CB-Note 121

LEAR Crystal Barrel Experiment, PS197
Offline Reconstruction Software Vers. 1.27/00

Gunter Folger / Michael Doser / Christian Voelcker
CERN

April 7^{th}, 1995

Contents

1 Description of Crystal Barrel Offline

The CBOFF programm provides a frame for analysis of Crystal Barrel data. It reads CB data tapes – or disk files – and may call reconstruction routines, depending on the type of input data and on input cards given by the user. The data tapes may be raw data tapes, Monte Carlo data, reconstructed CB data (DST tapes), or mini-DST tapes. The reconstruction of an event proceeds in steps

bctrak reconstruction of PED's in the CsI crystals, see CB note 92 [1]

tctrak charged track finding in the JDC and PWC using LOCATER, see CB note 123 [2] and also CB note 93 [3]

gttrak global tracking, i.e. matching of PED information with information on charged tracks, see CB note 118 [4],

kinfit kinematic fitting of events.

The results from any of these steps may read from tape (DST), and reconstruction for this step will then not be repeated.

Prior to calling any of the reconstruction routines, CBOFF loads the proper calibration constants into memory using a CB database program, see CB note 122 [5]. If there are several sets of valid constants, the one last entered into the data base is automatically selected. The user can select a set of calibration constants different from the default set (see section on input cards, CALI).

The results are available in user written subroutines. Access to all data is possible at various stages during reconstruction in subroutines written by individual users. The data are available in data banks managed by ZEBRA [8, 10] and/or in common blocks. For some banks there exist routines to ease extraction of the information stored in that bank.

1.1 Requirements to run CBOFF

To run CBOFF with either real data or Monte Carlo data one has to supply:

- a set of user routines as described in section 2. Some or all of those may of course be dummy subroutines. A set of dummy subroutines can be found in source code of CBOFF in the +PATCH,USER .
- a set of data cards as described in section 3 .
- file definitions (ie. SETENV on the decstations, FILEDEF on VM, DEFINE on VAX, DD in JCL, ..) for input and output files, see section 4 and 9.3.17 .

2 User Routines

To run an analysis program the user has to add code for his particular analysis. A few user written routines are called by CBOFF to allow for interaction and analysis of the data. The user has to provide the following routines

usinit initialize, ie. book histograms ...

user analysis on every event

uslast Terminate analysis, i.e. print results, histogramms ...

usevnt called during reconstruction, see below.

usrun a run started or ended, do run based statistic

Any subroutine may be a dummy subroutine which immediately returns. In the following a more detailed description of each of these subroutines is given.

2.1 Subroutine USINIT

Arguments: IERR
This routine is called by the offline program once at the beginning of program execution. It allows to initialize eg. HBOOK and to book histograms. It is called after reading the data cards, initialisation of ZEBRA and all initialisation of reconstruction routines. As a consequence of initialisation of ZEBRA one has to call HLIMIT with a negative argument.

In this subroutine, you can overwrite default values set during from initialisation. The most useful things are

- If output of data was turned on, then one can decide which divisions and thereby the output of which reconstruction step will be written to tape. You may also omit certain banks from being written to tape by requesting that they be lifted into the IXEVNT division (set IX*bank* = IXEVNT, where *bank* is the name of the data bank)

- Select the type of non-physics events written to DST
  You must include the line +SEQ,CBLOUT. in your routine.
  Slow control events : set OUTSLC to TRUE/FALSE (default=TRUE)
  Table events : set OUTTAB to TRUE/FALSE (default=TRUE)
  Events failing analysis : set OUTERR to TRUE/FALSE (default=TRUE)

If on return the argument IERR$\neq$ 0 the analysis is stopped. This should be used to stop the program in case an error occurs in the initialisation.

2.2 Subroutine USER

Arguments: none
This is the main user routine. It is called once for every physics event[1] after all reconstrution for this event is finished. For a DST tape – normally – no reconstruction is necessary, except for filling a few common blocks.

When reconstructing, an event which failed in some step will not get into USER. It will get into USEVNT instead, see 2.5.

If output of data was turned on, the variable ACCECB from COMMON /CBCNTL/ should be used to decide if the event will be written or not. Also banks which should not appear on output tape should be dropped using MZDROP.

As an example, assume you want to drop all global tracking banks, i.e. the supporting bank HTRK and its two sub-banks TVTX and TTKS. Then you would have to include the following lines in your code:

```
CALL MZDROP(IXSTOR,LHTRK,'LV')
CALL MZDROP(IXSTOR,LHTRK,' ')
LHTRK = 0
LTVTX = 0
LTTKS = 0
```

2.3 Subroutine USLAST

Arguments: none
Called once during the termination phase of program execution, USLAST allows the user to print or store results and histograms.

