

# DAQ and electronics

Ben Smith

GRIDS 2025

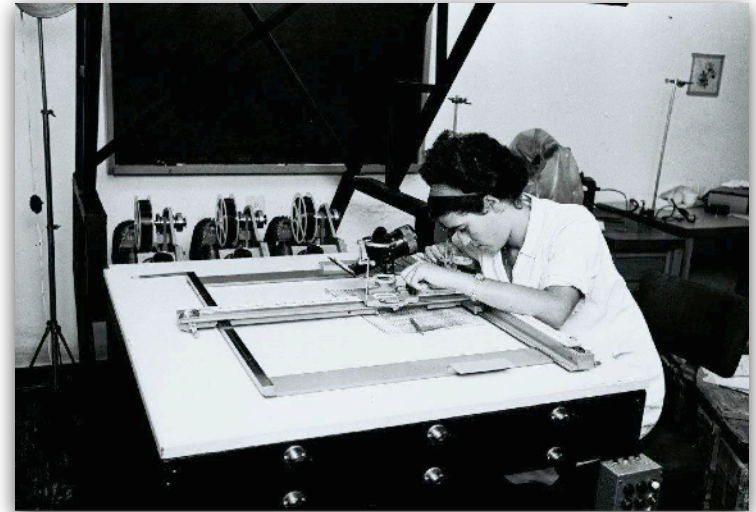
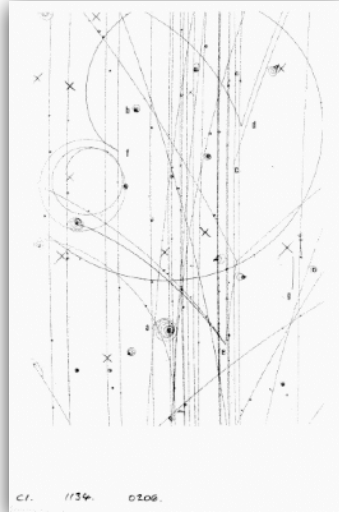
# Data AcQuisition

- Convert a physical signal into data that can be analysed
- This talk will try to illuminate this black box



# DAQ in the 1960s

- "Scanner" and "computer" were job titles
- Not very scalable



# DAQ now

- Almost universal approach
  - Convert physical phenomenon into **electrical signal**
  - Amplify/filter
  - Trigger
  - Simplify
  - Digitise
  - Write to disk
- The ordering can vary and some steps may be skipped

# Before we dive in to the details

- Keep in mind that DAQ needs to be considered when you're designing your experiment
  - DAQ output must be sufficient for physics goals
  - Can be a large part of your budget!
- Your physics analysis and simulation also need to know about the DAQ
  - Deadtime? Pileup? Saturation? Efficiencies?  
Resolution? Algorithmic decisions? Rounding?
- Even if it seems simple, there are probably subtleties...

Introduction

# Signal manipulation

Triggering

Wiring and crates

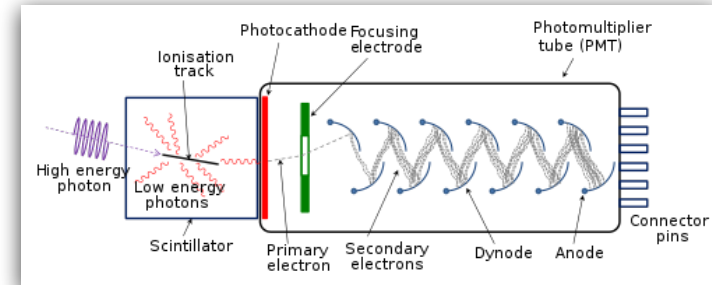
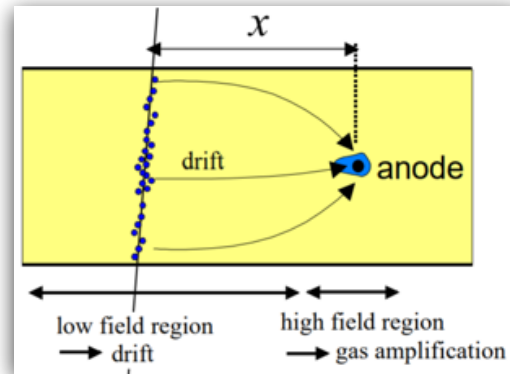
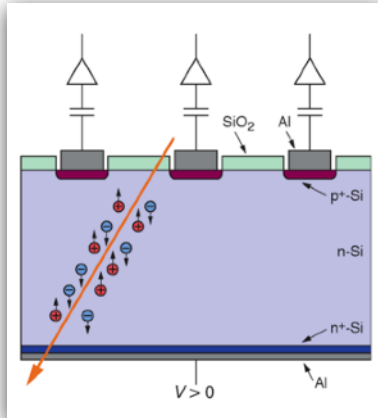
DAQ frameworks

Software and firmware

Example experiments

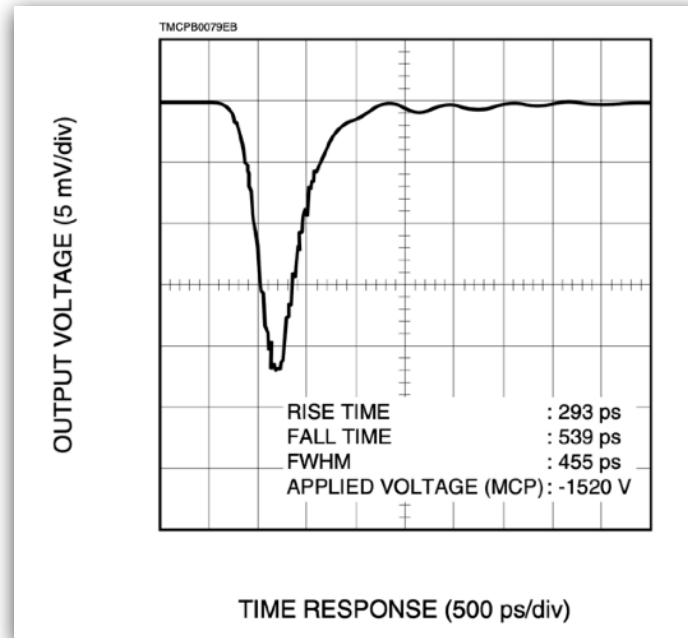
# Electrical signals

- Electrical signals are well suited for transport, manipulation, digitisation
- Amplification within detector of  $10^5$  to  $10^{10}$  typical
  - $Q_e = 1.6 \times 10^{-19} \text{ C}$ ;  $100\text{mV} \times 10\text{ns} = 10^{-9} \text{ C}$
- See other lectures for how detectors convert physical phenomena to electrical signals



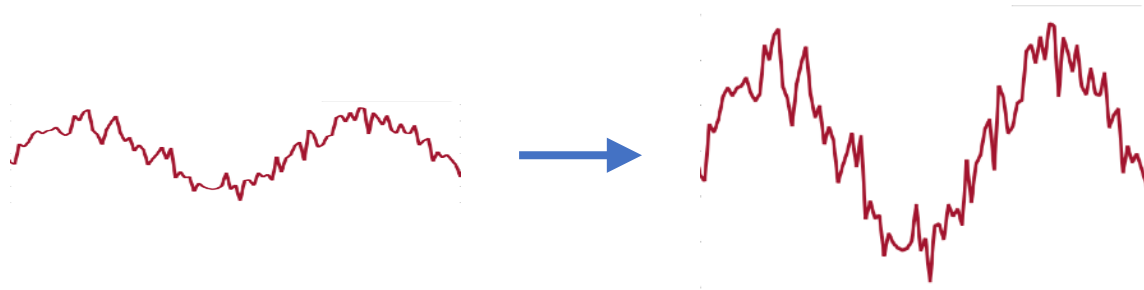
# Typical detector signals

- Amplitude and time-scale vary based on device
- PMT and SiPM for a single photon  $\sim 100\text{mV} \times 100\text{ns}$
- MCP for a single ion  $\sim 10\text{mV} \times 1\text{ns}$



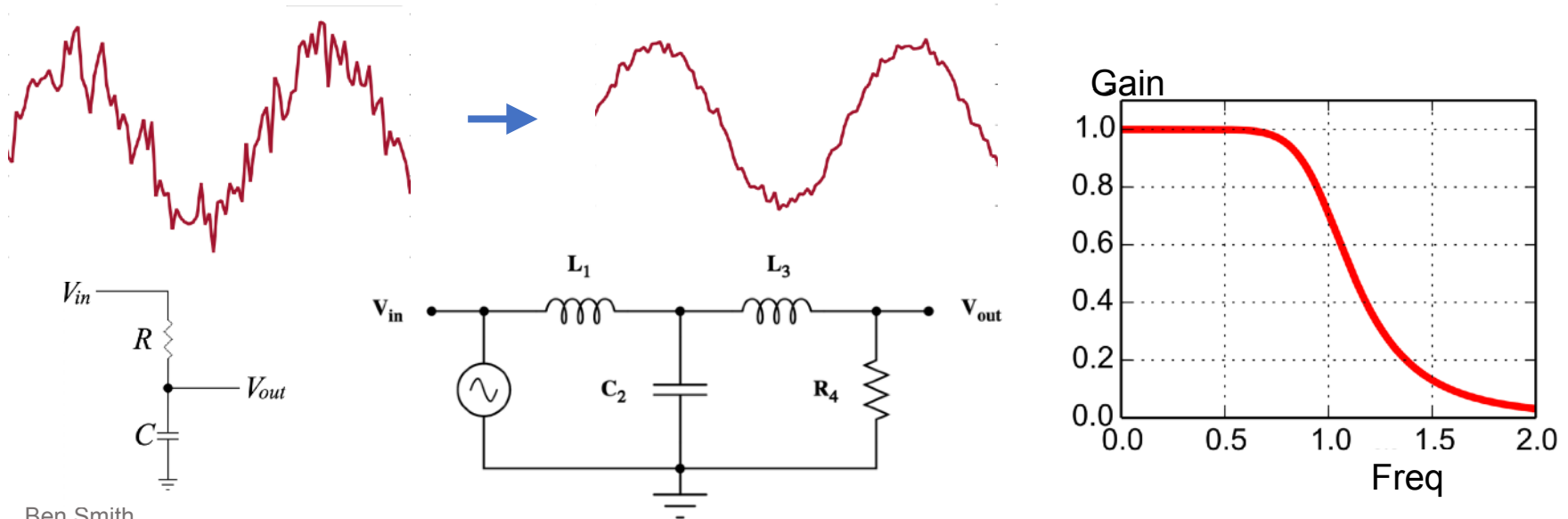
# Amplification

- If signal is very small, may want to amplify it further
- Reduce impact of any noise introduced after amplification (e.g. while transmitting through cables)
- Better match subsequent electronics (e.g. digitiser has a  $\pm 2\text{V}$  range)



