Hall C Analyzer

Hall C Winter Collaboration Meeting

Eric Pooser

01/20/2017

In Collaboration with: Steve Wood, Mark Jones, Jure Bericic, Bob Michaels, Ole Hansen, Brad Sawatzky
Hall C Analyzer Overview

- Hall C ROOT Analysis framework (HCANA) is written in C++ and is an extension of the Hall A analyzer “podd”
  - Based on previous Fortran Analyzer ENGINE infrastructure
- Hall C ROOT Analyzer Wiki
- Maintained on GitHub
- Users fork off of “develop” branch
- Contributions are made in local/remote personal repository of HCANA
- Changes are pushed back onto the development branch which (may) get merged into the development branch
- Detailed "How-To" wiki: instructions how to “Git” started with HCANA and GitHub
In order to build and compile HCANA follow the [ROOT Analyzer/Compiling Wiki](#).

HCANA & podd are able to be built against ROOT 5.32 or later.

If developing in HCANA (or podd), ROOT 6.06 or greater is **HIGHLY** recommended.

HCANA can now be built with scons (recommended).

HCANA example analysis instructions found on the [ROOT Analyzer/Running Wiki](#).

---

**ROOT Analyzer/Compiling**

(Redirected from [Analyzer/Compiling](#))

The new Hall C analyzer, **hcana**, based on the Hall A analyzer is available and may be retrieved with git, see [ROOT Analyzer/Git](#).

The only prerequisite, beyond what should normally be installed on a system, is root, version 5.32 or later, but not version 6. On the JLab CUE systems, root may be setup by typing (or putting in your login script):

```
ssetup root
```

Note: Previous instructions used a version of root in `/group/hallc/root`. As the default version of root in `/app` is sufficient, this group disk version is no longer needed. References to it should be removed from your `.login` and `.cshrc` files.

On your own system, install root somehow.

Before compiling, go to the directory where the code is and setup the environment for hcana with

```
s source setup.csh
```

(Or "source setup.sh" if you use bash.) This setup script can be called in your login script. Make sure that in addition to the Hall C code in the `src` directory, the Hall A code is in the `podd` directory. See [Analyzer/Git](#) if it is not.

The analyzer may be compiled with either the traditional "make" utility, or with SCons, a new utility which allows additional auto-configuration abilities. SCons is available on JLab machines. If you do not have SCons installed on the local machine that you are compiling on, go to [SCons.org](http://scons.org) to install the latest version.

**To use "make", simply type:**

```
mke
```

**To use SCons, type:**

```
scons
```
**Hall C Analyzer Overview**

- [Doxygen page](#) documents HCANA source code

**Hall C ROOT/C++ Analyzer (hcana)**

### THcDC Class Reference

**Main detector classes**

This class analyzes a package of horizontal drift chambers. It uses the first letter of the apparatus name as a prefix to parameter names. The parameters, read in the Setup method, determine the number of chambers and the number of parameters per plane.

**Author**

S. A. Wood, based on Fortran ENGINE

**Public Member Functions**

```c
THcDC (const char *name, const char *description":", THaApparatus *a=NULL)  
virtual ~THcDC ()  
virtual Int_t Decode (const THaEvData &)
virtual EStatus Init (const TDateTime &run_time)
virtual Int_t CoarseTrack (TClonesArray &tracks)
virtual Int_t FineTrack (TClonesArray &tracks)
virtual Int_t ApplyCorrections (void)
  Int_t GetNWires (Int_t plane) const
  Int_t GetNChamber (Int_t plane) const
  Int_t GetWireOrder (Int_t plane) const
  Double_t GetPitch (Int_t plane) const
  Double_t GetCentralWire (Int_t plane) const
  Int_t GetTdcWinMin (Int_t plane) const
```
Recent Updates to HCANA

- New classes created
- **THcDummySpectrometer**
  - Used to test individual detector components by excluding dependency on higher level tracking objects
  - Primarily used to investigate raw ADC & TDC spectra
- **THcTrig** is family of classes created to make use of trigger variables
  - Reference times and detector ADC sums
Recent Updates to HCANA

