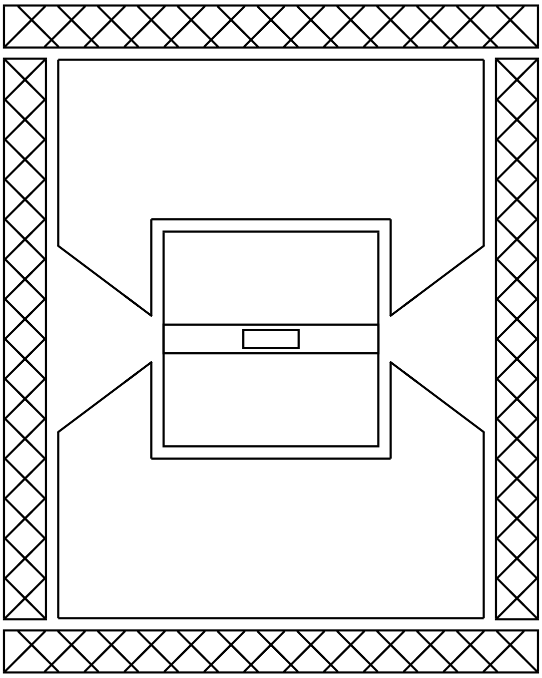
CB–Note 191

# LEAR Crystal Barrel Experiment, PS197
# User Guide for CBDROP
# A Charged Splitoff Supression Package

Curtis A. Meyer
Universität Zürich

30 June, 1992 — updated on 29 March, 1995



# Contents

# 1 Introduction

The CBDROP code is designed to try and enhance data samples in the mixed charged and neutral environment. In these mixed channels, many charged particles will produce more than one energy deposit in the barrel. However, for global tracking to make any sense, it can attach at most *one* PED to a given track. The purpose of CBDROP is to attempt to figure out which if any additional PED's should also be considered as charged split–offs.

The method is based on a fast 4–C kinematic fit to momentum and energy balance. The event is initially given to the fitter as it stands. This leads to a confidence level, $P_0$, in which none of the neutral PED's are considered as charged split-offs, ( *drop 0*). Next, for $n$ neutral PED's in the event, $n$ fits are performed, each dropping a different neutral PED. The drop combination that yields the maximum confidence level, $P_1$ is retained as the best *drop 1* combination. Additionally, every possible pair of neutral PED's can be dropped, yielding a best *drop 2* confidence level, $P_2$. This can be continued to yield a *drop 3* up to a *drop n.* If one allows at most $m$ drops, (typically one per charged track), then the maximum of $P_0$, $P_1$ ... $P_m$ is retained as the true event configuration.

This code is based on the analysis procedure developed by B. Schmid for her analsysis of $\pi^+\pi^-\pi^\circ\pi^\circ$. CBDROP is the generalization of this method in a documented package form and and is presently being used by D. Urner for $\pi^+\pi^-\pi^\circ\pi^\circ\eta$ and C. A. Meyer for $\pi^+\pi^-\pi^\circ\pi^\circ\pi^\circ$. Any questions, comments or complaints should sent to:

```
cmeyer@vogon.physik.unizh.ch
```

# 2 Installation

CBDROP is distributed as a CMZ file. The code is slightly machine dependent, but the normal Crystal Barrel machine flags are used. The CMZ file can be found in the following places:

- CB DECSTATIONS /cboff/pro/cmz/cbdrop.cmz
- e–mail cmeyer@ernest.phys.cmu.edu

You can get the CMZ using binary FTP. Then create a source subdirectory under the account containing CBDROP.CMZ. You can now use the automatic installation procedure, which on a Unix platform is:

```
cmz -install cbdrop src_directory [options]
```

While under VMS this is:

```
CMZ/INSTALL=CBDROP,[.src_directory],[options]
```

The allowed options are:

- ALL Create source code, a library, and the manual.
- LIB Create the library.
- SOURCE Create the compiled source code.
- TEXT Create the manual.