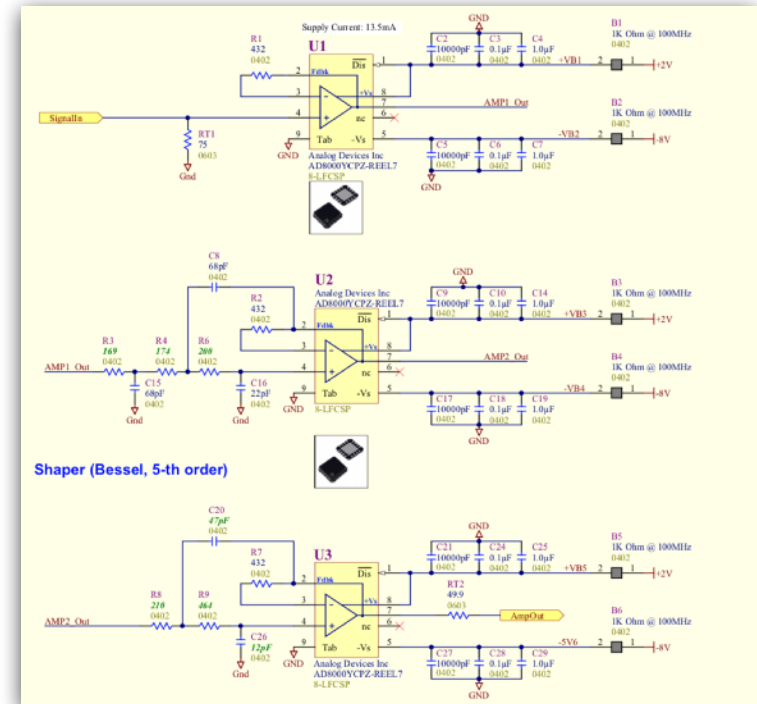
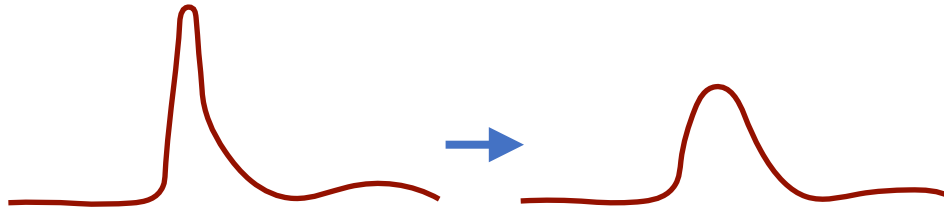
# Filtering / noise reduction

- Can use a filter to reduce high-frequency noise without affecting low-frequency signal
- Can be as simple as resistor+capacitor, but "higher order" arrangements give better response



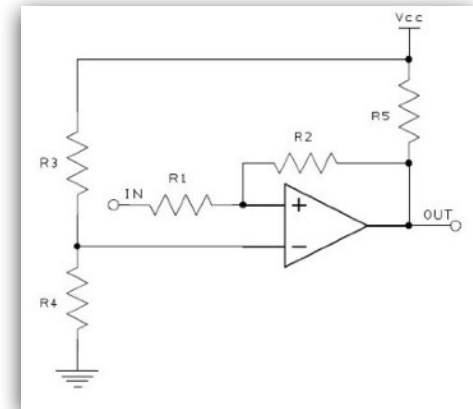
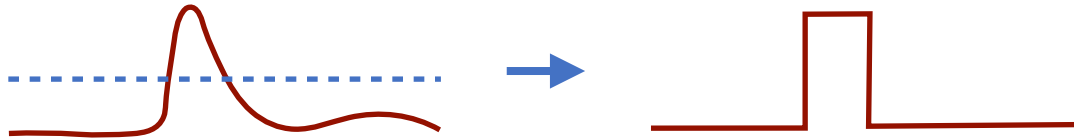
# Shaping

- Can stretch a fast signal to take longer
- E.g. to better match time resolution of a digitiser
  - Very fast digitisers are very expensive!
- Amplifiers+capacitors+resistors



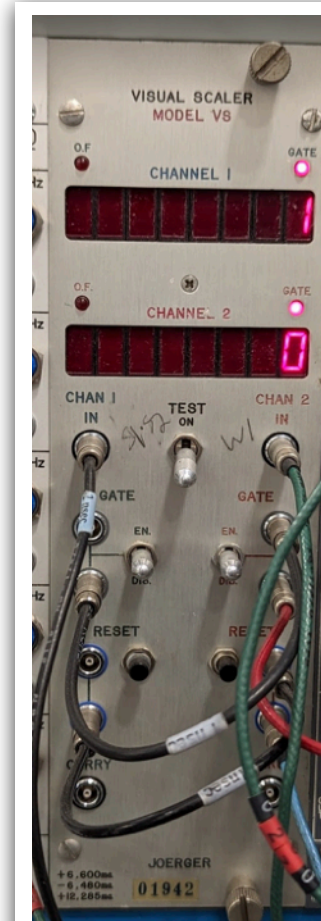
# Threshold crossing - discriminators

- We've looked at analog electronics so far
- Discriminators convert an analog signal to a digital (0/1)
  - Easier to process and check for coincidences etc.
  - Good for detection and timing
- Simple discriminators implemented by comparing to a set voltage



# Counting pulses - scalers/MCS

- Simplest thing you can do with digital signals - count!
- Visual / blind versions available
- Multi-channel scaler (MCS) has a terribly confusing name...
  - Should really be called a histogramming scaler
  - Counts pulses in each time bin (dwell time)



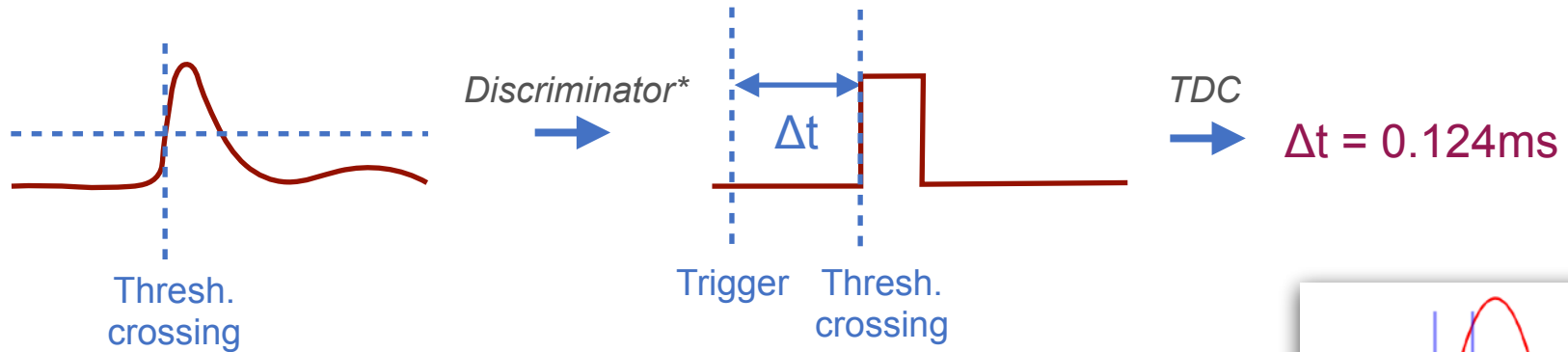
Visual scaler



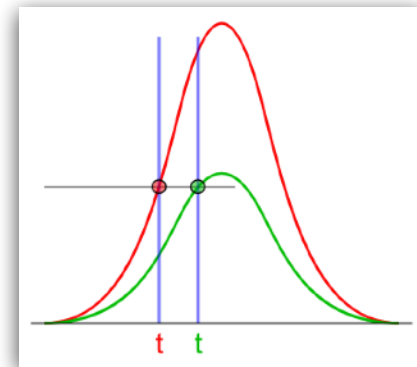
Blind MCS

# Time of pulse - TDC

- Time to digital converter
- Tells you the time of a pulse relative to a trigger signal
- 3ps resolution now available commercially

