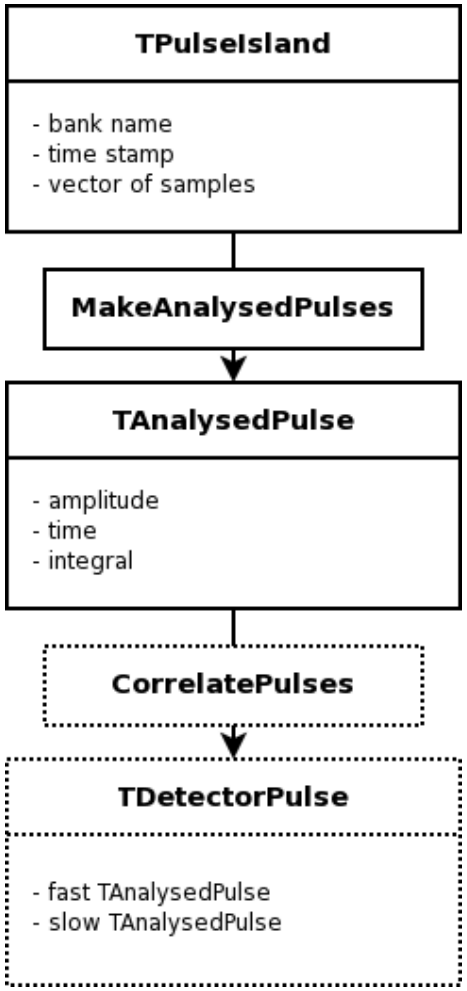
Pulse Finding and Processing

AlCap Collaboration Meeting

Outline

- Overview of Pulse Processing
- Obtaining time, amplitude and integral of the pulse
- Finding pulses
- Discussion Points

Pulse Processing Flow



To be implemented

- Generated by alcapana and stored in the tree file
- Converts TPI --> TAP
- Lightweight class that we will store for future use
- Will pair up fast and slow pulses from the same detector
- Will allow us to take the time from the fast pulse and the amplitude (i.e. energy) from the slow pulse

Converting TPI to TAP

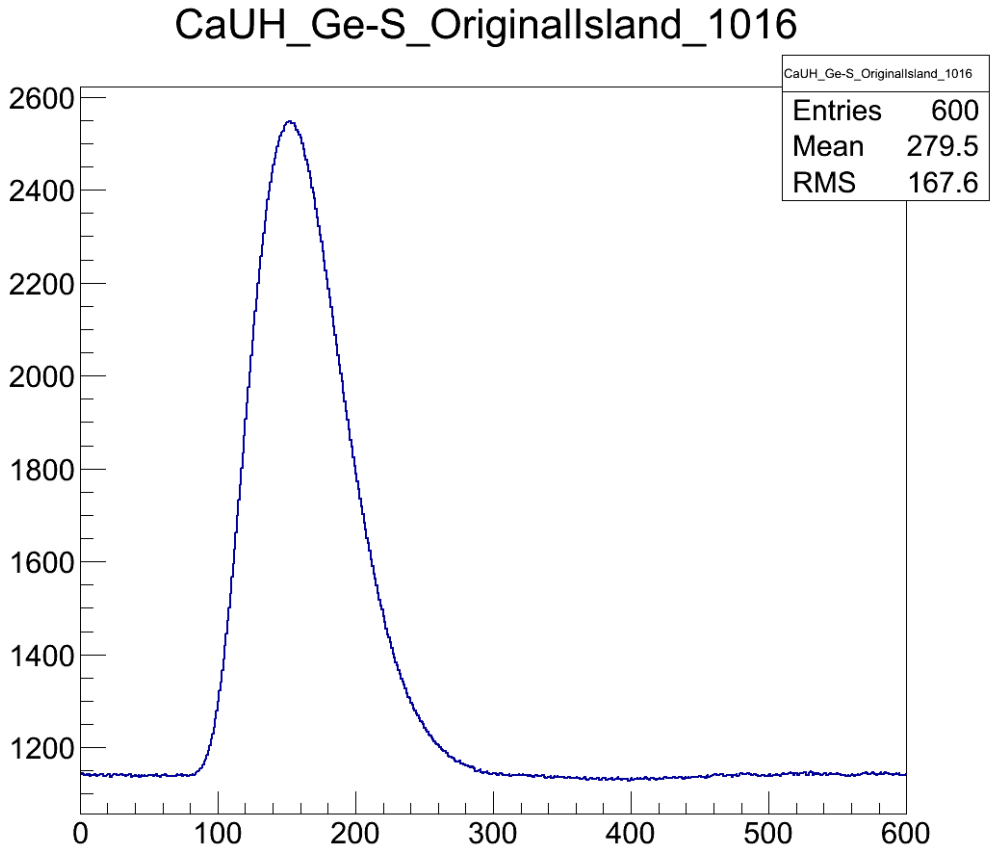
- More details in Ben's talk tomorrow
  - rootana module: MakeAnalysedPulses
  - obtaining time / amplitude / integral is done by classes derived from TVAnalysedPulseGenerators
  - we can develop multiple generators and select which ones we want at run time

TAP Generators

- Current generators:
  - MaxBinAPGenerator: uses the max bin method
- Future generators:
  - using templates
  - using constant fraction for timing

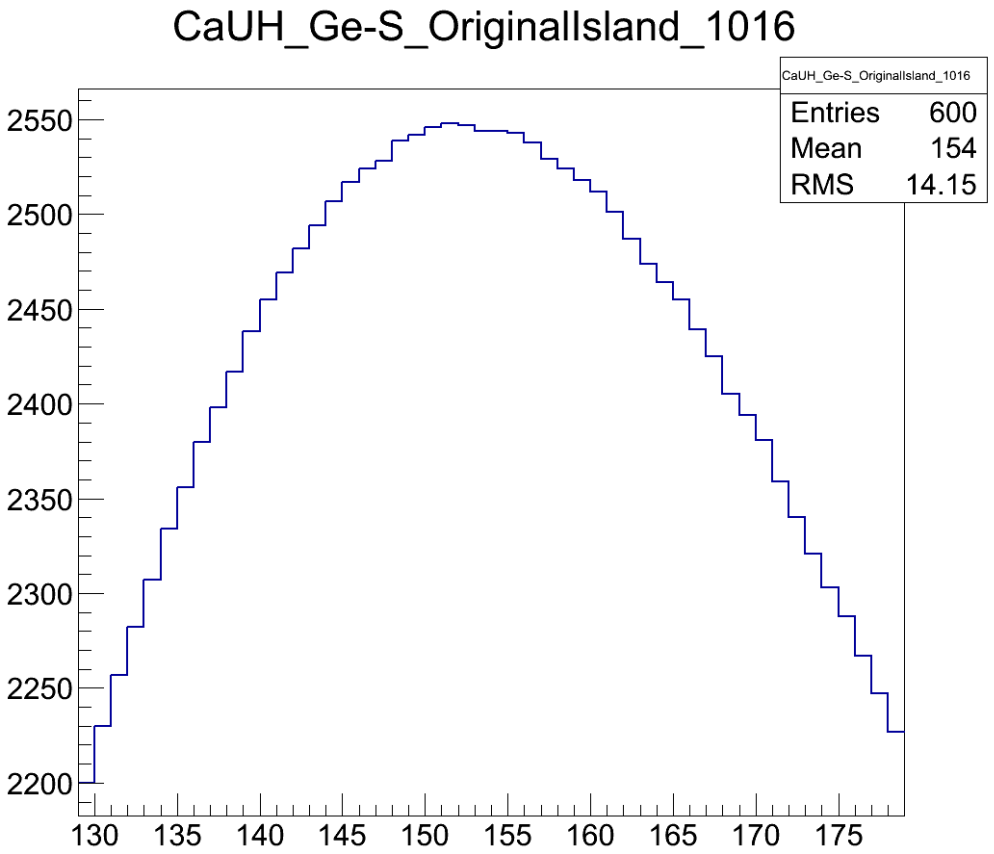
Max Bin Method

- Find the bin with the maximum value:
  - Amplitude: value in that bin
  - Time: the position of that bin
  - No integral