# 3 How To Use CBDROP

To use the CBDROP code, one must first initialize the package. Then for every event for which there is a chance that dropping photons can help, one should call the CBDROP code. Even though CBDROP will check to make sure that the event is consistent with your requirements before running, it is still faster to *prefilter* the events. If you are looking for final states such as $\pi^+\pi^-6\gamma$, then it makes no senese to call CBDROP when the event has more than 2 tracks at the vertex, or when the total number of neutral PED's is smaller than 6.

## 3.1 Initialization

In order to initialize the CBDROP code, it is necessary to call the USDINI routine from your USINIT routine. This will define a *default* configuration for CBDROP. This default configuration can then be modified to suit your own needs, (see the example code at the end of this section). To modify the default configuration, you need to include the /USDCUT/ common in your USINIT code, (see below). You can then modify the following variables. Note that these are only used to steer the CBDROP code, their values will never be changed by the CBDROP routine.

- EGAMUS is a REAL variable which defines the minimum energy of a photon, (default is zero). The photon energy is taken from word 15 of the **TTKS** (global tracking) bank. The default will accept **all** neutral particles in the **TTKS** bank as photons. Setting a different value will take only those particles whose energy is larger than EGAMUS as photons.

- ECENUS is a REAL variable which defines the minimun energy in the central crystal of a PED. It has a defualt value of zero.

- EBALUS is a REAL variable which defines the energy constraint applied in the 4-C kinematic fit. The default value is 2 proton masses in units of MeV.

- PBALUS is a REAL variable which defines the $z$–momentum constraint applied in the 4-C kinematic fit. The default value is zero in units of MeV/c. No check is made to assure that EBALUS and PBALUS are internally consistent.

- ESMCUS is a real variable which defines a cut applied to the unfit energy balance before a 4–C fit is performed. If $\mid E_{sum}-$ EBALUS $\mid>$ ESMCUS, then the Confidence Level is set to 0, and no fit is made The default is 10000.0, which is essentially no cut.

- PSMCUS is a REAL variable which defines a cut applied to the momentum balance before a 4–C fit is performed. If $\mid P_{sum}- \mid$ PBALUS$\mid\mid>$ PSMCUS, then the Confidence Level is set to 0, and no fit is made. The default is 10000.0, which is essentially no cut.

- VSHFUS is a REAL array of dimension 3. In conjunction with ISHFUS, the user can move the photon vertex from the one defined in the data base to VSHFUS. The default is $(0, 0, 0)$, and the option is turned off.

- LEGAUS is a LOGICAL variable that when set, only accepts PEDS whose energy is larger than EGAMUS.

- LCENUS is a LOGICAL variable that when set, only accepts PEDS whose central energy is larger than ECENUS. The default value of LCENUS is .FALSE..

- LTAXI2 is a logical that when set forces CBDROP to repsepct the spilt–off decision taken by TAXI2. Its default value is .TRUE..

- LDOLBY is a logical that when set forces CBDROP to respect the split–off decision taken by DOLBY. Its default value is .TRUE..

- LSMART is a logical that when set TRUE forces CBDROP to respect the split–off decision taken by SMART. Only SMART code 1 PEDS are rejected. This variable has a default value of .FALSE..

- NTRKUS is an INTEGER variable which defines the number of charged tracks allowed in the event. The default is zero, so you will certainly need to change this. Events without exactly NTRKUS tracks attached to the primary vertex are rejected. All tracks are assumed to be pions unless you modify IGTRAK.

- LONGUS is an INTEGER to define the minimum length of each track. All of the NTRKUS tracks must span at least LONGUS layers. The default is ten. (Note that for a track to be connected to a vertex, it must already start no later than layer five.)

- IVRTUS is an INTEGER variable to select if the track data are taken from the **TTKS**, **TCTR** or **TCVT** data banks. The default value of zero will select data from the **TTKS** data bank. Setting this to 1 will select data from the **TCTR** data bank, and 2 will select data from the **TCVT** banks. Note that if you choose option one or two, then ALL charged particles will always be treated as pions. Only with option zero can you choose to accept the global tracking identification, (see IGTRAK).