- Modifications made to THcRawAdcHit class to include the dynamic output from the FADC250’s (excludes DC’s)
  - Includes both raw FADC FPGA outputs & “readable” outputs
  - All detector classes were modified to process and output the new variables
  - Both raw and reference time subtracted TDC values are also returned
  - Old style leaves were preserved for now, eventually these leaves will be abandoned with a tagged version

Eric Pooser
01/20/17
Hall C Winter Collaboration Meeting
JLab FADC250 Modules

- 16 Channel 250 MHz VME64x flash-to-analog digital converters
- Configurable to 0.5, 1.0, 2.0V dynamic range per channel
- Provides dynamic output of pulse profile parameters
  - Mode 9: pedestal, integral, amplitude, high resolution time
  - Pulse time is time-walk free!
    - 62.5 ps resolution
  - Mode 10: Mode 9 + pulse profile
- Multi-hit capability!
  - Configurable to register up-to four hits in the programmable trigger window
Hall A/C FADC250 Decoder

- Revamped Fadc250Module class has been written and resides in podd
- Backwards compatible with old (pre 0x0C00) firmware versions
- New standalone utility tstfadc_main.C included in podd/hana_decode
- Scans crate for F250 modules, histograms all possible data in crate/slots identified as F250

![FADC Mode 9 Pulse Peak Data Slot 8](image)

**ProjectionY of binx=12 [x=10.5..11.5]**

- Entries: 15217
- Mean: 876
- Std Dev: 436.6
Hall A/C FADC250 Decoder

- Revamped Fadc250Module class has been written for podd
- Backwards compatible with old (pre 0x0C00) firmware versions
- New standalone utility tstfadc_main.C included in podd/hana_decode
- Scans crate for F250 modules, histograms all possible data in crate/slot identified as a F250

![Graph showing 25 Raw Events of FADC Mode 10 Sample Data Slot 3 Channel 15]
**Hall C Replay**

- Framework designed to facilitate the reconstruction of events in the Hall C spectrometers
- The [Hall C Replay Structure Wiki](https://example.com) provides general information regarding the infrastructure
- The Hall C replay code is maintained in the [JeffersonLab/hallc_replay](https://github.com/JeffersonLab/hallc_replay) GitHub repository

**Experiments should maintain their own version of this repository**
Hall C Replay

CONFIGURATION FILES

RUN
PARAM
STD
KINEM
HMS
SHMS
GEN
TRIG

DBASE

PARAM

1-99999
gpbeam=999
gtarg_num = 999
hteta_lab = 999
stheta_lab = 999
hpcentral = 999
spcentral = 999
hpcentermass = 0.00051099
spcentermass = 0.00051099
Hall C Replay

CONFIGURATION FILES

RUN
PARAM
STD
KINEM
HMS
SHMS
GEN
TRIG

DBASE
SHMS
HMS
DETEC
TRIG

DEF FILES

PARAM
SHMS
HMS
CRATE

FILES

MAPS
void replay_phodo_test_stand(Int_t RunNumber=0, Int_t MaxEvent=0)
{
    // Get RunNumber and MaxEvent if not provided.
    if(RunNumber == 0)
    {
        cout << "Enter a Run Number (-1 to exit): ";
        cin >> RunNumber;
        if( RunNumber<= 0 ) return;
    }
    if(MaxEvent == 0)
    {
        cout << "Number of Events to analyze: ";
        cin >> MaxEvent;
        if(MaxEvent == 0 )
        { cerr << "...Invalid entry\n";
            exit;
        }
    }