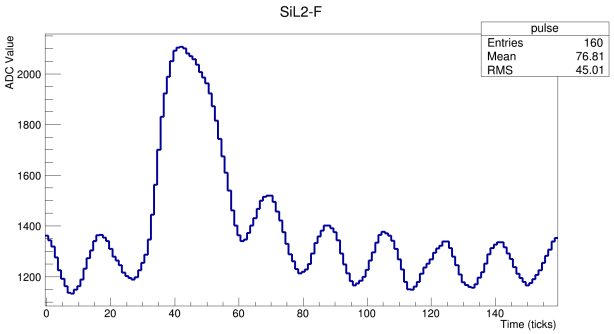
Max Bin Method

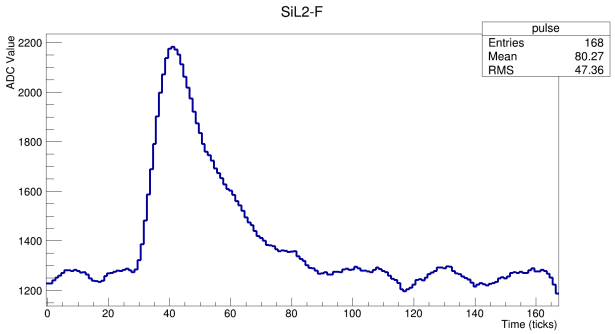
- Not good for broad peaks
  - especially for timing

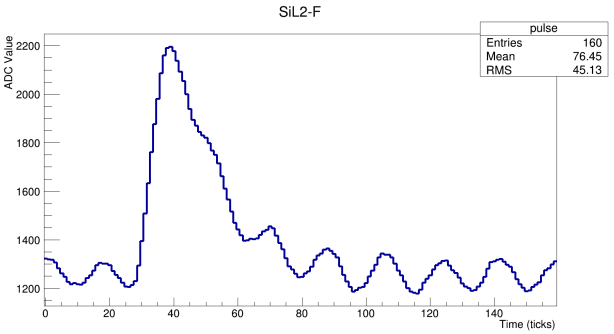


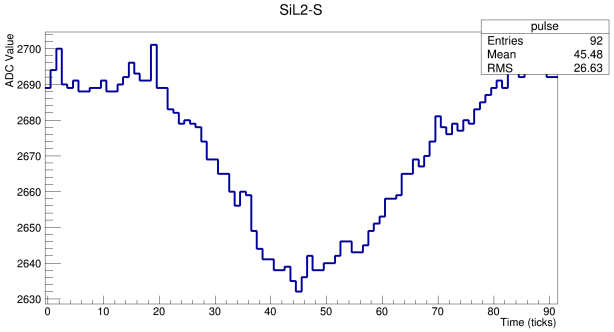
Pulse Templates

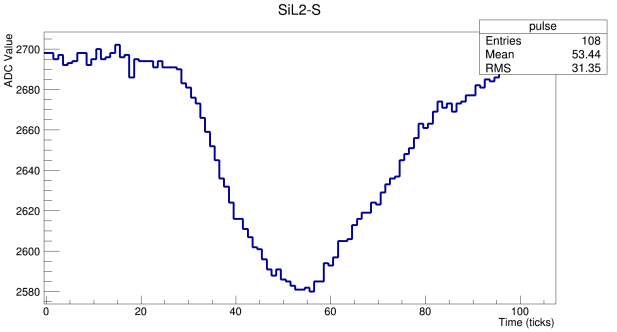
Useful when shape is constant:

