Optical Detector Changes

Alex Himmel, Mike Wallbank, Gleb Sinev

LArSoft Coordination Meeting

April 14th, 2015

Breaking Changes

- These are, in principle, breaking changes
  - Doesn't change data products, but can break existing code
  - In practice, all OpDet code is in larsim and larana
    - I have checked and nothing within uboonecode breaks because of this
- Under the circumstances, I think it is worth the hassle to make sure the usage is correct
  - Already found examples of current functions used incorrectly
- Suggest we include 3 OpDet changes together
  - OpDet Channel Mapping
  - New data product: raw::OpDetWaveform
  - Scintillation light pre-scaling

Introduction: Channel Mapping

- Implementing channel mapping for optical detectors.
  - Needed for DUNE 35ton – DAQ assigns channels 0-11 for each front end board
- Take advantage of ChannelMapAlg system to implement geometry-specific mappings
- Functionality should be unchanged unless optical detector channel mapping implemented for a particular geometry.

Current Naming Conventions

- "OpDet, Cryo"
  - Numbers optical detectors within a cryostat
  - Used to access geometry object:
    - `geom->Cryostat(Cryo).OpDet(OpDet)`
- "Optical channel"
  - Unique number for each optical detector

Current Functions

- Counts:
  - `int NOpDet(int c)`
    - Number of optical detectors in cryostat c
  - `int NOpChannels()`
    - Total number of optical detectors
- Conversion:
  - `int OpDetCryoToOpChannel(int o, int c)`
  - `void OpChannelToCryoOpDet(int Ch, int O, int C)`
- Other:
  - `int GetClosestOpChannel(double *xyz)`

New Naming Conventions

- ~~"OpDetCryo, Cryo"~~
  - Hide from the end user – encapsulate with geometry service
- "Optical detector"
  - Unique number for each optical detector in the geometry
- "Hardware channel"
  - Refers to a channel number specific to a piece of electronics, can repeat
- "Optical channel"
  - Unique numbering of each hardware channel

Changes to Current Functions

- Counts:
  - ~~`int NOpDet(int c)`~~
    - Available in `CryostatGeo` if looping within a single cryostat is absolutely necessary
    - Frequently used incorrectly, causing code to be incorrect with >1 cryostat
  - `int NOpDets()`
    - Total number of optical detectors
  - `int NOpChannels()`
    - Total number of optical channels across all detectors

- Conversion:
  - ~~`int OpDetCryoToOpChannel(int o, int c)`~~
  - `int OpDetCryoToOpDet(int o, int c)`
  - ~~`void OpChannelToCryoOpDet(int OpChannel, int O, int C)`~~
  - `const OpDetGeo& OpDetGeoFromOpDet(int OpDet)`
  - `const OpDetGeo& OpDetGeoFromOpChannel(int OpChannel)`

- Other:
  - ~~`int GetClosestOpChannel(double *xyz)`~~
  - `int GetClosestOpDet(double *xyz)`

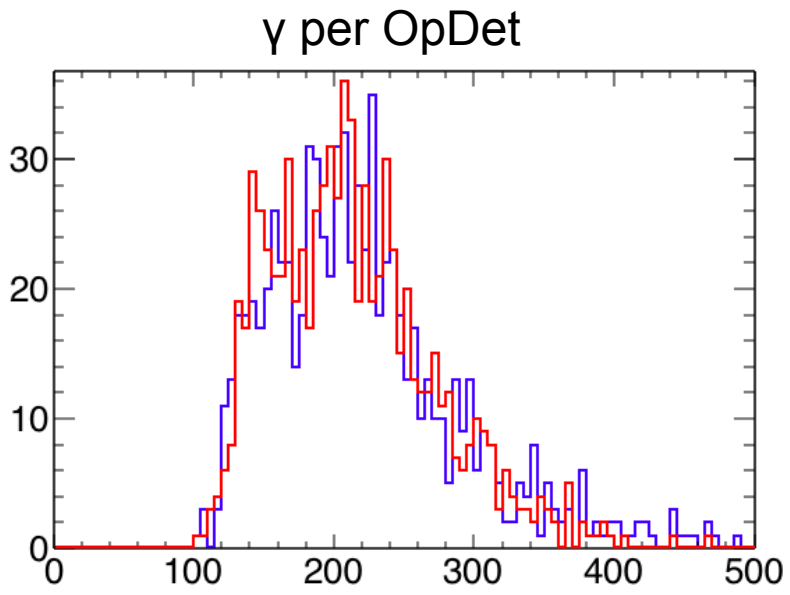
New Functions

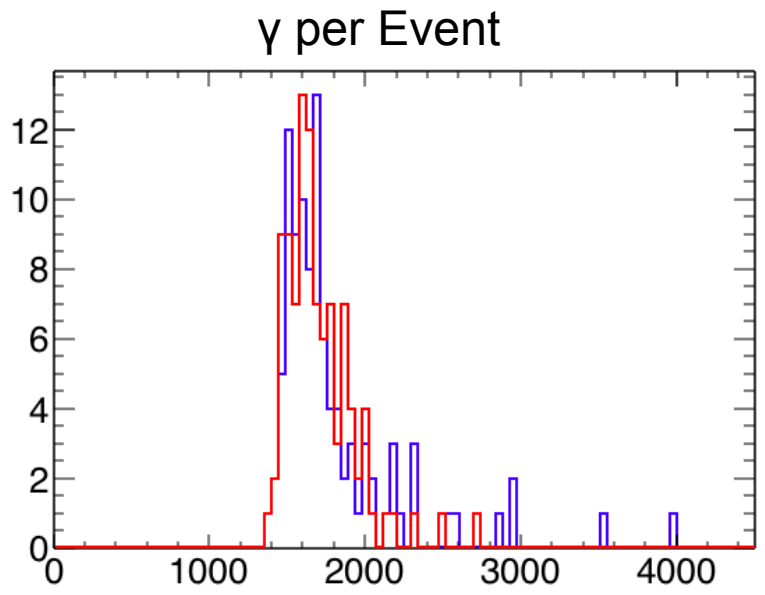
- `int NHardwareChannels(int opDet)`
  - Number of hardware channels for a given optical detector
- `int OpChannel(int opDet, int hChannel)`
  - Optical detector + Hardware Channel → Optical Channel
  - Defaults to returning `opDet`
- `int OpDetFromOpChannel(int opChannel)`
  - Optical Channel → Optical detector
  - Defaults to returning the `opChannel`
- `int HardwareChannelFromOpChannel(int opChannel)`
  - Optical Channel → Hardware Channel
  - Defaults to returning `0`