- MASKUS is an INTEGER bit mask for rejecting charged tracks at the vertex, the default is 15, where the meaning is in the following table.

```
*---Bit 0: Charge does not sum to zero.
*---Bit 1: All tracks are not long, (LONGUS).
*---Bit 2: At least one track is poorly fit.
*---Bit 3: At least one track starts outside layer 5.
*---Bit 4: The vertex convergence is poor.
*---Mask 13 = 01101 ==> Q_sum, E_code, Start_layer
*---Mask 15 = 01111 ==> Q_sum, Length, E_code, Start_layer
```

- ISHFUS is an INTEGER variable. If set to zero, (default), no vertex corrections to the photons are made. If set to one, then the vertex is corrected to VSHFUS(1:3). If set to two, then the vertex is even event corrected to the charged vertex.

- IGTRAK is an INTEGER variable. For charged particles, the default is to assume that everything is a pion, (IGTRAK=0). However, if IGTRAK is set to one (1), the the particle ID, (kaon or pion), is taken from the global tracking bank. This is determined by looking at the words +28 and +29, and choosing which is larger. Setting IGTRAK=2 will force all charged tracks to be identified as kaons. Note that this variable can only be used for IVRTUS of 0.

- INTLUS If set to a postive value, (default is zero), then the so called *smart–mode* is activated. In this mode the variables NGMNUS and NGMXUS are used to define the maximum and minimum desired photons. If the number of potential photons is smaller than NGMNUS or the number minus the maximum allowed drop is larger than NGMXUS, then the events will **not** be fit. Also, the actual maximum number dropped may not be the number passed. In this mode, the routine will perform no fits with fewer than NGMNUS photons, (see section 3.3).

- NGMNUS used to set a lower bound on the number of photons in the event. See INTLUS, the default is zero.

- NGMXUS used to set an upper bound on the number of photons in the event. See INTLUS, the default is zero.

It may also be necessary to modify the errors related to charged tracks. In order to do this is in a consitent manner, the common block /TCCVRX/ has been provided. By setting scale factors here which scale the error (not the error squared), the TCHLDD[S] routines will set reasonable errors for the charged tracks. One should note that when tracks are taken from either the **TTKS** or **TCTR** banks, CBDROP realizes that all the errors in $\psi$ need to be doubled to compensate for the fact that one is not fitting at the vertex. This is done automatically.

- ALPHUS The factor by which the error is $\alpha$ is scaled. The default is 1.00.
- TGLMUS The factor by which the error in $\tan\lambda$ is scaled. The default is 1.00.
- PSIFUS The factor by which the error in $\psi$ is scaled. The default is 1.00. Note there is an automatic factor of two in this factor internal in CBDROP. Do **not** set this value to 2 to get the vertex correction described above.
- AFACUS Allows one to scale the transverse momentum by this factor. The deafult is 1.00.
- XENEUS Scale factor applied to the error in the $\sqrt{E_\gamma}$. The default is 1.00.
- XTHEUS Scale factor applied to the error in $\theta_\gamma$, the default is 1.00.
- XPHIUS Scale factor applied to the error in $\phi_\gamma$, the default is 1.00.

A possible example setup of CBDROP is given in the following snipit of the USINIT routine. This is used to select two prong events.

```
      SUBROUTINE USINIT
          ...
+CDE,USDCUT.
+CDE,TCCVRX.
          ...
       CALL USDINI
       NTRKUS = 2
       EGAMUS = 20.0
          ...
```

It also seems necessary to modify the default tracking errors. The following values seem reasonable for the November ’90 data.

```
       ALPHUS = 2.33
       PSIFUS = 1.20
       TGLMUS = 1.20
*
       AFACUS = 1.0725
*
       XENEUS = 1.30
       XTHEUS = 1.30
       XPHIUS = 1.30
```

