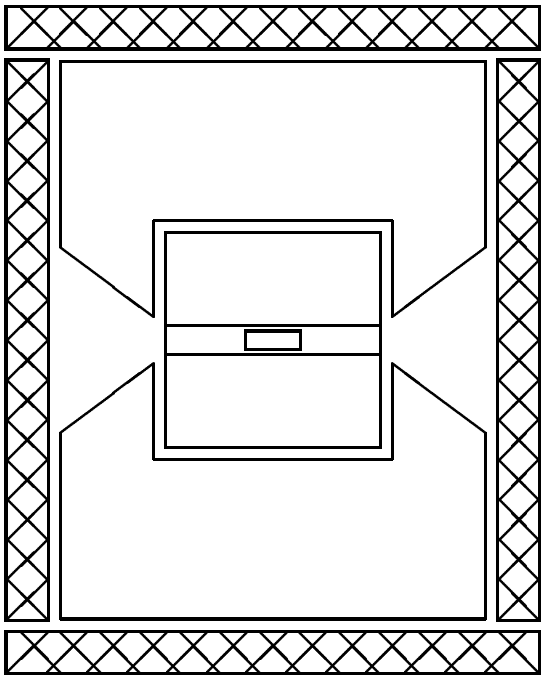
LEAR Crystal Barrel Experiment, PS197
Chamber Reconstruction Software
Locater Version 2.00

Curtis A. Meyer
Carnegie Mellon University

20 July, 1994



Contents

List of Tables

List of Figures

Updates to the Code

Version 1.30 to 1.44

- The /RJPRMS/ common block has changed to allow different time offsets in each FADC crate. The variables ITFFRJ and ITFGRJ have been replaced with ITFCRJ(16).
- Variables which are no longer used have been dropped from the /RJDATA/ common block.
- The cards **ITFF** and **ITFG** have been removed. They have been replaced by the **ITFC** data card. To use this card, it is necessary to have 17=1 as part of the input.
- The **RJDC** bank has been modified to include a self describing header. Also, the units in which time is stored have been changed from 1ns per channel to 200ps per channel.
- The RJTFIT routine has been removed. It is now included inline in the RJPULS subroutine.
- The TJZPOS and TJTIME routines have been merged into one subroutine, TJTIME. Since they were always called in pairs, this should cut down on overhead.
- The time fitting algorithem for raw pulses has been changed from a *bin of maximum difference* to a *first electron* method.
- The double pulse separation algorithem in TJDCGT has been modified to allow separation down to much smaller levels when the pulses are well separated in charge division.
- The *User service routine* TCRHIT has an additional call argument. It now returns the address of each hit in the **TCHT** bank in addition to the previous information.
- The addition of two new *User service routines*, TCHLDD and TCHLDS. These will return the three momentum and covariance matrix of a track in the **TCTR** data bank in either double or single precision.
- The addition of a calibration routine for performing a kinematic fit to the constraints of three–momentum balance, CJKFIT. The input is of the form generated by the TCHLDS subroutine.
- The value $\sigma_{dE/dx}$ in the **TJDC** bank has been replaced with a term σ_ϕ near the edge of the cell.
- The value $\sigma_{dE/dx}$ in the **TCHT** bank has been replaced with a term σ_ϕ near the edge of the cell.
- The subroutine TCVERS now will return the version number, and optionally print out the version information. This code is now section on *User Service Routines*.
- Inclusion of a new calibration data bank, **TJT0**. This contains time offsets for every wire in the JDC.
- The polynomials parameterizing the drift–time distance relation ship are now Hermite instead of Taylor polynomials. This allowed me to decrease the order by one and retain the same accuracy. It also makes the covraiance matrix more diagonal.
- The code has been put in the CMZ code manager for ease of updating. All PATCHY commands still work.
- Expanded the number of PATCHes in the code to facilitate updates.

- Modified version of the helix fit routines to remove a singularity at 90°.
- Included Laguerre Polynomials in the allowed types for the drift time to distance relationship. These are better than the Hermite.
- Included Slow Control monitoring code. This is both in the calibration phase, and the auto–updating of calibration constants based on changing conditions in the chamber.
- Included the subroutine BCLDD to load crystal data for the TCKFT4 and TCKFT3 routines.
- Modified the form of the **TCVP** data bank to accommodate a three by three covariance matrix for the vertex momenta.

version 1.44 to Version 1.45

Version 1.45 to Version 1.46

- The slow control information is now used to correct the r–ϕ look–up table.
- The **TCHT** data bank has been changed. It now has three additional words of information.

Version 1.46 to Version 1.50

- A new filtering scheme for outlyers has been implemented into the software. This causes about 20 percent of the data to be thrown away.
- New r–ϕ calibrations tables have been created based on input from Garfield.
- The handling of online data processing has been implemented, and steering cards RJDD and RJDF have been added to control this.
- Error logging is now handled by the ERRLOG subroutine. The slow control software will no longer produce hundreds of pages of useless output.
- PWC software has been implemented, and PWC hits are now included in the helix and vertex fits. Note that the inclusion of the PWC hits does not happen until after all circle fits are done. This means that they have no effect on the pattern recognition.
- The code has been modified to recognize and handle JDC data which has been processed online. There have also been cards introduced to allow the user to steer between the various types of raw data.
- The patches in the locater CMZ file have been reorganized. It is necessary to pick up a fresh card file at this release.

1 Generation of Code

1.1 Building the LOCATER code

The Locater code is distributed as a PAM file, LOCATER.PAM, and a default cradle file, LOCATER.CRA. In order to generate the most basic version of the code, one need only issue the command:

```
YPATCHY LOCATER.PAM LOCATER LOCATER.CRA .GO
```

However, other options have been installed in the PAM file which allow the user to:

- Install the chamber calibration software.
- Install debug print print lines, at many different levels.
- Select only *standard* Fortran–77.

In the most basic of installations, the user needs to have LOCATER.PAM, CBOFF.PAM, (the main code), and the following lines in LOCATER.CRA.

```
+USE,machine flag.
+USE,COMMCB,TCCOMMON.
+USE,TC_MAIN,TC_GAIN,TC_RAWS,TC_PATT,TC_CIRC.
+USE,TC_HELX,TC_VERT,TC_SERVC,TC_UTIL.
+EXE.
+PAM,12,T=ATTACH.            LOCATER.PAM
+PAM,11,R=COMMCB,T=ATTACH.   CBOFF.PAM
+QUIT.
```

Machine flags supported by LOCATER are as follows.

- ALT Alliant FX/8 for the Alliant fortran compiler.
- DECS For DecStations, using f77 fortran compiler.
- IBM For IBM-CMS and IBM-MVS computers.
- NXT For NeXT stations using the f77 fortran compiler.
- SUN For the SUN Work stations, f77 fortran compiler.
- UNIX Generic unix flag.
- VAX For VAX/VMS computers.

For installation of the calibration code in its most basic form, the cradle should be.

```
+USE,machine flag.
+USE,COMMCB,TCCOMMON.
+USE,TC_MAIN,TC_GAIN,TC_RAWS,TC_PATT,TC_CIRC.
+USE,TC_HELX,TC_VERT,TC_SERVC,TC_UTIL.
+USE,TC_CALIBRATE.
+EXE.
+PAM,12,T=ATTACH.            LOCATER.PAM
+PAM,11,R=COMMCB,T=ATTACH.   CBOFF.PAM
+QUIT.
```

To install debug lines, the following patches and flags are used.

- **TCDEBUG** is the patch containing all debug print statements. It also installs a minimum amount of event logging to let the user know which sections of the code were called for each event.
- **TCPRINT** installs statements which cause the results of each level of the code to be printed out. All of these statements are installed in the TCTRAK and CJCALB routines.
- **DEBUGRAW** installs debug print lines in the routines which process the data in the **RJDF** data bank.
- **DEBUGPAT** installs debug print lines in the pattern recognition section of the code.
- **DEBUGFIT** installs debug print lines in the circle fitting section of the code.
- **DEBHELIX** installs debug print lines in the helix fitting section of the code.
- **DEBVERTX** installs debug print lines in the vertex fitting section of the code.
- **DEBUGCAL** installs debug print lines in the calibration code.
- **DDDDEBUG** installs print lines that produce a lot of mostly useless output.
- **DEBFALSE** causes the default value of DEBGTC, the parameter which turns on and off printing to default to .FALSE..

As an example, the following would create code which included calibration, and debug print lines in the pattern recognition and circle fitting sections, and a general event monitor.

```
+USE,machine flag.
+USE,COMMCB,TCCOMMON.
+USE,TC_DEBUG.
+USE,TCPRINT,DEBUGPAT,DEBUGFIT.
+USE,TC_CALIBRATE.
+USE,TC_MAIN,TC_GAIN,TC_RAWS,TC_PATT,TC_CIRC.
+USE,TC_HELX,TC_VERT,TC_SERVC,TC_UTIL.
+EXE.
+PAM,12,T=ATTACH.          LOCATER.PAM
+PAM,11,R=COMMCB,T=ATTACH. CBOFF.PAM
+QUIT.
```

The user will find that several pilot cradles have been installed in the LOCATER code itself. The following is a list of what is available.

- ***LOCATE** Installs the basic code with no calibration and no debug print lines.
- ***DEBUG** installs the basic code with all debug options on.
- ***CALIB** installs the calibration code with no debug lines.
- ***DEBUGC** installs the calibration code with all debug options on.

Finally, because of both help in debugging the code, and readability of output, the LOCATER code is not written in absolutely standard Fortran–77. Such extensions as IMPLICIT NONE and print statements containing lower case characters have been used throughout the code. However, it is possible to remove these in the normal, (no calibration, no debug print lines) version of the code

by inclusion of the PATCHY flag, **+USE,F77.** However, unless the user's computer forbids these extensions, it is recommended that the above flag *not* be used.

Since version 1.42, LOCATER has only been supported 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
```

The same pilots used for PATCHY are also used for CMZ. To generate the standard program, (source code only), do the following:

```
cmz
CMZ[0] file locater -R
locater[1] set  locater.for -F
locater[2] pilot *LOCATE
locater[3] sel  machine_flag, (i.e. ALT,DECS,IBM,SUN or VAX)
locater[4] set  calibration_set (i.e. dec_89_1, jun_90_1, jul_90_1)
locater[5] seq  commcb
locater[6] ctof -P
locater[7] exit
```

1.2 The USER code

As discussed in the **Offline Reconstruction Software** Manual, the user is able to provide several specific routines to the entire software framework. These routines are usually provided in the framework of a PATCHY cradle file, USER.CRA. In order to have access to all the common blocks used routinely throughout the rest of the software, the following PATCHY commands can be used in the beginning of the uses's cradle.

```
+USE,machine flag.
+USE,COMMCB.
+USE,TCCOMMON.
+USE,P=CBPHYS,D=CBMAIN.
+USE,USERCODE.
+EXE.
+PAM,11,R=COMMCB,CBPHYS,T=ATTACH. CBOFF.PAM
+PAM,12,R=TCCOMMON,T=ATTACH.      LOCATER.PAM
+PATCH,USERCODE.
+DECK, ...


+QUIT.
```

This of course assumes that the user already has the PAM files for CBOFF and LOCATER. In the next sections are described several *standardizations* which are used throughout the offline software, and can/should also be used throughout the user's own code.

1.3 The value of π

The value of π, $\pi/2$ and 2π in both single and double precision forms are obtained with the PATCHY statement +CDE,PI2PI. This call will include the following piece of code.

```
*
      REAL       PI,PI2,PIHLF
      PARAMETER (PI    = 3.141592653589793)
      PARAMETER (PI2   = 6.283185397179866)
      PARAMETER (PIHLF = 1.570796326794896)
*
      DOUBLE PRECISION DPI,DPI2,DPIHLF
      PARAMETER (DPI    = 3.141592653589793D0)
      PARAMETER (DPI2   = 6.283185307179866D0)
      PARAMETER (DPIHLF = 1.570796326794896D0)
*
```

1.4 Access to Track Bank Sizes

Many of the data banks used to contain tracking information have been organized in a table format. For each entity in the bank, there are a fixed number of attributes. As an example, the **TJDC** data bank contains hits, and for each hit there are 16 attributes. So, if one wants to have access to the n'th hit in the bank at LTJDC, then one computes ITJDC as LTJDC $+16 \cdot (n-1)$. It is possible at some future date that it will become necessary to add more information on each of these hits. In order to simplify that transition, the length of each block in these data banks has been specified as a parameter which can be obtained using the PATCHY command +CDE,TRKPRM. This will then include the following information. It is recommended that the user use these where ever possible to avoid possible confusions in later versions of the program.

```
INTEGER    LENTJ,LENHT,LENTK,LENTR,LENHP,LENVX,LENVP,LENPW,LENCL
PARAMETER (LENTJ=16,LENHT=20,LENTK=30,LENTR=32,LENHP=6)
PARAMETER (LENVX=14,LENVP=13,LENPW=10,LENCL=12)
```

- LENTJ is the length of the repeating block in the **TJDC** bank.
- LENHT is the length of the repeating block in the **TCHT** bank.
- LENTK is the number of data words found in a **TCTK** bank.
- LENTR is the number of data words found in a **TCTR** bank.
- LENHP is the length of the repeating block in the **TCHX** bank.
- LENVX is the number of data words found in a **TCVT** bank.
- LENVP is the length of the repeating block in the **TCVP** bank.
- LENPW is the length of the repeating block in the **TPWC** bank.
- LENCL is the length of the repeating block in the cluster banks.

1.5 Input and Output

Throughout this code, a standardized set of *io* parameters has been used. These are obtained using the PATCHY command +CDE,CBUNIT.. This then includes a set of integers whose values are shown in table 1.

Unit	*Name*	*Description*
		In the /CBUNIT/ *common.*
4	LLOG	*Log file for the program*
4	LPRNT	*Alternate log file if needed*
6	LTERM	*Terminal for interactive jobs*
7	LDBG	*Debug print file.*
8	LERR	*Error file for the program*
9	LNORM	*Short term unit. This unit is used in setup routines for reading in files. All files using this unit should be one–pass routines which* OPEN *and* CLOSE *the file before returning .*
10	LHST	HBOOK *histogram file*
10	LHST	*This is also used by the* GRAFIKS *routines for printing of* meta *files.*
17		*Unit from which the data base file is read.*
20	LDST	*Unit for output of processed data*
21	LRDT	*Unit from which the data is read into the program*
30	LTIME	*Unit to read the fera look–up–table.*
31	LTIML	*Unit to read the 2282 look–up–table.*
70	LJTOUT	*write out large calibration files.*
72	LJGOUT	*write out the corrected gain file.*
81	LJCAL	*read in the* JDC *calibration card file.*
82	LJGAIN	*read in the* JDC *gain file.*
		In the /ZUNIT/ *common.*
	IQREAD	
	IQPRNT	
	IQPR2	
	IQLOG	
	IQPNCH	
	IQTTIN	
	IQTYPE	

Table 1: A list of all *io* units presently defined. The first two sections of the table are accessed through a +CDE,CBUNIT command, while the third section is obtained through a +CDE,ZUNIT command. The third section seems to be an internal ZEBRA common, and is used in the graphics sections of the code.

2 Data Banks for CB Chambers

2.1 Description of the Chamber Data Banks

In this section are described all data banks used in processing the chamber information from the Crystal Barrel. There are three types of banks so far: [1] those containing raw or unprocessed data, [2] those containing the results of pattern recognition and track reconstruction, and [3] scratch or temporary banks used through out the analysis. In figure 1 is shown the general structure of master and subbanks used throughout the chamber reconstruction. Because of the large amount of overhead required in setting up data banks[1], it is desirable to store many hits in the chamber in the same data bank. There are two natural divisions in the JDC, by sector, (30) and by layer (23). Both of these divisions have been utilized in storing chamber hit information. A top level, or master bank is set up, and then a series of subbanks for either each sector or each layer containing data. The link area in the master bank then contains pointers to all the subbanks which are addressed via their sector or layer number, and the data area of the master bank contains the number of hits stored in each subbank.

As a convention, all data banks which exist on the raw data tape have the letter R as the first letter in their names. For example, the data banks containing the results of the Qt analysis in the JDC are called **RJDC**. Banks that are either used in tracking, or contain tracking have T as the first letter in their names, while the second letter denotes the detector from which the data came, (J for the JDC, P for the PWC and C for the chambers in general). Beyond this, banks whose first letter is a C are used for calibration purposes, while banks whose first letter is K result from kinematic fits.

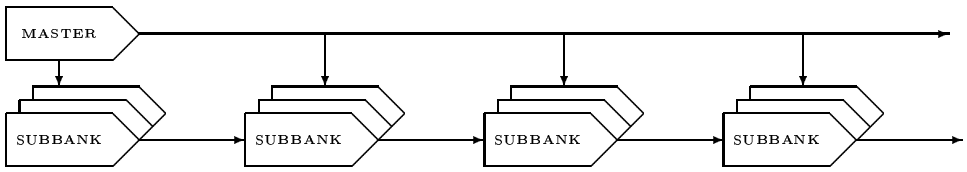


Figure 1: Layout of master and subbank structure used in chamber reconstruction. The master bank contains a down link to each array of subbanks in the link area, and the number of events stored in each subbank in the data area. The subbanks contain the actual event data, repeated for each event in the bank.

[1] A minimum of 10 data words per bank are required.

2.1.1 RJDC

The **RJDC** data bank contains the processed raw JDC data. This bank can either be produced by LOCATER from the **RJDF** bank, or be produced online by the FEP's. All data stored in this bank are integers, however the data is packed to try to optimize storage space. The bank format can also change from time to time, so a large header has been provided to identfy what the data format is. The data as shown in in table 2 is the standard format.

offset	TYPE	*Quantity*	
		bits 17–32	*bits 1–16*
+1	BYTE	*N I*2*	*Header*
+2	BYTE	*Length*	*Time Conv.*
+3	BYTE	t_0 *Subtracted*	*Dynamic Ped.*
+4		*unused*	
+5	BYTE	*Format*	*n I*2*
+6	BYTE	*sector/layer*	*Drift time*
+7	BYTE	*A Left*	*A Right*
⋮	⋮	*repeat +6 and +7 for all hits*	

Table 2: The data format in the **RJDC** data bank.

- *Header* is a header word set to **FFFD** hex.
- *N I*2* is the number of INTEGER*2 words in the data bank.
- *Length* is the length of the subunit in words which contain data. In the default bank this is 2, corresponding to words +6 and +7.
- *Time Conv.* is the conversion between time units in the **RJDC** bank, and nanoseconds. The normal storage time is 200ps per count, in which case the conversion constant is 5.
- *Dynamic Ped*estal indicates if the pedestals have been taken as given in the **RJDF** bank, or computed dynamically. A value of zero means they are taken from the **RJDF** bank, while a value of one means they have been dynamically computed.
- t_0 *subtracted* indicates if the time offset from the FADC crate has been subtracted from the data. If the value is zero, then this has been subtracted, if it is one, then it has not been subtracted.
- *Format* is a format word set to **FFFD** hex.
- *n I*2* is the number of number of remaining INTEGER*2 words in the bank, (excluding itself, but including the *Format* word.
- *Sector/Layer* contains the sector number (high order byte) and layer number, (low order byte) of the hit.
- *Drift Time* is the drift time in units of 200ps. Note that it is possible this may change, so to convert to nanoseconds, one should always use the provided *Time Con*version word.
- *A Left* is the amplitude from the *left* or $+z$ preamplifier.
- *A Right* is the amplitude from the *right* or $-z$ preamplifier.

2.1.2 RJDF

This bank contains the unprocessed pulse information from the JDC. This is the full pulse information for the chamber and tends to be very large in volume. This bank will only be available on the raw data tapes, at all later levels, this bank will have been replaced with the **RJDF** data bank. The bank contents are shown in table 3.

offset	TYPE	*Quantity*	
		bits 17–32	*bits 1–16*
+1	BYTE	*Format*	*N(I*4)*
+2	BYTE	*n(I*2)*	*Sector/layer*
+3	BYTE	*Not used*	*Time Offset*
+4	BYTE	*Ped. Right*	*Ped. Left*
+5	BYTE	$Chan_1$ *L/R*	$Chan_2$ *L/R*
⋮	⋮	⋮	⋮
$+4+(n(I*2)+1)/2$	BYTE	*n(I*2)*	*Sector/layer*
⋮	⋮	⋮	⋮

Table 3: The data format in the **RJDF** data bank.

- *Format* is a format word for the bank, **FFFB** hex.
- *N(I*4)* is the number of INTEGER*4 words in this bank.
- *n(I*2)* is the number of INTEGER*2 words in this hit. Note, if this is odd, then the INTEGER*4 word containing the last data word is padded out so that the next hit information has exactly the same structure.
- *Sector/Layer* contains the sector number, (high order byte) and the layer number, (low order byte) of this hit.
- *Time Offset* is the time offset in **FADC** channels, (10 ns) to the start of the recorded information.
- *Ped. Right* is the *right* or $-z$ pedestal.
- *Ped. Left* is the *left* or $+z$ pedestal.
- $Chan_i$ *L/R* is the packed pulse data from the flash ADC's. The high order eight bits contain the non–linear data from the left end, and the low order eight bits contain that from the right end.

2.1.3 RPWC

The data stored in the **RPWC** bank contains a list of all wires which fired in both PWC's. The bank is filled in the TPWCGT routine, and then used in the TPWPOS routine for obtaining physical positions. The data format is shown in table 4. The wire numbers run continuously from 0 to 319. Wires 0–119 are in the inner chamber, while wires 129–308 are wires 128–309 in the outer chamber. Wires 120–127 and 310–319 are not physically connected to anything. The wire data is stored as two wires per integer word. Bits 17 to 32 contain the first wire, while bits 1 to 16 contain the second wire.

Offset	TYPE	*Quantity*	
		bits 17–32	*bits 1–16*
-1	INTEGER	*Number of data words*	
+1	INTEGER	*Number of wires +2*	
+2	BYTE		*wire 1*
+3	BYTE	*wire 2*	*wire 3*
⋮	⋮	⋮	⋮

Table 4: The data stored in the **RPWC** data bank.

2.1.4 TJDC

The **TJDC** bank is a bank structure whose top level bank is the **TJDC** bank. Under this bank, there are up to 30 **TJDS** subbanks, (one for each JDC sector containing data). The information stored here is the result of the Qt analysis on the JDC FADC information. As well as semi–processed hit information. The chamber information is stored in the **TJDS** subbanks. The structure of the **TJDC** bank contains 30 pointers and 30 data words. The pointers point to the subbanks for each sector in the JDC, while the data words contain the number of fired wires in each sector. The structure of the **TJDS** subbanks contains 4 data words for each fired wire in the sector. The **TJDC** bank is filled in TJDCGT routine, and then used throughout the pattern recognition section of the code, (TCSGMT, TCRAW1 and TCRAW2). After the level of pattern recognition, this bank is no longer needed. Table 5 shows the structure of the **TJDC** data bank, where each of the contained quantities is described in detail as follows.

Offset	TYPE	*Quantity*
+1	INTEGER	*Layer number*
+2	INTEGER	*Resolution code*
+3	INTEGER	*Segment number*
+4	INTEGER	*Hit number in* **TCHT**
+5	REAL	t_D
+6	REAL	A_+
+7	REAL	A_-
+8	REAL	x_l [cm] *in Sector coordinates*
+9	REAL	y_l [cm] *in sector coordinates*
+10	REAL	x_r [cm] *in sector coordinates*
+11	REAL	y_r [cm] *in sector coordinates*
+12	REAL	z [cm] *in CB coordinates*
+13	REAL	σ_z [cm]
+14	REAL	dE/dx [MeV]
+15	REAL	σ_ϕ (Edge)[radians]
+16	REAL	$\tan\lambda$ *in CB coordinates*
⋮	⋮	*repeated for every hit in this sector*

Table 5: The data stored in the subbanks of the **TJDC** bank, (**TJDS**). This structure is repeated for each hit in the bank.

- *Layer* is the layer in the JDC in which this hit was found.
- *Resolution* is a code indicating if the left–right ambiguity has been resolved for this point. A value of 0 indicates it has not been resolved, a value of 1 indicates left and a value of 2

indicates right.

- *Seg. No.* A number identifying with which segment this hit has been identified in the $\tan\lambda$ association, (see TCSGMT routine).
- *Hit no.* is the number of the hit in the TCHT data bank where this point has been stored.
- t_D Drift time for this hit, [μs].
- A_+ Amplitude from $+z$ or the left end of the JDC.
- A_- Amplitude from $-z$ or the right end of the JDC.
- x_l and y_l are the x and y coordinates of the hit under the assumption that it came from the left side of the cell. These points are in what I define as a Sector–coordinate System. In this system, the sector is assumed to have the x–axis running down it's middle, an the y–axis points up, (see TJTIME routine).
- x_r and y_r are the same as above, except assuming that the hit came from the right side of the cell.
- z and σ_z are the z–coordinate and the error in z–coordinate for the hit.
- dE/dx is the energy loss at this point.
- σ_ϕ is the error in ϕ due to edge effects. This error term is zero unless the hit is within YCUTTJ of the cell edge, (see the /TJPRMS/ common block.
- $\tan\lambda$ is the tangent of the opening angle of this hit under the assumption that the interaction was at the center of the target. We have that $\tan\lambda = \frac{z}{r}$ where z is obtained from charge division, and r is the radius of the hit wire.

2.1.5 TCHT

The **TCHT** bank structure contains reconstructed hit information on each hit in the JDC. The **TCHT** data bank is a top level data bank that contains one subbank for each layer containing hits in the chamber, **TCLY**. The **TCHT** bank has 23 pointers, and 23 data words. Each pointer points to one of the **TCLY** banks, and the data word contains the number of hit wires in that layer. The stored information on each hit is given in table 6. The **TCHT** banks are lifted and filled in TCRAW1 and TCRAW2. At this point, a warning should be made about the addressing of this bank. Because many hits are stored in the same physical data bank, the convention of the first data word being at one added to the bank pointer has not been strictly followed. In some of the code, the computed pointer actually points to the first data word, while in other routines it points to the word immediately before the first data word. Eventually this inconsistency will be removed, but until then, most of the patter recognition uses the latter scheme, while the fitting sections tend to use the former scheme.

- *Track number* identifies if the point has been incorporated into a segment, and if so what the number of the track in **TCTK** is. A 0 indicates no connection has been made.
- *Resolution code* identifies if the point has been resolved or not. It is defined the same as in the **TJDC** bank.
- *Forward pointer for layer* is the layer number of the next hit in this segment.

Offset	TYPE	*Quantity*
+1	INTEGER	*Track number*
+2	INTEGER	*Resolution code*
+3	INTEGER	*Forward pointer for layer*
+4	INTEGER	*Forward pointer for hit*
+5	INTEGER	*Backward pointer for layer*
+6	INTEGER	*Backward pointer for hit*
+7	INTEGER	*Sector number*
+8	INTEGER	*Hit number in* **TJDC**
+9	REAL	x_l [cm] *in CB coordinates*
+10	REAL	y_l [cm] *in CB coordinates*
+11	REAL	x_r [cm] *in CB coordinates*
+12	REAL	y_r [cm] *in CB coordinates*
+13	REAL	z [cm] *in CB coordinates*
+14	REAL	σ_z [cm]
+15	REAL	dE/dx [MeV]
+16	REAL	σ_ϕ *(Edge)* [rad]
+17	REAL	r [cm] *in CB coordinates*
+18	REAL	ϕ [rad] *in CB coordinates*
+19	REAL	σ_r [cm]
+20	REAL	σ_ϕ [rad]
+21	REAL	*Drift Time* μs
+22	REAL	*Left Amplitude*
+23	REAL	*Right Amplitude*
⋮	⋮	*repeated for every hit in this layer*

Table 6: The data stored in the subbanks of the **TCHT** bank, (**TCLY**). The subbanks are arranged by layer, and the structure is repeated for each hit in the layer.

- *Forward pointer for hit* is the hit number of the next point in this segment in the **TCHT** bank.
- *Backward pointer for layer* is the layer number of the previous hit in this segment.
- *Backward pointer for hit* is the hit number of the previous hit in this segment in the **TCHT** bank.
- *Sector* is the sector in which this hit was found.
- *Hit No.* is the hit number in the sector. These last two entries are used to refer back to the **TJDC** bank.
- x_l and y_l are the x and y coordinates of the hit assuming that it was on the left side of the cell. These coordinates are in the CB system, (z along the direction of the incoming antiproton, y pointing up, and x such that the system is right handed).
- x_r and y_r are the x and y coordinates of the hit assuming that it was on the right side of the cell.
- z and σ_z are the z–coordinate and its error for the hit.
- dE/dx is the energy loss for the hit.
- σ_ϕ (Edge) is the error in ϕ due to effects at the edge of the drift cell.
- r and ϕ are said coordinates for the hit. If the hit has not been resolved, then they are not set.
- σ_r and σ_ϕ are the 1σ errors in r and ϕ, (see the TCRESL routine for calculations of these quantities).
- *Drift Time*
- *Left Amplitude*
- *Right Amplitude*

2.1.6 TCTK

The **TCTK** bank structure contains the results of the first level reconstruction for each track found in the JDC. The **TCTK** is a master bank over the **TCSG** data banks. **TCTK** contains 50 pointers and one data word. The data word is the number of tracks found, and the pointers then point to the **TCSG** bank for each found track. The data stored in each **TCSG** bank is shown in table 7. The **TCTK** banks are lifted in the TCRAW1 and TCRAW2 routines, but filled mostly in the *circle fitting* section of the code, (TCLOAD). Additional information is coded into the header word of each sub–bank. If bit 1 is set to 1, then isochron corrections have already been performed. The TCIFIX routine checks this to make sure it does not apply isochron corrections twice. If bit 2 is set to 1, then the track crosses sector boundaries in the JDC. The circle fit parametrization is given by the formula:

$$\kappa \cdot r_i + c^2/r_i + \sin(\phi_i - \psi_0) = 0$$

In this form, if the center of the circle is at the point (a, b), and it has a radius of ρ, then:

$$\begin{aligned} \kappa &= \frac{1}{2 \cdot \sqrt{a^2 + b^2}} \\ c^2 &= (\frac{1}{2 \cdot \kappa})^2 - \rho^2 \end{aligned}$$

Offset	TYPE	*Quantity*
+1	INTEGER	*Number of Hits*
+2	INTEGER	*Layer of Hit 1*
+3	INTEGER	*Hit number of hit 1*
+4	INTEGER	*Layer of Hit n*
+5	INTEGER	*Hit number of hit n*
+6	INTEGER	*Number of dE/dx hits*
+7	INTEGER	*Error code from fit*
+8	REAL	*Charge*
+9	REAL	*Mass* $[MeV]$
+10	REAL	dE/dx $[MeV/cm]$
+11	REAL	$\sigma_{dE/dx}$ $[MeV/cm]$
+12	REAL	$p_\perp$ $[MeV/c]$
+13	REAL	ψ_0 [radians]
+14	REAL	p_L $[MeV/c]$
+15	REAL	$\delta p_\perp$ $[MeV/c]$
+16	REAL	$\delta\psi_0$ [radian]
+17	REAL	$s \cdot \kappa$ $[cm^{-1}]$
+18	REAL	c^2 $[cm^2]$
+19	REAL	$\tan\lambda_0$
+20	REAL	a $[cm]$
+21	REAL	$C_{r\phi}$
⋮	⋮	⋮
+27	REAL	$C_\lambda(1,1)$
+28	REAL	$C_\lambda(2,1)$
+29	REAL	$C_\lambda(2,2)$
+30	REAL	χ^2

Table 7: The data stored in the subbanks of the **TCTK** bank, (**TCSG**). There is one **TCSG** data bank for each track found in the chambers.

- *Number of Hits* is the number of hits incorporated into this track.
- *Layer of hit 1* is the layer number of the first hit in this track.
- *Hit number of hit 1* is the number in **TCHT** of hit 1.
- *Layer of hit n* is the layer number of the last hit in this track.
- *Hit number of hit n* is the number in **TCHT** of hit n.
- *Number of dE/dx hits* is the number of hits used for dE/dx calculations.
- *Error code* is initially set in the TCRAWS routine to either 0 or 1. A value of 1 indicates that the track crosses sector boundaries, while a value of 2 indicates that the track is contained in one JDC sector. The routines TCFITR and TCTHET later change the value of the error code to have the following meanings:

  0 The fit converged normally, and exited because the χ^2 became small enough.

  1 The initial guess was good enough, and no iteration was performed.

  2 The fit converged to a value which was too large, but the TCSPLT routine could find nothing wrong with the track.

  3 The fit converged to a χ^2 larger than the value of the cutoff.

  4 The value of χ^2 began to diverge after the third iteration in the fit.

  5 The fit failed to converge in NITRTC iterations.

  6 There were not enough points to fit a track, fewer than three.

  7 The fit did not converge because the program tried to invert a singular matrix. Do not trust the results, particularly the covariance matrix.

  8 The TCTHET routine reconstructed a λ of 90°.

  9 The TCTHET routine got lost in stepping through the track.

- *Charge* is the electric charge of this track, ± 1.
- *Mass* is the mass in MeV/c of this particle.
- dE/dx is the average dE/dx of this particle.
- $\sigma_{dE/dx}$ is the error in the dE/dx measurement.
- $p_\perp$ is the momentum perpendicular to the beam direction, [MeVc].
- ψ_0 is the direction of the particle in the x–y plane at the point of closest approach to the origin.
- p_L is the longitudinal component of the particle's momentum, [MeV/c].
- $\delta p_\perp$ is the error in the fit momentum.
- $\delta\psi_0$ is the error in the fit ϕ.
- κ comes from the $r-\phi$ circle fit. It is $1/2R$ where R is the distance from the center of the fit circle from the origin, (If the circle center is at (a,b), then $R=\sqrt{a^2+b^2}$).

- c^2 is a parameter from the circle fit which is related to the distance of closest approach to the origin. The radius of curvature of the particle is given as

$$\rho = \sqrt{R^2 - c^2}$$

  and the distance of closest approach is then $\mid \rho - R \mid$.

- $\tan\lambda$ is the tangent of the opening angle obtained from the r–z fit.

- $C_{r\phi}$ is the 3 by 3 covariance matrix obtained from the fit to a circle in $r - \phi$ The matrix is symmetric, and stored as follows.

$$\begin{pmatrix} +21 & * & * \\ +22 & +24 & * \\ +23 & +25 & +26 \end{pmatrix} = \begin{pmatrix} \sigma_{RKRK} & \sigma_{RK\psi} & \sigma_{RKc^2} \\ \sigma_{RK\psi} & \sigma_{\psi\psi} & \sigma_{\psi c^2} \\ \sigma_{Rkc^2} & \sigma_{\psi c^2} & \sigma_{c^2c^2} \end{pmatrix}$$

- $C_\lambda(1,1) = \sigma_{\tan\lambda\tan\lambda}$.

- $C_\lambda(2,1) = \sigma_{\tan\lambda a}$.

- $C_\lambda(2,2) = \sigma_{aa}$.

- χ^2 is the χ^2 of the circle fit performed in the TCFITR routine. It has units of cm^2.

2.1.7 TCTR

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 8. The **TCTR** banks are lifted and filled in the helix fitting section of the code. These banks are generically referred to as the **TCTR** data banks. The header word, IQ(ITCTR) also contains some coded information about the tracks. If bit 2 is set to 1, then the track crossed at least one sector boundary, whereas if it is set to 0, then the track is contained in one sector.

- *Nhits* is the number of hits incorporated into this track, (JDC).

- *NPED* is the number of the PED to which this track connects. This word is filled during global tracking, and is not available until that has been performed.

- *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.

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	*N*PWC
+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	$C_\xi[1,1]$
⋮	⋮	⋮
+32	REAL	$C_\xi[5,5]$

Table 8: The data stored in the subbanks of the **TCTR** bank, (**TCTX**). There is one **TCTX** data bank for each track found in the chambers.

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. They are stored as the first entries of the **TCHX** bank for the 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.

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

2.1.8 TCHX

The **TCHX** data bank contains the coordinates of the hits used in the helix fits. 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. Because of this, one should not perform the helix fit twice. After the 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** 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 9: 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.

2.1.9 TCVX

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 2.

2.1.10 TCVT

The **TCVT** data banks are subbanks to the **TCVX** data bank, and contain information regarding the found vertices. The data are shown in table 10.

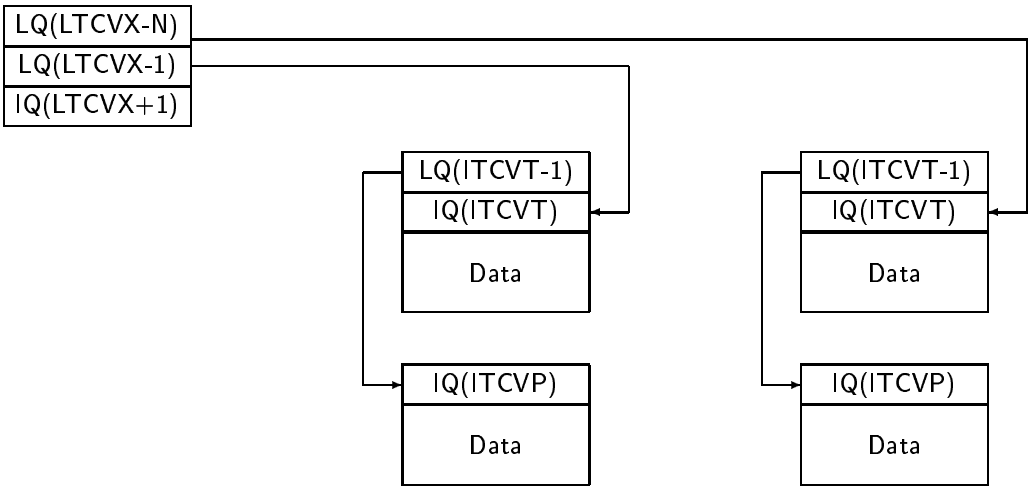


Figure 2: Layout of the bank structure containing all found vertices.

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

- *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.
- *Nneutral* is the number of neutral tracks assigned to this vertex. At present, there is no list available as to which tracks 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:

$$\begin{pmatrix} +6 & * & * \\ +7 & +8 & * \\ +9 & +10 & +11 \end{pmatrix} = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix}$$

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

2.1.11 TCVP

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 describing 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.

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

- E is the energy under the assumption the particle is a pion.
- The remaining six terms are the unique elements of the three by three covariance matrix for the momentum.

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

2.2 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 12.

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 12: 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 chamber 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 13. 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.

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

- 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 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 chamber if desired.
*
      ENDIF
```

