CLAS NOTE 2007-008

Brief User Guide to GPID

Eugene Pasyuk
Arizona State University

May 1, 2007

Overview:

- What is GPID and where to get it
- Where is input information for GPID coming from
- Structure of GPID bank
- Structure of `GPID.map`
- Calibration procedure and `gpid_mon` utility
- Usage hints, limitations, things to do.

- GPID is extension of PART/TBID for photon runs with the Start Counter. It tries to do particle ID on a track by track basis. The method uses the momentum of detected particle, and sequentially calculates trial values of β for the particle for all possible particle identities. Each one of the possible identities is tested by the trial value β for a given particle type to the empirically measured value of β (as determined by CLAS tracking and time-of-flight information) The particle is assigned the identity that provides the closest trial value of β to the empirically measured value of β. Th GPID algorithm also attempts to find a matching photon in the tagging system for every charged particle detected in CLAS.

- The source is in clas CVS repository: `packages/pid/make_gpid.c`
  The source code has many comments in it. They explain how and what it is doing.

- List of banks required to build GPID bank: PART, TBID, TBTR, TDPL, SCRC, STR, TAGR

- Map file used: `GPID.map` (or GPID system from caldb)

- Functions:
  `int initGPID(int run)`
  reads cuts from `GPID.map` for run number `run`

  `clasGPID_t *makeGPID(int bankNum, int calib)`
  makes GPID bank. `bankNum` is a bank number to make. It also directs which PART/TBID banks to use. Usually it is 0 during cooking and 1 when you rebuild BID banks. If `calib=1`, `makeGPID` runs in calibration mode with cuts wide open. For normal running `calib=0`

Table 1: Structure of GPID bank

int	pid	Particle id (GEANT)
vector3_t	vert	track origin (x, y, z) from TBTR
vector4_t	p	particle four-momentum $(E, \vec{p})$
int	q	Charge
int	trkid	Index to TBID bank, counting from 1
int	sec	Sector track is in
int	paddle	SC paddle number
float	dedx	Energy deposited in TOF
float	beta	$\beta = \|\vec{p}\|/E$ with nominal mas of identified particle
int	sc_stat	Status of hit matching to SC sc_stat = 0: no SC for this track
float	sc_time	SC calibrated time for this track (ns)
float	sc_len	Track length from origin to SC (cm)
int	st_stat	Status of matching to ST st_stat = 0: no ST for this track
float	st_time	ST calibrated time for this track (ns)
float	st_len	Track length from origin to ST (cm)
float	mass	Particle mass using β from TOF (betam)
int	mass_ref	0: SC&TAG used 1: SC&ST used -1: neutral or no SC 2: from PART
float	betam	β from TOF 0, 2: $\beta = \frac{sc\_len}{c(sc\_time - tpho - tprop)}$ 1: $\beta = \frac{sc\_len - st\_len}{c(sc\_time - st\_time)}$
float	epho	Photon energy (GeV), 0 if not found
float	tpho	RF corrected photon time (RF time in the center of CLAS), 0 if not found
int	tagrid	Index to TAGR bank, counting from 1, 0 if not found
int	ngrf	Number of photons in the same RF bucket
int	ppid	Particle id as seen in PART bank

Structure of GPID.map

```
Map: GPID.map
        Subsystem: deuteron,    nitems: 3
                Item: dbeta,    length: 1,      type: float,    narray:1
                Item: high,     length: 50,     type: float,    narray:1
                Item: low,      length: 50,     type: float,    narray:1
        Subsystem: kaon,        nitems: 3
                Item: dbeta,    length: 1,      type: float,    narray:1
                Item: high,     length: 50,     type: float,    narray:2
                Item: low,      length: 50,     type: float,    narray:2
        Subsystem: kpi, nitems: 2
                Item: offset,   length: 1,      type: float,    narray:1
                Item: slope,    length: 1,      type: float,    narray:1
        Subsystem: pion,        nitems: 3
                Item: dbeta,    length: 1,      type: float,    narray:1
                Item: high,     length: 50,     type: float,    narray:2
                Item: low,      length: 50,     type: float,    narray:2
        Subsystem: proton,      nitems: 3
                Item: dbeta,    length: 1,      type: float,    narray:1
                Item: high,     length: 50,     type: float,    narray:2
                Item: low,      length: 50,     type: float,    narray:2
        Subsystem: triton,      nitems: 3
                Item: dbeta,    length: 1,      type: float,    narray:1
                Item: high,     length: 50,     type: float,    narray:1
                Item: low,      length: 50,     type: float,    narray:1
```

gpid_mon utility

Location: packages/utilities/gpid_mon/

```
asu.jlab.org{pasyuk}: gpid_mon -h

Usage: gpid_mon [options] file1 [file2] etc....

  Options:
        -o<outfile>       output hbook file (default=gpid_monXXXXX.hbook)
        -M<#>             Process only # number of events
        -R                Regenerate the TBID/PART and associated banks
        -c                Run in calibration mode
        -T<#>             Set trigger bit mask (default=0xffff)
        -h                Print this message.
```