## 3.2 Event Processing

In your USER routine, you need to call the CBDROP subroutine. You then need to make a decision on whether to keep or reject the event based on what CBDROP returns. It is also necessary that you provide sufficient working space for CBDROP. The following code snippet indicates how to setup and call cbdrop for dropping a maximum of NTRY= 2 neutral PED's.

```
      SUBROUTINE USER
          ...
+CDE,USDCUT.
+CDE,USPMOM.
          ...
      INTEGER    NTRY
      PARAMETER (NTRY=2)
*
      INTEGER NKEEP(0:NTRY),IKEEP(20,2,2,0:NTRY)
      INTEGER NDROP(0:NTRY),IDROP(20,2,2,0:NTRY)
      INTEGER INDX
      REAL    CLEV(0:NTRY),CLEV2(0:NTRY)
          ...
      IERR = 0
      CALL CBDROP(NTRY,INDX,NDROP,NKEEP,IDROP,IKEEP,CLEV,
     &            CLEV2,IERR,NLOW)
      IF (IERR .NE. 0) RETURN
*
```

The variable IERR is an error status word that gives some indication of why the fit failed. Only events with IERR equal to 0 should be accepted. Other possible values with their meanings are:

-10 Passed NTRY is negative.

-07 Event fails *smart–mode* criteria, see INTLUS.

-06 More than 20 particles in the event.

-05 More than 40 particles in **TTKS** bank.

-04 Photon Loading fails.

-03 Track Mask is wrong, event does not satisfy all conditions required in MASKUS.

-02 Track load fails.

-01 Number of tracks was not NTRKUS.

+00 **Successful fit of data.**

+01 Global tracking bank does not exist.

+03 Chisqure from the kinematic fit is larger than 1,000,000.

+05 Kinematic fit failed to converge in 10 iterations.

+07 Kinematic fit tried to invert singular matrix.

Assuming that IERR is zero, then NLOW is the number of photons in the **TTKS** bank that were simply ignored as their energy was too low. You will now need to address the data. The variable INDX indicates which drop combination had the largest confidence level. The best confidence level is CLEV(INDX), the number of photons dropped in the best case is NDROP(INDX), and the number kept is NKEEP(INDX). A list of the $n$ kept photons **TTKS** id is given as IKEEP(1:N,2,1,INDX), while their **TBTK** id are given as IKEEP(1:N,1,1,INDX). Similar lists of the dropped photons are given in the IDROP(1:N,2:1,1,INDX) array. Also, at this point the 4–C kinematically fit variables for the best combination areavailable in the USPMOM common block, (see section 4.4).

In addition to the previous information, the same information is available for the best of each of the NTRY drops. These data are addressed by simply replacing INDX in the previous section with the number of dropped photons, (*e.g.* 0,1,...,NTRY).

Finally, if this is not *preverse* enough, the second best solution for each of the NTRY drop combunations are also available. The confidence levels for these are stored in the CLEV2 array. The list of kept photons for the second best combination of dropping $m$ are available as IKEEP(1:N,2:1,2,M), while the dropped list is similarly IDROP(1:N,2:1,2,M).

Assuming that you are willing to believe that the best combination as chosen by CBDROP is really the true combination, the following snippet of code shows how to extract the data. First, you might want to make a cut on the number of photons in the event.

```
      IF( (NKEEP(INDX) .LT. NGMNUS) .OR.
     &    (NKEEP(INDX) .GT. NGMXUS) ) RETURN
```

Now make sure that the confidence level is reasonable. Here we reject events with a confidence level smaller than 1%.

```
      IF( CLEV(INDX) .LT. 0.01 ) RETURN
```

Now it would be useful save the global tracking numbers of the kept particles, (charged plus neutral), for possible use by CBKFIT. We will store these in a varaible LISTUS. Note that rather than the IKEEP list of photons, I am taking this list out of the USPMOM common block.

```
      DO 100 I = 1,NPRTUS
         LISTUS(I) = IGIDUS(I)
 100  CONTINUE
```