2.3 Calibration Data Banks

In this section are described the data banks which contain JDC calibration constants. These are the constants used for adjusting the gains of all preamplifiers, and for defining the drift region in the

JDC. They are used in the following formula, where P_i are polynomials of order i.

$$z^i = z_0^i + z_l^i \cdot \left[\frac{A_+ - \alpha^i \cdot A_-}{A_+ + \alpha^i \cdot A_-}\right] \quad (1)$$

$$\frac{dE}{dx} = c_E^i \cdot \left[A_+ + \alpha^i \cdot A_-\right] \quad (2)$$

$$y_l^i = s^i + a_0^i(1 - e^{-a_5^i\sqrt{t_D}}) + a_1^i \cdot P_0(t_D) + a_2^i \cdot P_1(t_D) + a_3 \cdot P_2(t_D) + a_4^i \cdot P_3(t_D) \quad (3)$$

$$x_l^i = r^i + b_0^i \cdot P_0(y_l^i - s^i) + b_1 \cdot P_1(y_l^i - s^i) + b_2 \cdot P_2(y_l^i - s^i) \quad (4)$$

$$y_r^i = s^i + a_6^i(1 - e^{-a_{11}^i\sqrt{t_D}}) + a_7^i \cdot P_0(t_D) + a_8^i \cdot P_1(t_D) + a_9 \cdot P_2(t_D) + a_{10}^i \cdot P_3(t_D) \quad (5)$$

$$x_r^i = r^i + b_3^i \cdot P_0(y_r^i - s^i) + b_4 \cdot P_1(y_r^i - s^i) + b_5 \cdot P_2(y_r^i - s^i) \quad (6)$$

At present, the allowed polynomials are Taylor, Hermite and Laguerre. These are selected by the flag word in the **TJCP** data bank. The forms of these polynomials are given by:

$T_0(t) = t$	$H_0(t) = 1$	$L_0(t) = 1$
$T_1(t) = t \cdot T_0(t)$	$H_1(t) = t$	$L_1(t) = t - 1$
$T_2(t) = t \cdot T_1(t)$	$H_2(t) = 2 \cdot (t \cdot H_1(t) - H_0(t))$	$L_2(t) = \frac{1}{2}((t-3)L_1(t) - 1)$
$T_3(t) = t \cdot T_2(t)$	$H_3(t) = 2 \cdot (t \cdot H_2(t) - 2 \cdot H_1(t))$	$L_3(t) = \frac{1}{3}((t-5)L_2(t) - 2L_1(t))$

Where T_i are Taylor polynomials, H_i are Hermite polynomials and L_i are Laguerre polynomials. Note that in the case of Taylor polynomials, no constant term is used. Also, the scale factor is absorbed into the fit constant for these terms. The various constants in the above equations are stored in the following data banks.

2.3.1 TJCE

The **TJCE** bank contains the 690 c_E^i constants used to convert the modified amplitude sums to an energy. There is one constant for every wire in the jdc, and the are addressed in the bank as:

$$c_i^E = (s-1) \cdot 23 + l$$

where s is the sector number, (1–30), and l is the layer number, (1–23).

2.3.2 TJCP

The **TJCP** data bank contains a set of parameters for generating the look–up table in the **TJCT** data bank. These are the a_k^i and b_k^i values in the last four equations. The data bank contains 20 words per layer, with words 1 through 10 containing a_0 to a_5, b_0 to b_2 and a radius parameter. Words 11 to 20 then contain a_6 to a_{11}, b_3 to b_5 and a second radius parameter. The word IQ(LTJCP) contains information about what these parameters mean. Bits 1,2 and 3 are set according to the following code scheme:

000 Use Taylor polynomials for the expansion.

001 Use Hermite polynomials for the expansion.

010 Use Laguerre polynomials for the expansion.

011 Use Garfield tables.

2.3.3 TJCZ

The **TJCZ** bank contains the 690 α^i constants used to compute the z position on each wire. They are a measure of the relative gain between the two ends of the chamber, and are addressed in the same way as the constants in the **TJCE** bank.

2.3.4 TJCT

The **TJCT** bank contains a look–up table used to convert drift time to position in the JDC. The positions are stored in sector coordinates, (see the description of the TJTIME routine), and are computed every 25ns of drift time. For each drift time, there are two possible positions in the chamber, so four data words are stored for each word. This bank is generated from the data in the **TJCP** data bank in the TJTIMI routine.

The **TJCT** data is actually a top level bank which contains 23 sub–banks, one for each layer in the JDC. The data in the TJCT bank is shown in table 14. Once the pointer, L and the number

LQ(−23)	INTEGER	*Pointer for layer 23*
⋮	⋮	⋮
LQ(−1)	INTEGER	*Pointer for layer 1*
IQ(+1)	INTEGER	*Number of entries for layer 1*
⋮	⋮	⋮
IQ(+23)	INTEGER	*Number of entries for layer 23*

Table 14: The pointers and data words in the **TJCT** data bank.

of entries, N_e has been obtained from the **TJCT** bank, the data in the table can be accessed. The storage locations of the data for each of the four quantities is shown in table 15.

Quantity	*From*	*To*
x_l	$L+1$	$L+N_e$
y_l	$L+N_e+1$	$L+2N_e$
x_r	$L+2N_e+1$	$L+3N_e$
y_r	$L+3N_e+1$	$L+4N_e$

Table 15: Storage of data in the Look–up table.

2.3.5 TJRF

This data bank contains reference conditions in the JDC. The bank is created at the same time the $r\phi$ calibration is performed. The bank is filled with the averages and standard deviations of various slow control data. The bank has 200 entries, of which the following entries are filled.

Offset TYPE *Quantity*
+001 INTEGER *Calibration Date.*
+002 INTEGER *Calibration Time.*
+003 REAL *JDC Set Voltage Chan. 1.*
⋮
+034 REAL *JDC Set Voltage Chan. 32.*
+035 REAL *Standard Deviation of Voltage 1.*
⋮
+066 REAL *Standard Deviation of Voltage 32.*
+067 REAL *JDC True Voltage Chan. 1.*
⋮
+098 REAL *JDC True Voltage Chan. 32.*
+099 REAL *Standard Deviation of Voltage 1.*

⋮
+130 REAL *Standard Deviation of Voltage 32.*
+132 REAL *JDC Isobutane Flow.*
+133 REAL *JDC CO2 Flow.*
+134 REAL *JDC Mixed Flow.*
+137 REAL *Standard Deviation of JDC Isobutane Flow.*
+138 REAL *Standard Deviation of JDC CO2 Flow.*
+139 REAL *Standard Deviation of JDC Mixed Flow.*
+140 REAL *HDC Potential Voltage.*
+141 REAL *HDC Grid Voltage.*
+142 REAL *HDC Drift Voltage.*
+145 REAL *Standard deviation of Potential Voltage.*
+146 REAL *Standard deviation of Grid Voltage.*
+147 REAL *Standard deviation of Drift Voltage.*
+150 REAL *HDC Drift Time 1.*
+151 REAL *HDC Drift Time 1 Sigma.*
+152 REAL *HDC Drift Time 2.*
+153 REAL *HDC Drift Time 2 Sigma.*
+154 REAL *HDC Drift Time Difference.*
+155 REAL *HDC Drift Time Difference Sigma.*
+160 REAL *Standard Deviation of Time 1.*
+161 REAL *Standard Deviation Time 1 Sigma.*
+162 REAL *Standard Deviation Time 2.*
+163 REAL *Standard Deviation Time 2 Sigma.*
+164 REAL *Standard Deviation Time Difference.*
+165 REAL *Standard Deviation Time Difference Sigma.*
+170 REAL *JDC Absolute Pressure.*
+171 REAL *JDC Differential Pressure.*
+172 REAL *Gas mixture as fraction Isobutane.*
+175 REAL *Standard Deviation JDC Pressure.*
+176 REAL *Standard Deviation JDC Diff. Pres.*
+177 REAL *Standard Deviation of mixture.*
+180 REAL *JDC Temperature 1 in Kelvins.*
+181 REAL *JDC Temperature 2 in Kelvins.*
+182 REAL *JDC Temperature 3 in Kelvins.*
+183 REAL *JDC Temperature 4 in Kelvins.*
+184 REAL *JDC Temperature in Kelvins.*
+185 REAL *Sigma in Temperature 1.*
+186 REAL *Sigma in Temperature 2.*
+187 REAL *Sigma in Temperature 3.*
+188 REAL *Sigma in Temperature 4.*
+189 REAL *Sigma in Temperature.*

2.3.6 TJST

This data bank contains the stagger in centimeters of every wire in the JDC, s^i. It allows for the possibility that the position of some wires are not within tolerance. The data is addressed in the same manner as the **TJCE** data bank.

2.3.7 TJT0

This bank contains the time offset to be used for every wire in the JDC. At present, the data is read in from unit 81 with the rest of the gain information. If the information is not found in the file, then the program takes the crate–wise values in the ITFCRJ array found in the /RJPRMS/ common block. These values can be forced by using the **ITFC** control card. If 17=1 is used on this card, then the values from the common block are always used.

2.3.8 TJWR

This bank contains a coded list of wires which are found to have poor z–resolution. The bank contains 23 unsigned integer words, one for each layer in the JDC. In each word, a bit corresponds to a sector in the JDC, (i.e. bit one to sector 1 and bit 30 to sector 30). If a particular bit is set to one, then that wire should not be used. At the start of analysis, this bank is unpacked into the /TJWIRE/ common block, which is then used by the TJDCGT subroutine. During normal analysis, the z–coordinate for these wires is assigned to be zero, and the error in z is set to 20 cm.

2.3.9 TJZL

This bank contains the length of each wire in the JDC, z_l^i. These are not necessarily the same for every wire as the position of the electrical connection between the wire and crimp pins can vary by several millimeters at both ends of the chamber. This data is addressed in the same manner as in the previous bank.

2.3.10 TJZ0

This bank contains the z–position at the center of each sense wire, $(z_l/2)$. This can also be different from zero for the same reasons as with the previous data bank. As with the last bank, the addressing is as in the **TJCE** data bank.

2.4 Monte Carlo data banks

2.4.1 RMCB

The **RMCB** data bank contains event information from GEANT. This bank contains the GEANT **JVERTEX** bank and the **JKINE** banks, which have been shunted to LQ(L-1) and LQ(L-2) resectively. See the **GEANT 3** manual for a better description.

3 User Callable Routines

3.1 User Service Routines

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 a hold 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. All of these routines are found in the TC_SERVC patch in the LOCATER card file.

3.1.1 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, consult 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.

3.1.2 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.

3.1.3 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.

3.1.4 SUBROUTINE RJRJDC

Author: Curtis A. Meyer
Creation Date: 15 January, 1989
References:
Call Arguments: (LUN).
Common Blocks Used: CBBANK, CBLINK.
Subroutines Referenced: None.

Print the contents of the **RJDC** data bank on unit LUN.

3.1.5 SUBROUTINE RJRJDF

Author: Curtis A. Meyer
Creation Date: 20 May, 1989
References:
Call Arguments: (LUN).
Common Blocks Used: CBBANK, CBLINK.
Subroutines Referenced: None.

Print the contents of the **RJDF** data bank on unit LUN.

3.1.6 SUBROUTINE RPRPWC

Author: Curtis A. Meyer
Creation Date: 15 January, 1989
References:
Call Arguments: (LUN).
Common Blocks Used: CBBANK, CBLINK.
Subroutines Referenced: None.

Print the contents of the **RPWC** data bank on unit LUN.

3.1.7 SUBROUTINE RJTJDC

Author: Curtis A. Meyer
Creation Date: 25 May, 1989
References:
Call Arguments: (LUN).
Common Blocks Used: CBBANK, CBLINK.
Subroutines Referenced: None.

This routine will print only the information found in the **RJDC** data bank, but the information will be taken from the **TJDC** data bank, rather than the **RJDC** bank. This has the advantage that the information will be sorted by sector and layer number. All information will be printed to logical unit LUN.

3.1.8 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.

3.1.9 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.

3.1.10 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 precision. For single precision values, see the TCHLDS subroutine.

3.1.11 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.)

3.1.12 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 following 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. Note that because this problem is linear, only one iteration is performed.

3.1.13 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: DINV.

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, (dimensions 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, 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 three equations of constraint:

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

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.

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)
*
```

3.1.14 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.

3.1.15 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.

3.1.16 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.

3.1.17 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.

3.1.18 SUBROUTINE TCTCHT

Author: Curtis A. Meyer
Creation Date: 15 January, 1989
References:
Call Arguments: (LUN).
Common Blocks Used: CBBANK, CBLINK.
Subroutines Referenced: None.

Print the contents of the **TCHT** data bank on unit LUN.

3.1.19 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)
*
```

3.1.20 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 returned values of PMOM and COVR are double precison. The user can specify the minimum track length as LONGT. For single precision values, see the TCVLDS subroutine.

3.1.21 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.

3.1.22 SUBROUTINE TCVLDD

Author: Curtis A. Meyer
Creation Date: 21 April, 1990
References:
Call Arguments: (IVRT, *NTRK, *ICODE, *CHRG, *PMOM, *COVR, *IERR, LONGT).
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. The user can specify the minimum track length as LONGT. For single precision values, see the TCVLDS subroutine.

3.1.23 SUBROUTINE TCVLDS

Author: Curtis A. Meyer
Creation Date: 21 April, 1990
References:
Call Arguments: (IVRT, *ICODE, *NTRK, *CHRG, *PMON, *COVR, *IERR, LONGT).
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.

3.1.24 SUBROUTINE TJTJDC

Author: Curtis A. Meyer
Creation Date: 15 January, 1989
References:

Call Arguments: (LUN).
Common Blocks Used: CBBANK, CBLINK.
Subroutines Referenced: None.

Print the contents of the **TJDC** data bank to unit LUN.

3.1.25 SUBROUTINE TPTPWC

Author: Curtis A. Meyer
Creation Date: 15 January, 1989
References:
Call Arguments: (LUN).
Common Blocks Used: CBBANK, CBLINK.
Subroutines Referenced: None.

Print the contents of the **TPWC** data bank on unit LUN.

3.2 Utility Routines

All of these routines are found in the TC_UTIL patch in the LOCATER card file. Most of these routines are called directly by LOCATER. However, they could also be useful to users of the code.

3.2.1 FUNCTION KRATE

Author: Curtis A. Meyer
Creation Date: 12 April, 1990.
References:
Call Arguments: (ISEC, ILYR).
Common Blocks Used: None.
Called by: RJPROC.
Subroutines Referenced: None.

Given a wire described by sector ISEC and layer ILYR, this function will compute the FADC crate number in which the wire's data is processed. The allowed numbers are 1 to 16. A value of 0 is returned for nonsensical layer or sector numbers.

3.2.2 SUBROUTINE SM353

Author:
Creation Date:
References:
Call Arguments: (Y, Z, N).
Common Blocks Used: None.
Called by:
Subroutines Referenced: None.

This routine will smooth data using the 3G53CQH method. The data are passed as an array Y containing N points. The smoothed data is returned in the array Z. Y and Z can be the same array.

3.2.3 SUBROUTINE SORTIL

Author:
Creation Date:
References:
Call Arguments: (A, IDX, N).
Common Blocks Used: None.
Called by: TJDCGT.
Subroutines Referenced: None.

This subroutine will sort a list of integer from smallest to largest. The numbers are given in A, and a set of indices given in IDX. There are N elements in the list. If the value of N is negative, then the routine creates the index list.

3.2.4 SUBROUTINE SORTFL

Author: Curtis A. Meyer
Creation Date: .
References:
Call Arguments: (A, IDX, N)
Common Blocks Used: None.
Called by:
Subroutines Referenced: None.

This routine is identical to SORTIL, except that it sorts a list of real numbers, rather than integers.

3.2.5 SUBROUTINE SRTREE

Author: Brigitt Schmid and Curtis A. Meyer
Creation Date: 01 Jun2, 1990.
References:
Call Arguments: (DIST, LIST, N, *NRED, *INDX)
Common Blocks Used: None.
Called by:
Subroutines Referenced: None.

Given a list of N pairs of numbers, LIST(2,N), and a distance DIST(N) corresponding to each pair, this routine will find the minimum spanning set over all the data. The value of NRED will be returned as the number of reduced pairs, and the array INDX will contain a pointer to the original two lists for each pair chosen.

As an example, consider the problem of making π^0's and η's from a list of photons. To be specific, consider a system containing four photons. From these four photons, there are six ways to form π^0's and six ways to form η's. The problem is then what is the most likely combination. If we then compute the probability P_i for each of these 12 combinations, and then define the distance as $D_i = 1 - P_i$, the routine SRTREE will return INDX pointing to the two combinations that should be taken to maximize the probability. In using this routine, it is most important to get the errors on the invariant massed correct. A suggested error for a π^0 invariant mass is:

$$error = min\left(4\sigma\left[\pi^0 width\right], max\left(1\sigma\left[\pi^0 width\right], \sigma_{m_{\gamma\gamma}}\right)\right)$$

4 Chamber Reconstruction Software

This chapter describes the software normally used to analyze the chamber data in the experiment. This chapter is divided into several subsections, each dealing with a particular section of the code. The first is an overview of all the common blocks used in this code, and a description of their contents. The remaining sections describe the various sections of the code, and contain a *flow diagram* of the calling sequence in that particular section. The code itself has been broken into the following sections: General Routines, Raw Data Processing, Pattern Recognition, Circle Fitting and Helix/Vertex Fitting. The descriptions found here are essentially those found in the header blocks of each program. In all of the subroutines described here, an asterisk, (*) in front of a call argument's name means that its value is returned by the subroutine. Also, all variable types conform to fortran defaults unless specifically mentioned.

4.1 Description of the Chamber Reconstruction Common Blocks

Here is contained a description of the common blocks used in the tracking program. In order to include one of these is a subroutine, use the PATCHY command +CDE,*name*. Where *name* is the six character name of the common block.

4.1.1 LGHOLD

The variables in this common block are set to .TRUE. if the user wants to prevent the data base from overwriting input constants.

```
      LOGICAL         ANGLLG,BMAGLG,IAMPLG,ITFCLG,OPWCLG
      LOGICAL         ZOFFLG,WIRELG
      COMMON /LGHOLD/ ANGLLG,BMAGLG,IAMPLG,ITFCLG,OPWCLG
     &               ,ZOFFLG,WIRELG
      SAVE   /LGHOLD/
```

- ANGLLG
- BMAGLG
- IAMPLG
- ITFCLG
- OPWCLG
- ZOFFLG
- WIRELG

4.1.2 RJDATA

The /RJDATA/ common block is used to convert the **RJDF** hit information into **RJDC** format. It is used in the RJPROC, RJPULS, and RJAFIT routines, and is not meaningful outside of those routines. At most four hits per pulse are allowed. Hit number zero and hit number five are only used for bookkeeping; hit zero refers to the start of the pulse, and hit five refers to the end of the pulse.

```
      INTEGER          NCHARJ,ITOFRJ,INTGRJ,ILPDRJ,IRPDRJ
      INTEGER          MINPRJ(0:5,2),MAXPRJ(0:5,2)
      INTEGER          IENDRJ(0:5,2),IDTIRJ(0:5,2)
      COMMON /RJDATA/ NCHARJ,ITOFRJ,INTGRJ,ILPDRJ,IRPDRJ,
     &                MINPRJ,MAXPRJ,IENDRJ,IDTIRJ
```

- NCHARJ is the number of FADC channels in the pulse.
- ITOFRJ is the time offset to the beginning of the pulse, (in FADC channels).
- INTGRJ is the integration time in FADC channels.
- ILPDRJ is the left pedestal, (linearized and times ten).
- IRPDRJ is the right pedestal, (linearized and times ten).
- MINPRJ is the start position of all hits found in the pulse.
- MAXPRJ is the end position of all hits found in the pulse.
- IENDRJ is the bin at the end of each hit.
- IDTIRJ is the fit drift time of each hit. IPULRJ.

