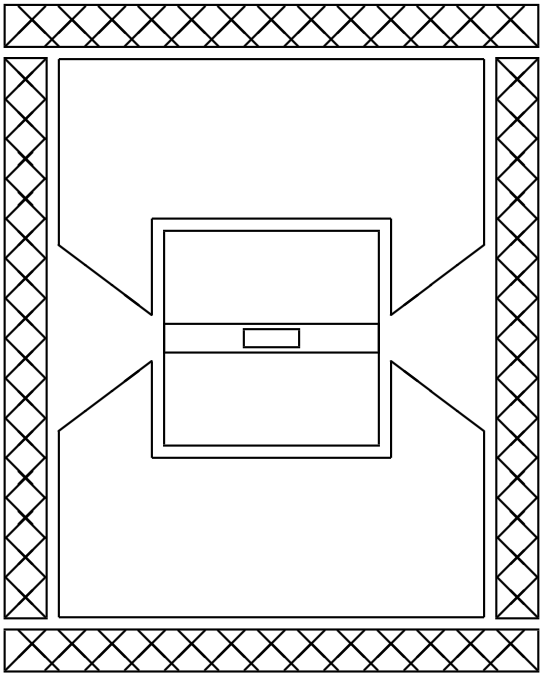
CB–Note 123/Revised

LEAR Crystal Barrel Experiment, PS197
User Guide for Locater 2.00

Curtis A. Meyer
Carnegie Mellon University

20 July, 1994



Contents

1 Introduction **1**

2 Building the Locater code **1**

3 Input to Locater **3**
3.1 External Calibration Files . . . 3
3.2 Steering of Locater in CBOFF . . . 3
3.3 Internal Steering of Locater . . . 4

4 Output From Locater **5**
4.1 The HTJD Data Bank . . . 6
4.2 The TCTR Data Bank . . . 6
4.3 The TCHX Data Bank . . . 9
4.4 The TCVX Data Bank . . . 9
4.5 The TCVT Data Bank . . . 9
4.6 The TCVP Data Bank . . . 11
4.7 The TPWC Data Bank . . . 12

5 User Access to the Locater Output **14**
5.1 SUBROUTINE BCLDD . . . 15
5.2 SUBROUTINE BCLDS . . . 15
5.3 SUBROUTINE CJOORD . . . 15
5.4 SUBROUTINE TCBARL . . . 15
5.5 SUBROUTINE TCBARX . . . 16
5.6 SUBROUTINE TCHLDD . . . 16
5.7 SUBROUTINE TCHLDS . . . 16
5.8 SUBROUTINE TCKFT3 . . . 17
5.9 SUBROUTINE TCKFT4 . . . 17
5.10 SUBROUTINE TCOORD . . . 19
5.11 SUBROUTINE TCPRNT . . . 20
5.12 SUBROUTINE TCRHIT . . . 20
5.13 SUBROUTINE TCRSLT . . . 21
5.14 SUBROUTINE TCVERS . . . 21
5.15 SUBROUTINE TCVHLD . . . 22
5.16 SUBROUTINE TCVHLS . . . 22
5.17 SUBROUTINE TCVLDD . . . 23
5.18 SUBROUTINE TCVLDS . . . 23

6 The Log File From Locater **23**

1 Introduction

This Users Manual is meant to explain all details of the LOCATER code as need by the *general* user. By general, I mean anyone not involved in either calibration or tracking software development. The manual starts by explaining how to create a version of LOCATER. Next, all input parameters necessary for normal running are explained, and then the output from this is clarified. Finally, a set of routines which are useful to access information stored in the data banks are described.

LOCATER is the track finding and track fitting portion of the Crystal Barrel Offline software. The code is approximately twenty–five thousand lines long, of which approximately fifteen thousand are in–line comments. The code accepts as input either processed or unprocessed data from the Flash ADC's attached to the JDC and the output data from the PICOS III attached to the PWC. The code then converts the raw data into space coordinates. These space coordinates are then passed to a pattern recognition package which locates tracks in the chambers. The tracks are then fit using simple circle parameterizations in the x–y plane. The program then tries to merge these fit tracks into longer tracks, and pick up unused points in the chamber. Once all of this is accomplished, the tracks are given to a full helix fit for simultaneously fitting the x, y and z coordinates. The helix fit also *improves* all the measured points so that they lie on the fit helix. Finally, the output of the helix fit are passed to a simple vertex finding and vertex fitting routine. The final output from LOCATER are then a set of *helix* banks, and a set of *vertex* banks. These will be described later.

2 Building the Locater code

Since version 1.42, LOCATER has been supported and maintained within the CMZ program, (Code Manager under Zebra). The released Locater card file can be taken into CMZ using the following commands.

```
cmz
CMZ[0] make locater
locater[1] ytoc locater.car
locater[2] exit
```

Updates to the LOCATER.CMZ file are distributed as CMZ cradles, (LOCATER.CRA). To update a local version of LOCATER, do the following under CMZ.

```
cmz
CMZ[0] file locater
locater[1] use locater.cra
locater[2] update *
locater[3] exit
```

To create the Fortran code for LOCATER, compile the code, and put it into a library, do the following.

```
cmz
CMZ [0] file locater -R
locater [1] set  locater.for -F
locater [2] set  locater.lib -L
locater [3] set  locater.exec -XACLD
locater [4] sel  machine_flag, (i.e. ALT,DECS,IBM,SUN or VAX)
locater [5] pilot *LOCATE
locater [6] set  calibration_set (i.e. dec_89_1, jun_90_1, jul_90_1)
```

```
locater [7] seq  commcb
locater [8] cflib -P
locater [9] exit
```

Machine dependent .KUMAC files for doing the above are available in the INSTRUCT PATCH within the code. The above could also be done by the following:

```
cmz
CMZ [0] file locater
locater [1] cd instruct
locater [2] select machine_flag (i.e. ALT,DECS,IBM,SUN or VAX)
locater [3] set locater.kumac -D
locater [4] ctot -Y kumac
locater [5] release locater
CMZ [6] exec locater.kumac
CMZ [7] exit
```

As the LOCATER code is distributed as a PATCHY CARD file, LOCATER.CAR, it is still possible to create the code using PATCHY. The following cradle file will produce the usual Fortran source code for LOCATER. This cradle can be found in the INSTRUCT PATCH, in the CRADLE DECK.

```
+USE,machine-flag.
+USE,*LOCATE.
+EXE.
+PAM,11,R=COMMCB,T=ATTACH,CARDS.    CBOFF.CAR
+PAM,12,T=ATTACH,CARDS.             LOCATER.CAR
+QUIT.
```

In order to generate the FORTRAN code, one need only issue the command:

```
YPATCHY - LOCATER LOCATER.CRA .GO
```

This code can then be compiled and linked into the offline software. If for some reason, the user's compiler insists on *standard* FORTRAN–77, the following should be added to the cradle.

```
+USE,F77.
```

Machine flags supported by LOCATER are as follows.

- ALT Alliant FX/8 for the Alliant fxc fortran compiler.
- DECS For Dec workstations running ULTRIX.
- IBM For IBM-CMS and IBM-MVS computers.
- NXT For a NeXt station.
- SUN For the SUN Work stations, f77 fortran compiler.
- UNIX Generic Unix Flag.
- VAX For VAX/VMS computers, including ALPHA/AXP.

For a description of all available pilots in LOCATER, consult the long write up, (CB–Note 93).

3 Input to Locater

The input to Locater is of two forms. The first is a data file containing calibration information which needs to be read in. The second consist of two input card files. The data files will eventually be contained in a data base, but until that time the user need to make sure that they are properly assigned. Since October 1990, all calibration files are handled by the Crystal Barrel Data Base program.