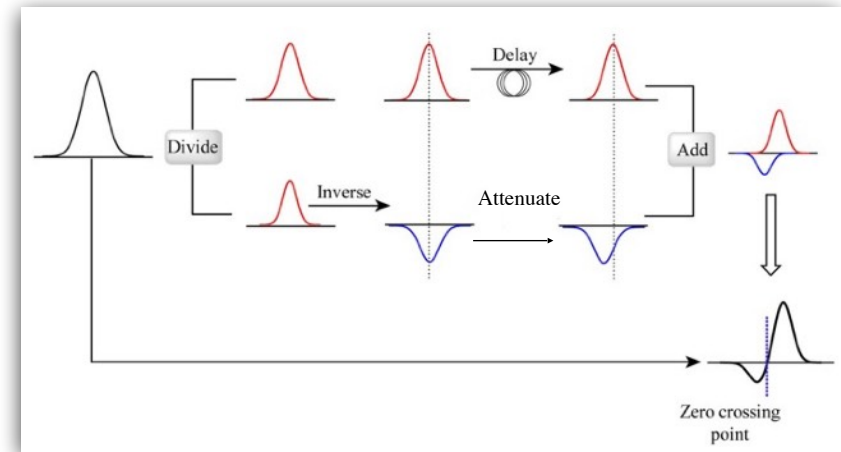
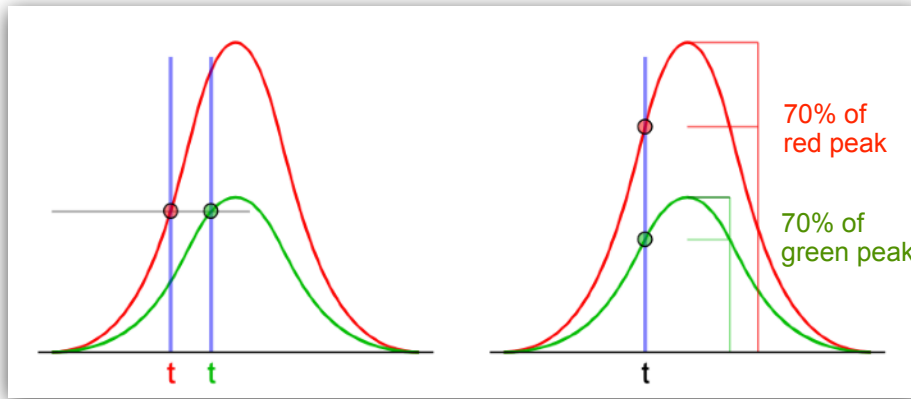


- Need to worry about time-walk effect if discriminator uses constant threshold voltage!



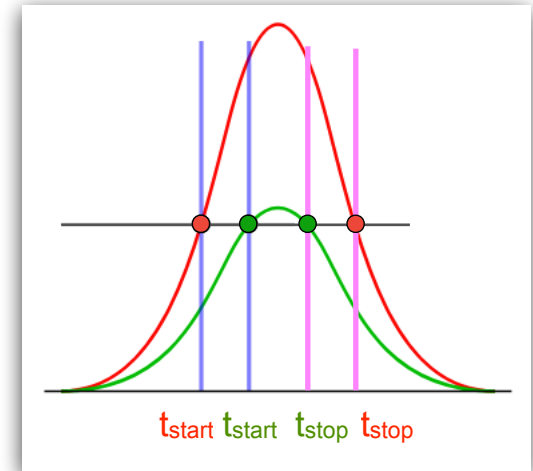
# Constant fraction discriminator

- Classic approach to dealing with time-walk correction
- Trigger at X% of pulse height, rather than fixed voltage
- Only works if signal has same rise time regardless of amplitude - e.g. scintillators



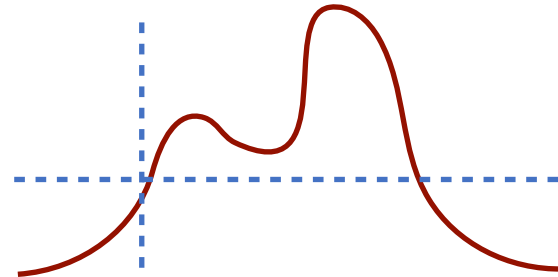
# Alternative approach to time-walk

- Record start time and time-over-threshold for each pulse
  - Correlated with pulse charge / amplitude
- Do time-walk correction in software
- More flexible than CFD
  - Works even if pulse shape changes with amplitude
- You must have excellent knowledge of how your signal varies with amplitude!



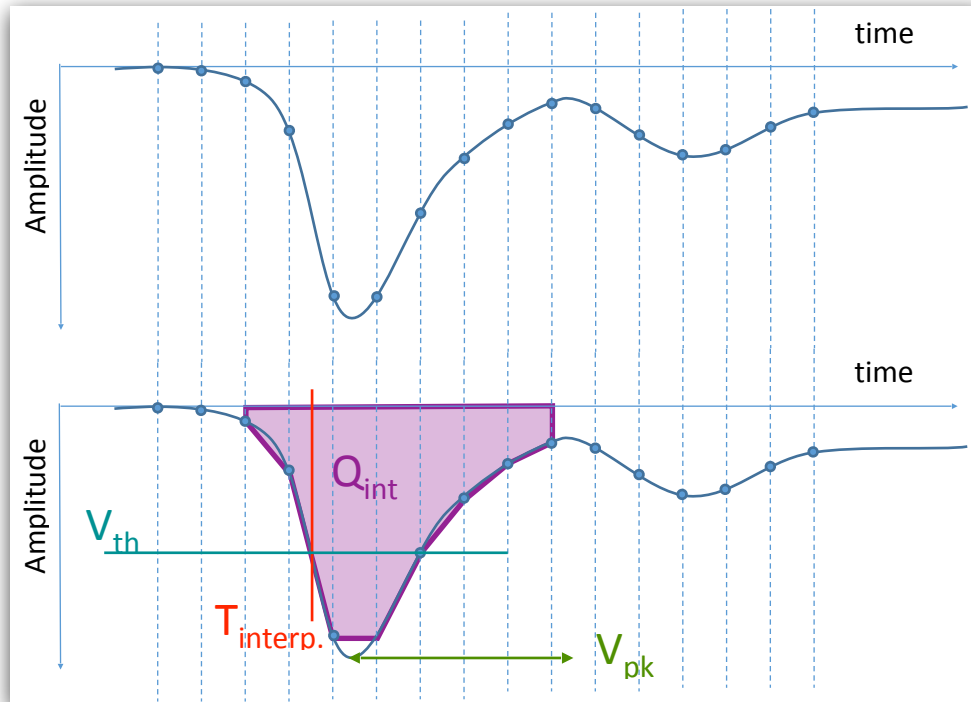
# Pileup and complex waveforms

- Some experiments need to count every photon
  - E.g. dark matter experiments where signal is time distribution of  $\sim 100$  photons
- Pileup of 2 photons is a problem for scaler/TDC - only get 1 count/timestamp!
- Need to either calculate the charge of the pulse (QT) or read out full waveform for offline analysis (digitiser)



# ADCs / digitisers

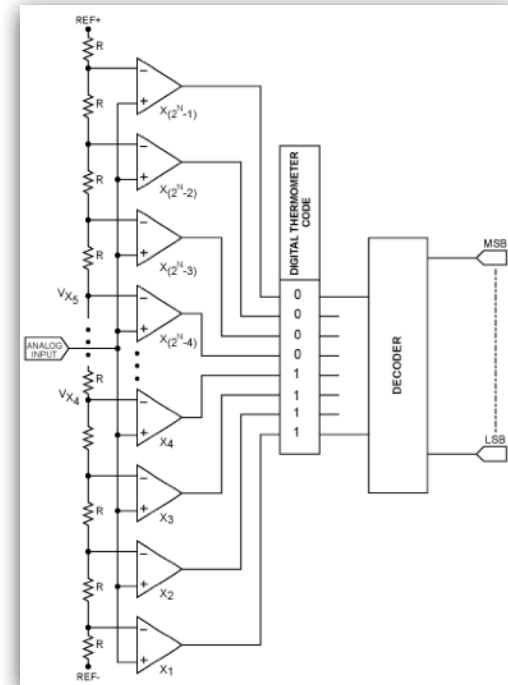
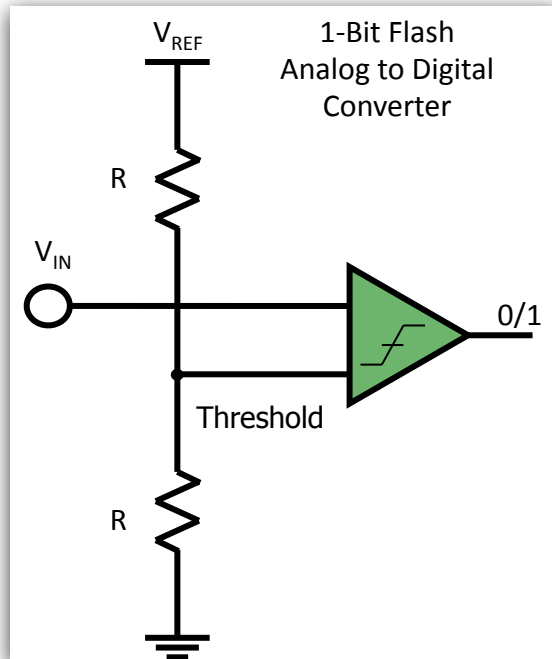
- Sample the waveform at specified sampling rate
- 10-bit resolution for 2V range: 2mV steps (4-bit: 125mV)



ADC Type	Resolution (bits)	Sampling rate
Dual Slope	12-20	100 sample/s
Successive approximation	8-18	10 Msample/s
Flash	4-12	10 Gsample/s
Pipeline	8-16	1 Gsample/s
Delta-sigma	8-32	1 Msample/s

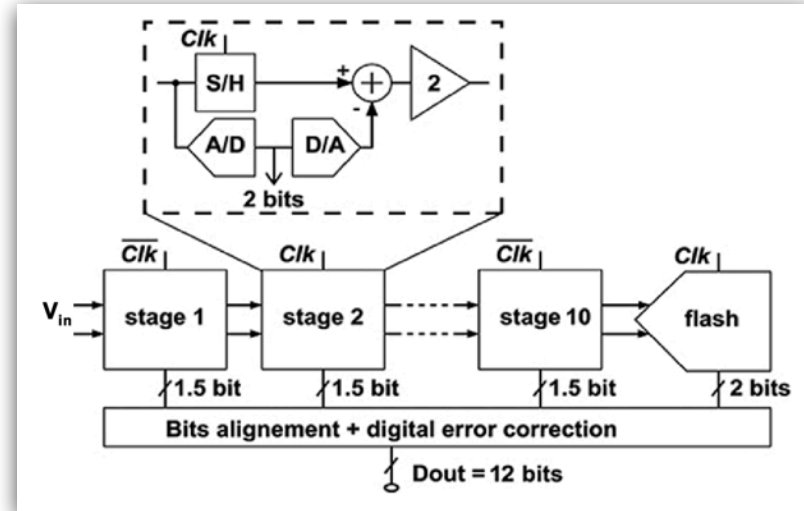
# Flash ADC implementation

- Conceptually simple, but cost grows exponentially with resolution
- Need  $2^{10}$  threshold comparators for 10-bit ADC!



