Technote 90
Event Classification Documentation in TWIST

Blair Jamieson

University of British Columbia

April 1, 2004

1 Introduction

This note is meant to summarize the current method used to do event classification with TWIST analysis package (MOFIA). This note summarizes the algorithm used for sorting TDC hits into the MOFIA "time" window structure. The algorithm can be broken up into the following steps, in the order in which they are done:

- Calculate TDC hit statistics in bins of time
- Set window times
- Look for outlier noise hits at large radius
- Calculate TDC hit statistics in window times
- Calculate start and stop planes and try to tag possible cosmic ray
- Try to match scintillator hits to windows
- Assign hits to the window structure
  - Evaluate window types one at a time
  - Review window types looking at whole event
  - Assign hits to window
  - Optionally try to extract beam positron PC overlap tracks
- Do Clustering (not discussed here)
- Decide on overall event type

Most of the items above will be discussed in their own section in the remaining portion of this document. Note that this document describes what is meant to be done in the event classification, and does not address how well it actually does this.

2 Calculate TDC Hit Statistics in Bins of Time

The first step is to calculate several different statistics on TDC hits, binned in time. The size of the time bin was chosen to be 50ns, so there are a total of 321 timebins from -6000ns to 10000ns. The statistics calculated are stored in the structures PCt(timebin) and DCt(timebin). The structure for the DC timebins for example is:

```
DCt(timebin)%hitcount    ! Count of Plane hit times binned in time
DCt(timebin)%uhitcount   ! Count of Plane hit times binned in time
DCt(timebin)%vhitcount   ! Count of Plane hit times binned in time
DCt(timebin)%timemin     ! Minimum of plane hit times binned in time
DCt(timebin)%timemax     ! Maximum of plane hit times binned in time
DCt(timebin)%timesum     ! Sum of plane hit times binned in time
DCt(timebin)%timesum2    ! Sum of plane hit times binned in time
DCt(timebin)%timeavg     ! Average of Plane hit times binned in time
DCt(timebin)%timesig     ! SIG of Plane hit times binned in time
DCt(timebin)%widthsums   ! Sum of Plane widths binned in time
DCt(timebin)%widthavg    ! Average Plane width binned in time
DCt(timebin)%first       ! First Plane number hit binned in time
DCt(timebin)%last        ! Last Plane number hit binned in time
DCt(timebin)%usum        ! Sum of U coordinates binned in time
DCt(timebin)%usum2       ! Sum sqared of U coordinates binned in time
DCt(timebin)%uavg        ! Average of U coordinates binned in time
DCt(timebin)%umax        ! Maximum U coordinate binned in time
DCt(timebin)%umin        ! Minimum U coordinate binned in time
DCt(timebin)%usig        ! SIG of U coordinates binned in time
DCt(timebin)%vsum        ! Sum of V coordinates binned in time
DCt(timebin)%vsum2       ! Sum sqared of V coordinates binned in time
DCt(timebin)%vavg        ! Average of V coordinates binned in time
DCt(timebin)%vmax        ! Maximum V coordinate binned in time
DCt(timebin)%vmin        ! Minimum V coordinate binned in time
DCt(timebin)%vsig        ! SIG of V coordinates binned in time
```

3 Set window times

This is a step that needed to be revised a few times between January and June of 2003. It hasn't changed since then, so it must be mostly right now. The basic idea is to separate track hits in time. This is done by looking for peaks in both the PC and DC timebin hit count statistics. Most of the time only PC timebin statistics are used, but occasionally DC statistics are used.

The algorithm is to usually ignore lonely PC hits, unless there are lots of DC hits at the same time, and there has not been any other particle for DCOverlapTime (windowbj namelist default value=1050ns).

The exact algorithm is in the subroutine CalcPCPidTimes in windowbj_mod.f90, and here I try to describe how it is done. We loop over the time bins, and decide if a given time bin should make a new track, or be added to an existing track. This is done in two steps.

