Hall A Analyzer Tutorial

Ole Hansen

Jefferson Lab

April 11, 2016

http://hallaweb.jlab.org/podd/

Brief Introduction

The Hall A C++ Analyzer (“Podd”)

- Class library on top of ROOT
- Analysis controlled via interpreted or compiled C++ scripts (using ROOT’s CINT interpreter)
- Toolbox of analysis modules
- Special emphasis on modularity
  - “Everything is a plug-in”
  - External user libraries dynamically loadable at run time
  - User code separate from core code
  - No need for users to write more than the code really needed (hopefully)
  - Core analyzer suitable for fixed installation, like ROOT itself
- Predefined modules for generic analysis tasks and standard Hall A equipment
- SDK available for rapid development of new module libraries

Status

- Latest stable version: 1.5.31 (8 Apr 2016) ▸ web
  - Minor bugfixes
  - ROOT 6 compatibility
  - 1.5.x releases are binary compatible
- Preview release version 1.6-beta1 ▸ web
  - Many new features (see next slides), not all fully implemented yet
  - Hope to finalize this spring for fall run
  - Preliminary Release Notes ▸ web
- Development version 1.6.0-devel ▸ GitHub
  - For experts (code may change at any time in incompatible ways)
  - To download, clone the repository with

    ```
    git clone https://github.com/JeffersonLab/analyzer.git
    ```

  - Git "cheat sheet" available here ▸ web

What's New in 1.6: Completed Items

- EVIO from external library ✓
  - Support for latest version (currently 4.4.6). Easy to update
  - If not installed, the `analyzer` build system will automatically download & install EVIO
- Modular decoder (Bob Michaels) ✓
  - Easy to add support for new front-end electronics via plug-in modules (drivers)
  - Processing of different event or trigger types configurable via "event type handler" plug-ins
  - Preliminary support for 12 GeV pipelined electronics modules (FADC250, F1TDC, etc.)
  - Detailed documentation: web
- Miscellaneous ✓
  - Improved formula & test package (removed limitations)
  - Simulation event data decoder API and example implementation
  - `scons` build system
  - Rewritten, modular hardware channel decoder (`THaDecData`)
  - Many small code improvements

What's New in 1.6: Work In Progress

- Universal database interface ✎
    - All analysis modules now use this interface
    - Users must convert their existing databases. Conversion utility program available (90% finished)
    - May eventually support multiple backends (text files, SQL database, CCDB, etc.) Currently, backend for Hall A-style text files available.
- Improved VDC track reconstruction ✎
    - Known bugs fixed
    - Reconstruction of multi-cluster events should be greatly improved (to be demonstrated)
    - Still needs careful testing. Testers welcome
    - Old code will remain available as an alternative tracking algorithm

Resources

- Web site (not a Wiki) ▸ home page
- Mailing list (new): `halla_software@jlab.org`
  Subscribe on ▸ mailman
- Bi-weekly Hall A/C software meeting: Tuesdays, 11am, L210A
- Bug tracker ▸ GitHub
- Analysis Workshop archive ▸ archive (includes tutorials)

Next Analysis Workshop: Summer 2016 (in preparation for fall run)

Using the Analyzer

Analyzer Concepts: Analysis Objects

- Any module that produces “results”
- Every analysis object has unique name, *e.g.* R.s1
- Results stored in **“global variables”**, prefixed with the respective module’s name, *e.g.* R.s1.nhits
- `THaAnalysisObject` common base class:
  - Support functions for database access
  - Support functions for global variable handling
- Actual objects implement various virtual functions
  - `DefineVariables()`
  - `ReadDatabase()`
  - `Decode()`
  - etc.