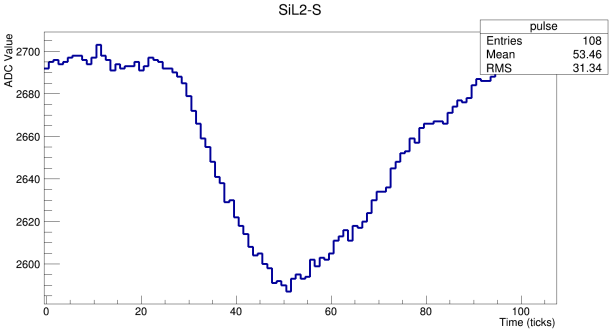


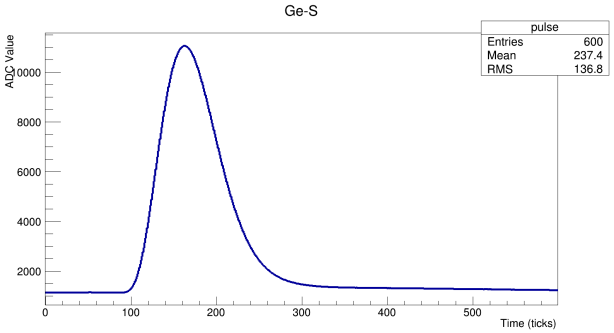


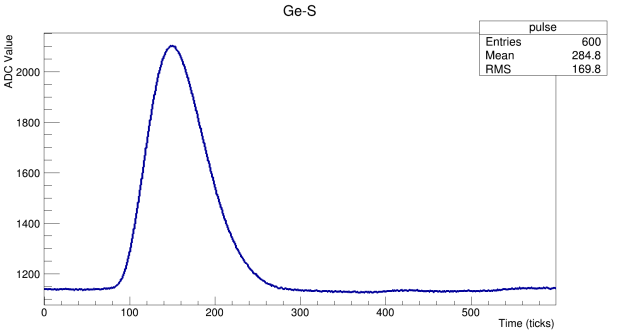






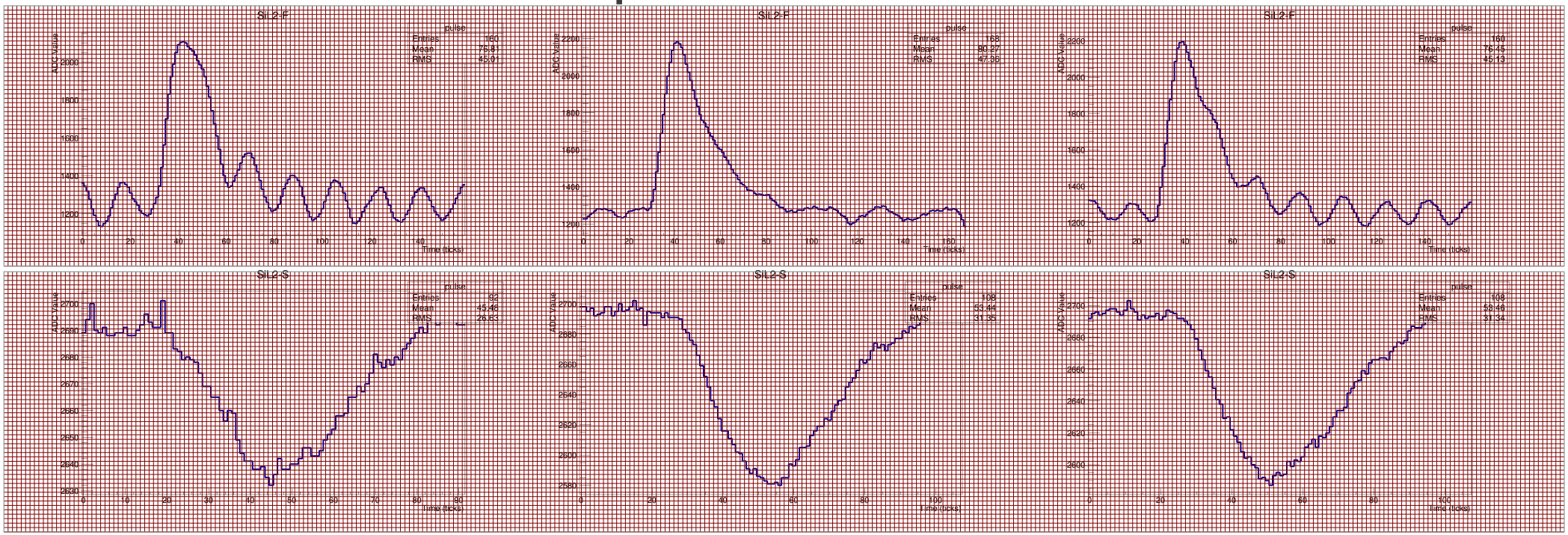


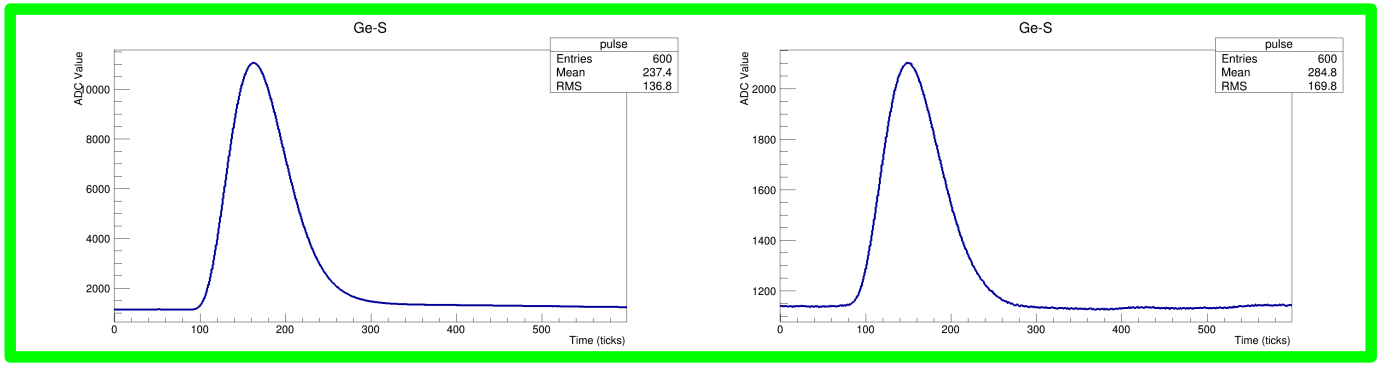




Pulse Templates

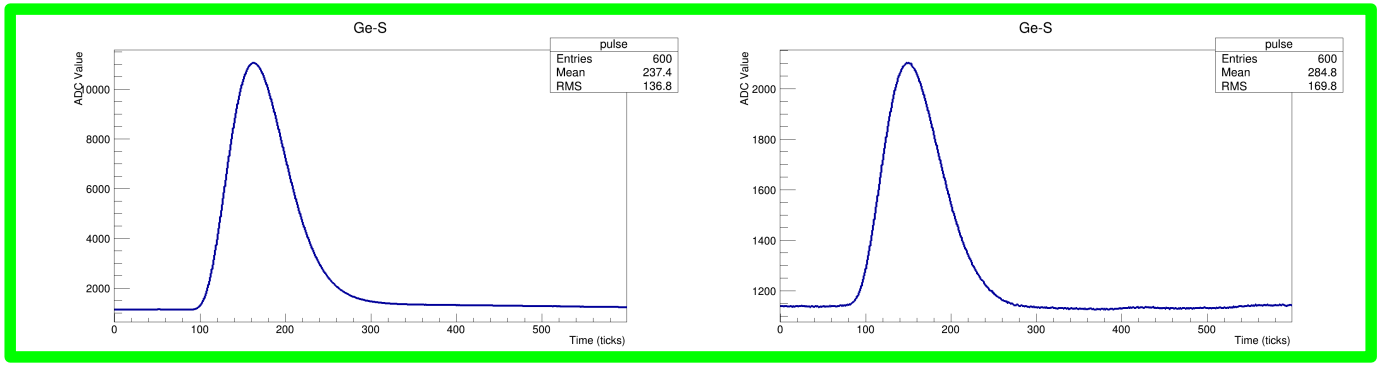
Useful when shape is constant:



