The BESIII `FSFilter` Package (version 00-00-00)

Ryan Mitchell

October 31, 2011

Abstract

This document describes how to use the `FSFilter` package to simultaneously reconstruct multiple final states at BESIII. Final states can be reconstructed either exclusively or inclusively. The output of the `FSFilter` package is a set of root trees that can then be used as the basis for further analysis.

Contents

1 Purpose

`FSFilter` is a package that can be used to convert BESIII data into a format that can be more easily analyzed. `FSFilter` "Filters" information from many "Final States" into root trees. Any number of final states can be selected using input parameters in the job options file. The final states can be reconstructed either exclusively (in which case a kinematic fit to the initial four-momentum is performed) or inclusively (in which case a range of missing masses can be specified). Only very loose cuts are applied at this stage – it is assumed that most of the analysis will be done using the output root trees.

2 Overview

The `FSFilter` algorithm follows three basic steps:

1. Lists of particle types are made according to the loose selection criteria described in section 3. The particles considered are Λ, $\overline{\Lambda}$, e^+, e^-, μ^+, μ^-, p, $\overline{p}$, η (decaying to $\gamma\gamma$), γ, K^+, K^-, K_S^0 (decaying to $\pi^+\pi^-$), π^+, π^-, and π^0 (decaying to $\gamma\gamma$). Tracks and showers are allowed to appear in multiple lists.
2. The particle lists are combined to form a specified final state. All combinations are considered. Here, tracks and showers can only be used once per combination to avoid double-counting. If a final state is being exclusively reconstructed, a kinematic fit to the initial four-momentum is performed and a loose χ^2 cut is applied. For inclusive reconstruction, missing mass squared cuts are applied. This is described further in section 4.
3. Finally, the resulting information is written out to a series of root trees, described in section 8.

In the following documentation, the particle selection criteria (section 3) and event selection criteria (section 4) are first discussed. Then a number of details are given regarding the final state numbering scheme (section 5) and the job options files (sections 6 and 7). Finally, the contents of the output root trees are listed in section 8.

3 Particle Selection

The `FSFilter` algorithm begins by creating lists of final state particles according to the selection criteria discussed in this section.

3.1 Tracks ($e^{\pm}, \mu^{\pm}, \pi^{\pm}, K^{\pm}, p^{\pm}$)

Tracks are selected using a combination of the `RecMdcKalTrack` and `ParticleID` packages. No additional requirements are made on the track quality, momentum or angles beyond those made within the `RecMdcKalTrack` package. For particle identification, the `ParticleID` package is set up as follows:

```
ParticleID* pid = ParticleID::instance();
pid->init();
pid->setMethod(pid->methodProbability());
pid->setRecTrack(*iTrk);
pid->usePidSys(pid->useDedx()  |
               pid->useTof1()  |
               pid->useTof2());
pid->identify(pid->onlyPion()     |
              pid->onlyKaon()     |
              pid->onlyProton()   |
              pid->onlyElectron() |
              pid->onlyMuon());
pid->calculate();
```

Very loose cuts are then placed on the probabilities of different hypotheses. For a track to be counted as a $(e^{\pm}, \mu^{\pm}, \pi^{\pm}, K^{\pm}, p^{\pm})$ the probability of the $(e^{\pm}, \mu^{\pm}, \pi^{\pm}, K^{\pm}, p^{\pm})$ hypothesis must be greater than 10^{-5}. Note that tracks that satisfy more than one hypothesis are included in multiple lists. All possible combinations are considered when forming events, which is discussed in section 4.

