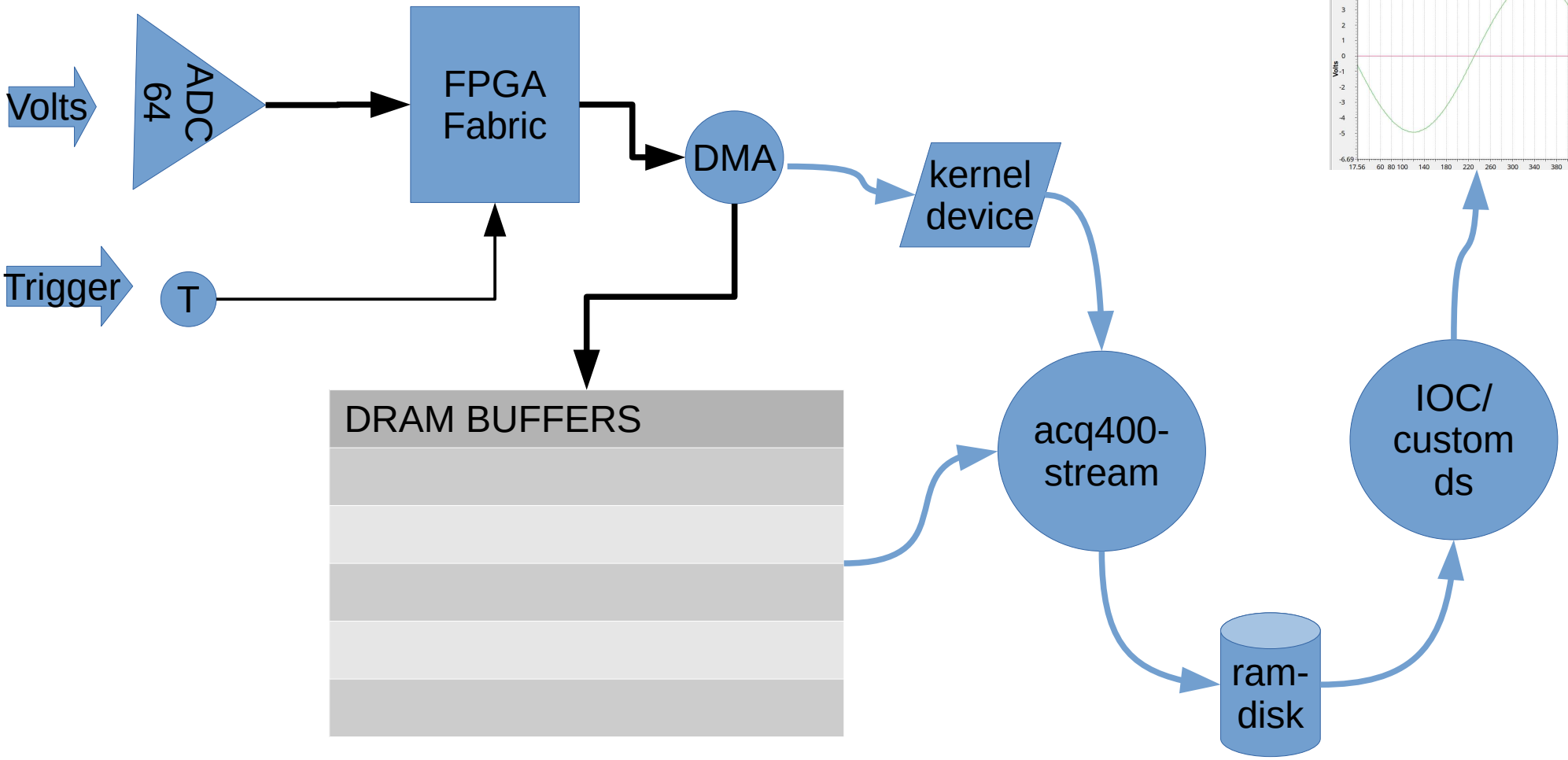
Outputs:

All data is streamed on Ethernet to archive:
25 * 64 * 2 * 1024 = 3.3MB/s (slow!)
Internal EPICS “Judgment Process”
compares each pulse with a mask (judgment mask)
outputs a 64 element Pass/Fail WF record
The record is updated at 25Hz

Implementation: HW

2 x ACQ1001+2xACQ423ELF-32-200-16
2 x 64 channels, 50kSPS.