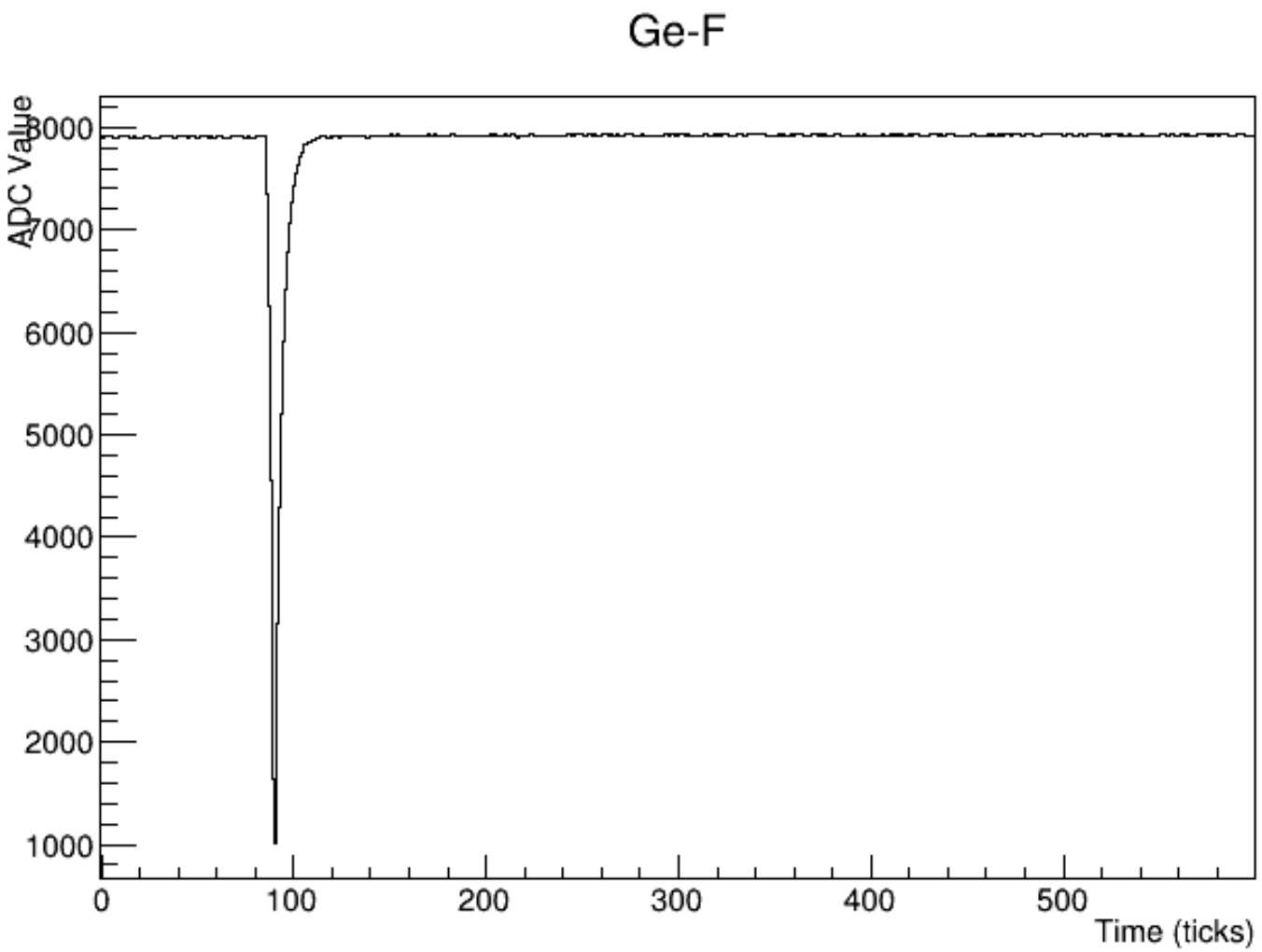


Pulse Templates

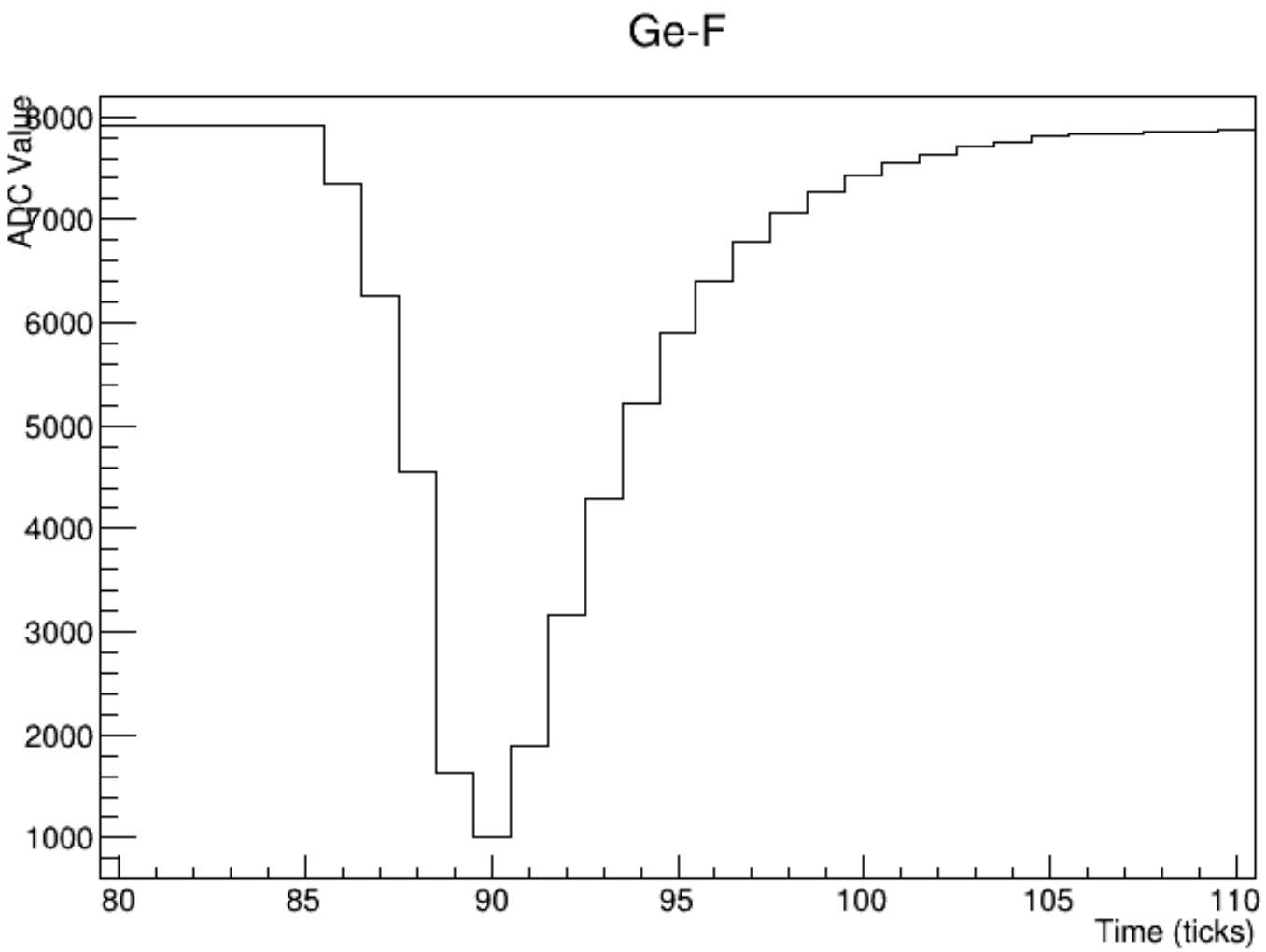
Needs to be implemented still, but on back burner



