Brief Document On the Hall C Coincidence Time Physics Module

Carlos Yero

May 7, 2018

The Hall C Coincidence Time Physics Module, THcCoinTime.cxx(.h), was created to facilitate the analysis of coincidence time data by adding coincidence time variables to the ROOT Tree, so that users may apply any cuts they desise to clean-up the data. This document intends to briefly cover what was done in physics module, as well as the mathematical background, which was written by Latif Kabir during the start of Commissioning in 2018.

1 Discussion on the Physics of Coincidence Timing

The coincidence time measured between two spectrometers is defined as the time difference between the two single-arm spectrometer triggers. The time difference is determined using the raw (uncorrected) TDC times, therefore, additional corrections will be needed to reduce the natural spread in the coincidence peak. As pointed out in Latif's paper, the corrections are as follows:

1. The leading order correction is the time taken by the partilcle to arrive at the focal plane following the central path, L_0, of the spectrometer.
2. The second order correction is due to the fact that the particle may not necessarily follow the central path, therefore, the time difference between particle path and central path, ΔL, needs to be accounted for.
3. The third order correction is the fluctuations in the focal plane time with respect to the central time.
4. The last correction has more to do with a general offset of the coincidence peak, rather than an effect on the resolution. The peak is typically shifted to zero, to represent the occurrence of a true coincidence event originating from the same electron beam bunch.

The raw coincidence times in the physics module were mathematically expressed as

$$t_{\text{coin, raw}} = (t_{\text{SHMS, raw}} + t_{\text{SHMS, fp}}) - (t_{\text{HMS, raw}} + t_{\text{HMS, fp}}) \tag{1}$$

where the focal plane times were included as a correction to the raw tdc trigger times to simplify calculations later on. The corrected coincidence time can then be expressed as

$$t_{\text{coin, corr.}} = t_{\text{coin, raw}} \pm (t_{\text{elec. corr}} - t_{\text{hadron corr.}}) - \delta_{\text{offset}} \tag{2}$$

where the sign convention is

$$\text{sign} = \begin{cases} + & \text{SHMS (hadrons), HMS (electrons)} \\ - & \text{SHMS (electrons), HMS (hadrons)} \end{cases}$$

The sign depends on the particle type the spectrometer is set to detect, as we want to ensure the correction is *subtracted off* from the right spectrometer raw trigger time. For example, if the SHMS is set to detect electrons, then substituting Eq. 1 in Eq. 2, one obtains

$$\begin{aligned} t_{\text{coin, corr.}} &= (t_{\text{SHMS, raw}} + t_{\text{SHMS, fp}}) - (t_{\text{HMS, raw}} + t_{\text{HMS, fp}}) - (t_{\text{elec. corr}} - t_{\text{hadron corr.}}) - \delta_{\text{offset}} \\ &= (t_{\text{SHMS, raw}} + t_{\text{SHMS, fp}} - t_{\text{elec. corr}}) - (t_{\text{HMS, raw}} + t_{\text{HMS, fp}} - t_{\text{hadron corr.}}) - \delta_{\text{offset}} \end{aligned} \tag{3}$$

By subtracting the electron/hadron timing corrections, the dependence on the particle pathlength and beta, β, are removed. Therefore, if a *true* coincidence occurs, it will land at the same location in the coincidence timing spectrum, regardless of the particle type. See Figure 1.

The timing corrections for electrons and hadrons in Eq. 3 can be expressed in the general form