Other Modifications

- Some usage changes required in larsim and larana
  - The tricky thing is taking care to use the right reference number – channels or opdets
    - Library can be generated with either filling the "channel"definition
    - OpDetResponseService can handle choosing the right variant
- Tested running photon simulation with an existing library in:
  - lbnecode (without channel mapping)
  - lbnecode (with channel mapping)
  - uboonecode (no channel mapping so far)

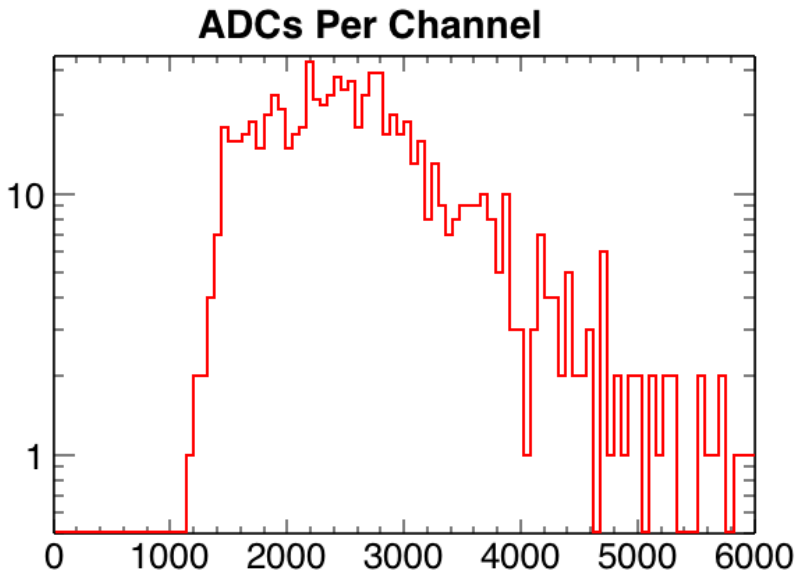
lbnecode: geant4

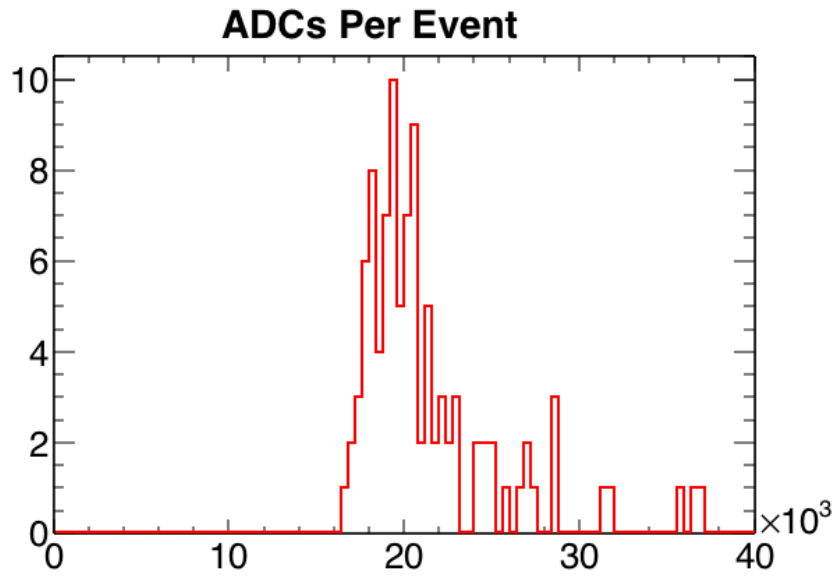




- First, at the geant4 level:
  - Good agreement both in photons per optical detector and photons per event.
  - Existing library loads correctly
  - Statistical variation due to different random seeds