For step 1 if there is more than one PC hit in the timebin then we look to see if this is a continuation of an already started track. It is a continuation if the first and last PC hit in this timebin are the same and have TDC width less than 40ns, or if there are fewer than four hits in this PC timebin with width less than 40ns and within 8 timebins of the last PC timebin. Otherwise if it has more than 3 DC hits, or more than 3 PC hits, or more than 3 DC hits in each of the next three timebins then we go on to step two. If there is one PC hit in the timebin, then we look to see if the second step has told us the next timebin with hits will be a new track and that there are more than 10 DC hits in this timebin with width > XTDCWidthCut (40ns) to send the time bin on to step two. A check is made to not make a new window if this timebin's minimum PC time is within PCOverlapTime (200ns) of the first for this track. Another check is made to see if this timebin's minimum PC time is later than MaxWindowLength (1500ns) of the first for this track.

For step 2 there are two cases: the first is if we already think this is a new track, and we start a new window. The second case is more complex, and it is here that we try to decide if this is a continuation of the previous track or a new one. Essentially the algorithm looks to see if a peak in the hitcount has been reached. Once a peak is reached, then we don't start a new window unless there is another peak more than Win_TBinsGapNewWin (3 time bins) away.

In this step the number of PC windows found (npcpid) is incremented, and information about the tracks found are stored in the structure:

```
PCpid(ipc)%Nbins             ! Number of tbins for this particle "window"
PCpid(ipc)%Nhits             ! Number of PC hits for this particle
PCpid(ipc)%Tidx(MAX_TBINS)  ! Index into PCt of tbins for this particle
PCpid(ipc)%Tmin              ! Minimum PC time
PCpid(ipc)%Tmax              ! Maximum PC time
PCpid(ipc)%Tstart            ! Particle start time
PCpid(ipc)%Tend              ! Particle end time
PCpid(ipc)%TNsum             ! Time bin avg time * Number of hits
PCpid(ipc)%TNsum2            ! Time bin avg time * Number of hits squared
PCpid(ipc)%Tavg              ! Average particle time
PCpid(ipc)%Tsig              ! Sigma of pc hits for particle
PCpid(ipc)%Npeaks            ! Number of peaks in PC hits spectrum
PCpid(ipc)%overlap           ! Does this look like an overlap of 2 or more particles
PCpid(ipc)%dconly            ! Pacticle track started with only DC hits
```

Note: For each of these PC windows, two entries in the ordinary window structure will be made. The even window and odd window before it will have the same start and end times, and are meant to try to separate hits in space. The even window will contain the decay positron hits, and the odd window before it the hits that are in time coincidence with the decay positron, which are obviously not part of the decay positron track.

4 Look for outlier noise hits at large radius

The idea is to try to ignore a few outlier hits that are more than 5cm in u,v from the bulk of the other hits. The decision to make this cut is done separately for the upstream and downstream halves of the detector. If the cut is made, then the following variables define the cut in cm:

```
win(iwin)%Up%rcut
win(iwin)%Dn%rcut
```