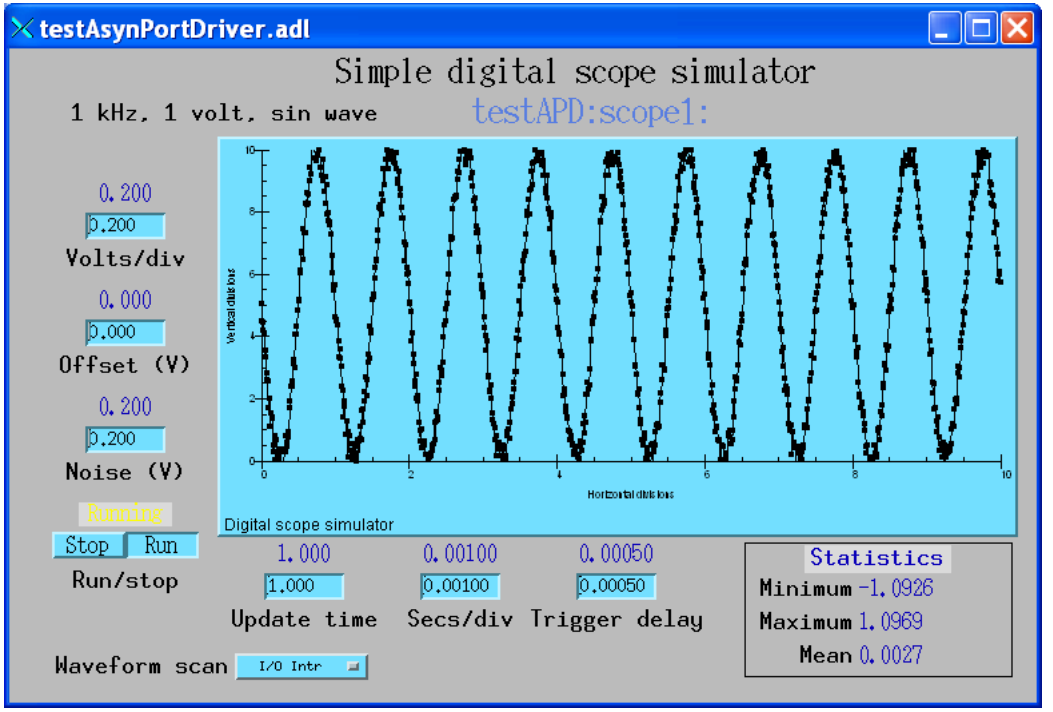
Implementation Software ~2005



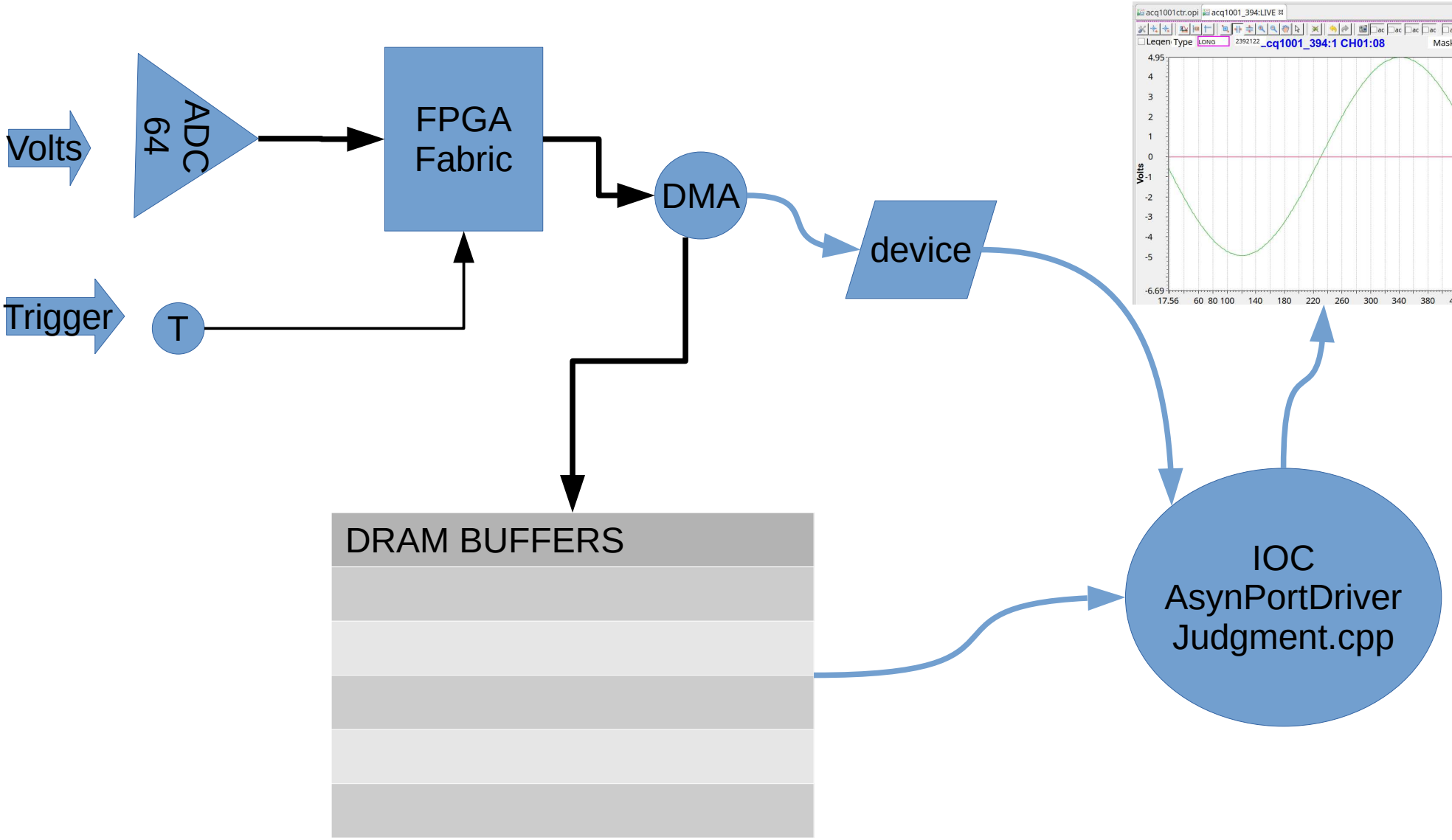
AsynPortDriver

https://epics.anl.gov/modules/soft/asyn/R4-12/asynPortDriver.html

- asynPortDriver is a base C++ class that is designed to greatly simplify the task of writing an asyn port driver. It handles all of the details of registering the port driver, registering the supported interfaces..
- Working Scope Simulator Demo.
  - FAST!