3.1 External Calibration Files

As well as the LOCATER.PAM and LOCATER.CRA file, there are several *gain* files corresponding to various run periods. There is also a file for tracking Monte Carlo data. The user needs to make sure that the correct gain file is externally assigned to logical unit 82. Since October 1990, all calibrations are correctly handled by the data base program. It is no longer necessary to have the external gain files.

3.2 Steering of Locater in CBOFF

In the general card file for CBOFF, there is one card for steering LOCATER, (see the Offline Reconstruction Software manual by Gunter Folger). This is the data card **CHAM** which can have the following arguments:

'TRAK' turn on track reconstruction.

'GPWC' include the PWC data in reconstruction

'RAWS' unpack the raw JDC data into a **TJDC** data bank. This is performed in the subroutine TJDCGT.

'PATT' do pattern recognition on the data, (done in the TCPATT subroutine).

'CIRC' Perform a circle fit to the pattern recognized tracks, (done in the TCCIRC routine).

'HELX' Perform a helix fit to the found tracks, (done in the TCHELX routine).

'VERT' Perform a vertex fit to the found tracks, (done in the TCVERT routine).

'RTRK' Enable retracking of charged data.

The order of the arguments does not matter. The most common cases are:

1 No **CHAM** card in file. If you are looking at a raw data tyape, this will perform full tracking. If you are looking at a DST, the results will vary. This is not recomended for non production jobs.

2 **CHAM 'NONE'** No tracking is performed.

3 **CHAM 'TRAK' 'RTRK' 'GPWC' 'RAWS' 'PATT' 'CIRC' 'HELX' 'VERT'** performs full retracking of an already tracked data sample.

It is also possible to do various levels of retracking of a DST. However, some care may be needed. On a DST, only the **RJDC**, **TCTR**, **TCHX** and **TCVX** banks are written, (as described in this manual). If no **CHAM** card is given, then the data in the **RJDC** and **RPWC** banks will be retracked through the circle fit, however the helix and vertex fits will remain. It is NOT possible to retrack the helix banks without retracking all the data. It is possible to only retrack the vertex banks. However, one needs to add the following block of code to the USEVNT routine on entry zero. This code disconnects the helix tracks from the fit vertex.

```
IF((LTCTR.GT.0).AND.(IQ(LTCTR+1).GT.0)) THEN
   DO ITRK = 1,IQ(LTCTR+1)
      ITCTR = LQ(LTCTR-ITRK)
      IQ(ITCTR+2) = 0
      IQ(ITCTR+3) = 0
   END DO
ENDIF
```

3.3 Internal Steering of Locater

There is a second, optional card file which can be assigned to logical unit 81. In this file, the user can modify many of the tracking parameters. For a full description, one should consult the CJINIT routine in the Chamber Reconstruction Software manual. Given here are only the cards for which the general user may have use. For Monte Carlo data, the user should not have this second card file; all default values in the program should be correct.

ANGL sets the value of ANGLTC, the rotation, in degrees, from the angle $\phi = 0°$ to the center of JDC sector one. **When using this card, there needs to be a second real number which is non–zero.**

BMAG sets the value of BMAGTC, the magnetic field strength in the JDC, (units are kG). If this value is changed, then the other parameters dependent upon this are also changed. **When using this card, there needs to be a second real number which is non–zero.**

DELZ sets the value of the 23 entries in DELZTJ. These are used as the 1σ errors in z. To use this card you must also enter a non–zero 24'th entry.

IAMP sets the value of IAMPTJ in the /TJCUTS/ common block. This is used in the TJDCGT routine to discard noise hits. **When using this card, there needs to be a second number which is non–zero.**

ITFC sets the value of ITFCRJ in the /RJPRMS/ common block. It is a t_0 offset for the fit pulses in RJPROC as a function of Flash ADC crate number, (1–16). **When using this card, one must include 17=1, or the program will ignore the input.**

OPWC sets the values of OPWCTP(1) and OPWCTP(2), the angle, in **radians**, from $\phi = 0$ to wire number zero of PWC one and PWC 2. **When using this card, there needs to be a third number which is non–zero.**

SY02

SYDF

ZOFF sets the value of ZOFFTC in the /TCPRMS/ common. This is the nominal z_0 as used in the tracking software.

BXPR allows the user to either set an absolute pressure, (with a positive pressure), or scale existin pressures, (with a negative argument). BXPR 708. 1. would force the pressure to 708 torr. BXPR -1.001 1.0 would multiply the database pressure by 1.001.

In general, the correct values are in the database, but under some conditions, strange results can arise when there is no card file. A safe and harmless one is:

```
STOP
```

For Monte Carlo data, the card should be something like:

```
STOP
```

4 Output From Locater

The output from LOCATER consists of a series of data banks described below, and several words in the event header. The event header is stored in the /CBHEAD/ common block and is described in the *Offline Reconstruction Software* manual. In particular, the following words are set by Locater.

- IEHDCB(9) is the number of charged tracks found.
- IEHDCB(11) is the number of long tracks in the JDC. A long track has at least ten hits.
- IEHDCB(15) is a coded vertex word. This is the number of long tracks at the primary vertex plus 100 if the charge at the vertex, (taken from the vertex banks) does not sum to zero plus 1000 if the charge at the primary vertex, (taken from the helix banks) does not sum to zero.
- IEHDCB(17) is the number of tracks at the primary vertex.
- IEHDCB(18) is the total number of hits found in the chamber.
- IEHDCB(19) is the number of hits that have been incorporated into tracks.

The information as stored in the data banks may not always be in the form desired by the user. As such, a series of *User Service Routines* as described in the next section have been written. In particular, for access to track momentums, the user should consult the TCRSLT routine.

To loop over all the data stored in the two helix banks, the following code can be used:

```
      IF((LTCTR .GT. 0).AND.(IQ(LTCTR+1).GT.0)) THEN
         NTRK = IQ(LTCTR+1)
         DO 1000 ITRK = 1,NTRK
            ITCTR = LQ(LTCTR-ITRK)
            ITCHX = LQ(LTCHX-ITRK)
*
*              The user's code goes here.
*
 1000    CONTINUE
      ENDIF
```

Similarly, to access the information in the vertex banks, the following code can be used:

```
      IF((LTCVX .GT. 0).AND.(IQ(LTCVX+1) .GT. 0)) THEN
         NVRTX = IQ(LTCVX+1)
*
*           Loop over the found verticies.
*
         DO 1000 IVRTX = 1,NVRTX
            ITCVT = LQ(LTCVX-IVRTX)
            ITCVP = LQ(ITCVT-1)
*
*              Loop over the tracks at this vertex.
*
            DO 900 IT = 1,IQ(ITCVT+1)
               ITRK = IQ(ITCVP+1+LENVP*(IT-1))
               ITCTR = LQ(LTCTR - ITRK)
*
*                 Users' code goes here.
```

```
*
  900       CONTINUE
*
 1000    CONTINUE
      ENDIF
```

4.1 The HTJD Data Bank

This is the top level tracking data bank. All output banks from Locater are actually down links from this bank. The basic struture is shown in table 1.

offset	TYPE	*Quantity*
LQ(LHTJD-7)	INTEGER	*Link to the TCVX bank*
LQ(LHTJD-6)	INTEGER	*Link to the TCHX bank*
LQ(LHTJD-5)	INTEGER	*Link to the TCTR bank*
LQ(LHTJD-4)	INTEGER	*Link to the TCTK bank*
LQ(LHTJD-3)	INTEGER	*Link to the TCHT bank*
LQ(LHTJD-2)	INTEGER	*Link to the TPWC bank*
LQ(LHTJD-1)	INTEGER	*Link to the TJDC bank*
IQ(LHTJD+1)	INTEGER	*Locater Version Number*
IQ(LHTJD+2)	INTEGER	*Date of Bank Creation*
IQ(LHTJD+3)	INTEGER	*Time of Bank Creation*

Table 1: The data and links stored in the the **HTJD** bank.