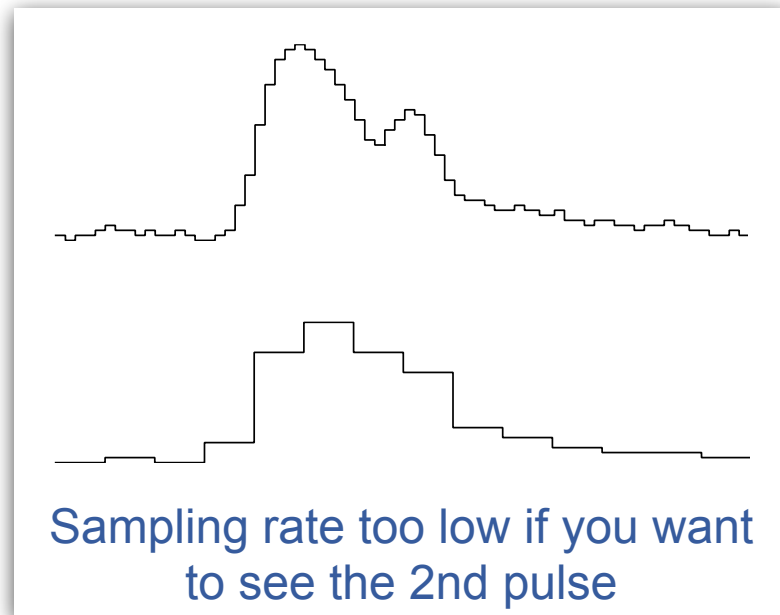
# Pipeline ADC

- Use multiple few-bit flash ADCs in sequence
- Uses successive approximation. E.g.
  - Stage 1 is 1V resolution (say 2-3V)
  - Subtract 2V from signal
  - Stage 2 is 0.1V resolution (say 0.3-0.4V)
  - Final result is sum of all stages (say 2.3V)
- Much cheaper (cost grows linearly with resolution)
- But, there's a delay as voltage passes through stages
  - Might impact your trigger?



# ADC considerations

- Sampling frequency
  - Need  $f_{\text{sampling}} > 2 \times f_{\text{signal}}$  (Nyquist-Shannon theorem)
- Sampling delay
  - Impacts trigger logic?
- Dynamic range
  - Need to amplify signal?
- Dead time during readout
  - Only for some ADC types



# Recap of possible outputs

- Discriminator + scaler
  - Number of pulses
- Discriminator + multi-channel scaler
  - Number of pulses in each time bin
- Discriminator + TDC
  - Time of each pulse (as a list)
- ADC/digitizer
  - Waveform of each pulse (at some time base and voltage resolution)



```
0x1000, 0x1001, 0x1004, 0x100a, 0x1014, 0x1010, 0x1007,
0x1002, 0x1000, 0x1002, 0x1003, 0x100b, 0x1015, 0x1011,
0x1008, 0x1003, 0x1001, 0x1000, 0x1001, 0x1000, 0x1001,
0x1000, 0x1001, 0x1004, 0x100a, 0x1014, 0x1010, 0x1007,
0x1002, 0x1000, 0x1002, 0x1003, 0x100b, 0x1015, 0x1011,
0x1008, 0x1003, 0x1001, 0x1000, 0x1001, 0x1000, 0x1001,
0x1000, 0x1001, 0x1004, 0x100a, 0x1014, 0x1010, 0x1007,
0x1002, 0x1000, 0x1002, 0x1003, 0x100b, 0x1015, 0x1011,
0x1008, 0x1003, 0x1001, 0x1000, 0x1001, 0x1000, 0x1001,
```

Introduction

Signal manipulation

# Triggering

Wiring and crates

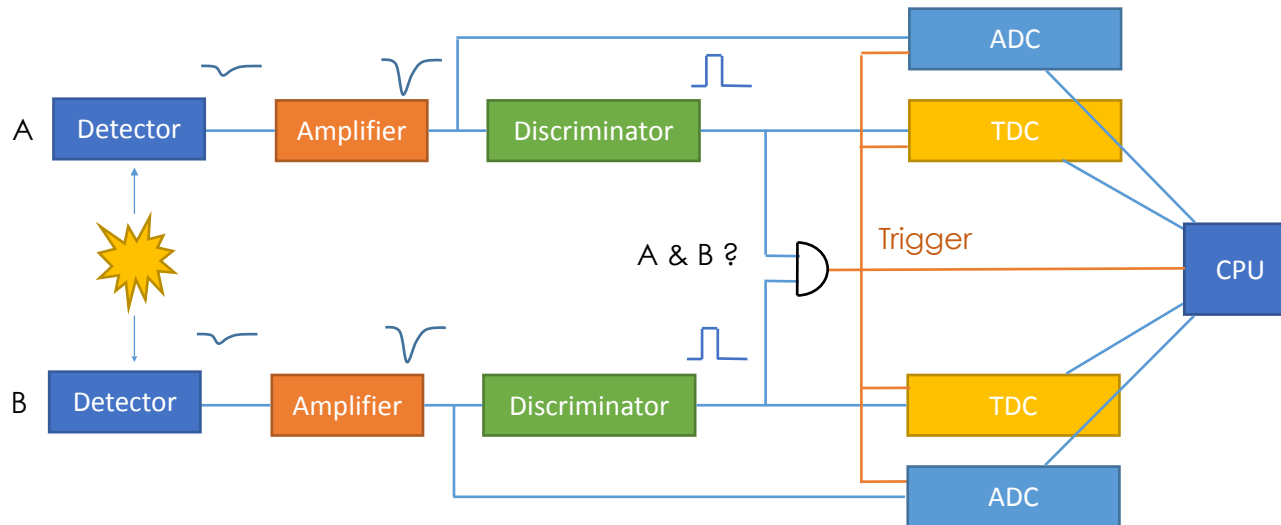
DAQ frameworks

Software and firmware

Example experiments

# Event-based (internal) triggering

- Detectors produce continuous electrical signals
- Reduce data rate enormously if you only trigger data readout when something "interesting" happens
- Event-based triggering uses detector signals themselves to decide when to trigger (e.g. a coincidence between channels)



# External trigger

- Trigger may be even easier if your experiment uses laser pulses / particle beam pulses
- Can work in both directions
  - DAQ tells laser/beam to pulse
  - Laser/beam tells DAQ that it has pulsed
- Either way, you can trigger your electronics using that same signal

# Trigger-less DAQ

- Used in some large-scale experiments (that don't have the same resources as LHC experiments)
  - Thousands/millions of detector channels
  - Too much activity to make a coincidence trigger
- Don't use a global trigger at all
- Each channel triggers independently
  - Read out every single hit (stream is many GB/s)
  - Reconstruct events in server farm
  - Decide what data to save (few MB/s)

# Why does data reduction matter?

- Data builds up quickly for experiments that run 24/7
  - 60MB/s is 2PB/yr
- Costs \$\$\$ to store data
- Simpler data is easier and faster to process/analyse
- BUT, you have to understand your detector
  - Experiments often start off recording lots of data, then implement more aggressive reduction once they've proved that the "lost" information doesn't affect the science analysis

# How much can you simplify?

- BetaNMR use multi-channel scalers, sum up the histograms in software, and only save those summed histograms
  - Written 1.5GB total since 2000
- Darkside-20k have thousands of channels and record the time/charge of each pulse
  - Write the same amount of data per second as BetaNMR write per year!

Introduction

Signal manipulation

Triggering

## Wiring and crates

DAQ frameworks

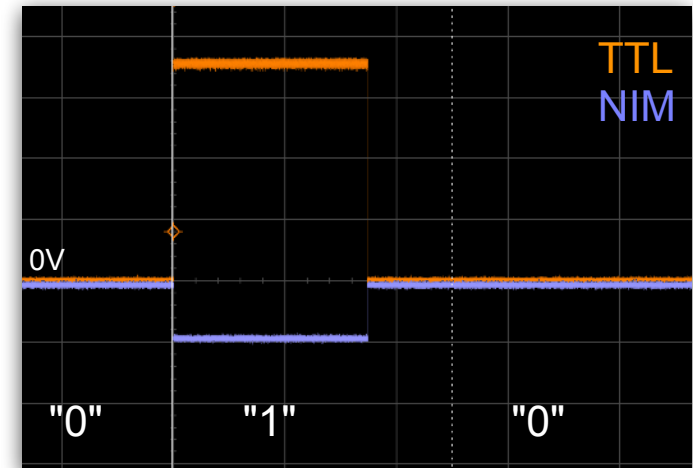
Software and firmware

Example experiments

# Digital levels

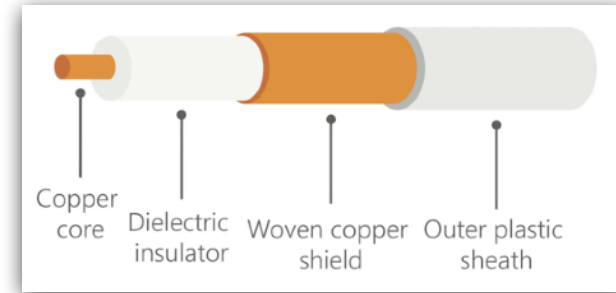
- Different standards evolved over time
  - Available technology, speed, power consumption...
  - NIM/TTL are the most common in physics
- Level translators available if you need to mix