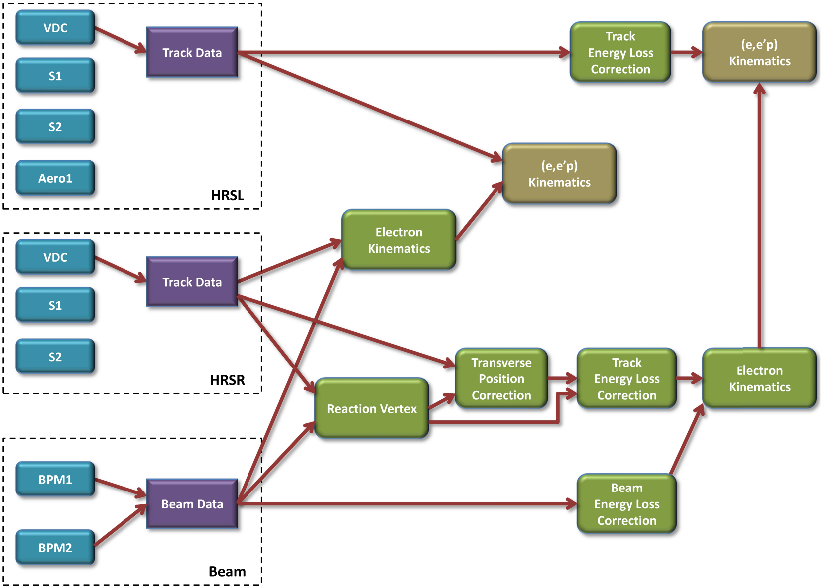
Types of Analysis Objects

- "Detector"
  - Code/data for analyzing a type of detector.
    Examples: Scintillator, Cherenkov, VDC, BPM
  - Typically embedded in an Apparatus
- "Apparatus" / "Spectrometer"
  - Collection of Detectors
  - Combines data from detectors
  - "Spectrometer": Apparatus with support for tracks
- "Physics Module"
  - Combines data from several apparatuses
  - Typical applications: kinematics calculations, vertex finding, coincidence time extraction
  - Toolbox design: Modules can be chained, combined, used as needed

Frequently Used Modules

- `THaHRS` (Spectrometer)
  - HRS spectrometer (includes VDC tracking detector)
  - Need to add other detector (scintillators, Cherenkovs, etc.) as needed
- `THaIdealBeam` (Apparatus)
  - Description of the incident beam
  - Constant position and direction (configurable)
  - Needed to compute vertex (reaction point), kinematics
  - More advanced: `THaUnRasteredBeam`, `THaRasteredBeam`
- `THaElectronKine` (PhysicsModule)
  - Single-arm electron kinematics (e,e′)
  - "e" from beam apparatus, "e′" from one spectrometer
- `THaGoldenTrack` (Physics Module)
  - Picks out the "golden track" from a spectrometer's track array
  - Definition depends on spectrometer
  - `THaHRS` has two definitions, selectable with SetTrSorting()

A (complex) Module Configuration Example



Tutorial

Getting The Software

1. Set up ROOT
2. Download the analyzer source code
3. Unpack and build
4. To install, simply set environment variables

Setting Up The Software on JLab CUE

```
ifarm1102> source /apps/root/PRO/setroot_CUE
ifarm1102> curl -O http://hallaweb.jlab.org/podd/download/analyzer-1.6.0-beta1.tar.xz
ifarm1102> tar xf analyzer-1.6.0-beta1.tar.xz
ifarm1102> cd analyzer-1.6.0
ifarm1102> make -j4
ifarm1102> setenv PATH ${PWD}:${PATH}
ifarm1102> setenv LD_LIBRARY_PATH ${PWD}:${LD_LIBRARY_PATH}
ifarm1102> analyzer
analyzer [0]
```

More details on the web ▸ docs

Things You'll Need

1. Replay script
   - Defines detectors/apparatuses to be analyzed, kinematics, calculations to be done, file locations, tree variable names etc.
   - Many examples available from previous experiments
   - Simple or fancy ▸ fancy example. Try to start out simple
   - May be compiled
2. Set of database files
   - Usually one file per detector, `db_<name>.dat`
   - Run database, `db_run.dat`, defines beam energy, spectrometer angles
   - `db_cratemap.dat` and `scaler.map`, define decoder parameters
     $\rightarrow$ get these files from DAQ expert
3. Output definition file
   - Defines which variables to write to the tree in the output ROOT file
4. Raw data (CODA file)