5 Calculate TDC hit statistics in window times

Here we loop through the TDC hits and calculate some average quantities to help later in deciding what is contained in each window. Note that the win structure is binned on the number of different PC windows, which is half the number of windows in the old window structure. The statistics are stored in the following structure of variables:

```
win(iwin)%PC                         ! statistics on PCs
            %hitcount  ! Count of Plane hit times binned in PC win
            %uhitcount ! Count of Plane hit times binned in PC win
            %vhitcount ! Count of Plane hit times binned in PC win
            %timemin   ! Minimum of plane hit times binned in PC win
            %timemax   ! Maximum of plane hit times binned in PC win
            %timesum   ! Sum of Plane hit times binned in PC win
            %timesum2  ! Sum of Plane hit times binned in PC win
            %timeavg   ! Average of Plane hit times binned in PC win
            %timesig   ! SIG of Plane hit times binned in PC win
            %widthsums ! Sum of Plane widths binned in PC win
            %widthavg  ! Average Plane width binned in PC win
            %first     ! First Plane number hit binned in PC win
            %last      ! Last Plane number hit binned in PC win
            %usum      ! Sum of U coordinates binned in PC win
            %usum2     ! Sum sqared of U coordinates binned in PC win
            %uavg      ! Average of U coordinates binned in PC win
            %umax      ! Maximum U coordinate binned in PC win
            %umin      ! Minimum U coordinate binned in PC win
            %usig      ! SIG of U coordinates binned in PC win
            %vsum      ! Sum of V coordinates binned in PC win
            %vsum2     ! Sum sqared of V coordinates binned in PC win
            %vavg      ! Average of V coordinates binned in PC win
            %vmax      ! Maximum V coordinate binned in PC win
```

```
          %vmin      ! Minimum V coordinate binned in PC win
          %vsig      ! SIG of V coordinates binned in PC win
          %ravg      ! Average of coordinates binned in PC win
          %rsig      ! Spread of coordinates

win(iwin)%DC                         ! statistics on DCs
          %hitcount  ! Count of Plane hit times binned in PC win
          %uhitcount ! Count of Plane hit times binned in PC win
          %vhitcount ! Count of Plane hit times binned in PC win
          %timemin   ! Minimum of plane hit times binned in PC win
          %timemax   ! Maximum of plane hit times binned in PC win
          %timesum   ! Sum of Plane hit times binned in PC win
          %timesum2  ! Sum of Plane hit times binned in PC win
          %timeavg   ! Average of Plane hit times binned in PC win
          %timesig   ! SIG of Plane hit times binned in PC win
          %widthsums ! Sum of Plane widths binned in PC win
          %widthavg  ! Average Plane width binned in PC win
          %first     ! First Plane number hit binned in PC win
          %last      ! Last Plane number hit binned in PC win
          %usum      ! Sum of U coordinates binned in PC win
          %usum2     ! Sum sqared of U coordinates binned in PC win
          %uavg      ! Average of U coordinates binned in PC win
          %umax      ! Maximum U coordinate binned in PC win
          %umin      ! Minimum U coordinate binned in PC win
          %usig      ! SIG of U coordinates binned in PC win
          %vsum      ! Sum of V coordinates binned in PC win
          %vsum2     ! Sum sqared of V coordinates binned in PC win
          %vavg      ! Average of V coordinates binned in PC win
          %vmax      ! Maximum V coordinate binned in PC win
          %vmin      ! Minimum V coordinate binned in PC win
          %vsig      ! SIG of V coordinates binned in PC win
          %ravg      ! Average of coordinates binned in PC win
          %rsig      ! Spread of coordinates

win(iwin)%Up                         ! statistics on US of target
          %Planes    ! Number of Planes Hit
          %HitsPP    ! Hits Per Plane
          %RHitsPP(6)! Restricted hits per plane
          %Ucount    ! Number of U hits
          %Vcount    ! Number of V hits
          %Usum         ! Sum of U coordinates
          %Usum2        ! Sum squared of U coordinates
          %Uavg         ! Average U coordinate
          %Usig         ! Sigma of U coordinates
          %Vsum         ! sum of V coordinates
```

```
            %Vsum2         ! Sum squared of V coordinates
            %Vavg          ! Average V coordinate
            %Vsig          ! Sigma of V coords
            %ravg          ! average coordinate
            %rsig          ! spread of coordinates
            %rwire(WIN_NRBINS) ! histogram of wire positions
            %rcut          ! radius at which to cut 'noise' hits
            %tsum          ! Sum of TDC times
            %tavg          ! Average TDC time

win(iwin)%Dn                          ! stats on DS of target
            %Planes    ! Number of Planes Hit
            %HitsPP    ! Hits Per Plane
            %RHitsPP(6)! Restricted hits per plane
            %Ucount    ! Number of U hits
            %Vcount    ! Number of V hits
            %Usum          ! Sum of U coordinates
            %Usum2         ! Sum squared of U coordinates
            %Uavg          ! Average U coordinate
            %Usig          ! Sigma of U coordinates
            %Vsum          ! sum of V coordinates
            %Vsum2         ! Sum squared of V coordinates
            %Vavg          ! Average V coordinate
            %Vsig          ! Sigma of V coords
            %ravg          ! average coordinate
            %rsig          ! spread of coordinates