Standard	Digital 0	Digital 1
NIM	0 V	-0.8 to -1 V
TTL	0 to +0.7 V	+1.5 to +5 V
ECL	-1.75 V	-0.9 V



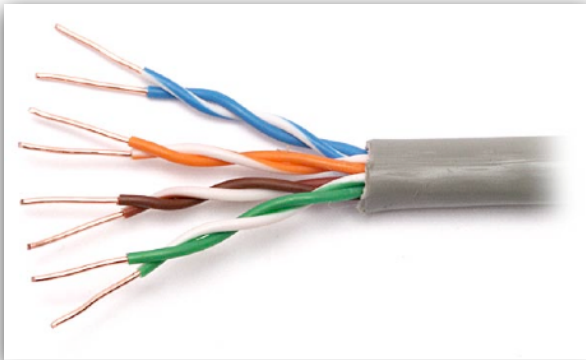
# Cables and connectors

- Most signals transmitted using coaxial cables
- Different connector options
  - SHV is for safe high voltage connections
  - LEMO 00 is very often used in physics
    - Latches into the receiver
    - Pull the outer housing, not the cable!
  - BNC also common
    - Larger but cheaper

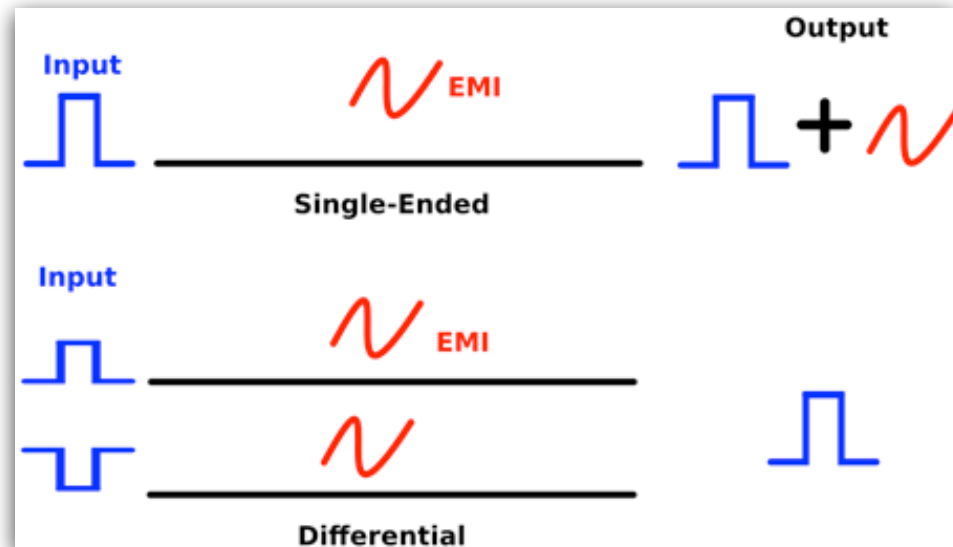


# LVDS

- Low voltage *differential* signalling
- Instead of sending +2V, send (A) +1V and (B) -1V
  - Receiver does (A-B) to get +2V
  - But any "pickup noise" is cancelled out!
- Usually implemented in a "twisted pair" cable



Ben Smith

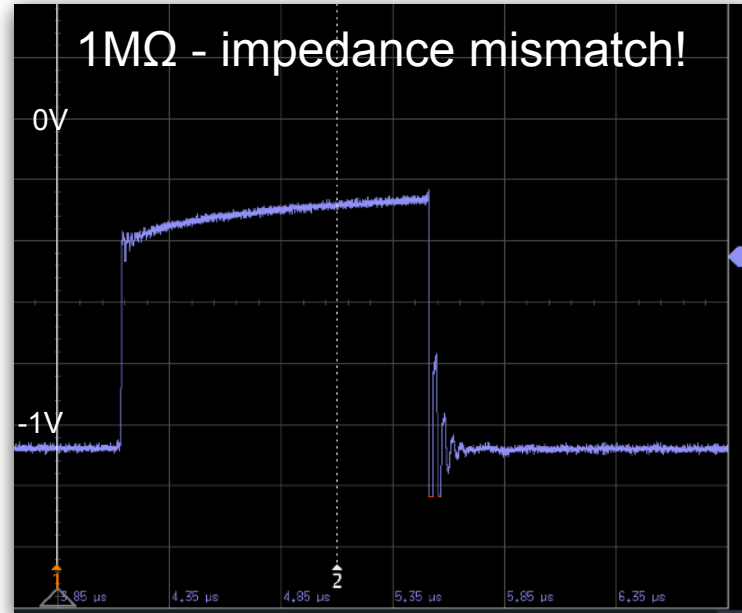
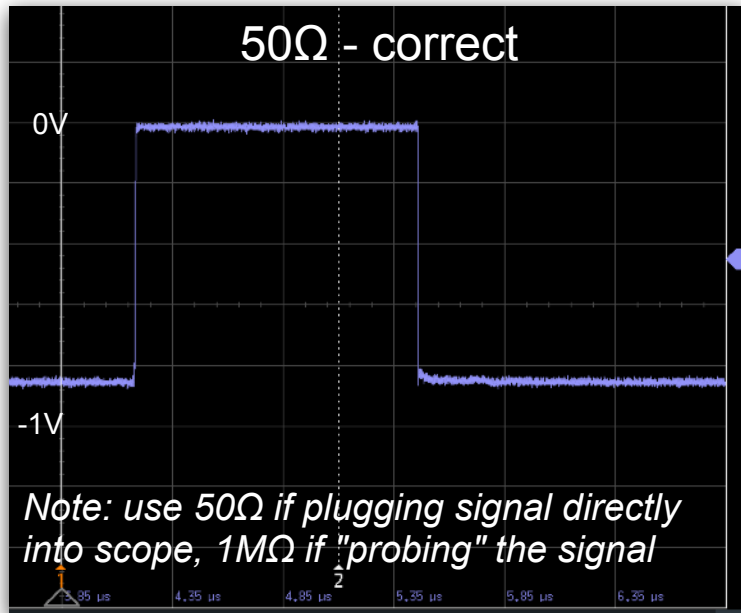


# Impedance matching and reflections

- Impedance ( $Z$ ) is the complex version of resistance
  - $Z = R + jX$ ;  $R$  is resistance (DC);  $X$  is reactance (AC)
  - Ideally resistors are pure- $R$ , capacitors are pure- $X$
  - Reactance varies with frequency of signal
- If there's a mismatch between source/cable/receiver, can get reflections in the signal
- Almost all devices/connectors you'll use are  $50\Omega$ 
  - Beware - RF systems often use  $75\Omega$

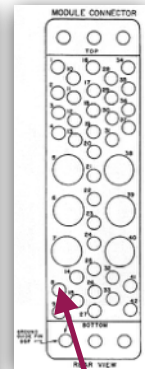
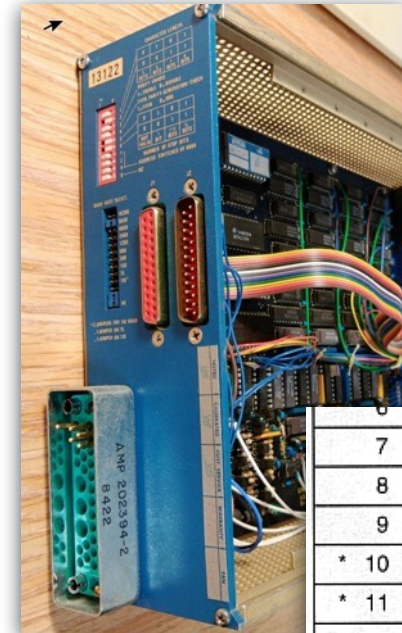
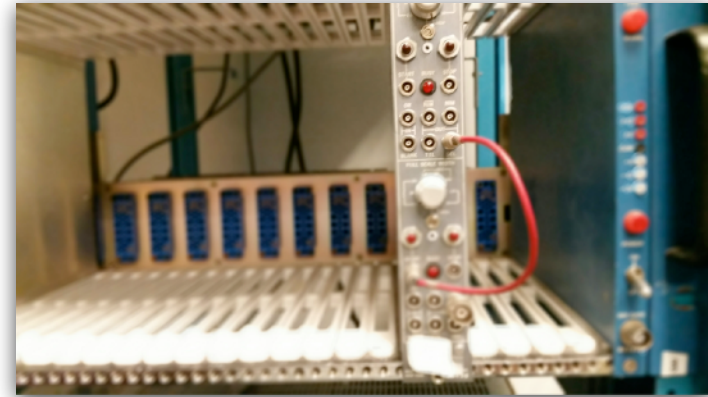
# But I don't design electronics or deal with RF...

- The impedance mismatches you're likely to see are:
  - Oscilloscope being in  $1\text{M}\Omega$  termination instead of  $50\Omega$
  - Forgetting to terminate daisy-chains



# NIM crates

- Very simple (and old) standard
- Provides power to NIM modules through a standardised connector
  - One of the power pins is 200V!!
- Many modules available
  - Amplifier, discriminator, fan-out, coincidence, scaler, ...
- Not used much outside physics



6	
7	
8	+200 V D.C.
9	SPARE
* 10	+6 V
* 11	-6 V
12	RESERVED

"NIM" has two meanings - crate and voltage levels!