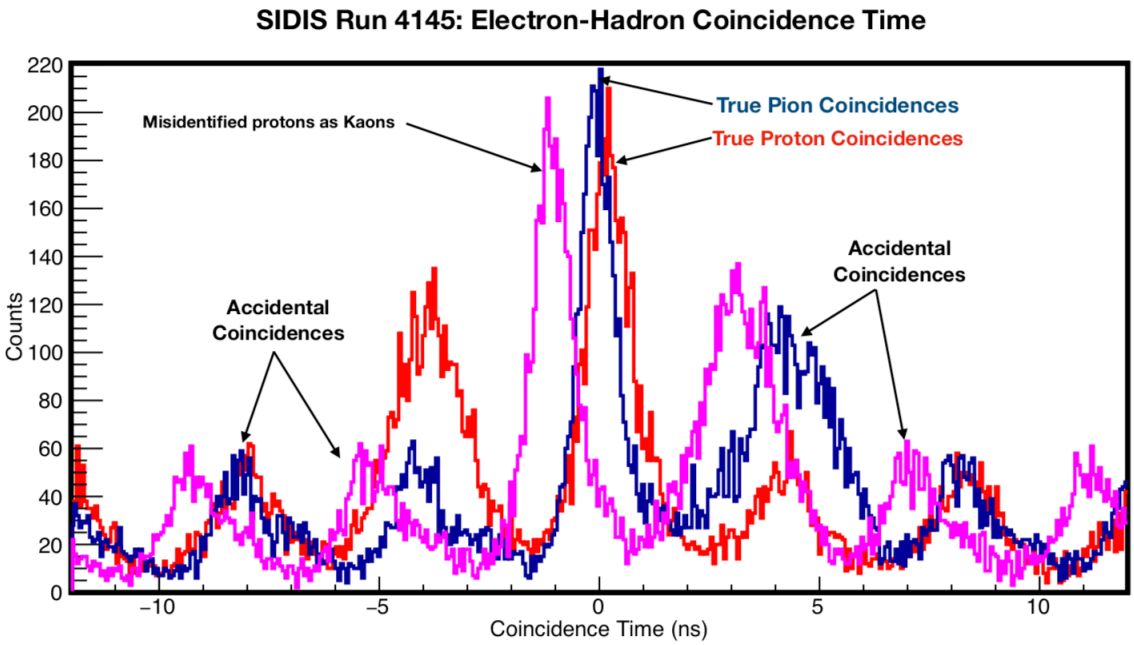

Figure 1: Electron-hadron coincidence time, assuming proton, π^+ and K^+ mass.

$$t_{\mathrm{e(h)\ corr.}} = \frac{L_0 + \Delta L}{\beta c}, \text{ where } \beta \equiv \frac{p}{E} = \frac{p}{\sqrt{m^2 + p^2}} \tag{4}$$

The variables (L_0, ΔL) are the 1st and 2nd order path length corrections, corresponding to either the HMS or SHMS, depending on which particle is assigned to which spectrometer. The variable β, represents the particle fractional velocity relative to the speed of light, and depends on the particle mass and momentum. Therefore, subtracting off all these dependences from the raw coincidence time, places the coincidence peak at a certain offset, δ_{offset}, which can then be determined experimentally, and shifted to zero (0 ns). (See Figure 2)