            %rwire(WIN_NRBINS) ! histogram of wire positions
            %rcut          ! radius at which to cut 'noise' hits
            %tsum          ! Sum of TDC times
            %tavg          ! Average TDC time

win(iwin)%UpPC                        ! info from US PC 1-4
              %hitcount  ! Count of upstream PC hit times binned in PC win
              %tmin      ! Earliest PC time
              %tmax      ! Latest PC time
              %tsum      ! sum of PC times
              %tsum2     ! sum squared of PC times
              %tavg      ! average PC time
              %tsig      ! sigma of PC times
              %widthsum  ! Sum of upstream PC widths binned in PC win
              %widthsum2 ! Sum squared of upstream PC widths bin in PC win
              %widthavg  ! Average PC upstream width binned in PC win
              %widthsig  ! Sigma of PC upstream width binned in PC win
              %widthcount(WIN_NWIDBINS) ! count of hits in bin of width
              %hits(4)   ! hit count for each plane
```

```
            %wsum(4)   ! width sum for each plane
            %wavg(4)   ! width average for each plane

win(iwin)%DnPC                       ! info from DS PC 9-12
            %hitcount  ! Count of upstream PC hit times binned in PC win
            %tmin      ! Earliest PC time
            %tmax      ! Latest PC time
            %tsum      ! sum of PC times
            %tsum2     ! sum squared of PC times
            %tavg      ! average PC time
            %tsig      ! sigma of PC times
            %widthsum  ! Sum of upstream PC widths binned in PC win
            %widthsum2 ! Sum squared of upstream PC widths bin in PC win
            %widthavg  ! Average PC upstream width binned in PC win
            %widthsig  ! Sigma of PC upstream width binned in PC win
            %widthcount(WIN_NWIDBINS) ! count of hits in bin of width
            %hits(4)   ! hit count for each plane
            %wsum(4)   ! width sum for each plane
            %wavg(4)   ! width average for each plane

win(iwin)%TgtPC                      ! info from TGT PC 5-8
             %hitcount  ! Count of upstream PC hit times binned in PC win
             %tmin      ! Earliest PC time
             %tmax      ! Latest PC time
             %tsum      ! sum of PC times
             %tsum2     ! sum squared of PC times
             %tavg      ! average PC time
             %tsig      ! sigma of PC times
             %widthsum  ! Sum of upstream PC widths binned in PC win
             %widthsum2 ! Sum squared of upstream PC widths bin in PC win
             %widthavg  ! Average PC upstream width binned in PC win
             %widthsig  ! Sigma of PC upstream width binned in PC win
             %widthcount(WIN_NWIDBINS) ! count of hits in bin of width
             %hits(4)   ! hit count for each plane
             %wsum(4)   ! width sum for each plane
             %wavg(4)   ! width average for each plane

win(iwin)%plane                      ! start,stop plane info
             %hits(56)     ! number of hits in plane by window
             %cosmic(56)   ! do the hits in this plane seem like they could be a cosmi
             %stoptype     ! type of stopping plane (P or T or D)
             %stop         ! stopping plane number (global plane#)
             %starttype    ! type of starting plane (P or T or D)
             %start        ! starting plane number (global plane#)
             %ngaps        ! # of gaps of 2 or more planes missing
```

```
          %gapstart(20) ! start plane number for a gap of planes
          %gapstop(20)  ! stop plane number for a gap of planes
          %noiseorcosmic! does this window look like just noise?

win(iwin)%nNonZeroSlides             ! # of non-zero slides
win(iwin)%NonZeroSlides(WIN_NSLIDES) ! Array of non-zero slide indices
win(iwin)%slide(WIN_NSLIDES)         ! stats by slide
                        %uhitcount ! Count of U hits by window and slide
                        %vhitcount ! Count of V hits by window and slide
                        %widsum    ! Sum of Widths by window and slide
                        %widavg    ! Average Width by window and slide
                        %usum      ! Sum of U coords by window and slide
                        %usum2     ! Sum squared of U coords by window and slide
                        %uavg      ! Avg U coord by window and slide
                        %umax      ! Maximum U coord by window and slide
                        %umin      ! Minimum U coord by window and slide
                        %usig      ! Sigma of U coords by window and slide
                        %vsum      ! Sum of V coords by window and slide
                        %vsum2     ! Sum squared of V coords by window and slide
                        %vavg      ! Avg V coord by window and slide
                        %vmax      ! Maximum V coord by window and slide
                        %vmin      ! Minimum V coord by window and slide
                        %vsig      ! Sigma of V coords by window and slide
                        %ravg      ! Average hit coords by win and slide
                        %rsig      ! Spread of hit coords by win and slide

win(iwin)%hitmuscint                 ! was mu scint hit
win(iwin)%hitt0scint                 ! was t0 scint hit
win(iwin)%muscinttime                ! time of mu scint hit
win(iwin)%t0scinttime                ! time of t0 scint hit
win(iwin)%tof                        ! tof from PC info
win(iwin)%hitsPP                     ! all window hits per plane
win(iwin)%RhitsPP(6)                 ! restricted all window hits per plane
win(iwin)%toofewplaneshit            ! too few planes for tracking
win(iwin)%toomanyhitspp              ! too many hits for tracking
```