4.1.3 RJPRMS

The /RJPRMS/ common block contains parameters used in the routines which process the **RJDF** data banks, (RJPROC, RJPULS, RJTFIT and RJAFIT). All the parameters in this bank can be changed during calibration with a card in the CJINIT routine, (see CJINIT).

```
      LOGICAL          LGT0RJ,LGPDRJ,LGRJDD,LGRJDF
      INTEGER          NPULRJ,ITMIRJ,ITIMRJ,MXPLRJ,ITFCRJ(16)
      COMMON /RJPRMS/ LGT0RJ,LGPDRJ,LGRJDD,LGRJDF
     &               ,NPULRJ,ITMIRJ,ITIMRJ,MXPLRJ,ITFCRJ
```

- LGT0RJ Identify if t_0 subtraction is done in RJPROC.
- LGPDRJ Identify if dynamic pedestal calculation is done in RJPROC.
- LGRJDD Take online processed data if it exists.
- LGRJDF Perform Qt-analysis on the raw JDC pulses. Ignore any existing online processed data.
- NPULRJ Minimum pulse height above the height at the pulse start for a pulse in the **RJDF** bank. It has a default value of 5, but can be changed using the **NPUL** card.
- ITMIRJ Minimum time separation between two pulses in RJPULS. It has a default value of 100ns, but can be changed using the **ITMI** card.
- ITIMRJ Integration time in the RJAFIT routine. It has a default value of 40 FADC channels, but can be changed using the **INTG** card.
- MXPLRJ The maximum time difference between the left and right channel in order to accept the two pulses as one. It has a default value of 25ns, but can be changed using the **MXPL** card.
- ITFCRJ Is a t_0 offset subtracted from the fit pulse time in the RJPROC routine. It has a nominal value set for the December, 1989 data set, but can be changed with the **ITFC** card. It is given in units of 200ps. This time offset is a function of the FADC crate number, (0-16).

4.1.4 TCANGL

The /TCANGL/ common block contains the double precision sines and cosines of all hits incorporated into tracks. These are stored in order to reduce computation time in the circle fit, (routine TCITER).

```
      DOUBLE PRECISION SINTTC(50),COSTTC(50),RDEVTC(50),CHISTC(50)
      COMMON /TCANGL/  SINTTC,COSTTC,RDEVTC,CHISTC
      SAVE   /TCANGL/
```

- SINTTC contains $\sin\phi_i$ for each of up to 50 points in the present track. Its values are loaded in the TCLOAD routine.
- COSTTC contains $\cos\phi_i$ for each of up to 50 points in the present track. Its values are loaded in the TCLOAD routine.
- RDEVTC contains the deviation of the measured points from the circle fit, (units of cm) for the fit track. Its values are set in the TCITER routine.
- CHISTC is the contribution to χ^2 of each point along the track. This is filled in the TCITER routine.

4.1.5 TCCUTS

The /TCCUTS/ common block contains all the cuts used throughout the charged tracking code. They are all initialized in the TCINIT routine.

```
      INTEGER          NITRTC,NHLXTC,NVTXTC,LYVXTC
      REAL             CXSQTC,CLMBTC,XSQRTC,DXSQTC
      REAL             TIMETC(23),RSLVTC,CHCTTC
      REAL             VPRBTC,VFRCTC,ZVTXTC
      DOUBLE PRECISION CCUTTC,CHDSTC,CHLXTC,CVXCTC
      COMMON /TCCUTS/  CCUTTC,CHDSTC,CHLXTC,CVXCTC,
     &                 CXSQTC,CLMBTC,XSQRTC,DXSQTC,TIMETC,CHCTTC,
     &                 NITRTC,NHLXTC,NVTXTC,LYVXTC,RSLVTC,
     &                 VPRBTC,VFRCTC,ZVTXTC
      SAVE /TCCUTS/
```

- CXSQTC is a chisquare cut used in the TCSGMT routine a a measure of track closeness in dy/dr. It has a default value of 1.75, and can be altered using the **CXSQ** card.
- CLMBTC is chisquare cut for associating tracks in $\tan\lambda$. It is used in the TCSGMT routine for putting the segments together, and in the TCROSS routine for associating tracks in different sectors. Its nominal value is 1.70 which can be changed using the **CLMB** card.
- XSQRTC is a χ^2 cut for connecting points in the JDC. It is used in the TCFSRC and TCRSRC routines. Its nominal value is $0.050[cm^2]$, it can be changed using the **XSQR** card.
- DXSQTC is a cut used in the TCFSRC and TCRSRC subroutines to determine if a track has crossed the wire plane. It is the difference between the left and right χ^2, and has a nominal value of 0.01. It can be changed during calibration using the **DXSQ** card.
- TIMETC is an array containing the minimum time in each JDC layer a hit can have, and still be matched with a hit in the adjacent sector. This cut is used in the TCROSS routine.

- NITRTC is the maximum number of iterations in TCITER. Its nominal value is 10, which can be altered with the **NITR** card.

- CCUTTC is used to decide if a track is well fit in the TCITER routine. If the track has converged, and the last variation in the first fit parameter is larger than 0.10, then the value of χ^2 must be smaller than CCUTTC times the number of points in the track. This has a nominal value of 6.0, but can be changed using the **CCUT** card.

- CHDSTC is the convergence criteria in the TCITER and CJITER routines. If the computed χ^2 varies by less than the value of CHDSTC between any two iterations, then the routine stops iterating. Also, if the value of χ^2 falls below the value of CHDSTC, the routine also stops iterating. This has a nominal value of 0.100, but can be changed using the **CHDS** card.

- CHCTTC is a closeness parameter, (χ^2) used to determine if two different track segments belong to the same track and is used in the TCASSC routine. Its nominal value is 4.50, but it can be modified by use of the **CHCT** card.

- NHLXTC is the maximum number of iterations in the TCHELX routine. Its nominal value is 5 which cna be changed with the **NHLX** card.

- CHLXTC is a convergence criteria in the TCHELX routine. Its nominal value is 1.0 which can be modified with the **CHLX** card.

- NVTXTC is the maximum number of iterations in the TCVRTX routine. Its nominal value is 10, but it can be changed using the **NVTX** card.

- LYVXTC is the outermost inner layer a track can have and be considered for the primary vertex. It has a default value of six, but this can be changed using the **LYVX** card.

- CVXCTC is a χ^2 cut for convergence in the TCVRTX routine. Its nominal value is $1.0[cm^2]$, but it can be changed using the **CVXC** card.

- RSLVTC is a parameter used to decide if a point can be added to an existing fit track. Its value can be set using the **RSLV** card, (See routine TCRSLV).

- VPRBTC is a probaility cut used in the vertex fit to decide if tracks should be dropped from the vertex. It has a default value of 0.001, but can be changed using the **VPRB** data card.

- VFRCTC is a cut to decide if tracks should be dropped from the vertex. If the probability is smaller than VPRBTC and one track contributes VFRCTC of the total χ^2, then that track is not used in the vertex fit. The default value is 0.900, but can be changed using the **VFRC** data card.

- ZVTXTC is a criteria in the vertex fit to decide if a track belongs to the primary vertex. The track must have z_0 within ZVTXTC of the nominal z–vertex to be used. This has a default value of 10cm, but can be changed using the **ZVTX** data card.

4.1.6 TCFLAG

The TCFLAG common contains flags which control which parts of the track reconstruction code are executed. The nominal value of all these cards is .TRUE., but they can be varied via the **CHAM** card in the main code.

```
      LOGICAL         GPWCTC
      LOGICAL         TRAKTC,RAWSTC,PATTTC,CIRCTC,HELXTC,VERTTC
      COMMON /TCFLAG/ GPWCTC,
     &                TRAKTC,RAWSTC,PATTTC,CIRCTC,HELXTC,VERTTC
```

- GPWCTC identifies if the PWC was used.
- TRAKTC controls whether CBPHYS calls TCTRAK.
- RAWSTC controls whether the routine process raw data.
- PATTTC controls whether pattern recognition is done.
- CIRCTC controls whether circle fitting is done.
- HELXTC controls whether helix fits are performed.
- VERTTC controls whether vertex fitting is performed.

4.1.7 TCHITS

The TCHITS common block is filled by the TCLOAD routine, and identifies which points in a track have not been resolved, (left/right). This is then later used by the TCRSLV routine to resolve those points.

```
      LOGICAL         LGUNTC,LGCNTC(50),LGNEAR(50)
      INTEGER         NTOTTC,NPTSTC,IPNTTC(50)
      COMMON /TCHITS/ LGUNTC,LGCNTC,LGNEAR,NTOTTC,NPTSTC,IPNTTC
```

- LGUNTC is a logical flag that is set .TRUE. if the track contains any unresolved points.
- LGCNTC is an array of logicals. Each entry corresponds to a hit in the track, and if the value is .TRUE., then the hit has been resolved.
- LGNEAR is set .TRUE. if the hit is within 1.5mm of a sense wire. TCFITR is then allowed to switch the resolution on this point if it would improve the track fit.
- NTOTTC is the total number of hits in the track.
- NPTSTC is the number of resolved hits in the track.
- IPNTTC is an array of pointers to the **TCHT** data bank for every hit in the track.

4.1.8 TCLIFT

The /TCLIFT/ common block contains bank information used in Lifting all data banks used in locater. All this information is filled in the ZBFORM routine. This means that all bank formats can be changed in the same place. The common block contains the following information. For a detailed description, consult the data bank information.

```
      INTEGER         MRJDC(5),MTJDC(5),MTJDS(5),MTCHT(5),MTCLY(5)
      INTEGER         MTCTK(5),MTCSG(5),MTCTR(5),MTCTX(5)
      INTEGER         MTCHX(5),MTCHP(5),MTCVX(5),MTCVT(5),MTCVP(5)
      INTEGER         MTPWC(5),MTPCH(5),MHTJD(5)
      INTEGER         MTJCT(5),MTJXX(5),MTJCP(5),MTJRF(5),MTJST(5)
      INTEGER         MTJCZ(5),MTJCE(5),MTJZ0(5),MTJZL(5),MTJWR(5)
```

```
 COMMON /TCLIFT/ MRJDC,MTJDC,MTJDS,MTCHT,MTCLY
&               ,MTCTK,MTCSG,MTCTR,MTCTX
&               ,MTCHX,MTCHP,MTCVX,MTCVT,MTCVP
&               ,MTPWC,MTPCH,MHTJD
&               ,MTJCT,MTJXX,MTJCP,MTJRF,MTJST
&               ,MTJCZ,MTJCE,MTJZ0,MTJZL,MTJWR
 SAVE /TCLIFT/
```

4.1.9 TCPRMS

The /TCPRMS/ common block contains parameters used in the reconstruction of tracks. All of these parameters are initialized in the TCINIT routine.

```
 INTEGER         NLYRTC,LJDCTC
 REAL            PHSETC,PHOFTC,ZOFFTC
 REAL            SINETC(30),COSETC(30),ANGSTC(30)
 REAL            RLYRTC(23),RINVTC(23),STAGTC(23)
 REAL            ANGLTC,BMAGTC,BINVTC,BMGCTC,BSGNTC,DELZTC(23)
 COMMON /TCPRMS/ NLYRTC,LJDCTC,PHSETC,PHOFTC,ZOFFTC,
&               SINETC,COSETC,ANGSTC,RLYRTC,RINVTC,STAGTC,
&               ANGLTC,BMAGTC,BINVTC,BSGNTC,BMGCTC,DELZTC
```

- NLYRTC is the total number of chamber layers, XDC/PWC and JDC.
- LJDCTC is the number of JDC layers, nominally 23.
- PHSETC is the angle in radians spanned by one JDC sector.
- PHOFTC is the offset angle in radians to the center of JDC sector 1.
- ZOFFTC is the offset of the interaction plane in the target from the center of the JDC. This parameter is used in the TJZPOS routine for computing quantities for the TCSGMT routine. It can be set with the **ZOFF** card.
- SINETC contains the sines of the rotation angle to the center of each JDC sector.
- COSETC contains the cosines of the rotation angle to the center of each JDC sector.
- ANGSTC contains ϕ angle at the center of each JDC sector in radians.
- RLYRTC contains the radius of each layer in the JDC in units of *cm*.
- RINVTC contains $1/r$ for each layer in the JDC.
- STAGTC contains the wire stagger, (in cm) for each layer in the JDC.
- ANGLTC is an offset angle for the first sector of the JDC. It specifies how many radians away from 0 it is, and can be set with the **ANGL** card.
- BMAGTC is the magnitude of the magnetic field along the z–axis in units of KG. It can be changed with the **BMAG** card.
- BINVTC is $\frac{1}{eB}$ in the units above. It is calculated in the TCINIT subroutine.

- BMGCTC is eB in units of $\frac{MeV/c}{cm}$. The value of e is

$$e = 0.299792458 \frac{MeV/c}{KG \cdot cm}$$

  Its value is computed in the TCINIT subroutine from BMAGTC.

- BSGNTC is the sign of the magnetic field. +1 is along the $+z$ axis, while –1 is along the $-z$ axis. It is computed from the value of BMAGTC.

- DELZTC is the nominal error in the measured z coordinate for each wire in the JDC. It has a nominal value of 0.8cm, but can be changed using the **DELZ** card.

4.1.10 TCSCAT

The TCSCAT common block contains the data used to compute the multiple scattering contribution to the covariance matricies. This data is initially loaded in the TCINIT routine. The initial data depends on whether the XDC or PWC is inside the JDC. A description of how the multiple scattering contribution is computed can be found in the section of the TCMCST subroutine.

```
REAL            GASXTC,R0XTTC,R1XTTC,R2XTTC
COMMON /TCSCAT/ GASXTC,R0XTTC,R1XTTC,R2XTTC
```

- GASXTC is the inverse radiation length of the gas in the JDC. It is is units of cm^{-1}.

- R0XTTC is a sum over all discrete scatterers between the interaction point and the first hit in the JDC. The sum is of the quantity t_i/x_i where t_i is the thickness of the scatterer and x_i is its radiation length.

- R1XTTC is a sum over all discrete scatterers of the quantity: $t_i \cdot r_i/x_i$ where r_i is the radius of the scatterer. The dimensions of this quantity are cm.

- R2XTTC is a sum over all discrete scatterers of the quantity: $t_i \cdot r_i^2/x_i$. It has units of cm^2.

4.1.11 TCSEGS

The TCSEGS common block contains information on the track segments found in the JDC. It is essentially an organization of the points in each sector according to $\tan\lambda$ where λ is the track opening angle. The routine TCSGMT is called to fill the data in this common block, and then the TCRAW1 and TCRAW2 routines use this information in building tracks in the chamber.

```
LOGICAL         GCNCTC(10,30)
INTEGER         NSEGTC(30),NSGPTC(2,10,30),JSECTC(4,10,30)
REAL            LMBDTC(4,10,30)
COMMON /TCSEGS/ GCNCTC,NSEGTC,NSGPTC,JSECTC,LMBDTC
```

- GCNCTC is a logical array identifying which segments have been processed. TCSGMT sets this value to .FALSE. for all segments, then as TCRAW1 and TCRAW2 use the data, they change the value to .TRUE. .

- NSEGTC is an array containing the number of segments found in each sector of the JDC.

- NSGPTC contains the number of points found in each segment, and the number of points incorporated into tracks. There is storage space for up to 10 tracks per sector.

- JSECTC contains information on the beginning and end of each found segment.

JSECTC(1,i,j) layer number of the innermost hit in segment i of sector j.

JSECTC(2,i,j) layer number of the outermost hit in segment i of sector j.

JSECTC(3,i,j) hit number of the innermost hit in segment i of sector j.

JSECTC(4,i,j) hit number of the outermost hit in segment i of sector j.

- LMBDTC contains the cuts on $\tan\lambda$ for each found segment in each sector.

LMBDTC(1,i,j) $\tan\lambda$ for the outermost hit in segment i in sector j.

LMBDTC(2,i,j) contains $\sum \tan\lambda$ for all points in segment i in sector j.

LMBDTC(3,i,j) contains the average of $\tan\lambda$ for all hits in segment i in sector j.

LMBDTC(4,i,j) $1.0/\sigma^2_{\tan\lambda}$ for segment i in sector j. $\sigma^2_{\tan\lambda} = 0.260(1.0 + \tan^2\lambda)$.

4.1.12 TCSTAT

The /TCSTAT/ common block is used to keep track of run statistics throughout the LOCATER code. For a precise description of each variable, one should consult the TCCOMMON Patch in the LOACTER PAM file. Information from this common is printed during a call to TCDONE.

```
INTEGER         ISTATC(20),ISTKTC(50)
COMMON /TCSTAT/ ISTATC,ISTKTC
```

- ISTATC contains global statistics.
- ISTKTC contains statistics from the circle fitting code.

4.1.13 TJCONV

The /TJCONV/ common block contains variables which are passed to the TJTIME, TJZPOS and TCRESL routines. This allows these routines to have no call arguments.

```
INTEGER         IRESTJ,ISECTJ,ITJDTJ,ITCHTJ
COMMON /TJCONV/ IRESTJ,ISECTJ,ITJDTJ,ITCHTJ
```

- IRESTJ is the resolution code for the point, (0,1,2).
- ISECTJ is the sector number in the JDC.
- ITJDTJ is a pointer to the **TJDC** data bank for this hit.
- ITCHTJ is a pointer to the **TCHT** data bank for this hit.

4.1.14 TJCUTS

The TJCUTS common block contains cuts applied only to the JDC. Its values are initialized in the TCINIT routine.

```
      INTEGER         IAMPTJ,IGAPTJ,JGAPTJ,ITDFTJ,ITMXTJ(23)
      REAL            TCRSTJ(23),TMINTJ,YMAXTJ(23),YCUTTJ
      REAL            DMINTJ,DMAXTJ
      COMMON /TJCUTS/ IAMPTJ,IGAPTJ,JGAPTJ,ITDFTJ,ITMXTJ,TCRSTJ,
     &                TMINTJ,YMAXTJ,YCUTTJ,DMINTJ,DMAXTJ
      SAVE   /TJCUTS/
```

- IAMPTJ is the minimum accepted amplitude in the TJDCGT routine. It has a nominal value of 45 FADC channels, but can be changed with the **IAMP** card.
- IGAPTJ is the largest gap in layers allowed within a segment in the TCSGMT routine. It has a default value of 4, but can be changed using the **IGAP** card.
- JGAPTJ is the largest gap in layers allowed in the TCRSRC and TCFSRC routines. It has a default value of 2, but can be changed using the **JGAP** card.
- ITMXTJ is the minimum drift time distance in ns as used in the TJDCGT routine. It has a value of 250ns, but can be set using the **ITDF** card.
- ITMXTJ is the maximum drift time allowed on each layer of the JDC, (ns). The values can be changed using the **itmx** card.
- TCRSTJ is a time cut used in the TCFSRC and TCRSRC subroutines when trying to resolve a second point in a track. The cut says that if we know the resolution of one point along the track, a second point can have that same resolution only if it is within TCRSTJ μs of the first point. The nominal value is $0.7\mu s$.
- TMINTJ is the minimum time a hit can have and still be resolved with the $(t_1 + t_3)/2 - t_2$ method. It is nominally set at 90 ns, but can be changed with the **TMIN** card. This cut is used in the TCRAWS routine.
- YMAXTJ contains the maximum allowed value of y for each layer in the JDC.
- YCUTTJ is used in the TJTIME routine to decide when a point is physically outside of a sector. If the point reconstructs outside the sector, but not more than the value in YMAXTJ, then it is assigned the coordinates of the sector boundary. Otherwise it is assigned an unphysically large position. The default value is 1mm, but it can be changed using the **YCUT** card.
- DMINTJ is the minimum value of $(t_1 + t_3)/2 - t_2$ with which one can identify the left or right side of the wire. This has a default value of $0.025\mu s$, but can be changed using the **DMIN** card. Note that the program will naturally decrease the value of this parameter if no track can be extracted.
- DMAXTJ is the maximum value of $(t_1 + t_3)/2 - t_2$ which can be used to decide left and right in the chamber. It has a default value of $0.130\mu s$, but can be modified using the **DMAX** card. Note that the program will naturally increase this cut if no tracks can be extracted.

4.1.15 TJPRMS

The /TJPRMS/ common block contains data for computing the position error to be associated with every hit in the JDC. See the TCRESL routine for a description of the error calculation.

```
 LOGICAL         LGT2TJ
 REAL            DELYTJ,DLY2TJ,DELTTJ,RISOTJ(23,2)
 REAL            SY02TJ,SYDFTJ,SYDQTJ,SXDQTJ
 COMMON /TJPRMS/ LGT2TJ,DELYTJ,DLY2TJ,DELTTJ,RISOTJ,
&                SY02TJ,SYDFTJ,SYDQTJ,SXDQTJ
 SAVE   /TJPRMS/
```

- LGT2TJ is a flag to control if we use a complicated form normally used in chamber calibration. Its value is set with the **LGT2** card.

- DELYTJ The linear term in the computation. It has a nominal value of 0.0250 [cm], but can be changed using the **DELY** card.
- DLY2TJ is used in calibrating the JDC. It has the function of DELYTJ in normal running. It has a default value of 0.025 [cm], and can be changed by using the **DLY2** card.
- DELTTJ the quadratic term, only applied if the flag is true. It has a nominal value of 0.0250 [cm], but can be changed using the **DELT** card.
- RISOTJ is the maximum isochron radius on each side of each wire.
- SY02TJ is the flat term in the chamber resolution, it has a nominal value of 90 microns, but can be changed using the **SY02** card.
- SYDFTJ is the diffusion term which is proportional to the square root of the drift distance. It has a nominal value of 100 micorns, but can be changed using the **SYDF** card.
- SYDQTJ is nominally zero, and must be changed using the **SYDQ** card. It is used during calibration to allow for an error in the drift distance proportional to the drift distance.
- SXDQTJ is nominally zero, and must be changed using the **SXDQ** card. It is used during calibration to allow for an error in the x coordinate proportional to the drift length, essentially a Lorentz angle.

4.1.16 TJSLCN

The /TJSLCN/ common block is used to monitor the JDC slow control events.

```
      LOGICAL         LCLRTJ
      INTEGER         NTMPTJ,NMIXTJ,NPRSTJ
      REAL            STMPTJ,SMIXTJ,SPRSTJ
      COMMON /TJSLCN/ STMPTJ,SMIXTJ,SPRSTJ
     &               ,NTMPTJ,NMIXTJ,NPRSTJ
     &               ,LCLRTJ
```

4.1.17 TJWIRE

The /TJWIRE/ common block identifies which wires can and cannot be used in analysis. It is used in the TJDCGT routine.

```
      LOGICAL         LGWIRE(23,30)
      COMMON /TJWIRE/ LGWIRE
```

If an entry, (layer,sector) in the LGWIRE array is .TRUE., then the wire should not be used in analysis.

4.1.18 TPPRMS

The /TPPRMS/ common block contains parameters which describe the PWC.

```
      REAL            RPWCTP(2),APWCTP(2),OPWCTP(2)
      REAL            XPWCTP(0:319),YPWCTP(0:319)
      COMMON /TPPRMS/ RPWCTP,APWCTP,OPWCTP,XPWCTP,YPWCTP
```

- RPWCTP contains the radius in centimeters of the two layers in the PWC.

- APWCTP contains the angular spacing between wires in the PWC for the two layers. The units are in radians.
- OPWCTP contains the angular offset for wire number zero in both layers of the PWC, the units are radians, and this can be set using the **OPWC** card.
- XPWCTP contains the x coordinate of every wire as a function of the wire number in the **RPWC** data bank.
- YPWCTP contains the y coordinate of every wire as a function of the wire number in the **RPWC** data bank.

4.2 Description of the General and Steering Software

The following subroutines are found in the TC_MAIN patch in the LOCATER card file.

4.2.1 SUBROUTINE TCDONE

Author: Curtis A. Meyer
Creation Date: 27 July, 1988
References:
Call Arguments: (ICODE)
Common Blocks Used: CJFLAG.
Subroutines Referenced: CJTAVG, CJTIMO,
CERN LIBRARY, DATIMH.

This routine is called once at the end of a run to write out the status of the tracking code. If calibrations are done, this routine calls the routines to update the look up table, and to write out the new table.

4.2.2 SUBROUTINE TCINIT

Author: Curtis A. Meyer
Creation Date: 12 February, 1988
References:
Call Arguments: (*IERR).
Common Blocks Used: CBBANK, CBLINK, TCLIFT, ZBDIVS, TCPRMS, TCCUTS, TCSCAT and CJGAIN.
Subroutines Referenced: TCVERS, TJTIMI and CJINIT.

This routine is called once, at the beginning of each run. It's purpose is to set up the constants used throughout the tracking section. The returned value IERR is zero for no problems, and set equal to one if *io* problems were encountered. In this case the calling program should terminate as the drift chamber calibration tables will be incorrect.

4.2.3 SUBROUTINE TCTRAK

Author: Curtis A. Meyer
Creation Date: 12 February, 1988
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK, TCLIFT, ZBDIVS, TCTEST and SMTEST.

Subroutines Referenced: TJDCGT, TPWCGT, TCPATT TCCIRC, TCDEDX, TCPCNT, TCHELX, TCMSCT, TCVERT and USEVNT.
ZEBRA LIBRARY MZLIFT.

This subroutine controls the flow of the tracking programs for normal data analysis. It's only job is to call other subroutines. The value of IEVNT is the event number, LGFIN is a logical that identifies if the end of data was encountered, and IERR is an error code from the routine. If IERR is returned not equal to zero, the run should be ended.

The subroutine USEVNT is called throughout this routine to enable the user to abort event processing. To do so, the user need only set the variable ACCECB in the /CBCNTL/ common block to .FALSE.. The following entries are made to USEVNT.

20 After unpacking of the PWC data, (TPWCGT).

21 After unpacking of the JDC data, (TJDCGT).

22 After pattern recognition, (TCPATT).

23 After circle fitting, (TCCIRC).

26 After helix fitting, (TCHELX).

27 After vertex fitting, (TCVERT).

29 After calibration, (CJCALB).

4.2.4 SUBROUTINE TJGAIN

Author: Curtis A. Meyer
Creation Date: 1 March, 1989.
References:
Call Arguments: (*IERR)
Common Blocks Used: CBBANK, CBLINK and TCLIFT.
Subroutines Referenced: (ZEBRA LIBRARY:) MZLIFT.

This routine will first lift the **TJCE** and **TJCZ** data banks in the constant division. It will then read in the JDC energy and z calibrations from logical unit LJGAIN and store them in the banks. The returned value of IERR is set to zero if the routine is successful.

4.2.5 SUBROUTINE TJGAOT

Author: Curtis A. Meyer
Creation Date: 1 March, 1989.
References:
Call Arguments: None.
Common Blocks Used: CBBANK and CBLINK.
Subroutines Referenced: None.

This routine will write the contents of the **TJCE** and **TJCZ** banks out to the file JDCGAIN.NEW through the logical unit LNORM. This routine is only used during calibration.

4.2.6 SUBROUTINE TJSLOW

Author: Curtis A. Meyer
Creation Date: 21 August, 1990
References:
Call Arguments: (*IERR).
Common Blocks Used: SCLINK, CBLINK, CBBANK and TCLIFT.
Subroutines Referenced: (ZEBRA): MZLIFT

This routine will monitor the slow control data from the JDC, and print out warnings when it deviates by two much from the reference data as stored in the **TJRF** data bank. The routine is also supposed to correct the $r\phi$ calibrations based on temperature, pressure and gas mixture as seen in the JDC. This is not yet implemented.