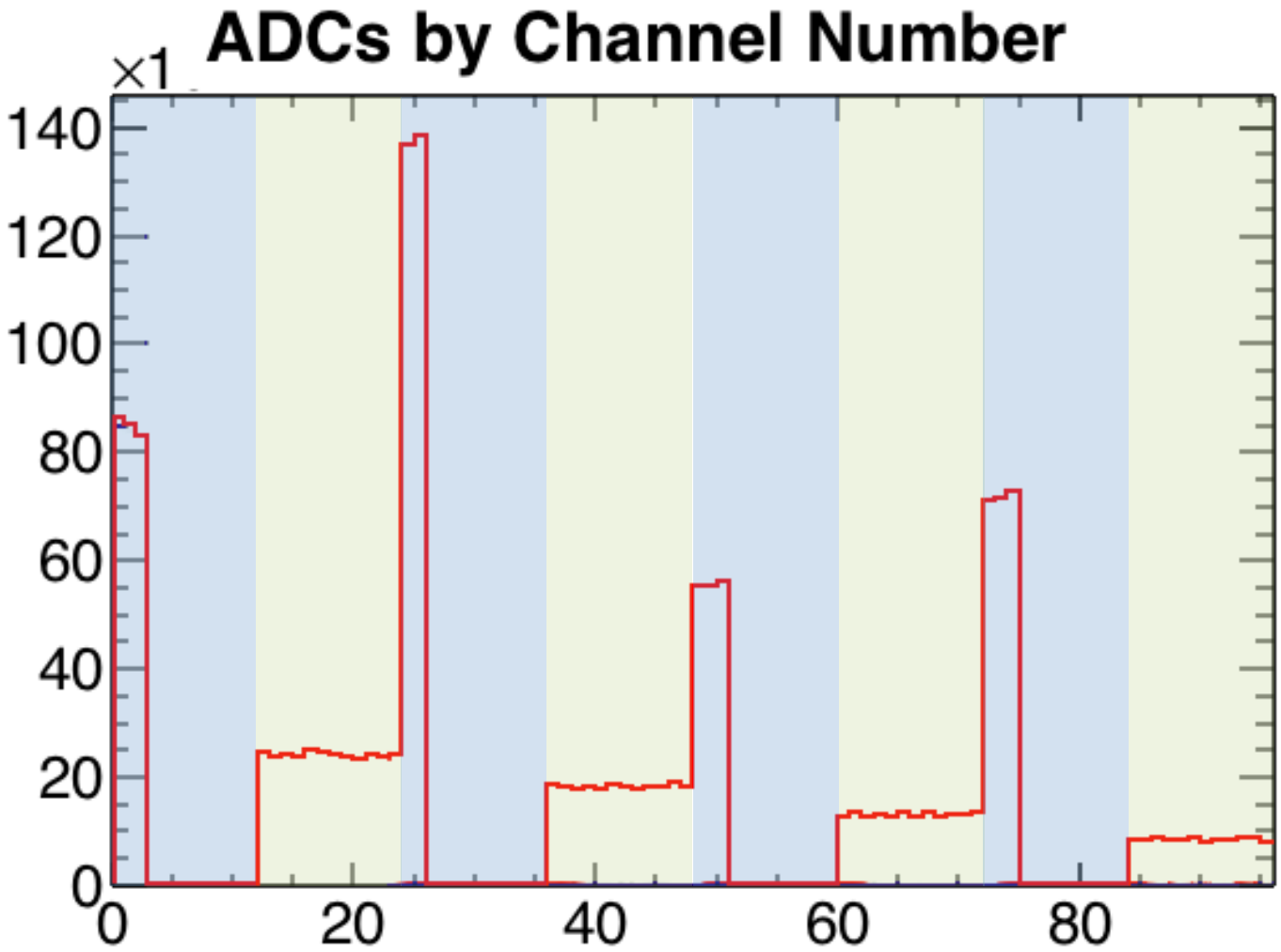
lbnecode: digitized





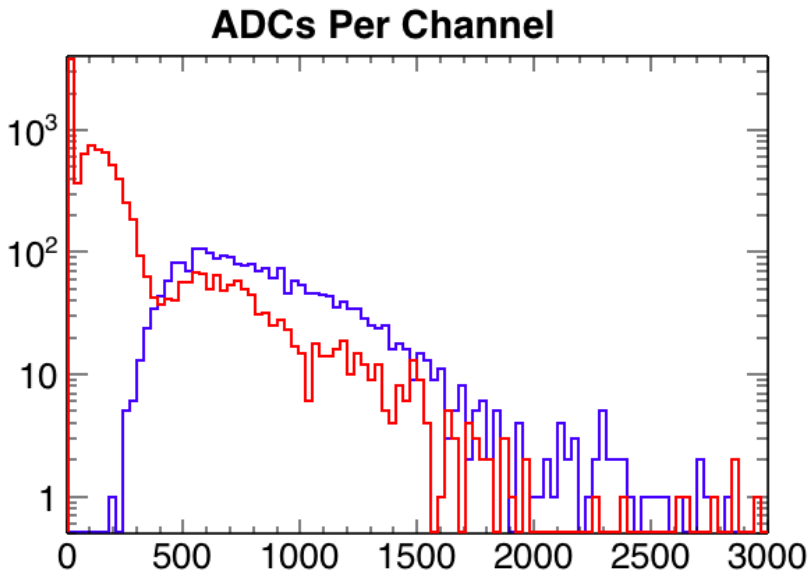
- When channel mapping is not in use, the matching is exact.
  - Started from the same g4 file

