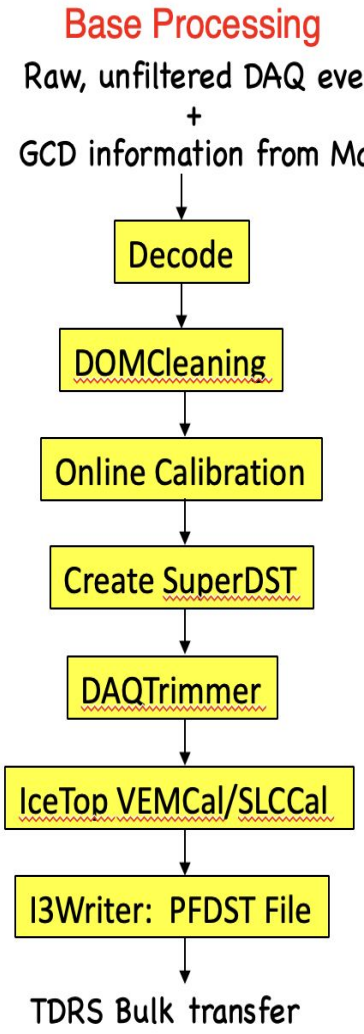
Upgrade Software & Data Decoding Tools

Delaney Butterfield & Dawn Williams
IceCube Summer School
June 3, 2026

Thanks to Erik Blaufuss, Jim Braun, and Luke Bloom for resources contributing to these slides!

How do we read data?

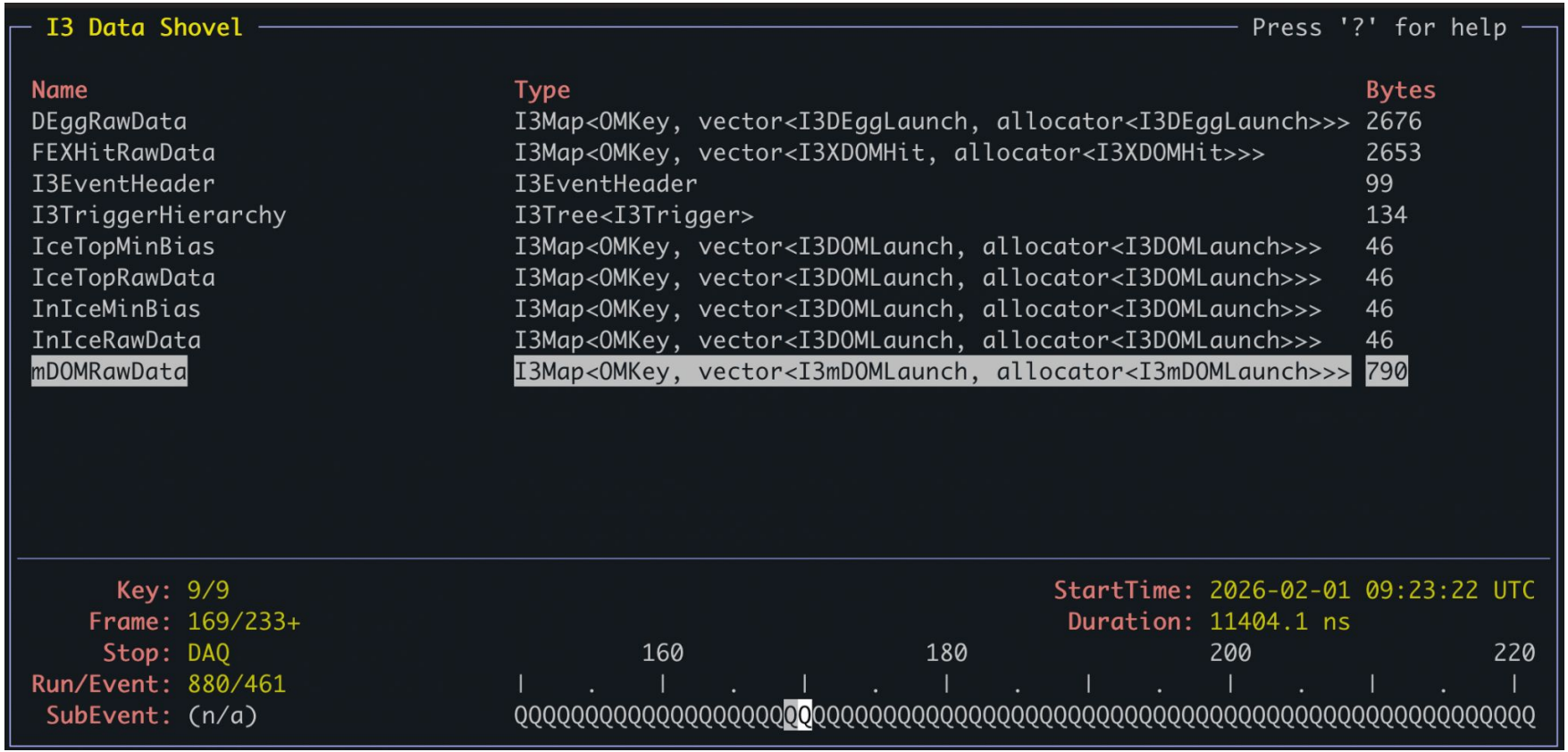
- In-ice OMs → stringhubs → pre-trigger & trigger → filter → “the north”
- Data stored in binary files → needs to be decoded to be able to read
- We use IceTray to decode and hold readout data from optical modules
  - Properly calibrate data signals (waveforms & on-board feature extractions) → timing, charge
  - Verify basic readout & calibration



What is in the data?

- Upgrade has *lots* of devices → lots of types of raw data
  - Different readout classes for each data type
- I3xDOMHit → time & charge from onboard feature extraction from mDOMs & DEggs
- I3mDOMLaunch → Full ADC, waveform readout, baseline info, from an mDOM PMT readout
- I3DEggLaunch → Full ADC, waveform readout, baseline info, from a DEgg PMT readout
- All stored per OMKey (String, OM, PMT #)