4.2.7 SUBROUTINE TJTIMI

Author: Klaus Peters
Creation Date: 9 September, 1988
References:
Call Arguments: (*IERR,MKxxxx).
Common Blocks Used: CBLINK, CBBANK and TCLIFT.
Subroutines Referenced: (ZEBRA): MZLIFT

This subroutine will read in the lookup table to convert drift time into position in the JDC. If an *io* error occurs, then the value of IERR is returned as one. Otherwise it is returned as zero. The data is stored in the **TJCT** data bank, which is lifted in this routine. The function MKXXXX is used to tell TJTIMI from where the input should come. The possible functions included in LOCATER are:

- MKDUMY A do–nothing routine used when data is taken from the data base.
- MKREAL is the generic name for calibration sets for real data. The correct sets are chosen using CMZ select flags.
- MKGEAN is the calibration table for GEANT output.
- MKDFPL is a starting table for $r\phi$ calibrations if the field is positive.
- MKDFMN is a starting table for $r\phi$ calibrations if the field is negative.

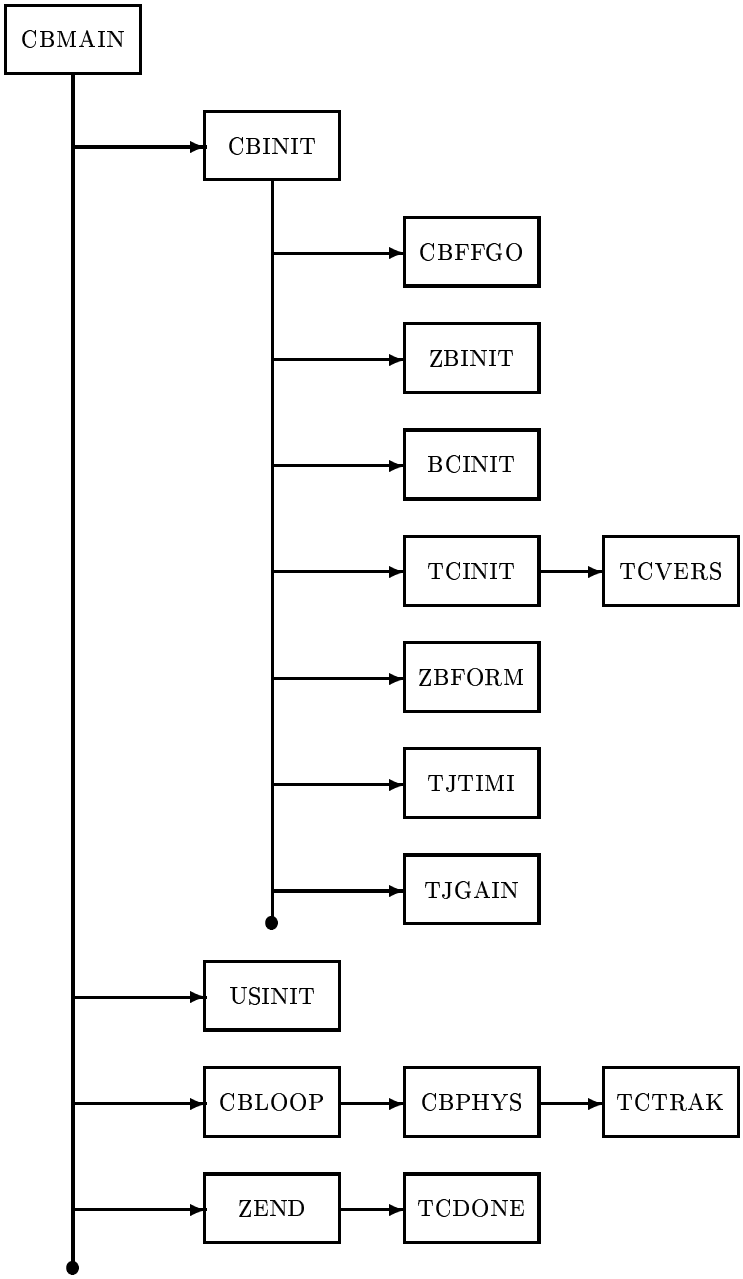


Figure 3: The calling sequence for the offline analysis code.

4.3 Description of the Raw Data Processing Software

The following subroutines are found in the TC_RAWS patch of the LOCATER card file.

4.3.1 SUBROUTINE RJAFIT

Author: Curtis A. Meyer
Creation Date: 8 May, 1989
References: Online routine **AMP1** by H. von der Schmitt.
Call Arguments: (*LHIT1, *LHIT2, *LGHIT, *IERR, *IAMPR, *IAMPL, NPULS, MXPLD, IBEGL, IBEGR, IAOFL, IAOFR)
Common Blocks Used: RJDATA.
Subroutines Referenced: RJPULS.

This subroutine computes the amplitude of a pulse by integrating the FADC contents for a fixed time, (ITIMRJ in the /RJPRMS/ common). Integration is performed simultaneously on both the left and right channels, with the resulting amplitudes being returned in the IAMPL and IAMPR variables. Integration is started at channel number IBEGL on the left, and IBEGR on the right. The routine also looks for the appearance of a second pulse during integration by identifying if either pulse starts falling, and then starts rising again. If this happens, the RJPULS routine is called, and if it finds a pulse, the LHIT variables are set true. If a second pulse is found, then the integration of the first pulse is stopped early, and its total amplitude is corrected for the fraction of time remaining. This correction is also returned in the IAOFL and IAOFR variables for subtraction from the second pulse. The returned amplitudes are linearized.

The criteria for a second pulse is given as the following:

- The maximum of the present pulse has been reached.
- The channel ISEPRJ channels away from the present channel is over for non–linear bins higher than the present channel.
- The RJPULS routine says that it is a second pulse.

4.3.2 SUBROUTINE RJPROC

Author: Curtis A. Meyer
Creation Date: 8 May, 1989
References:
Call Arguments: (*IERR).
Common Blocks Used: RJDATA, CBBANK, CBLINK, and CBUNIT.
Called by: TJDCGT **Subroutines Referenced:** RJPULS and RJAFIT.
ZEBRA LIBRARY: MZLIFT and MZPUSH.

This routine is the top level routine in the section to convert JDC data from the **RJDF** format into the **RJDC** format. The routine is called from the TJDCGT routine when no **RJDC** bank exists, but an **RJDF** bank does exist. The output from this routine is an **RJDC** data bank.

The routine loops over all pulses found in the **RJDF** bank, and processes them in exactly the same manner as the online system. The pulses are extracted from the bank, linearized using the formula:

$$l = (64 \cdot i)/(64 - 0.7539 \cdot i)$$

and stored in two working arrays, one for the left side and one for the right side of the pulse. The routine then calls the RJPULS for both the left and right pulse, which returns the number of found

hits, the start position of each hit, the position of the maximum for each hit, and the fit time for each hit, (the times are in units of 200ps). (The returned variables are passed through the /RJDATA/ common block.)

The routine then looks simultaneously at the hits on the left and right side. Two hits which are within MXPLRJ time units of each other are assumed to the two sides of the same hit. The drift time for these is taken as the average of the left and right time. Two hits which are farther apart then MXPLRJ, but closer than 100ns are assumed to be the same hit, but the drift time is taken as only the earlier time. Otherwise hits are assumed to be separate.

When both sides of a hit have been identified, the routine defines an integration length. This is taken as the shortest of either ITIMRJ channels, or the difference between the start time of the present hit and the next hit in the pulse. This is done for both the left and right side, and the final integration time is taken as the minimum of the two. The routine RJAFIT is then called to simultaneously integrate the two hits over the given integration length. The resulting drift time, and amplitudes are then stored in the RJDC data bank.

The returned value of IERR is used to flag the routine exit status.

IERR= 0 is normal completion with no errors.

IERR= 1 tags a header problem. The routine was unable to unpack the **RJDF** bank and has exited.

4.3.3 SUBROUTINE RJPULS

Author: Curtis A. Meyer
Creation Date: 8 May, 1989
References: Online routine **PULSDF** by G. Eckerlin.
Call Arguments: (ISIDE, *NPULS, IPULS).
Common Blocks Used: RJDATA and RJPRMS.
Called by: RJPROC.
Subroutines Referenced: KRATE.

This routine looks at the pulse stored in the IPULS array to find the starts, maximums and drift times of all hits. The search looks for one rising bin followed by a bin which does not fall. This is defined as a hit start. The hit maximum is then defined as the bin just before the first falling bin. Hits must be separated by at least ITMIRJ bins. If a hit is found, then the drift time is fit using a first electron method. A line is fit to the rising edge of the pulse, and the intersection with the pedestal is defined as the drift time, (time units are 200ps). The information on the starting bin, maximum bin, and drift time are returned to the calling routine through the /RJDATA/ common block. The number of hits found is returned as NPULS. The following conditions must be met in order for a hit to be accepted as a hit, (all values are found in the /RJPRMS/ common block):

- The pulse height as defined as the maximum minus the minimum must be larger than NPULRJ, (Default value of 30, but can be changed using the **NPUL** card.
- The number of bins between the start of the present hit and that of the previous, (for more than one hit), must be at least NTMIRJ, (Default value of 10, but can be changed using the **NTMI** card).
- There can be no more than four hits in a passed pulse.
- A second hit must be at least 80% as high as the previous hit.

4.3.4 SUBROUTINE TJDCGT

Author: Curtis A. Meyer
Creation Date: 1 February, 1989.
References:
Call Arguments: (IERR).
Common Blocks Used: CBBANK, CBLINK, TCLIFT, ZBDIVS and MJDATA.
Called by: TCTRAK.
Subroutines Referenced: RJPROC and SORTIL,
(ZEBRA LIBRARY:) MZLIFT and MZWORK.

This routine is called once per event. It will take the JDC data out of the magnetic tape format, and load it into the **TJDC** data banks. The format of the raw data can be either the **RJDC** or **RJDF** bank format. If the **RJDC** format exists, it is used by default. However, if the **RJDC** does not exist, and the **RJDF** does exist, the the RJPROC routine is called to create an **RJDC** data bank.

This routine then examines all hits in the **RJDC** bank, and moves those that pass the following cuts into the **TJDC** data banks. All accepted hits must satisfy the following:

- The total amplitude of the hit is larger than the IAMPTJ cutoff in the /TJCUTS/ common block.
- The drift time of the hit must be smaller than the value of TMAXTJ(LAYER) in the /TJCUTS/ common block.
- If more than one hit is found on any wire, the minimum time separation between the hits must be larger than TDIFTJ in the /TJCUTS/ common block. If they are closer than this, they are assumed to be the same hit, and are merged.

Finally, if more than one hit remain on any wires, the SORTIL routine is called to arrange those hits in order of drift times, smallest to largest.

4.3.5 SUBROUTINE TJTIME

Author: Curtis A. Meyer
Creation Date: 8 July, 1989.
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK, TJCONV, TJCUTS and TCPRMS.
Called by: TJDCGT.
Subroutines Referenced: None.

This subroutine will take the time and amplitude information given in the **TJDC** data bank through the pointer ITJTTJ and compute the (x, y) position in the JDC for both the left and right possibilities, the z coordinate through charge division, and the dE/dx at the point. The results are stored in the **TJDC** data bank using the pointer ITJTTJ. The x and y positions are obtained by interpolating in a look up table stored in the **TJCT** data bank which relates measured drift time to position for every wire in the chamber. The positions are computed in sector coordinates, as shown in Figure 4. A standard sector is defined with the x–axis along the wire plane, and the y–axis vertical. The origin is defined as the center of the JDC. Note that all pointers are passed in the /TJCONV/ common block.

The z coordinate is computed through charge division, using constants stored in the **TJCZ**, **TJZL** and **TJZ0** data banks. The dE/dx at the hit is computed using data stored in the **TJCE** data banks. For more information on how conversions are performed for these coordinates, see the subsection of *Calibration Data Banks*.

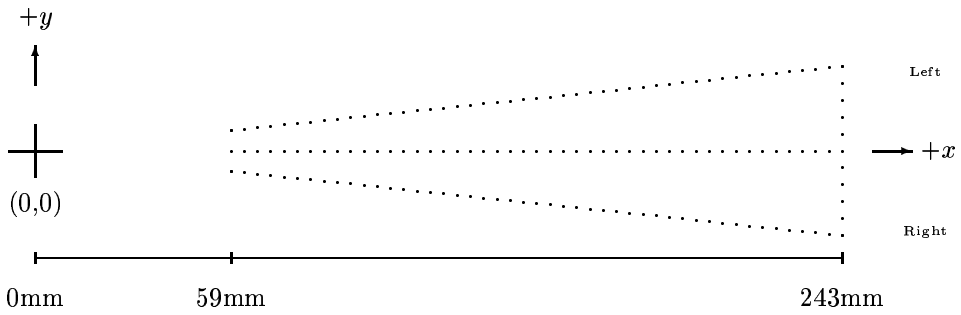


Figure 4: Sector coordinates for the JDC, all interpolation for position from drift times is done in a cell of the above form.

4.3.6 SUBROUTINE TPWPOS

Author: Curtis A. Meyer
Creation Date: 13 January, 1989.
References:
Call Arguments: None.
Common Blocks Used: TPPRMS, CBBANK, CBLINK, TCLIFT and ZBDIVS.
Subroutines Referenced: (ZEBRA LIBRARY) MZLIFT .

This routine takes the information in the **RPWC** data bank, converts the hits into position, and stores the resulting information in the **TPWC** data bank.

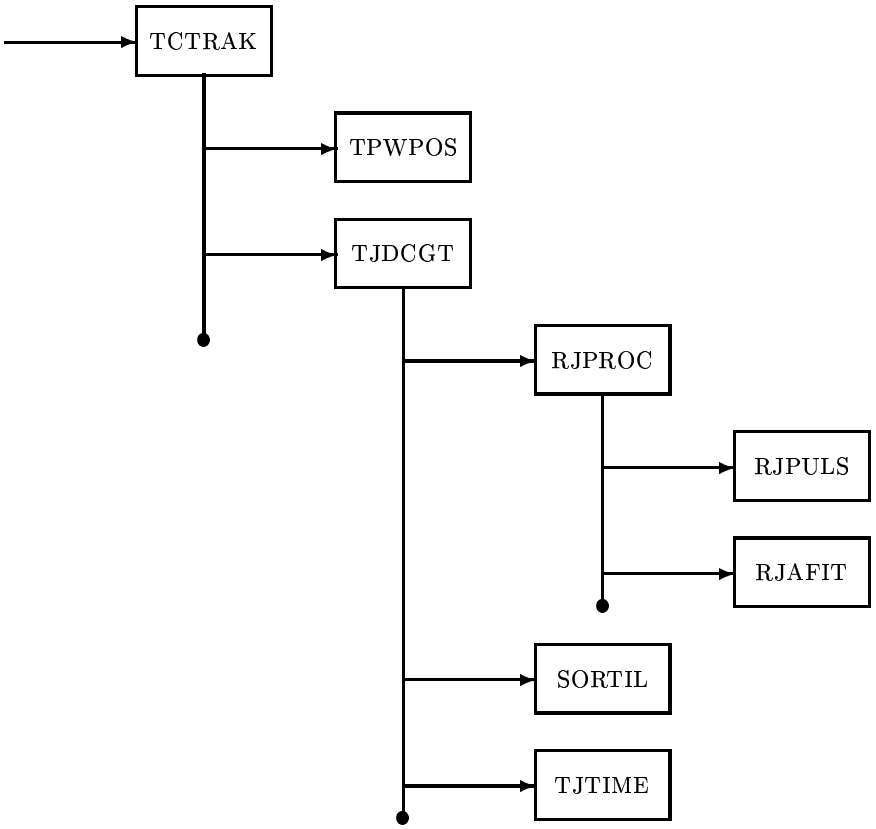


Figure 5: The flow diagram for the processing of Raw data in the JDC.

4.4 Description of the Pattern Recognition Software

The following subroutines are found in the TC_PATT patch of the LOCATER card file.

4.4.1 SUBROUTINE TCFSRC

Author: Curtis A. Meyer
Creation Date: 1 February, 1988
References:
Call Arguments:(I, ISECT, IRES, ITJDC, ITCHT, IOPT, ITRK, *CHX, *NCHX).
Common Blocks Used: CBBANK, CBLINK,TCSEGS and TCCUTS.
Subroutines Referenced: TCRESL.

This subroutine starts at the point I in sector ISECT which has been resolved as specified by IRES, (1=left, 2=right) and searches forward through the sector to resolve as many additional points as possible. The procedure uses two points of known resolution to predict the third point, and then compare this with the two possible choices. The routine stops searching if all the data is used up, or if a gap larger than 3 layers is found. ITJDC and ITCHT point to the hit in the **TJDC** and **TCHT** data banks respectively. IOPT can take the values 0 or 1. When it is zero, TCFSRC assumes that only the point given is known on the track. It then has to figure out a second point on the track before proceeding. If IOPT is 1, then TCFSRC uses the forward pointer of this hit to get the next hit, and then starts its search. Finally, ITRK is the track number assigned to the track.

The basic algorithm in this routine, (as well as the TCRSRC routine) is to use two points of known resolution, (x_1, y_1) and (x_2, y_2) to predict a third point (x_3, y_3) using a linear projection. This third point is taken as the next point in the present sector, and the distance squared between the left and right choices to the line are computed. If exactly one of these is smaller than the value of XSQRTC in the /TCCUTS/ common block, then that choice is taken. If both are smaller than the cutoff, and the absolute difference between the two is less than DXSQTC in the /TCCUTS/ common block, then the routine needs to figure out if the track is very close to, or actually crossing the sense wire plane. This is done by stepping ahead in this sector, and identifying three adjacent points whose drift times are longer than TCRSTJ in the /TCCUTS/ common block. These points are then resolved using the quantity $\Delta = (t_1 + t_3)/2 - t_2$, as described in the TCRAW2 routine. This routine is not restricted to only points identified as in the same segment, (see TCSGMT), as the first point, but examines all points in the sector, and chooses the best match in r/phi.

The returned value of CHX is the sum of the deviations squared of every added point, while NCHX is the number of added points.

4.4.2 SUBROUTINE TCPATT

Author: Curtis A. Meyer
Creation Date: 15 November, 1989
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK, TCLIFT, ZBDIVS, TCPRMS, TCCUTS and TCSEGS.
Called By: TCTRAK .
Subroutines Referenced: TCROSS .

This is the steering routine for pattern recognition. It first calls the TCSGMT to build loose roads through the data. Then it sets a minimum road length of 10 hits, and calls the TCRAW2 routine to search through the chamber, and try to resolve those roads with more than 10 hits. Next, the TCRAW1 routine is called to try to connect tracks which have crossed sector boundaries. Finally,

the TCRAW2 routine is called again with the minimum length set to three to try and sweep up left over track pieces.

After the above search, the routine then copies all unused data in the **TJDC** bank into the **TCHT** bank as unresolved and not connected. Presently, if more than 10 tracks are found in this routine, the event is aborted by calling MZWIPE on the tracking division.

4.4.3 SUBROUTINE TCRAW1

Author: Curtis A. Meyer
Creation Date: 15 January, 1988
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK, TCLIFT, ZBDIVS, TCPRMS, TCCUTS and TCSEGS.
Subroutines Referenced: TCROSS .

This routine looks through all the segments located in the TCSGMT routine, and locates all places where segments in two adjacent sectors can be connected into a track. If such a connection can be made, it has the advantage of unambiguously resolving from which side of the sense wire the hits came. This search is done by forming a loop over the 30 sectors in the JDC, and then a loop over every segment in the sectors. Given a segment either starts on some layer other than layer 1, or ends on some layer other than layer 23, a search is made over all the segments in the adjacent two sectors. One wants to find a segment which matches the $\tan\lambda$ of the first segment and can sensibly be connected to the segment in terms of layers. If such a segment is found, the TCROSS routine is called to connect. The TCROSS routine will itself call the TCRESL routine to resolve the hits, and the TCRSRC and TCFSRF routines to connect more points to the given track start.

The above search is performed as two loops over the chamber. The first forces the starting segment to have at least 6 hits in it. The second then picks up everything which is left over.

4.4.4 SUBROUTINE TCRAW2

Author: Curtis A. Meyer
Creation Date:
References:
Call Arguments: NMIN.
Common Blocks Used: CBBANK, CBLINK, TCLIFT, ZBDIVS, TCPRMS, TCCUTS and TCSEGS.
Subroutines Referenced: TCRESL, TCFSRC, TCRSRC, TCCNCT,
(ZEBRA LIBRARY): MZLIFT.

This routine looks through all sectors in the JDC and examines those roads, (TCSGMT) which contain at least NMIN hits. The time difference algorithem, $\Delta = (t_1 + t_3)/2 - t_2$ is used to resolve the left–right ambiguity in the chamber. This algorithem requires that three adjacent wires have signals, and the drift time of these signals is larger than TMINTJ. If this is true, then Δ is formed. If Δ is larger than DMINTJ and smaller than DMAXTJ, then this routine can identify which side of the wire plane we are on. (All the above cut are in the /TJCUTS/ common block.)

This routine searches through all the hits until three adjacent hits satisfy the above requirements. These three hits are then resolved, and the TCFSRC and TCRSRC routines are used to connect other points into the track. If both of these routines fail to add points to the track, and the road length is at least 10, then the routine has made an error, and the track start is dropped. The routine then continues looking through the data until it finds three more adjacent hits to use.

If after searching through a sector, no tracks are found, then the values of TMINTJ and DMINTJ are lowered and DMAXTJ increased in steps to the values of $0.025\mu s$, $0.020\mu s$ and $0.300\mu s$ respectively,

and the search is repeated. This allows the routine to find high momentum tracks which skim along a wire plane.

Finally, every track found in this routine has a 1 stored in the **TCTK** data bank at position IQ(ITCTK+7) to indicate that it did not cross a sector boundary.

4.4.5 SUBROUTINE TCRESL

Author: Curtis A. Meyer
Creation Date: 25 February, 1988
References:
Call Arguments: None.
Common Blocks Used: CBBANK, TJCONV, TJPRMS and TCPRMS.
Subroutines Referenced: None.

This subroutine uses the value of IRESTJ, (0=unresolved, 1=left, 2=right) and then converts the point referenced by ITJTTJ from sector coordinates into CB–coordinates, and stores the results in the **TCHT** data bank using ITCHTJ as a pointer. (For a definition of sector coordinates, see Figure 4.) The conversion involves simply rotating the *standard* sector by the angular offset of the sector, and in order to facilitate this rotation, the sines and cosines of the angle to every sector have been precalculated and stored in the /TCPRMS/ common block. All pointers to this routine are passed through the /TJCONV/ common block.

The routine also computes an error for every measurement and stores that in the data bank as well. Given that the r and ϕ coordinates of every hit are, the nominal error calculation is given by:

$$r = \sqrt{x^2 + y^2}$$

$$\phi = \tan^{-1}(y/x)$$

the errors can be given as:

$$\sigma_r = \mid \frac{y}{r} \mid \cdot \sigma_y$$

$$\sigma_\phi = \mid \frac{x}{r^2} \mid \cdot \sigma_y$$

where σ_y is the measurement error of the position, and is given by DELYTJ in the /TJPRMS/ common block.

If the code has been compiled using the calibration flags, then the value of σ_y can be modified. This is done by setting the LGT2TJ flag in the /TJPRMS/ common block to .TRUE., (use the **LGT2** card in CJINIT. If this is done, the value of σ_y is given as:

$$\sigma_y = \sigma_{y0}^2 + t_D \cdot \sigma_t$$

where σ_{y0} is TLY2TJ, t_D is the drift time in microseconds, and σ_t is given as DLYTTJ in the /TJPRMS/ common block. During calibration, they can be set using the **TLY2** and DELT cards in the CJINIT routine.

4.4.6 SUBROUTINE TCROSS

Author: Curtis A. Meyer
Creation Date: 25 January, 1988
References:
Call Arguments: (ISECT, KSECT, I, K, *IRES, *KRES, *LGFND)
Common Blocks Used: CBBANK, CBLINK, TCLIFT, ZBDIVS, TCSEGS and TCCUTS.
Subroutines Referenced: TCRESL, TCFSRC and TCRSRC,
ZEBRA LIBRARY MZDROP.

This subroutine looks for sector crossings between sectors ISECT and KSECT. By definition, ISECT is the sector containing the innermost layer. A sector crossing can occur only if the times in the layers are larger than a minimum time given in the TIMETC array. If a crossing can be found, then the TCRESL routine is used to resolve the two adjacent points, and then the routines TCFSRC and TCRSRC are used to extend the resolved regions of the tracks as far as possible. I and K refer to the segment number of the hits in sectors ISECT and KSECT respectively. IRES and KRES are the left–right resolution for the two hits, and if they can be resolved, the LGFND is returned as .TRUE..

4.4.7 SUBROUTINE TCRSRC

Author: Curtis A. Meyer
Creation Date: 1 February, 1988
References:
Call Arguments:(I, ISECT, IRES, ITJDC, ITCHT, IOPT, ITRK, *CHX, *NCHX).
Common Blocks Used: CBBANK, CBLINK,TCSEGS and TCCUTS.
Subroutines Referenced: TCRESL.

This subroutine starts at the point I in ISECT which has been resolved as specified by IRES, (1=left, 2=right) and searches backwards through the sector to resolve as many additional points as possible. The procedure is to use to points of known resolution to predict the third point, and then compare this with the two possible positions. The routine stops searching if all the data is used up, or if a gap larger than 3 layers is found. ITJDC and ITCHT point to the hit in the **TJDC**, and **TCHT** data banks respectively. IOPT can take the values 0 or 1. When it is zero, TCRSRC assumes that only the point given is known on the track. It then has to figure out a second point on the track before proceeding. If IOPT is 1, then TCRSRC uses the backward pointer of this hit to get the next hit, and then starts its search. Finally, ITRK is the track number assigned to the track. See the TCFSRC routine for a more detailed description.

4.4.8 SUBROUTINE TCSGMT

Author: Curtis A. Meyer
Creation Date: 27 January, 1989
References:
Call Arguments: (*LGHIT).
Common Blocks Used: CBBANK, CBLINK,TCSEGS, TJCONV and TCCUTS.
Subroutines Referenced: TJTIME and TJZPOS
(ZEBRA LIBRARY): MZLIFT.

This routine builds roads through the data in the **TJDC** data bank. The roads are built using two criteria. The first is that the value of $\tan\lambda$ for every point in a road be close to the average value for the entire road, and the second is that the value of dy/dr not change suddenly from point to point in the road. The opening angle of every point λ is computed as

$$\tan\lambda = z/r,$$

where r is the radius of the hit wire, and z is computed through charge division. Every point included must satisfy the condition:

$$\chi^2 = \frac{(\overline{\tan\lambda} - \tan\lambda_i)^2}{\sigma^2_{\tan\lambda}} < \lambda_{cut},$$

where the cutoff is given by CLMBTC in the /TCCUTS/ common block. It is also necessary that the quantity $dy/dr = (\Delta y_l/\Delta r$ not change by more than about 0.3 unless the hits are close to a wire.

The routine looks at every hit in each layer, and compares them to every existing segment, (road) found so far. It then chooses the combination of connections which minimizes the overall χ^2 of the problem. This is done by forming a matrix of χ^2 for every combination, and choosing the route that gives the minimum. At present, the routine has trouble if there are more than two tracks close in $\tan\lambda$ in the same sector in the JDC. If this turns out to occur quite often, a minor fix will need to be implemented. At present, if this occurs, a waring message is printed to the logical unit LERR.

4.4.9 SUBROUTINE TJDROP

Author: Curtis A. Meyer
Creation Date: 01 December, 1989
References:
Call Arguments: (ITRK).
Common Blocks Used: CBBANK, CBLINK.
Called by: TCRAW2, TCROSS.
Subroutines Referenced: ZBERA MZDROP.

This routine will drop a track at the pattern recognition level. The dropped track is assumed to be the last track lifted, and if this is not the case, severe errors can occur, (this is checked for in the routine). Dropping at the pattern recognition level implies disconnecting the hits in the **TJDC** bank, and removing the hits from the **TCHT** banks. The user should not try to use this routine.