6 Assign hits to the window structure

In order to assign hits to the window structure, we want to get an idea of what the track or tracks are in the window. To do this we first evaluate window types, one at a time, then make corrections looking at the whole event. Then we can try to separate hits so that mostly decay positron hits are in an even window, and any hits that overlap in time are put in the previous odd window. The following subsections describe the window types, and hit assignment to windows.

6.1 Evaluate window types one at a time

A summary of the window types, and a short description of what the window type is meant to be is presented in the following subsections. This decision process is complex, and the descriptions given here are approximate. If you need to know more detail, this decision process is contained in evalwin_mod.f90. The *window types* are:

1. Muon
   Only a muon with possible deltas or hard scattering appears to be in this window.

2. Upstream Decay
   Only an upstream decay with all hits on upstream side of a muon decay appear in this window.

3. Downstream Decay
   Only a downstream decay with all hits on downstream side of a muon decay appear in this window.

4. Beam Positron
   Only a beam positron in this window.

5. Empty
   An odd window, which may contain hits which do not appear to be from a muon or decay positron track.

6. Overlap involved
   PC time between two tracks of more than 200ns, more than 1.5 hits per plane, no muons yet or in track.

7. Upstream Decay (DS delta removed)
   Upstream decay with hits downstream that are all at one u,v positron (within 1cm RMS).

8. Downstream Decay (US delta removed)
   Downstream decay with hits upstream that are all at one u,v positron (within 1cm RMS).

9. Upstream Decay (Scatter back downstream)
   Upstream Decay that has hits on both sides of muon stop. There needs to be more than 5 hits per plane upstream, more than 15 hits in a particular plane, or more than 80 hits in any group of 8 planes.

10. Downstream Decay (Scatter back upstream)
    Downstream Decay that has hits on both sides of muon stop. There needs to be more than 5 hits per plane downstream, more than 15 hits in a particular plane, or more than 80 hits in any group of 8 planes.