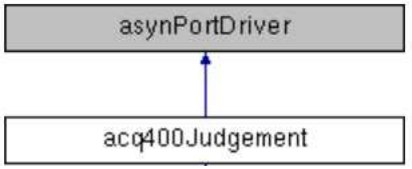
Judgement Implementation



acq400_judgement.cpp April 14

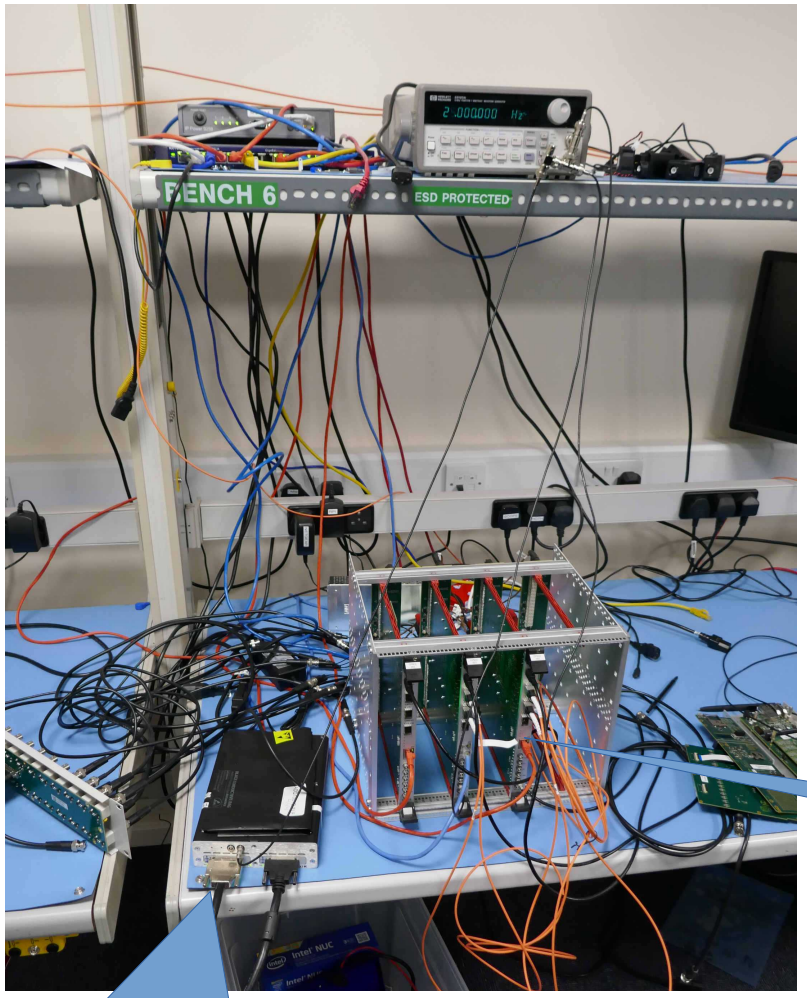
https://github.com/D-TACQ/acq400ioc/commits/epics7

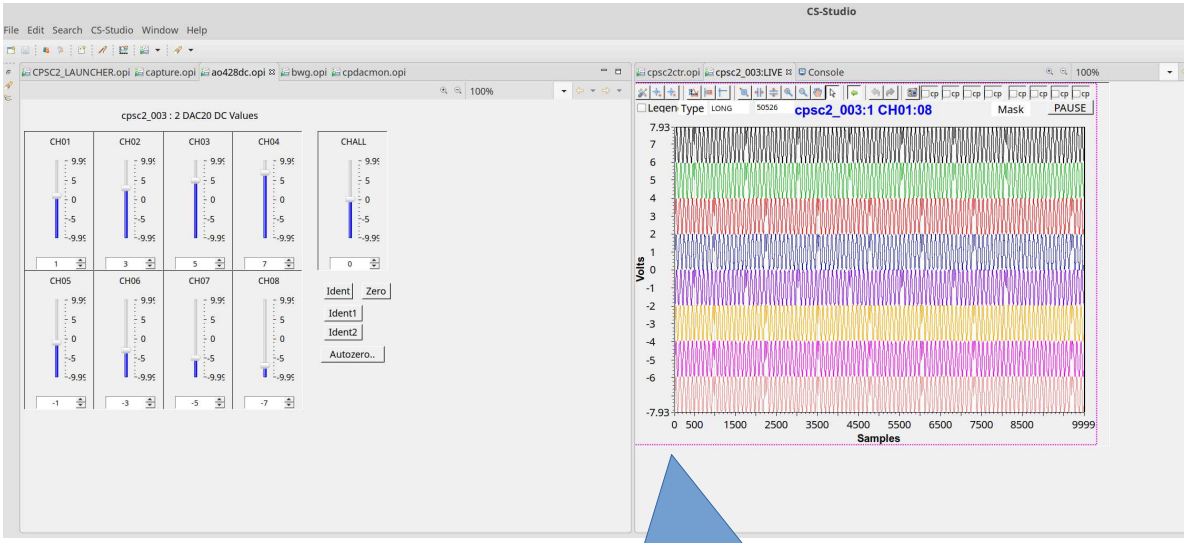
```
acq400Judgement::acq400Judgement(const char* portName, int _nchan, int _nsam):
      asynPortDriver(portName, _nchan,
                     asynInt32Mask | asynFloat64Mask ...
```




```
 int acq400Judgement::factory(const char *portName, int maxPoints, int nchan)
 {
       new acq400Judgement(portName, maxPoints, nchan);
       return(asynSuccess);
 }


void acq400Judgement::task()
{
       int fc = open("/dev/acq400.0.bq", O_RDONLY);

       while((ib = getBufferId(fc)) >= 0){
             handle_burst(ib*2,               0);
             handle_burst(ib*2+1,    nsam*nchan);
       }
}


bool acq400Judgement::calculate(epicsInt16* raw, const epicsInt16* mu, const epicsInt16* ml)
{
      for (int isam = 0; isam < nsam; ++isam){
            for (int ic = 0; ic < nchan; ++ic){
                  int ib = isam*nchan+ic;
                  epicsInt16 xx = raw[ib];

                  if (xx > mu[ib] || xx < ml[ib]){
                        FAIL_MASK32[ic/32] |= 1 << (ic&0x1f);
                        RESULT_FAIL[ic+1] = 1;
                        fail = true;
                  }
            }
      }
      return onCalculate(fail);
}
```