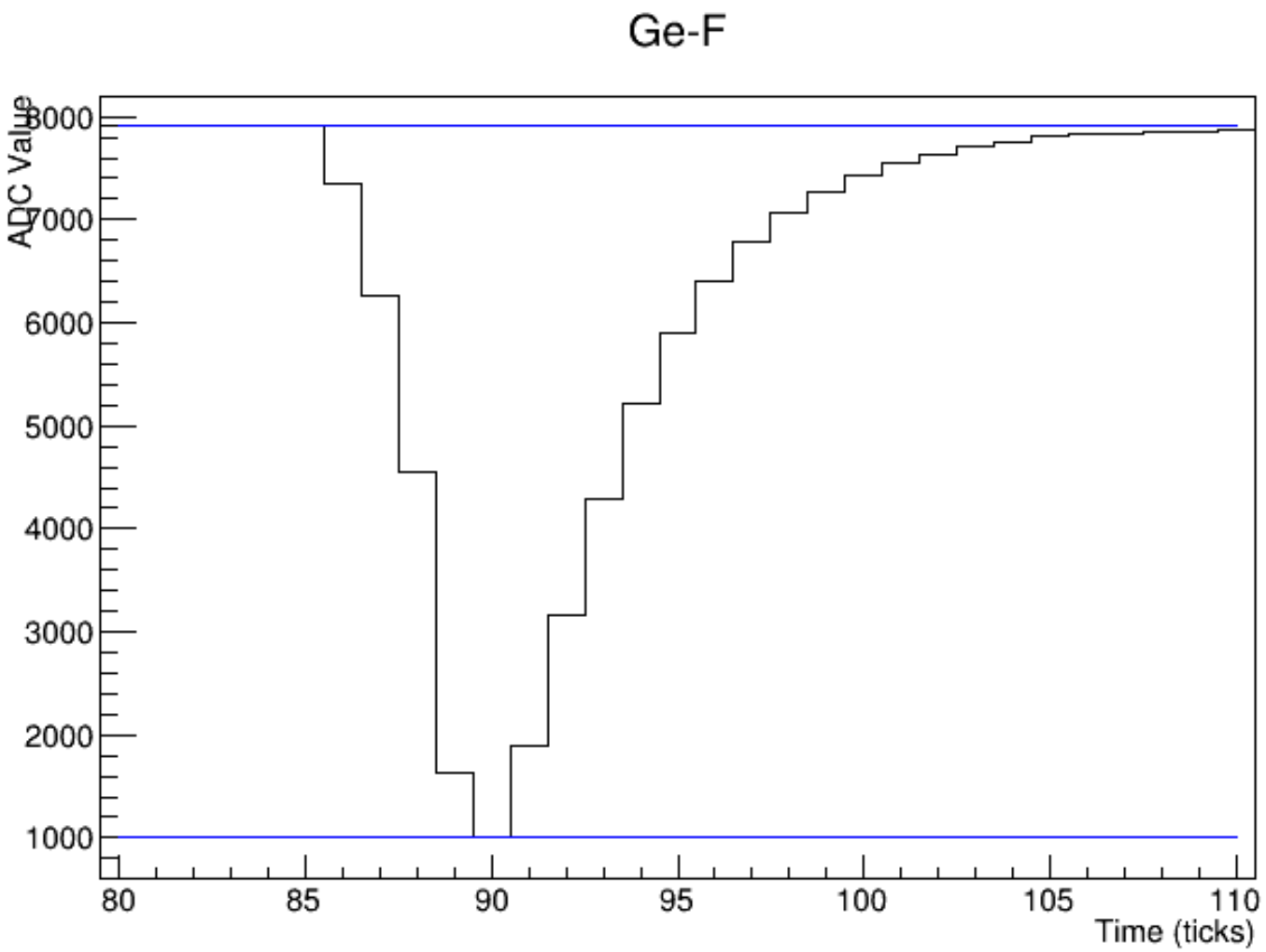
Constant Fraction: 50%



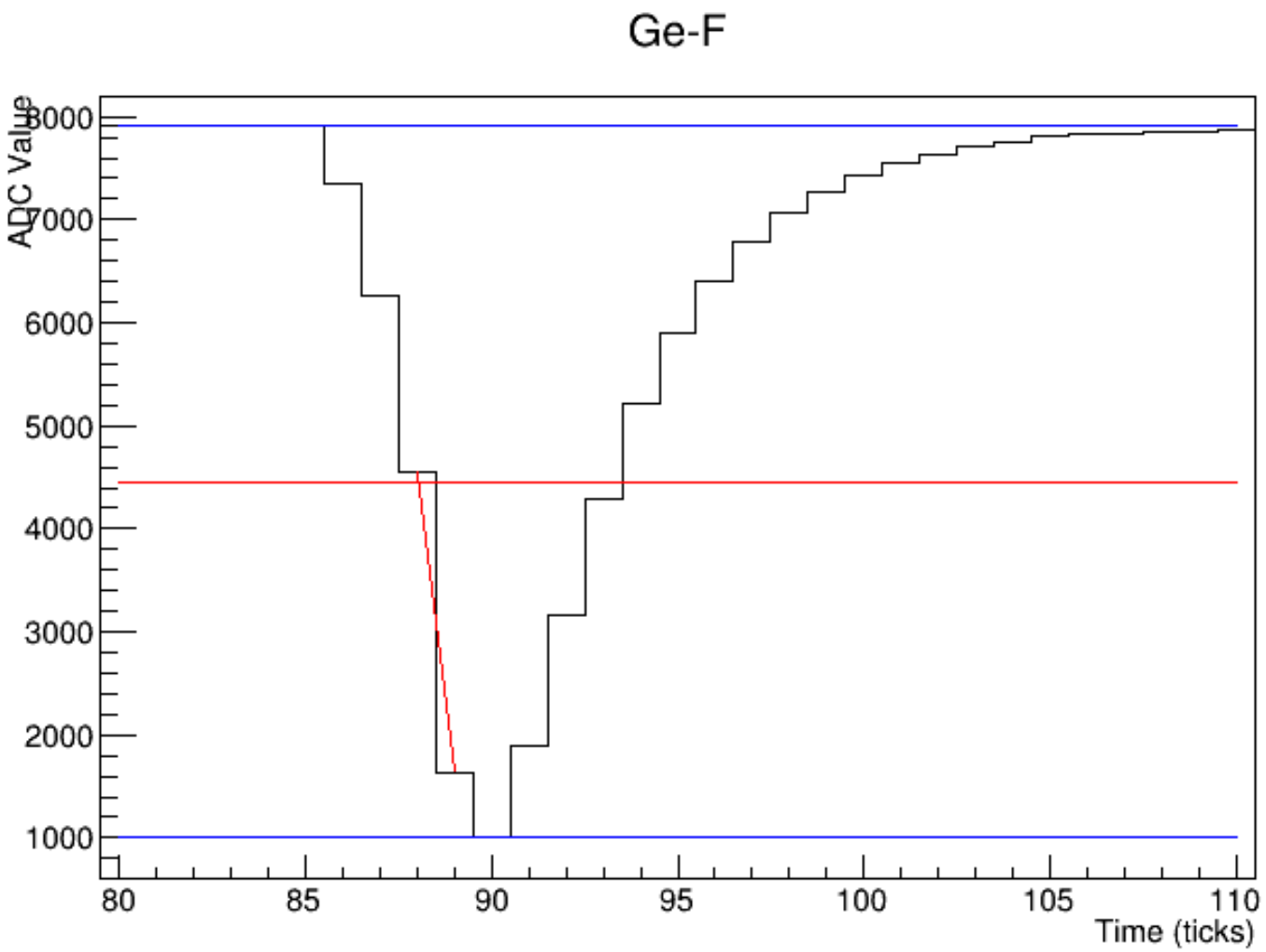
Constant Fraction: 50%



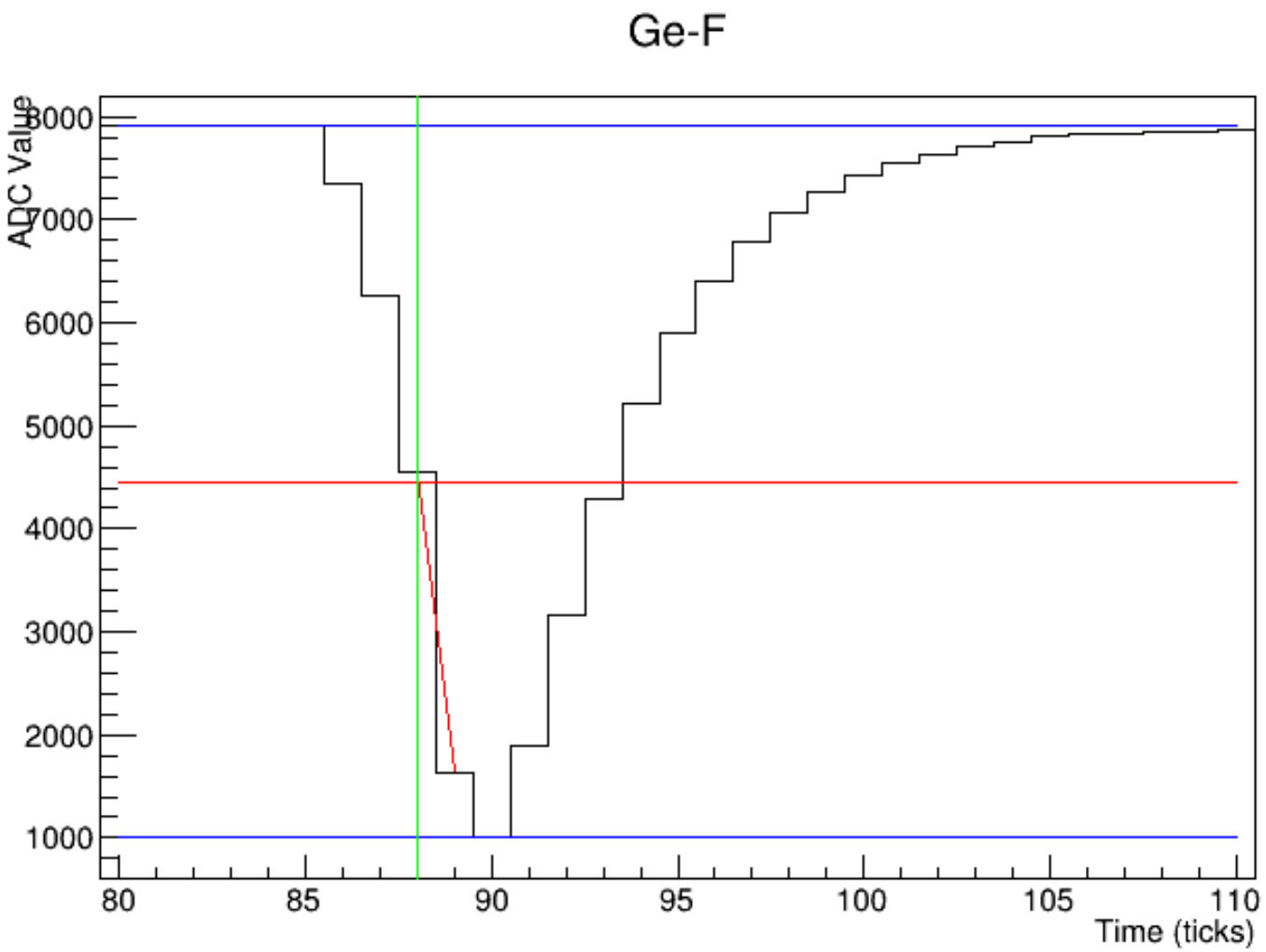
Constant Fraction: 50%



Constant Fraction: 50%



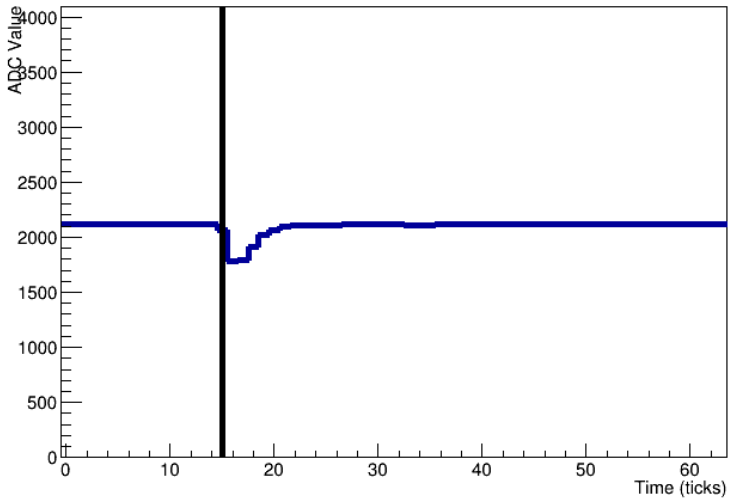
Constant Fraction: 50%



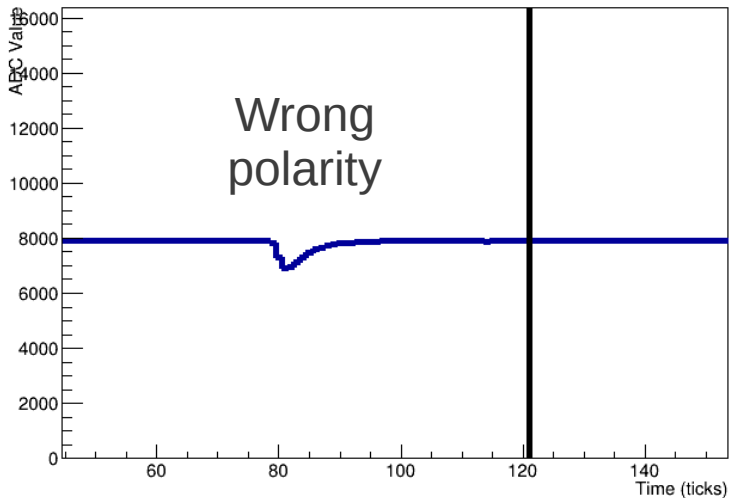