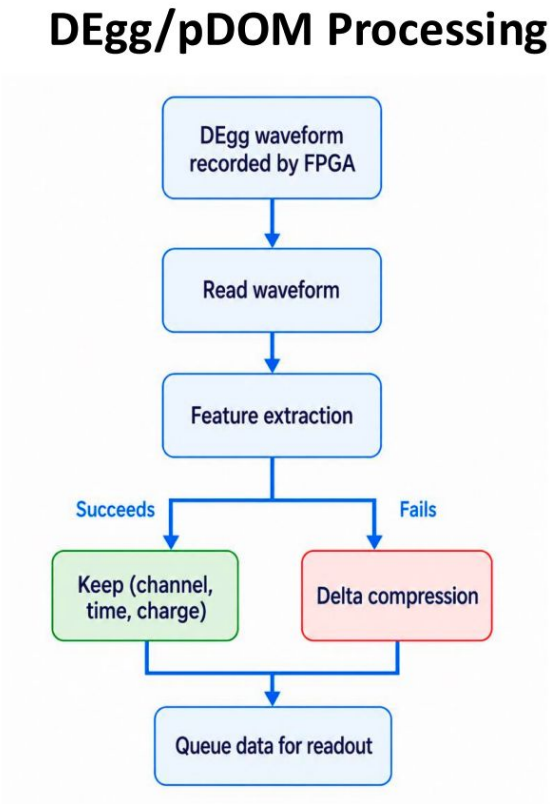
Readout classes

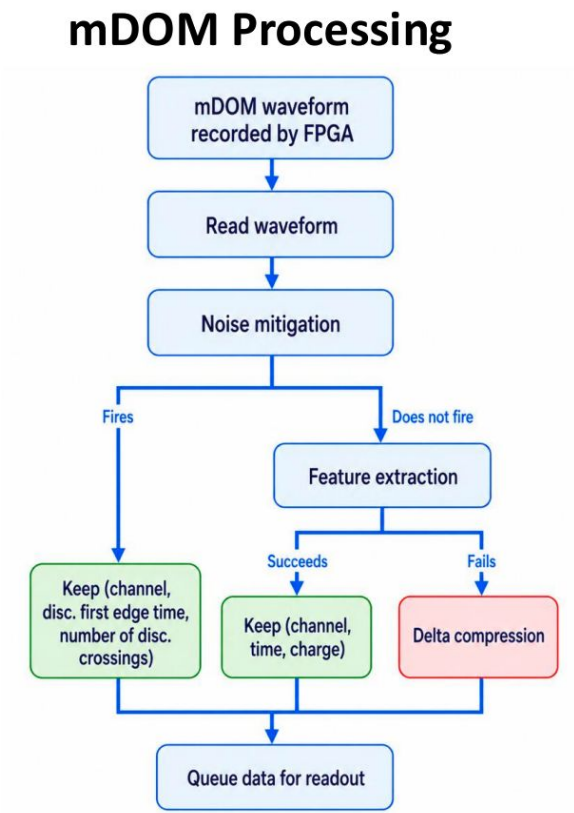


Different types of hits

- “FEX” hits → feature extracted hits
  - Single photoelectron (“SPE”) PMT hit
  - “Feature extracted” ⇒ yields only time, charge, and channel number
  - Minimizes main cable bandwidth
  - Most hits are FEX hits!
- Delta-Compressed Waveform
  - Waveforms that cannot be feature extracted, preserved for external processing
  - All information from original FPGA payload is encoded in the record
  - Waveform is compressed using delta-compression