11. Muon and Fast Decay Downstream
    Appears to be a muon with a fast decay downstream within 200ns.

12. Muon and Fast Decay Upstream
    Appears to be a muon with a fast decay upstream within 200ns.

13. Muon and beam positron
    Appears to be a muon and a beam positron within 200ns.

14. Upstream Decay (DS part of beam e+ removed)
    Appears to be an upstream decay and a beam positron within 200 ns.

15. Downstream Decay (US part of beam e+ removed)
    Appears to be a downstream decay and a beam positron within 200 ns.

16. Soft uncorrelated track
    All u,v hits at one position with RMS less than 0.5cm, that does not have one end near a muon stop (more than 2 planes, or more than 5cm in u,v).

17. Cosmic, noise, or beam gas
    Appears to be just a cosmic ray, noise, or beam gas. Not enough planes hit or too far from any muon stops to be a muon or decay positron.

18. Too early
    Track has only DC hits with earliest time less than -5800ns, or PC hits earlier than -5800ns.

19. Too late
    Earliest window hit time is later than 9800ns.

20. Don't know
    Well I have some idea what these are. These are mostly decay positrons, with no muon that can be seen in the event. They could also be some leakage from other window types that for whatever reason had average hit distributions that just did not seem right.

21. Pion
    No events should get this window type since the default namelist parameters are set such. These would be windows that appear to have a pion (large m12width, specific TCAP time, stop before a particular DC plane).

The algorithm to determine the window types uses different decision processes if:

- the window is determined to be the trigger window (within 100ns of zero time.)
- the window is within (DCConfusionTime) 900ns of any other window, and
- there has or has not been a muon found yet.

6.2 Review window types looking at whole event

After looping through each window separately to decide on window types, a second pass is made to look for pathological cases, and correct the window types for these cases.

6.3 Assign hits to window

Using the already determined PC windows that have been tagged as being of a particular type, we fill the old time window structure with two windows for each PC window. For each PC window we put the hits that we think belong to the decay positron into the even window, and hits that do not belong to the muon or beam positron into the previous odd window.

6.4 Optionally try to extract beam positron PC overlap tracks

There is some experimental code to try to remove beam positrons by doing a helix fit to the side of the track with the beam positron. Some improvements need to be made here, since it tends to remove too many hits from the decay positron track. (namelist windowbj enableExtractBeame)

7 Decide on overall event type

This part is pretty easy since we just use information about what each window type is, and the time for each window to assign the overall type. If you want to look at the details of how this decision is made see classifybj_mod in the subroutine evalevent. The *event types* are:

1. Simple Clean
   Muon and decay positron, separated in time by more than 1050ns. No other windows in the event.

2. Time Clean
   Muon, decay, and beam Positron(s) each separated in time by more than 1050ns.

3. Simple DC Overlap
   Muon and decay positron separated in time by 200ns, but less than 1050 ns apart. No other windows in the event.

4. Time DC Overlap
   Muon, decay positron and beam positron(s) each separated in time by 200ns or more, but at least one time between windows is less than 1050ns.

5. PC Overlap
   There is at least one window which has two tracks within 200ns of each other.

6. Simple Delta Cleaned
   Muon and decay positron, separated in time by more than 1050ns. The decay positron has a delta which extends to the other side of the muon stop. No other windows in the event.

7. Time Delta Cleaned
   Muon, Decay positron and beam positron(s), each separated in time by more than 1050ns. The decay positron has a delta which extends to the other side of the muon stop.

8. Simple DC Overlap Delta Cleaned
   Muon and decay positron, separated in time by more than 200ns but less than 1050ns. The decay positron has a delta which extends to the other side of the muon stop. No other windows in the event.

9. Time DC Overlap Delta Cleaned
   Muon, Decay positron and beam positron(s), each separated in time by more than 200ns but at least one is separated in time by less than 1050ns. The decay positron has a delta which extends to the other side of the muon stop.