4.2 The TCTR Data Bank

The **TCTR** bank structure contains all the tracking results for each track located in the chambers. This bank is a master bank over the **TCTX** data banks, (one **TCTX** for each track). The data stored in each **TCTX** bank is shown in Table 2. The **TCTR** banks are lifted and filled in the helix fitting section of the code. Normally, all of these banks are generically referred to as the **TCTR** banks. The header word of theses sub–banks also contains some possibly useful information. If the track crossed a sector boundary, then bit 2 is set two 1. Otherwise it is set to zero.

The helices are parametrized as a function of β in the following form.

$$\begin{aligned} x(\beta) &= r_0 \cdot \sin\psi_0 + \frac{1}{\alpha} \cdot (\cos\beta + s \cdot \sin\psi_0) \\ y(\beta) &= -r_0 \cdot \cos\psi_0 + \frac{1}{\alpha} \cdot (\sin\beta - s \cdot \cos\psi_0) \\ z(\beta) &= z_0 - \frac{s \cdot \tan\lambda}{\alpha} \cdot \left(\beta - \psi_0 - s \cdot \frac{\pi}{2}\right) \end{aligned}$$

The value of the parameter β at the point of closest approach to the origin is defined as $\beta_0 = \psi_0 + s \cdot \frac{\pi}{2}$. If one wants to step through the helix, then $\beta = \beta_0 - s \cdot \Delta\beta$ where $\Delta\beta$ is positive, (if $s < 0$ then $\Delta\beta > 0$ and if $s > 0$ then $\Delta\beta < 0$). The value of r_0 can, and often is less than zero. The sign convention is arranged such that if during the helix fit, the charge of the particle changes, only the parameter α will change sign. All other parameters will retain their signs. However, the parameter α will always be positive. If it does change sign during the fit, then at the end of the fit, the value of s is changed so that α is again positive.

offset	TYPE	*Quantity*
+1	INTEGER	*Nhits*
+2	INTEGER	*NPED*
+3	INTEGER	*Vertex*
+4	INTEGER	$Layer_1$
+5	INTEGER	$Layer_N$
+6	INTEGER	*Error code*
+7	REAL	*Charge*
+8	INTEGER	NPWC
+9	REAL	dE/dx
+10	REAL	$\sigma_{dE/dx}$
+11	REAL	r_0
+12	REAL	z_0
+13	REAL	α
+14	REAL	$\tan\lambda$
+15	REAL	ψ_0
+16	REAL	s
+17	REAL	χ^2
+18	REAL	$C_\xi[1,1]$
$\vdots$	$\vdots$	$\vdots$
+32	REAL	$C_\xi[5,5]$

Table 2: The data stored in the subbanks of the **TCTR** bank, (**TCTX**). There is one **TCTX** data bank for each track found in the chambers.

In order to convert the six helix parameters, $(r_0, z_0, \alpha, \tan\lambda, \psi_0; s)$ to momentum, the following transformations should be used.

$$\begin{aligned} p_\perp &= \frac{e \mid B \mid}{\alpha} \\ p &= p_\perp \cdot \sqrt{1+\tan^2\lambda} \\ p_x &= p_\perp \cdot \cos\psi_0 \\ p_y &= p_\perp \cdot \sin\psi_0 \\ p_z &= p_\perp \cdot \tan\lambda \\ q &= +s \cdot \frac{B}{\mid B \mid} \end{aligned}$$

and the point of closest approach to the origin is given as

$$\begin{aligned} x &= r_0 \cdot \sin\psi_0 \\ y &= -r_0 \cdot \cos\psi_0 \\ z &= z_0 \end{aligned}$$

The value of eB is available in the /TCPRMS/ common block as the variable BMGCTC while the value of $\frac{B}{|B|}$ is available as BSGNTC. The contents of table 2 are described as follows.

- *Nhits* is the total number of hits incorporated into this track, (JDC and PWC).
- *NPED* is the number of the PED to which this track points. It is filled in global tracking.
- *Vertex* is the number of the vertex from which this track originated.

- *Layer*$_1$ is the layer number of the first hit in this track.
- *Layer*$_N$ is the layer number of the last hit in this track.
- *Error code* Is a combination of the fit error in the TCTK bank, and the error returned from the TCHELX routine. It's value is the TCTK error code plus the following:

  0 Completely normal convergence in the TCHELX routine.

  10 The TCHELX routine was not implemented as there were only 3 points found in the track.

  30 The χ^2 in the TCHELX routine converged to a value larger than the allowed cutoff.

  40 The χ^2 in the TCHELX routine began to diverge.

  50 The maximum number of iterations was exceeded.

  70 An attempt was made to invert a singular matrix.

- *Charge* is the electric charge of this particle, ± 1.
- *N*PWC is the number of PWC hits attached to this track.
- *dE/dx* is the average energy loss per centimeter of track length.
- $\sigma_{dE/dx}$ is the error in the above dE/dx.
- r_0 is the distance of closest approach of the helix to the z–axis. It is possible for this number to be negative.
- z_0 is the z–coordinate at the radius r_0.
- α is the curvature of the track. This number is positive.
- $\tan\lambda$ is the tangent of the opening angle of the track.
- ψ_0 is direction angle from the circle fit.
- s is a sign parameter of the track. It is determined by looking at the angle β_0, the angle as measured from the center of the circle describing the track to the point of closest approach, (r_0, z_0). $\beta_0 = \psi_0 + s \cdot \frac{\pi}{2}$.
- χ^2 is the returned the χ^2 of the helix fit with $3 \cdot Nhits - 5$ degrees of freedom.
- C_ξ is the covariance matrix for the above six parameters. This is a 5 by 5 symmetric matrix which is stored as follows.

$$\begin{pmatrix} +18 & * & * & * & * \\ +19 & +20 & * & * & * \\ +21 & +22 & +23 & * & * \\ +24 & +25 & +26 & +27 & * \\ +28 & +29 & +30 & +31 & +32 \end{pmatrix} = \begin{pmatrix} \sigma_{rr} & \sigma_{rz} & \sigma_{r\alpha} & \sigma_{r\lambda} & \sigma_{r\psi} \\ \sigma_{rz} & \sigma_{zz} & \sigma_{z\alpha} & \sigma_{z\lambda} & \sigma_{z\psi} \\ \sigma_{r\alpha} & \sigma_{z\alpha} & \sigma_{\alpha\alpha} & \sigma_{\alpha\lambda} & \sigma_{\alpha\psi} \\ \sigma_{r\lambda} & \sigma_{z\lambda} & \sigma_{\alpha\lambda} & \sigma_{\lambda\lambda} & \sigma_{\lambda\psi} \\ \sigma_{r\psi} & \sigma_{z\psi} & \sigma_{\alpha\psi} & \sigma_{\lambda\psi} & \sigma_{\psi\psi} \end{pmatrix}$$

4.3 The TCHX Data Bank

The **TCHX** data bank contains the coordinates of all hits used in the helix fits, (PWC and JDC). The bank structure is one **TCHX** bank to which is attached one **TCHP** subbank for every track. The data stored in this bank are the improved coordinates from the helix fit. Also, PWC and JDC hits have equal footing here. The first NPWC entries will be the PWC hits. Because these coordinates are improved during the helix fit, one should not perform the helix fit twice. After a second iteration, the errors will be nonsense. To prevent this from happening, the routine TCHXLD sets the lowest order bit of the status word, IQ(LTCHP), to 1. The routine will not refit tracks with this bit set. The contents of the **TCHP** bank follows, where n is the number of hits in the track, which is taken from the corresponding **TCTR** data bank. These banks are generically referred to as the **TCHX** data banks.