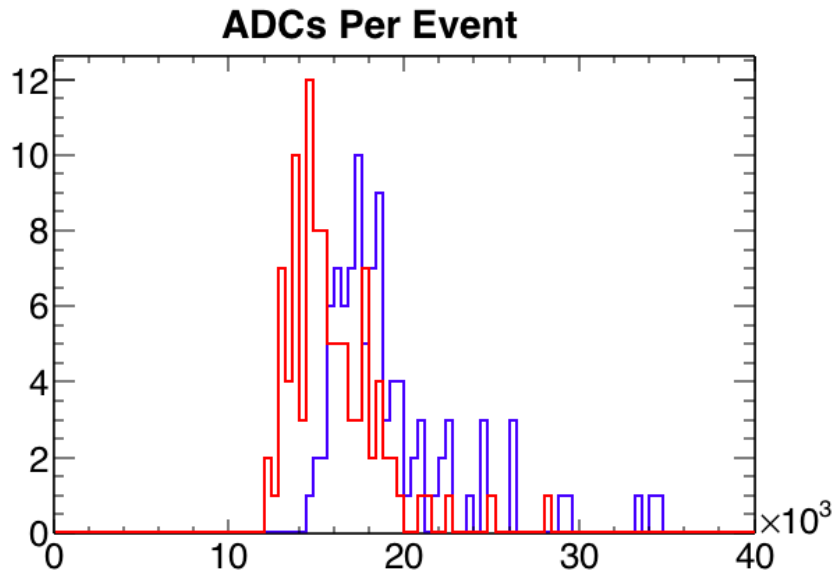
lbnecode: digitized



- Channel mapping behaving as expected
  - 2, 3, or 12 channels per opdet

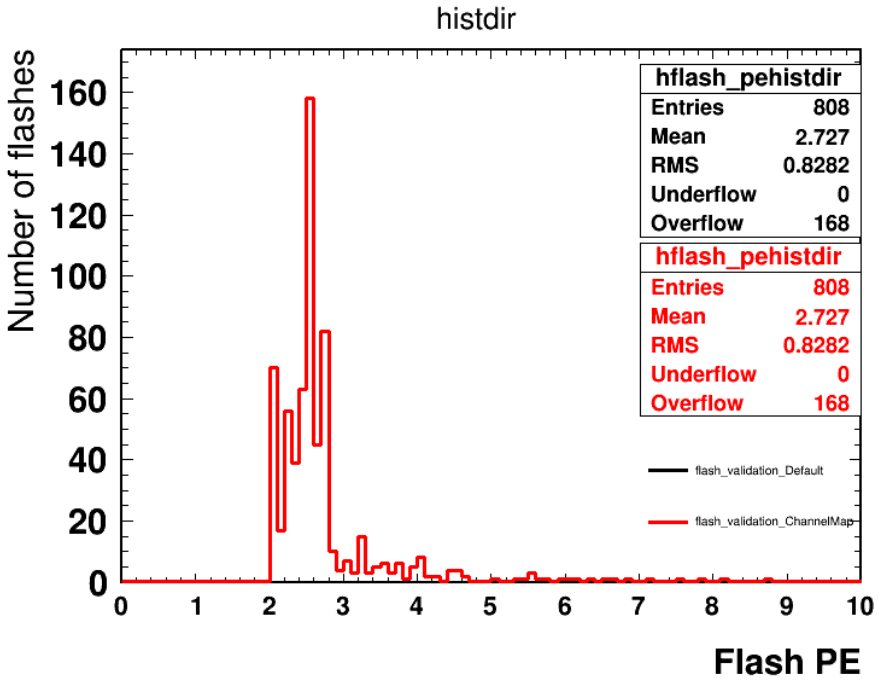
lbnecode: digitized

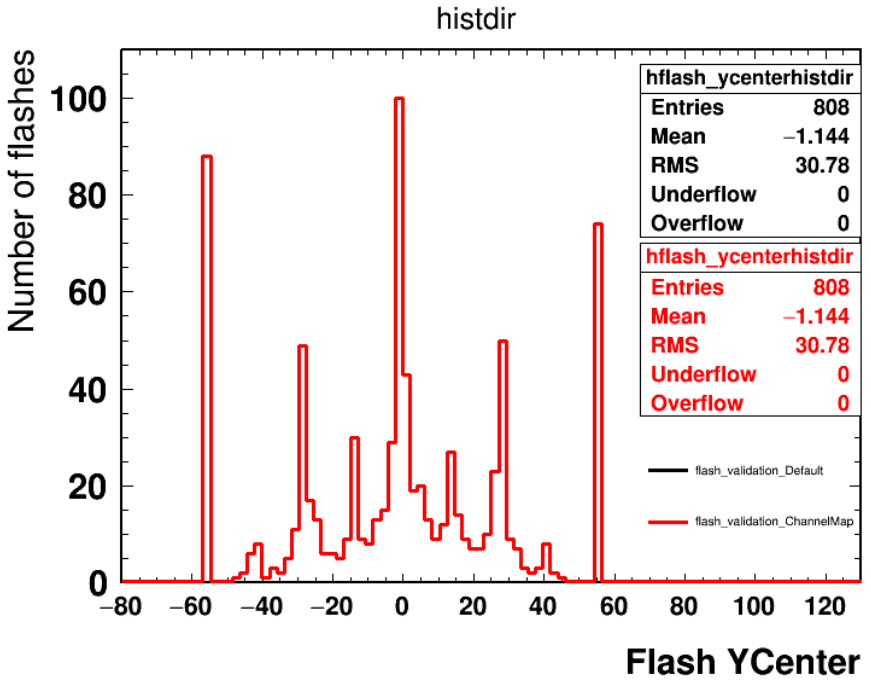




- DUNE digitizer turns each PD into 2, 3, or 12 channels.
- Fewer counts per channel
- Nearly the same total number of ADCs
  - Some loss in the float → int step because of the digitization scale
  - Tested with a digitizer we're about to deprecate, so it wasn't worth fixing.

uboonecode: Reco





- uboone chain from generator through 2d reco.
- Identical results
- Would like to test FlashHypothesis
  - Need to find some sample files/code to work with.

Conclusions: Channel Mapping

- Can be found on:
  - larcore/feature/wallbank_NewOpticalChannelMap
  - larsim/feature/ahimmel_NewOpticalChannelMap
  - larana/feature/ahimmel_NewOpticalChannelMap
  - lbnecode/feature/wallbank_NewOpticalChannelMap
- Testing so far looks good
- Breaking changes, but I think worth the cost for clarity and correctness moving forward

Introduction: Shared Raw Data Format

- The goal is to have a single data product that can be used by all experiments to hold raw Optical Detector data
  - Waveform, channel number, time information
- By having a shared data format, we can create reusable reconstruction algorithms
  - For example, want to take advantage of μBooNE flash hypothesis.
- Want to implement this ASAP so it can be used in raw data monitoring, event displays, etc. etc.