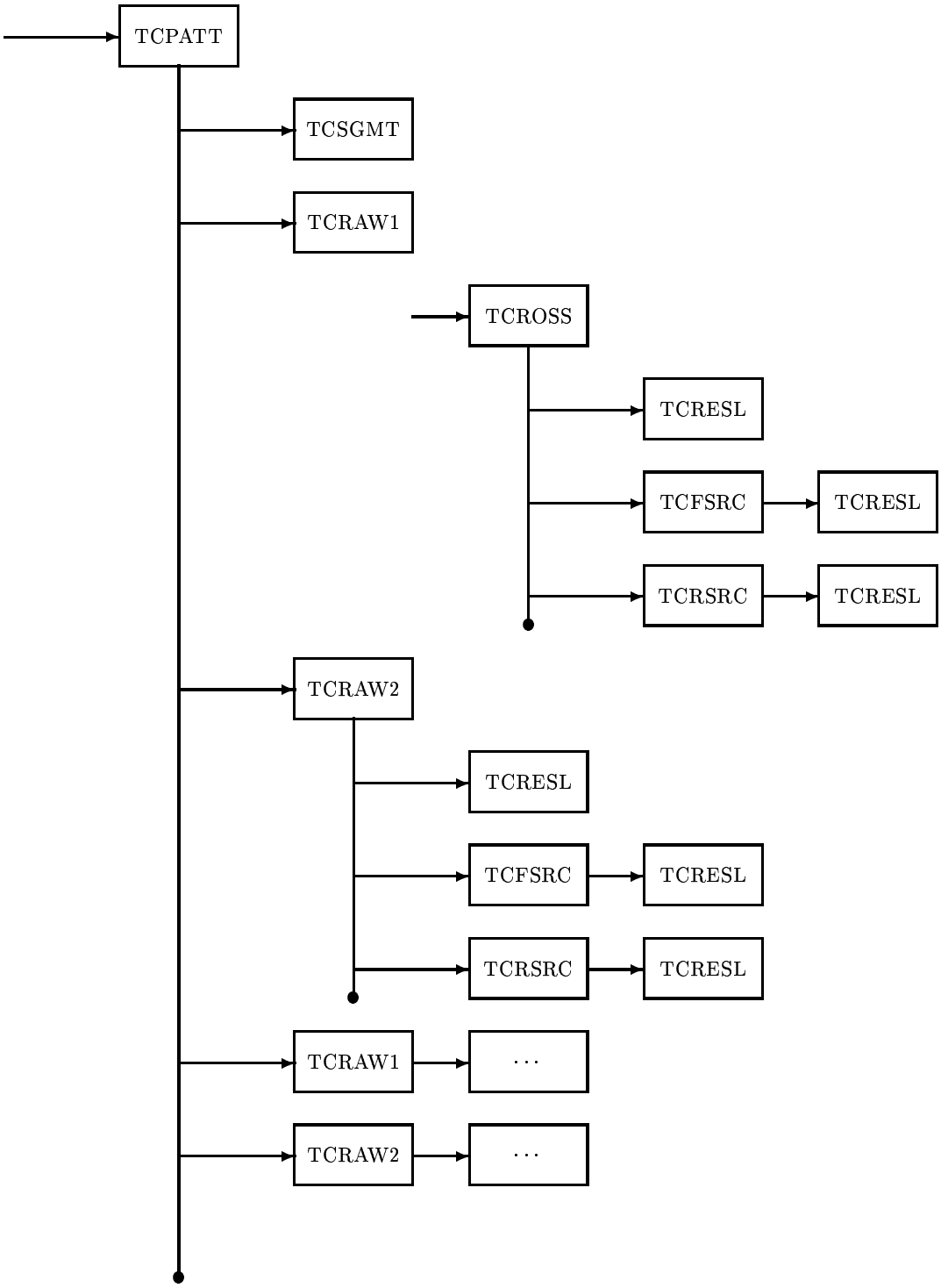


Figure 6: Flow of chamber software, pattern recognition section.

4.5 The Circle Fitting Software

The following subroutines are found in the TC_CIRC patch of the LOCATER card file.

4.5.1 SUBROUTINE TCADD

Author: Curtis A. Meyer
Creation Date: 16 November, 1989
References:
Call Arguments: (ITRK, IRSL, ITCHT).
Common Blocks Used: CBBANK, CBLINK and TJPRMS.
Called by: TCSWEP.
Subroutines Referenced: None.

This routine will add the point at **itcht** to the track ITRK with resolution IRSL. This routine can only be called by TCSWEP, and should not be generally used.

4.5.2 SUBROUTINE TCAHIT

Author: Curtis A. Meyer
Creation Date: 16 November, 1989
References:
Call Arguments: (LYR, X, XERR, *KTCHT, *IMIN, *IRSL, *LGADD, IS, IDS)
Common Blocks Used: CBBANK, CBLINK and TCCUTS.
Called by: TCSWEP.
Subroutines Referenced: TCHECK.

This subroutine will loop over all hits in layer LYR, and see which ones which are unresolved and finds the point KTCHT and the resolution IRSL which minimizes the distance to the track specified by the circle fit parameters X and XERR. The parameters IS and IDS specify the sector number of the previous hit, and the allowed variation in sector numbers of the added hit. IMIN is returned as the hit number of the matched hit.

4.5.3 SUBROUTINE TCASSC

Author: Curtis A. Meyer
Creation Date: 18 November, 1987
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK, TCLIFT, TCPRMS, TCCUTS and TCANGL.
Subroutines Referenced: TCLOAD, TCCNCT, TCITER, and TCPLOW.

This routine is designed to examine all tracks found in the pattern recognition section, and use the fit parameters as obtained from the TCFITR routine to determine if any of the tracks are really the same track.

In the first part, all tracks containing fewer than three points are dropped using the TCCNCT routine. Then the remaining tracks are ordered by the layer number of the innermost hit in each, smallest layer number to largest, and stored in a track list..

In the second section, a search is made through all tracks in the track list. Each track is examined to see if it could be sensibly have one of the later tracks added to the end of it. If so, then

a comparison is made with all tracks occurring after the first track in the track list to see if they are close to each other. This closeness is computed through the quantity:

$$\chi^2 = \Delta p_\perp^2/\sigma_{p_\perp}^2 + \Delta\psi_0^2/\sigma_\psi^2$$

where

$$\sigma_{p_\perp}^2 = \sigma_{p1}^2 + \sigma_{p2}^2$$

and

$$\sigma_\psi^2 = \sigma_{\psi 1}^2 + \sigma_{\psi 2}^2.$$

If this χ^2 is smaller than a cutoff, CHCTTC in the /TCCUTS/ common block for any pair of tracks, then the routine uses the TCLOAD routine to put both tracks into a temporary track, and then calls TCITER to see if the track can be fit. If TCITER returns with no error, the the TCCNCT routine is used to connect the two tracks. If the fit is not good, then no connection is made. For the cases where one or both of these tracks have fewer than six points, or an error code from TCFITR larger than 3, the above χ^2 is defined as

$$\chi^2 = 2 \cdot (min(\Delta\psi_0^2/\sigma_\psi^2, \Delta p_\perp^2/\sigma_{p_\perp}^2)).$$

This redefinition is used because even in a very poor fit, one of the two quantities is usually correct within errors.

During the above search, a list is compiled of all tracks with $p_\perp$ smaller than 100 MeV/c, or that are short and disconnected, *ie*, which start in the middle layers of the JDC. This list is passed to the TCPLOW routine, which then tries to associate these tracks in a method which is better suited for low momentum tracks.

4.5.4 SUBROUTINE TCCIRC

Author: Curtis A. Meyer
Creation Date: 15 November, 1989
References:
Call Arguments:
Common Blocks Used: CBBANK, CBLINK and TCANGL.
Called by: TCTRAK.
Subroutines Referenced: sc tcfitr, TCASSC, TCSWEP, TCTHET and TCIFIX.

This routine is the steering routine for circle fitting in the JDC. It first calls the TCFITR routine to fit all tracks found in pattern recognition. It then uses the TCASSC routine to try and put these tracks together into larger tracks. Then the pointers to all the tracks are loaded into a work area, and the TCSWEP routine is used to try to add additional points to the tracks. The TCTHET routine is then called to fit the track in z and ϕ, and then the TCIFIX routine is called to correct the coordinates based on their crossing angle through the chamber. Finally, the tracks are stored back into the appropriate data banks.

4.5.5 SUBROUTINE TCCNCT

Author: Curtis A. Meyer
Creation Date: 22 April, 1988
References:
Call Arguments: (ITRK1, ITRK2, I, ITCTK, JTCTK).
Common Blocks Used: CBBANK, CBLINK and TCANGL.
Subroutines Referenced: ZEBRA MZDROP and ZSHUNT.

This routine attaches track 2 whose number is ITRK2 to the end of track 1, (ITRK1). I is the number of hits in track 1, ITCTK is the address of track 1 in the **TCTK** data bank and JTCTK is the address of track 2 in the **TCTK** data bank. The connection consists of setting the forward and backward pointers in the **TCHT** data bank to connect the two tracks, adjusting the total number of hits in track 1, and finally dropping the track 2 data bank. After calling this subroutine the total number of found tracks is decreased by one. This routine does not set the track number of all the hits in track 2 to be ITRK1.

If ITRK1 and ITRK2 are the same track, then this routine drops the **TCTK** data bank for track ITRK1.

4.5.6 SUBROUTINE TCDEDX

Author: Klaus Peters and C. Strassburger
Creation Date: 20 December, 1989.
References:
Call Arguments:
Common Blocks Used: /CBBANK/, /CBLINK/ and /CJEXFT/.
Called by: TCTRAK .
Subroutines Referenced: TCRHIT and TCOORD,
CERNLIB: FLPSOR.

No documentation available.

4.5.7 SUBROUTINE TCDISC

Author: Curtis A. Meyer
Creation Date: 30 October, 1989
References:
Call Arguments: (ITRK)
Common Blocks Used: CBBANK, CBLINK and TCANGL.
Subroutines Referenced: TCCNCT.

This routine will completely disconnect all hits of track ITRK in the **TCHT** data banks, and then use the TCCNCT routine to drop the track.

4.5.8 SUBROUTINE TCDROP

Author: Curtis A. Meyer
Creation Date: 26 May, 1988
References:
Call Arguments: (ITRK, IPNT, *IERR).
Common Blocks Used: CBBANK and CBLINK.
Subroutines Referenced: None.

This routine will remove the point IPNT from the track ITRK. The removal consists of connecting the two adjacent hits in the **TCHT** bank over this hit, and decreasing the total number of hits in this track by 1. The returned value of IERR is 0 for normal completion and 1 when an error is encountered.

4.5.9 SUBROUTINE TCFITR

Author: Curtis A. Meyer

Creation Date: 14 September, 1987
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK, TCLIFT, TCPRMS, TCCUTS and TCANGL.
Subroutines Referenced: TCLOAD, TCSPLT TCRSLV and TCITER.

This subroutine loops through all the found tracks in the TCRAWS routine and uses the TCLOAD routine to put the hit information into a working area and to make a guess for the circle which passes through three of the points. It then passes this information to the TCITER routine. If the resulting fit from TCITER is considered poor, then the TCSPLT routine is called to either drop points out of the track, or split the track up. The final fit results are then stored in the **TCTK** data banks.

The call to TCITER will usually improve upon the guess from TCLOAD. However, sometimes problems do occur. The TCITER returns an error code, IERR as well as its improved results. TCFITR then processes the error code as follows:

- IERR=0,1 means that the fit was good. The results are simply stored in the **TCTK** data bank using the TCLOAD routine.
- IERR=3 means that the resulting χ^2 of the TCITER fit is too large. TCFITR then calls the TCSPLT routine to try and either remove points, or break the track into several pieces. If TCSPLT returns an ICODE=0, (see TCSPLT for descriptions of ICODE values), then IERR is set to 2, and the track is stored. If ICODE=1,2, then the track is refit using TCITER, and if ICODE=3, the track is simply stored with IERR=3.
- IERR=4,5,7 means convergence was not reached in TCITER. The TCSPLT routine is called and if ICODE=1,2 the track is refit. If ICODE=0,3, the track is simply stored using TCLOAD. There is one special case in which IERR=7 and ICODE=0,3. IERR=7 means the error matrix was singular, and could not be inverted to yield the covariance matrix. In this case, the covariance matrix is assigned rather large values.

4.5.10 SUBROUTINE TCHECK

Author: Curtis A. Meyer
Creation Date: 27 April, 1989
References:
Call Arguments: (ITCHT, X, RIN, XERR, *CHL, *CHR, *CHCUT).
Common Blocks Used: CBBANK, CBLINK and TCCUTS.
Subroutines Referenced: None.

This routine will examine the hit in the **TCHT** data bank pointed to by the ITCHT pointer, and determine if it can fit on the circle described by the circle parameters X whose errors are given in XERR, ($\vec{x} = (\kappa, \psi 0, c^2)$). RIN is passed as $1/2 \cdot x(1)$, and the routine returns CHL and CHR, as the distance in centimeters from the left and right hand solution to the circle respectively.

The distance from a point, (x,y) and a circle of radius ρ, centered at the point (a,b) is given by the formula:

$$d = \sqrt{(x-a)^2 + (y-b)^2} - \rho.$$

In our circle parametrization, the values of a, b and ρ are given as:

$$\rho = \sqrt{(\frac{1}{2 \cdot \kappa})^2 - c^2}$$

$$\begin{aligned} a &= \frac{\sin\psi_0}{2\cdot s\cdot\kappa} \\ b &= -\frac{\cos\psi_0}{2\cdot s\cdot\kappa} \end{aligned}$$

The routine also computes the error in the distance based on the erros in the circle parameters, XERR, ($\vec{x}_e = (\sigma^2\kappa, \sigma^2\psi_0, \sigma^2 c^2)$). The error in the distance is given as:

$$\sigma_d = \sqrt{\frac{(a\cdot\sigma_a)^2 + (b\cdot\sigma_b)^2}{\sqrt{(x-a)^2+(y-b)^2}} + \sigma_\rho^2}$$

where the errors in a, b and ρ are:

$$\begin{aligned} \sigma_a^2 &= (b\cdot\sigma_{\psi 0})^2 + (a\cdot\frac{\sigma_\kappa}{\kappa})^2 \\ \sigma_b^2 &= (a\cdot\sigma_{\psi 0})^2 + (b\cdot\frac{\sigma_\kappa}{\kappa})^2 \\ \sigma_\rho^2 &= (\frac{1}{2\cdot\rho})^2\cdot(1+(\frac{\sigma_\kappa}{2\cdot\kappa^3})^2). \end{aligned}$$

4.5.11 SUBROUTINE TCIFIX

Author: Curtis A. Meyer
Creation Date: 8 July, 1989
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK, TCLIFT, CBUNIT, TCPRMS, TCCUTS, TJPRMS and TCDEBG.
Subroutines Referenced: TCHECK.

This routine will perform a crossing angle dependent correction to the fit hit positions in the JDC. During pattern recognition, and the circle fits to the tracks, it is assumed that all tracks cross through the jet cells in a direction parallel to the wire planes. For tracks which have momentum less than infinity, this is rarely true. Instead, they cross through the cell at some angle, (α) away from the assumed direction. In order to correct the positions in the chamber based on the crossing angle, it has been assumed that the isochronus curves can be well described as arcs of circles. The true position in the chamber can then be computed as the point on the isochron–circle where the slope of the track and the slope of the circle are the same. It is this correction which is performed by this routine.

In order to correct the hit positions along a track, we start with the fit circle for the track, (TCFITR routine),

$$\frac{r_i}{2qR} + \frac{c^2}{2qRr_i} + \sin(\phi_i - \psi_0) = 0$$

where R, c^2 and ψ_0 are the three fit parameters. We note that as long as one rotates both ϕ and ψ_0 by the same amount, this equation is rotationally invariant. As such, it is simply expressed in *sector coordinates*, (see TCRESL routine). In sector coordinates, the angle at which a track crossed a jet cell is

$$\tan\alpha = \frac{dy}{dx}$$

and defining $\Phi_i = \phi_i - s_i$, where s_i is the angle at the center of the JDC sector, then

$$\frac{dy}{dx} = \frac{\sin\Phi + \cos\Phi\cdot r\cdot\frac{d\Phi}{dr}}{\cos\Phi - \sin\Phi\cdot r\cdot\frac{d\Phi}{dr}}$$

Given α, the center of the isochron circle (a,b) and the circle radius, r, the position in the chamber is given as

$$x = a - r\sin\alpha$$

and

$$y = b + r\cos\alpha.$$

The isochron parameters, a, b and r are obtained from the nominal fit position in sector coordinates by the following transformations. x_s and y_s are the x–y position in sector coordinates, and s_i is the stagger of the wire.

$$a = x_s$$

$$y_f = y_s - s_i$$

$$r = min(y_f, r_{max})$$

$$b = y - r$$

The isochron correction then *pushes* the y coordinate out to a larger value to compensate for the crossing angle.

$$y_c = y + r \cdot (\sec\alpha - 1)$$

In addition, there is an error term associated with this correction which is estimated at 7.5% of the correction term.

Once the positions have been computed, the errors in the r and ϕ coordinates are computed from two parameters, σ_{y0} and σ_d. These two parameters describe the resolution of the chamber as a function of drift time via the relation:

$$\sigma_y^2 = \sigma_{y0}^2 + t_d \cdot \sigma_d^2 + (0.075 \cdot r \cdot (\sec\alpha - 1))^2$$

The errors in r and ϕ are:

$$\sigma_r = y_s \cdot \sigma_y / r$$

$$\sigma_\phi = x_s \cdot \sigma_y / r^2$$

The parameters σ_{y0}^2 and σ_d^2 are SY02TJ and SYDFTJ in the /TJPRMS/ common block. During calibration, they can be set using the **SY02** and **SYDF** cards in CJINIT.

4.5.12 SUBROUTINE TCITER

Author: Curtis A. Meyer
Creation Date: 14 September, 1987
References: Datenanalyse by Siegmund Brandt, p.271.
Call Arguments: (ITRK, Y, GYINV, *X, *COVR, *CHISQ, *IERR).
Common Blocks Used: CBBANK, CBLINK, TCPRMS, TCCUTS and TCANGL. **Subroutines Referenced:** None.

This routine fits the track data as (r, ϕ) pairs with known errors to the equation of a circle

$$\kappa \cdot r_i + \frac{\kappa \cdot c^2}{r_i \cdot q \cdot R} + \sin(\phi_i - \psi_0) = 0$$

where the fit parameters are κ, ψ_0 and c^2. κ is defined to be $\frac{q \cdot B_s}{2 \cdot R}$ where q is the charge of the particle, B_s is the direction of the magnetic field, and R is the distance to the center of the circle from the origin. ψ_0 is the direction of the track at the point of closest approach to the origin, and c^2 is defined as $c^2 = R^2 - \rho^2$ where ρ is the radius of the circle. Note that c^2 can be negative.

An initial guess for these parameters is passed in the double precision array X(3), and the final values are returned in the same array. The covariance matrix from the fit is returned in the double precision array COVR(3,3). The track coordinates are passed as (r, ϕ) pairs in the double precision array Y(N2), while their errors are passed in the double precision array GYINV(N2). The routine also returns the χ^2 of the fit in the single precision variable CHISQ and an error code IERR. The error code identifies how well the iteration converged, and is given the meanings as follows.

- IERR=0 Normal convergence, no problems.
- IERR=1 Initial guess was very good, no iterations done.
- IERR=2 Not set in this routine.
- IERR=3 The χ^2 was too large for this track.
- IERR=4 The value of ξ^2 started to diverge after 3 iterations.
- IERR=5 The track failed to converge in NITRTC iterations.
- IERR=6 There were too few points to fit ($n < 3$).
- IERR=7 Matrix inversion of a singular matrix, don't trust the results of the covariance matrix.

The fit is done by first writing the (r, ϕ) data pairs in a vector $\vec{Y}$ as:

$$\vec{Y} = \begin{pmatrix} r_1 \\ \phi_1 \\ r_2 \\ \phi_2 \\ \vdots \\ r_n \\ \phi_n \end{pmatrix}.$$

The errors of these points are in a 2NPTS by 2NPTS matrix $\mathcal{G}_y^{-1}$ given as:

$$\mathcal{G}_y^{-1} = \begin{pmatrix} \delta r_1^2 & & & & & \\ & \delta\phi_1^2 & & & & \\ & & \ddots & & & \\ & & & \delta r_n^2 & & \\ & & & & \delta\phi_n^2 & \end{pmatrix}$$

We then express the three fit parameters as a vector $\vec{X}$,

$$\vec{X} = \begin{pmatrix} q \cdot B_s/R \\ \psi_0 \\ c^2 \end{pmatrix}.$$

With these, we form NPTS equations of constraint using the above equation for a circle.

$$f_i = X_1 \cdot Y_{2i-1} + \frac{X_1 \cdot X_3}{Y_{2i-1}} + \sin(Y_{2i} - X_2).$$

The NPTS by 3 matrix $\mathcal{A} = \partial f / \partial X$ is defined as:

$$\mathcal{A} = \begin{pmatrix} \frac{\partial f_1}{\partial X_1} & \frac{\partial f_1}{\partial X_2} & \frac{\partial f_1}{\partial X_3} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial X_1} & \frac{\partial f_n}{\partial X_2} & \frac{\partial f_n}{\partial X_3} \end{pmatrix}$$

The NPTS by 2NPTS matrix $\mathcal{B} = \partial f/\partial Y$ is defined as:

$$\mathcal{B} = \begin{pmatrix} \frac{\partial f_1}{\partial Y_1} & \cdots & \frac{\partial f_1}{\partial Y_{2n}} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial Y_1} & \cdots & \frac{\partial f_n}{\partial Y_{2n}} \end{pmatrix}$$

and the NPTS elements of the vector $\vec{C}$ are given as the values of f_i. We now define the matrix

$$\mathcal{G}_B = (\mathcal{B}\mathcal{G}_y^{-1}\mathcal{B}^T)^{-1}$$

and with this, we form the iterative sequence:

$$\vec{X}_{i+1} = \vec{X}_i - (\mathcal{A}^T\mathcal{G}_B\mathcal{A})^{-1}(\mathcal{A}^T\mathcal{G}_B\vec{C})$$

and the covariance matrix for these values of $\vec{X}$ is given as:

$$\mathcal{C}_X = (\mathcal{A}^T\mathcal{G}_B\mathcal{A})^{-1}.$$

4.5.13 SUBROUTINE TCLOAD

Author: Curtis A. Meyer
Creation Date: 19 April, 1988
References:
Call Arguments: (ITRK, IOPT, ITCTK, *X, COVR, *CHRG, *Y, *GYINV, CHISQ).
Common Blocks Used: /CBBANK/, /CBLINK/, /TCPRMS/, /TCCUTS/ /TCHITS/and /TCANGL/.
Subroutines Referenced: None.

This subroutine loads and stores all the hit coordinates, r and ϕ, for track number ITRK into the Y vector, and their errors into the vector GYINV. It also loads the sines and cosines of each point into the /TCANGL/ common block and makes sure that every hit in the track has been assigned the track number ITRK. Finally, the routine makes a guess for the circle parameters, and places that in the vector X. It also guesses what the charge is, places that in CHRG.

The passed value of IOPT determines which information is loaded according to the following rules.

0 Load the (r, ϕ) information and (σ_r, σ_ϕ) information from the TCHT data bank, then make a guess for the track parameters.

1 Load the (r, ϕ) and (σ_r, σ_ϕ) information, but do not make a guess.

2 Load the (r, ϕ) and (σ_r, σ_ϕ) information from this track to the end of the track already loaded. Do not make a guess.

3 Only make a guess using the previously loaded information

4 Store the fit information for X and its covariance matrix, COVR in the **TCTK** data bank.

5 Store the information for X in the **TCTK** data bank, and the fit information in Y and GYINV in the **TCHT** data banks.

The initial guess for the circle parameters are made by fitting three points on the track to a circle of the form

$$(x-a)^2 + (y-b)^2 = r^2.$$

The points chosen are the first hit, (x_1, y_1), the last hit, (x_3, y_3) and a hit in the middle of the track, (x_2, y_2). The parameters are then given as:

$$\begin{aligned} b &= \frac{\frac{x_2^2+y_2^2-x_1^2-y_1^2}{2(x_2-x_1)} - \frac{x_3^2+y_3^2-x_1^2-y_1^2}{2(x_3-x_1)}}{\frac{y_1-y_2}{x_1-x_2} - \frac{y_1-y_3}{x_1-x_3}} \\ a &= \frac{x_2^2 - y_2^2 - x_1^2 - y_1^2}{2(x_2 - x_1)} - b \cdot \frac{y_2 - y_1}{x_2 - x_1} \\ r^2 &= (x_1 - a)^2 + (y_1 - b)^2 \end{aligned}$$

The circle parameters, $X = (R, \psi_0, c^2)$ are then obtained as follows:

$$\begin{aligned} R &= q \cdot B_s / \sqrt{a^2 + b^2} \\ \psi_0 &= \tan^{-1}(-b/-a) - s \cdot \frac{\pi}{2} \\ c^2 &= R^2 - r^2 \\ q &= s \cdot b_s \end{aligned}$$

where q is the charge of the particle, and b_s is the direction of the magnetic field, ($+1$ along the z–axis and -1 against the z–axis). The value s is determined from the formula

$$\psi_0 = \beta_0 - s \cdot \frac{\pi}{2}.$$

The angle β_0 is defined as the angle relative to the center of the circle to the point of closest approach of the track, it can be obtained from the formula: $\beta_0 = atan2(-b, -a)$. The angle ψ_0 measures the direction of the track at that point.

forming the two values of ψ_0, $(\pm\frac{\pi}{2})$ and determining which one is closer to the ϕ angle of the first hit in the track. If the $+$ case is closer, the charge is positive while if the $-$ case is closer, the charge is negative. The routine then returns X and CHRG containing the 3 fit parameters and the charge respectively. If GUESS is .FALSE., then nothing is done to the passed values of X and CHRG.

In storing the values of X, COVR and CHISQ into the **TCTK** data bank for track ITRK. X is double precision and contains the three fit parameters, (R, ψ_0, c^2), while COVR is the double precision covariance matrix for these values, and the single precision variable CHISQ is the chi–square of the circle fit. This routine also computes the transverse momentum, and its error and puts it in the **TCTK** data bank.

$$p_\perp = e \cdot B \cdot r$$

where

$$r = \sqrt{R^2 - c^2}.$$

$$\sigma_p = \frac{e \cdot B}{r} \cdot \sqrt{\sigma_{RR} + \frac{1}{4}\sigma_{cc}}.$$

4.5.14 SUBROUTINE TCRSLV

Author: Curtis A. Meyer
Creation Date: 10 November, 1988
References:
Call Arguments:(ITRK, X, COVR, *N).
Common Blocks Used: CBBANK, CBLINK, TCHITS TJCONV and TCCUTS.
Subroutines Referenced: TCHECK, TCRESL and TCDROP.

This routine is called after the circle fit to a track has been made. It tries to resolve points for which the left–right ambiguity was unresolvable during pattern recognition. The routine takes the circle fit information for track ITRK given as the double precision variables X and COVR, (see TCFITR for a description), and checks each point which has been unresolved, (identified in the /TCHITS/ common block) to see if one of the two solutions could be sensibly added to this track. The check is made using the TCHECK routine, and if the point can be added, the TCRESL routine is called to resolve it. If the point can not be added, then the TCDROP routine is used to discard the point from the track.

The TCHECK routine returns three parameters to this routine, a χ_l χ_r and χ_{cut}. The routine then uses the RSLVTC parameter in the /TCCUTS/ common block to set a cut level by simply multiplying it by the returned cut off.

4.5.15 SUBROUTINE TCSPLT

Author: Curtis A. Meyer
Creation Date: 6 May, 1988
References:
Call Arguments: (ITRK, IOPT, DEVI, CHLIM, *ICODE).
Common Blocks Used: TCANGL and TCCUTS.
Called by: TCFITR and TCSWEP.
Subroutines Referenced: SM353.

This routine is called by TCFITR and TCSWEP to try and remove outlyers in the r–ϕ plane from track ITRK in the **TCTK** data bank. The definition of an outlyer is controlled by the passed arguments DEVI and CHLIM. Where DEVI is the outlyer distance in centimeters, and CHLIM is the contribution which the hit must make to the total χ^2 to be considered an outlyer. The control argument IOPT identifies if the outlyer is dropped, (IOPT=0), or the error in ϕ is blown up, (IOPT=1). The returned value of ICODE indicates how many points were either dropped or tagged.

The routine works by using data in the /TCHITS/ common block which are filled by the TCITER routine. As such this routine should only be called immediately after TCITER has been called.

TCSPLT uses the SM353 smoothing algorithem to establish a baseline deviation of all points from their measured values. It then looks at how far every point was moved from its baseline value. Those points were moved more than DEVI, and contribute more than CHLIM to the total χ^2 are then tagged for either dropping or error explosion.

The appropriate action is then taken on all bad points. However, if more than 4 points are tagged for dropping, no action is taken.

Figure 7: Two causes of bad track fits. (a) shows a track with one bad point, and (b) shows a track that will be split into two tracks.

4.5.16 SUBROUTINE TCSWEP

Author: Curtis A. Meyer
Creation Date: 6 May, 1988
References:
Call Arguments: (*IERR).
Common Blocks Used: TCANGL and TCCUTS.
Called by: TCCIRC.
Subroutines Referenced: TCAHIT, TCADD.

This routine looks through all fit tracks, and tries to add unused hits to them. It begins by looking for gaps in the existing track, and then tries to extend the track back to layer 1 in the JDC. Finally, an outward projection is made to either layer 23, or until a gap of five layers is found.

4.5.17 SUBROUTINE TCTHET

Author: Curtis A. Meyer
Creation Date: 26 October, 1987
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK, TCLIFT, TCPRMS, TCCUTS and TCANGL.
Subroutines Referenced: None.