Offset	TYPE	*Quantity*
$+1$	REAL	x_1
$+2$	REAL	y_1
$+3$	REAL	z_1
$+4$	REAL	σ^2_{x1}
$+5$	REAL	σ^2_{y1}
$+6$	REAL	σ^2_{z1}
$\vdots$	$\vdots$	$\vdots$
$+6n-5$	REAL	x_n
$+6n-4$	REAL	y_n
$+6n-3$	REAL	z_n
$+6n-2$	REAL	σ^2_{xn}
$+6n-1$	REAL	σ^2_{yn}
$+6n-0$	REAL	σ^2_{zn}

Table 3: The data stored in the **TCHX** data banks. All hits for one track are stored in their own bank.

- (x, z, y) are the coordinates of the points used in this track. These points have been improved by the helix fit. They will not refer to exactly the same point as in the **TCHT** data banks.
- $(\sigma_x, \sigma_y, \sigma_z)$ are the errors is the used coordinates. These are not the input errors, but the errors coming from the helix fit.

4.4 The TCVX Data Bank

The **TCVX** data bank is a *steering* bank to all found vertices in the event. It contains one data word, which is the number of found vertices, and then one pointer to the **TCVT** bank for each vertex. Access to the vertex information is shown in figure 1.

4.5 The TCVT Data Bank

The **TCVT** data banks are subbanks to the **TCVX** data bank, and contain information regarding the found vertices. The data are shown in table 4.

- *Ntrks* is the number of tracks fit to this vertex. The set of pointers to each of these tracks in the **TCTR** bank is in the pointer section of this bank.

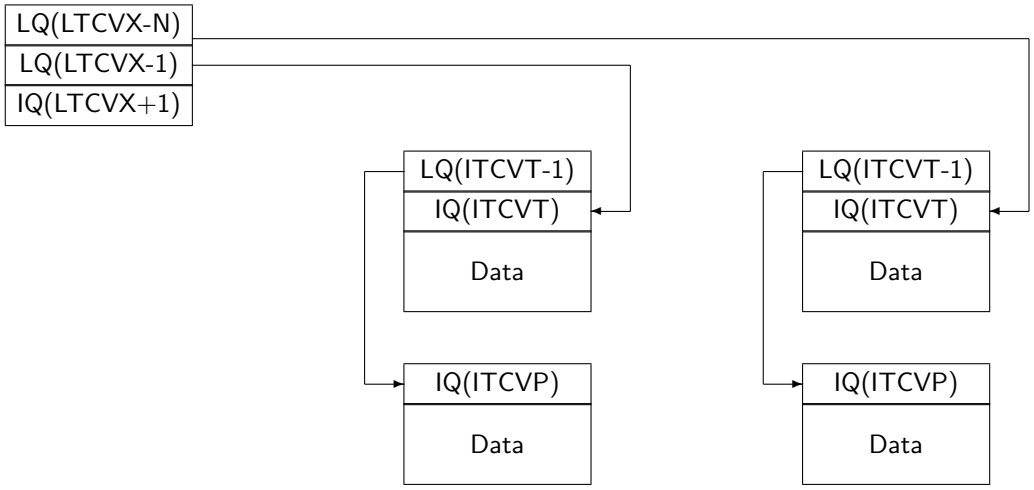


Figure 1: Layout of the bank structure containing all found vertices.

Offset	TYPE	*Quantity*
+1	INTEGER	*Ntrks*
+2	INTEGER	*Nneutral*
+3	INTEGER	*Error code*
+4	INTEGER	N_{DF}
+5	REAL	$x[cm]$
+6	REAL	$y[cm]$
+7	REAL	$z[cm]$
+8	REAL	C_x
⋮	⋮	⋮
+14	REAL	χ^2

Table 4: The data stored in the **TCVT** data bank. There is one bank for every vertex.

- *Nneutral* is the number of neutral tracks assigned to this vertex. This number is filled in global tracking, and is not available within the context of LOCATER. At present, there is no list available as to which PEDS these are.
- *Ierr* is an error code returned from the TCVRTX routine for this vertex. It has the following meanings:

  0 Normal convergence occurred.

  100 Only one track fit to this vertex.

  300 The iterative routine converged with a too large χ^2.

  400 The χ^2 started to diverge for this track.

  500 The routine did not converge in the allowed number of iterations.

  600 The number of tracks to fit was out of range.

  700 The routine tried to invert a singular matrix.
- N_{DF} is the number of degrees of freedom from the fit.
- x is the x coordinate of the fit vertex, [cm].
- y is the y coordinate of the fit vertex, [cm].
- z is the z coordinate of the fit vertex, [cm].
- C_x contains the 6 unique elements of the 3 by 3 symmetric covariance matrix for $\vec{x}$. They are stored as follows:

$$\begin{pmatrix} +8 & * & * \\ +9 & +10 & * \\ +11 & +12 & +13 \end{pmatrix} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix}$$

- χ^2 is the resulting χ^2 of the fit with N_{DF} degrees of freedom.

4.6 The TCVP Data Bank

The **TCVP** data bank contains fit momentum information for all tracks coming from the corresponding vertex. The **TCVP** bank is a subbank under the **TCVT** data bank desribing the vertex. It contains the following information for every track. The length of each block is given by the parameter LENVP obtained using +CDE,TRKPRM..

- *Track number* is the track number from the **TCTR** data bank.
- *Quality word* is the track length in hits plus one hundred times the first layer of the track plus ten thousand times the error code from the helix fit.
- *Charge* is the electric charge of this particle. It is possible for this to be different from the charge in the **TCTR** bank, as the vertex fit is allowed to change the charge of a track.
- p_x is the improved x–component of momentum.
- p_y is the improved y–component of momentum.
- p_z is the improved z–component of momentum.
- E is the energy under the assumption the particle is a pion.

Offset	TYPE	*Quantity*
+1	INTEGER	*Track Number in* **TCTR**
+2	INTEGER	*Quality Word*
+3	REAL	Charge of particle.
+4	REAL	p_x *[MeV/c]*
+5	REAL	p_y *[MeV/c]*
+6	REAL	p_z *[MeV/c]*
+7	REAL	*E [MeV]*
+8	REAL	$\sigma^2[px]$ *[Mev/c]*^{2}
+9	REAL	$\sigma[p_{xy}]$ *[Mev/c]*^{2}
+10	REAL	$\sigma^2[p_y]$ *[Mev/c]*^{2}
+11	REAL	$\sigma[p_{xz}]$ *[Mev/c]*^{2}
+12	REAL	$\sigma[p_{yz}]$ *[Mev/c]*^{2}
+13	REAL	$\sigma^2[p_z]$ *[Mev/c]*^{2}
⋮	⋮	⋮

Table 5: The data stored in the subbank of the **TCVT** data bank, (the **TCVP** bank). The above information is repeated for every track.

- The remaining six terms are the unique elements of the three by three covariance matrix for the momentum.

$$\begin{pmatrix} +8 & * & * \\ +9 & +10 & * \\ +11 & +12 & +13 \end{pmatrix} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix}$$

4.7 The TPWC Data Bank

The **TPWC** data bank contains the position information from each hit in the PWC's, and from each cluster in the PWC. The hit information is stored in two **TPCH** subbanks under the **TPWC** bank. These banks are attached at the -1 and -2 downlinks. The format of this information is given in table 6.