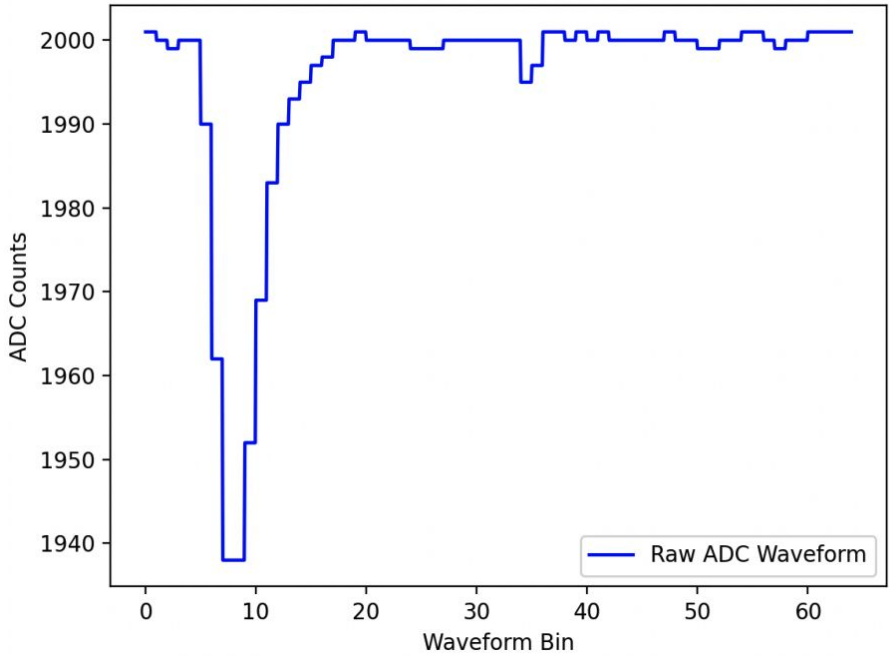
Different types of hits





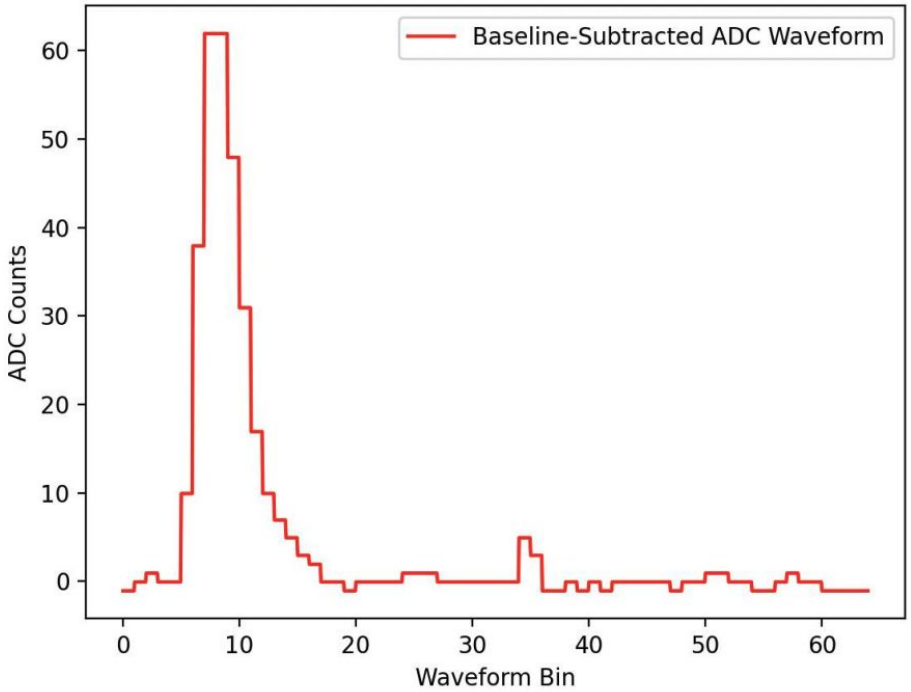
Feature Extraction Method

1. Read raw waveform payload from FPGA



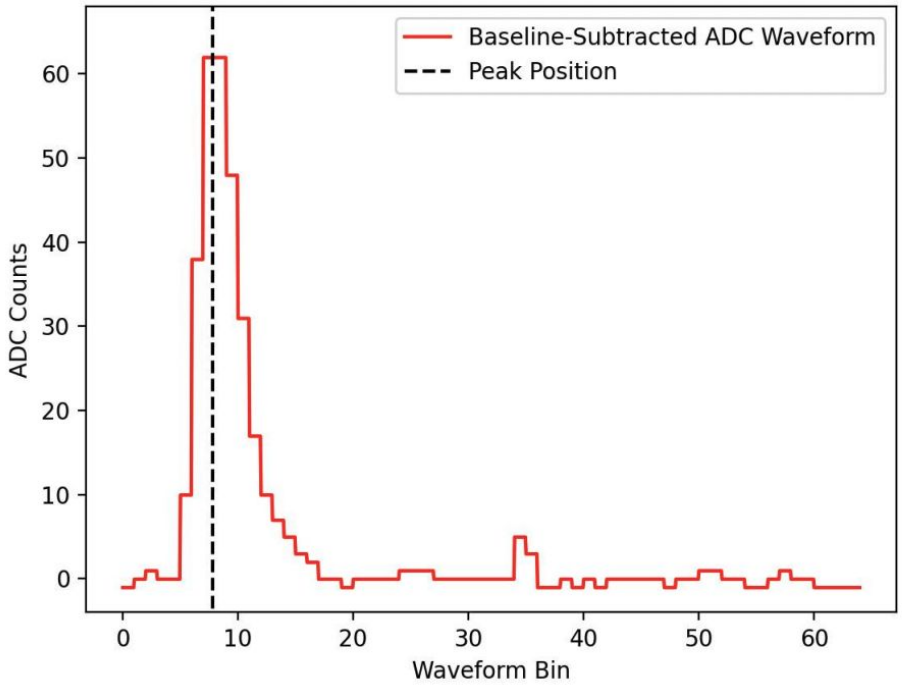
Feature Extraction Method

1. Read raw waveform payload from FPGA
2. Subtract FPGA rolling baseline



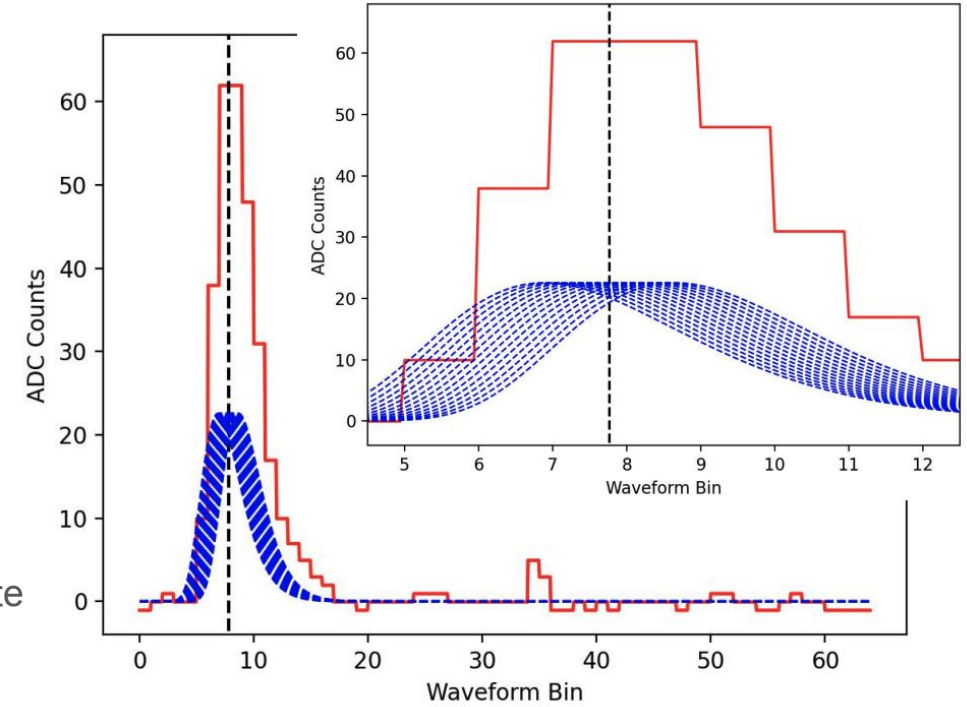
Feature Extraction Method

1. Read raw waveform payload from FPGA
2. Subtract FPGA rolling baseline
3. Find peak
   a. mDOM → use discriminator time & pre-determined offset
   b. DEgg → use quadratic fit of three bins near peak



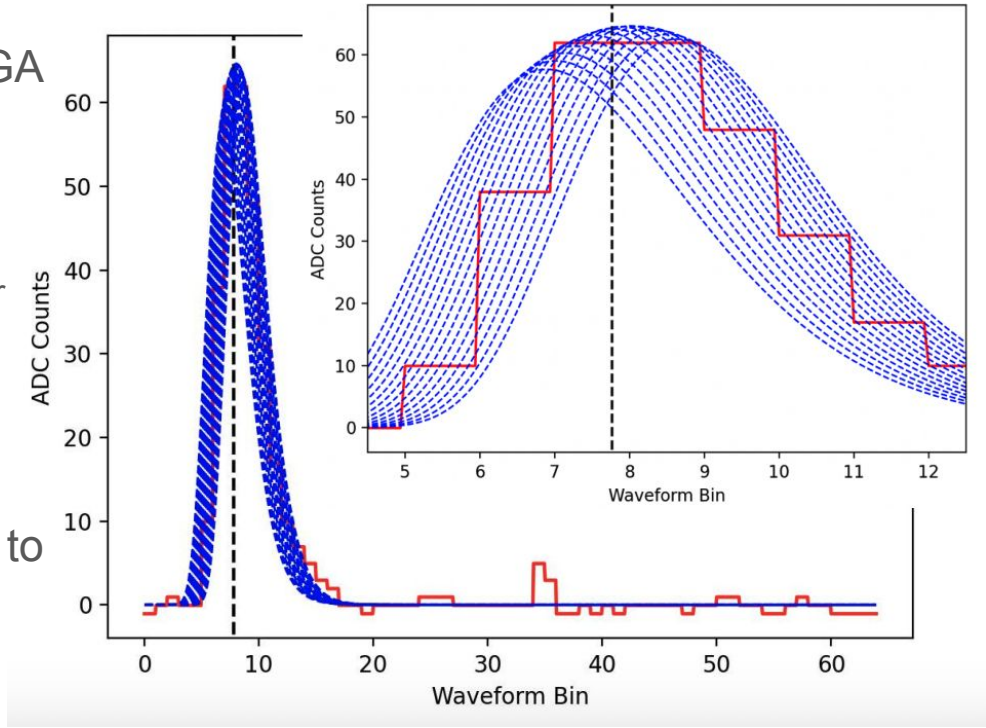
Feature Extraction Method

1. Read raw waveform payload from FPGA
2. Subtract FPGA rolling baseline
3. Find peak
   a. mDOM → use discriminator time & pre-determined offset
   b. DEgg → use quadratic fit of three bins near peak
4. Create limited basis set
   a. Basis member shape from pulse template
   b. Select evenly spaced array of members centered on peak



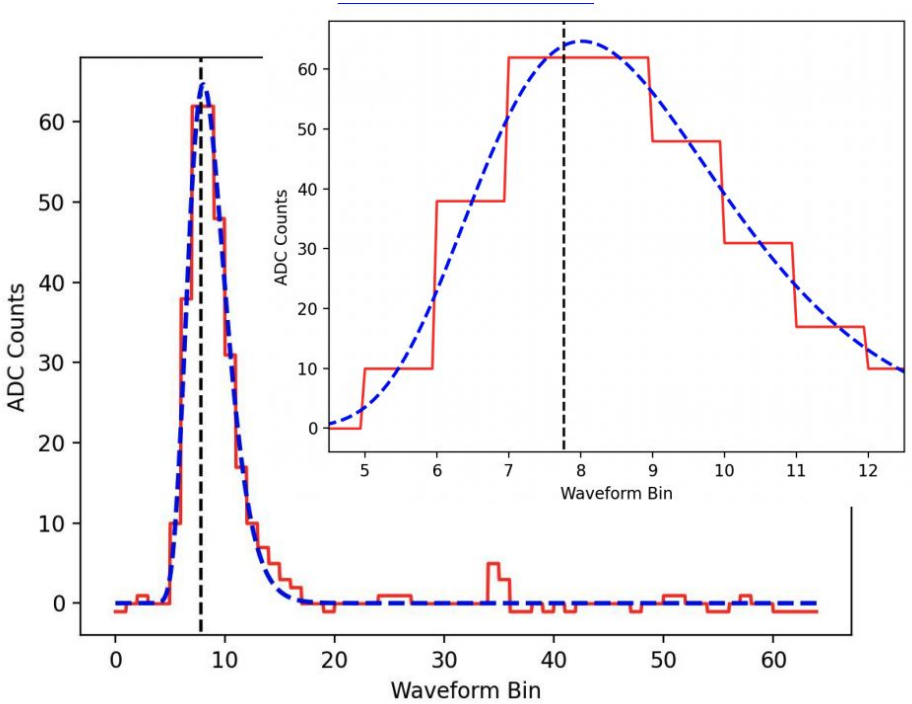
Feature Extraction Method

1. Read raw waveform payload from FPGA
2. Subtract FPGA rolling baseline
3. Find peak
   a. mDOM → use discriminator time & pre-determined offset
   b. DEgg → use quadratic fit of three bins near peak
4. Create limited basis set
   a. Basis member shape from pulse template
   b. Select evenly spaced array of members centered on peak
5. For each member: do least squares fit to waveform
   a. Inner product between basis function and waveform