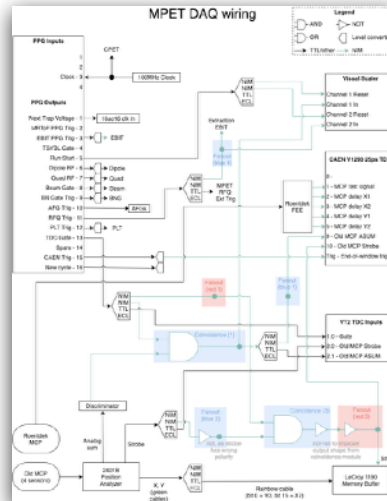
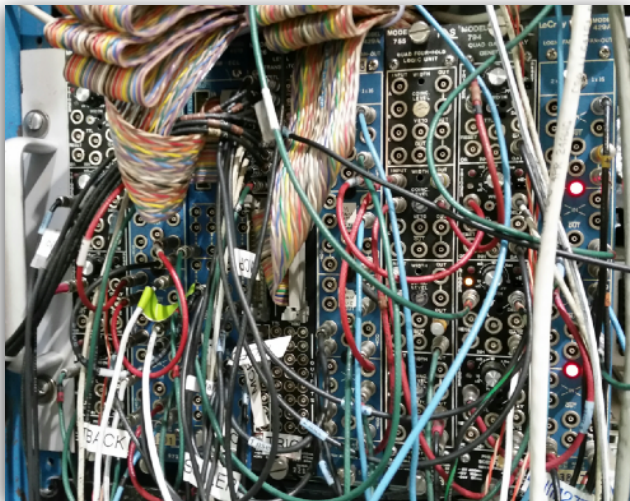
# Backplane-based crates

- Backplane allows boards to talk to each other
  - Often put a single-board computer in the crate to read out data from all boards
- VME (1981)
  - Still used in physics, military, railroads...
  - Max  $\sim 1\text{Gb/s}$  through backplane (generally less)
  - Some boards now only use backplane for power - read data out via ethernet
- uTCA (2003)
  - Developed by telecommunications industry
  - Higher data rate through backplane ( $100\text{Gb/s}$ )



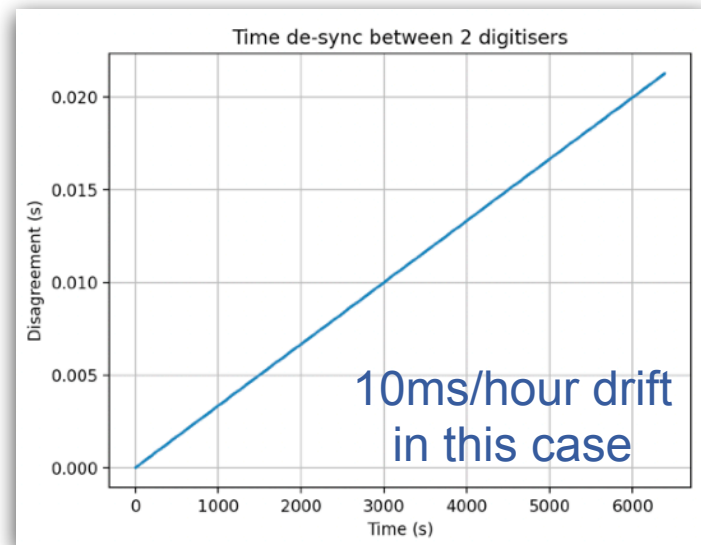
# Wiring top tips

- Can be time-consuming to trace the logic
  - Make a diagram and label your cables
  - [drawio.com](https://drawio.com) is a great tool
- Labs have lots of modules lying around that are OLD!
  - Assume every connection is bad until checked...



# A quick word about clocks

- Many devices have on-board clocks
- These are generally quite inaccurate and *will* drift
  - Bad if you need to correlate data from multiple boards
- If you have multiple devices, use one as the master and distribute that clock signal to others
- If you need true and stable, use a Rb clock or similar
- If you need the true "real world" time (e.g. for supernova search) use GPS



Introduction

Signal manipulation

Triggering

Wiring and crates

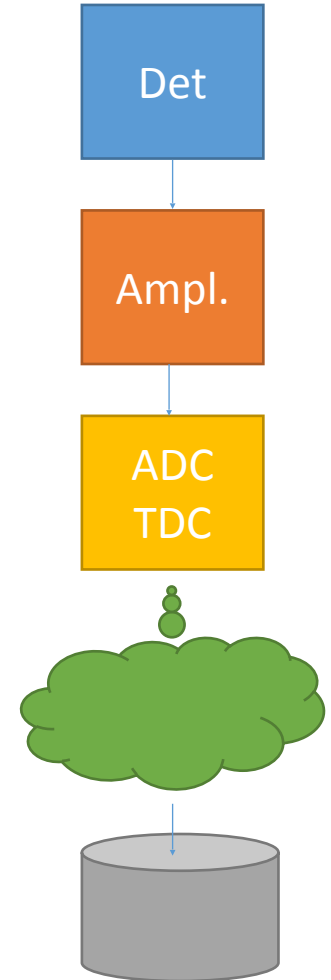
## DAQ frameworks

Software and firmware

Example experiments

# Final piece of the puzzle

- Need to read data from digitisers, TDCs etc and store on disk for final physics analysis
- Often want to combine data from multiple detectors into a single "event"
- Generally acquire data in "runs" with specific experimental conditions
  - E.g. PMT voltages, calibration source location, beam energy, ...
- Several frameworks available to help

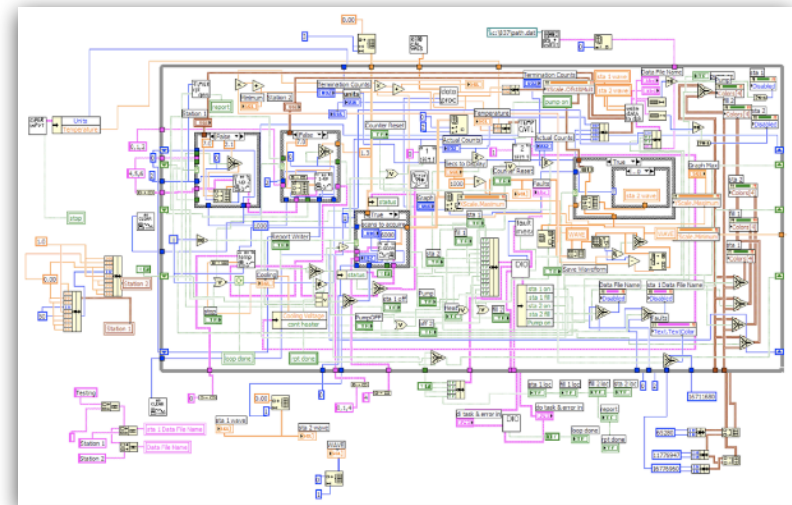
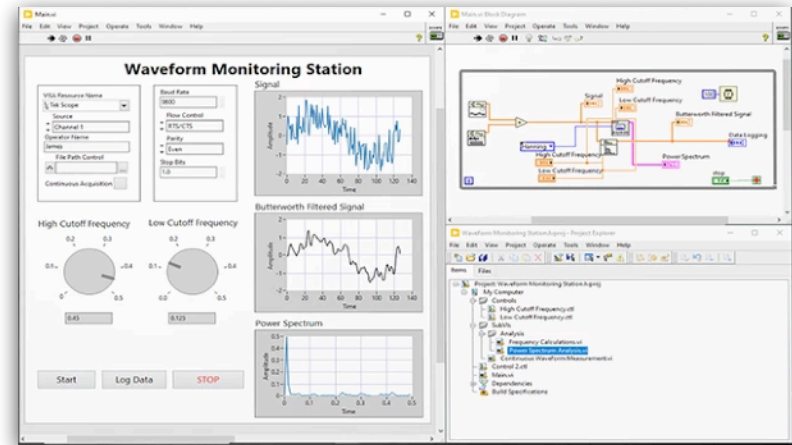


# What we want from a DAQ framework

- Read data from many types of hardware
- Configure devices and store the configuration
  - Including "slow control" of voltages,
- Collect data from many devices into a single "event"
  - Multiple programs, possibly on multiple computers
- Stable - will it still work in 10 years?
- Efficient and performant - GB/s input?
- Nice to have - alarms, event displays, web interface, ...

# LabView

- "Low code" proprietary system by National Instruments
- All NI devices will come with a LabView driver
- Simple to setup tabletop experiments
- Does not scale well for experiments that need high speed, complex logic, many devices, or use Linux



# Other DAQ frameworks

- Many labs have developed their own systems

Framework	Developed by	Written in	Used by
<b>ORCA</b>	North Carolina	Objective-C (MacOS only)	Katrin, Majorana, SNO+, ...
<b>artdaq</b>	Fermilab	C++	Lariat, mu2e, SBND, ...
<b>CODA</b>	Jefferson	Java (plus need special VME hardware)	JLab
<b>MIDAS NewGen</b>	Rutherford	Tcl	Rutherford
<b>MIDAS</b>	PSI/TRIUMF	C++ and python	mu3e, T2K, Alpha-G, MEG, Darkside, ...