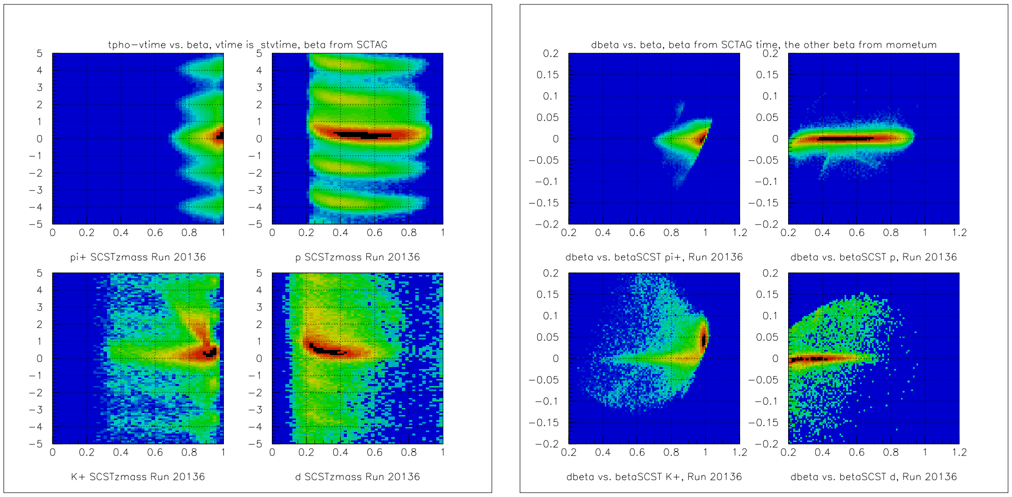

Figure 1: Vertex time difference vs. `betam` (left). `betam-beta` vs. `betam` (right)

There are kumac files in the same directory.

```
mpr_deuteron.kumac
mpr_kaon.kumac
mpr_pion.kumac
mpr_proton.kumac
```

These macros slice appropriate histograms of vertex time difference vs. beta into 50 slices. In each beta slice one should choose appropriate cuts (`low/high`) around central peak. Use your judgment. Usually it is about 0.8 – 1.5 ns from the peak. Don't cut too tight.

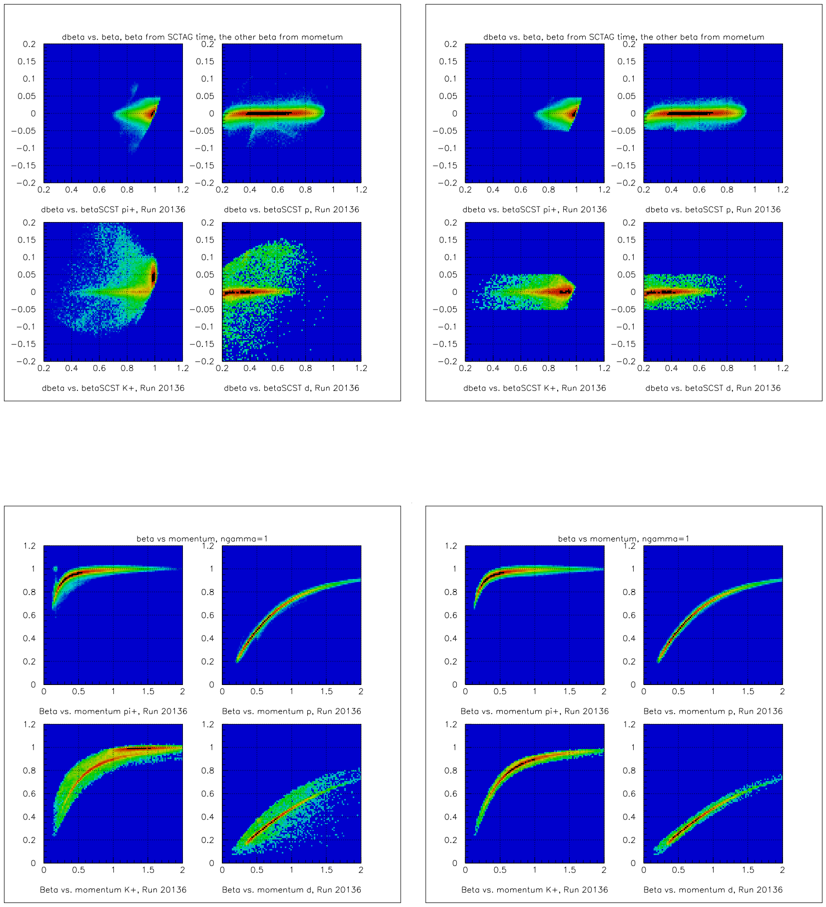

Figure 2: Left is before the cuts applied. Right, after the cuts.

After calibration you will see something like this. `dbeta` cut inverts the sign of `pid` if particle fails it. At the moment this feature is commented out.

Basic hints on usage

- *There is no universal recipe, each analysis is unique*
- Use particles with `ngrf>0`.
- Ignore particles with `ngrf=0` as if they were not detected at all. Do not include them in number of particles if it is one of your event selection criteria
- `ngrf=1` is unambiguous. `ngrf>1` requires special treatment. You have two choices: either throw away this event and account for this in inefficiency, or use means other than timing to choose between photons (kinematical cuts)
- As usual, be careful when selecting kaons. For skimming kaons do not rely on `pid` that comes out from GPID alone. Use a cut on the `mass` too. Something like this:

```
if (abs(GPID->gpid[j].pid)  == KPlus ||
     (GPID->gpid[j].mass <= 0.7 &&
      GPID->gpid[j].mass >= 0.3 &&
      GPID->gpid[j].q >0))
   Kp_found++;
```

- For multi track events it is possible that GPID associates particles with the same photon, but they are coming from different interactions (accidental coincidence). A comparison of their vertex times and z-components of vertex often helps.

Status, Limitations and Things to Do

- It is working
- GPID is included in `a1c` and `gflux`
- Neutral particles identification is not done in GPID. It has just a copy from PART/TBID. This is general problem with all PID packages used in CLAS.