TPC Hit Digitization at Cornell
Dan Peterson
03-Dec-2004

This note describes the hit digitization process used by the Cornell analysis of TPC detector response. The digitization starts with the LCD simulation, which provides a list of 3-d space points corresponding to the track crossings with coaxial cylinders. The digitization is in FORTRAN and is stored in the CLEO library system.

# FORTRAN files

Below is a list of the FORTRAN files used for signal digitization. Links to the listings of these routines may be found on the web page:
http://w4.lns.cornell.edu/~dpp/linear_collider/lcd_doit_tracking_info.html .

from directory ~/doit/hitlist/

tpc_add_hit.F
tpc_pad_response.F
tpc_ionization_centers.F
tpc_hitlist_xcheck.F
tpc_time_response.F obsolete
tpc_random_noise.F
tpc_hitlist.F
tpc_FADC_response.F
tpc_FADC_cluster.F

from directory ~/doit/initialize/
tpc_init_sequence.F
tpc_init_from_lcd.F
tpc_init_cdgeom.F
tpc_init_det_response.F

Below is a list of "include" files which define the parameters, variables, and common blocks used in the digitization. Links to the listings may also be found on the web page described above.

from directory ~/doit/duseq/
cdscrtcd.inc
tpccom.inc

from directory ~ /src/cl3seq/cdgm3/
cdgeompa.inc ( required only to define the geometry array size: kwircd)
cdgeomcd.inc

# Geometry

The first step is to set up a geometry. This is done within the structure of the CLEO central detector geometry, "cdgeom". All geometry initialization is accomplished with through tpc_init_sequence. The structure of subsequent calls is:

```
tpc_init_sequence
        tpc_init_cdgeom
                tpc_init_det_response
                        tpc_init_from_lcd.
```

I will describe the function of each of these routines starting from the deepest call.

**tpc_init_from_lcd**

This extracts some LCD geometry values from the LCD file that was provided to me. I don't think it is necessary for me to describe how these values were extracted from the LCD file.

Returned variables:

ret_bfield……the longitudinal magnetic field
ret_IR…….inner radius in meters
ret_or……..outer radius in meters
ret_haflen…….....TPC half length in meters,
ret_nlay……..number of cylinders in LCD

These values are returned to tpc_init_det_response. In looking at the code, I see that ret_nlay is not filled in tpc_init_from_lcd. It does not matter, the value will be overwritten in tpc_init_det_response.

**tpc_init_det_response**

This fills many constants (*Constants are those things that can vary*.) used in the digitization as well as the pattern recognition. For now, I will only describe the constants which are required for digitization.

First, tpc_init_det_response calls tpc_init_from_lcd. Then, values for ret_IR, ret_or, and ret_nlay are overwritten because the values, as extracted from the LCD file, are not correct.

Below is a list of constants that are used in digitization and are filled in tpc_init_det_response. This includes both geometry and some electronics characteristics. Expanded descriptions for some of these can be found in the FORTRAN listing.

TPC_IR.....inner radius in meters
TPC_OR.....outer radius in meters
TPC_haflen.....half length of the TPC in meters
TPC_cyl_crossings.....number of crossing cylinders in the LCD simulation

TPC_nominal_cell_wid..nominal pad width in meters (The actual pad width will
be adjusted slightly to provide an integer number of pads.)
TPC_ncell_devis.....number of pads in each layer will be divisible by this value
TPC_cell_z_width.....z cell width ( I am using an essentially infinite z width for
one sided readout.)
(TPC_TIME_INHIBIT is not used)
TPC_TIME_AMP.....amplifier shaping time in picosec (falling edge)
TPC_SIGNAL_SIGMA_PADS….sigma of gaussian charge spread on pads
(in units of pad size)
TPC_CUTOFF_TOT_PADS.....cut off of charge spread based on number of pads
TPC_CUTOFF_FRAC_HT.....cut off of charge spread based on pulse height
TPC_min_ion_PH.....pulse height, for the ionization center hit
TPC_randomH_area.....area per random noise hit: width*length /occupancy
TPC_randomH_Zwidth.....length (width in Z) of a random hit, in meters
TPC_randomH_maxPH.....maximum pulse height of random hits
TPC_DRIFT_VEL.....drift velocity in meters/ps
TPC_FADC_BIN_TIME.....period of the FADC in ps
TPC_FADC_LENGTH_XTR.....FADC extra length, in meters,
TPC_EXPON_TAIL_CUT.....pulse height value in exponential of pulse to
cut off adding ph to FADC time bin
TPC_TIME_AMP_BINS.....amplifier shaping time converted to FADC bins
TPC_FADC_SCALE_SINGLE_CELL_MAX .....
the maximum PH in the FADC from …..
( I think this is only used for scaling graphics)