Example Replay Script

```
// Set up left arm HRS with the detectors we're interested in
THaHRS* HRSL = new THaHRS("L", "Left HRS");
HRSL->AddDetector( new THaVDC("vdc", "Vertical Drift Chambers" ));
HRSL->AddDetector( new THaScintillator("s1", "S1 scintillator" ));
HRSL->AddDetector( new THaCherenkov("cer", "Gas Cherenkov counter" ));
gHaApps->Add(HRSL);

// Ideal beam (perfect normal incidence and centering)
THaIdealBeam* ib = new THaIdealBeam("IB", "Ideal beam");
gHaApps->Add(ib);

// Simple kinematics and vertex calculations
Double_t mass_tg = 12*931.494e-3; // C12 target
THaPhysicsModule *ekine, *rpl, *Lgold;
ekine = new THaElectronKine( "L.ekine", "Electron kinematics L", "L", "IB", mass_tg );
rpl   = new THaReactionPoint( "rpl", "Reaction vertex L", "L", "IB" );
Lgold = new THaGoldenTrack( "L.gold", "LHRS golden track", "L" );
gHaPhysics->Add(ekine);
gHaPhysics->Add(rpl);
gHaPhysics->Add(Lgold);

// The CODA data file we want to replay
THaRun* run = new THaRun("/rawdata/run_12345.dat");

// Set up and run standard analyzer (event loop)
THaAnalyzer* analyzer = new THaAnalyzer;
analyzer->SetOutFile( "/bigdisk/run_12345.root" );
analyzer->SetOdefFile("HRSL.odef");    // Define output
analyzer->Process(run);                // Process all invents in the input
```

Example Replay Script

```
// Set up left arm HRS with the detectors we're interested in
THaHRS* HRSL = new THaHRS("L", "Left HRS");
HRSL->AddDetector( new THaVDC("vdc", "Vertical Drift Chambers" ));
HRSL->AddDetector( new THaScintillator("s1", "S1 scintillator" ));
HRSL->AddDetector( new THaCherenkov("cer", "Gas Cherenkov counter" ));
gHaApps->Add(HRSL);

// Ideal beam (perfect normal incidence and centering)
THaIdealBeam* ib = new THaIdealBeam("IB", "Ideal beam");
gHaApps->Add(ib);

// Simple kinematics and vertex calculations
Double_t mass_tg = 12*931.494e-3; // C12 target
THaPhysicsModule *ekine, *rpl, *Lgold;
ekine = new THaElectronKine( "L.ekine", "Electron kinematics L", "L", "IB", mass_tg );
rpl   = new THaReactionPoint( "rpl", "Reaction vertex L", "L", "IB" );
Lgold = new THaGoldenTrack( "L.gold", "LHRS golden track", "L" );
gHaPhysics->Add(ekine);
gHaPhysics->Add(rpl);
gHaPhysics->Add(Lgold);

// The CODA data file we want to replay
THaRun* run = new THaRun("/rawdata/run_12345.dat");

// Set up and run standard analyzer (event loop)
THaAnalyzer* analyzer = new THaAnalyzer;
analyzer->SetOutFile( "/bigdisk/run_12345.root" );
analyzer->SetOdefFile("HRSL.odef");    // Define output
analyzer->Process(run);                // Process all invents in the input
```

Note: Modules are set up by including them in analysis lists

Example Replay Script

```
// Set up left arm HRS with the detectors we're interested in
THaHRS* HRSL = new THaHRS("L", "Left HRS");
HRSL->AddDetector( new THaVDC("vdc", "Vertical Drift Chambers" ));
HRSL->AddDetector( new THaScintillator("s1", "S1 scintillator" ));
HRSL->AddDetector( new THaCherenkov("cer", "Gas Cherenkov counter" ));
gHaApps->Add(HRSL);

// Ideal beam (perfect normal incidence and centering)
THaIdealBeam* ib = new THaIdealBeam("IB", "Ideal beam");
gHaApps->Add(ib);

// Simple kinematics and vertex calculations
Double_t mass_tg = 12*931.494e-3; // C12 target
THaPhysicsModule *ekine, *rpl, *Lgold;
ekine = new THaElectronKine( "L.ekine", "Electron kinematics L", "L", "IB", mass_tg );
rpl   = new THaReactionPoint( "rpl", "Reaction vertex L", "L", "IB" );
Lgold = new THaGoldenTrack( "L.gold", "LHRS golden track", "L" );
gHaPhysics->Add(ekine);
gHaPhysics->Add(rpl);
gHaPhysics->Add(Lgold);

// The CODA data file we want to replay
THaRun* run = new THaRun("/rawdata/run_12345.dat");

// Set up and run standard analyzer (event loop)
THaAnalyzer* analyzer = new THaAnalyzer;
analyzer->SetOutFile( "/bigdisk/run_12345.root" );
analyzer->SetOdefFile("HRSL.odef");    // Define output
analyzer->Process(run);                // Process all invents in the input
```