Feature Extraction Method

1. Read raw waveform payload from FPGA
2. Subtract FPGA rolling baseline
3. Find peak
   a. mDOM → use discriminator time & pre-determined offset
   b. DEgg → use quadratic fit of three bins near peak
4. Create limited basis set
   a. Basis member shape from pulse template
   b. Select evenly spaced array of members centered on peak
5. For each member: do least squares fit to waveform
   a. Inner product between basis function and waveform
6. Select basis member with largest gradient as reconstructed SPE pulse
   a. Keep PMT channel, extracted charge, ICM time of extracted pulse



Feature Extraction Method

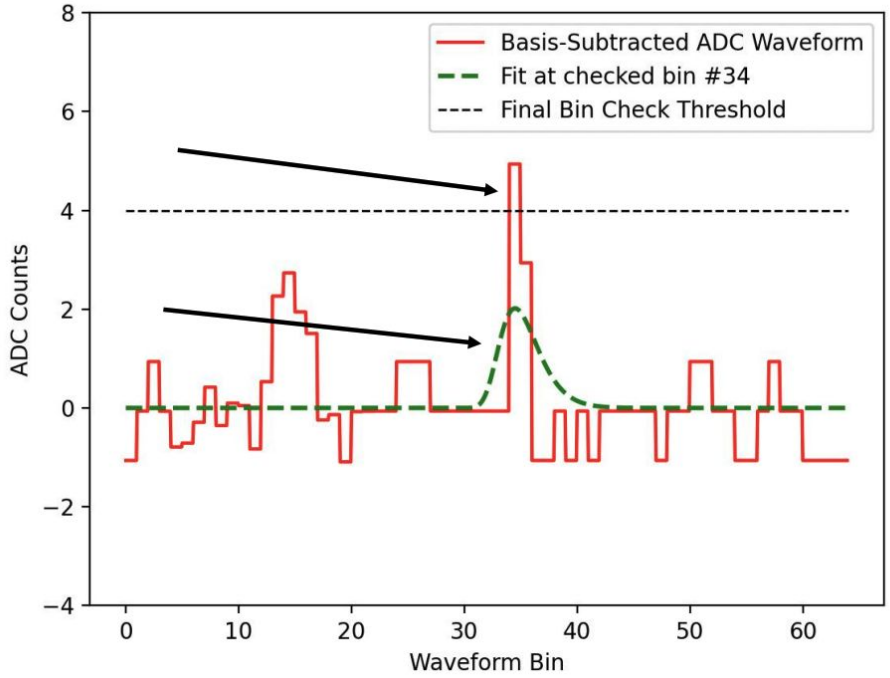
7. Subtract reconstructed SPE pulse from waveform

8. Identify “checked” waveform bins with residual amplitude above a pre-set threshold

9. Fit a basis function at each checked bin

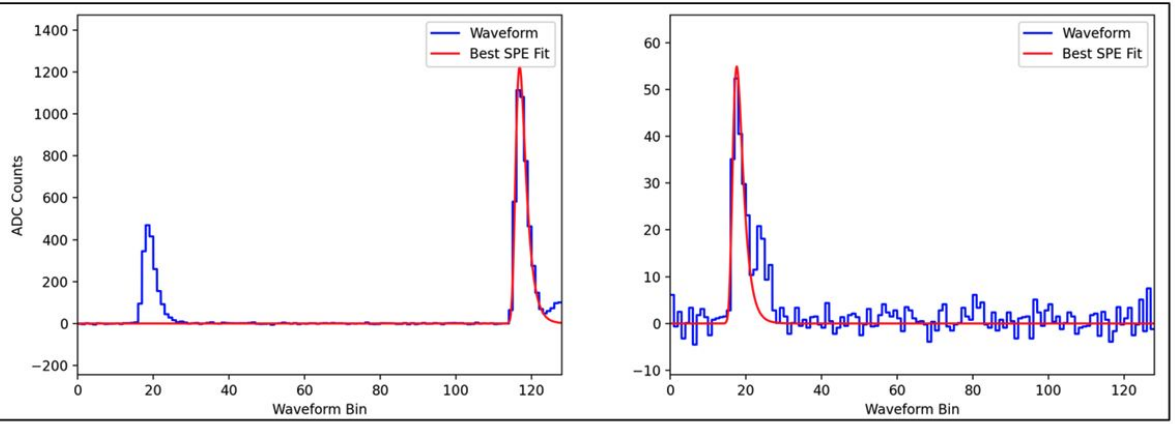
10. Reject if:

    a. Gradient (~amplitude) of any remnant basis function is above a pre-set threshold (likely is MPE)
    b. Chosen basis member is at the edge (want to be sure we found the best fit)
    c. Pulse charge is too large