The following are used to extract threshold crossings from the FADC.
TPC_FADC_NOISE_TO_SIGNAL.. noise-to-signal ratio, electronic noise
TPC_FADC_PED_I.....pedestal for initiating a hit in the FADC bins
TPC_FADC_PED_C.....pedestal for continuing a hit in the FADC bins
TPC_FADC_PED_FRAC.....upper limit of dynamic pedestal,
TPC_FADC_N_SMOOTH.....number of FADC bins to average

**tpc_init_cdgeom**

This fills the cdgeom geometry variables, defined in cdgeomcd.inc, required for the digitization. Many other variables filled in this routine are not required for the digitization. Rather, they are used to describe attributes of the layer for the pattern reconstruction. I will describe only the relevant variables.

NLYRCD……...number of readout layers
Note: the number of crossing layers, TPC_cyl_crossings, differs
from NLYRCD in that the cylinder crossings are used as the
entry and exit radii of the readout layers.
Thus NLYRCD = TPC_cyl_crossings - 1
RCD(layer)…….central radius of layer
NWIRCD(layer)…….number of azimuthal divisions in the layer

Note: the code used to calculate this creates a number of
cells divisible by TPC_ncell_devis
CELLCD(layer)……..width of cell; $2\pi$/ NWIRCD(layer)

PHIFCD(layer)……azimuthal angle of the first cell in the layer
CELRCD(layer)……..radial height of a cell
ZENDCD(layer)……..ABS of extent of active volume
in longitudinal direction (chamber half length)
CELZCD(layer)……..length of cell in longitudinal direction
(I have been using a one-sided readout.
For a 2-sided readout, CELZCD would be
the chamber half length.)
INDXCD(layer)……..Cell number of the first cell in the layer
in a global numbering of all the cells in the
chamber.

# TPC Hits

**cdscrtcd.inc**

TPC hits are defined by the variables in the common blocks in cdscrtcd.inc and tcccom.inc. The include file, cdscrtcd.inc, defines the variables which are common to the CLEO tracking reconstruction. There are multiple names for some of the variables; "equivalence" statements are used to tie the names to the same variable. A straightforward list of the names can be found in the cdscrtcd.inc listing under the heading "NEW VARIABLE NAMES".

Almost all of these variables are used in the pattern recognition. However, it is not clear to me how many of these variables should be provided directly by LCD rather than being defined by a reconstruction program that reads the LCD file.

**tpccom.inc**

The include file, tpccom.inc, defines all the variables initialized in tpc_init_det_response as described above. In addition, the FADC is defined here.

**the FADC**

One channel of simulated FADC, defined in tpccom.inc, is used to combine the pulse height signals in each cell that overlap in time. Only one channel is define because it would require excessive storage to define a non-sparsified FADC for each pad. This combines signals from both track related ionization and random ionization signals that have been added to the channel. As described under "tpc_FADC_cluster" it is also possible to combine electronic noise. These are attributes of the FADC:

FADC_PH.....pulse height values in FADC
FADC_PH_SUM.....sum of the last TPC_FADC_N_SMOOTH pulse height values for smoothing
FADC_PH_CLUS_SUM.....sum of the pulse height values since start of cluster
FADC_PH_PED_SUM.....sum of the pulse height values contributing to pedestal
The other variables associated with the FADC, and defined in tpccom.inc, are for diagnostic purposes.