Demo1 Hardware





CPSC2:
8CH AWG for DEMO!

CPSC2:
8CH 20bit PSU controller.

ACQ1002+2xACQ423
64AI x 200kSPS, 16 bit

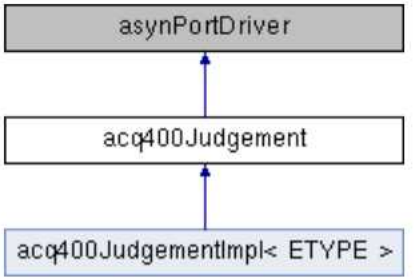
#1 Judgement inside

Enhancements Requested

- Conditional Updates
  - UPDATE_ALWAYS, UPDATE_ONFAIL ..
- Window Function
  - Apply mask over a set portion of the waveform
    - WIN:L, WIN:R : reduced timebase
- What about 24 bit ADC?.
  - 32 bit data.
    - Use Templates.

Specialize with template

```
template <class ETYPE> class acq400JudgementImpl : public acq400Judgement {

};


/* factory */
int acq400Judgement::factory(const char *portName, int maxPoints, int nchan, unsigned data_size){
    switch(data_size){
    case sizeof(short):
        new acq400JudgementImpl<epicsInt16> (portName, maxPoints, nchan);
        return(asynSuccess);
    case sizeof(long):
        new acq400JudgementImpl<epicsInt32> (portName, maxPoints, nchan);
        return(asynSuccess);
}

  bool calculate(ETYPE* raw, const ETYPE* mu, const ETYPE* ml)
  {
        for (int isam = 0; isam < nsam; ++isam){
              for (int ic = 0; ic < nchan; ++ic){
                    int ib = isam*nchan+ic;
                    ETYPE xx = raw[ib];

                    if (isam >= WINL[ic] && isam <= WINR[ic]){      // make Judgement inside window
                          if (xx > mu[ib] || xx < ml[ib]){
                                FAIL_MASK32[ic/32] |= 1 << (ic&0x1f);
                                RESULT_FAIL[ic+1] = 1;
                                fail = true;
                          }
                    }
              }
        }
        return onCalculate(fail);
  }
```



https://github.com/D-TACQ/acq400ioc/blob/epics7/acq400iocApp/src/acq400_judgement.cpp