Offset	TYPE	*Quantity*
+1	INTEGER	*Track number*
+2	INTEGER	*Wire number*
+3	REAL	x_i [cm]
+4	REAL	y_i [cm]
+5	REAL	z_i [cm]
+6	REAL	r_i [cm]
+7	REAL	ϕ_i [radians]
+8	REAL	σ_x [cm]
+9	REAL	σ_y [cm]
+10	REAL	σ_ϕ [radians]

Table 6: The data stored in the two subbanks of the **TPWC** data bank, (**TPCH**). This format is repeated for every cluster each chamber.

- *Track number* refers to the track to which this hit is connected.
- *Wire number* is the number of the fired wire.

- The x coordinate as determined from the wire number.
- The y coordinate as determined from the wire number.
- The z coordinate as projected by the JDC.
- r is the radius of the wire.
- The ϕ position of the wire.
- σ_x is the error in the x position.
- σ_y is the error in the y position.
- σ_ϕ is the error in the ϕ angle.

To obtain the number of hits found in each PWC, and then extract the hit data, use the following code. (Note that the parameter LENPW is obtained using the +CDE,TRKPRM.)

```
      IF(LTPWC .GT. 0) THEN
         NHIT1 = IQ(LTPWC+1)
         NHIT2 = IQ(LTPWC+2)
         IF(NHIT1 .GT. 0) THEN
            ITPWC = LQ(LTPWC-1)
            DO 1000 IHIT = 1,NHIT1
               XHIT = Q(ITPWC+3)
               YHIT = Q(ITPWC+4)
               ZHIT = Q(ITPWC+5)
                   ...
               ITPWC = ITPWC + LENPW
 1000       CONTINUE
         ENDIF
*
*           Repeat the above for the outer chanber if desired.
*
      ENDIF
```

The cluster information is contained in two **TPCL** banks attached to the -3 and -4 down links of the **TPWC** bank. The format of these cluster banks are given in table 7. It is important to note that the cluster information is linked to the JDC, and not the hit information.

- *Track number* refers to the track to which this hit is connected.
- *Wire 1* is the first wire in the cluster.
- *Cluster Size* is the number of wires in the cluster.
- *Central Wire Number* is the *pseudo* wire at the center of the cluster.
- The x coordinate as determined from the wire number.
- The y coordinate as determined from the wire number.
- The z coordinate as projected by the JDC.
- r is the radius of the wire.

Offset	TYPE	*Quantity*
+1	INTEGER	*Track number*
+2	INTEGER	*Wire 1*
+3	INTEGER	*Cluster Size*
+4	REAL	*Central Wire Number*
+5	REAL	x_i [cm]
+6	REAL	y_i [cm]
+7	REAL	z_i [cm]
+8	REAL	r_i [cm]
+9	REAL	ϕ_i [radians]
+10	REAL	σ_x [cm]
+11	REAL	σ_y [cm]
+12	REAL	σ_ϕ [radians]

Table 7: The data stored in the two subbanks of the **TPWC** data bank, (**TPCL**). This format is repeated for every cluster each chamber.

- The ϕ position of the wire.
- σ_x is the error in the x position.
- σ_y is the error in the y position.
- σ_ϕ is the error in the ϕ angle.

To obtain the number of clusters found in each PWC, and then extract the cluster data, use the following code. (Note that the parameter LENCL is obtained using the +CDE,TRKPRM.)

```
      IF(LTPWC .GT. 0) THEN
         NCL1 = IQ(LTPWC+3)
         NCL2 = IQ(LTPWC+4)
         IF(NCL1 .GT. 0) THEN
            ITPWC = LQ(LTPWC-3)
            DO 1000 IHIT = 1,NCL11
               XHIT = Q(ITPWC+5)
               YHIT = Q(ITPWC+6)
               ZHIT = Q(ITPWC+7)
                   ...
               ITPWC = ITPWC + LENCL
 1000       CONTINUE
         ENDIF
*
*           Repeat the above for the outer chanber if desired.
*
      ENDIF
```

5 User Access to the Locater Output

The following routines have been provided to allow the user easier access to the data stored in the tracking data banks. It is of course possible, and usually faster to explicitly access the data, however often the user is interested in getting ahold of the data without actually knowing where it is stored.

For this reason, the following routines have been written. The consist of two classes of routines; one which prints tracking data to a specified logical unit, and one which returns fit information to the user. Except in special cases, (debug versions and calibration versions), these routines are not actually called by any of the tracking software. However, there use in USER routines is recommended to avoid errors in addressing the bank structures.

5.1 SUBROUTINE BCLDD

Author: Brigitt Schmid
Creation Date: May, 1990
References:
Call Arguments: (NPED, *DY, *DCOVR, *IERR).
Common Blocks Used: CBBANK and CBLINK.
Subroutines Referenced: None.

This routine will extract the double precision 3–momentum, (p_x, p_y, p_z), and the 3 by 3 covariance matrix for ped number NPED. The momentum is returned in the vector DY, and the covariance matrix is returned in the 3 by 3 array DCOVR. The value of IERR is returned as zero upon successful completion. For single precision values, conslut the BCLDS subroutine. It is important to observe that the 3 by 3 covariance matrix is not diagonal in the these coordinates. In order for the TCKFT3 and TCKFT4 routines to work, it is necessary to load the photons using this routine.

5.2 SUBROUTINE BCLDS

Author: Brigitt Schmid
Creation Date: May, 1990
References:
Call Arguments: (NPED, *SY, *SCOVR, *IERR).
Common Blocks Used: CBBANK and CBLINK.
Subroutines Referenced: None.

This is just a single precision entry point to the BCLDD routine. Consult the BCLDD routine for a description.

5.3 SUBROUTINE CJOORD

Author: Curtis A. Meyer
Creation Date: 15 January, 1989
References:
Call Arguments: (ITRK, *NPTS, *ISEC, *ILYR, *ISID, *TIME, *XTJ, *YTJ, *XTC, *YTC).
Common Blocks Used: CBBANK, CBLINK and TCPRMS.
Subroutines Referenced: None.

This routine will return the sector number, layer number, resolution code, drift time, x and y coordinates from the **TJDC** bank, and x and y coordinates from the **TCHT** data bank for the NPTS points in track number ITRK.

5.4 SUBROUTINE TCBARL

Author: Curtis A. Meyer
Creation Date: 26 April, 1989

References:
Call Arguments: (*NTRKS, *IFLAG, *XTRK, *YTRK, *ZTRK).
Common Blocks Used: CBBANK, CBLINK.
Subroutines Referenced: None.

This routine will return the intersection point, (x,y,z) of every track in the **TCTR** bank. NTRKS is the number of tracks, and the three vectors contain the coordinates of each of the NTRKS tracks. The returned value of IFLAG indicates if it is safe to use this projection. A value of zero means it is good, and any other value is at the users own risk. From version 1.40 on, this routine is just a second entry point to the TCBARX routine.

5.5 SUBROUTINE TCBARX

Author: Curtis A. Meyer
Creation Date: 15 March, 1990
References:
Call Arguments: (*NTRKS, *IFLAG, *XTRK, *YTRK, *ZTRK, *DXTRK, *DYTRK, *DZTRK).
Common Blocks Used: CBBANK, CBLINK.
Subroutines Referenced: None.

