PID @ GlueX

- Currently we are using cut-based selections on simple variables, e.g. dE/dx, ToF
- There currently exists the possibility to do more
- Follow me down the rabbit hole…

DChargedTrackHypothesis

We are currently using this

```
//Timing
double t0(void) const{return dTimingInfo->dt0;}
double t0_err(void) const{return dTimingInfo->dt0_err;}
double t1(void) const;
double t1_err(void) const;
DetectorSystem_t t0_detector(void) const{return dTimingInfo->dt0_detector;}
DetectorSystem_t t1_detector(void) const;
double Get_TimeAtPOCAToVertex(void) const{return dTimingInfo->dTimeAtPOCAToVertex;}
unsigned int Get_NDF_Timing(void) const{return dTimingInfo->dNDF_Timing;}
double Get_ChiSq_Timing(void) const{return dTimingInfo->dChiSq_Timing;}
double Get_PathLength(void) const;
double measuredBeta(void) const{return Get_PathLength()/(29.9792458*(t1() - t0()));}

//totals for overall PID determination
unsigned int Get_NDF(void) const{return dTimingInfo->dNDF;}
double Get_ChiSq(void) const{return dTimingInfo->dChiSq;}
double Get_FOM(void) const{return dTimingInfo->dFOM;}


//Tracking
void Set_TrackTimeBased(const DTrackTimeBased* locTrackTimeBased
void Set_ChiSq_DCdEdx(double locChiSq, unsigned int locNDF);
```

DChargedTrackHypothesis

We are currently using this

We could also use this

```
//Timing
double t0(void) const{return dTimingInfo->dt0;}
double t0_err(void) const{return dTimingInfo->dt0_err;}
double t1(void) const;
double t1_err(void) const;
DetectorSystem_t t0_detector(void) const{return dTimingInfo->dt0_detector;}
DetectorSystem_t t1_detector(void) const;
double Get_TimeAtPOCAToVertex(void) const{return dTimingInfo->dTimeAtPOCAToVertex;}
unsigned int Get_NDF_Timing(void) const{return dTimingInfo->dNDF_Timing;}
double Get_ChiSq_Timing(void) const{return dTimingInfo->dChiSq_Timing;}
double Get_PathLength(void) const;
double measuredBeta(void) const{return Get_PathLength()/(29.9792458*(t1() - t0()));}

//totals for overall PID determination
unsigned int Get_NDF(void) const{return dTimingInfo->dNDF;}
double Get_ChiSq(void) const{return dTimingInfo->dChiSq;}
double Get_FOM(void) const{return dTimingInfo->dFOM;}


//Tracking
void Set_TrackTimeBased(const DTrackTimeBased* locTrackTimeBased
void Set_ChiSq_DCdEdx(double locChiSq, unsigned int locNDF);
```

DChargedTrackHypothesis

We are currently using this

Or this

We could also use this

```
//Timing
double t0(void) const{return dTimingInfo->dt0;}
double t0_err(void) const{return dTimingInfo->dt0_err;}
double t1(void) const;
double t1_err(void) const;
DetectorSystem_t t0_detector(void) const{return dTimingInfo->dt0_detector;}
DetectorSystem_t t1_detector(void) const;
double Get_TimeAtPOCAToVertex(void) const{return dTimingInfo->dTimeAtPOCAToVertex;}
unsigned int Get_NDF_Timing(void) const{return dTimingInfo->dNDF_Timing;}
double Get_ChiSq_Timing(void) const{return dTimingInfo->dChiSq_Timing;}
double Get_PathLength(void) const;
double measuredBeta(void) const{return Get_PathLength()/(29.9792458*(t1() - t0()));}

//totals for overall PID determination
unsigned int Get_NDF(void) const{return dTimingInfo->dNDF;}
double Get_ChiSq(void) const{return dTimingInfo->dChiSq;}
double Get_FOM(void) const{return dTimingInfo->dFOM;}


//Tracking
void Set_TrackTimeBased(const DTrackTimeBased* locTrackTimeBased
void Set_ChiSq_DCdEdx(double locChiSq, unsigned int locNDF);
```

DChargedTrackHypothesis

```
//Timing
double t0(void) const{return dTimingInfo->dt0;}
double t0_err(void) const{return dTimingInfo->dt0_err;}
double t1(void) const;
double t1_err(void) const;
DetectorSystem_t t0_detector(void) const{return dTimingInfo->dt0_detector;}
DetectorSystem_t t1_detector(void) const;
double Get_TimeAtPOCAToVertex(void) const{return dTimingInfo->dTimeAtPOCAToVertex;}
unsigned int Get_NDF_Timing(void) const{return dTimingInfo->dNDF_Timing;}
double Get_ChiSq_Timing(void) const{return dTimingInfo->dChiSq_Timing;}
double Get_PathLength(void) const;
double measuredBeta(void) const{return Get_PathLength()/(29.9792458*(t1() - t0()));}

//totals for overall PID determination
unsigned int Get_NDF(void) const{return dTimingInfo->dNDF;}
double Get_ChiSq(void) const{return dTimingInfo->dChiSq;}
double Get_FOM(void) const{return dTimingInfo->dFOM;}


//Tracking
void Set_TrackTimeBased(const DTrackTimeBased* locTrackTimeBased
void Set_ChiSq_DCdEdx(double locChiSq, unsigned int locNDF);
```