Template Specialization

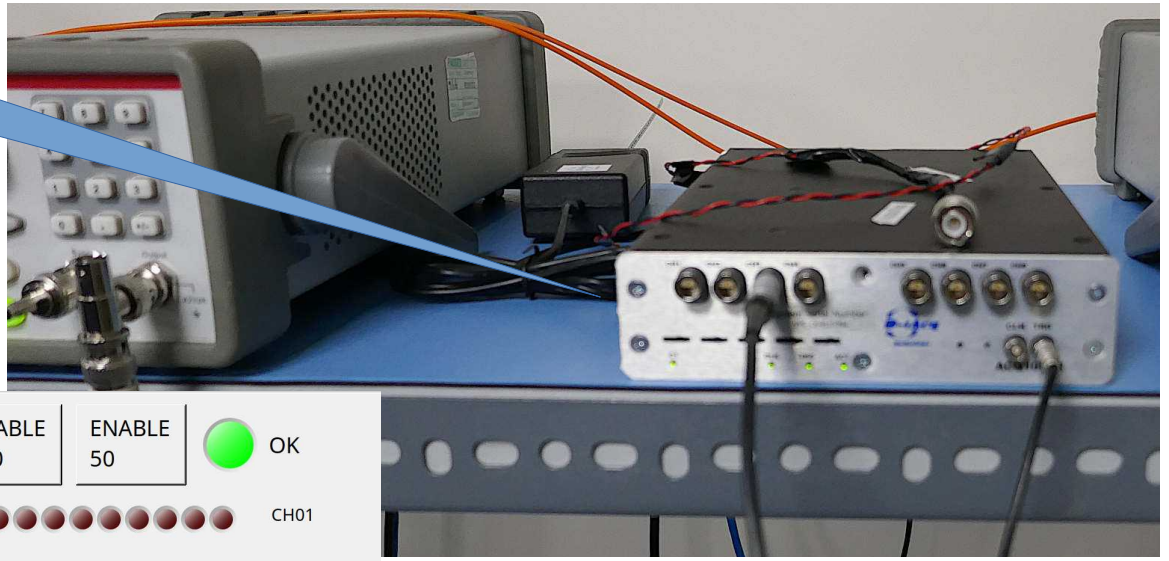
```
template<> const epicsInt16 acq400JudgementImpl<epicsInt16>::MAXLIM = 0x7fe0;
template<> const epicsInt32 acq400JudgementImpl<epicsInt32>::MAXLIM = 0x7ffffef0;
template<> const epicsInt32 acq400JudgementImpl<epicsInt32>::MINLIM = 0x80000010;
template<> const int acq400JudgementImpl<epicsInt16>::SCALE = 1;
template<> const int acq400JudgementImpl<epicsInt32>::SCALE = 256;
```

