From Simulation to Digis

- **Hits:** Path length, energy deposition
- Total Energy Deposition (MeV); Convert to MIPs
- Split signal between PMTs; Convert to number of pe’s
- Apply Poisson Fluctuations for each PMT
- If auto_calibrate, determine MIPtoFullScale conversion; otherwise, apply Gain factor for each PMT
- Add Gaussian noise for Discriminators; if above thresh, set bits
- Apply supplied or calculated conversion factor
  # MIPs for full scale
- PHA and Discriminators for each PMT
**INPUT**

- Global mean_pe_per_MIP
  - For each PMT: mean_pe_per_MIP with global default 18
- acd_MIPS_fullscale
- auto_calibrate_flag
- Optional: gain factors with global default 1.0
  - std_dev PHA noise
  - std_dev Veto Thresh noise
  - std_dev CNO Thresh noise

**OUTPUT:**

- Total Energy -> MIPs -> number of pe’s
- Split between PMTs and apply Poisson Fluctuations
- If auto_calibrate proceed to MIP conversion, otherwise apply gain factors
- Convert to MIPs
- Apply noise for PHA and discriminators
- Check if discriminator thresholds have been reached, if so set bits
- Obtain PHA value using supplied or calculated conversion factor
- Check for zero suppression

**PHA and discriminators for each PMT**

i.e. 2 PHA and 2 sets of discriminators for each ACD Tile
Conversion to photoelectrons

- Convert total energy deposited (MeV) to MIPs
- Split the signal evenly between the 2 PMTs
- Define a new input parameter for each PMT, the mean number of photoelectrons per MIP: $acd\_mean\_pe\_per\_MIP$
  The PMT will be identified by its identifier in the input file.
- The global default is 18 pe’s
Poisson Fluctuations

• For each PMT, we now have the number of true pe’s (photons) detected
• Calculate fluctuation using CLHEP::RandPoisson, with mean = true pe’s detected
• The result of RandPoisson is the actual number of pe’s seen in the PMT.
• Repeat this process for all PMTs
Auto_calibrate and Gains

• User has a choice between auto calibration and applying gain factors for individual PMTs.

• If using auto calibration – no gain is applied the conversion factor to convert from MIPs to PHA is determined at run time.
  – \( acd\_MIPS\_fullscale \) scaled by the ratio of global pe/MIP to pe/MIP for this PMT.

• Otherwise, gain factors are applied, where global default gain = 1.0
  PMT specific gains are accepted as input, using PMT ids
Gaussian Noise

• Convert from pe’s to MIPs

• New input parameters defined:
  \( \text{acd} \_\text{noise}\_\text{std}\_\text{dev}\_\text{pha}, \text{acd}\_\text{noise}\_\text{std}\_\text{dev}\_\text{veto}, \text{acd}\_\text{noise}\_\text{std}\_\text{dev}\_\text{cno} \)

• Calculate noise using CLHEP::RandGauss
  with mean=0 and stdDev = \( \text{acd}\_\text{noise}\_\text{std}\_\text{dev}\_\text{*} \)
  Separate noise calculation for PHA, Veto and CNO discriminators.

• Add the noise to the number of actual MIPs,
  resulting in observed MIPs
Conversion to PHA

• Determine if MIP with added noise is above or below veto and CNO thresholds – set bits as necessary.

• Apply supplied or calculated conversion factor to the number of MIPs to determine the PHA value.

• Output:
  PHA and discriminators for each PMT