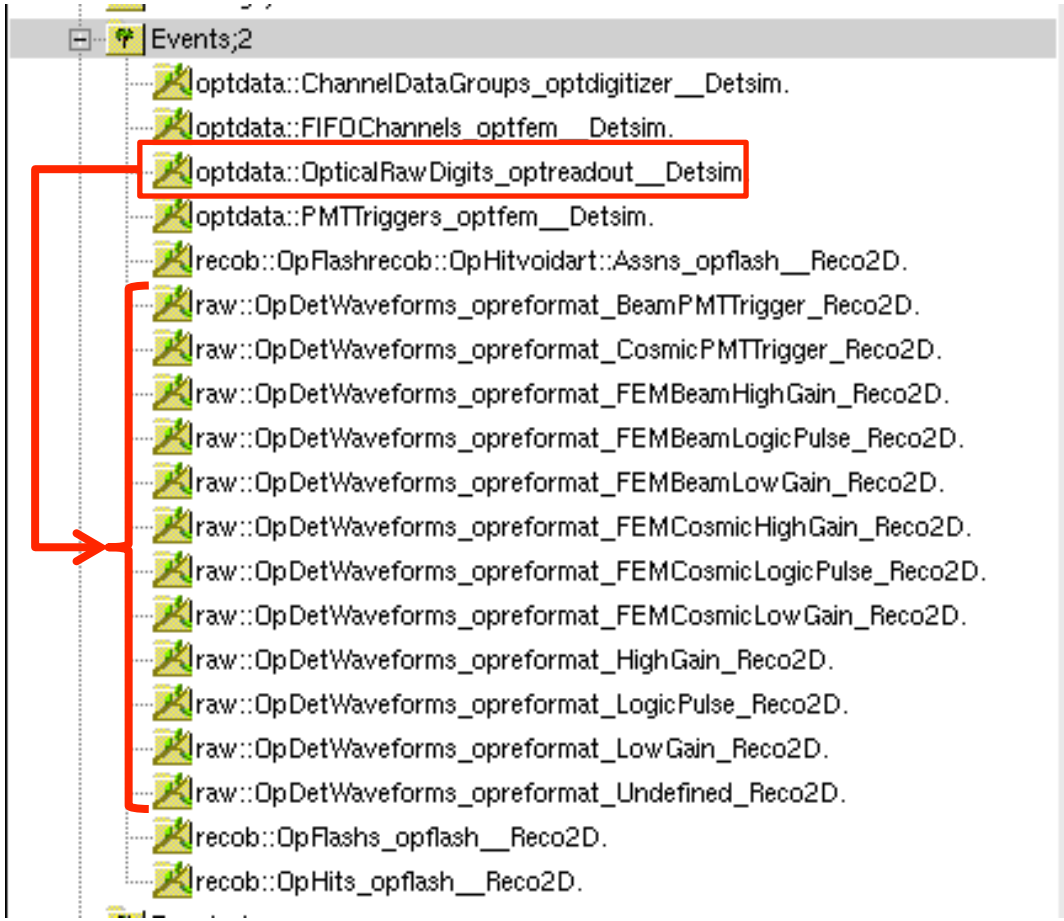
raw::OpDetWaveform

Waveform	vector<short>	• Allows bipolar signals • Inherits from this for consistency with algorithms
ChannelNumber	int	
TimeStamp	double	• Absolute time stamp in μs • Comes from TimeService

- A new data product designed to be experiment-agnostic.
- Avoid coding in electronics-specific time formats
  - TimeSlice/Frame
  - NOvA 64bit time stamp
  - etc.
- No experiment-specific DAQ information
  - MicroBooNE categories

Reformatter

- Convert OpticalRawDigit into OpDetWaveforms
- Existing files will not need to be remade
- Microboone “categories” are made into separate collections
  - Processed together by the flash finder
- 1 producer module to add to µBooNE’s Reco2D

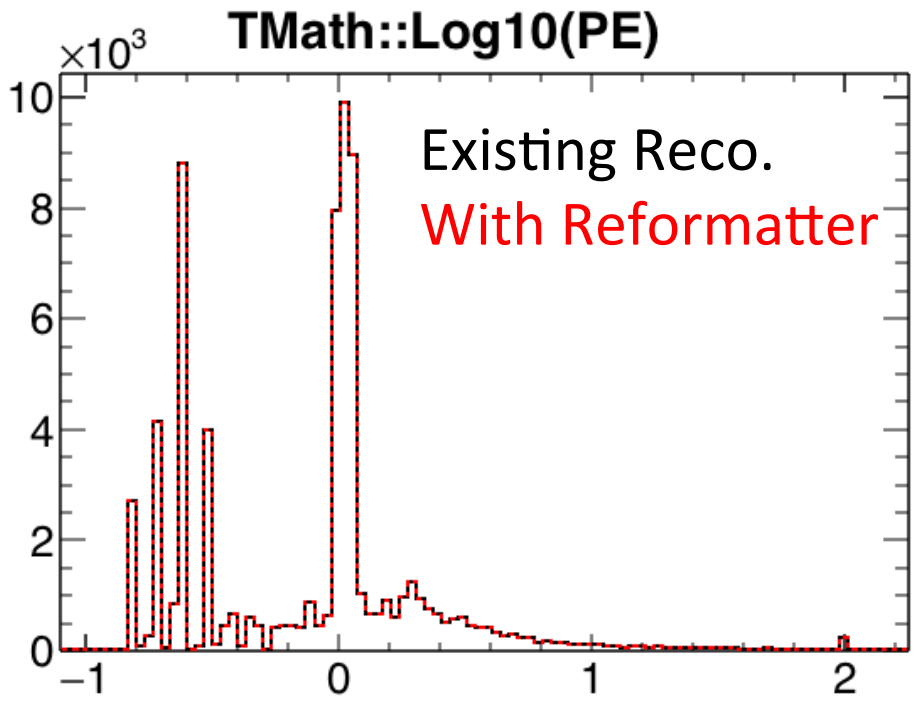


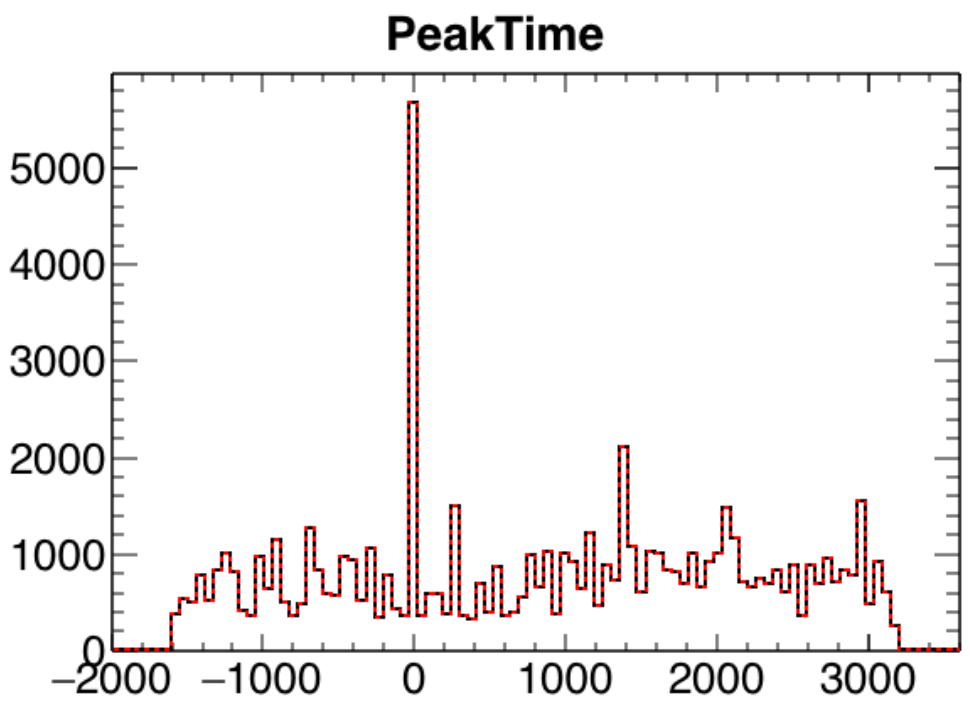

```
### OpticalRawData Converter
opreformat:          @local::microboone_opdetreformatter
```