One could also get the list of kept and dropped photons as follows:

```
      DO 200 I = 1,NKEEP(INDX)
        JJKEEP(I) = IKEEP(I,2,1,INDX)
 200  CONTINUE
*
      DO 300 I = 1,NDROP(INDX)
        JJDROP(I) = IDROP(I,2,1,INDX)
 300  CONTINUE
```

It is also possible for the user to prevent individual unmatched PED's from being used by CBDROP. The variables NBADUS and IBADUS in the /PALLUS/ common block have been provided for this purpose. In order to implement this, add the following snipit of code to you USER routine before calling CBDROP.

```
+CDE,USPALUS.
    ...
        NBADUS = 0
    ...
```

Now you need to decide which PED's you do not want to use. Assuming that you decide that **TTKS** particles 4, 7 and 12 are not wanted, then:

```
NBADUS = 3
IBADUS(1) = 4
IBADUS(2) = 7
IBADUS(3) = 12
  ...
CALL CBDROP(...)
```

It is very important that you reset the value NBADUS to zero at the start of every event. Because of the nature of this cut, CBDROP has no sensible method of resetting NBADUS at the start of a new event. If you did not change NBADUS, then the above three PED's would be disabled on every event.

### 3.3 Event Processing in Smart–Mode

If you are looking for a specific final state, (not several simultaneously), and you are planning to use CBKFIT in addition to CBDROP, then you probably want to consider using the *smart–mode* of CBDROP. If you run in this mode, you must specify the number of photons allowed in the final state. CBDROP will then vary its dropping rules to try to maximize the number of events in your configuration. It will also skip over events which have no chance of fullfilling your specific final state criteria.

- If the number of photons in the event is smaller than the number you requested, the event will be skipped. In this case an error code of $-7$ is returned by CBDROP.
- If the number of photons minus the maximum to drop is larger than the number of requested, the event will be skipped. In this case an error code of $-7$ is returned by CBDROP.
- CBDROP will only drop sufficient photons to exactly satisfy your request, NGMNUS. If you ask for 6, allow CBDROP to drop up to 2, and the event has 7 potoential photons, then on this event, CBDROP will not try the drop–0 and drop–2 combinations, only the drop–1 will be performed.

To set up this mode, you must specify the number of photons you want, and turn on the mode. This is done in USINIT after your call to USDINI. In the case of 6 photons, the following code must be used.

```
NGMNUS = 6
NGMXUS = 6
INTLUS = 1
```

After this is done, the code will try to maximize the number of events with exactly six photons. This can be dangerous as it not only increases the true signal, but also the background. For this reason, it is strongly reccomended that this mode be used only in conjunction with CBKFIT, *i.e.* the 7–C fit to $3\pi^{\circ}$'s. If this is done correctly, it can give an extra 20–30% data with minimal extra background. If you do not plan to use CBKFIT with extra constraints, then you should be rather wary of using this mode.

## 4 Full Common Block Description

All of the following common blocks can be picked up in the usual method.

```
+CDE,Common_block_name.
```

## 4.1 USDCUT

The USDCUT keep is a common block used for steering the drop job. Not all variables in this common are used by CBDROP, but they are present due to historical reasons. They can be used by the USER to steer the running of CBDROP.

```
      REAL            EGAMUS,ECENUS,EBALUS,PBALUS,ESMCUS,PSMCUS
      REAL            VSHFUS(3)
      INTEGER         NDRPUS,NGMNUS,NGMXUS,NGAMUS,NTRKUS,LONGUS
      INTEGER         IVRTUS,MASKUS,ISHFUS,IGTRAK,INTLUS
      LOGICAL         LEDGUS,LCENUS,LEGAUS,LTAXI2,LDOLBY,LSMART
      COMMON /USDCUT/ EGAMUS,ECENUS,EBALUS,PBALUS,ESMCUS,PSMCUS,
     &                VSHFUS,NDRPUS,NGMNUS,NGMXUS,NGAMUS,NTRKUS,
     &                LONGUS,IVRTUS,MASKUS,ISHFUS,IGTRAK,LEDGUS,
     &                LCENUS,LEGAUS,LTAXI2,LDOLBY,LSMART,INTLUS
      SAVE   /USDCUT/
```