Note: module names → prefix for database keys & global variables

Example Replay Script

```
// Set up left arm HRS with the detectors we're interested in
THaHRS* HRSL = new THaHRS("L", "Left HRS");
HRSL->AddDetector( new THaVDC("vdc", "Vertical Drift Chambers" ));
HRSL->AddDetector( new THaScintillator("s1", "S1 scintillator" ));
HRSL->AddDetector( new THaCherenkov("cer", "Gas Cherenkov counter" ));
gHaApps->Add(HRSL);

// Ideal beam (perfect normal incidence and centering)
THaIdealBeam* ib = new THaIdealBeam("IB", "Ideal beam");
gHaApps->Add(ib);

// Simple kinematics and vertex calculations
Double_t mass_tg = 12*931.494e-3; // C12 target
THaPhysicsModule *ekine, *rpl, *Lgold;
ekine = new THaElectronKine( "L.ekine", "Electron kinematics L", "L", "IB", mass_tg );
rpl   = new THaReactionPoint( "rpl", "Reaction vertex L", "L", "IB" );
Lgold = new THaGoldenTrack( "L.gold", "LHRS golden track", "L" );
gHaPhysics->Add(ekine);
gHaPhysics->Add(rpl);
gHaPhysics->Add(Lgold);

// The CODA data file we want to replay
THaRun* run = new THaRun("/rawdata/run_12345.dat");

// Set up and run standard analyzer (event loop)
THaAnalyzer* analyzer = new THaAnalyzer;
analyzer->SetOutFile( "/bigdisk/run_12345.root" );
analyzer->SetOdefFile("HRSL.odef");    // Define output
analyzer->Process(run);                // Process all invents in the input
```

Note: In 1.6, HRS no longer contains VDC/S1/S2 detectors by default!

Example Replay Script

```
// Set up left arm HRS with the detectors we're interested in
THaHRS* HRSL = new THaHRS("L", "Left HRS");
HRSL->AddDetector( new THaVDC("vdc", "Vertical Drift Chambers" ));
HRSL->AddDetector( new THaScintillator("s1", "S1 scintillator" ));
HRSL->AddDetector( new THaCherenkov("cer", "Gas Cherenkov counter" ));
gHaApps->Add(HRSL);

// Ideal beam (perfect normal incidence and centering)
THaIdealBeam* ib = new THaIdealBeam("IB", "Ideal beam");
gHaApps->Add(ib);

// Simple kinematics and vertex calculations
Double_t mass_tg = 12*931.494e-3; // C12 target
THaPhysicsModule *ekine, *rpl, *Lgold;
ekine = new THaElectronKine( "L.ekine", "Electron kinematics L", "L", "IB", mass_tg );
rpl   = new THaReactionPoint( "rpl", "Reaction vertex L", "L", "IB" );
Lgold = new THaGoldenTrack( "L.gold", "LHRS golden track", "L" );
gHaPhysics->Add(ekine);
gHaPhysics->Add(rpl);
gHaPhysics->Add(Lgold);

// The CODA data file we want to replay
THaRun* run = new THaRun("/rawdata/run_12345.dat");

// Set up and run standard analyzer (event loop)
THaAnalyzer* analyzer = new THaAnalyzer;
analyzer->SetOutFile( "/bigdisk/run_12345.root" );
analyzer->SetOdefFile("HRSL.odef");    // Define output
analyzer->Process(run);                // Process all invents in the input
```

Note: Choosing output definitions

Example Replay Script

```
// Set up left arm HRS with the detectors we're interested in
THaHRS* HRSL = new THaHRS("L", "Left HRS");
HRSL->AddDetector( new THaVDC("vdc", "Vertical Drift Chambers" ));
HRSL->AddDetector( new THaScintillator("s1", "S1 scintillator" ));
HRSL->AddDetector( new THaCherenkov("cer", "Gas Cherenkov counter" ));
gHaApps->Add(HRSL);

// Ideal beam (perfect normal incidence and centering)
THaIdealBeam* ib = new THaIdealBeam("IB", "Ideal beam");
gHaApps->Add(ib);

// Simple kinematics and vertex calculations
Double_t mass_tg = 12*931.494e-3; // C12 target
THaPhysicsModule *ekine, *rpl, *Lgold;
ekine = new THaElectronKine( "L.ekine", "Electron kinematics L", "L", "IB", mass_tg );
rpl   = new THaReactionPoint( "rpl", "Reaction vertex L", "L", "IB" );
Lgold = new THaGoldenTrack( "L.gold", "LHRS golden track", "L" );
gHaPhysics->Add(ekine);
gHaPhysics->Add(rpl);
gHaPhysics->Add(Lgold);

// The CODA data file we want to replay
THaRun* run = new THaRun("/rawdata/run_12345.dat");

// Set up and run standard analyzer (event loop)
THaAnalyzer* analyzer = new THaAnalyzer;
analyzer->SetOutFile( "/bigdisk/run_12345.root" );
analyzer->SetOdefFile("HRSL.odef");    // Define output
analyzer->Process(run);                // Process all invents in the input
```