# Hit Digitization

The parent routine is tpc_hitlist. All other routines are called from the parent with this structure.

- tpc_hitlist
  - tpc_hitlist_xcheck
  - tpc_inonization_centers
    - tpc_pad_response
    - tpc_add_hit
  - tpc_random_noise
    - tpc_add_hit
  - tpc_FADC_response
    - tpc_FADC_cluster
      - tpc_add_hit

**tpc_hitlist**

The purpose of this routine is to convert LCD cylinder crossings into segments of ionization. (The LCD cylinder crossings have been associated with the radial boundaries of cdgeom layers of TPC cells in tpc_init_cdgeom.) Within a loop over the LCD cylinder crossings, pairs of cylinder crossings are used to determine azimuthal, and longitudinal, entry and exit locations for ionization that will be created in the cdgeom layers of TPC cells. See **Figure 1**, linked from the web page.

This routine is quite long. However, the parts that are relevant to an LCD implementation are a small part. I will go through the routine in detail to separate the relevant parts from the diagnostics. First, lines within the compilation switch,
"#if defined(CLEO_SFDIAG)…#endif#", are for diagnostic purposes only.

The initialization sequence,

```
KILL=0
TAG_MATCH=1
EVENT_MATCH=1
SFZSLWLOVRD=-999
SFZSLWMOVRD=999,
```

is for diagnostics and control of the pattern recognition.

The initialization sequence,

```
HLIMIT=EvntHitMax
EvntHit_Num=0
```

is relevant. It sets up the hit overflow protection and zeroes the hit list.

The initialization sequence,

```
      EvntHit_OpnLoc1=1
      EvntHit_OpnLocL=EvntHitMax
      DO 53 I=1,EvntHitMax
        EvntHit_OpnLocN(I)=I+1
 53   CONTINUE
      EvntHit_OpnLocN(EvntHitMax)=0
```

is to set up a fragmented use of the hit arrays. It is not currently being used.

The initialization sequence,

```
      CLIMIT=KWS1CD
      CALL VZERO(EvntHit_MapDet,KWS1CD)
      LLIMIT=KLYRCD
      CALL VZERO(EvntHit_1stLyr,KLYRCD)
```

is relevant. It zeros certain parts of the hit structure.

The main loop for cylinder crossings starts with the lines

```
      trk_old=-2
        do 219 i=1,NTRKHITS
          TRK_NOW=trkhitmcpart(i)        +1
          IF(TRK_NOW.ne.trk_old)THEN .
```

The lines that immediately follow are mostly to recognize the outgoing track segments for the purpose of defining a list of tracks that should be found by the reconstruction, the “plausible track list”. These will not be relevant to the LCD implementation.

The use of trl_old and trk_now to determine if the input track number has changed is important. The routine, tpc_hitlist, sends layer entry and exit values, of the azimuthal angle, to the routine tpc_inonization_centers. The entry and exit values are calculated from the concentric cylinder crossings which define definine the cdgeom cell boundaries. Thus, it is necessary to distinguish between the first cylinder crossing of a new LCD track and subsequent cylinder crossings on the same track. This will become more clear in later discussion.

The code for a new track, relevant to an LCD implementation, is

```
            ICROSS_ENTR=-1
            ICROSS_2PREV=-1
            trk_old=TRK_NOW
          ENDIF
```

The above lines set the condition that there was no hit which could be used as an entry into this layer. In addition, there is no hit that is 2 hits back.

The next lines are used for processing all cylinder crossings.

```
          ICROSS=trkhitlayer(i)
          XCROSS=trkhit(1,i)/100.
          YCROSS=trkhit(2,i)/100.
          RAD_EXIT=SQRT(XCROSS**2+YCROSS**2)
          PHI_EXIT=ATN2PI(YCROSS,XCROSS)
          Z_EXIT=trkhit(3,i)/100.
```

These put the values into local variables, convert to meters, and define some other values, ie. radius and angle. Note that the radius and angle are labeled "exit". This means that it is the exit of a tpc cell layer; recall that the cylinder crossings are used as entries and exits of the tpc cell layers. It is an exit because the entry would have been defined by a previous cylinder crossing.