FlashFinder

- FlashFinder has been made more independent of μBoone
  - Before, each μBooNE event is broken into several time regions called "frames"
  - Now, all hits are processed together
  - 1 FHICL parameter changed to use real time (μs) instead of ticks
- Uses getManyByType to access all category collections
  - InputModule fhicl parameter is now ignored. Remove from interface?
- Other improvements along the way
  - OpHit-OpFlash associations now working
  - Length of the readout window length no longer must be known ahead of time – adapts on the fly
- Have been testing with 10 μBooNE neutrino MC events

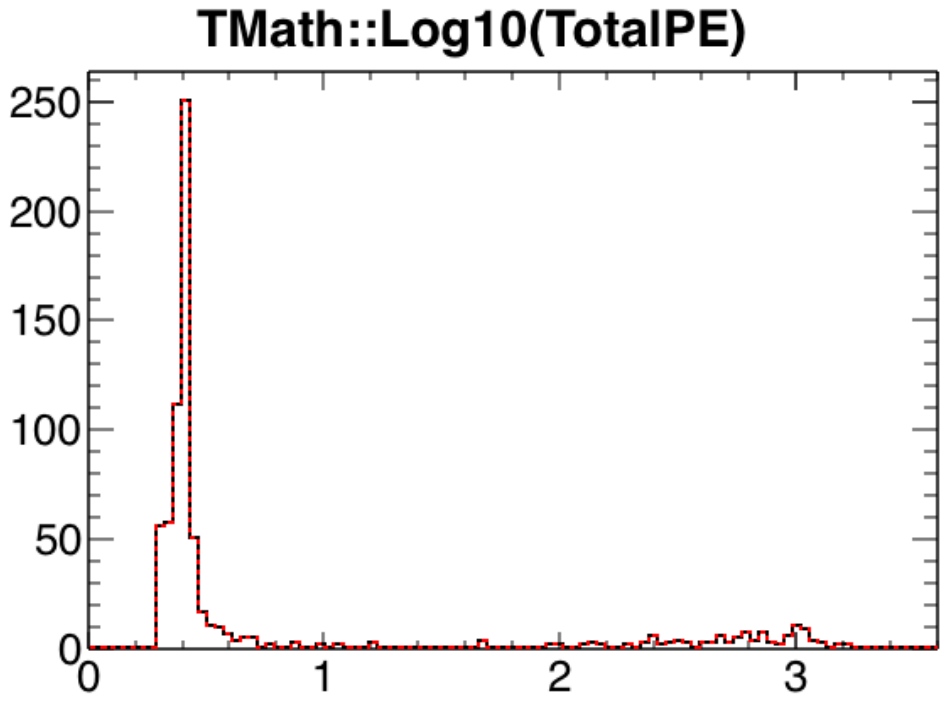
Validation: OpHits

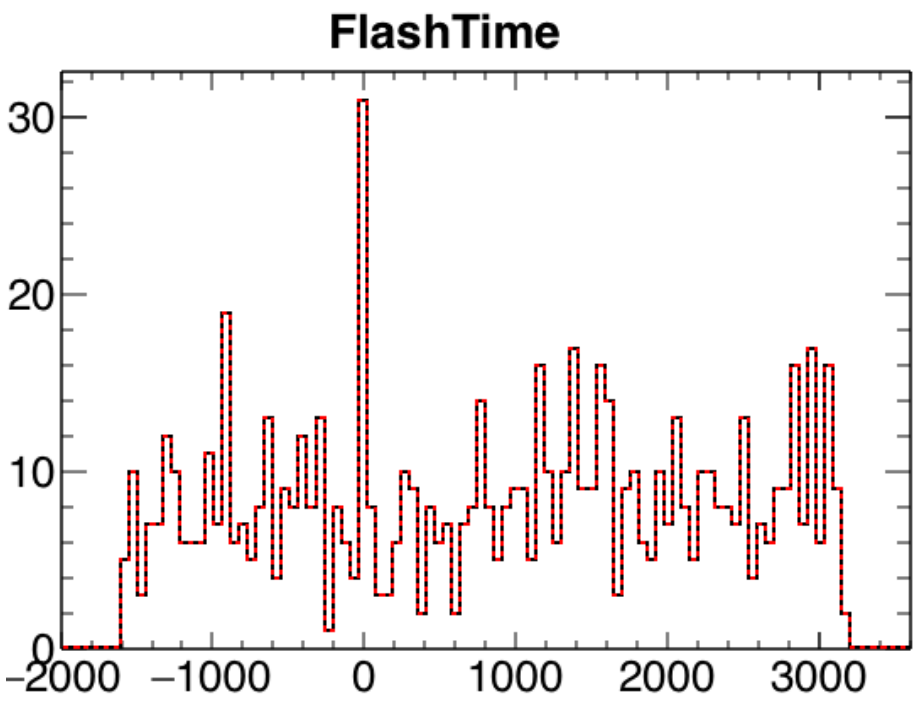




- The hits are reconstructed identically
  - Distributions checked, as well as hit-by-hit