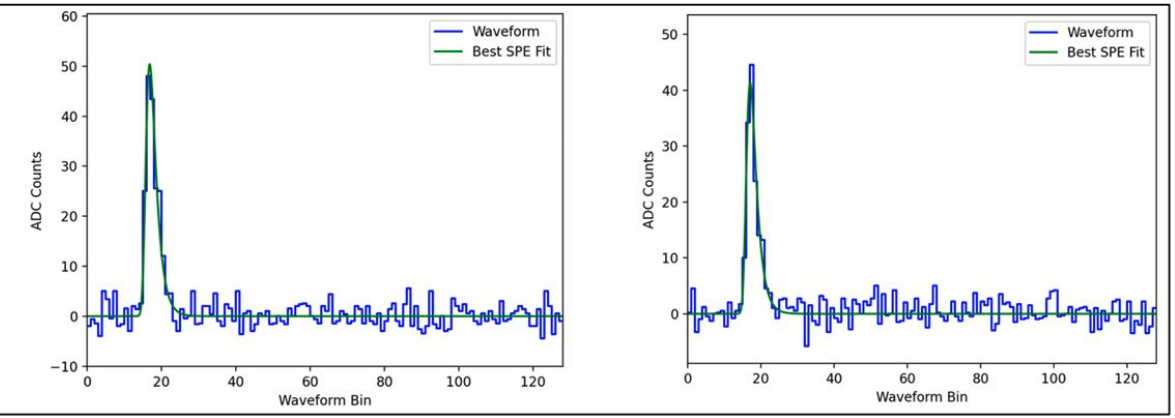


Real DEgg Example

Rejected

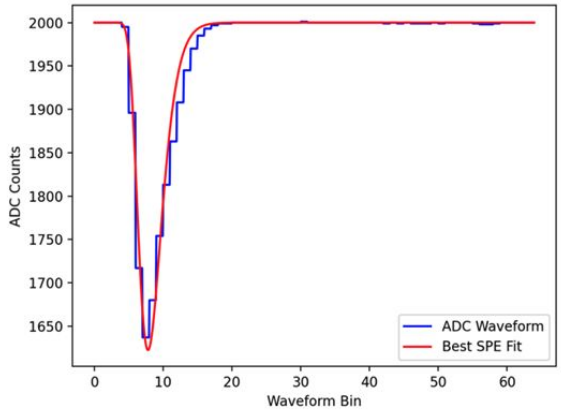


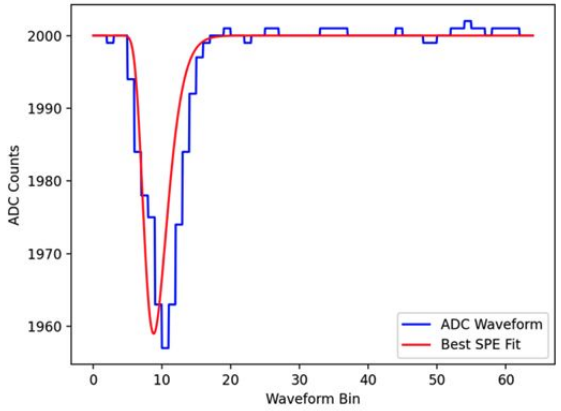
Accepted



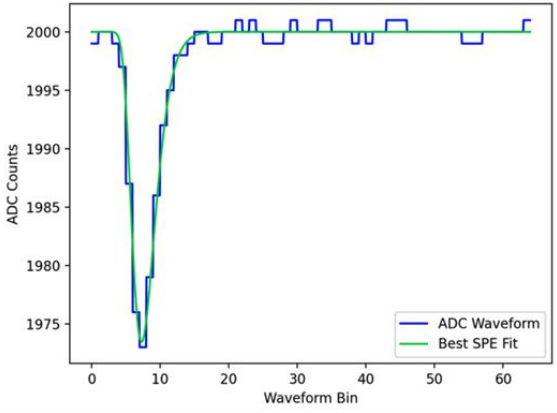
Real mDOM Example

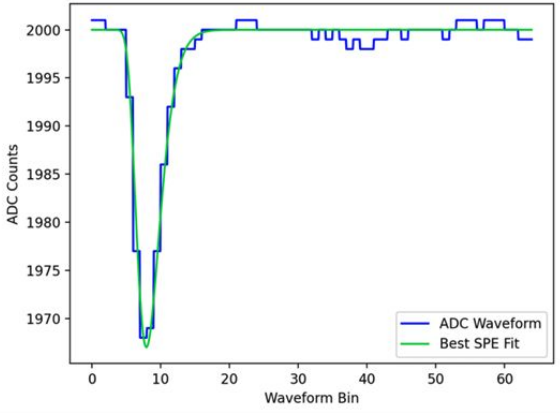
Rejected





Accepted





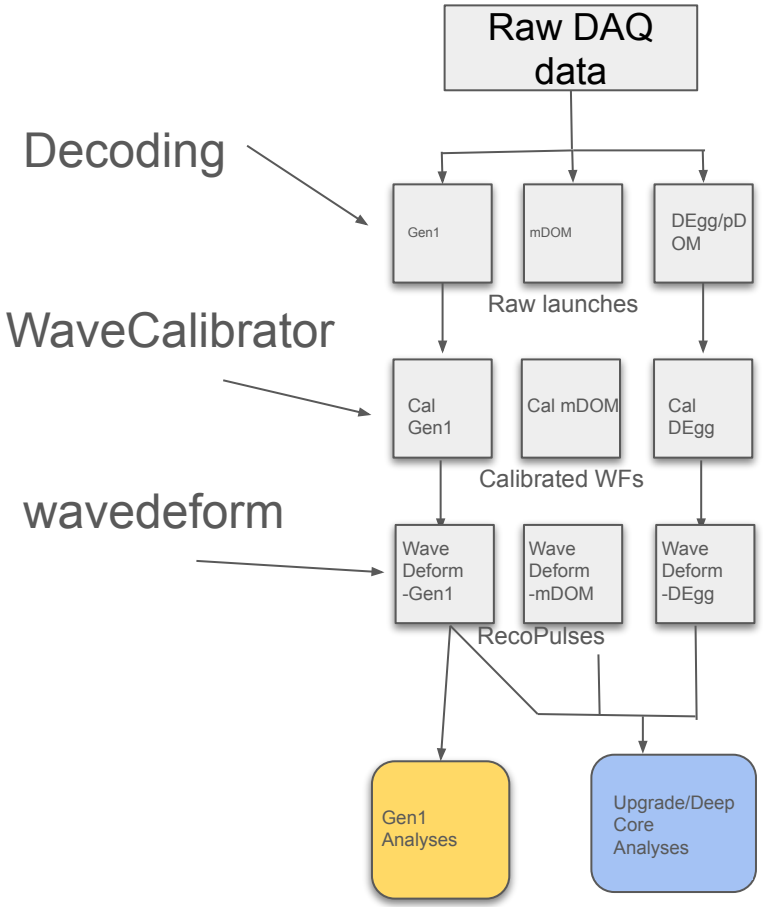
Decoding the Data

- I3DAQReader reads raw DAQ output files and convert to I3 files
- Detector settings, calibrations, etc. held in GCD files
  - For each Upgrade data taking run, there needs to be a GCD file to hold all information
  - GCDs are constantly changing as detector is further commissioned

```
input = sys.argv[1:]

for i in input:
  print("will read in file ...", i)

workspace = expandvars("$I3_SRC")
tray = I3Tray()

tray.AddService("I3PayloadParsingEventDecoderFactory", "decoder",
                minbiasid="MinBias",
                flasherdataid="Flasher",
                cpudataid="BeaconHits",
                specialdataid="SpecialHits",
                specialdataoms=SPECIAL_DATA_OMS,
                onboardledid="LED",
                testdataid="Test",
                deggdataid="DEggRawData",
                mdomdataid="mDOMRawData",
                fexhitdataid="FEXHitRawData",
                )


tray.AddModule("I3DAQReader", "reader",
               files=input,
               #skipeventsthatthrow=False,
               )

tray.AddModule("I3Writer", "writer",
               filename="./pdaq-to-i3.i3",
               )

tray.AddModule("TrashCan", "trash")


tray.Execute()
```

Workflow for Upgrade Data

- Decode raw launches
- Calibrate waveforms and feature extractions from mDOM and DEggs
- Use wavedeform to get pulse series
- Wavedeform is still in process, but we have draft code for the waveform calibration, which enables us to look at charge and time



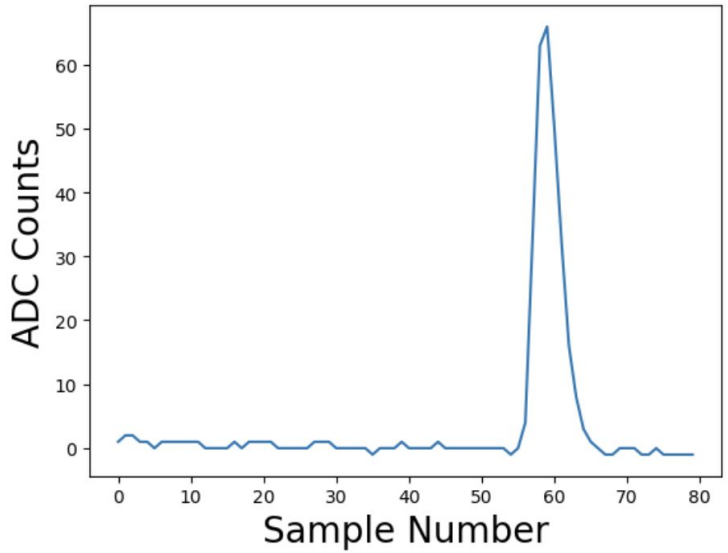
Waveform calibration: what is it?