The next relevant block of lines checks for a match of the radius of the cylinder crossing with the cdgeom layer radius to which it points.

```
          VALID_TPC_CROSSING=.FALSE.
          CREATE_HITS_WITH_CROSSING=.FALSE.

          IF(TRK_NOW.GT.0)THEN
            IF(ICROSS.GT.0)THEN
              IF(abs(RAD_EXIT-(RCD(ICROSS)+CELRCD(1,ICROSS)/2.))
     1             .lt.0.005)VALID_TPC_CROSSING=.TRUE.
            ELSEIF(ICROSS.EQ.0)THEN
              IF(abs(RAD_EXIT-(RCD(1)-CELRCD(1,1)/2.))
     1             .lt.0.005)VALID_TPC_CROSSING=.TRUE.
            ENDIF
          ENDIF
```

The variables, valid_hit_crossing and create_hits_with_crossing, are used to control the creation of TPC hits. The value, rad_exit, is derived from the LCD track crossings. The cylinder crossing also provides an LCD layer number, assigned to icross. The above code matches the cylinder crossing to the to the geometry layer indexed by icross. If the cylinder crossing is zero, this is a crossing inside the first geometry later. If the layer match is satisfied, the logical, valid_tpc_crossing, is true. However, the other logical, create_hits_with_crossing, is not set until the next section.

The next relevant block of lines is used to distinguish between four distinct situations of a track crossing through a tpc layer: outgoing, incoming, top of curler, and bottom of curler. These are illustrated in the **Figure 2**, linked from the web page. The code distinguishes between these situations based on the relation of the present crossing layer and the 2 previous crossing layers. For valid situations, the cdgeom layer number, ilcd, is assigned and create_hits_with_crossing is set true.

In the next relevant block of lines, and in the case that create_hits_with_crossing is set true, tps hits will be created by a call to tpc_ionization_centers.

```
            IF(CREATE_HITS_WITH_CROSSING)THEN
                CALL TPC_IONIZATION_CENTERS(ILCD,
     2               PHI_EXIT,PHI_ENTR,Z_EXIT,Z_ENTR,TRK_NOW)
```

Following these lines, there are several lines for the purpose of recognizing outgoing track segments that are not relevant to the LCD implementation.
The "IF" is closed by

```
C-----------------------------------------------------------------------
C end of condition CREATE_HITS_WITH_CROSSING
C-----------------------------------------------------------------------
            ENDIF  .
```

This is followed by the set-up for the next pass of the loop

```
            ICROSS_2PREV=ICROSS_ENTR
            ICROSS_ENTR=ICROSS
            Z_ENTR=Z_EXIT
            RAD_ENTR=RAD_EXIT
            PHI_ENTR=PHI_EXIT
```

or the treatment for the condition that a bad hit, or LCD track number change, was encountered,

```
          ELSE
            ICROSS_2PREV=-1
            ICROSS_ENTR=-1
          ENDIF
```

and the end of the loop.

```
219    continue
```

After processing all hits, there are calls to tpc_ramdom_hits and tpc_FADC_response. The remaining code is diagnostic or related to defining the plausible track list.

**tpc_ionization_centers**

This routine is called from tpc_hitlist. It creates the ionization centers, which are locations in the layer where the pad response function is used to create hits on the pads. This process allows for multiple ionization centers when the azimuthal extent on the track crossing in a geometry layer is wide. Thus, large entrance angle and tops of curlers create pulses on several pads. Inputs are
geometry layer,
azimuthal entry and exit,
longitudinal entry and exit,
and a "tag" which is used to pass the current LCD track number.

The values phi_strt and phi_stop are derived from the inputs phi_entr and phi exit. The new variables have the property that $0 < (\text{phi\_stop} - \text{phi\_strt}) < \pi$.