Validation: OpFlashes





- Flashes are also reconstructed identically
  - Distributions checked, as well as flash-by-flash
- The only exception is the "Frame" variable associated with the flash
  - For backwards compatibility, this is recalculated from the final flash time
  - 1 flash (out of 736) which is right on the border where the frames don't match

Conclusions: Shared Raw Data Format

- Changes can be found on:
  - lardata/feature/SharedOpticalRawDigit
  - larana/feature/SharedOpticalRawDigit
- This change is non-breaking
  - Reformatter allows old files to be read in
  - Output recob objects are unchanged
- Reconstructed objects left unchanged for now
  - recob::OpHit and recob::OpFlash
  - These objects have a “Frame” variable we may want to remove

Introduction: Simulation Prescaling

- Argon scintillation produces ~24,000 photons/MeV
- In practice, only a small fraction are collected since QE reduces this by a large factor
- Simulating all of the photons, only to throw away 99% is wasteful, often impossible due to memory constraints

Current Solution

- Past solution: scale the amount of scintillation light in the fhicl files by a QE factor:

```
# set quantum efficiency supressed scint yield to 0.03 * 24000
services.user.LArProperties.ScintYield: 720
```

- Easy to get wrong since the QE is multiplied into another number and then not accessible
- Breaks down with more complicated QE's
  - vs. wavelength, position, etc.

Prescaling

- A better solution is to add an additional LArProperty: `ScintPreScale`
- Current:

```
ScintYield: 24000 → 720
QE:             0.03 →   1
```

- With Prescaling:

```
ScintYield:    24000
ScintPreScale:  0.03
QE:             0.03
```

Prescaling Implementation

- Incorporated directly into the ScintYield property used in simulation:

```
//Return the scaled scint. yield
LarProp->ScintYield(true);
```

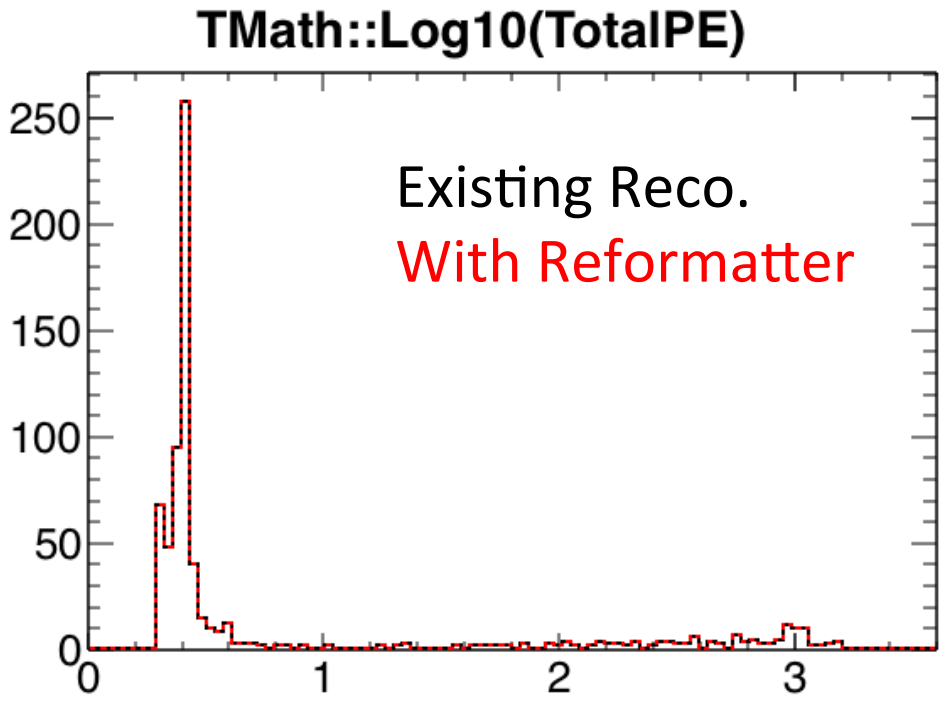
- Corrected out when QE is set.
  - lbnecode (in LBNEOpDetResponse)

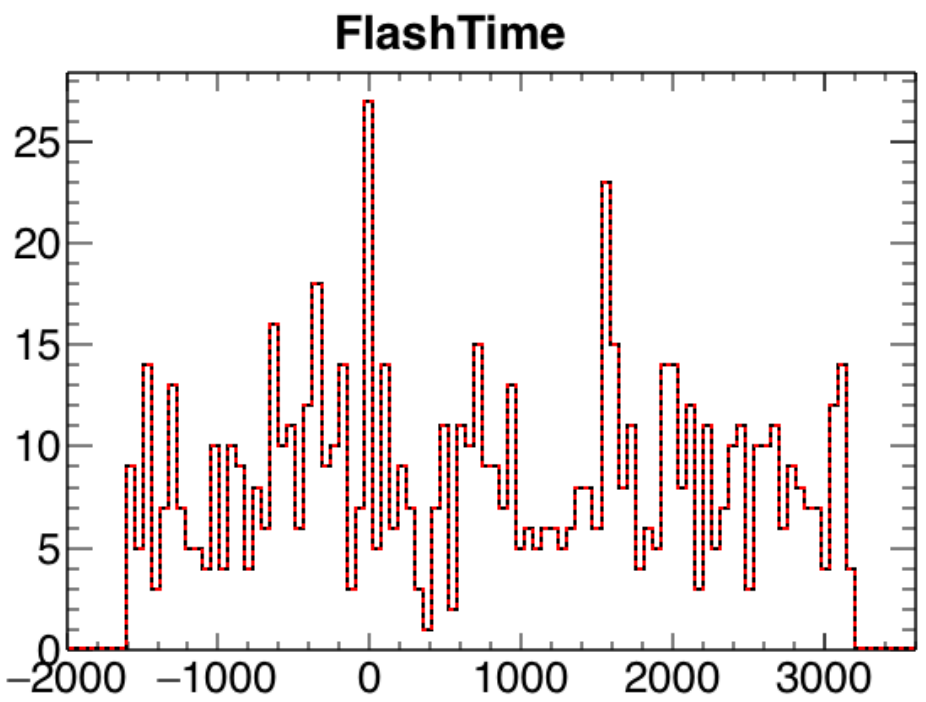
```
double tempfQE = pset.get<double>("QuantumEfficiency");
art::ServiceHandle<util::LArProperties> LarProp;
fQE = tempfQE / LarProp->ScintPreScale();
```