This routine will return the intersection point, (x,y,z) of every track in the **TCTR** bank with the barrel. It also returns a direction vector at the intersection point, (dx/dr,dy/dr,dz/dr). NTRKS is the number of tracks in the helix bank, and the three vectors contain the coordinates of each of the NTRKS tracks. The returned value of IFLAG indicates if it is safe to use this projection. A value of zero means it is good, and any other value is at the users own risk. Please not that the direction vector is the direction in which the track is traveling at the point of intersection. It is **not** the direction cosines of the track in the **TBTK** sense. The direction cosines in the **TBTK** sense can be obtained by using the vertex psotion, and the (x,y,z) coordinates of the intersection point. The purpose of the direction vector is to aide in connecting when the intersetion point is not near a PED, or near many PED's.

5.6 SUBROUTINE TCHLDD

Author: Curtis A. Meyer
Creation Date: 21 April, 1990
References:
Call Arguments: (ITRK, *PMOM, *COVR, *IERR).
Common Blocks Used: CBBANK, CBLINK and TCPRMS.
Subroutines Referenced: None.

This subroutine will compute the momentum vector, PMOM as (p_x, p_y, p_z) and the three by three covariance matrix for the momentum, COVR for track number ITRK as stored in the **TCTR** data bank. If the routine is unable to compute these, then the value of IERR is returned as one; successful completion has IERR of zero. The returned values of PMOM and COVR are double precison. For single precision values, see the TCHLDS subroutine.

5.7 SUBROUTINE TCHLDS

Author: Curtis A. Meyer
Creation Date: 21 April, 1990

References:
Call Arguments: (ITRK, *PMOM, *COVR, *IERR).
Common Blocks Used: CBBANK, CBLINK and TCPRMS.
Subroutines Referenced: None.

This subroutine will compute the momentum vector, PMOM as (p_x, p_y, p_z) and the three by three covariance matrix for the momentum, COVR for track number ITRK as stored in the **TCTR** data bank. If the routine is unable to compute these, then the value of IERR is returned as one; successful completion has IERR of zero. The returned values of PMOM and COVR are single precison. For double precision values, see the TCHLDD subroutine. (Note, this is not a separate subroutine, but rather an entry point to the TCHLDD subroutine.)

5.8 SUBROUTINE TCKFT3

Author: Curtis A. Meyer
Creation Date: 20 April, 1990.
References:
Call Arguments: (NPART, PMOM, COVR, *CHISQ, *CHI, *IERR).
Common Blocks Used: None.
Subroutines Referenced: CERNLIB: MATIN2.

This subroutine will perform a kinematic fit to the constraints of three-momentum balance for the NPART particles whose three momentum, (p_x, p_y, p_z) are passed in the PMOM array, and whose three by three covariance matricies are passed in the COVR array. Internally, PMOM and COVR are dimensioned as:

```
REAL    PMOM(3,NPART),COVR(3,3,NPART),CHI(NPART)
```

The routine will return the improved values of momentum in the PMOM variable, the χ^2 for three degrees of freedom in the variable CHISQ, the contribution to CHISQ from each of the NPART particles in the CHI array, and an error code IERR. If IERR is not reurned as zero, then the fit has failed.

The routine uses the follwoing three equations of constraint:

$$\begin{aligned} 0 &= \sum_{i=1}^{n} p_{x_i} \\ 0 &= \sum_{i=1}^{n} p_{y_i} \\ 0 &= \sum_{i=1}^{n} p_{z_i} \end{aligned}$$

and does a kinematic fit to the momentum using the passed covariance matricies. The passed variables are all single precision, but internally, the routine works in double precision.

5.9 SUBROUTINE TCKFT4

Author: Curtis A. Meyer
Creation Date: 20 April, 1990.
References:
Call Arguments: (NPART, *PMOM, *COVR, MASS, *CHISQ, *CHI, *IERR).
Common Blocks Used: None.
Subroutines Referenced: CERNLIB: MATIN2.

This subroutine will perform a kinematic fit to the constraints of three-momentum and energy balance for the NPART particles whose three momentum, (p_x, p_y, p_z) are passed in the PMOM array, whose three by three covariance matricies are passed in the COVR array and whose masses are passed in the MASS array, (units of MeV/c^2). Internally, the passed variables are dimensioned as:

```
REAL    PMOM(3,NPART),COVR(3,3,NPART),MASS(NPART),CHI(NPART)
```

The routine will return the improved values of momentum in the PMOM variable, the χ^2 for four degrees of freedom in the variable CHISQ, the contribution to CHISQ from each of the NPART particles in the CHI array, an improved covariance matrix in the array COVR and an error code IERR. If IERR is not reurned as zero, then the fit has failed. The value of IERR then indicates the reason for failure as follows.

- IERR=−1 Loading problem, (more than twenty tracks).
- IERR=0 Successful completion.
- IERR=3 Iterative procedure diverged.
- IERR=5 Iteration limit, (10) exceeded.
- IERR=7 Attempted to invert a singular matrix.

The routine uses the follwoing four equations of constraint:

$$\begin{aligned} 0 &= \sum_{i=1}^{n} p_{x_i} \\ 0 &= \sum_{i=1}^{n} p_{y_i} \\ 0 &= \sum_{i=1}^{n} p_{z_i} \\ 0 &= -2 \cdot m_p + \sum_{i=1}^{n} \sqrt{p_{x_i}^2 + p_{y_i}^2 + p_{z_i}^2 + m_i^2} \end{aligned}$$

and does a kinematic fit to the momentum using the passed covariance matricies. The passed variables are all single precision, but internally all computations are done in double precision.

In order to facilitate use of this routine, the subroutines BCLDS, TCHLDS and TCVLDS have been provided to extract neutral and charged data from the correct data banks. The BCLDS routine will load photons from the **TBTK** into the working arrays. The routine TCHLDS will load individual tracks from the **TCTR** data bank into the working area, and the TCVLDS routine will load all tracks from the **TCVT** data bank.

An example for loading the data for a four prong event at vertex number 1, and two photons from PED numbers 7 and 8 is as follows. For two prong data, or if you want to take the helix bank momentum rather than the vertex bank, replace the call to TCVLDS with one to TCVHLS.

```
*
      INTEGER  NTRK,ICODE(20),IERR
      REAL     CHRG(20),PMOM(3,20),COVR(3,3,20),MASS(20),CHI(20),CHISQ
*
      REAL       MPI
      PARAMETER (MPI=139.5673)
```

```
*
*         Load the particles at the vertex using TCVLDS, then make sure
*         that the vertex and tracks are ok.
*
       CALL TCVLDS(1,NTRK,ICODE,CHRG,PMOM,COVR,IERR)
       IF(IERR .NE. 0 .OR. NTRK .NE. 4) GOTO 5000
       MASS(1) = MPI
       MASS(2) = MPI
       MASS(3) = MPI
       MASS(4) = MPI
       NTRK = NTRK + 1
*
*         Load the two PEDs using the BCLDS routine. Then make sure
*         that they are correctly loaded.
*
       CALL BCLDS(7,PMOM(1,NTRK),COVR(1,1,NTRK),IERR)
       IF(IERR .NE. 0) GOTO 5000
       MASS(NTRK) = 0.0
       NTRK = NTRK + 1
       CALL BCLDS(8,PMOM(1,NTRK),COVR(1,1,NTRK),IERR)
       IF(IERR .NE. 0) GOTO 5000
       MASS(NTRK) = 0.0
*
*         Perform the kinematic fit using the loaded data.
*
       CALL TCKFT4(NTRK,PMOM,COVR,MASS,CHISQ,CHI,IERR)
*
```

5.10 SUBROUTINE TCOORD

Author: Curtis A. Meyer
Creation Date: 19 December, 1988
References:
Call Arguments: (ICODE, ITRK, *NPTS, *X1, *X2, *X3).
Common Blocks Used: CBBANK, CBLINK, TCPRMS, and TCANGL.
Subroutines Referenced: None.