- A raw waveform does not convey any physics information!
- Waveform calibration turns the ADC signal into a trace in units of Voltage or Current vs. time, which tells us how much light was seen by a PMT and when
- This is the information we need to reconstruct the energy, direction and flavor of particles!

Raw, baseline-subtracted waveform

DEgg: $\text{Counts} = (\text{ADC Counts}) - (\text{Baseline})$

mDOM: $\text{Counts} = (\text{Baseline}) - (\text{ADC Counts})$

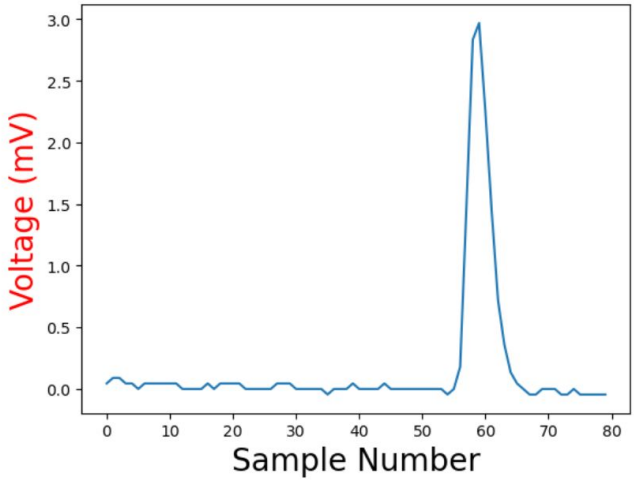


Calibrate the Waveforms…

Start by converting the ADC Counts to the PMT Voltage

- Multiply by a single conversion factor obtained for each module-type

$$\text{Voltage} = C \times (\text{ADC Counts})$$



31

ADC Counts-to-Volts factor = the conversion factor between the Analog-to-Digital Counts and the PMT Voltage

Calibration Parameters:

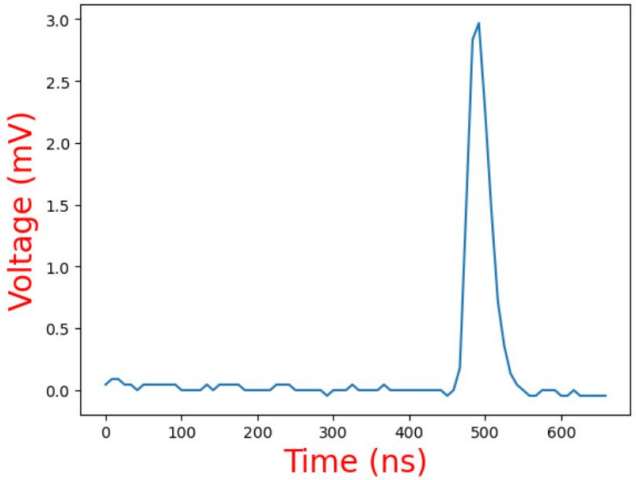
- ADC Counts-to-Volts factor

Calibrate the Waveforms…

Then convert the Sample Bin Number to a Time

- Divide by the intrinsic sampling rate of the digitizer

Time = (1/f) × (Sample Number)



32

ADC Sample Rate = the rate at which the digitizer samples the analog signal and outputs some Count value per bin

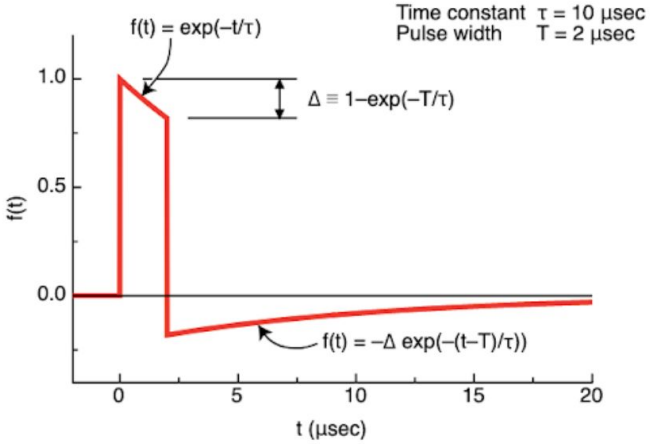
Calibration Parameters:

- ADC Counts-to-Volts factor
- ADC Sample Rate

Calibrate the Waveforms…

Then correct for waveform droop

- Both the Gen1 DOMs and the D-Eggs have a toroidal transformer coupled to their PMTs that act as a high pass filter to the signal
- This results in an exponential droop effect to the waveform
- This effect can be corrected once you obtain the time constant τ



30

Droop Time-constant parameters = the parameters that describe how the time-constant for the exponential droop decay changes as a function of temperature. This is only required for the D-Egg in the Upgrade.

$$\tau(T) = p^{(1)} + \frac{p^{(2)}}{1+e^{-T/p^{(3)}}}$$

Calibration Parameters:

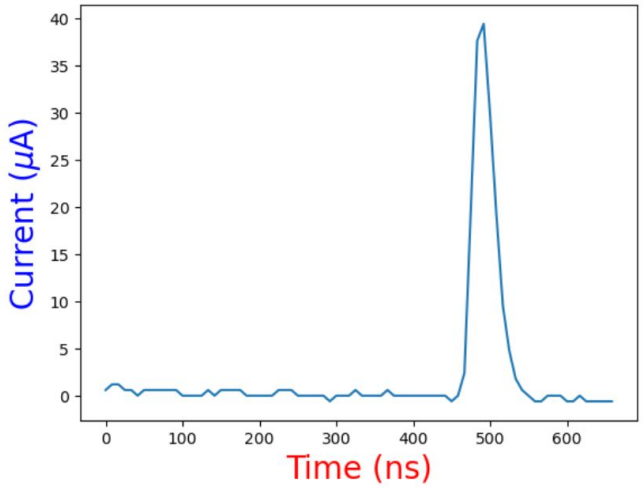
- ADC Counts-to-Volts factor
- ADC Sample Rate
- Droop Time-constant parameters **[D-Egg only]**

Pulse information is extracted…

Convert the waveform from Voltage to Current

- Divide by the total resistance of the circuits that directly interface with the PMT

$$I = \frac{V}{\Omega}$$



35

Front-end Impedance = the total resistance of the circuits that directly interface with the PMT, otherwise known as the “front-end”

Calibration Parameters:

- ADC Counts-to-Volts factor
- ADC Sample Rate
- Droop Time-constant parameters **[D-Egg only]**
- Front-end Impedance

Calculating charge

- Multiplying the current by the time in each waveform bin gives the total charge
- We typically give charge in units of “photoelectrons”, dividing charge in Coulombs by the electron charge of 1.6e-19 C
- We then divide by the gain to get the number of photoelectrons recorded by the digitizer

Calibration Parameters:

- ADC Counts-to-Volts factor
- ADC Sample Rate
- Droop Time-constant parameters **[D-Egg only]**
- Front-end Impedance
- PMT gain

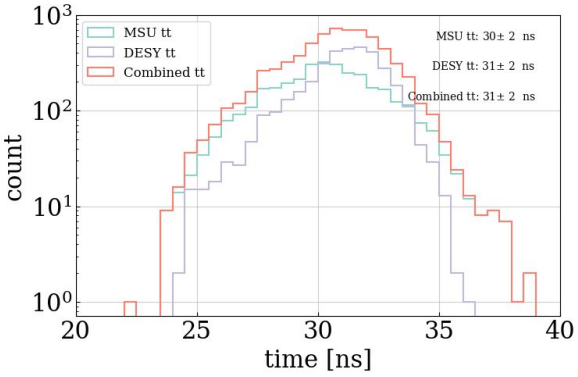
Calibration parameters

	D-egg	**mDOM**
ADC Volts/Count	0.075e-3	0.045e-3
Samping rate (Hz)	240e6	120e6
Impedance (Ohms)	36.36	75.35
Nominal gain	1e7	5e6