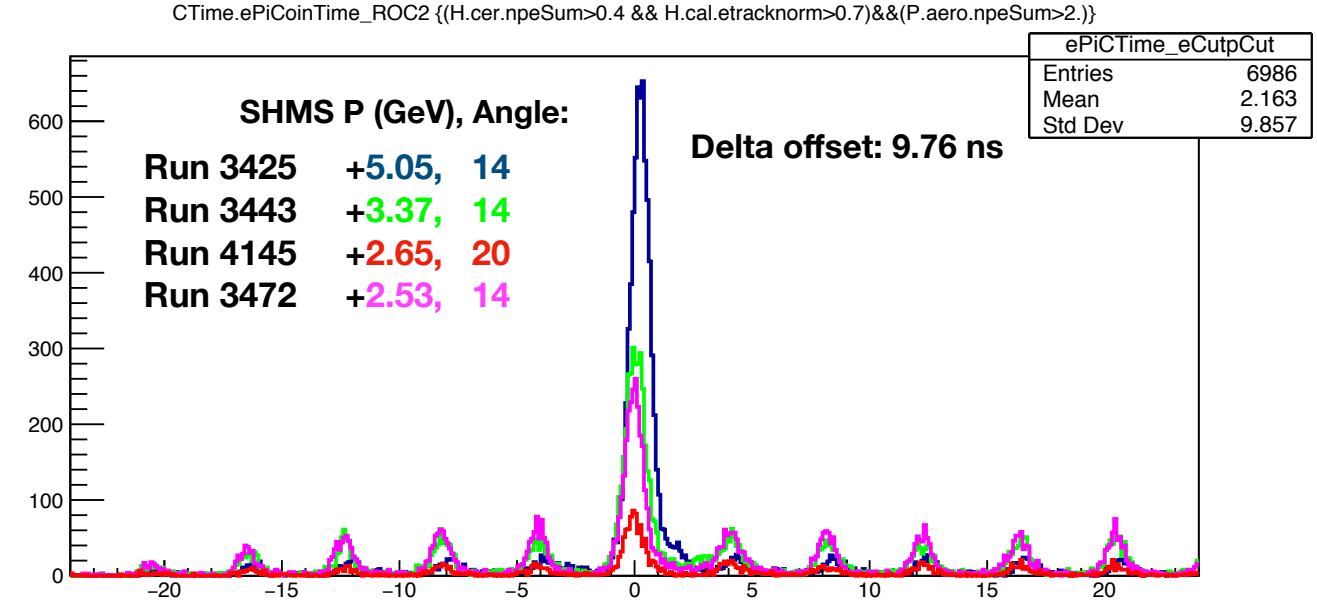

Figure 2: $e\pi^+$ coincidence time, assuming pion mass. A δ_{offset} has been determined and applied to several SIDIS runs.

A δ_{offset} of 9.76 ns was determined from SIDIS run 4145, using clean $e^-\pi^+$ coincidences. A calorimeter and Cherenkov cut was applied on the HMS for selecting electrons, and an Aerogel photoelectron cut was applied on the SHMS to eliminate proton contamination showing up in the accidentals. After obtaining a clean sample of $e^-\pi^+$ coincidences, a fit was done to the main peak to determine the offset. The stability of the offset was tested for various SIDIS runs at different SHMS momenta. (See Figure 2)

2 Discussion on Coincidence Time Module Workflow

The coincidence time physics module can be found under hcana/src directory, with the name of `THcCoinTime.cxx(.h)`. The module is a derived class from more general Hall A class `THaPhysicsModule.C(.h)`, and hence, it also uses all of its member functions/methods. The class is called in the main Hall C analysis coincidence replay scripts, located in: `/hallc_replay/SCRIPTS/COIN/PRODUCTION/`. I will start by discussing how the physics module is calle in the analyzer, and then briefly discuss the relevant methods used for calculating the coincidence time in chronological order.

Listing 1: Adding the Module in the Analyzer

```
1   void replay_production_coin_hElec_pProt(Int_t RunNumber = 0, Int_t MaxEvent = 0) {
2
3     //A few lines later, towards the end of the code (line # are not same)
4     //This replay script is in: /hallc_replay/SCRIPTS/COIN/PRODUCTION/
5
6     THcCoinTime* coinTime = new THcCoinTime("CTime", "Coincidence Time Determination",
7     "P", "H", "T.coin");
8     gHaPhysics->Add(coinTime);
9
10  }
```

In the snippet of code above (Listing 1), an instance (coinTime) of the class is created and initialized with arguments that will be used later on. The order in which the arguments "P" and "H" are called is important, as they dictate which spectrometer will be the electron or hadron arm, an so they must be consistent with the replay script that is called.

Listing 2: THcCoinTime.cxx, The Constructor Method

```
1     THcCoinTime::THcCoinTime (const char *name, const char* description,
2     const char* hadArmName, const char* elecArmName, const char* coinname) :
3
4   THaPhysicsModule(name, description),
5   fCoinDetName(coinname),
6   fhadArmName(hadArmName),                    //initialize spectro names
7   felecArmName(elecArmName),
8   fhadSpectro(NULL),                         //initialize spectro objects
9   felecSpectro(NULL),
10  fCoinDet(NULL)
11 {
12
13 }
```

In the Listing 2 snippet above, the constructor is defined with some arguments, which were called in Listing 1. The first argument, *name*, represents the name of the class and will be the *prefix-* in every ROOT leaf variable that is added in the DefineVariables() method. The third and fourth arguments, take the char value "P" or "H", depending on which spectrometer will be the hadron/electron arm. The arguments in the constructor are paseed on in lines 4-7. Additional `THcHallCSpectrometer.cxx` objects are initialized in lines 8-10, and will be used later on.

Listing 3: THcCoinTime.cxx, The Init() Method

```
1   THaAnalysisObject::EStatus THcCoinTime::Init( const TDatime& run_time )
2   {
3     // Initialize THcCoinTime physics module
4
5   fhadSpectro = dynamic_cast<THcHallCSpectrometer*>
6     ( FindModule( fhadArmName.Data(), "THcHallCSpectrometer"));
7   if( !fhadSpectro ) {
8     cout << "THcCoinTime module  Cannnot find Hadron Arm = " <<  fhadArmName.Data() << endl;
9     fStatus = kInitError;
10    return fStatus;
11  }
12
```

```
13    //The other two objects are also casted, but due to space, were not included.
```

In the `Init()` method above, the objects initialized to NULL in the constructor are converted to `THcHallCSpectrometer` objects via a `dynamic_cast`, which enables the object to use all member functions of mentioned class.