3.2 Showers (γ)

Showers are reconstructed using the `RecEmcShower` package. In addition to the requirements imposed by the `RecEmcShower` package, the following standard selection criteria must be satisfied:

```
(1) 0 < shower time < 14
```

```
(2) |cos(theta)| < 0.80 (barrel photons)
       or 0.86 < |cos(theta)| < 0.92 (endcap photons)
(3) for barrel photons:  E > 25 MeV
(4) for endcap photons:  E > 50 MeV
```

3.3 $\pi^0 \to \gamma\gamma$ and $\eta \to \gamma\gamma$

Lists of $\pi^0 \to \gamma\gamma$ and $\eta \to \gamma\gamma$ decays are made using the `EvtRecPi0` and `EvtRecEtaToGG` packages, respectively. The following additional cuts are imposed:

```
(1) chi2 < 2500 (from the fit to the eta or pi0 mass)
(2) for pi0:  0.107 < mass(gamma gamma) < 0.163 GeV/c2
    for eta:  0.400 < mass(gamma gamma) < 0.700 GeV/c2
```

Both photons from the π^0 and η are also required to pass the shower cuts listed in section 3.2.

3.4 $K_S^0 \to \pi^+\pi^-$, $\Lambda \to p\pi^-$ and $\overline{\Lambda} \to \overline{p}\pi^+$

$K_S^0 \to \pi^+\pi^-$, $\Lambda \to p\pi^-$ and $\overline{\Lambda} \to \overline{p}\pi^+$ reconstruction is done with the `EvtRecVeeVertex` package with the following additional cuts:

```
(1) chi2 < 100 (from the fit to the Ks or Lambda vertex)
(2) for Ks:      0.471 < mass(pi+ pi-) < 0.524 GeV/c2
    for Lambda:  1.100 < mass(p pi) < 1.300 GeV/c2
```

4 Event Selection

Once the lists of particles are made, they are combined to form different final states. All combinations are considered. To avoid double-counting, tracks and showers (including tracks and showers from π^0, η, K_S^0, and Λ decays) are used only once per combination. Parameters can be specified in the job options file (see section 6) to reduce the combinatorics.

For both inclusive and exclusive final states, a primary vertex fit (using the `VertexFit` package) is performed using tracks originating from the primary vertex. The primary vertex and the resulting track parameters are saved. If the final state contains no tracks from the primary vertex, the beam spot is used as the primary vertex and no fit is performed. Secondary vertices due to K_S^0 and Λ decays are then created with the `SecondVertexFit` package.

For inclusive final states, the missing mass is calculated using the final state particles and is required to be between the bounds specified in the job options file (see section 6). Multiple combinations are allowed to be recorded per event.

For exclusive final states, the `KalmanKinematicFit` package is used to fit the four-momenta of the final state particles to the initial four-momentum of the e^+e^- interaction. Additional constraints are placed on the π^0, η, K_S^0, and Λ masses. A loose cut (that can be tuned in the job options file) is placed on the resulting χ^2 of each combination.

5 Final State Numbering

`FSFilter` can reconstruct any final state composed of a combination of Λ (called `Lambda`, decaying to $p\pi^-$), $\overline{\Lambda}$ (called `ALambda`, decaying to $\overline{p}\pi^+$), e^+, e^-, μ^+, μ^-, p^+ (proton), p^- (anti-proton), $\eta(\rightarrow \gamma\gamma)$, γ, K^+, K^-, $K_S^0(\rightarrow \pi^+\pi^-)$, π^+, π^-, and $\pi^0(\rightarrow \gamma\gamma)$. Final states are designated using two integers, "code1" and "code2". The digits of each integer are used to specify the number of different particle types in the final state:

```
    code1 = abcdefg
            a = number of gamma
            b = number of K+
            c = number of K-
            d = number of Ks  ( --> pi+ pi- )
            e = number of pi+
            f = number of pi-
            g = number of pi0 ( --> gamma gamma )

    code2 = hijklmnop
            h = number of Lambda (--> p+ pi-)
            i = number of ALambda (--> p- pi+)
            j = number of e+
            k = number of e-
            l = number of mu+
            m = number of mu-
            n = number of p+
            o = number of p-
            p = number of eta  ( --> gamma gamma )
```

These integers are sometimes combined into a single string of the form:

```
    "code2_code1"
```