Note: Module chaining

Example Replay Script

```
// Set up left arm HRS with the detectors we're interested in
THaHRS* HRSL = new THaHRS("L", "Left HRS");
HRSL->AddDetector( new THaVDC("vdc", "Vertical Drift Chambers" ));
HRSL->AddDetector( new THaScintillator("s1", "S1 scintillator" ));
HRSL->AddDetector( new THaCherenkov("cer", "Gas Cherenkov counter" ));
gHaApps->Add(HRSL);

// Ideal beam (perfect normal incidence and centering)
THaIdealBeam* ib = new THaIdealBeam("IB", "Ideal beam");
gHaApps->Add(ib);

// Simple kinematics and vertex calculations
Double_t mass_tg = 12*931.494e-3; // C12 target
THaPhysicsModule *ekine, *rpl, *Lgold;
ekine = new THaElectronKine( "L.ekine", "Electron kinematics L", "L", "IB", mass_tg );
rpl   = new THaReactionPoint( "rpl", "Reaction vertex L", "L", "IB" );
Lgold = new THaGoldenTrack( "L.gold", "LHRS golden track", "L" );
gHaPhysics->Add(ekine);
gHaPhysics->Add(rpl);
gHaPhysics->Add(Lgold);

// The CODA data file we want to replay
THaRun* run = new THaRun("/rawdata/run_12345.dat");

// Set up and run standard analyzer (event loop)
THaAnalyzer* analyzer = new THaAnalyzer;
analyzer->SetOutFile( "/bigdisk/run_12345.root" );
analyzer->SetOdefFile("HRSL.odef");    // Define output
analyzer->Process(run);                // Process all invents in the input
```

Note: Setting target mass parameter in script, overrides run database

Example Replay Script

```
// Set up left arm HRS with the detectors we're interested in
THaHRS* HRSL = new THaHRS("L", "Left HRS");
HRSL->AddDetector( new THaVDC("vdc", "Vertical Drift Chambers" ));
HRSL->AddDetector( new THaScintillator("s1", "S1 scintillator" ));
HRSL->AddDetector( new THaCherenkov("cer", "Gas Cherenkov counter" ));
gHaApps->Add(HRSL);

// Ideal beam (perfect normal incidence and centering)
THaIdealBeam* ib = new THaIdealBeam("IB", "Ideal beam");
gHaApps->Add(ib);

// Simple kinematics and vertex calculations
Double_t mass_tg = 12*931.494e-3; // C12 target
THaPhysicsModule *ekine, *rpl, *Lgold;
ekine = new THaElectronKine( "L.ekine", "Electron kinematics L", "L", "IB", mass_tg );
rpl   = new THaReactionPoint( "rpl", "Reaction vertex L", "L", "IB" );
Lgold = new THaGoldenTrack( "L.gold", "LHRS golden track", "L" );
gHaPhysics->Add(ekine);
gHaPhysics->Add(rpl);
gHaPhysics->Add(Lgold);

// The CODA data file we want to replay
THaRun* run = new THaRun("/rawdata/run_12345.dat");

// Set up and run standard analyzer (event loop)
THaAnalyzer* analyzer = new THaAnalyzer;
analyzer->SetOutFile( "/bigdisk/run_12345.root" );
analyzer->SetOdefFile("HRSL.odef");    // Define output
analyzer->Process(run);                // Process all invents in the input
```

More complex code usually needed to deal with file locations & split runs

Example Database

```
[ tutorial ]$ ls -FR DB
DB:
20120306/  db_cratemap.dat  db_run.dat  scaler.map

DB/20120306:
db_L.s1.dat  db_L.s2.dat  db_L.vdc.dat
```

- Recommended to set `DB_DIR` to point to top-level database directory
- Database files in current directory always take precedence
- Usually only detectors require databases (for the detector map). Physics modules sometimes read the run database (for masses, angles)
- Contents of files depends on corresponding module. (Will switch to consistent key/value format with version 1.6)
- Documentation on the web ▸ docs
- Hall C modules use Hall C-style parameter files

Example Output Definition File

```
# All variables from the GoldenTrack module
block L.gold.*

# Calculated quantities for inclusive electron scattering measured
# by the LHRS, form the ElectronKine physics module
block L.ekine.*

# All LHRS track data (focal plane as well as reconstructed to target)
block L.tr.*
```

- Much more possible
  - Arithmetic expressions
  - Using/defining cuts
  - 1D and 2D histograms
  - EPICS variables
  - Scalers
- Full documentation on the web ▸ docs (Bob Michaels)

FAQ: Help! Where Do I Find Those Variable Names?