This routine fits the tracks which have passed through TCASSC for opening angle, λ and z traceback. In doing this fit, at least 3 points are required, and at most 15 are used. The data are fitted to a line form:

$$z_i = a + b \cdot r_i,$$

using a weighted least squares method. From this fit, the tangent of the opening angle is given as:

$$\tan\lambda = b,$$

The values of a and b are obtained from a least squares fit, and are:

$$\begin{aligned}
a &= \frac{1}{\Delta}\cdot\left(\sum\frac{z_i}{\sigma_i^2}\cdot\sum\frac{r_i^2}{\sigma_i^2} - \sum\frac{r_i}{\sigma_i^2}\cdot\sum\frac{r_i\cdot z_i}{\sigma_i^2}\right)\\
b &= \frac{1}{\Delta}\cdot\left(\sum\frac{1}{\sigma_i^2}\cdot\sum\frac{z_i\cdot r_i}{\sigma_i^2} - \sum\frac{z_i}{\sigma_i^2}\cdot\sum\frac{r_i}{\sigma_i^2}\right)\\
\Delta &= \sum\frac{1}{\sigma_i^2}\cdot\sum\frac{r_i^2}{\sigma_i^2} - \left(\sum\frac{r_i}{\sigma_i^2}\right)^2\\
s^2 &= \frac{1}{N-2}\cdot\sum(z_i - ab\cdot r_i)^2
\end{aligned}$$

and the errors in the fit quantities a and b are given as:

$$\begin{aligned}
\sigma_a^2 &= \frac{1}{\Delta}\cdot s^2\cdot\sum\frac{r_i}{\sigma_i^2}\\
\sigma_b^2 &= \frac{1}{\Delta}\cdot s^2\cdot N.
\end{aligned}$$

This can then be expressed as a covariance matrix between $\tan\lambda$ and a as:

$$\begin{pmatrix} \sigma_{\lambda\lambda} & \sigma_{\lambda a}\\ \sigma_{\lambda a} & \sigma_{aa}\end{pmatrix} = \begin{pmatrix} \sigma_b^2 & 0\\ 0 & \sigma_a^2\end{pmatrix}$$

If somehow the value of λ were $\pm 90°$, the value of the error code in the **TCTK** data bank would be set to 8, and and a value of $\tan\lambda = \pm 10^8$ would be returned. If the routine becomes lost in stepping through the track, the error code is set to a value of 9.

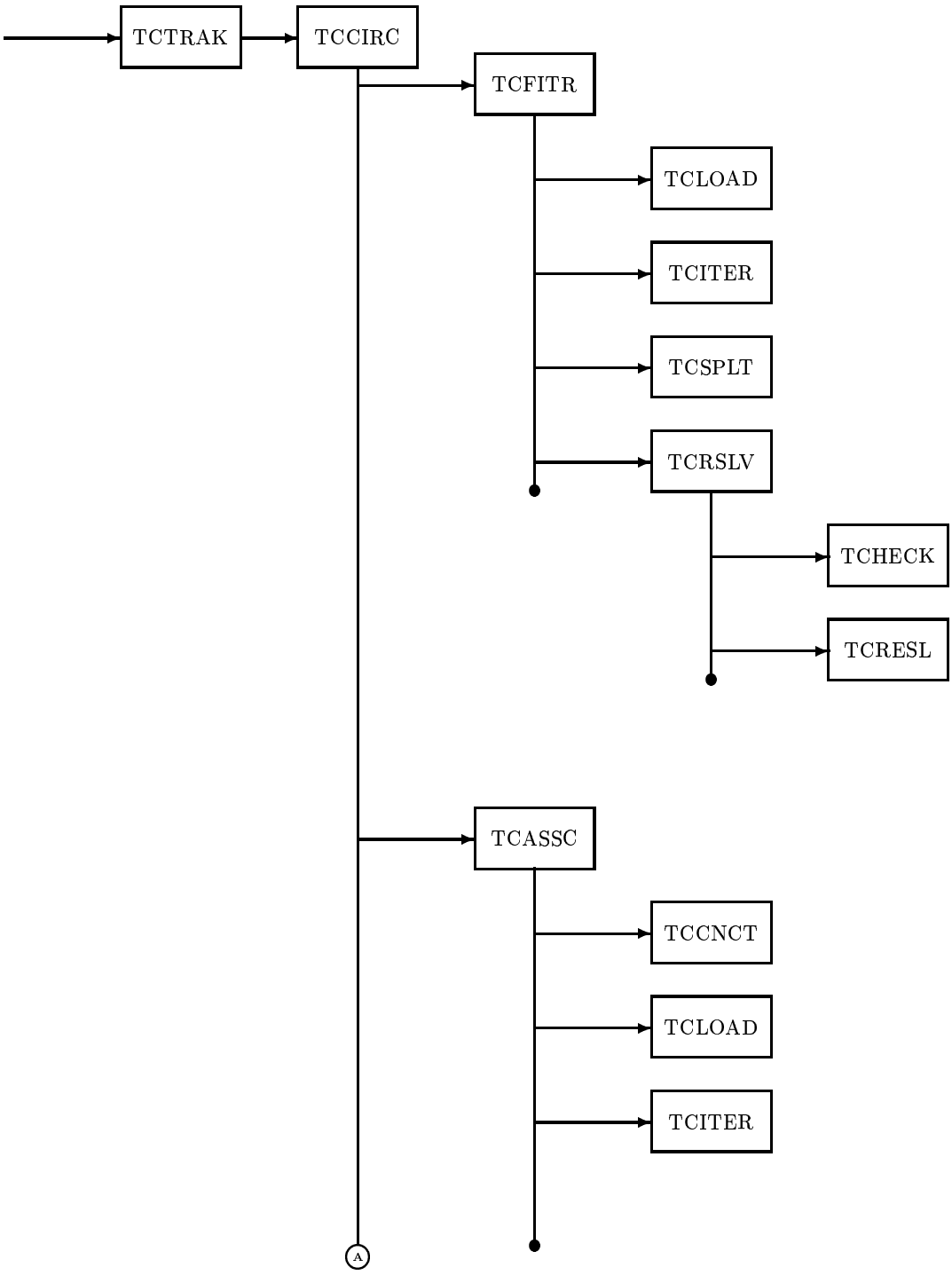


Figure 8: Flow of the chamber software, circle fitting section.

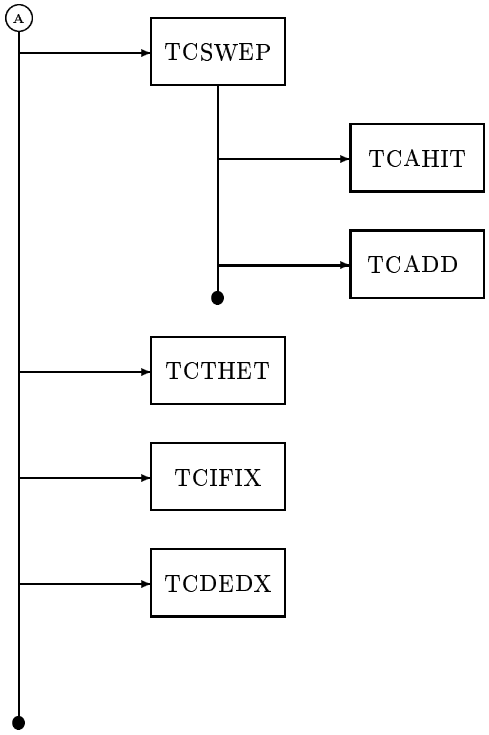


Figure 9: Flow of the chamber software, circle fitting section.

4.6 Helix Fitting Software

The following subroutines are found in the TC_HELX patch of the LOCATER card file.

4.6.1 SUBROUTINE TCHELX

Author: Curtis A. Meyer
Creation Date: 27 May, 1988
References: Datenanalyse by Siegmund Brandt, p.271.
CERN Program Library, routine F101, (MATIN2).
Call Arguments: (IOPT, NTRK).
Common Blocks Used: CBBANK, CBLINK, TCLIFT, TCPRMS, TCCUTS and TCANGL.
Subroutines Referenced: TCHXLD,
(CERN LIBRARY) DINV.

This routine loops over the NTRK tracks found, and fits the points in track ITRK with a helix of the form:

$$x_i = r_0 \cdot \sin\psi_0 + \frac{1}{\alpha} \cdot (\cos\beta_i + s \cdot \sin\psi_0) \quad (7)$$

$$y_i = -r_0 \cdot \cos\psi_0 + \frac{1}{\alpha} \cdot (\sin\beta_i - s \cdot \cos\psi_0) \quad (8)$$

$$z_i = z_0 - \frac{s \cdot \tan\lambda}{\alpha} \cdot (\beta_i - \psi_0 - s \cdot \frac{\pi}{2}) \quad (9)$$

where the angle β_i is the azimuthal angle of the line from the center of the helix to the point (x_i, y_i). The value of IOPT is used to tell the TCHXLD routine from which bank the coordinates are to be taken, and if they should be updated during the iteration.

- IOPT=0 The routine assumes that the **TCTR** banks do not exist, and creates them. It then takes the data from the **TCTK** bank and stores it in the TCTR bank. Also, the (x,y,z) coordinates are improved at each iteration, and the improved values are stored in the **TCTR** bank.
- IOPT=1 This is the same as option 0, except the improved coordinates are not stored at the end of the iteration.
- IOPT=2 The routines assumes that the **TCTR** bank does exist, and takes the track data from it. The improved coordinates are then stored in the bank upon completion.
- IOPT=3 This is the same as 2, except the improved coordinates are not stored upon completion.

The five parameters that describe the helix are $\vec{\xi} = (r_0, z_0, \alpha, \tan\lambda, \psi_0; s)$. r_0 is the distance of closest approach to the z–axis, and can be negative. The sign is chosen to make the helix fit stable when the charge of the particle is changed by the fit. z_0 is the value of z at the radius $\mid r_0 \mid$, α is the inverse of the radius of curvature of the helix, λ is the opening angle of the helix, ψ_0 is the angle from the circle fit and s is a sign parameter which is related to the particle charge through the direction of the magnetic field.

In order to fit these equations to the measured points $(x, y, z)_i$ with measurement errors $(\sigma_x, \sigma_y, \sigma_z)_i$, we solve for the parameter β_i by rewriting the first two equations as:

$$\cos\beta_i = \alpha \cdot x_i - \sin\psi_0 \, (\alpha \cdot r_0 + s) \quad (10)$$

$$\sin\beta_i = \alpha \cdot y_i + \cos\psi_0 \, (\alpha \cdot r_0 + s) \quad (11)$$

$$(12)$$

These can be solved to yield β_i:

$$\beta_i = \tan^{-1}(\sin\beta_i/\cos\beta_i) \tag{13}$$

$$\tag{14}$$

We now have two equations of constraint for every point i on the track:

$$f_{1i} = \frac{1}{\alpha}\left(\cos^2\beta_i + \sin^2\beta_i - 1\right) \tag{15}$$

$$f_{2i} = z_0 - z_i - \frac{s\cdot\tan\lambda}{\alpha}\left(\beta_i - \psi_0 - s\cdot\frac{\pi}{2}\right) \tag{16}$$

The first equation can also be written as:

$$f_{1i} = \alpha\left(x_i^2 + y_i^2 + r_0^2\right) + 2\cdot s\cdot r_0 + 2\cdot(\alpha\cdot r_0 + s)\left(y_i\cos\psi_0 - x_i\sin\psi_0\right)$$

In order to make an iterative fit to the unknown helix parameters, $\vec{\xi} = (r_0, z_0, \alpha, \tan\lambda, \beta_0)$, (see the description of the TC2HLX routine), we form a vector out of all the measured quantities,

$$\vec{\eta} = (x_1, y_1, z_1, x_2, y_2, \cdots, z_n),$$

and define a $3N$ by $3N$ diagonal error matrix, $\mathcal{G}_\eta$ as:

$$\mathcal{G}_\eta^{-1} = \begin{pmatrix} \delta x_1^2 & & & & & & \\ & \delta y_1^2 & & & & & \\ & & \delta z_i^2 & & & & \\ & & & \delta x_2^2 & & & \\ & & & & \delta y_2^2 & & \\ & & & & & \ddots & \\ & & & & & & \delta z_N^2 \end{pmatrix}$$

Now the $2N$ by 5 matrix $\mathcal{A}$ is formed such that:

$$\mathcal{A} = \begin{pmatrix} \frac{\partial f_1}{\partial\xi_1} & \cdots & \frac{\partial f_1}{\partial\xi_5} \\ \vdots & & \vdots \\ \frac{\partial f_{2N}}{\partial\xi_1} & \cdots & \frac{\partial f_{2N}}{\partial\xi_5} \end{pmatrix},$$

the $2N$ by $3N$ matrix $\mathcal{B}$ is defined to be:

$$\mathcal{B} = \begin{pmatrix} \frac{\partial f_1}{\partial\eta_1} & \cdots & \frac{\partial f_1}{\partial\eta_{3N}} \\ \vdots & & \vdots \\ \frac{\partial f_{2N}}{\partial\eta_1} & \cdots & \frac{\partial f_{2N}}{\partial\eta_{3N}} \end{pmatrix}$$

and the $2N$ long vector $\mathbf{C}$ as $C_i = f_i$. If we now define that $2N$ by $2N$ matrix $\mathcal{G}_\mathcal{B}$ as:

$$\mathcal{G}_B = (\mathcal{B}\mathcal{G}_\eta^{-1}\mathcal{B}^T)^{-1}$$

and then form the iterative sequence for $\vec{\xi}$:

$$\vec{\xi}_{[i+1]} = \vec{\xi}_{[i]} - (\mathcal{A}^T\mathcal{G}_B\mathcal{A})^{-1}(\mathcal{A}^T\mathcal{G}_B\vec{C}).$$

The measurements, $\vec{\eta}$ are improved using the formula

$$\delta\vec{\eta} = \mathcal{G}_\eta^{-1}\mathcal{B}^T\mathcal{G}_B(\vec{C} - \mathcal{A}\delta\vec{\xi}).$$

On each step of the iteration, the quantity, $\vec{\epsilon} = \vec{\eta}_0 - \vec{\eta}_i$,

$$\chi^2 = (\mathcal{B}\vec{\epsilon})^T \mathcal{G}_B (\mathcal{B}\vec{\epsilon})$$

is computed. If χ^2 falls below the cutoff, CHLXTC in the /TCCUTS/ common block, or the variation in χ^2 between iterations becomes smaller than CHLXTC, then the routine stops iterating. The routine also stops if the number of iterations exceeds NHLXTC in the /TCCUTS/ common block, or the value of χ^2 begins to increase. If the iteration does successfully converge, the covariance matrix for the 5 fit parameters is given as:

$$\mathcal{C}_\xi = (\mathcal{A}^T \mathcal{G}_B \mathcal{A})^{-1},$$

while the fit errors of the measurements are computed as

$$\mathcal{G}^{-1}_{\eta,\text{final}} = \mathcal{G}^{-1}_\eta - \mathcal{G}^{-1}_\eta \mathcal{B}^T \mathcal{G}_B \mathcal{B} \mathcal{G}^{-1}_\eta + \mathcal{G}^{-1}_\eta \mathcal{B}^T \mathcal{G}_B \mathcal{A} \mathcal{C}_\xi \mathcal{A}^T \mathcal{G}_B \mathcal{B} \mathcal{G}^{-1}_\eta.$$

(Actually, only the diagonal elements of this matrix are computed, as the correlations are not needed.)

This routine uses an initial guess for $\vec{\xi}$ derived from the results of the separate $r-\phi$ and $r-z$ fits, (see TCHXLD and TC2HLX routines). In most cases, this initial guess should be good enough that no improvement on the fit values will be made. In that case, only the covariance matrix will be determined by this routine. It is important that this covariance matrix be correct in order to do a proper vertex fit at later levels of the analysis.

This routine assigns an error code to each track fit. The meanings of these codes are as follows:

- **0** means that the code converged normally.
- **10** means that the routine was not implemented for this track. This can happen if there are exactly three points in the track, or if the opening angle λ is close to 0.
- **30** means the code converged with a value of χ^2 which was too large.
- **40** means the χ^2 started to diverge during iteration.
- **50** means the iteration limit was exceeded.
- **60** means that there were fewer than 3 points in this track.
- **70** means that an attempt was made to invert a singular matrix.

The value of this code is added to the code from the TCFITR and TCTHET routines (that code is between 0 and 9), to form an error code for the fit track. If the error code from this routine is 0, then the results of the fit are stored in the **TCTR** data bank using the TCHXLD routine. For all other values of the error code, the guess to the parameters made in TCHXLD by combining the TCFITR and TCTHET fits is retained, but the error code is updated.

4.6.2 SUBROUTINE TCHXLD

Author: Curtis A. Meyer
Creation Date: 27 May, 1987
References:
Call Arguments: (ITRK, ICODE, *NPTS, *X, *Y, *GYINV, *SIGN, *COVR, *IERR).
Common Blocks Used: CBBANK, CBLINK, TCLIFT, and ZBDIVS.
Subroutines Referenced: TC2HLX,
(ZEBRA LIBRARY) MZLIFT.

This routine transfers data between the **TCTR** data banks, and the working double–precision arrays X, Y, GYINV, SIGN and COVR. The direction of the transfer depends on the value of the passed parameter ICODE.

In the case of ICODE=0, the routine first lifts a **TCHX** subbank for track ITRK. Then the results from TCFITR and TCTHET are converted into helix coordinates using the TC2HLX subroutine, and stored in both the **TCHX** data bank. The TC2HLX routine has already stored the helix coordinates in X and COVR. Next, the routine goes into the **TCHT** data bank for this track, and loads the (x, y, z) and $(\sigma_x^2, \sigma_y^2, \sigma_z^2)$ values into the **TCTR** data bank. It also loads them into the Y and GYINV arrays. The error code from the **TCTK** bank is also copied into the **TCTR** data bank. If the number of points is less than 3, then this routine returns with IERR=60, and if the number of points is exactly 3, then the routine returns with IERR=10. Since version 1.50, the routine also looks through the **TPWC** cluster banks, and loads any PWC information which has been connected to the track.

In the case of ICODE=1, the routine assumes that the **TCHX** data bank for track ITRK exists, and copies the information out of that bank into the X, Y, GYINV and COVR arrays.

In the case of ICODE=2, the routine stores the values in X, Y, GYINV and COVR in the **TCHX** data bank for track ITRK.

In the case of ICODE=3, the routine stores the values contained in X and COVR in the **TCHX** data bank for track ITRK.

In the case of ICODE=4, the routine updates the error code in the **TCTR** bank using the value of IERR, but does not store the passed data.

4.6.3 SUBROUTINE TCMSCT

Author: Curtis A. Meyer
Creation Date: 31 August, 1988
References: R.L. Gluckstern, **NIM 24** 381, (1963).
Call Arguments: None.
Common Blocks Used: TCSCAT, TCPRMS, CBBANK and CBLINK.
Subroutines Referenced: None.

This routine corrects the covariance matrices obtained in the helix fit for multiple scattering in both the JDC itself, and all the material from the interaction point to the JDC. Within the JDC, the multiple scattering only from the chamber gas has been included. Multiple scattering off the sense and guard wires has not been included at this point, but could be roughly added by changing the radiation length of the chamber gas in the /TCSCAT/ common block.

This routine loops over all the tracks in the **TCTR** data bank, and computes the multiple scattering contributions as follows. For the scattering in the JDC gas, the following formula's from Gluckstern are employed.

$$\begin{aligned}
\sigma_{\alpha\alpha} &= \Lambda^2 \cdot \frac{C_n}{L} \cdot \sec^4\lambda \\
\sigma_{\alpha\beta} &= \Lambda^2 \cdot D_n \cdot \sec^3\lambda \\
\sigma_{\tan\lambda\tan\lambda} &= \Lambda^2 \cdot F_n \cdot L \cdot \sec^4\lambda \\
\sigma_{\beta\beta} &= \Lambda^2 \cdot E_n \cdot L \cdot \sec^2\lambda
\end{aligned}$$

where

$$\Lambda^2 = \left(\frac{14.1 MeV/c}{p \cdot v/c}\right)^2 \cdot \frac{1}{x_r}$$

and the constants are given as:

$$C_n = 1.43, D_n = 0.21, E_n = 0.23, F_n = 0.20.$$

and x_r is the radiation length of the gas in the JDC.

For multiple scattering in the material traversed by the particle before entering the JDC, the following apply:

$$\begin{aligned}
\sigma_{rr} &= \Gamma^2 \cdot \sec^2\lambda \cdot \sum \frac{t_i \cdot r_i^2}{x_i} \\
\sigma_{zz} &= \Gamma^2 \cdot \sec^4\lambda \cdot \sum \frac{t_i \cdot r_i^2}{x_i} \\
\sigma_{\tan\lambda\tan\lambda} &= \Gamma^2 \cdot \sec^4\lambda \sum \frac{t_i}{x_i} \\
\sigma_{\beta\beta} &= \Gamma^2 \cdot \sec^4\lambda \sum \frac{t_i}{x_i} \\
\sigma_{\tan\lambda z} &= -\Gamma^2 \cdot \sec^4\lambda \sum \frac{t_i \cdot r_i}{x_i} \\
\sigma_{r\beta} &= \Gamma^2 \cdot \sec^2\lambda \cdot \sum \frac{t_i \cdot r_i}{x_i}
\end{aligned}$$

where

$$\Gamma^2 = \left(\frac{14.1 MeV/c}{p \cdot v/c}\right)^2$$

and the sums run over discrete scatterers of thickness t_i at radius r_i with radiation length x_i. All of the above corrections are added directly to the covariance matricies as stored in the **TCTR** data banks.

4.6.4 SUBROUTINE TC2HLX

Author: Curtis A. Meyer
Creation Date: 16 June, 1988.
References:
Call Arguments: (ITRK, *XHLX, *CVHLX).
Common Blocks Used: CBBANK, CBLINK.
Subroutines Referenced: None.

This routine takes the output from the TCFITR and TCTHET routines, and transforms it into helix coordinates. The initial data is given as:

$$\begin{aligned}
\vec{\xi}_{circ} &= (1/2R, \psi_0, c^2; s) \\
\vec{\xi}_{rz} &= (\tan\lambda, a_0) \\
\vec{x} &= (1/2R, \psi_0, c^2, \tan\lambda, a_0; s) \\
s &= sign(1, 1/2R)
\end{aligned}$$

along with a 3 by 3 and a 2 by 2 covariance matrices, which can be combined into a 5 by 5 matrix, C_x. The parameter s measures the direction of the track, and is related to the charge q and the magnetic field direction, b_s. The helix coordinates are defined as:

$$\vec{\xi}_{helix} = (r_0, z_0, \alpha, \tan\lambda, \psi_0; s)$$

The transformation is given by defining the radius of the fit circle as

$$\rho = \sqrt{R^2 - c^2}$$

Then we obtain that:

$$\begin{aligned} r_0 &= s \cdot \rho - R \\ z_0 &= a \\ \alpha &= \frac{1}{\rho} \\ \tan\lambda &= \tan\lambda \\ \psi_0 &= \psi_0 \\ q &= +s \cdot b_s \end{aligned}$$

One should note that it is possible for r_0 to be less than zero. The helix parameters are chosen in such a way that if the charge of the particle changes, then the parameter α will simply change signs. All other parameters will be unaffected. This makes it possible for the helix fit to correct wrong charge assignments performed in previous routines, and for the vertex fit to later correct the charges found in the helix fits.

4.6.5 SUBROUTINE TPCNCT

Author: Bruce Barnett
Creation Date: July 1991.
References:
Call Arguments:
Common Blocks Used: CBBANK, CBLINK and TCPRMS.
Called by: TCTRAK .
Subroutines Referenced: None.

This routine takes as input the **TPWC** hit banks, (at down links -1 and -2). It then performs a cluster search, and creates the **TPWC** cluster banks at down links -3 and -4. A loop is then made over all tracks in the **TCTK** banks, and connections are attempted to all clusters in both PWC's. The best connections are then identified by stamping the **TCTK** track number into word +1 of the cluster banks. These clusters are not physically used in the circle fit, but are picked up by the helix fit before the fit is made.

It is possible to call this routine from an external USER routine without ill effects. This may be useful in the case of all neutral data where the cluster banks are useful.

4.7 Vertex Fitting Software

The following subroutines are found in the TC_VERT patch of the LOCATER card file.

4.7.1 SUBROUTINE TCVERT

Author: Curtis A. Meyer
Creation Date: 22 July, 1988
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK and TCLIFT.
Subroutines Referenced:TCVRTX.
(ZEBRA LIBRARY) MZLIFT.

This routine looks through all the tracks from the helix fit, and divides them into two classes. The first is all tracks whose first JDC layer is less than LYVXTC in the /TCCUTS/ common block, (the default value is 6, but this can be changed using the **LYVX** card) and which have a nominal z–traceback withing ZVTXTC of ZOFFTC in the /TCCUTS/ and /TCPRMS/ common blocks. These tracks are then passed to the TCVRTX routine where they are fit to a common vertex.

4.7.2 SUBROUTINE TCVRTX

Author: Curtis A. Meyer
Creation Date: 2 February, 1990
References:
Call Arguments: (NTRKS, NTRAK, IVRTX, *NDROP, *IDROP, *IERR, XGUES).
Common Blocks Used: CBBANK, CBLINK and TCCUTS.
Subroutines Referenced: CERNLIB: DINV.

This routine looks at the NTRKS tracks whose track numbers are in the NTRAK array and tries to fit them to a common vertex. The returned value of X is the (x, y, z) coordinates of this vertex, COVR is a 3 by 3 covariance matrix for the vertex, CHSQR is the χ^2 of the fit, and IERR is an error code from the fit. GUES is a guess made by the user to the vertex. The meanings of IERR are given as follows.

000 No error, normal convergence o a good vertex.

100 Only one track was fit to this vertex.

200

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

400 The routine began to diverge.

500 The iteration limit was exceeded before convergence.

600

700 Inversion of a singular matrix was attempted.

The philosophy behind this routine is to attempt to get an rough vertex position which can be passed to the Crystal routines. The routine does not try to correlate particles with one another, (*i.e.* π^+ and π^- into a K_s), it instead assumes that there is exactly one vertex in the given volume, and fits to it. As one may later want to assume that there was more than one vertex in the volume, the routine does not vary the helix parameters nor their covariance matricies. It only looks for the best vertex given the constraints of the fit parameters. The fit then works as follows.

Given n tracks from the helix fit with fit parameters $\vec{x}_i$ and covariance matricies C_x^i, one forms a vector $\vec{\eta}$ such that

$$\vec{\eta} = (\vec{x}_1, \cdots, \vec{x}_n),$$

with

$$\vec{x}_i = (r_i, z_i, \alpha_i, \tan\lambda_i, \psi_{0_i}; s_i)$$

and a 5 by 5 covariance matrix C_η, such that:

$$C_\eta = \begin{pmatrix} C_x^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & C_x^n \end{pmatrix}$$

The routine uses this information to obtain the vertex,

$$\vec{\xi} = (x_v, y_v, z_v)$$

and its 3 by 3 covariance matrix C_ξ.