Here are a few examples:

```
    "0_111":      pi+ pi- pi0
```

```
"0_1000002":  gamma pi0 pi0
"1_220000":   eta K+ K+ K- K-
"11000_110":  mu+ mu- pi+ pi-
```

6 Job Options Parameters

6.1 FSFilter.FS(N) = (EXC/INC)(code2)_(code1);

Use a list of these commands to specify which final states to reconstruct.

```
(N): a unique integer between 0 and 9999 for each
     reconstructed final state (this is only used to
     differentiate FS(N) commands, and isn't used
     elsewhere)

(EXC/INC): use EXC to reconstruct the final state
           exclusively (using a kinematic fit);
           use INC for inclusive reconstruction (with
           bounds on the missing mass)

(code2) and (code1):  the final state numbering integers
                      described in the "final state
                      numbering" section
```

Examples:

```
Reconstruct eta K+ K- exclusively:

    FSFilter.FS100 = EXC1_110000;

Reconstruct pi+ pi- inclusively:

    FSFilter.FS200 = INC0_110;
```

6.2 FSFilter.maxShowers = (n);

Only consider events with less than or equal to `(n)` showers. The default is 50. The purpose of this requirement is just to speed up reconstruction. When there are more than 50 showers in an event, the possible combinations of showers used to form photons or π^0 or η can become very large.

6.3 `FSFilter.maxExtraTracks = (n);`

(EXCLUSIVE RECONSTRUCTION) For exclusive reconstruction, only allow `(n)` extra tracks (tracks beyond those required in the final state) in the event. The default is 2. This requirement is ignored for inclusively reconstructed final states.

6.4 `FSFilter.cmEnergy = (E);`

Specify the center of mass energy `(E)` in units of GeV. The default is the $\psi(2S)$ mass, 3.686093 GeV. (Eventually the default will come from an external `BeamEnergy` package.)

6.5 `FSFilter.crossingAngle = (theta);`

Set the crossing angle in radians. The default is 0.011 radians.

6.6 `FSFilter.energyTolerance = (deltaE);`

(EXCLUSIVE RECONSTRUCTION) For exclusive reconstruction, only combinations of final state particles that have a total energy within `(deltaE)` of the initial total energy will be processed. `(deltaE)` is given in GeV. This reduces the number of kinematic fits that are performed and helps speed processing. The default is 0.250 GeV.

6.7 `FSFilter.momentumTolerance = (deltaP);`

(EXCLUSIVE RECONSTRUCTION) Like `energyTolerance`, this command requires the total momentum of final state particles be within `(deltaP)` (in GeV/c) of the initial total momentum. The default is 0.250 GeV/c.

6.8 `FSFilter.maxKinematicFitChi2DOF = (chi2DOF);`

(EXCLUSIVE RECONSTRUCTION) For exclusive reconstruction, only record events that have a kinematic fit χ^2/dof less than `(chi2DOF)`. Using this can reduce file sizes. The default is 100.

6.9 `FSFilter.lowerMissingMass2 = (lowMM);`

(INCLUSIVE RECONSTRUCTION) Require the missing mass squared of an inclusive combination of final state particles be greater than `(lowMM)` (in GeV^2/c^4). The default is set at -25.0 GeV^2/c^4.

6.10 `FSFilter.upperMissingMass2 = (highMM);`

(INCLUSIVE RECONSTRUCTION) Require the missing mass squared of an inclusive combination of final state particles be less than `(hignMM)` (in GeV^2/c^4). The default is set at 25.0 GeV^2/c^4.

6.11 `FSFilter.YUPING = (true/false);`

If `true`, use the kaon and pion tracking corrections developed by Yuping Guo. The default is `false`.

6.12 `FSFilter.printTruthInformation = (true/false);`

If `true`, print MC truth information to the screen. This can be useful for debugging. The default is `false`.