Options:

- Inspect each module's `DefineVariables` function
- `Init()` analyzer (instead of `Process()`), then print variable list. May use wildcards to select subsets.

Variable List Printout (from tutorial, see later)

```
analyzer [0] .x init.C
analyzer [1] gHaVars->Print("","L.ekine.*")
Collection name='THaVarList', class='THaVarList', size=203
 OBJ: THaVar    L.ekine.Q2      4-momentum transfer squared (GeV^2)
 OBJ: THaVar    L.ekine.omega   Energy transfer (GeV)
 OBJ: THaVar    L.ekine.W2      Invariant mass of recoil system (GeV^2)
 OBJ: THaVar    L.ekine.x_bj    Bjorken x
 OBJ: THaVar    L.ekine.angle   Scattering angle (rad)
 OBJ: THaVar    L.ekine.epsilon Virtual photon polarization factor
 OBJ: THaVar    L.ekine.q3m     Magnitude of 3-momentum transfer
 OBJ: THaVar    L.ekine.th_q    Theta of 3-momentum vector (rad)
 OBJ: THaVar    L.ekine.ph_q    Phi of 3-momentum vector (rad)
 OBJ: THaVar    L.ekine.nu      Energy transfer (GeV)
 OBJ: THaVar    L.ekine.q_x     x-cmp of Photon vector in the lab
 OBJ: THaVar    L.ekine.q_y     y-cmp of Photon vector in the lab
 OBJ: THaVar    L.ekine.q_z     z-cmp of Photon vector in the lab
 ...
```

Other Useful Things You Might Need

1. Cut/test definition file
   - Defines logical tests to be evaluated at various analysis stages
   - Special "master" tests at each stage can be used as cuts to reject certain events
2. Environment setup script
   - Shell script to set up environment variables for replay
3. Calibration scripts
   - Special replay scripts
   - Some standardized (VDC time offsets), mostly experiment-specific
   - This is where the real work begins
4. Disk space!
   - Output ROOT files tend to be big, and numerous
   - Do not write production output to a home directory!
5. Software Development Kit
   - Skeleton classes for rapid development of your own code
   - Largely self-documenting ▸ download

Software Development Kit (SDK) Examples I

Example Physics Module Process() Function (simplified)

```
Int_t THaPrimaryKine::Process( const THaEvData& )
{
  // Calculate electron kinematics for the Golden Track of the given spectrometer

  // 4-momenta of incident & outgoing particle & target
  // NB: fP0 etc. are TLorentzVector objects

  fP0.SetVectM( fBeam->GetBeamInfo()->GetPvect(),     fM );  // e
  fP1.SetVectM( fSpectro->GetTrackInfo()->GetPvect(), fM );  // e'
  fA.SetXYZM( 0.0, 0.0, 0.0, fMA );                          // Target at rest

  // Textbook single-arm kinematics
  fQ          = fP0 - fP1;
  fQ2         = -fQ.M2();
  fQ3mag      = fQ.P();
  fOmega      = fQ.E();
  fA1         = fA + fQ;
  fW2         = fA1.M2();
  fTheta      = fP0.Angle( fP1.Vect() );
  fEpsilon    = 1.0 / ( 1.0 + 2.0*fQ3mag*fQ3mag/fQ2 *
                          TMath::Power( TMath::Tan(fTheta/2.0), 2.0 ));
  fThetaQ     = fQ.Theta();
  fPhiQ       = fQ.Phi();

  fDataValid = true;
  return 0;
}
```