Relative Timing calibration

$$T_{\mathrm{Hit}} = T_{\mathrm{Launch}} + \Delta t - (T_{\mathrm{Transit}} + T_{\mathrm{offset}})$$

$$\Delta t = \frac{\text{Sample Number}}{\text{Sample Rate}}$$



mDOM transit time

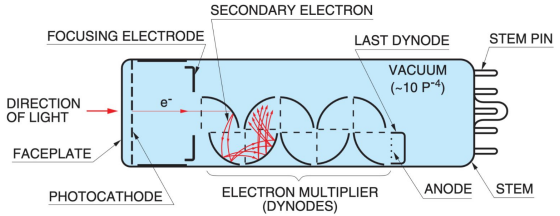
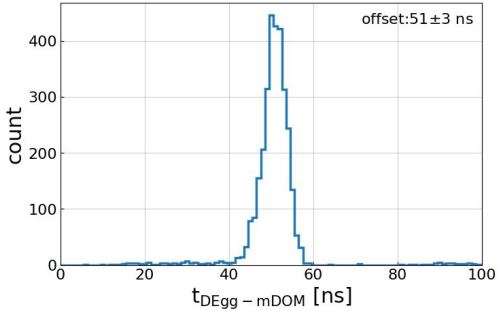


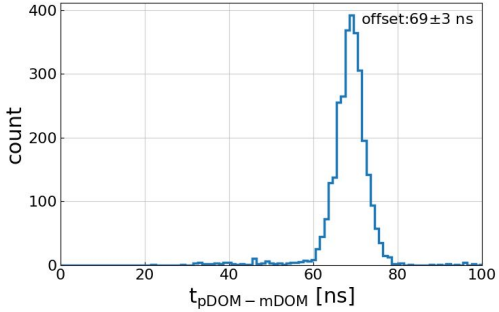
Figure 2-1: Construction of a photomultiplier tube

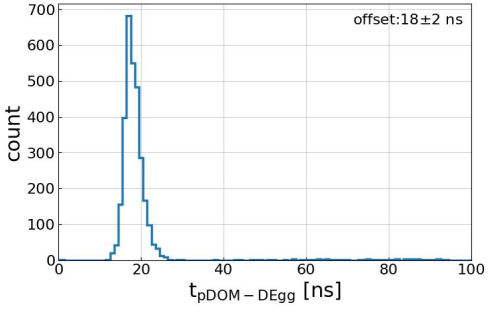
Hamamatsu: Photomultiplier Tubes, Basics and Applications

D-egg transit time =
55.21 − 0.02753 (HV − 1500)