Constant Fraction: 50%

Varying success

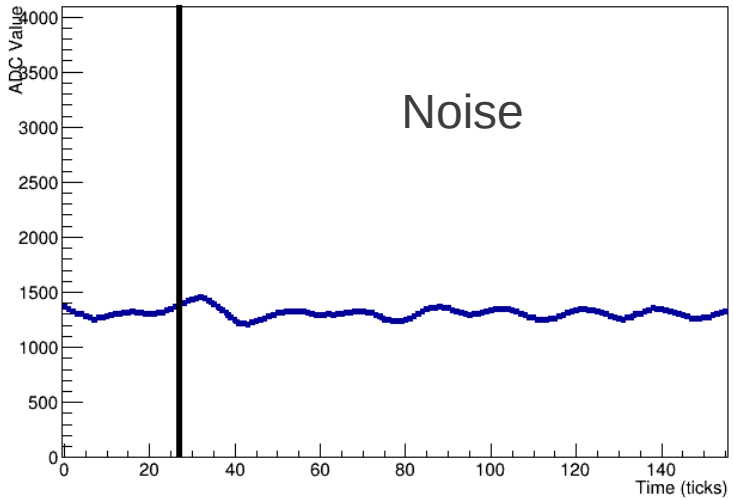
muScA



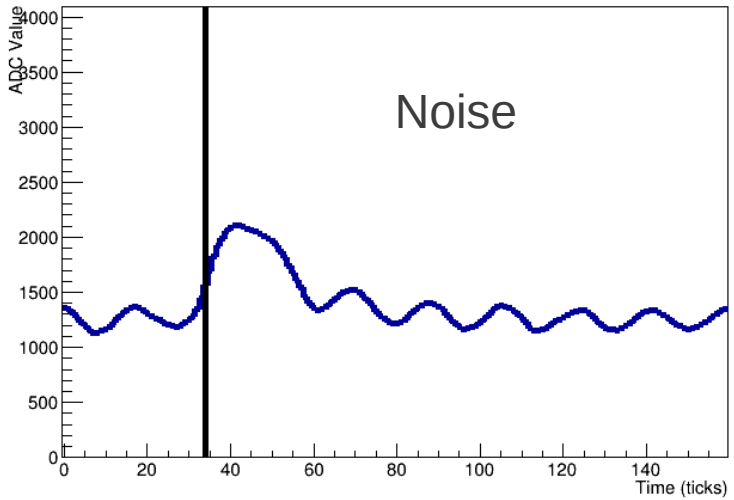
Ge-F



SiR1-1-F



SiL2-F

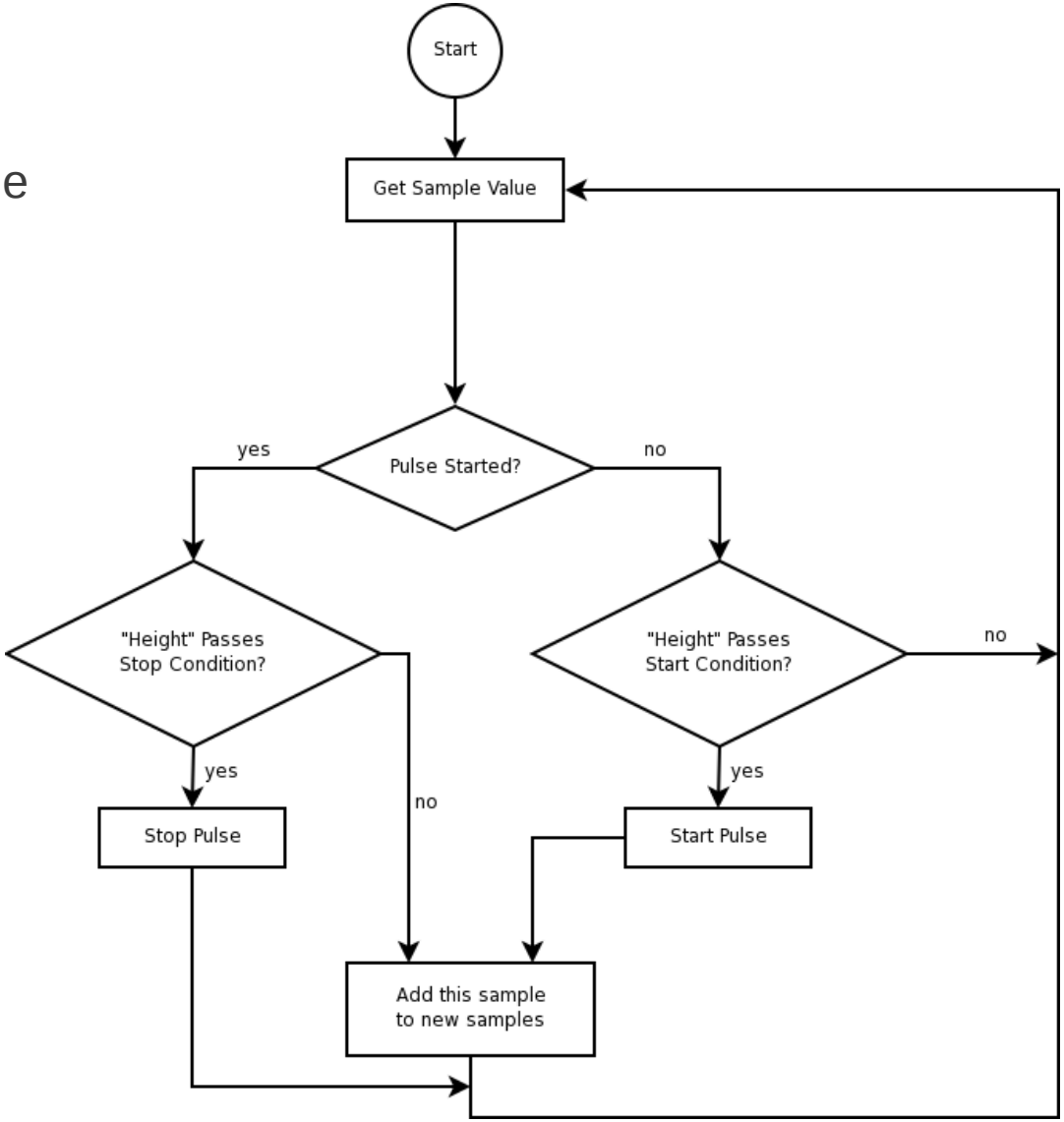


Pulse Finding

- There may be more than one pulse on a pulse island
  - we will want to find them all
- There is a TVPulseFinder class
  - analogous to TVAnalysedPulseGenerator
  - so, again, we can develop different pulse finders and select the one we want at run time
- Current pulse finders:
  - FirstPulseFinder: sees if the height goes above a certain value to start a pulse or below a certain value to stop a pulse
  - NullPulseFinder: just returns the TPI