    // Create file name patterns.
    const char* RunFileNamePattern = "raw/shms_all_%05d.dat";
    const char* ROOTFileNamePattern = "ROOTfiles/phodo_replay_%d.root";
    // Add variables to global list.
    ghcParms->Define("gen_run_number", "Run Number", RunNumber);
    ghcParms->AddString("g_ctp_database_filename", "DATABASE/standard.database");
    // Load variables from files to global list.
    ghcParms->Load(ghcParms->GetString("g_ctp_database_filename"), RunNumber);
    g_ctp_parm_filename and g_decode_map_filename should now be defined.
    ghcParms->Load(ghcParms->GetString("g_ctp_kinematics_filename"), RunNumber);
    ghcParms->Load(ghcParms->GetString("g_ctp_parm_filename"));
    // Load params for HMS trigger configuration
    ghcParms->Load("PARAM/TRIG/tshms.param");
    // Load the Hall C style detector map
    ghcDetectorMap = new THcDetectorMap();
    ghcDetectorMap->Load("MAPS/SHMS/DETEC/phodo_ptrig.map");
    // Set up the equipment to be analyzed.
    THcApparatus* SHMS = new THcHallCScintillator("P", "SHMS");
    ghcApps->Add(SHMS);
    // Add hodoscope to HMS apparatus
    THcHodoscope* hodo = new THcHodoscope("hod", "Hodoscope");
    SHMS->AddDetector(hodo);
    // Add trigger apparatus
    THcApparatus* TRG = new THcTrigApp("T", "TRG");
    ghcApps->Add(TRG);
    // Add trigger detector to trigger apparatus
    THcDet* shms = new THcDet("shms", "SHMS Trigger Information");
    TRG->AddDetector(shms);
    // Set up the analyzer - we use the standard one,
    // but this could be an experiment-specific one as well.
    // The Analyzer controls the reading of the data, executes
    // tests/cuts, loops over Acpparatus's and PhysicsModules,
    // and executes the output routines.
    THcAnalyzer* analyzer = new THcAnalyzer;
    // A simple event class to be output to the resulting tree.
    // Creating your own descendant of THcEvent is one way of
    // defining and controlling the output.
    THcEvent* event = new THcEvent;
    // Define the run(s) that we want to analyze.
    // We just set up one, but this could be many.
    char RunFileName[100];
    sprintf(RunFileName, RunFileNamePattern, RunNumber);
    THcRun* run = new THcRun(RunFileName);
    // Eventually need to learn to skip over, or properly analyze
    // the pedestal events
    run->SetEventRange(1, MaxEvent); // Physics Event number, does not
    // include scaler or control events.
    run->SetNscan(1);
    run->SetDataRequired(0x7);
    run->Print();
    // Define the analysis parameters
    TString ROOTFileName = Form(ROOTFileNamePattern, RunNumber);
    analyzer->SetCountMode(2); // 0 = counter is # of physics triggers
    // 1 = counter is # of all decode reads
    // 2 = counter is event number
    analyzer->SetEvent(1);
    analyzer->SetEventMapFile("MAPS/db_crateemap.dat");
    analyzer->SetOutFile(ROOTFileName.Data());
    analyzer->SetOutput(1 /*DEF-files/SHMS/HOD0/phodoana.def*/);
    analyzer->SetCutFile(1 /*DEF-files/SHMS/HOD0/phodoana_cuts.def*/); // optional
    // File to record cuts and naming information
    analyzer->SetSummaryFile("summary_example.log"); // optional
    // Start the actual analysis.
    analyzer->Process(run);
    // Create report file from template.
    analyzer->PrintReport(); // optional
    // "TEMPLATES/dcanva.template",
    // Form("REPORT_OUTPUT/replay_hms_%05d.report", RunNumber)
    //);
Hall C Replay

**Configuration Files**

- **RUN**
- **PARAM**
- **STD**
- **KINEM**
- **HMS**
- **SHMS**
- **GEN**
- **TRIG**

**Analysis Files**

- **HODO**
- **DC+CAL**
- **HMS**
- **SHMS**
- **CER+DC**
- **AERO**

**Results**

- **ROOT TREE**
- **DEF-FILE HISTOS**
- **ONLINE GUI**
Hall C Replay: ROOT Tree
Hall C Replay: DEF-Files Histograms

SHMS 1X+ Raw TDC vs. Paddle Number

- Entries: 263426
- Mean x: 7.27
- Mean y: -1118
- Std Dev x: 3.602
- Std Dev y: 71.69

Canvas_1 Editor 1

Command:

Command (local):
Hall C Online GUI

- Contained within the Hall C online replay framework & [GitHub repository](#)

- “Macros” directory contains ROOT macros which analyze root trees produced by HCANA
  - ROOT files produced *via* Hall C replay

- “Config” directory contains files which call specific methods contained in ROOT macros to display specific histograms in the online GUI

- Useful "How-To" PDF discusses the details of the online GUI located in top level directory
Hall C Online GUI: Making Plots Great Again!