# MIDAS

- Maximum Integrated Data Acquisition System
- Very customisable
  - Can implement data readout in C++ or python
  - Webpages in javascript
- Free (GPL)
- Good for 1-50 computers and ~ 1GB/s data rate

ds Alarms: None 13 Jun 2022, 22:19:57 UTC+2

Status

Transition

ODB

Messages

Chat

Alarms

Programs

Buffers

History

Sequencer

Config

Event Dump

Save and load

CAEN SY1527

CAEN VX2740

PDU Plus

PDU Slim

ELOG

dsanana

Runviewer

b01 firmware

### Run Status

Run 1327 Start: Mon Jun 13 21:39:47 2022 Stop: Mon Jun 13 21:41:02 2022

Stopped Start Alarms: On Restart: Off Data dir: /storage/dsdata/setup-2/sandbox

Experiment Name: **DarkSide Setup-2 (ds)**

Run type: self-trigger

Shifter: ZB AC ES

Light level: Off

Comment: fully populated pdu, 138.5 V, all tiles on for all quadrants, 5.6us window, DARKSIDE FIRMWARE, 48-sample moving average

Write Data: false

1655149264 21:41:04.045 2022/06/13 [rvprovider,LOG] Program rvprovider on host darkside-daq stopped

### Equipment

Equipment +	Status	Events	Events[/s]	Data[MB/s]
VX2740_Config_Group_000	VX2740_Group_00@darkside-daq.na.infn.it	13	0.0	0.000
VX2740_Data_Group_000	VX2740_Group_00@darkside-daq.na.infn.it	158933	0.0	0.000
EpicsFrontend	EpicsFrontend@darkside-daq.na.infn.it	0	0.0	0.000
SteeringModulePDUSlim	Frontend stopped	0	0.0	0.000
SteeringModule	Frontend stopped	0	0.0	0.000
CAEN_HV	Frontend stopped	0	0.0	0.000
SteeringModulePDUPlus	SteeringModulePDUPlus@darkside-daq.na.infn.it	0	0.0	0.000

### Logging Channels

Channel	Events	MB written	Compr.	Disk Level
#0: run01326sub0000.mid	0	0.000	0.0%	27.6%

Lazy Label	Progress	File Name	# Files	Total
runcopy	0.0%	run01326sub0000.mid	4155	0.0%

### Clients

Client	Sequencer	EpicsFrontend
saveload [darkside-daq.na.infn.it]	Sequencer [darkside-daq.na.infn.it]	EpicsFrontend [darkside-daq.na.infn.it]
SteeringModulePDUPlus [darkside-daq.na.infn.it]	Logger [darkside-daq.na.infn.it]	VX2740_Group_00 [darkside-daq.na.infn.it]
mhttpd [darkside-daq.na.infn.it]	Lazy_runcopy [darkside-daq.na.infn.it]	mserver [darkside-daq.na.infn.it]



Introduction

Signal manipulation

Triggering

Wiring and crates

DAQ frameworks

**Software and firmware**

Example experiments

# Software

- Any non-trivial experiment requires custom DAQ software
  - Data readout, event building, realtime analysis, ...
- Language depends on your DAQ framework
- May need to write low-level code to talk to devices
  - C functions, register maps, sockets, memory buffers
  - Some vendors provide higher-level libraries

# Software - Python

- Easy to learn with rich ecosystem of libraries in PyPI
  - E.g. python-vxi11 for talking to many DAQ devices
- Slower to execute compared to C/C++
  - Can (carefully) call C code/libraries from Python
- Added python support to midas a few years ago
  - Very popular, especially for low-data-rate applications
  - Correct usage of numpy can make code much faster

# Software - C/C++

- Compile your software, then run it
  - Compiler is your friend, not your enemy
  - Try to fix all warnings/errors
- You need to know a little more about memory, pointers
  - "New" smart pointers make memory usage safer
  - `std::shared_ptr` etc.
- Modern C++ has many features
  - Keep it simple



# Software - development tips

- Write code that is readable!
  - Make your life as easy as possible when you need to debug/extend
- Try to separate layers of complexity
  - Don't obscure the high-level algorithm with low-level implementation details
- Give your functions sensible names
  - The principle of least surprise
- Document the *why* not the *what*



# Software - outsourcing to AI

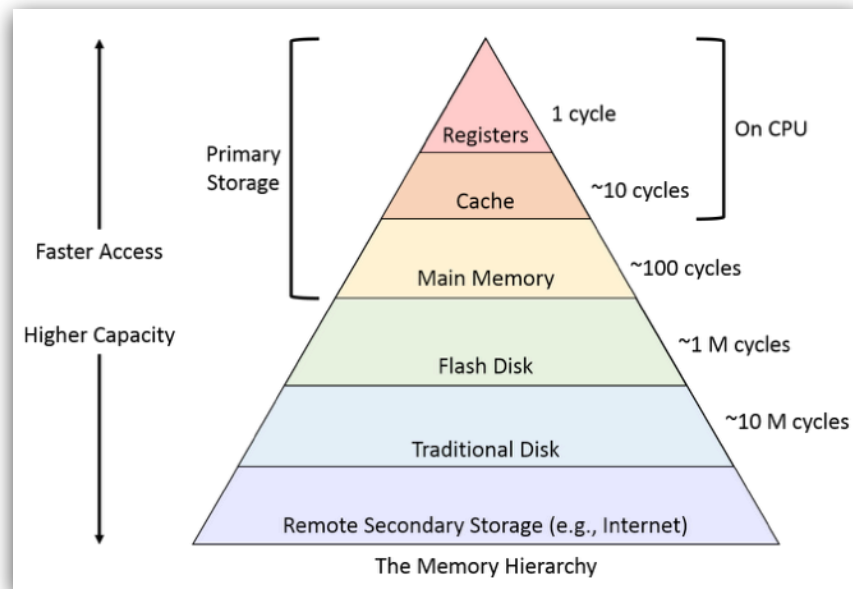
- ChatGPT / CoPilot / Claude / Cursor can generate code
- *May* generate working code quickly
- I think it's fine for smaller tasks
  - E.g. how to add a label to a plot in matplotlib / ROOT
- Long-term it's a risk for bigger tasks
  - Sometimes it's wrong, or does the right thing in a weird way
  - Harder to maintain/explain/standardise AI-gen code
  - Normally at least 1 person understands the codebase (the original author) - now it may be 0?
- Ask me to rant about "vibe coding" when we're at the pub...

# Software - testing

- Test any code you write!
  - Just because your code compiles doesn't mean it's correct...
  - Ensure you don't introduce "regressions" as code evolves
- Strongly consider testing frameworks
  - unittest module for Python
  - Catch2/GoogleTest for C++
  - May want to write a dummy/mock device so you can run tests without talking to real hardware devices
- I structure my code so I can test ~99% of it on my laptop

# Software - getting faster

- You can (almost) always make your code faster
  - Use a more efficient algorithm
  - Use a compiled language
  - Buy more/faster computers
  - Optimise your in-memory data layout/usage
  - Use a GPU (for very parallel problems)
  - Write firmware

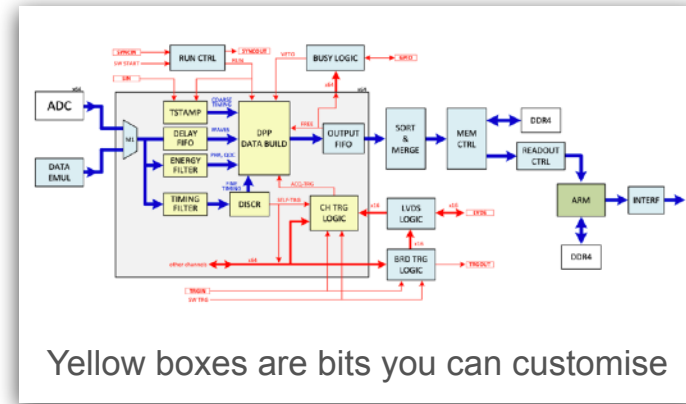


# Software vs firmware

- Software runs on a general purpose PC
  - You don't have tight control over what happens when
  - You can try to declare yourself as a "high priority" program, but the OS still decides how much CPU time each program can have
- Firmware runs on dedicated hardware
  - Your code defines what should happen every single clock cycle (multiple instructions possible)
  - You know *precisely* what will happen 100 million times per second (for 100MHz clock)

# Firmware development

- Hardware design languages - VHDL/SystemVerilog
- Writing firmware code takes a LOT longer than software
- Generally an engineer-level position
- BUT companies are starting to offer "open firmware" solutions, e.g. CAEN OpenFPGA
  - CAEN write most of the firmware, but allow you to implement some custom logic
  - They provide a development kit to make your work easier



Introduction

Signal manipulation

Triggering

Wiring and crates

DAQ frameworks

Software and firmware

**Example experiments**

# Reminder of our building blocks

## Signal manipulation

Amplifier  
Noise filter  
Stretching  
Summation

## Digitisation

Discriminator  
Scaler/MCS  
TDC  
Digitiser/ADC

## Digital logic

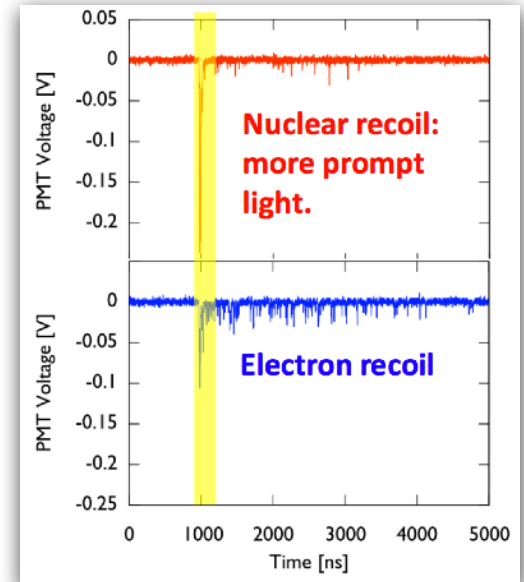
Coincidence (AND)  
Fan-in (OR)  
Inverter (NOT)