Listing 4: THcCoinTime.cxx, The ReadDatabase() Method

```
1  Int_t THcCoinTime::ReadDatabase( const TDatime& date )
2  {
3    // Read database. Gets variable needed for CoinTime calculation
4
5
6    DBRequest list[]={
7      {"eHadCoinTime_Offset",  &eHad_CT_Offset, kDouble, 0, 1},
8      {"HMS_CentralPathLen",  &HMScentralPathLen, kDouble, 0, 1},
9      {"SHMS_CentralPathLen", &SHMScentralPathLen, kDouble, 0, 1},
10     {0}
11   };
12
13   //Default values if not read from param file
14   eHad_CT_Offset = 0.0;
15
16   HMScentralPathLen = 22.0*100.;
17   SHMScentralPathLen = 18.1*100.;
18
19
20   gHcParms->LoadParmValues((DBRequest*)&list, "");
21
22   return kOK;
23 }
```

The `ReadDatabase()` method searches for a `coincidence time offset` and `SHMS/HMS Path Length` parameters. The `coincidence time offset` parameter is located in `/PARAM/TRIG/tcoin.param`, and the `SHMS/HMS Path Length` parameters are located in `/PARAM/SHMS/GEN/pcana.param` and `/PARAM/HMS/GEN/hcana.param` directories, respectively. In the scenario that any of the above parameters were NOT found, they would be initialized by default under this method. (See line # 14, 16 and 17 of this method)

Listing 5: THcCoinTime.cxx, The Process() Method

```
1  Int_t THcCoinTime::Process( const THaEvData& evdata )
2  {
3
4   //Create THaTrack object for hadron/elec arms to get relevant golden track quantities
5    if (felecArmName=="H") {
6      theSHMSTrack =(fhadSpectro->GetGoldenTrack());
7      theHMSTrack = (felecSpectro->GetGoldenTrack());
8    } else{
9      theSHMSTrack =(felecSpectro->GetGoldenTrack());
10     theHMSTrack = (fhadSpectro->GetGoldenTrack());
11   }
12   //Gather relevant variables for Coincidence time calculation
13   lightSpeed = 29.9792; // in cm/ns
14
15   //Particle Masses (HardCoded)
16   elecMass =  0.510998/1000.0; // electron mass in GeV/c^2
17   positronMass =  0.510998/1000.0;
18   protonMass = 938.27208/1000.0; // proton mass in GeV/c^2
19   kaonMass = 493.677/1000.0;     //charged kaon mass in GeV/c^2
20   pionMass = 139.570/1000.0;     //charged pion mass in GeV/c^2
21
22
```

```
23          //SHMS arm
24          Double_t shms_xptar = theSHMSTrack->GetTTheta();
25          Double_t shms_dP = theSHMSTrack->GetDp();
26          Double_t SHMS_FPtime = theSHMSTrack->GetFPTime();
27
28          //HMS arm
29          Double_t hms_xfp = theHMSTrack->GetX();
30          Double_t hms_xpfp = theHMSTrack->GetTheta();
31          Double_t hms_ypfp = theHMSTrack->GetPhi();
32          Double_t HMS_FPtime = theHMSTrack->GetFPTime();
33
34          //Get raw TDC Times for HMS/SHMS (3/4 trigger)
35          pTRIG1_rawTdcTime_ROC1 = fCoinDet->Get_pTRG1_ROC1_rawTdctime();
36          pTRIG4_rawTdcTime_ROC1 = fCoinDet->Get_pTRG4_ROC1_rawTdctime();
37          pTRIG1_rawTdcTime_ROC2 = fCoinDet->Get_pTRG1_ROC2_rawTdctime();
38          pTRIG4_rawTdcTime_ROC2 = fCoinDet->Get_pTRG4_ROC2_rawTdctime();
39
40          DeltaSHMSpathLength = -0.11*atan2(shms_xptar,1)*1000 - 0.057*shms_dP;
41          DeltaHMSpathLength = 12.462*hms_xpfp + 0.1138*hms_xpfp*hms_xfp - 0.0154*hms_xfp
42              - 72.292*hms_xpfp*hms_xpfp - 0.0000544*hms_xfp*had_xfp - 116.52*hms_ypfp*hms_ypfp;
43
44              // default assume SHMS is electron arm
45          Double_t ElecPathLength=SHMScentralPathLen +  DeltaSHMSpathLength;
46          Double_t HadPathLength=HMScentralPathLen +  DeltaHMSpathLength;
47
48              elec_P = theSHMSTrack->GetP();        //electron golden track arm momentum
49              had_P = theHMSTrack->GetP();          //hadron golden track arm momentum
50
51              Int_t sign=-1;
52
53              if (felecArmName=="H") {
54                  ElecPathLength=HMScentralPathLen +  DeltaHMSpathLength;
55             HadPathLength=SHMScentralPathLen +  DeltaSHMSpathLength;
56                  elec_P = theHMSTrack->GetP();     //electron golden track arm momentum
57                  had_P = theSHMSTrack->GetP();     //hadron golden track arm momentum
58                  sign=1;
59          }
60
61          //beta calculations beta = v/c = p/E
62          elecArm_BetaCalc = elec_P / sqrt(elec_P*elec_P + elecMass*elecMass);
63          hadArm_BetaCalc_proton = had_P / sqrt(had_P*had_P + protonMass*protonMass);
64
65
66          //Coincidence Corrections
67          elec_coinCorr = (ElecPathLength) / (lightSpeed * elecArm_BetaCalc );
68          had_coinCorr_proton = (HadPathLength) / (lightSpeed * hadArm_BetaCalc_proton );
69
70          //Raw, Uncorrected Coincidence Time
71          fROC1_RAW_CoinTime =  (pTRIG1_rawTdcTime_ROC1*0.1 + SHMS_FPtime)
72              - (pTRIG4_rawTdcTime_ROC1*0.1 + HMS_FPtime);
73
74          //Corrected Coincidence Time for ROC1/ROC2 (ROC1 Should be identical to ROC2)
75          //PROTON
76          fROC1_epCoinTime = fROC1_RAW_CoinTime + sign*( elec_coinCorr-had_coinCorr_proton)
77              - eHad_CT_Offset;
78
79
80      return 0;
81
82  }
```