The mathematics begin with the $3 \cdot n$ helix equations for each track.

$$\begin{aligned} f_{[i,1]} &= x_v - r_i \cdot \cos\psi_{0_i} - \frac{1}{\alpha_i} \cdot (\cos\beta_i - \cos\psi_{0_i}) \\ f_{[i,2]} &= y_v + r_i \cdot \sin\psi_{0_i} - \frac{1}{\alpha_i} \cdot (\sin\beta_i - \sin\psi_{0_i}) \\ f_{[i,3]} &= z_i - z_v - \frac{s_i \cdot \tan\lambda_i}{\alpha_i} \cdot \left(\beta_i - \psi_{0_i} - s_i \cdot \frac{\pi}{2}\right) \end{aligned}$$

where the index i runs over the n tracks. The first two of these equations are then solved to yield two new equations.

$$\begin{aligned} \cos\beta_i &= \alpha_i \cdot x_v - \sin\psi_{0_i}\,(\alpha_i \cdot r_i + s_i) \\ \sin\beta_i &= \alpha_i \cdot y_v + \cos\psi_{0_i}\,(\alpha_i \cdot r_i + s_i) \end{aligned}$$

These can then be combined to yield an expression for β_i, and an equation of constraint:

$$\begin{aligned} \beta_i &= ATAN2(\sin\beta_i, \cos\beta_i) \\ \frac{1}{\alpha} &= \frac{1}{\alpha}\left(\sin^2\beta_i + \cos^2\beta_i\right) \end{aligned}$$

This will then yield $2 \cdot n$ equations of constraint given as:

$$\begin{aligned} f_{[1,i]} &= \frac{1}{\alpha_i}\left(\cos^2\beta_i + \sin^2\beta_i - 1\right) \\ f_{[2,i]} &= z_i - z_v - \frac{s_i \cdot \tan\lambda_i}{\alpha_i} \cdot (\beta_i - \psi_{0_i} - s_i \cdot \frac{\pi}{2}) \end{aligned}$$

An alternate form for the first equation is given as:

$$f_{[1,i]} = \alpha_i\left(x_v^2 + y_v^2 + r_i^2\right) + 2 \cdot s_i r_i + 2 \cdot (\alpha_i \cdot r_i + s_i) \cdot (y_v \cdot \cos\psi_{0_i} - x_v \cdot \sin\psi_{0_i})$$

A vector $\vec{C}$ is then defined as the $2n$ f_i's. As has been done in the TCITER and TCHELX routines, matricies $\mathcal{A}$ and $\mathcal{B}$ are defined via:

$$\begin{aligned} A_{ij} &= \frac{\partial f_i}{\partial \xi_j} \\ B_{ij} &= \frac{\partial f_i}{\partial \eta_j} \end{aligned}$$

and a matrix $\mathcal{G}_\mathcal{B}$ is now defined as:

$$\mathcal{G}_B = (\mathcal{B}\mathcal{C}_\eta\mathcal{B}^T)^{-1}.$$

With this, an iterative sequence for $\vec{\xi}$ is formed,

$$\vec{\xi}_{i+1} = \vec{\xi}_i - (\mathcal{A}^T\mathcal{G}_B\mathcal{A})^{-1}(\mathcal{A}^T\mathcal{G}_B\vec{C})$$

with the covariance matrix at each iteration given as

$$\mathcal{C}_\xi = (\mathcal{A}^T\mathcal{G}_B\mathcal{A})^{-1}$$

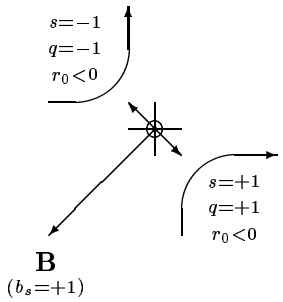


Figure 10: The distance of closest approach to the origin of a fit circle. If $r_0 < 0$, then the track's point of closest approach is between the center of the circle and the origin.

and $\vec{\epsilon} = \vec{\eta}_i - \vec{\eta}_0$

$$\chi^2 = (\mathcal{B}\vec{\epsilon})^T \mathcal{G}_B (\mathcal{B}\vec{\epsilon})$$

The covariance matrix, $\mathcal{C}_\xi$ can be obtained from the Jacobian of the above transformation, $\mathcal{T}_{ij} = \partial \xi_i / \partial x_j$.

$$\mathcal{C}_\xi = \mathcal{T} \mathcal{C}_x \mathcal{T}^T.$$

The matrix $\mathcal{T}$ can be explicitly expressed as:

$$\mathcal{T} = \begin{pmatrix} -\alpha \cdot r_0 & 0 & -\alpha & 0 & 0 \\ |\, \alpha \cdot r_0 \,| \cdot \tan\lambda & 0 & \alpha \cdot \tan\lambda & |\, r_0 \,| & 1 \\ -R \cdot \alpha^3 & 0 & \frac{1}{2}\alpha^3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

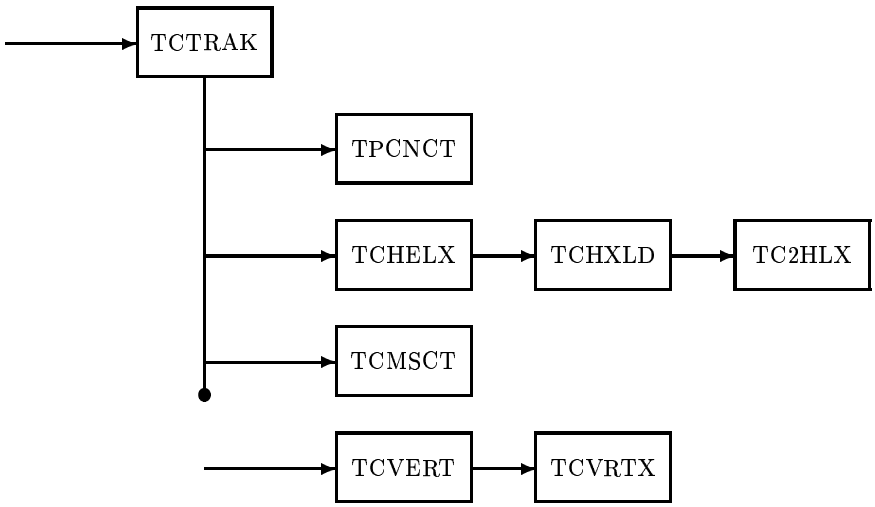


Figure 11: Flow of the chamber software; helix and vertex fitting

5 Chamber Calibration Software

Calibration of the JDC involves several different procedures, all of which need to be iterated to yield a final set of calibration constants. These can be divided into two classed of calibration, the first to determine the physical position of the JDC and PWC relative to the target and crystals, and the second to determine actual calibration constants of the JDC. With regard to these constants, there are essentially three types. Conversion of drift time into $r - \phi$ position in the JDC, conversion of amplitude difference into z position, and conversion of amplitude sums into dE/dx. This section details the software available for performing these procedures.

During the calibration, it is necessary to have direct access to most of the tracking cuts defined in chapter 2. To avoid having to recompile and relink the code for every change, a special calibration card file which is read in on logical unit LJCAL in the /CBUNIT/ common block needs to be provided. This file uses the FFREAD program to input new values for most of the tracking parameters and cutoffs, (see the CJINIT routine for a description of all available cards). Included in this are two cards for steering the calibration of the chamber. The first allows the user to turn on and off calibration, (the **JDCL** card), and the second identifies the calibration procedure to be performed, (the **ICAL** card).

The purpose of the **JDCL** card is two–fold. First one may want to check analysis results after a calibration step has been performed, which then means one does not have to recompile and relink the code. The second is for debugging of the code. In this light, two additional cards, (the **DEBG** nd **NDBG** cards) are provided for turning on and off debug output. If the user has built the locater code using both the debug patches and the calibration patches, then the **DEBG** flag will turn on and off the debugging of events. Secondly, if debug output is desired for only one event, then that event can be specified with the **NDBG** card. The parameters steered by these cards are available in the /TCDEBG/ common block as described in the next section.

5.1 Description of the Chamber Calibration Common Blocks

5.1.1 CJCUTS

The /CJCUTS/ common block contains cuts applied to the calibration code. All values in this common block can be changed using the CJINIT subroutine.

```
      DOUBLE PRECISION CXGZCJ
      LOGICAL          LSECCJ(30)
      INTEGER          MHITCJ,MTRKCJ,MCIRCJ
      REAL             CSMXCJ,DYMXCJ
      REAL             ZCUTCJ,ZPOSCJ,ZRINCJ,ZROTCJ,ZAMPCJ,ZPRBCJ
      REAL             ZCTTCJ,ZXYRCJ,ZMOVCJ,CFRACJ,CLIMCJ
      COMMON /CJCUTS/  CXGZCJ,LSECCJ,MHITCJ,MTRKCJ,MCIRCJ,
     &                 CSMXCJ,DYMXCJ,
     &                 ZCUTCJ,ZPOSCJ,ZRINCJ,ZROTCJ,ZAMPCJ,ZPRBCJ,
     &                 ZCTTCJ,ZXYRCJ,ZMOVCJ,CFRACJ,CLIMCJ
```

- CXGZCJ is the convergence criteria used in the CJGZFT routine. It has a default value of 0.01, but can be changed using the **CXGZ** card.
- LSECCJ is a logical array identifying which sectors are to be calibrated in the CJGZFT subroutine. Its default is all sectors on, however this can be changed using the **CJGZ** card.
- MHITCJ is the minimum number of hits per track to use the track in the CJGZFT and CJZFIT routines. It has a nominal value of 5, but can be modified with the **MHIT** card.

- MTRKCJ is the minimum number of tracks per event in the CJGZFT routine. It is set to 2, but can be changed using the **MTRK** card.
- MCIRCJ is the minimum number of points per track to fit a track using the CJITER routine. It has a default value of 15, but can be changed using the **MCIR** card.
- CSMXCJ is the minimum value of $\cos\alpha$ to write out a hit in CJUPDT. This has a default value of cos(25), but can be changed using the **CSMX** card.
- DYMXCJ is the maximum deviation of a hit to be written out by the CJUPDT routine. This has a default value of 0.05cm, but can be changed using the **DYMX** card.
- ZCUTCJ is an acceptance window around ZOFFTC in the z calibration of the JDC. Events that do not trace back to this window will not be accepted. Its value can be set using the **ZCUT** card. This has a default value of 10 cm.
- ZPOSCJ is used in the z calibration of the JDC. It defines the maximum range in z position on the sense wires to be used in the calibration. Its value can be set using the **ZPOS** card. This has a default value of 19 cm.
- ZRINCJ is used in the z calibration of the JDC by CJGZFT. It defines the minimum radius in x and y an event can have to be used. The default value is 0.00 cm, but can be changed using the **ZRIN** card.
- ZROTCJ is used in the z calibration of the JDC by CJGZFT. It defines the maximum radius in x and y an event can have to be used. The default value is 5.00 cm, but can be changed using the **ZROT** card.
- ZAMPCJ is used in z calibration of the JDC by CJGZFT to decide if data on a particular wire should be written to the calibration file. In order for a hit to be written out, the amplitudes on both the left and right end of the wire must be larger than ZAMPCJ, which has a default value of 200. This can be changed using the **ZAMP** card.
- ZPRBCJ is used in z calibration of the JDC by CJGZFT to decide if a fit event should be accepted. The probability of the event must be larger than ZPRBCJ, which has a default value of 0.00. This can be changed using the **ZPRB** card.
- ZCTTCJ is used in z calibration of the JDC by CJGZFT to decide if a fit event should be accepted. The fit z vertex of the event must be within ZCTTCJ cm of ZOFFTC, where the nominal value is 5.00 cm. This can be changed using the **ZCTT** card.
- ZXYRCJ is used in z calibration of the JDC by CJGZFT to decide if an event should be kept. All tracks in the event must be closer than ZXYRCJ cm from the fit vertex in x and y. The nominal value is 0.50cm, but can be changed using the **ZXYR** card.
- ZMOVCJ is the amount a point can be shifted in the CJGZFT routine, and still be used for calibration. This has a default value of 4 cm, but can be changed using the **ZMOV** card.
- CFRACJ is used by CJFITR to discard outliers from tracks. If a track whose chisquare is larger than CLIMCJ has a point which contributes CFRACJ of chisquare, then that point will be dropped before fitting. This has a default value of 0.900, but can be changed using the **CFRA** data card.
- CLIMCJ See the previous entry for the description. The default value is 150., and can be changed using the **CLIM** card.

5.1.2 CJEXFT

The /CJEXFT/ common block is used in dE/dx calibrations. No further documentation is available.

```
REAL            ENERCJ(30,23),EMINCJ,TNERCJ(30,23)
INTEGER         NENRCJ(30,23),NMINCJ
COMMON /CJEXFT/ ENERCJ,TNERCJ,NENRCJ,NMINCJ,EMINCJ
```

- ENERCJ
- EMINCJ
- TNERCJ
- NENRCJ
- NMINCJ

5.1.3 CJFLAG

The /CJFLAG/ common block contains flags a control switches for steering the calibration of the JDC. The values of these parameters can be controlled using the CJINIT routine.

```
LOGICAL         JDCLCJ,LREFCJ
INTEGER         ICALCJ
COMMON /CJFLAG/ JDCLCJ,LREFCJ,ICALCJ
```

- JDCLCJ This logical flag controls if calibration of the JDC is performed. If the calibration code has been selected using PATCHY, then this value defaults to .TRUE.. The existence of this parameter allows one to turn off the calibration code if it has been included, which can be done using the **JDCL** card.
- LREFCJ Inhibits the updating of constants when slow control values change.
- ICALCJ This parameter identifies the type of calibration to be performed. It defaults to a value of 0, but can be set to other values using the **ICAL** card.

  0 Calibration of the JDC using pion tracks which cross sector boundaries in the JDC.

  1 Calibration of the JDC using events $p\overline{p} \to \pi^+\pi^-$ and $p\overline{p} \to K^+K^-$.

  2 z–fit on a track–by–track basis for the gains of the preamps.

  3 Energy gains in the JDC.

  4 Global z–fit for relative gains of the preamps.

5.1.4 CJGAIN

The /CJGAIN/ common block is used as a storage area when calibrating the gains of the preamps on the JDC.

```
INTEGER         IFITCJ(690)
REAL            ZFITCJ(690),EFITCJ(690),SMSQCJ(690)
REAL            TIMXCJ(23,2)
COMMON /CJGAIN/ IFITCJ,ZFITCJ,EFITCJ,SMSQCJ,TIMXCJ
```

5.1.5 CJGLOZ

The /CJGLOZ/ common block is used by the CJGZFT routine to pass information back to the USER routine. It contains information on the hits which were fit. Not all of the possible information will be available in all applications. It is necessary to consult the code to determine what information is available.

```
      INTEGER    NTRKCJ,ITRKCJ(10),NPTSCJ(10)
      INTEGER    JTCHCJ(50,10),JLYRCJ(50,10),JSECCJ(50,10),JRESCJ(50,10)
      REAL       XPFTCJ(50,10),DXFTCJ(50,10),SXFTCJ(50,10),GXFTCJ(50,10)
      REAL       YPFTCJ(50,10),DYFTCJ(50,10),SYFTCJ(50,10),GYFTCJ(50,10)
      REAL       ZPFTCJ(50,10),DZFTCJ(50,10),SZFTCJ(50,10),GZFTCJ(50,10)
      REAL       TDRFCJ(50,10),ALFTCJ(50,10),ARGTCJ(50,10)
      COMMON /CJGLOZ/ NTRKCJ,ITRKCJ,NPTSCJ
     &               ,JTCHCJ,JLYRCJ,JSECCJ,JRESCJ
     &               ,XPFTCJ,DXFTCJ,SXFTCJ,GXFTCJ
     &               ,YPFTCJ,DYFTCJ,SYFTCJ,GYFTCJ
     &               ,ZPFTCJ,DZFTCJ,SZFTCJ,GZFTCJ
     &               ,TDRFCJ,ALFTCJ,ARGTCJ
```

- NTRKCJ is the number of tracks used in the global fit.
- ITRKCJ is a list of the tracks stored in this common.
- NPTSCJ is the number of points in each of up to 10 tracks.
- JTCHCJ is an array of pointers to the TCHT banks for the hits.
- JLYRCJ is the layer number of the hits.
- JSECCJ is the sector number of the hits.
- JRESCJ is the left–right resolution of each hit.
- XPFTCJ is the fit x or r position in centimeters for each hit. The value depends on which calibration routines are used.
- DXFTCJ is the difference between the fit and computed x position in centimeters.
- SXFTCJ is the fit error in the x position, σ_x^2.
- GXFTCJ is the origional error in the x–position, σ_x^2.
- YPFTCJ is the fit y or ϕ position, for each hit.
- DYFTCJ is the difference between the fit and computed y position.
- SYFTCJ is the fit error in the y position, σ_y^2.
- GYFTCJ is the origional error in the y–position, σ_y^2.
- ZPFTCJ is the fit z position, in centimeters for each hit.
- DZFTCJ is the difference between the fit and computed z position in centimeters.
- SZFTCJ is the fit error in the z position, σ_z^2.
- GZFTCJ is the origional error in the z–position, σ_z^2.

- TDRFCJ is the drift time of every hit.
- ALFTCJ is the left or $+z$ amplitude of every hit.
- ARGTCJ is the right or $-z$ amplitude of each hit.

5.1.6 CJPCAL

The common block is filled by the CJ4PRG subroutine. The data is from four–prong events that are passed through a 4–C kinematic fit.

```
      REAL            PINICJ(3,4),PFITCJ(3,4),DELPCJ(3,4),CHSQCJ(4)
      REAL            CVINCJ(3,3,4),CVFTCJ(3,3,4),CHRGCJ(4),MASSCJ(4)
      COMMON /CJPCAL/ PINICJ,PFITCJ,DELPCJ,CHSQCJ,CVINCJ,CVFTCJ,
     &                CHRGCJ,MASSCJ
      SAVE   /CJPCAL/
```

- PINICJ contains the initial values of p_x, p_y and p_z for each of the four tracks.
- PFITCJ contains the fit values of p_x, p_y and p_z for each of the four tracks.
- DELPCJ contains the change in p_x, p_y and p_z.
- CHSQCJ contains the contribution to χ^2 from each track.
- CVINCJ contains the initial 3 by 3 covariance matrix for each track.
- CVFTCJ contains the fit 3 by 3 covariance matrix for each track.
- CHRGCJ contains the charge of each track.
- MASSCJ contains the mass of each particle.

5.1.7 CJSTAT

```
      INTEGER         ISTACJ(100),IESTCJ(100),IUPDCJ(23,2)
      COMMON /CJSTAT/ ISTACJ,IESTCJ,IUPDCJ
```

5.1.8 RJSTAT

The /RJSTAT/ common block is included in the routines for processing the **RJDF** data when the calibration code is installed. It is filled on an event by event basis, and is then available to the user for monitoring the raw pulse information in the JDC.

```
      INTEGER         NHITRJ,ILENRJ(200),IMAXRJ(200,2)
      COMMON /RJSTAT/ NHITRJ,ILENRJ,IMAXRJ
```

- NHITRJ is the number of hits stored in the **RJDF** bank for the present event.
- ILENRJ is the number of FADC channels needed for each hit.
- IMAXRJ contains the maximum pulse height on the left and right side for each hit in the **RJDF** bank.

5.1.9 TCDEBG

The /TCDEBG/ common block is used to steer the debug print lines when they have been installed.

```
LOGICAL         DBGGTC(10),DEBGTC
INTEGER         NDBGTC
COMMON /TCDEBG/ DBGGTC,DEBGTC,NDBGTC
```

- DBGGTC identifies which section of the code should be debugged. The meaning of each of the ten elments is given below.
  - – Overall debug switch for the program. Must be .TRUE. to print out any debug lines.
  - 1 Print debug lines during processing of raw data.
  - 2 Print debug lines during pattern recognition.
  - 3 Print debug lines during circle fitting.
  - 4 Print debug lines during helix fitting.
  - 5 Print debug lines during vertex fitting.
  - 6 Print debug lines during calibration.
  - 7 Not yet used.
  - 8 Not yet used.
  - 9 Not yet used.
  - 10 Print debug lines during slow control processing.
- DEBGTC When set to .TRUE., then all installed debug print lines are enabled. The value of this variable is steered in the TCTRAK routine depending on the values of DBGGTC.
- NDBGTC When given a non–zero value, the TCTRAK routine will set DEBGTC .TRUE. only for event number NDBGTC. For all other events, it will print a one–line message that the event has been processed, but the value if DEBGTC will be set to false.

5.2 Description of the Chamber Calibration Software

The following subroutines are found in the JDCALIBR patch in the LOCATER card file.

5.2.1 SUBROUTINE CJCALB

Author: Curtis A. Meyer
Creation Date: 20 September, 1988.
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK, TCLIFT and CJFLAG.
Subroutines Referenced: TCPATT, TCCIRC, TCHELX, CJFITR, CJZFIT, CJGZFT, CJ4PRG and CJDEDX
ZEBRA LIBRARY: MZLIFT

This routine has the same function as the TCTRAK routine does for normal tracking. It is called by TCTRAK to control the program flow when calibrating the JDC. The type of calibration performed is controlled by the value of ICALCJ in the /CJFLAG/ common block. The present options are as follows.

-1 Calibration check using CJFITR and all tracks.

0 Calibrate the JDC using tracks which cross sector boundaries in the JDC.

1 Calibrate the JDC using CJ4PRG, 4C–Kinematic fit.

2 Calibrate the JDC z constants using the CJZFIT routine.

3 Calibrate the JDC dE/dx using CJDEDX.

4 Calibrate the JDC z constants using the CJGZFT routine.

5.2.2 SUBROUTINE CJDCAL

Author: Klaus Peters
Creation Date: 27 November, 1989.
References:
Call Arguments: None.
Common Blocks Used: CBBANK, CBLINK, CBSTOP and CJEXFT.
Called by: CJDEDX .
Subroutines Referenced: None.

No documentation available.

5.2.3 SUBROUTINE CJDEDX

Author: Klaus Peters
Creation Date: 20 December, 1989
References:
Call Arguments: None.
Common Blocks Used: /CBLINK/ and /CJEXFT/.
Called by: CJCALB.
Subroutines Referenced: TCOORD, TCRHIT and CJDCAL.
CERNLIB: FLPSOR.

No documentation available.

5.2.4 SUBROUTINE CJFITR

Author: Curtis A. Meyer
Creation Date: 20 September, 1988.
References:
Call Arguments: None.
Common Blocks Used:CBLINK, CBBANK, CBUNIT, TCHITS and CJSTAT.
Subroutines Referenced:TCLOAD, CJITER, CJUPDT, TCIFIX and TCCNCT.

This routine functions in the same manner that TCFITR does for the normal reconstruction. It takes all the tracks located in the pattern recognition section, and selects those containing at least 15 points and which cross between sector boundaries in the JDC. Those that do not satisfy these conditions have their **TCTK** data banks dropped using the TCCNCT routine. Other wise the tracks are fit in a first pass using the CJITER routine.

After all tracks have been fit, the TCIFIX routine is called to correct the positions of the hits based on the crossing angle through the jet cells, and then the tracks are re–fit using the CJITER routine. The resulting coordinates of these hits are then passed to the CJUPDT routine for output.

5.2.5 SUBROUTINE CJGZFT

Author: Curtis A. Meyer
Creation Date: 26 July, 1989.
References:
Call Arguments: (IOPT).
Common Blocks Used: CBLINK, CBBANK, TCLIFT, TCPRMS, CJCUTS, CJGZFT, CBUNIT and TCDEBG.
Called by: CJCALB.
Subroutines Referenced: CJZCHK,
(CERNLIB: MATIN2
(ZEBRA): MZWORK and MZLIFT.

his routine will perform a global straight line fit to all tracks in the JDC to a common vertex. The routine is called via calibration option 4 in the CJCALB routine. If the passed value of IOPT is zero, the routine will write out the fit data. Otherwise no data will be written. In order for this routine to work, there must be at least MTRKCJ tracks in the JDC which contain no fewer than MHITCJ hits each. The flow of the code is divided into three sections, each of which are detailed below.

In the first section, a search is made through the JDC for all tracks which have more than 5 hits, and reside in a sector which has not been turned off in the LSECCJ array in the /CJCUTS/ common block. Up to 10 of these tracks are allowed, and when one is found the number of hits in the track, the pointer to the **TCHT** bank for every hit, the layer number of every hit and the sector number of every hit are stored in the /CJGZFT/ common block. Also, the (r,z) coordinate of every hit is stored in the **Y** vector as per the CJHXLD routine, and the errors , (σ_r^2,σ_z^2) are stored in the **GYINV** vector. At the same time this loading is done, the straight line fits:

$$\begin{aligned} z &= \tan(\lambda)\cdot r + z_0 \\ \cos(\psi)\cdot y &= \sin(\psi)\cdot x + \beta \end{aligned}$$

are made for each track. The fit values of $\sin(\psi)$, $\cos(\psi)$ and β for each track are then used to determine the point (x_0,y_0) which minimizes the sum of the squares of the distances to each line. This quantity is:

$$\chi^2 = \sum_{i=1}^{n} \left[\cos(\psi_i)\cdot y_0 - \sin(\psi_i)\cdot x_0 - \beta_i\right]^2 .$$

This is solved by inverting the matrix equation:

$$\begin{pmatrix} \sum\cos^2(\psi_i) & -\sum\cos(\psi_i)\cdot\sin(\psi_i) \\ -\sum\cos(\psi_i)\cdot\sin(\psi_i) & \sum\sin^2(\psi_i) \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} -\sum\sin(\psi_i)\cdot\beta_i \\ \sum\cos(\psi_i)\cdot\beta_i \end{pmatrix}$$

If after loading all tracks, there are still at least two tracks, and the radius of the (x,y) vertex is smaller than 5.0 cm, then the routine passes into the fitting section.

In the next section, the N tracks located in the first section are assumed to satisfy the following linear equations:

$$\begin{aligned} x_{j;k} &= X_0 + \cos\lambda_k\cdot\cos\psi_k\cdot\alpha_{j;k} \\ y_{j;k} &= Y_0 + \cos\lambda_k\cdot\sin\psi_k\cdot\alpha_{j;k} \\ z_{j;k} &= Z_0 + \sin\lambda_k\cdot\alpha_{j;k} \end{aligned}$$

where the index k runs over the N tracks, and the index j runs over the n_k points in track k. The first two of these equations are solved to yield α as:

$$\alpha_{j;k} = \sec(\lambda_k)\cdot\left[-(x_0\cdot\cos(\psi_k)+y_0\cdot\sin(\psi_k)) + \sqrt{(x_0\cdot\cos(\psi_k)+y_0\cdot\sin(\psi_k))^2 + r_i^2 - x_0^2 - y_0^2}\right]$$

The remaining equations in z can then be written as $\sum_{k=1}^{N}(n_k)$ equations of constraint. However, before continuing, it is also necessary to define the set of measured coordinates, $Y_{j;k}$, such that $Y_{2j-1;k} = r_{j;k}$ and $Y_{2j;k} = z_{j;k}$. We also define the fit vector X_l such that $X_k = \tan(\lambda_k)$, and $X_{N+1} = Z_0$, The equations of constraint are then:

$$\begin{aligned} f_{j;k} &= X_{N+1} - Y_{2j;k} - X_K \times \\ &\quad \left[-(x_0\cdot\cos(\psi_k)+y_0\cdot\sin(\psi_k)) + \sqrt{(x_0\cdot\cos(\psi_k)+y_0\cdot\sin(\psi_k))^2 + Y_{2j-1;k}^2 - x_0^2 - y_0^2}\right] \end{aligned}$$

We now have a system of $2\cdot\sum_{k=1}^{N}(n_k)$ measurements, $\sum_{k=1}^{N}(n_k)$ constraints and $N+1$ unknowns to solve.

The solution is obtained using the same iterative procedure as used in all other fitting routines. The matricies $\mathcal{A}$ and $\mathcal{B}$, and the vector $\mathbf{C}$ are defined such that:

$$\begin{aligned} \mathcal{A}_{ij} &= \frac{\partial f_i}{\partial X_j} \\ \mathcal{B}_{ij} &= \frac{\partial f_i}{\partial Y_j} \\ \mathbf{C}_i &= f_i \end{aligned}$$

The matrix $\mathcal{G}_B = (\mathcal{B}^T\mathcal{G}_Y^{-1}\mathcal{B})^{-1}$ is then computed, (where $\mathcal{G}_Y^{-1}$ is the covariance matrix for the measured quantities, Y.) This matrix turns out to be diagonal, and can be written simply as:

$$\mathcal{G}_B^{j;k} = \frac{1}{\delta^2 Y_{2j;k} + b_{j;k}\cdot\delta^2 Y_{2j-1;k}}$$

where the quantity $b_{j;k}$ is defined to be:

$$b_{j;k} = \frac{X_k\cdot Y_{2j-1;k}}{\sqrt{(x_0\cdot\cos(\psi_k) + y_0\cdot\sin(\psi_k)^2 + Y_{2j-1;k}^2 - x_0^2 - y_0^2}}.$$

Using the above matricies, the corrections to X and Y can be computed as,

$$\begin{aligned} \mathbf{X}_{i+1} &= \mathbf{X}_i - (\mathcal{A}^T\mathcal{G}_B\mathcal{A})^{-1}\mathcal{A}^T\mathcal{G}_B\mathbf{C} \\ \mathbf{Y}_{i+1} &= \mathbf{Y}_i - \mathcal{G}_Y^{-1}\mathcal{B}^T\mathcal{B}_B(\mathbf{C} - \mathcal{A}(\mathcal{A}^T\mathcal{G}_B\mathcal{A})^{-1}\mathcal{A}^T\mathcal{G}_B\mathbf{C}) \end{aligned}$$

Upon completion of the iteration, the covariance matrix for $\mathbf{X}$ is given as

$$\mathcal{C}_X = (\mathcal{A}^T\mathcal{G}_B\mathcal{A})^{-1}.$$

The quantity χ^2 is now computed as:

$$\chi^2 = (\mathcal{B}\vec{\epsilon})^T\mathcal{G}_B(\mathcal{B}\vec{\epsilon})$$

with $\vec{\epsilon} = \vec{y}_0 - \vec{y}_i$, the difference between the initial value of $\vec{y}$ and its value at iteration i. The iteration is then continued until one of the following conditions is met.