This routine will return the (x, y, z) coordinates of all hits in the specified track, or all found vertices. The call argument ICODE determines what is returned as described below, while the argument ITRK specifies which track to examine. The returned information consists of the number of returned data points, NPTS, and three coordinates for each point given in X1, X2 and X3. The returned values are specified as follows by ICODE.

- ICODE=0 The routine returns the coordinates $(x_l, y_l.z)$ for all points along the track. Where x_l and y_l are the left side resolution of every point on the track. These points are returned in CB–coordinates with units of [cm], however these are not necessarily the points chosen.

- ICODE=1 The routine returns the coordinates $(x_r, y_r.z)$ for all points along the track. Where x_r and y_r are the right side resolution of every point on the track. These points are returned in CB–coordinates with units of [cm], however these are not necessarily the points chosen.

- ICODE=2 The routine returns the chosen coordinates (x, y, z) along the track in CB–coordinates as taken from the **TCHT** data bank.
- ICODE=3 The routine returns the chosen coordinates (r, ϕ, z) along the track in CB–coordinates as taken from the **TCHT** data bank.
- ICODE=4 The routine returns the chosen coordinates (x, y, z) along the track in CB–coordinates as taken from the **TCTR** data bank.
- ICODE=5 The routine returns the coordinates of all found vertices as (x, z, y). These data are taken from the **TCVX** bank.

5.11 SUBROUTINE TCPRNT

Author: Curtis A. Meyer
Creation Date: 30 June, 1988
References:
Call Arguments: (LUN, IOPT).
Common Blocks Used: CBBANK, CBLINK, CBHEAD and TCPRMS.
Subroutines Referenced: None.

This routine prints out tracking results to logical unit LUN, (the log file). The printing is controlled through the passed variable IOPT, and is given as follows.

- IOPT=0 Print the Monte Carlo data if available.
- IOPT=1 The results of TCFITR and TCTHET as stored in the **TCTK** data bank are printed out.
- IOPT=2 The results of TCHELX as stored in the **TCTR** data bank are printed.
- IOPT=3 The results of TCVERT as stored in the **TCVX** data bank are printed.

5.12 SUBROUTINE TCRHIT

Author: Curtis A. Meyer
Creation Date: 3 March, 1989
References:
Call Arguments: (ITRK, *NPTS, *ISEC, *ILYR, *IRES, *TIME, *ALFT, *ARGT, *DEDX, *NTCHT).
Common Blocks Used: CBBANK and CBLINK.
Subroutines Referenced: None.

This routine will return the raw hit information on track ITRK. The returned variables have the following meanings.

- NPTS is the number of hits in the track.
- ISEC(50) is the sector number of each hit.
- ILYR(50) is the layer number of each hit.
- IRES(50) is the resolution code of each hit, (left/right).
- TIME(50) is the drift time of each hit.

- ALFT(50) is the $+z$ amplitude of each hit.
- ARGT(50) is the $-z$ amplitude of each hit.
- DEDX(50) is the dE/dx of each hit.
- NTCHT(50) is the address in the **TCHT** bank for each hit.

5.13 SUBROUTINE TCRSLT

Author: Curtis A. Meyer
Creation Date: 21 March, 1989
References:
Call Arguments: (ICODE, ITRK, *CHRG, *XVEC, *PVEC, *COVR, *SIGP, *IERR).
Common Blocks Used: CBBANK, CBLINK and TCPRMS.
Subroutines Referenced: None.

This routine will return information about fit tracks. The passed variable ICODE identifies if *circle* or *helix* fit results are desired. The variable ITRK specifies the number of the fit track. If ICODE is zero, then track data from the **TCTK** data banks are returned, while for ICODE of one, data from the **TCTR** banks are returned. The returned variables are as follows.

CHRG is returned as the charge of the particle.

XVEC is a vector containing five entries. The have the following meanings. See bank descriptions for **TCTK** and **TCTR** for clarification of the variables.

XVEC(1) ICODE=0, $q \cdot R$; ICODE=1 r_0.

XVEC(2) ICODE=0, ψ_0 ; ICODE=1 z_0.

XVEC(3) ICODE=0, c^2 ; ICODE=1 α.

XVEC(4) ICODE=0, $\tan\lambda$; ICODE=1 $\tan\lambda$.

XVEC(5) ICODE=0, a_0 ; ICODE=1 ψ_0.

PVEC is a vector containing three entries. They have the following values:

PVEC(1) $p_\perp$, the transverse momentum, [MeV/c].

PVEC(2) ψ_0, the direction angle in the $r-\phi$ plane.

PVEC(3) $p_\parallel$, the longitudinal momentum, [MeV/c].

COVR is a five by five array which contains the covariance matrix for XVEC.

SIGP is a vector of length three containing the 1σ errors for PVEC.

IERR is the error code associated with the track fit.

5.14 SUBROUTINE TCVERS

Author: Curtis A. Meyer
Creation Date: 21 March, 1989
References:
Call Arguments: (LUN, *IVERS).
Common Blocks Used: None.
Subroutines Referenced: None.

This subroutine identified the present version number of locater. If the value of LUN is given as a positive number, then the version information will be printed on logical unit LUN. In all cases the returned value of IVERS is the integer form of the version number. For LOCATER version *1.44/01*, the returned version number is *14401*. This number is stored in the **HTJD** bank during tracking. In order to determine under which version data was track, use the following code.

```
*
      IVERS = IQ(LHTJD+1)
      IDATE = IQ(LHTJD+2)
      ITIME = IQ(LHTJD+3)
*
```

5.15 SUBROUTINE TCVHLD

Author: Curtis A. Meyer
Creation Date: 19 September, 1990
References:
Call Arguments: (IVRT, *NTRK, *ICODE, *CHRG, *PMOM, *COVR, *IERR, LONGT).
Common Blocks Used: CBBANK, CBLINK and TCPRMS.
Subroutines Referenced: TCHLDS.

This subroutine will compute the momentum vector, PMOM as (p_x, p_y, p_z) and the three by three covariance matrix for the momentum, COVR for all tracks connected to vertex IVRT. The number of tracks is returned as NTRK, the vertex quality word is returned as ICODE, and the charge is returned as CHRG. The data is taken from the **TCTR** data bank using the only the tracks connected to the vertex.. If the routine is unable to compute these, then the value of IERR is returned as -1; otherwise the error code is set bitwise with the following meanings:

Bit 0 Set if the charge sum is not zero.

Bit 1 Set if all tracks are not long.

Bit 2 Set if a track error codes is not good.

Bit 3 Set if a track starts outside layer 5.

Bit 4 Set if the vertex fit is poor.

The passed value of LONGT is used to determine if a track is long or not. The returned values of PMOM and COVR are double precison. For single precision values, see the TCVLDS subroutine.

5.16 SUBROUTINE TCVHLS

Author: Curtis A. Meyer
Creation Date: 18 September, 1990
References:
Call Arguments: (IVRT, *ICODE, *NTRK, *CHRG, *PMON, *COVR, *IERR, LONGT).
Common Blocks Used: CBBANK, CBLINK and TCPRMS.
Subroutines Referenced: TCHLDS.

This routine is identical to TCVHLD, except that it returns single precision arguments. See the TCVHLD routine for a description.

5.17 SUBROUTINE TCVLDD

Author: Curtis A. Meyer
Creation Date: 21 April, 1990
References:
Call Arguments: (IVRT, *NTRK, *ICODE, *CHRG, *PMOM, *COVR, *IERR).
Common Blocks Used: CBBANK, CBLINK and TCPRMS.
Subroutines Referenced: None.