  - uboonecode (in UBOpticalChConfig)

```
// Correct QE by prescaling set in LArProperties
art::ServiceHandle<util::LArProperties>   LarProp;
for (unsigned int i = 0; i < fParams.at(kQE).size(); i++)
  fParams.at(kQE)[i] /= LarProp->ScintPreScale();
```

Validation: OpFlashes



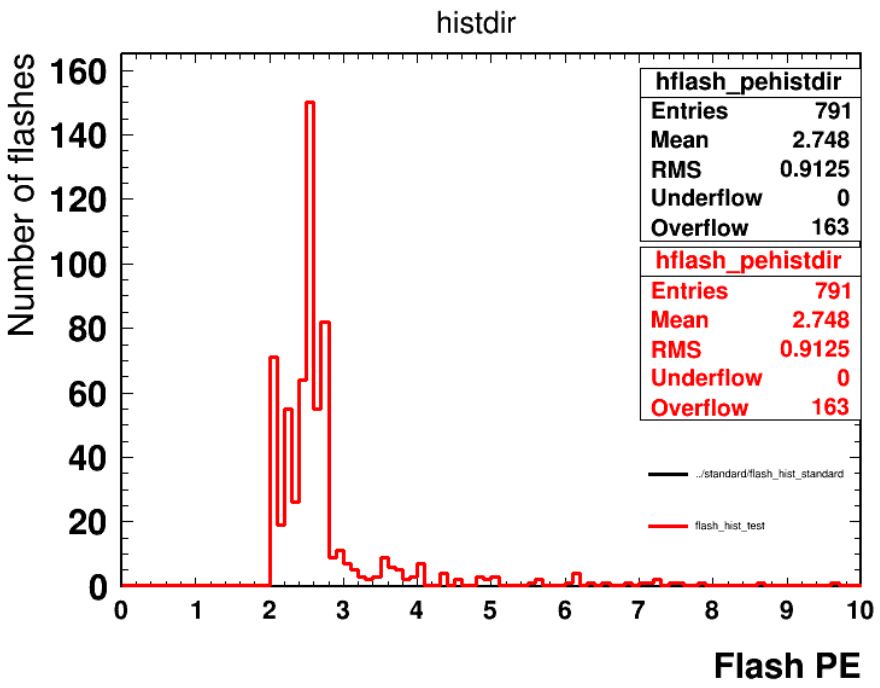


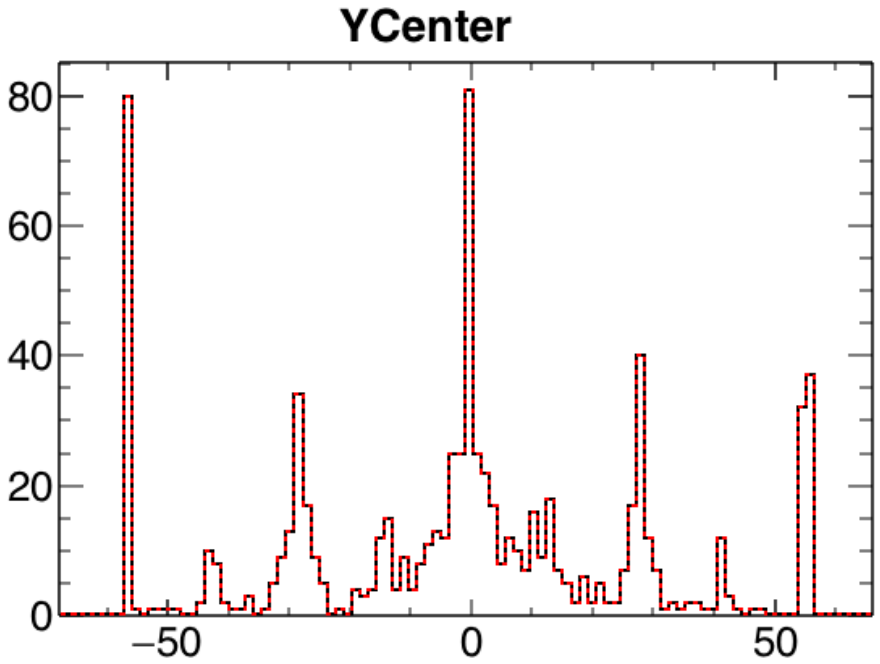
- Exact match
  - G4 simulation (SimPhotonCounter)
    - Used seed service to set matching seeds
  - Reconstruction: OpHits, OpFlashes entry-by-entry
- Also matches up for lbnecode

Conclusions: Prescaling

- Available soon:
  - lardata/ahimmel_scintPreScale
  - larana/ahimmel_scintPreScale
  - larsim/ahimmel_scintPreScale
  - lbnecode/ahimmel_scintPreScale
  - uboonecode/ahimmel_scintPreScale
    - cannot push – need a volunteer
- This way the real QE as well as the real scaling factor are documented in the fhicl configuration during and after running.
- Changes implementation, but recreates exactly method already in use – no changes in results expected

All Together





- I have also tested the results with all three branches merged into develop together.
- Results still identical
  - ubutil validation scripts (left)
  - My validation scripts (right)

Conclusions

- Given that these changes taken together are fairly significant and related, I propose we introduce them together in one version change
  - Breaks some existing code
    - Everything already in the repositories has been fixed
  - *Does not* deprecate existing files
  - *Does not* change results
- Next: turn manual validation tests I have done into real automatic tests
  - Requires some learning on my part.