We are currently using this

Or this

We could also use this

Or this (FOM and ChiSq are the same)

DChargedTrackHypothesis

```
//Timing
double t0(void) const{return dTimingInfo->dt0;}
double t0_err(void) const{return dTimingInfo->dt0_err;}
double t1(void) const;
double t1_err(void) const;
DetectorSystem_t t0_detector(void) const{return dTimingInfo->dt0_detector;}
DetectorSystem_t t1_detector(void) const;
double Get_TimeAtPOCAToVertex(void) const{return dTimingInfo->dTimeAtPOCAToVertex;}
unsigned int Get_NDF_Timing(void) const{return dTimingInfo->dNDF_Timing;}
double Get_ChiSq_Timing(void) const{return dTimingInfo->dChiSq_Timing;}
double Get_PathLength(void) const;
double measuredBeta(void) const{return Get_PathLength()/(29.9792458*(t1() - t0()));}

//totals for overall PID determination
unsigned int Get_NDF(void) const{return dTimingInfo->dNDF;}
double Get_ChiSq(void) const{return dTimingInfo->dChiSq;}
double Get_FOM(void) const{return dTimingInfo->dFOM;}


//Tracking
void Set_TrackTimeBased(const DTrackTimeBased* locTrackTimeBased
void Set_ChiSq_DCdEdx(double locChiSq, unsigned int locNDF);
```

We are currently using this

Or this

We could also use this

Or this (FOM and ChiSq are the same)

Why can’t we Get() this?

dE/dx ChiSq

- Not currently implemented:

```
virtual jerror_t CalcDCdEdxChiSq(DChargedTrackHypothesis
*locChargedTrackHypothesis) const = 0;
```

- Example in DParticleID_PID1
  - Calculation of expected dE/dx exists
  - Needs parameterization of dE/dx sigmas

DParticleID::Calc_ChargedPIDFOM()

```
void DParticleID::Calc_ChargedPIDFOM(DChargedTrackHypothesis* locChargedTrackHypothesis) const
{
	CalcDCdEdxChiSq(locChargedTrackHypothesis);

	unsigned int locTimingNDF = 0;
	double locTimingPull = 0.0;
	double locTimingChiSq = Calc_TimingChiSq(locChargedTrackHypothesis, locTimingNDF, locTimingPull);
	locChargedTrackHypothesis->Set_ChiSq_Timing(locTimingChiSq, locTimingNDF);

	unsigned int locNDF_Total = locChargedTrackHypothesis->Get_NDF_Timing() + locChargedTrackHypothesis->Get_NDF_DCdEdx();
	double locChiSq_Total = locChargedTrackHypothesis->Get_ChiSq_Timing() + locChargedTrackHypothesis->Get_ChiSq_DCdEdx();
	double locFOM = (locNDF_Total > 0) ? TMath::Prob(locChiSq_Total, locNDF_Total) : numeric_limits<double>::quiet_NaN();
	locChargedTrackHypothesis->Set_ChiSq_Overall(locChiSq_Total, locNDF_Total, locFOM);
}
```

Timing ChiSq

Probability from ChiSq

DParticleID::Calc_TimingChiSq()

```
double DParticleID::Calc_TimingChiSq(const DChargedTrackHypothesis* locChargedHypo, unsigned int &locNDF, double& locPull) con
{
	if((locChargedHypo->t0_detector() == SYS_NULL) || (locChargedHypo->t1_detector() == SYS_NULL))
	{
		// not matched to any hits
		locNDF = 0;
		locPull = 0.0;
		return 0.0;
	}

	double locStartTimeError = locChargedHypo->t0_err();
	double locTimeDifferenceVariance = (*locChargedHypo->errorMatrix())(6, 6) + locStartTimeError*locStartTimeError;
	locPull = (locChargedHypo->t0() - locChargedHypo->Get_TimeAtPOCAToVertex())/sqrt(locTimeDifferenceVariance);
	locNDF = 1;
	return locPull*locPull;
}
```

DChargedTrackHypothesis_factory ::Create_ChargedTrackHypothesis()

- Currently using approximations added to the charged particle flight time variance
  - σ(SC) = 300 ps
  - σ(TOF) = 100 ps
  - σ(BCAL) = 300 ps
  - σ(FCAL) = 500 ps
- Could use uncertainties of individual hits, if they are correct!
  - Also correlation between momentum and flight time
- T0 from SC (if available), same fixed resolution

Other places to modify

- Hit time uncertainties also used in DParticleID::Get_ClosestToTrack()
- Start time uncertainty in DTrackTimeBased_factory::CreateStartTimeList() ?
- Other CCDB constants?

Path Forward

- Need to measure variable resolutions as a function of track momentum and angle
  - The rest of the framework is mostly there
  - And also make sure that simulation agrees
- Can then use ChiSq variables (independently or combined) to determine standard PID cuts
  - Also need to define event samples to study these
  - Clean method to incorporate DIRC likelihoods