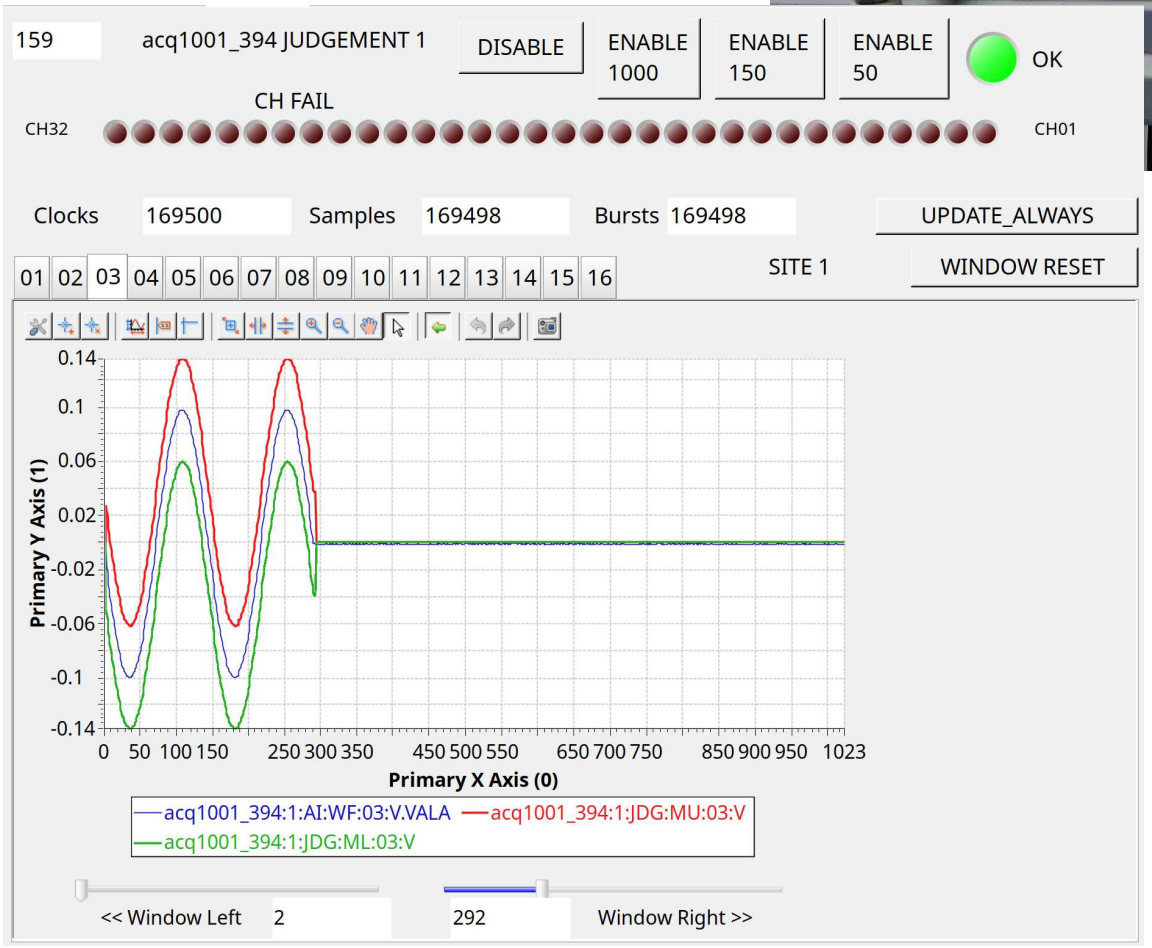
```
template<>
void acq400JudgementImpl<epicsInt16>::doDataUpdateCallbacks(int ic)
{
      doCallbacksInt16Array(&RAW[ic*nsam], nsam, P_RAW, ic);
      doCallbacksInt8Array(RESULT_FAIL,   nchan+1, P_RESULT_FAIL, 0);
}

template<>
void acq400JudgementImpl<epicsInt32>::doDataUpdateCallbacks(int ic)
{
      doCallbacksInt32Array(&RAW[ic*nsam], nsam, P_RAW, ic);
      doCallbacksInt8Array(RESULT_FAIL,   nchan+1, P_RESULT_FAIL, 0);
}
```

Specialization pre-configures,
to reduce in-loop decision to minimum!

Demo Hardware

ACQ1001+ACQ430-8
8AI x 128kSPS, 24 bit





So will it work?

- Not first time: 8 Channel Fail

```
FAIL_MASK32 = new epicsInt32[nchan/32];
```

- Scaling too small..
  - 150 codes at 16bit != 150 codes at 32 bit!
- Lesson #1 Must call callbacks

```
/* Do callbacks so higher layers see any changes */
    status = (asynStatus) callParamCallbacks();
```

- @@TODO : not all callbacks work?.

```
if (addr == ADDR_WIN_ALL){
      for (int ic = 0; ic < nchan; ++ic){
            setIntegerParam(ic, p_winx, winx[ic] = value);
            //callParamCallbacks(p_winx, ic); // callParamCallbacks(list=P_ARAM, addr=CH); @@todo BLOWS!
            //callParamCallbacks(ic, p_winx); // @@todo REMOVE me .. doesn't BLOW but does nothing
      }
}
```

Conclusion

- AsynPortDriver
  - Provides a convenient way to add custom data processing to the IOC
- We have powerful platform:
  - Wide range ADC, DAC, DIO
  - Plenty of FPGA resource.
  - Tightly coupled CPU with cycles to spare.
- *What can we do for you?*
  - *Try Judgement - available in firmware now! from*
  - *https://github.com/D-TACQ/ACQ400RELEASE/releases/tag/v395*
  - *Instructions: //usr/local/epics/README*