Software Development Kit (SDK) Examples II

Example DefineVariables() Function

```
Int_t THaPrimaryKine::DefineVariables( EMode mode ) {
  // Define/delete global variables.
  if( mode == kDefine && fIsSetup ) return kOK;
  fIsSetup = ( mode == kDefine );

  RVarDef vars[] = {
    { "Q2",      "4-momentum transfer squared (GeV^2)",     "fQ2" },
    { "omega",   "Energy transfer (GeV)",                   "fOmega" },
    { "W2",      "Invariant mass of recoil system (GeV^2)", "fW2" },
    { "angle",   "Scattering angle (rad)",                  "fTheta" },
    { "epsilon", "Virtual photon polarization factor",      "fEpsilon" },
    { "q3m",     "Magnitude of 3-momentum transfer",        "fQ3mag" },
    { "th_q",    "Theta of 3-momentum vector (rad)",        "fThetaQ" },
    { "ph_q",    "Phi of 3-momentum vector (rad)",          "fPhiQ" },
    { "nu",      "Energy transfer (GeV)",                   "fOmega" },
    { "q_x",     "x-cmp of Photon vector in the lab",       "fQ.X()" },
    { "q_y",     "y-cmp of Photon vector in the lab",       "fQ.Y()" },
    { "q_z",     "z-cmp of Photon vector in the lab",       "fQ.Z()" },
    { 0 }
  };
  return DefineVarsFromList( vars, mode );
}
```

Software Development Kit (SDK) Examples III

Example ReadDatabase() Function (simplified)

```
Int_t THaADCHelicity::ReadDatabase( const TDatime& date )
{
  Int_t err = THaHelicityDet::ReadDatabase( date );
  if( err )  return kInitError;  // these error checks omitted below

  FILE* file = OpenFile( date );

  vector<Int_t> heldef;
  fThreshold  = kDefaultThreshold;
  Int_t ignore_gate = -1;
  const DBRequest request[] = {
    { "helchan",      &heldef,       kIntV,   0, 0, -2 },
    { "threshold",    &fThreshold,   kDouble, 0, 1, -2 },
    { "ignore_gate",  &ignore_gate,  kInt,    0, 1, -2 },
    { 0 }
  };
  err = LoadDB( file, date, request, fPrefix );
  fclose(file);

  // Do the all-important consistency checks!
  if( heldef.size() != 3 ) {
    Error( Here(here), "Incorrect defintion of helicity data channel. Must be "
   "exactly 3 numbers (roc,slot,chan), found %d. Fix database.", heldef.size() );
    return kInitError;
  }
  // ...
  fIsInit = true;
  return kOK;
}
```

Old Fixed-Format Databases

Example Podd 1.5 scintillator database db_L.s1.dat

```
######################### LEFT SCINTILLATOR PLANE 1 ############################
Number of Left Scintillator 1 paddles ---------------------
       16
Crate,Slot,1st,Last ADC chans,Beg S1 chan, model ----------------
       3      18      6        11      1       1881 - ADCs pads 1-6  (right)
       3      18      0        05      7       1881 - ADCs pads 7-12 (left)
       3      10      88       93      1       1877 - TDCs pads 1-6  (right)
       3      10      80       85      7       1877 - TDCs pads 7-12 (left)
      -1       0       0       0       0
X,Y,Z coords (in m) of S1 front plane in spectrom cs --------------
   -0.129     0.0   1.2873                                  - Meters
Half of X, half of Y, full Z sizes (in m) of S1 ----------------
    0.88    0.18     0.005                                  - Meters
Rotation angle of the S1 plane ------------------------
     0.0                                                    - Degrees
TDC time offsets of S1 in TDC channels ---------------------
     2.45 6.38 7.58 3.78 -13.25 3.75                        - Left Paddles
    -14.13 -16.83 -0.40 -3.78 -22.70 -0.12                  - Right Paddles
Pedestal values of S1 ADC channels ----------------------
   343 327 330 324 405 410                                  - Left Paddles
   429 432 344 345 385 389                                  - Right Paddles
Amplitude transformation coefficients of S1 ADC channels ------------
   1.093  0.960   1.082   1.215   1.102   1.116       - Left Paddles
   1.058  1.042   1.110   0.973   0.967   0.694       - Right Paddles
etc. ....
```