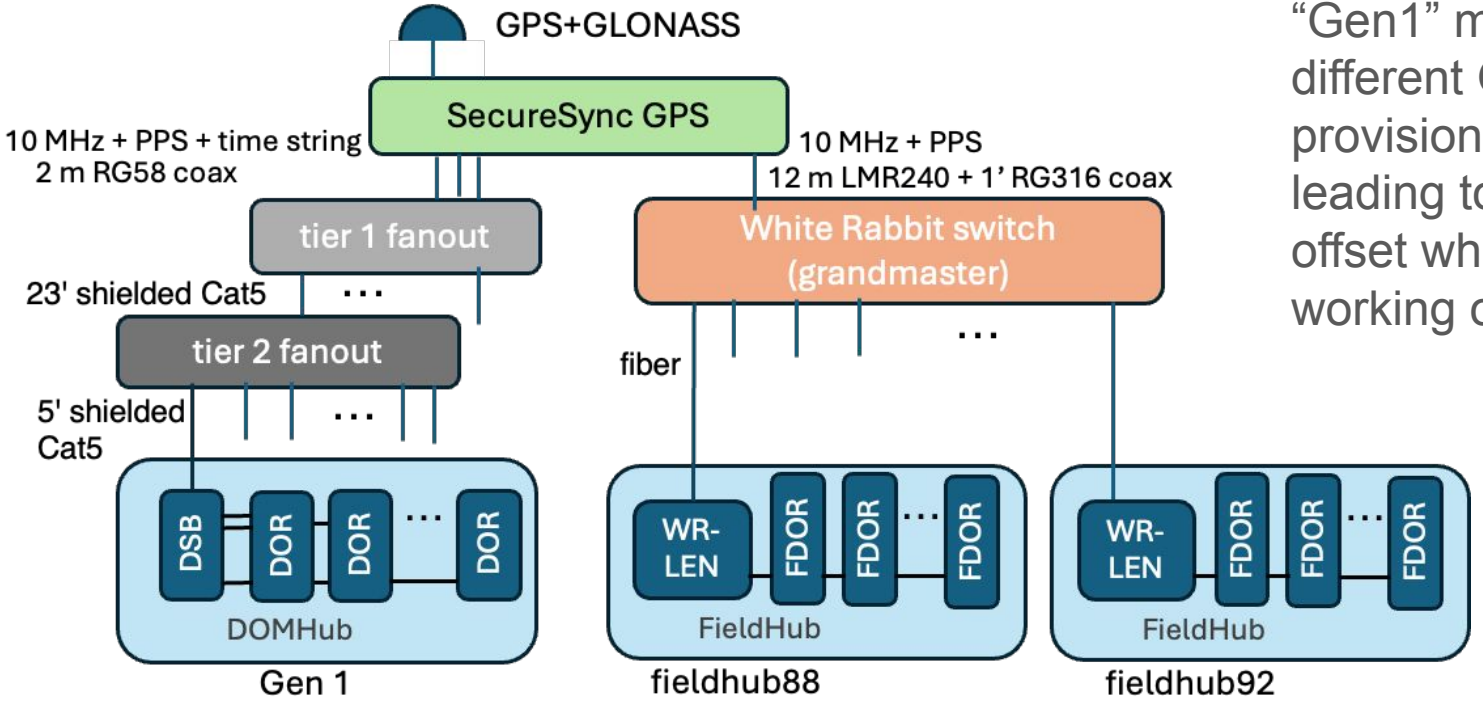
Offset times







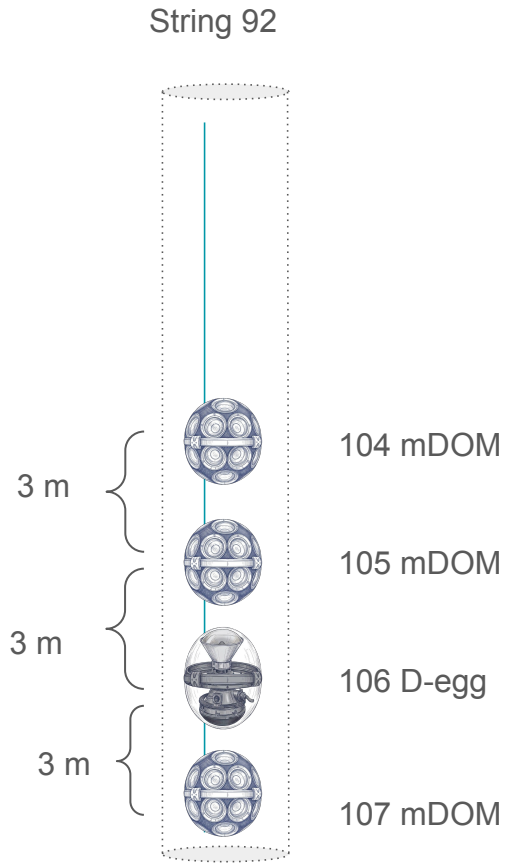
Absolute timing calibration note



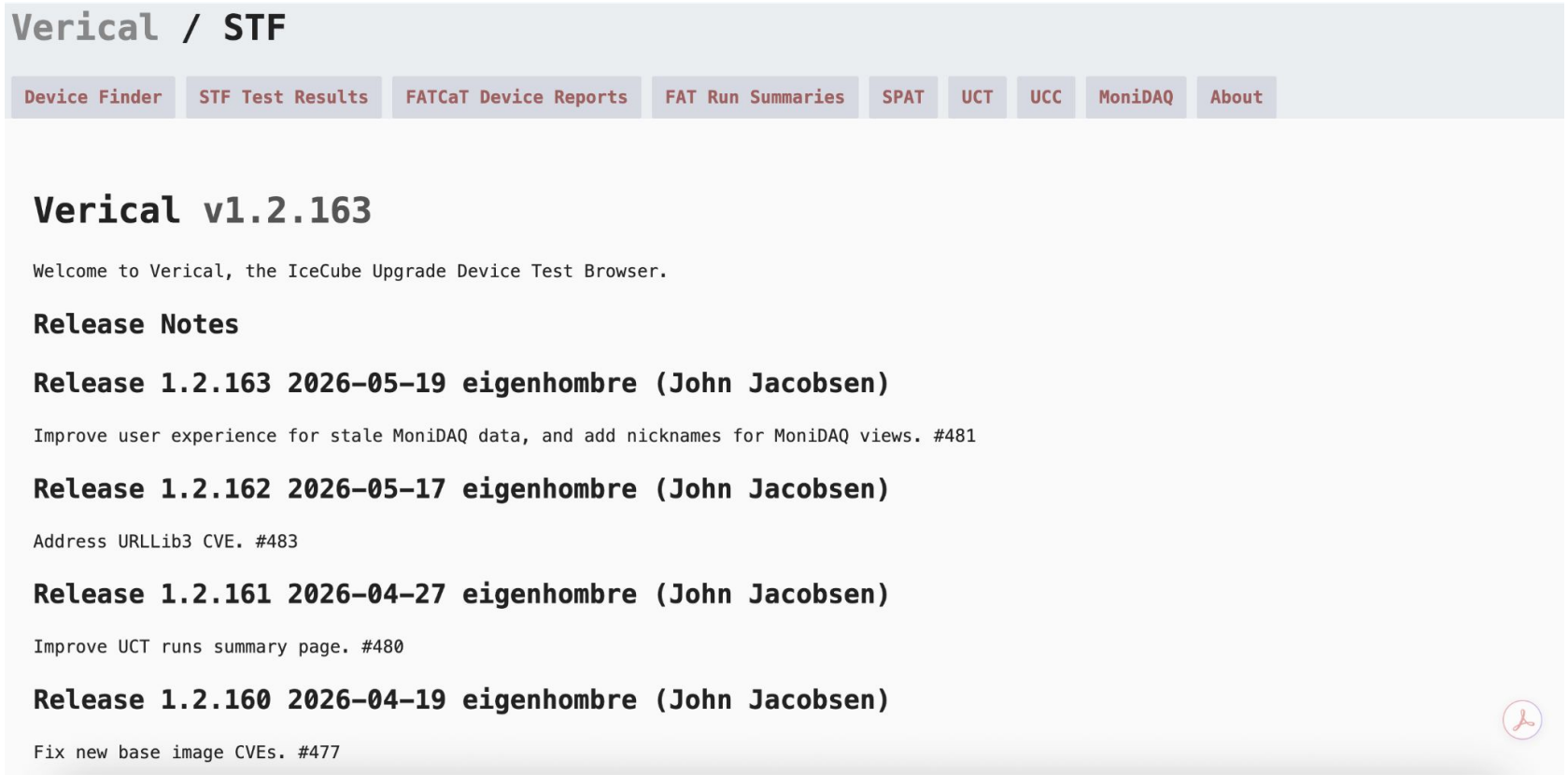
The IceCube Upgrade and the original "Gen1" modules have different GPS provisioning systems, leading to an overall offset which we are working on measuring

An example in ice

On string 92, we have an mDOM at position 104, an mDOM at position 105, a D-egg at position 106 and another mDOM at position 107. Each adjacent pair is 3 m apart. Suppose a vertical muon passes these modules. What time differences do we expect between pairs? We will look at this in the exercise.



What other data is out there? - verical!



DOMCal (UCC) & MoniDAQ

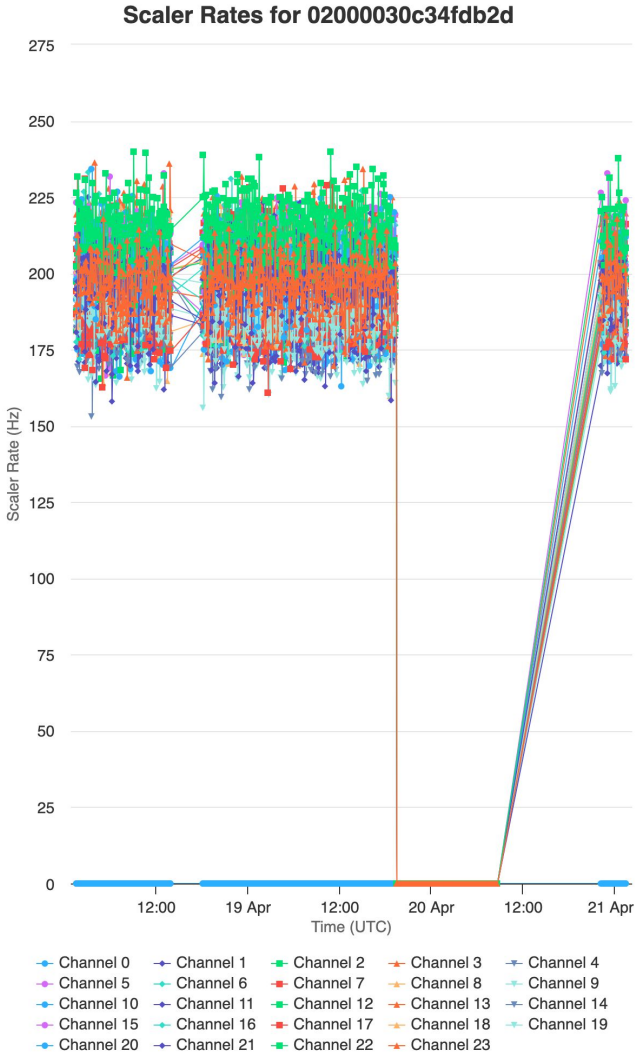
- DOMCal → Upgrade xDOM Calibration
  - Baseline, discriminator, gain calibrations
- MoniDAQ → in-ice device monitoring
  - Temperature, pressure, accelerometer, magnetometer
  - Per channel scalars and voltages for each in-ice device
  - Fieldhub wire pair voltages and currents

Example of scaler data

The scalers record how often the PMT sees hits, most of which are from local noise in the glass, not events in the ice.

Stability in the noise rates is an indicator of the health of the module

Eventually this data will be part of regular monitoring



Example of domcal data

As mentioned before, gain must be known in order to calibrate the waveforms

During the domcal process, we adjust the high voltage and calculate the gain at each step.

By fitting a line to the resulting points we can calculate the high voltage needed for any desired gain.