- HMS X1 - Raw TDCs
- HMS X1 - Raw TDCs
- HMS X1 - TDCs
- HMS X1 - TDCs
- HMS X1 - FADC Raw Pedestals
- HMS X1 - FADC Raw Pedestals
- HMS X1 - FADC Pedestals
- HMS X1 - FADC Pedestals
- HMS X1 - FADC Raw Integrals
- HMS X1 - FADC Raw Integrals
- HMS X1 - FADC Integrals
- HMS X1 - FADC Integrals
- HMS X1 - FADC Amplitudes Raw
- HMS X1 - FADC Amplitudes Raw
- HMS X1 - FADC Amplitudes
- HMS X1 - FADC Amplitudes
- HMS X1 - FADC Times Raw
- HMS X1 - FADC Times Raw

Prev Next Exit GUI Run #1000 Print To File
Hall C Online GUI: Making Plots Great Again!

SHMS Y2+ Raw TDCs
SHMS Y2+ Raw TDCs
SHMS Y2+ TDCs
SHMS Y2- TDCs
SHMS Y2+ FADC Raw Pedestals
SHMS Y2- FADC Raw Pedestals
SHMS Y2+ FADC Pedestals
SHMS Y2- FADC Pedestals
SHMS Y2+ FADC Raw Integrals
SHMS Y2- FADC Raw Integrals
SHMS Y2+ FADC Integrals
SHMS Y2- FADC Integrals
SHMS Y2+ FADC Amplitudes Raw
SHMS Y2- FADC Amplitudes Raw
SHMS Y2+ FADC Amplitudes
SHMS Y2- FADC Amplitudes
SHMS Y2+ FADC Times Raw
SHMS Y2- FADC Times Raw
Looking Forward

- Remaining HMS/SHMS Cherenkov & aerogel detectors need to be included in the Hall C replay infrastructure
- Remaining HMS & SHMS detectors need to be included in the online GUI
- Attention is needed for the hodoscope classes
- Physics classes need to be developed
- Input is needed from scheduled experiments regarding what they require from the analyzer
- Develop detailed wiki with examples for both the Hall C replay and online GUI frameworks
Announcements

- Hall A/C software meetings held bi-weekly at 11 am in L201: [Hall C 12 GeV Software Wiki](#)
- Mailing list [hallcsw@jlab.org](mailto:hallcsw@jlab.org)
- If folks are interested in a walk through session for building/compiling HCANA, executing the Hall C replay framework, and the utilizing the Online GUI, we will hold an informal tutorial Saturday (tomorrow) afternoon following the final talk
- Ambitious developers always encouraged to participate!
Backup Slides
JLab FADC250 FPGA

Threshold (TET)

PTW

PL

TC  (TC+NSAT)

NSB  NSAT  NSA

Trigger
FADC250 FPGA Constant Fraction Algorithm

- Analog Signal
- Pedestal Samples
- $V_{\text{ped}}$
- 4 ns Samples
- SN−1
- $t_{\text{mid}}$, $V_{\text{mid}}$
- SN +1
- $V_{\text{peak}}$

Volts

Time →

Threshold

Baseline
JLab FADC250 FPGA

Timewalk Effect

- Baseline
- Threshold

Volts

Time

Discriminator Output A
Discriminator Output B
Discriminator Output C

"Timewalk"

$t_A$, $t_B$, $t_C$
JLab FADC250 FPGA

FADC250 FPGA Constant Fraction Pulse Time

- Analog Signal A
- Analog Signal B
- Analog Signal C

Volts

Threshold

Baseline

Pulse Time Output A
Pulse Time Output B
Pulse Time Output C

No "Timewalk"

\[ t_A = t_B = t_C \]