7 Example Job Options File

Here is a job options file to exclusively reconstruct $\psi(2S) \to \gamma X_i$, where X_i is:

1. $\pi^+\pi^+\pi^-\pi^-$
2. $\pi^+\pi^+\pi^-\pi^-\pi^0\pi^0$
3. $\pi^+\pi^+\pi^+\pi^-\pi^-\pi^-$
4. $K^-K_S^0\pi^+$
5. $K^+K_S^0\pi^-$
6. $K^+K^-\pi^0$
7. $K^+K^-\pi^+\pi^-$
8. $K^-K_S^0\pi^+\pi^+\pi^-$
9. $K^+K_S^0\pi^+\pi^-\pi^-$
10. $K^+K^-\pi^+\pi^-\pi^0$
11. $K^+K^+\pi^+\pi^+\pi^-\pi^-$
12. $K^+K^+K^-K^-$
13. $\eta\pi^+\pi^-$
14. $\eta\pi^+\pi^+\pi^-\pi^-$

and to search for $\psi(2S) \to \gamma\eta_c(1S)$ using inclusive photons.

```
// **********************************************
//      EXAMPLE FSFILTER JOB OPTIONS FILE
// **********************************************

#include "$ROOTIOROOT/share/jobOptions_ReadRec.txt"
#include "$VERTEXFITROOT/share/jobOptions_VertexDbSvc.txt"
#include "$MAGNETICFIELDROOT/share/MagneticField.txt"
#include "$ABSCORROOT/share/jobOptions_AbsCor.txt"
#include "$PI0ETATOGGRECALGROOT/share/jobOptions_Pi0EtaToGGRec.txt"
#include "$FSFILTERROOT/share/jobOptions_FSFilter.txt"

FSFilter.cmEnergy = 3.686093;

FSFilter.FS1  = "EXC0_1000220";
FSFilter.FS2  = "EXC0_1000222";
FSFilter.FS3  = "EXC0_1000330";
FSFilter.FS4  = "EXC0_1011100";
FSFilter.FS5  = "EXC0_1101010";
FSFilter.FS6  = "EXC0_1110001";
FSFilter.FS7  = "EXC0_1110110";
FSFilter.FS8  = "EXC0_1011210";
FSFilter.FS9  = "EXC0_1101120";
FSFilter.FS10 = "EXC0_1110111";
FSFilter.FS11 = "EXC0_1110220";
FSFilter.FS12 = "EXC0_1220000";
FSFilter.FS13 = "EXC1_1000110";
FSFilter.FS14 = "EXC1_1000220";

FSFilter.lowerMissingMass2 = 6.76;
FSFilter.upperMissingMass2 = 10.24;

FSFilter.FS100 = "INC0_1000000";

// Input REC or DST file name
EventCnvSvc.digiRootInputFile = {"/bes3fs/offline/...."};

// Set output level threshold
// (2=DEBUG, 3=INFO, 4=WARNING, 5=ERROR, 6=FATAL )
MessageSvc.OutputLevel = 5;

// Number of events to be processed (default is 10)
ApplicationMgr.EvtMax = 500;

ApplicationMgr.HistogramPersistency = "ROOT";
NTupleSvc.Output = { "FILE1 DATAFILE='PsiPrimeGammaX.root'
                     OPT='NEW' TYP='ROOT'"};
```

8 Output Root Trees

`FSFilter` generates three types of root trees:

1. Reconstructed information is stored in trees named `nt(EXC/INC)(code2)_(code1)`, where `(EXC/INC)` is either `EXC` or `INC` depending on whether the reconstruction was done exclusively or inclusively, and `(code2)` and `(code1)` are the numbering codes described in section 5. There is one entry for each combination of particles that passes the loose cuts described in sections 3 and 4. Thus there may be more than one entry per event. This is the main tree for physics analysis.
2. MC generated information for each exclusive final state is stored in trees named `ntGEN(code2)_(code1)`. This type of tree contains exactly one entry per each event generated with this final state, regardless of whether or not the event was reconstructed. It is only used for exclusive final states. This tree can be used to look at generator-level distributions.
3. MC generated information for all final states is stored in one tree named `ntGEN`. This tree contains exactly one entry per event, independent of final state, and independent of reconstruction. Use this tree to count the number of events generated for different processes.