- The number of iterations exceeds 10. If this occurs, then the routine adds one to ISTACJ(36) in the /CJSTAT/ common block, and exits.
- The value of χ^2 increases by more than 5% on an iteration. If this happens, one is added to ISTACJ(37), and the routine exits.
- The value of χ^2 falls below CXGZCJ in the /CJCUTS/ common block. If this occurs, the routine passes into the final section.

- The value of χ^2 changes by less than CXGZCJ in one iteration. If this happens, the routine passes into the final section.

In the final section of this routine, the fit z coordinates along the tracks, as well as the left and right amplitude of each hit are used to compute what the z–calibration constant for each wire should be.

$$g_{j;k} = (a_r^{j;k}/a_l^{j;k}) \cdot (z_{j;k} - 20)/(z_{j;k} + 20)$$

These values are then added into the arrays in the /CJGAIN/ common block. The fit values of z are also stored in the **TCHT** and **TJDC** data banks, and the fit values of $\tan(\lambda_k)$ along with their errors are stored in the **TCTK** data banks. Next a **TCVX** bank is created for the fit vertex. Finally, the fit errors in z are computed as the $(2j, 2j)$ diagonal elements of the fit covariance matrix for the measurements:

$$\mathcal{C}_Y = \mathcal{G}_Y^{-1} - \mathcal{G}_Y^{-1}\mathcal{B}^T\mathcal{G}_B\mathcal{B}\mathcal{G}_Y^{-1} + \mathcal{G}_Y^{-1}\mathcal{B}^T\mathcal{G}_B\mathcal{A}(\mathcal{A}^T\mathcal{G}_B\mathcal{A})^{-1}\mathcal{A}^T\mathcal{G}_B\mathcal{B}\mathcal{G}_Y^{-1}.$$

These fit values of z, as well as the raw amplitudes are also written to the file **CALGLOBZ.DAT**. This data is then read in by a more complicated fitting procedure to allow more parameters per wire. This data is all packed into two integer words per hit. The first word contains the fit z coordinate as $iz = 25000 + 1000 \cdot z_{fit}$ packed into bits 1 through 22, and the wire number as $w = (s-1) \cdot 23 + l$ packed into bits 23 to 32. The second word contains the right or $-z$ amplitude packed into bits 1 to 16, and the left or $+z$ amplitude packed into bits 17 to 32.

At this point, the common block /CJGLOZ/ is loaded with the following data. The fit values of z are copied into the ZPFTCJ array, the difference between the initial and fit values of z are placed in the DZFTCJ array, and the computed errors in z are stored in the SZFTCJ array. These values are thereby made available to the USER routines. The routine will also fill the **TCTK** data bank as shown in table 16. Note, this is only done for those tracks used in the fit. As such, it is necessary to use the /CJGLOZ/ common in conjunction with the **TCTK** banks. Also, the *user service routines* are unable to process the fit data from these banks; it is necessary that the user unpack them by hand.

5.2.6 SUBROUTINE CJINIT

Author: Curtis A. Meyer
Creation Date: 5 September, 1988.
References: FFREAD long write up.
Call Arguments: None.
Common Blocks Used: TCCUTS, TCPRMS, CJCUTS and CJFLAG.
Subroutines Referenced:CJWIPE.
(FFREAD:) FFINIT, FFKEY, FFSET and FFGO.

This routine is called by the TCINIT routine to initialize the calibration part of the code. The main purpose of this routine is to allow the user to change some of the fit parameters used in calibration. The routine uses the FFREAD package to read in a card file from unit LJCRD. The allowed cards are as follows:

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. 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.**

Offset	TYPE	*Quantity*
+1	INTEGER	*Number of Hits*
+2	INTEGER	*Layer of Hit 1*
+3	INTEGER	*Hit number of hit 1*
+4	INTEGER	*Layer of Hit n*
+5	INTEGER	*Hit number of hit n*
+6	INTEGER	*Number of dE/dx hits*
+7	INTEGER	*Error code from fit*
+8	REAL	*Charge*
+9	REAL	*Mass* $[MeV]$
+10	REAL	dE/dx $[MeV/cm]$
+11	REAL	$\sigma_{dE/dx}$ $[MeV/cm]$
+12	REAL	0.0
+13	REAL	ψ_0 [radians]
+14	REAL	$\cos\psi_0$
+15	REAL	$\sin\psi_0$
+16	REAL	x_0 *vertex point.*
+17	REAL	y_0 *vertex point.*
+18	REAL	0.0
+19	REAL	$\tan\lambda_0$
+20	REAL	z_0 *vertex point.*
+21	REAL	0.0
⋮	⋮	⋮
+27	REAL	$\sigma^2_{\tan\lambda}$
+28	REAL	0.0
+29	REAL	$\sigma^2_{z_0}$
+30	REAL	χ^2

Table 16: The data stored in the subbanks of the **TCTK** bank, (**TCSG**). There is one **TCSG** data bank for each track found in the chambers.

CCUT sets the value of CCUTTC, the cutoff parameter used in TCITER and CJITER to define when convergence has been reached.

CFRA sets the value of CFRACJ in the /CJCUTS/ common block. The default value is 0.900.

CHCU sets the value of CHCTTC, a parameter for associating track segments in the TCASSC subroutine.

CHDS sets the value of CHDSTC, a cutoff parameter for defining the error code in the CJITER and TCITER routines.

CHLX sets the value of CHLXTC, the convergence cutoff in the TCHELX routine.

CLIM sets the value of CLIMCJ in the /CJCUTS/ common block. The default value is 150.

CLMB sets the value of CLMBTC, the χ^2 for associating tracks in $\tan\lambda$ in the TCSGMT subroutine.

CSMX sets the value of CSMXCJ in the /CJCUTS/ common block. This is the minimum value of $\cos\alpha$ to be written out by CJUPDT. To use this card, enter the value of α in radians.

CXGZ sets the value of CXGZCJ in the /CJCUTS/ common block. This is a convergence criteria in the CJGZFT routine.

CXSQ sets the value of CXSQTC, a parameter for associating hits in the TCSGMT subroutine.

DEBG sets the value of the DEBGTC flag in the /TCDEBG/ common block. Assuming that the code has been linked with the debug print lines, DEBGTC allows the user to turn them off. The default value is .TRUE., unless the PATCHY flag DEBFALSE was used in building the code, in that option the default will be .FALSE..

DELT sets the value of DELTTJ, the quadratic term in the error calculation.

DELY sets the value of DELYTJ, the linear term in the error calculation.

DELZ sets the value of DELYTC in the /TCPRMS/ common block. This is the error in the z position as assigned by the TJZPOS routine. If one uses this card, be sure to enter 23 values.

DMAX sets the value of DMAXTJ in the /TJCUTS/ common block. The default value is 0.130.

DMIN sets the value of DMINTJ in the /TJCUTS/ common block. This value is used in the pattern recognition routines. The default value is 0.028 .

DXSQ sets the value of DXSQTC in the /TCCUTS/ common block. This value is used in the TCRSRC and TCFSRC routines.

DYMX sets the value of DYMXCJ in the /CJCUTS/ common block. This is the maximum shift in centimeters a point can have, and be written out by the CJUPDT routine.

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.**

ICAL sets the value of ICALCJ, the JDC calibration steering parameter. This defaults to a value of 0, (see the CJCALB routine for a description).

IGAP sets the value of IGAPTJ in the /TJCUTS/common block. This is used in the TCSGMT routine do determine how large a gap a segment can have. It has a default value of 5.

INTG sets the value of ITIMRJ in the /RJPRMS/ common block. It is the integration time used in the RJAFIT routine, and has a nominal value of 15 FADC channels.

ISEP sets the value of ISEPRJ in the /RJPRMS/ common block. It is the minimum separation between double pulses in RJAFIT.

ITFC sets the value of ITFCRJ in the /RJPRMS/ common block. It is a t_0 offset for the fit pulses in RJPROC coming from the DL307 Flash ADC's. There are 16 values in the ITFCRJ array. **When using this card, the user should have 17=1 to cause the program to accept these values.**

ITMI sets the value of ITMIRJ in the /RJPRMS/ common block. It is the minimum pulse separation allowed in the RJPULS routine. It has a default value of 2.

IMAX sets the 23 integer values of ITMXTJ, the maximum drift time allowed in each layer of the JDC.

JDCL sets the value of JDCLCJ, the JDC calibration flag in the /CJFLAG/ common block. It defaults to .TRUE..

JGAP sets the value of JGAPTJ in the /TCCUTS/ common block. This is the maximum layer gap allowed in TCFSRC and TCRSRC, and has a default value of 2.

LGPD sets the value of LGPDRJ in the /RJPRMS/ common block. This tells the RJPROC routine if it should compute the pedestals dynamically from the presample. The normal value is .FALSE. which means pedestals are computed dynamically.

LGT0 sets the value of LGT0RJ in the /RJPRMS/ common block. This tells the RJPROC routine if t_0's should be subtracted from the times it stores in the **RJDC** data bank. The normal value is .FALSE., which means that t_0 subtraction is not performed.

LGT2 sets the value of LGT2TJ, a logical to control how errors in the JDC are computed.

LGWR is entered with a non–zero value to prevent the TJTIMI routine from overwriting the dead wire list. This is necessary during z calibrations.

LSEC is a card used to turn off sectors in the CJGZFT subroutine. The card can have up to 30 integer entries, which are the sector numbers to be turned off. If this card is not used, then all sectors are assumed on.

MCIR sets the value of MCIRCJ in the /CJCUTS/ common block. This is the minimum number of hits per track in the CJITER routine.

MHIT sets the value of MHITCJ in the /CJCUTS/ common block. This is the minimum number of hits per track in CJGZFT and CJZFIT.

MTRK sets the value of MTRKCJ in the /CJCUTS/ common block. This is the minimum number of tracks per event in the CJGZFT routine.

MXPL sets the value of MXPLRJ in the /RJPRMS/ common block. It is the maximum time separation between the left and right FADC pulses to be considered the same pulse.

NDBG sets the value of NDBGTC, which, if it is not set to zero, turns on the debug print lines for event number NDBGTC, and prints a simple message for every event processed. (Note: the code must have been installed with the debug print lines for this to work, otherwise it has no effect on the code.)

NHLX sets the value of NHLXTC, the maximum number of iterations in the TCHELX subroutine.

NITR sets the value of NITRTC, the maximum allowed number of iterations in both the TCITER and CJITER routines. The default value is 10.

NPUL sets the value of NPULRJ in the /RJPRMS/ common block. It is the minimum pulse height required in the RJPULS routine to accept a hit in the **RJDF** data bank. It has a default value of 6.

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.**

RJDD sets the value of the LGRJDD parameter in the /RJPRMS/ common block to .TRUE.. This forces the program to use online processed data.

RJDF sets the value of the LGRJDF parameter in the /RJPRMS/ common block to .TRUE.. This forces the program to perform Qt-analysis on the data, even if online processed data exist.

RSLV sets the value of the RSLVTC parameter in the /TCCUTS/ common block.

SXDQ sets the value of SXDQTJ in the /TJPRMS/ common block. The default value of this term is zero.

SY02 sets the value of SY02TJ in the /TJPRMS/ common block.

SYDF sets the value of SYDFTJ in the /TJPRMS/ common block.

SYDQ sets the value of SYDQTJ in the /TJPRMS/ common block. The default value of this term is zero.

TDIF sets the value of the TDIFTJ parameter in the /TJCUTS/ common block. This is used for merging hits in the TJDCGT routine.

TMIN sets the value of TMINTJ, the minimum time a hit can have, and still be resolved using $(t_1 + t_3)/2 - t_2$. The default value is 0.080 .

VFRC sets the value of VFRCTC in the /TCCUTS/ common block.

VPRB sets the value of VPRBTC in the /TCCUTS/ common block.

XSQR sets the value of XSQRTC, a pattern recognition cutoff, (see the TCRAW1 subroutine).

YCUT sets the value of YCUTTJ in the /TJCUTS/ common block. This is used in computing the hit position in the TJTIME routine.

ZAMP sets the value of ZAMPCJ in the /CJCUTS/ common. This is the minimum amplitude both left and right must have for the hit to be written to the calibration file. It has a default value of 200.

ZCTT sets the value of ZCTTCJ in the /CJCUTS/ common. This is the global z–vertex cut, and has a default value of 5 cm.

ZCUT sets the value of ZCUTCJ in the /CJCUTS/ common. This defines a window around ZOFFTC along the z–axis, into which tracks must project if they are to be used in z–calibrations.

ZMOV sets the value of ZMOVCJ in the /CJCUTS/ common. This is the amount a point can be shifted in the CJGZFT routine, and still be used for calibration. It has a default value of 4 cm.

ZOFF sets the value of ZOFFTC in the /TCPRMS/ common. This is used in computing $\tan\lambda$ in the TJZPOS routine.

ZPOS sets the value of ZPOSCJ in the /CJCUTS/ common. This sets the allowed range in z on all wires to be used in z–calibration.

ZPRB sets the value of ZPRBCJ in the /CJCUTS/ common. This is a probability cut in CJGZFT and has a default of 0.00.

ZRIN sets the value of ZRINCJ in the /CJCUTS/ common. This is an xy vertex inner cut, and defaults to 0.00 cm.

ZROT sets the value of ZROTCJ in the /CJCUTS/ common. This is an xy vertex outer cut, and defaults to 5.00 cm.

ZVTX sets the value of ZVTXTC in the /TCCUTS/ common block.

ZXYR sets the value of ZXYRCJ in the /CJCUTS/ common. This is the minimum distance in the xy plane all tracks must come to the xy vertex in order to accept the event. It defaults to 0.50 cm.

HH Finally, this routine will open the output files used by the various calibration routines for printing calibration data. The name of the opened file is dependent upon the value of ICALCJ in the /CJFLAG/ common block. For a value of zero, the file **CALCIRCLE.DAT** is opened; for the case of two, the file **CALZFIT.DAT** is opened; and for the case of four, the file **CALGLOBZ.DAT** is opened. (Note: these files are only opened if the VAX or ALT switch is used in the PATCHY cradle.)

5.2.7 SUBROUTINE CJITER

Author: Curtis A. Meyer
Creation Date: 19 September, 1988.
References: Datenanalyse by Siegmund Brandt, p. 271.
Call Arguments: (ITRK, *Y, *GYINV, *X, *COVR, *CHISQ,*IERR).
Common Blocks Used: CBBANK, CBLINK, CJSTAT, TCCUTS, TCHITS, CBUNIT and TCANGL.
Subroutines Referenced: ZEBRA LIBRARY: MZWORK.

This routine accepts as input the track number, ITRK, the measured quantities, Y and their errors, GYINV, and a guess to the fit parameters, X. The routine then iterates the equations as in the TCITER routine, but unlike the TCITER routine, it also updates the measured coordinates along the tracks. As in TCITER, the convergence criteria is controlled by the CCUTTC parameter in the /TCCUTS/ common block, and the returned error code is set using the CHDSTC parameter in the /TCCUTS/ common block. The values of these variables can be set using the **CCUT** and **CHDS** cards of the CJINIT routine.

Once convergence has been reached as in TCITER, the routine returns the fit parameters, X, their covariance matrix, COVR, the chisquare of the fit, CHISQ, an error code from the fit as per TCITER, IERR and the new values for both the measured quantities, Y and their errors, GYINV. Also available is the RDEVTC array in the /TCANGL/ common block. This contains the deviation of every point from the fit track.

5.2.8 SUBROUTINE CJSLOW

Author: Curtis A. Meyer
Creation Date: 22 August, 1990.
References:
Call Arguments: None.
Common Blocks Used: SCLINK, CBLINK, CBBANK, CJSLCN.
Called by: TJSLOW
Subroutines Referenced:

This routine averages the slow control data measured during a run and writes it to an external file that is later included in the $r\phi$ calibration work.

5.2.9 SUBROUTINE CJUPDT

Author: Curtis A. Meyer
Creation Date: 19 September, 1988.
Revised: 19 July, 1989.
References:
Call Arguments: (ITRK, ICODE, NPTS, Y).
Common Blocks Used: CBLINK, CBBANK, CBUNIT, CJSTAT, TJPRMS, CJGAIN and TCPRMS.
Subroutines Referenced: TCRHIT.

This routine will take a vector Y of NPTS fitted coordinates and write information to a claibration file for use in JDC calibrations. The passed argument, ITRK is the track number in either the **TCTK** or **TCTR** bank, and the value of ICODE identifies if the data in Y is from a circle fit or a helix fit.

For ICODE equal to zero, the data is assumed to be from a circle fit. Here the values in y are NPTS (r,ϕ) pairs. For ICODE equal to one, the data is assumed to be from a helix fit. Here the data in Y is NPTS (x,y,z) triplets.

This routine will write the the file assigned to logical unit LJTOUT which is usually opend as CALCIRCLE.DAT. There are two packed integer words per hit. In the first word, bit 1 identifies left or right, (0 is left and 1 is right), bits 2 to 8 contain the layer number, bits 9 to 16 contain the sector number, and bits 17 to 32 contain the drift time in units of 0.5 ns. In the second word, bits 1 to 15 contain the absolute value of x, bit 16 is the sign bit for x, bits 17 to 31 contain the absolute value of y and bit 32 is the sign bit for y.

5.2.10 SUBROUTINE CJWIPE

Author: Curtis A. Meyer
Creation Date: 19 September, 1988.
References:
Call Arguments: None.
Common Blocks Used: CBUNIT, CJFLAG, CJSTAT and CJGAIN.
Subroutines Referenced: None.

This routine performs zeroing and initialization necessary before the start of a calibration run.

5.2.11 SUBROUTINE CJZCHK

Author: Curtis A. Meyer
Creation Date: 06 October, 1989.
References:

Call Arguments: (ITRK, Y, GYINV, TGL, B).
Common Blocks Used: CJGLOZ.
Called By: CJGZFT **Subroutines Referenced:** SM353.

This routine will examine the straight line track, ITRK defined by the points in the double precision variable Y, and the errors in those points in the double precision variable GYINV, and determine if any of these points are too far away from the line given by TGL(ITRK) and B. This is determined by computing the deviation of each point from the fit line, and then smoothing these deviations using the SM353 routine. Those points that were shifted by more than 1.5 sigma in z are then dropped from the track. The track is then refit and returned to the calling routine.

5.2.12 SUBROUTINE CJZFIT

Author: Curtis A. Meyer
Creation Date: 2 March, 1989.
References:
Call Arguments: None.
Common Blocks Used: CJGLOZ, CJGAIN, CBBANK, TCLIFT and CBLINK.
Subroutines Referenced: (ZEBRA) MZLIFT.

This routine calibrates the c_i^z constants for the JDC. These constants are used to determine the z position on every wire, and contain the ratio of gains between the $+z$ end and $-z$ end preamplifiers. The calibration is performed by fitting a line to all tracks found in the r–z plane, and then computing the deviation in z from the fit track. With this deviation, (Δz), one can extract what the gain constant should be as:

$$c_i^z \quad = \quad \left[\frac{A_+ \cdot (20. + \Delta z)}{A_- (20. - \Delta z)} \right]$$

These constants are tallied in the ZFITCJ array of the /CJGAIN/ common block, and new values are computed in the CJUPDT routine. Information on an event by event basis is available in the /CJGLOZ/ common block. Also, a vertex bank is created for every track, which contains the projection to $r = 0$. A track must have more than MHITCJ hits to be used by this routine.

5.2.13 SUBROUTINE CJ4PRG

Author: Curtis A. Meyer
Creation Date: 22 April, 1990.
References:
Call Arguments: (*PROB3, *PROB4, LGHLX, LONG, LYR0, LYRN, LG3C).
Common Blocks Used: CBLINK, CBBANK, CJPCAL TRKPRM and CBUNIT.
Called by: CJCALB.
Subroutines Referenced: TCKFT3, TCKFT4, TCHLDS,
CERNLIB: PROB.

This is a filter routine used to select events of the type

$$\overline{p}p \to \pi^+\pi^-\pi^+\pi^-$$

and

$$\overline{p}p \to K^+K^-\pi^+\pi^-.$$

The routine first requires that there be a vertex which has exactly four tracks. Then each of these tracks is required to have at least LONG hits. Each track is also required to start no later than LYR0

and end no earlier than LYRN. Then the routine requires that the charge at the vertex sum to zero. If the value of LGHLX is .TRUE., the data is taken from the **TCTR** bank through the pointers in the **TCVP** bank. If LGHLX is .FALSE., then the data is simply taken from the **TCVP** data bank. The initial values of 4–momentum are then loaded into the /CJPCAL/ common block, and then a 3–C kinematic fit is performed, (see TCKFT3). The probability of this is then assigned to the variable PROB3. If the value of LG3C is set .TRUE., then these 3–C values are returned. If not, then a 4–C kinematic fit is performed, and the resulting momentum and covariance matricies are stored in the /CJPCAL/ common block. Then the probability of this fit is assigned to the variable PROB4.

When the variable LGHLX is given as .TRUE., then the TCHLDS routine is used to obtain the 3–momentum, and the full 3 by 3 covariance matrix for each track. This routine doubles all errors associated with the helix parameter ψ_0 because no vertex constraint has been placed on the event. When LGHLX is .FALSE., then the covariance matrix is simply diagonal, and taken from the **TCVP** data bank. This routine then doubles the errors of all momenta, (quadruples the diagonal of the covariance matrix).

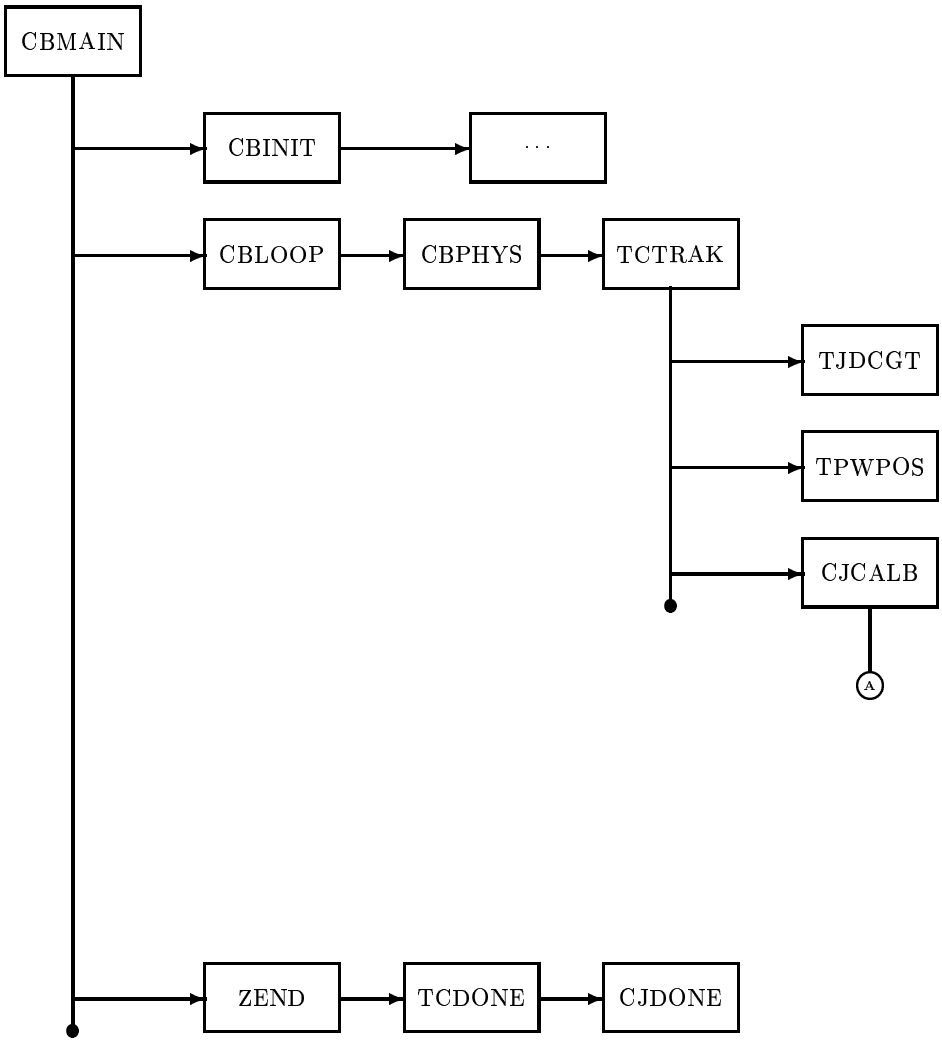


Figure 12: The calling sequence for the calibration code.

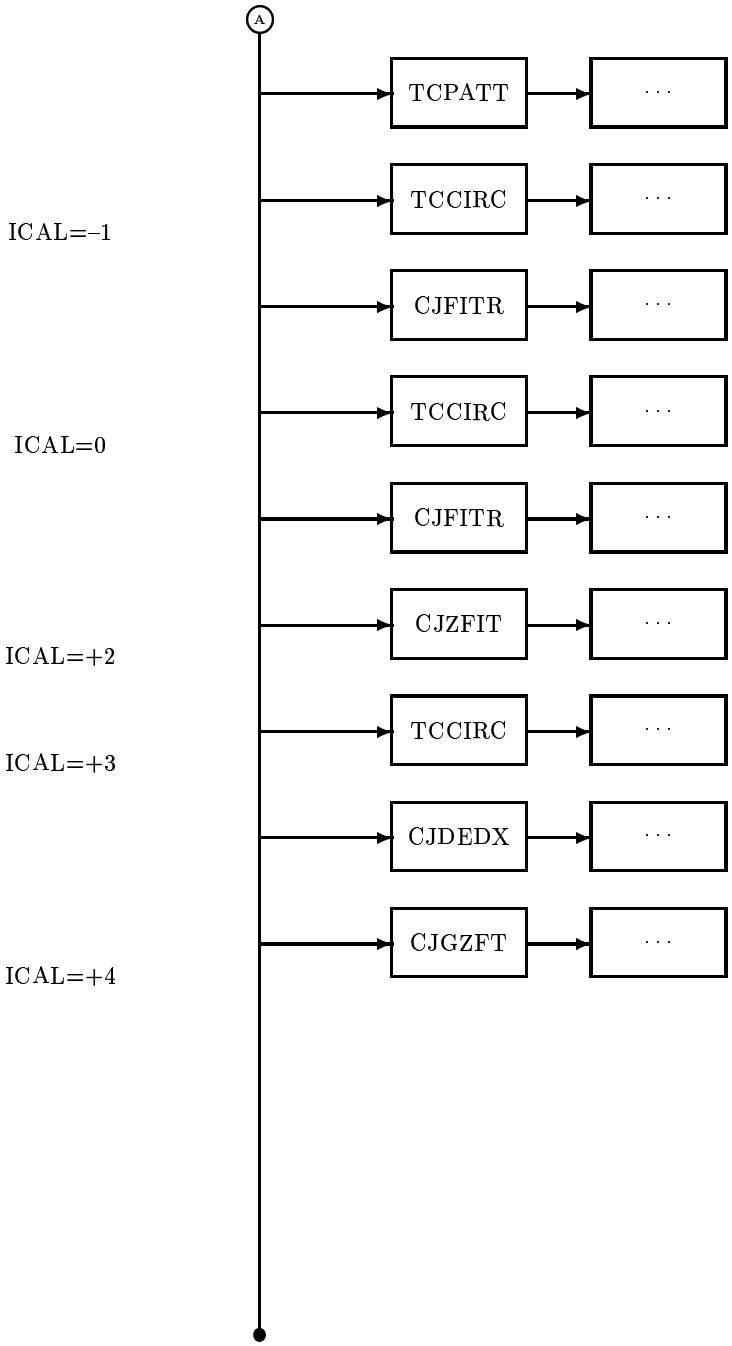


Figure 13: Flow of chamber calibration software.

References and further Information

- S. Brandt, **Datenanalyse** .
- R.L. Gluckstern, **NIM 24**, 381, (1963).
- CERN Program Library F101, (MATIN2).
- R. Brun, *et. al.*, **FFREAD Users Guide**. CERN Program Library I302, (1987).
- J. Zoll, **Zebra Reference Manual, MZ**. CERN Program Library Q100, (1987).
- J. Zoll, **Zebra Reference Manual, FZ**. CERN Program Library Q100, (1987).
- J. Zoll, **Zebra Reference Manual, Dia**. CERN Program Library Q100, (1987).
- R. Brun and D. Lienart, **HBOOK User Guide, Version 4**. CERN Program Library Y250, (1987).
- G.J. VanDalen, **Track fitting methods for the TPC Detector**, TPC–UCR–79–3, (1979).
- A. Weinstein, **More SLCFND documentation, and status of tracking**, MARKII/SLC NOTE # 127, (1986).
- A. Weinstein, **Yet more SLCFND documentation**, MARKII/SLC NOTE # 132, (1986).
- A. Weinstein, **Technical note on tracking resolution for small angle tracks**, MARKII/SLC NOTE # 133, (1986).
- J. Olsson, *et. al.*, **NIM 176**, 403, (1980).