Old Fixed-Format Databases

Example Podd 1.5 scintillator database db_L.s1.dat

```
######################### LEFT SCINTILLATOR PLANE 1 #############################
Number of Left Scintillator 1 paddles ---------------------
       16 (?)
Crate,Slot,1st,Last ADC chans,Beg S1 chan, model ----------------
       3      18      6        11      1       1881 - ADCs pads 1-6  (right)
       3      18      0        05      7       1881 - ADCs pads 7-12 (left)
       3      10      88       93      1       1877 - TDCs pads 1-6  (right)
       3      10      80       85      7       1877 - TDCs pads 7-12 (left)
      -1       0       0        0      0
X,Y,Z coords (in m) of S1 front plane in spectrom cs --------------
   -0.129     0.0   1.2873                                    - Meters
Half of X, half of Y, full Z sizes (in m) of S1 ----------------
    0.88    0.18     0.005                                    - Meters
Rotation angle of the S1 plane -------------------------
     0.0                                                      - Degrees
TDC time offsets of S1 in TDC channels ---------------------
     2.45 6.38 7.58 3.78 -13.25 3.75                          - Left Paddles
    -14.13 -16.83 -0.40 -3.78 -22.70 -0.12                    - Right Paddles
Pedestal values of S1 ADC channels -----------------------
   343 327 330 324 405 410                                    - Left Paddles
   429 432 344 345 385 389                                    - Right Paddles
Amplitude transformation coefficients of S1 ADC channels ------------
   1.093  0.960   1.082   1.215   1.102   1.116        - Left Paddles
   1.058  1.042   1.110   0.973   0.967   0.694        - Right Paddles
etc. ....
```

New Free-Format Key-Value Databases

Conversion: `dbconvert DB-old/ DB-new/`

```
----[ 2015-03-24 00:00:00 -0400 ]

L.s1.detmap =
      3    18     6    11     1  1881
      3    18     0     5     7  1881
      3    10    88    93     1  1877
      3    10    80    85     7  1877
L.s1.npaddles = 6
L.s1.position = -0.1290  0.0000  1.2873
L.s1.size = 0.88  0.18  0.01
L.s1.L.off = 2.45  6.38  7.58  3.78  -13.25  3.75
L.s1.R.off = -14.13  -16.83  -0.40  -3.78  -22.70  -0.12
L.s1.L.ped = 343  327  330  324  405  410
L.s1.R.ped = 429  432  344  345  385  389
L.s1.L.gain = 1.093  0.960  1.082  1.215  1.102  1.116
L.s1.R.gain = 1.058  1.042  1.110  0.973  0.967  0.694
L.s1.avgres = 1e-10
L.s1.atten = 0.7
L.s1.Cn = 1.49859e+08
L.s1.MIP = 1e+10
L.s1.tdc.res = 5e-10
L.s1.timewalk_params =
   0  0  0  0  0  0
   0  0  0  0  0  0
L.s1.retiming_offsets = 0  0  0  0  0  0
```

New Free-Format Key-Value Databases

Conversion: `dbconvert DB-old/ DB-new/`

```
----[ 2015-03-24 00:00:00 -0400 ]

L.s1.detmap =
      3    18     6    11     1  1881
      3    18     0     5     7  1881
      3    10    88    93     1  1877
      3    10    80    85     7  1877
L.s1.npaddles = 6
L.s1.position = -0.1290  0.0000  1.2873
L.s1.size = 0.88  0.18  0.01
L.s1.L.off = 2.45  6.38  7.58  3.78  -13.25  3.75
L.s1.R.off = -14.13  -16.83  -0.40  -3.78  -22.70  -0.12
L.s1.L.ped = 343  327  330  324  405  410
L.s1.R.ped = 429  432  344  345  385  389
L.s1.L.gain = 1.093  0.960  1.082  1.215  1.102  1.116
L.s1.R.gain = 1.058  1.042  1.110  0.973  0.967  0.694
L.s1.avgres = 1e-10
L.s1.atten = 0.7
L.s1.Cn = 1.49859e+08
L.s1.MIP = 1e+10
L.s1.tdc.res = 5e-10
L.s1.timewalk_params =
   0  0  0  0  0  0
   0  0  0  0  0  0
L.s1.retiming_offsets = 0  0  0  0  0  0
```

Using The Tutorial Files

Example GMp optics replay setup download

Setting Up And Running the Tutorial on JLab CUE

```
ifarm> source /apps/root/PRO/setroot_CUE
ifarm> wget http://hallaweb.jlab.org/podd/download/tutorial-apr16.tar.gz
ifarm> tar xf tutorial-apr16.tar.gz
ifarm> cd Podd_Tutorial
ifarm> tar xf analyzer-1.6.0-beta1.tar.xz
ifarm> cd analyzer-1.6.0
ifarm> make -j4
ifarm> cd ..
ifarm> source setup.csh
ifarm> ln -s /work/halla/gmp12/ole/raw  data
ifarm> ln -s /volatile/halla/<experiment>/<user>  rootfiles
ifarm> cd replay
ifarm> analyzer
analyzer [0] .x replay.C
analyzer [1] .x plot.C
```

See the included `README` file for more info