Each of these trees contains some subset of the blocks of information described in the following subsections. Only the reconstructed tree, `nt(EXC/INC)(code2)_(code1)`, for example, contains shower or track information.

By convention particles are listed in trees in the following order:

```
Lambda ALambda e+ e- mu+ mu- p+ p- eta gamma K+ K- Ks pi+ pi- pi0
```

For example, in the process $\psi(2S) \to \pi^+\pi^- J/\psi; J/\psi \to \mu^+\mu^-$, the μ^+ is particle 1, the μ^- is particle 2, the π^+ is particle 3, and the π^- is particle 4. Or as another example, in the process $\psi(2S) \to \gamma\chi_{c1}; \chi_{c1} \to \eta\pi^0\pi^0$, the η is particle 1, the γ is particle 2, one π^0 is particle 3, and the other π^0 is particle 4. In cases like this where there are identical particles, no ordering is assumed.

8.1 Event Information

The "Event Selection" block contains information about the event as a whole.

The following information is contained in all three tree types:

```
Run:          run number
Event:        event number
BeamEnergy:   beam energy
NTracks:      total number of tracks in the event
NShowers:     total number of showers in the event
```

The following information is only contained in `nt(EXC/INC)(code2)_(code1)`, where the total energies and momenta are calculated using the raw particle four-momenta (i.e., before any vertex or kinematic fitting):

```
TotalEnergy:    total energy of all final state particles
TotalPx:        total px of all final state particles
TotalPy:        total py of all final state particles
TotalPz:        total pz of all final state particles
TotalP:         total momentum of all final state particles
MissingMass2:   missing mass squared of the event
VChi2:          chi2 of the vertex fit if there was a
                  vertex fit (-1 otherwise)
```

Finally, this information is only contained in `ntEXC(code2)_(code1)`:

```
Chi2:         chi2 of the kinematic fit
Chi2DOF:      chi2/dof of the kinematic fit
```

8.2 Particle Four-Momenta

The "particle four-momenta" block contains the four-momentum for each particle in the final state. Different types of four-momenta are distinguished using prefixes. MC generated four-momenta have a prefix `MC`; raw four-momenta have a prefix `R`; vertex-constrained four-momenta have a prefix `V`; and the fully-constrained four-momenta resulting from the kinematic fit have no prefix. Different particles are differentiated using `P(n)`, where `(n)` is the number of the particle in the ordered list. For example, in the process $\psi(2S) \to \pi^+\pi^- J/\psi; J/\psi \to \mu^+\mu^-$, the raw energy of the π^+ is given by `REnP3`.

```
(prefix)PxP(n):  x momentum of particle (n)
(prefix)PyP(n):  y momentum of particle (n)
(prefix)PzP(n):  z momentum of particle (n)
(prefix)EnP(n):  energy of particle (n)
```

Particle four-momenta are included in both the `nt(EXC/INC)(code2)_(code1)` and the `ntGEN(code2)_(code1)` trees. In the `ntGEN(code2)_(code1)` tree, the four-momenta are actually written out twice, once with the `MC` prefix and once with no prefix. This makes it easier to treat the generated MC more like data in some applications.

In the reconstructed tree, `nt(EXC/INC)(code2)_(code1)`, MC generated information is also included for events in which the generated and reconstructed final states are the same. When there are identical particles, however, no attempt is made to ensure that the particles are ordered consistently, i.e., no attempt is made to match reconstructed particles with generated particles.