- EGAMUS A real variable which defines the minimum allowed energy for a photon. See the initialization section for a description.
- ECENUS A real variable which defines the minimum energy in the central crystal of a ped.
- EBALUS Is the energy constraint used in the 4–C fit.
- PBALUS Is the $p_z$ constraint used in the 4–C fit. You must guarantee that EBALUS and PBALUS are consistent.
- ESMCUS Used to preselect fit combinations based on the total energy. See section 3.1
- PSCMUS Used to preselect fit combinations based on the momentum balance. See section 3.1.
- VSHFUS Used to define an alternate fixed vertex for all events. This is only applied when ISHFUS is set to 1.
- NDRPUS Maximum number of PEDS to drop. Not used by the package, but made available for private use.
- NGMNUS The minimum number of photons desired. Only used in *smart–mode*, (see section 3.3).
- NGMXUS The maximum number of photons desired. Only used in *smart–mode*, (see section 3.3).
- NGAMUS The number of photons in the present event. Filled by the package upon completion.
- NTRKUS The number of tracks at the vertex. See Section 3.1 for a description. If the event does not have NTRKUS tracks at the vertex, it is not fit.
- LONGUS The minimum track length. See section 3.1 for usage. All NTRKUS tracks must have a length of at least LONGUS.
- IVRTUS Take track data from **TTKS** (0), **TCTR** (1) or **TCVX** (2). See section 3.1 for usage.
- MASKUS A track quality mask, see sectio 3.1 for a description and usage.
- ISHFUS Used to modify the vertex for photons. If set to 0, then no modification is done. If set to 1, all events are shifted to VSHFUS, and if set to 2, all events are shifted to the charged vertex.

- IGTRAK If set to one, then use the global tracking particle type for charged particles, (pion or kaon). When left zero, all charged particles are assumed to be pions. If set to two, all particles are assumed to be kaons.
- LEDGUS Reject Events with "edge" PEDS. This cut must be applied by the user. The variable is not used in the package.
- LCENUS Use central crystal energy. See section 3.1 for a description.
- LEGAUS Set to TRUE if CBDROP should place a cut on the PED energy.
- LTAXI2 Set to TRUE if CBDROP should respect the results of TAXI.
- LDOLBY Set to TRUE if CBDROP should respect the results of DOLBY.
- LSMART Set to TRUE if CBDROP should respect the results of SMART.
- INTLUS If set larger than zero, *smart–mode* is activated. In *smart–mode*, CBDROP will not fit any combinations in which the number of photons is smaller than NGMNUS. It will also make sure that given the allowed number of drops, that the event could come into the range NGMNUS to NGMXUS. (see section 3.3).

## 4.2 USPALL

The USPALL keep is a common block used to speed up loading of data for the kinematic fit. It contains entries for all particles which could possibly be used in the event. In order to access these particles, one needs to use the IGALUS values as a pointer to where the data are stored, (they are stored in the index corresponding to their global tracking number).

```
DO ITRK = 1,NALLUS
   I = IGALUS(ITRK)
   DO K = 1,3
      P(K) = PALLUS(K,I)
   END DO
     ...
END DO
```

The variables in this common block are given as follows.

```
 REAL         PALLUS(3,40),CVALUS(3,3,40),CHALUS(40),MSALUS(40)
 INTEGER      IGALUS(40),LGALUS(40),EDGEUS(40),NALLUS
 INTEGER      NBADUS,IBADUS(40)
 COMMON /USPALL/ PALLUS,CVALUS,CHALUS,MSALUS
&               ,IGALUS,LGALUS,EDGEUS,NALLUS
&               ,NBADUS,IBADUS
 SAVE   /USPALL/
```

- PALLUS Initial momentum vectors of all particles.
- CVALUS Initial covariance matrix of each particle.
- CHALUS Charge of the each particles.
- MSALUS Mass of the each particles.
- IGALUS The number of this particle in the TTKS data bank.