The coincidence time calculations are done in the Process() method. It may be non-trivial to follow everything that is done here, so I will attempt to summarize it based on the code line numbers.

- Lines 4-10: Based on whether the electron arm is set to HMS("H") or SHMS("P"), then associate the `GetGoldeTrack()` method with either `THaTrack.cxx` objects for HMS or SHMS. The quantities necessary to do the calculations can then be obtained from `theSHMSTrack` or `theHMSTrack THaTrack.cxx` objects.
- Lines 23-33: The relevant HMS/SHMS variables to be used in the Path Length correction are obtained from the track objects mentioned in the first bullet.
- Lines 34-38: Call functions to get the raw TDC Time from each spectrometer single arm triggers, `pTRIG1(3/4 SHMS)` and `pTRIG4(3/4 HMS)`. The `fCoinDet` object is an instance of the `THcTrigDet.cxx` class, and calls methods to get these raw times, which had to be added in the abovementioned class during the time this module was created.
- Lines 44-60: Start with the assumption that the SHMS is the electron arm, determine the corrected path length and golden track momentum, and most importantly, define the sign of the correction to be -1, so that the correction to the electron/hadron arms is actually subtracted from the raw times, and eliminating any dependence of the coincidence time on the particle type. If the electron arm is the HMS, then the corrected path length, golden track momentum and the sign invert roles between spectrometers.
- Lines 61-78: Once the electron and hadron arm are associated with a spectrometer, then lines 62-63 determine the velocity of the electron and hadron. (I only included the Proton, but the calculation was also done for Kaons, Pions and Positrons.) Lines 67-68 determine the time it takes the particle to traverse through the spectrometer into the focal plane. Lines 71-72 determine the raw coincidence times by taking the difference between the raw TDC times determined in Lines 35-38. The focal plane times are also included here, so simplify the calculation later on. Finally, Lines 76-77 combine the elements in Lines 66-72, along with the sign, to determine the corrected coincidence time. An offset is added towards the end in order to shift the coincidence peak to zero. This offset is read as a parameter in the `ReadDatabase()` method.