8.3 Shower Information

Shower information is written out for every reconstructed photon that is part of a final state. Individual photons have the prefix `Sh`; photons from π^0 decay have either the prefix `Pi0ShLo` (for the low energy photon) or `Pi0ShHi` (for the high energy photon); and similarly photons from η decay have the prefix `EtaShLo` or `EtaShHi`. As in the four-momenta case, showers from different particles are differentiated using `P(n)`, where `(n)` is the number of the particle in the ordered list. As examples: the energy of the low energy photon from the π^0 decay in $\psi(2S) \to \pi^+\pi^-\pi^0$ is called `Pi0ShLoEnergyP3`; and the timing of the photon in $\psi(2S) \to \gamma\eta$ is called `ShTimeP2`.

```
(prefix)TimeP(n):     timing information
(prefix)EnergyP(n):   shower energy
(prefix)CosThetaP(n): cos(theta) in the lab
(prefix)E925P(n):     E9/E25 (the energy in a 3x3 array
                       over the energy in a 5x5 array)
(prefix)Pi0PullP(n):  the pull of the best pi0 formed
                       with this shower
(prefix)DangP(n):     smallest angle between this shower
                       and tracks projected to the EMC
```

8.4 Track Information

Track information is written out for every reconstructed track that is part of a final state. Tracks from the primary vertex have the prefix `Tk`. Tracks from secondary vertices (K_S^0, Λ, and $\overline{\Lambda}$ decays) have the prefixes `VeeTk1` (for the π^+, p, and $\overline{p}$, respectively) and `VeeTk2` (for the π^-, π^-, and π^+).

```
(prefix)ProbPiP(n):   probability track is a pion
(prefix)ProbKP(n):    probability track is a kaon
(prefix)ProbPP(n):    probability track is a proton
(prefix)ProbMuP(n):   probability track is a muon
(prefix)ProbEP(n):    probability track is an electron
(prefix)RVtxP(n):     radial distance of closest approach
                        to the primary vertex (in cm)
(prefix)ZVtxP(n):     longitudinal distance
                        to the primary vertex (in cm)
(prefix)CosThetaP(n): cos(theta) in the lab
(prefix)EPP(n):       E/p (shower energy over
                        track momentum)
```

8.5 $\pi^0 \to \gamma\gamma$ Information

The following information for each $\pi^0 \to \gamma\gamma$ decay is recorded:

```
  Pi0MassP(n):    the unconstrained gamma gamma mass
  Pi0Chi2P(n):    the chi2 of the 1C fit to the pi0 mass
```

Note that the shower information for each γ from the π^0 is recorded along with the other showers.

8.6 $\eta \to \gamma\gamma$ Information

The decay $\eta \to \gamma\gamma$ is treated in the same way as $\pi^0 \to \gamma\gamma$:

```
  EtaMassP(n):    the unconstrained gamma gamma mass
  EtaChi2P(n):    the chi2 of the 1C fit to the eta mass
```

Note that the shower information for each γ from the η is recorded along with the other showers.

8.7 $K_S^0 \to \pi^+\pi^-$, $\Lambda \to p\pi^-$ and $\overline{\Lambda} \to \overline{p}\pi^+$ Information

This information is recorded for each $K_S^0 \to \pi^+\pi^-$, $\Lambda \to p\pi^-$ and $\overline{\Lambda} \to \overline{p}\pi^+$ decay:

```
  VeeMassP(n):    the unconstrained mass of the daughters
  VeeChi2P(n):    the chi2 of the secondary vertex fit
  VeeLSigma(n):   the separation between the primary and
                   secondary vertex (L) over its error (sigma)
```

Note that the track information for the daughter tracks are recorded along with other tracks.

8.8 Information to Tag $\psi(2S) \to XJ/\psi$

A few variables are included to try to identify $\psi(2S)$ transitions to J/ψ, sometimes useful for identifying J/ψ backgrounds. This information is currently included when running over all data sets, but it is only meaningful for $\psi(2S)$ data.

```
  JPsiPiPiRecoil:  the smallest difference between the J/psi
                     mass and any combination of pi+ pi-
                     recoil mass (using all positive and
                     negative tracks with assumed pion
                     masses)
  JPsiGGRecoil:    the smallest difference between the J/psi
                     mass and any combination of gamma gamma
                     recoil mass
```

8.9 Truth Information

When running over Monte Carlo, the truth information about a given event is included in all three tree types. These variables should allow one to separate and count signal and background events. The numbers in this block of information always refer to generator level information. For example, the “number of K_L^0” refers to the true number of generated K_L^0.