Pulse Finding

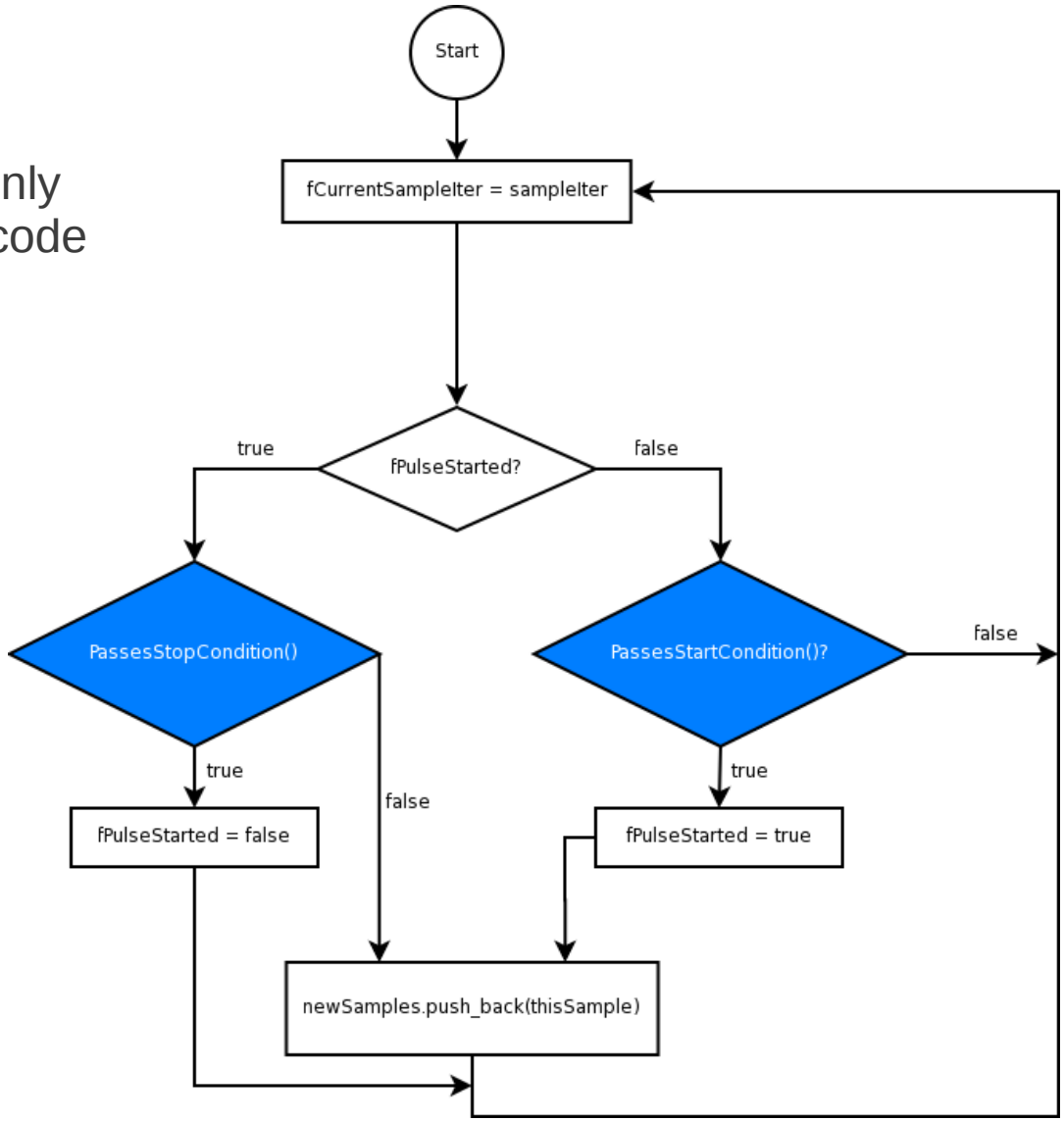
- We loop through the samples



General Overview

Pulse Finding

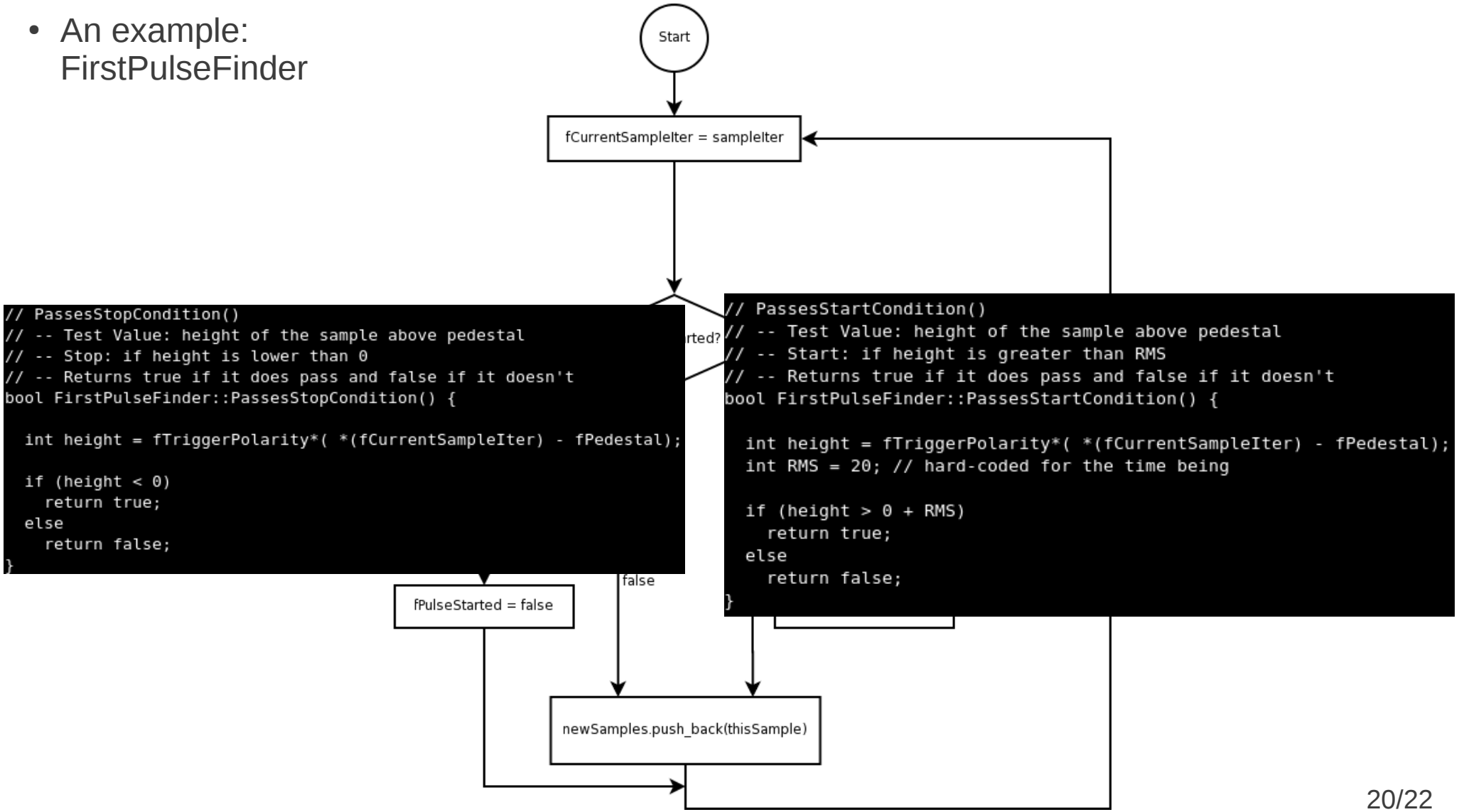
- Inherited classes only need to make the code for the highlighted functions



General Overview (with code snippets)

Pulse Finding

- An example: FirstPulseFinder



Open Questions

- How are we going to pass TAnalysedPulses between modules?
  - currently it is a global pointer to a map
- How do we quantify how “good” a pulse finder is?

Work To Do

- Pulse Processing
  - Create TDetectorPulse
  - Correlate fast and slow pulses
- TAP Generators
  - Get time / amplitude / integral from templates or with CF
- Pulse Finding
  - Develop a better pulse finder