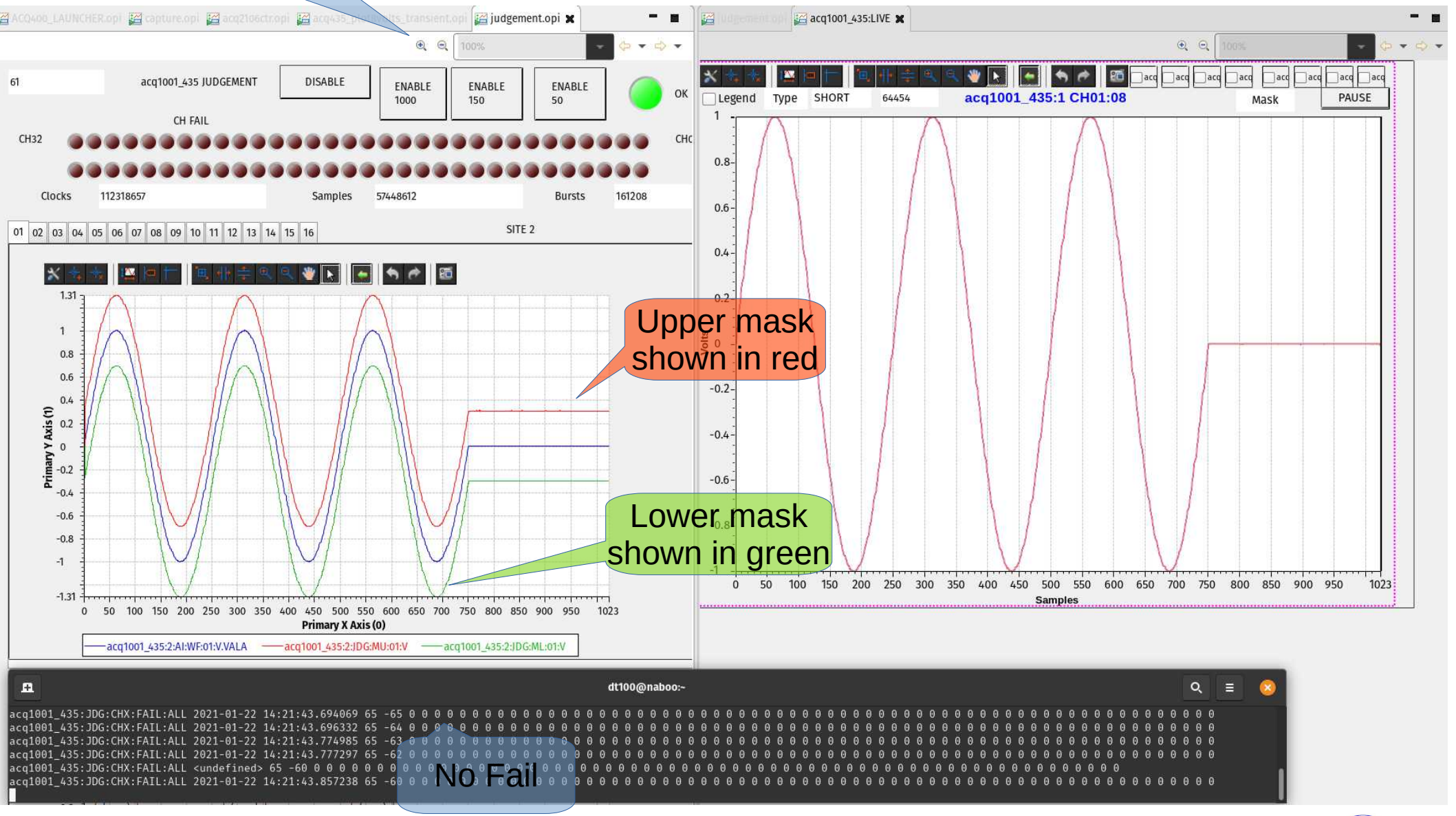
What we would like to do next..

- Use AsynPortDriver throughout.
  - Higher performance, easier maintenance.
- Adopt Phoebus
  - House distribution, easier adoption.
  - Reuse existing screens, but drop scripting.
  - Contractor?
- Use EPICS4 features
  - PVA, data streaming.

Judgement EPICS Inside

[Enable 1000]
Sets wide envelope

https://github.com/D-TACQ/ACQ400CSS



Judgement EPICS Inside 4