These variables label the generated final state:

```
MCDecayCode1:      the true code1, described in the
                     "final state numbering" section
MCDecayCode2:      the true code2, described in the
                     "final state numbering" section
MCExtras:          1000 * number of neutrinos +
                    100 * number of K_L +
                     10 * number of neutrons +
                      1 * number of antineutrons
MCTotalEnergy:     the total generated energy calculated
                     using particles in MCDecayCode(1,2)
                     and MCExtras:  use this to double-
                     check that no particles are missing
MCSignal:          1 if the reconstructed final state
                     matches the generated final state;
                     0 otherwise
```

These variables keep track of how various states were produced, where `X` stands for any particle:

```
MCChicProduction:  10 for X to gamma chi_c0
                   11 for X to gamma chi_c1
                   12 for X to gamma chi_c2
                    1 for any other production of chi_cJ
                    0 for no produced chi_cJ
MCJPsiProduction:  1 for X to pi+ pi- J/psi
                   2 for X to pi0 pi0 J/psi
                   3 for X to eta J/psi
                   4 for X to gamma J/psi
                   5 for X to pi0 J/psi
                   6 for any other production of J/psi
                   0 for no produced J/psi
MCHcProduction:    1 for X to pi+ pi- h_c
                   2 for X to pi0 pi0 h_c
                   3 for X to eta h_c
                   4 for X to gamma h_c
                   5 for X to pi0 h_c
                   6 for any other production of h_c
```

```
                    0 for no produced h_c
MCEtacProduction:  1 for X to gamma eta_c
                    2 for any other production of eta_c
                    0 for no produced eta_c
```

These variables count the number of ways in which various particles decayed:

```
MCEtaDecay:       10000 * number of gamma e+ e- +
                   1000 * number of gamma pi+ pi- +
                    100 * number of pi+ pi- pi0 +
                     10 * number of pi0 pi0 pi0 +
                      1 * number of gamma gamma
MCEtaprimeDecay:  1000000 * number of pi0 pi0 pi0 +
                   100000 * number of gamma gamma +
                    10000 * number of gamma omega +
                     1000 * number of gamma pi+ pi- +
                      100 * number of gamma rho +
                       10 * number of eta pi0 pi0 +
                        1 * number of eta pi+ pi-
MCPhiDecay:     10000 * number of gamma eta +
                 1000 * number of rho pi +
                  100 * number of pi+ pi- pi0 +
                   10 * number of Ks Kl +
                    1 * number of K+ K-
MCOmegaDecay:   100 * number of pi+ pi- +
                 10 * number of gamma pi0 +
                  1 * number of pi+ pi- pi0
MCKsDecay:      10 * number of pi0 pi0 +
                 1 * number of pi+ pi-
MCFSRGamma:   number of FSR gammas in this event
               (FSR gammas are ignored elsewhere)
```

The `MCDecayParticle` variables are an ordered list of the PDG ID numbers of the particles coming from the initial decay. For example, for $\psi(2S) \to \rho^+\pi^-$, `MCDecayParticle1` is -211 (for the π^-); `MCDecayParticle2` is 213 (for the ρ^+); and all the others are zero.

```
MCDecayParticle1:  the PDG ID of the 1st particle
MCDecayParticle2:  the PDG ID of the 2nd particle
MCDecayParticle3:  the PDG ID of the 3rd particle
MCDecayParticle4:  the PDG ID of the 4th particle
MCDecayParticle5:  the PDG ID of the 5th particle
MCDecayParticle6:  the PDG ID of the 6th particle
```