- LGALUS Does the particle pass energy cuts? 0=yes
- EDGEUS Edge Crystal Test? 0=not edge crystal.
- NALLUS Number of stored particles.
- NBADUS Number of PED's the user wants to disable.
- IBADUS List of NBADUS disables PED's.

## 4.3 USPART

The USPART keep is not actually a common block, but rather data on many different particles. The following particles are presently included:

```
      INTEGER    IGAM,IPIQ,IPI0,IKQ,IK0,IETA,IOMEG,IETAP,IPHI
      INTEGER    IRHQ,IRH0,IPROT
      PARAMETER (IGAM = 22)
      PARAMETER (IPI0 =111,IPIQ=211)
      PARAMETER (IKQ  =321,IK0 =311)
      PARAMETER (IETA =221,IOMEG=223)
      PARAMETER (IETAP=331)
      PARAMETER (IPHI=333)
      PARAMETER (IRHQ=213,IRH0=113)
      PARAMETER (IPROT=2212)

      REAL       GAMM,PI0M,PIQM,KQM,K0M,ETAM,OMEGM,ETAPM,PHIM
      REAL       RHQM,RH0M,PROTM
      PARAMETER (GAMM =  0.0000)
      PARAMETER (PI0M =134.9739, PIQM =139.5673)
      PARAMETER (K0M  =497.671 , KQM  =493.646 )
      PARAMETER (ETAM =548.8   , OMEGM=781.95  )
      PARAMETER (ETAPM=957.50  , PHIM=1019.412 )
      PARAMETER (RHQM =768.3   , RH0M=768.3    )
      PARAMETER (PROTM=938.27231)
```

```
    Particle      Code          Mass
    photon      IGAM   =   22  GAMM   =    0.0000
    pion(+/-)   IPIQ   =  211  PIQM   =  139.5673
    pion(0)     IPI0   =  111  PI0M   =  134.9739
    kaon(+/-)   IKQ    =  321  KQM    =  493.646
    kaon(0)     IK0    =  311  K0M    =  497.671
    eta(548)    IETA   =  221  ETAM   =  548.4
    omega(782)  IOMEG  =  223  OMEGM  =  781.95
    eta'(958)   IETAP  =  331  ETAPM  =  957.50
    phi(1020)   IPHI   =  333  PHIM   = 1019.412
    rho(+/-)    IRHQ   =  213  RHQM   =  768.3
    rho(0)      IRH0   =  113  RH0M   =  768.3
    proton      IPROT  = 2212  PROTM  =  938.27231
```

## 4.4 USPMOM

The USPMOM keep is a common block used to return the kinematic fit results. Upon completion of CBDROP, the variables will be loaded with the results for the *best* overall 4-C fit. These may or may not be the desired results, but are available as a by product of the CBDROP call without any extra effort. All momentum vectors are in Cartesian coordinates, $(p_x, p_y, p_z)$.

```
 REAL         PINIUS(3,20),PFITUS(3,20),DELPUS(3,20),CHSQUS(20)
 REAL         CVINUS(3,3,20),CVFTUS(3,3,20),CHRGUS(20),MASSUS(20)
 INTEGER      IGIDUS(20),NPRTUS
 COMMON /USPMOM/ PINIUS,PFITUS,DELPUS,CHSQUS,CVINUS,CVFTUS,
&                CHRGUS,MASSUS,IGIDUS,NPRTUS
 SAVE   /USPMOM/
```

- PINIUS Initial momentum vectors of the NPRTUS particles.
- PFITUS Fit momentum vectors of the NPRTUS particles.
- DELPUS Change in momentum vectors, initial minus fit.
- CHSQUS Contribution to chisquare of each track.
- CVINUS Initial covariance matrix of each particle.
- CVFTUS Fit covariance matrix of each partcile.
- CHRGUS Charge of the each particles.
- MASSUS Mass of the each particles.
- IGIDUS The number of this particle in the TTKS data bank.
- NPRTUS Number of particles in the fit.