This subroutine will compute the momentum vector, PMOM as (p_x, p_y, p_z) and the three by three covariance matrix for the momentum, COVR for all tracks connected to vertex IVRT. The number of tracks is returned as NTRK, the vertex quality word is returned as ICODE, and the charge is returned as CHRG. The data is taken from the **TCVT** data bank. If the routine is unable to compute these, then the value of IERR is returned as -1; otherwise the error code is set bitwise with the following meanings:

Bit 0 Set if the charge sum is not zero.

Bit 1 Set if all tracks are not long.

Bit 2 Set if a track error codes is not good.

Bit 3 Set if a track starts outside layer 5.

Bit 4 Set if the vertex fit is poor.

The returned values of PMOM and COVR are double precison. For single precision values, see the TCVLDS subroutine.

5.18 SUBROUTINE TCVLDS

Author: Curtis A. Meyer
Creation Date: 21 April, 1990
References:
Call Arguments: (IVRT, *ICODE, *NTRK, *CHRG, *PMON, *COVR, *IERR).
Common Blocks Used: CBBANK, CBLINK and TCPRMS.
Subroutines Referenced: None.

This routine is identical to TCVLDD, except that it returns single precision arguments. See the TCVLDD routine for a description.

6 The Log File From Locater

This section is meant only as an explanation of what the output from LOCATER found in the log file means. The following ouput is generated by the TCVERS routine, which is called at the beginning of analysis, and only serves to identify the version of the code being used.

```
 ##############################################
 #                                            #
 # Crystal Barrel Charged Tracking Software #
 #               Version 2.00/00              #
 #         Release date 20 August, 1994.      #
 #                                            #
```

```
#    Code linked using the Sun patch.      #
#                                          #
############################################
PAM-File and Version = LOCATER 2.00/00 920820 0930
Last PATCHY run on 940820 at 1300.

Analysis date:21. 8.1994
Analysis time: 8:22
Cputime used :       0.114 Seconds.
```

The following may be generated printed by the TJGAIN routine. It is only a warning, and refers to a future implementation of calibration.

```
<TJGAIN> LOAD DEFAULT T-0 VALUES
```

The following may also be printed by the TJGAIN routine. It means that the *gain* file was not assigned to logical unit 82.

```
<TJGAIN> ERROR OPENING THE JDC GAIN FILE, IOSTAT=xxxxx
       or
*** ERROR *** JDC Z CALIBRATIONS NOT READ, IOSTAT=xxxxx
       or
*** ERROR *** JDC E CALIBARTIONS NOT READ, IOSTAT=xxxxx
       or
*** EOF ENCOUNTERED IN JDC GAIN FILE
       or
*** ERROR *** Z_0 VALUES NOT READ IN, IOSTAT=xxxxx
       or
*** ERROR *** JDC WIRE LENGTHS NOT READ IN, IOSTAT=xxxxx
```

The following is generated in the TJTIMI code, and identifies which calibration is being used.

```
==================================
JDC CALIBRATION TABLE 90.04.18.011
==================================
<TJTIMI> ==> TJCT Bank has been loaded .
```

The following is printed in the TCDONE routine, and is a tally of how many times the major tracking routines have been called. Note that for data which come from DST's, the number of calls to RJPROC will be zero.

```
Tracking terminating on code =     0
Completion date 21. 8.1994
Completion time 10: 6
CPU time used       2430.272 seconds.
General Tracking Statistics:
Calls to TCTRAK            :    10215
Calls to TPWPOS            :    10215
Calls to TJDCGT            :    10215
Calls to RJPROC            :    10215
Calls to CJCALB            :        0
Calls to TCSGMT            :    10026
Calls to TCRAW1            :     9931
```

```
Calls to TCRAW2            :     9931
Calls to TCFITR            :     9906
Calls to TCASSC            :     9906
Calls to TCSWEP            :     9839
Calls to TCTHET            :     9839
Calls to TCIFIX            :     9839
Calls to TPCNCT            :     9839
Calls to TCHELX            :     9839
Calls to TCMSCT            :     9839
Calls to TCVERT            :     9839
```

The following are general tracking statistics from the various sections of the code. The Aborted events have more than twelve tracks after pattern recognition, and are spiraling tracks in the JDC. The events with no tracks are those with nothing found in the pattern recognition section. It is not the *all neutral* events as the pattern recognition code can put noise hits together.

```
Number of tracks found     :    34042
=====================================
Aborted events after patt. :       25
Events with no tracks      :      162
Successful patt. recog.    :     9906
Avg. No of tracks from patt:   4.0541
Total tracks from patt.    :    40160
Total number of fit tracks :    40160
Avg. No. of hits per track :  13.0139
Successful circle fit      :     9906
Avg. Assc. Tracks per event:   3.4365
Total circle tracks        :    34042
Avg. No. of hits per track : 16.45406
```

The following represent the results of the circle fit. The value of IERR is explained in the description of the **TCTK** data bank in the *Chamber Reconstruction Software* manual. The following values are typical for one run. The tracks which are well fit here have IERR smaller than 3.

```
Number of IERR = 0         :    35761
          IERR = 1         :     4144
          IERR = 2         :        0
          IERR = 3         :     1607
          IERR = 4         :        1
          IERR = 5         :       20
          IERR = 6         :      497
          IERR = 7         :        0
          IERR = 8         :        0
          IERR = 9         :        0
Changed charge in circ. fit:     1028
```

The following come from the helix fit. The meanings of the error codes are described in the **TCTR** bank description. All tracks with error code less than 30 have been well fit.

```
Number of ierr=00 in helix :    29360
Number of ierr=10 in helix :     2604
```

```
Number of ierr=20 in helix :        0
Number of ierr=30 in helix :        0
Number of ierr=40 in helix :     2002
Number of ierr=50 in helix :      231
Number of ierr=60 in helix :        0
Number of ierr=70 in helix :        0
Changed charge in helix fit:      457
```

The following come from the vertex fit. The meanings of the error codes are described in the **TCVT** bank description. All events with a code less than 200 have been well fit.

```
Number of ierr=000 in vertex fit:     8343
Number of ierr=100 in vertex fit:      754
Number of ierr=300 in vertex fit:        0
Number of ierr=400 in vertex fit:       18
Number of ierr=500 in vertex fit:      609
Number of ierr=600 in vertex fit:        8
Number of ierr=700 in vertex fit:        0
Changed charge in vertx fit     :      239
```

The following statistics come from the RJPROC routine. For data coming from a DST, all of these should be zero as the RJPROC routine will never be called. For raw data tapes, the follwoing values are typical. The first number tends to be 600000 to 700000 per run (10000 events). It represents the total number of pulses processed from the **RJDF** data banks. The remaining numbers in this block should all be quite small; if any are large, then one should question the validity of the run.

```
Number of RJDF pulses      :   684514
Number of bad RJDF headers :        0
Number of count mismatchs  :        0
Number of RJDF overflows   :        0
RJPROC short t_inegrate    :       21
Number of bad RJDF Packets :        0
Number recovered           :        0
Too much RJDF data         :        0
```

References and further Information

- *CMZ*, **A Source Code Management System CMZ**,
- *Gunter Folger*, **Offline Reconstruction Software, CB–Note 121**.
- *F.–H. Heinsius and T. Kiel*, **Crystal Data Reconstruction Software, CB–Note 92**.
- *C. A. Meyer*, **Chamber Reconstruction Software, CB–Note 93**.
- *Mark Burchell*, **Global Tracking Particle Bank Structure, CB–Note 118**.
- *F.–H. Heinsius* **Calibration Constants Database Software, CB–Note 122**.