# Subsystem List | Thoughts

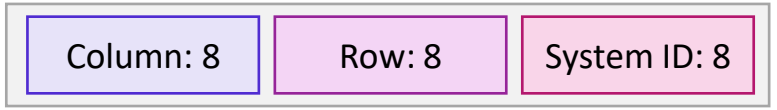
- **Following up on 03.09 WG Meeting:**
  - Decided to provide a utility to extract list of subsystems from reco particles (cf. #119)
  - Started working on a snippet, planning to upstream to EDM4eic utilities once tested
  - ☞ Wanted to bounce some ideas off everyone…

- **General strategy:**
  1) Loop through clusters, tracks (and PID?)
  2) Decompose into reco hits
  3) Extract system ID from cell ID, insert in a std::set
  4) Return set of system IDs

- **Two questions:**
  a) How to handle PID? LUTs store system ID, but tempted to defer since datamodel chain TBD…
  **b) How to extract Cell ID? Two options (see right)…**

**Sys ID Extraction (Opt. A):** use BitFieldCoder
- Easy to use, very transparent
- ~~But need to swap out underlying readout for each system:~~
  ~~a) Would need to maintain list of readouts somewhere~~
  ~~b) And would likely need a loop over readouts in utility~~

**Sys ID Extraction (Opt. B):** use bit operations
- We (should) always allocate 8 bits for system ID in CellIDs
- So can just extract system ID w/ bitwise algebra
  - ☞ But is the system ID always in the same position (eg. below)?



# Subsystem List | Intended API

**Intended Usage** (almost pseudocode)

```
auto detector = dd4hep::Detector::make_unique("");
detector->fromCompact("my_compact.xml");

podio::ROOTReader reader = podio::ROOTReader();
reader.openFile("my_input.edm4eic.root");

for (ievt = 0; ievt < reader.getEntries(podio::Category::Event); ++ievt) {
  auto frame = podio::Frame(reader.readNextEntry(podio::Category::Event));
  const auto& pars = frame.get<edm4eic::ReconstructedParticle>("ReconstructedParticles");
  for (const auto& par : pars) {
    std::set<uint32_t> systems = edm4eic::get_subsystem_list(par, detector); // <<<< using the tool
    if (systems.contains(103) {
      //... if par has EEEMCal info, do stuff ...//
    }
  }
}
```

- **Above:** some (almost) pseudocode to illustrate how tool might be used
  - Using a PODIO-based event loop as an example, but also could work in an RDataFrame
  - **Note:** might be missing some std::move's, etc

☒ **Limitations:** would NOT work in these contexts
  - TTreeReader-based analyses
  - Python (unless DD4hep has a python interface…)