10. Simple Scatter Cleaned
    Muon and decay positron, separated in time by more than 1050ns. The decay positron has lots of hits per plane and hits on both sides of the muon stop. No other windows in the event.

11. Time Scatter Cleaned
    Muon, Decay positron and beam positron(s), each separated in time by more than 1050ns. The decay positron has lots of hits per plane and hits on both sides of the muon stop.

12. Simple DC Overlap Scatter Cleaned
    Muon and decay positron, separated in time by more than 200ns but less than 1050ns. The decay positron has lots of hits per plane and hits on both sides of the muon stop. No other windows in the event.

13. Time DC Overlap Scatter Cleaned
    Muon, Decay positron and beam positron(s), each separated in time by more than 200ns but at least one is separated in time by less than 1050ns. The decay positron has lots of hits per plane and hits on both sides of the muon stop. No other windows in the event.

14. Beam Positron Trigger
    Event appears to have been triggered by a beam positron. There can be any combination of other window types in the event.

15. Multiple Muon Decays Simple Clean
    Two or more muons in event. All windows separated by more than 1050ns. Only muons and decay positrons, where the order of the windows is muon decay muon ...

16. Multiple Muon Decays Time Clean
    Two or more muons in event. All windows separated by more than 1050ns. Only muons, decay positrons, and beam positrons in event. The order of the muons and decay positron(s) is muon decay muon ... The beam positron window(s) can come anywhere in the order of the windows.

17. Multiple Muon Decays DC Overlap Simple Clean
    Two or more muons in event. All windows separated by more than 200ns, at least one window separation is less than 1050ns. Only muons and decay positrons, where the order of the windows is muon decay muon ...

18. Multiple Muon Decays DC Overlap Time Clean
Two or more muons in event. All windows separated by more than 200ns, at least one window separation is less than 1050ns. Only muons, decay positrons, and beam positrons in event. The order of the muons and decay positron(s) is muon decay muon ... The beam positron window(s) can come anywhere in the order of the windows.

19. Multiple Muon Decays Dirty
Two or more muon decays in wrong order, or window within 200ns.

20. Just a Muon and Beam Positrons
There is one or more muons in the event and zero or more beam positrons. There does not appear to be any decay positron.

21. Simple Beam Positron Cleaned
Muon and decay positron, separated in time by more than 1050ns. The decay positron appears to be within 200ns of a beam positron. No other windows in the event.

22. Time Beam Positron Cleaned
Muon, decay positron, and beam positron(s), separated in time by more than 1050ns. The decay positron appears to be within 200ns of a beam positron.

23. Simple DC Overlap Beam Positron Cleaned Muon and decay positron, separated in time by more than 200ns but less than 1050ns. The decay positron appears to be within 200ns of a beam positron. No other windows in the event.

24. Time DC Overlap Beam Positron Cleaned
Muon, decay positron, and beam positron(s), separated in time by more than 200ns, but at least one of the windows is separated by less than 1050ns. The decay positron also appears to be within 200ns of a beam positron.

25. Just beam positrons, unknown trigger
Events where the trigger window does not contain either a muon or a beam positron, and the event does not contain any muon(s) or decay positron(s).

26. Other unknown trigger
Events where the trigger window does not contain either a muon or a beam positron. There can by any combination of other window types in the event.

27. Other
Whatever is left.

28. Simple Clean – too few planes hit for event type 1,6,10,15
Simple clean events with fewer than 6 planes hit so that no reasonable fit could be done.

29. Time Clean – too few planes hit for event type 2,7,11,16
Time clean events with fewer than 6 planes hit for the decay positron so that no reasonable fit could be done.

30. Simple Clean – Too high angle for event type 1,6,10,15
    Simple clean events, where the decay positron has an average cluster area greater than 12.5, and average hits per plane more than 3.5.

31. Time Clean – Too high angle) for Type 2,7,11,16
    Time clean events, where the decay positron has an average cluster area greater than 12.5, and average hits per plane more than 3.5.