## Custom code

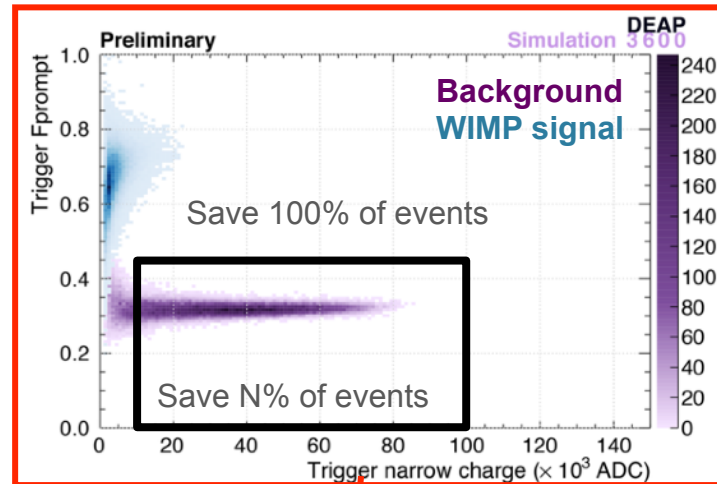
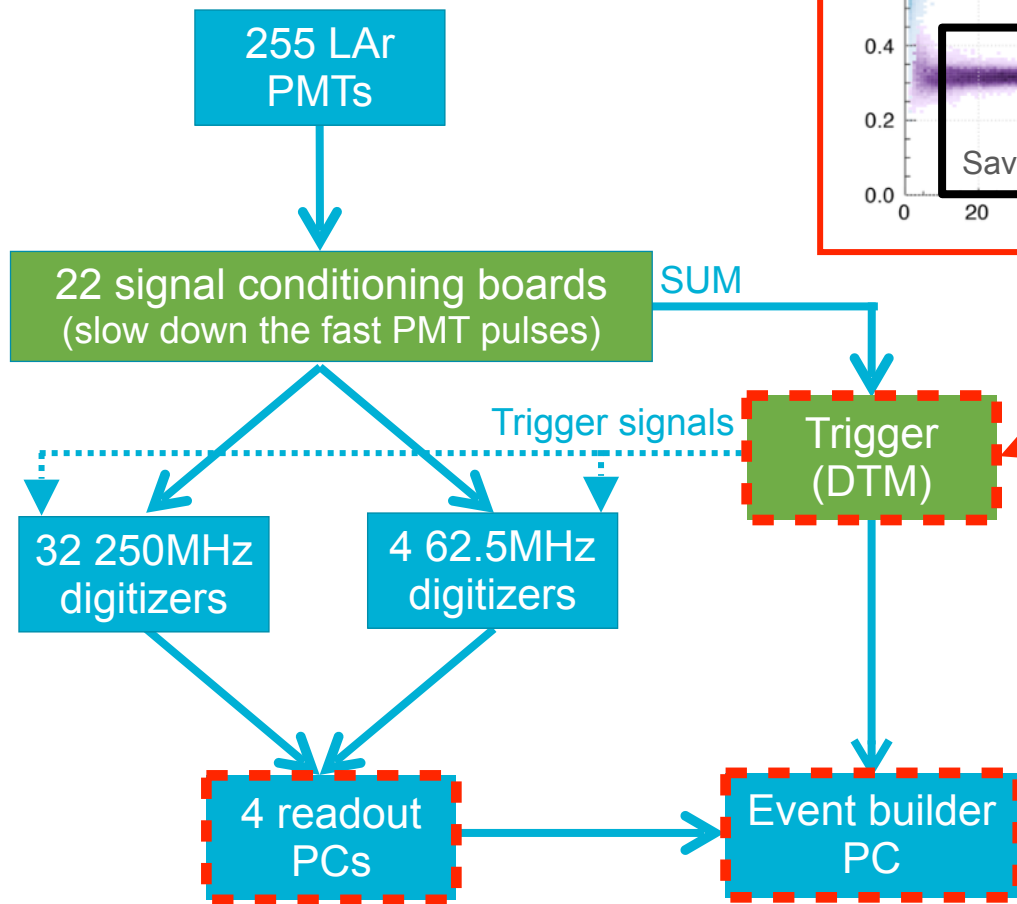
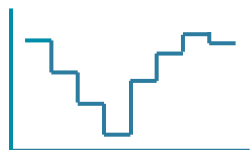
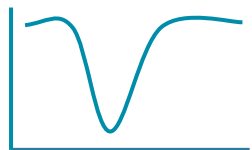
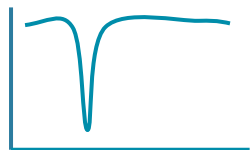
Firmware  
Software

# Example 1 - DEAP-3600

- Dark matter search - 255 PMTs in LAr
- High background rate (kHz)
- Distinguish signal/background using energy and time distribution of photons
  - Need to record detailed info for signal-like events
- Runs 24/7 for many years
  - Need to suppress background
  - Trigger needs custom firmware



# Example 1 - DEAP-3600



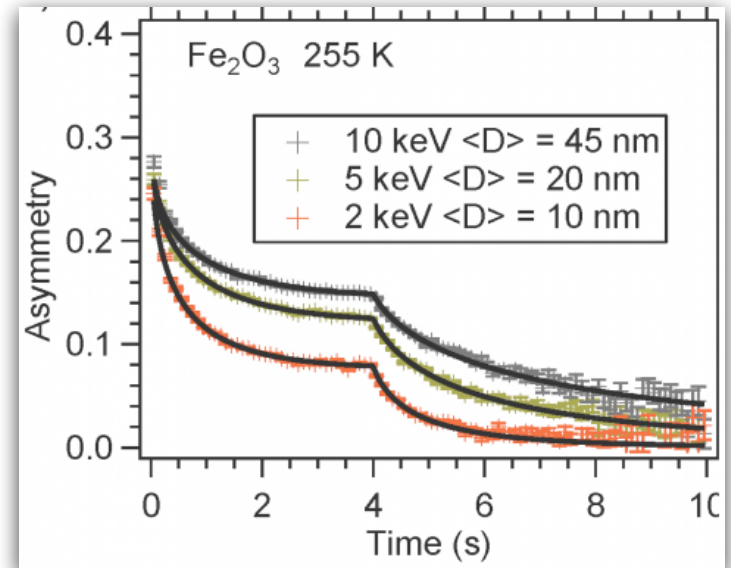
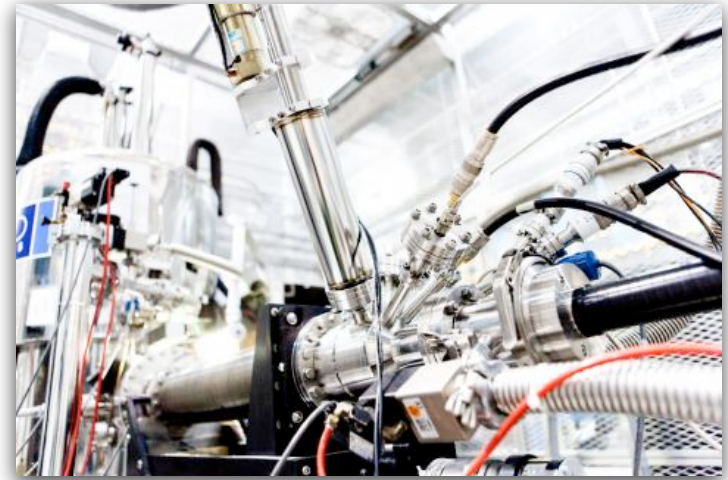
Custom hardware

Commercial hardware

Custom software/  
firmware

## Example 2 - BNMR

- Beamline experiment for condensed matter and material science
- Inject ions, count beta decays as a function of time
- Complex timing schemes for ion beam, RF, lasers
  - Different for each experiment and sample



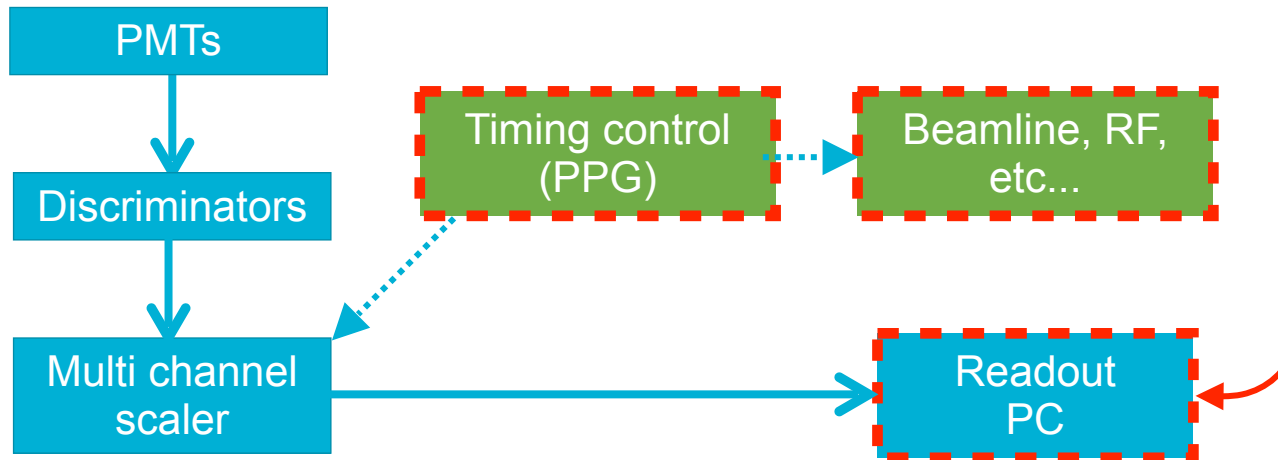
# Example 2 - BNMR

- Hardware is relatively simple
- Complexity is in the software
- Good user interfaces can greatly improve efficiency

Custom hardware

Commercial hardware

Custom software/  
firmware



# Conclusion

- DAQ links the hardware and data analysis
  - Tools for operating and validating the experiment
  - Knowledge of DAQ needed for proper data analysis
- Many decisions to make based on physics needs/budget
  - Important to consider this early in design process
- Lots of DAQ frameworks, but I recommend midas
- DAQ software spans low-level device communication through high-level user interfaces
- Test your code - your analysis depends on it!