2.4 Subroutine USRUN

Arguments: ICODE
This routine is called whenever a run ends (ICODE=2), a new run begins (ICODE=1), at the end of the input file (ICODE=4) and at the end of the input tape (ICODE=5). This routine could be used for run by run statistics. It can also be used to skip past an end of file or an end of tape, as in the following code segment. Note that CBNTPE and CBNFIL are decstation specific routines, and are not supported on other machines. In addition, see the section on input cards (FZIN) before attempting to use the following code.

```
*
*  end of input file, should we continue with next file ?
```

[1] All other types of events (e.g. flasher, begin of run, etc, see section 9.2.1) are not run through the analysis chain and are therefore not available in USER. These events are available in USEVNT(0), see 2.5.

```
*  we only get ICODE=4 from Zebra if we set FZIN = 'TX1' or 'X1'
*  in the input data cards
*
      ELSEIF( ICODE.EQ.4 ) THEN
         CALL CBNFIL(ISTAT)
*
*  if we hit eot, zebra will *still* give code 4, so one could try
*  to load the next tape whenever CBNFIL comes back with an error.
*  risky ? you bet !
*
         IF (ISTAT.NE.0) THEN
            CALL CBNTPE(ISTAT)
         ENDIF
          STOPCB = ISTAT.NE.0
         IF (.NOT.STOPCB) THEN
           CALL FZENDO(LDST,'C')
         ENDIF
*
*  end of input tape, should we continue with next tape ?
*
      ELSEIF( ICODE.EQ.5 ) THEN
         CALL CBNTPE(ISTAT)
         STOPCB = ISTAT.NE.0
         IF (.NOT.STOPCB) THEN
           CALL FZENDO(LDST,'C')
         ENDIF
      ENDIF
```

It may also be used to steer reading of data : called after the start of a new run, i.e. called with I=1 the user can tell the CBOFF program to skip this run by setting the variable ACCRCB to .FALSE. This variable is available in a common block and should be included via +SEQ,CBCNTL.

2.5 Subroutine USEVNT

Arguments: ICODE
Called at various stages of during processing of data, as indicated by the parameter ICODE. Currently the possible values of the parameter ICODE are :

ICODE=-1 event header IEHDCB read in, the rest of the event is still pending. This allows a fast preselection of events using the characteristics of the event stored in IEHDCB, see table 1. The event may be skipped by including the line SEQ,CBCNTL. and setting the variable ACCECB to .FALSE. .

ICODE=0 event is read in. No reconstruction has been done. Here one may skip this event from analysis by setting ACCECB to .FALSE. This variable is available by including the CBCNTL common via SEQ,CBCNTL.
This routine could also be used for analysis of non physics events as for example flasher events.

ICODE=10 called from BCTRAK before any BC routine.

ICODE=11 called after BCDECF is done, FERA raw data block is decoded

ICODE=12 called after BCDECF is done, 2282 raw data block is decoded

ICODE=13 called after BCALCE is done, energies are available either in the bank TBEN or in COMMON BCENER

ICODE=14 called after BCLUST is done, cluster are available in the bank TBCL

ICODE=15 called after BCPEDS is done, peds are available in the bank TBTK

ICODE=20 After unpacking of the **pwc** data, (TPWCGT).

ICODE=21 After unpacking of the **jdc** data, (TJDCGT).

ICODE=22 After pattern recognition, (TCPATT).

ICODE=23 After circle fitting, (TCCIRC).

ICODE=26 After helix fitting, (TCHELX).

ICODE=27 After vertex fitting, (TCVERT).

ICODE=29 After calibration, (CJCALB).

3 Data Cards

At the beginning of program execution data cards are read in by the subroutine CBFFGO using the FFREAD package[7]. This allows to change program flow and to change constants without recompiling and relinking the code.

The cards are read from the standard input unit, normally unit 5. Normally it is not necessary to specify all cards. If a card is not present, the defaults are used.

3.1 Data Card LIST

Format: LIST
This card has no parameters. It directs FFREAD to echo all following cards.

3.2 Data Card FZIN

Format: FZIN par1 par2 par3 par4 **Default:** none

This card tells the program what format the in/out data are.

par1 **type** Character*4

This parameter is given without checking to the Zebra routine FZFILE as variable CHOPT. Typical values are (for a complete list see [9, 8]) :

”	reading a disk file in native format
’X’	reading a disk file in eXchange format (i.e. to read slow control tapes)
’A’	reading a disk file in exchange format using Alpha mapping
’T’	reading from Tape in native format
’TX’	reading from Tape in eXchange format
’Y’	now needed to process tapes on the IBM at CERN
’1’	to read several files from one tape

The following are thus usual combinations (note that data from online are in exchange format) :
’TX’ : usual on decstations
’X’ : for slow control tapes or MC files
’TX1’ : read several files from one tape
’TXY’ : usual on CERN IBM

par2 **type** Integer

The second parameter sets the logging level for the data input file with a call to FZLOGL. Possible values are from -3 to +4 (see [9, 8]), the most useful values are :

-2 print only error messages

0 normal mode

1 as 0, plus details of conversion problems

par3 **type** Integer

The third parameters sets record size in words for Zebra if eXchange format is used. This parameter normally is not needed, as the program uses the correct CB default values (900 for disk, 5760 for tape).

par4 **type** Integer

This parameter is presently not used. It is reserved for a future extension.

The defaults for the values are par1=" (ie. Native format from disk), par2=0 and par3=0.

3.3 Data Card FZOUt

Format: FZOUt par1 par2 par3 par4 **Default:** none

The meaning of parameters is identical to FZIN, except refering to output. For par3 only a values of 900 or 5760 is accepted. If you omit this card, no events will be written out.

3.4 Data Card FZSEl

Format: FZSEl par1 par2 par3 par4
Default: none

This card is not used.

3.5 Data Card RSELect

Format: RSELect par1 par2 par3 par4 . . . par50
Default: none

Use this card if you want to selectivly want to analyse runs. There are two choices, which may be used together :

- Analyse from run/event to run/event.

  par1,par2 first run,event to analyse

  par3,par4 last run,event to analyse, par3=0 continue to end of data, par4=0 continue to end of run given in par3.

- Exclude some runs from analysis

  par4. . . par50 do not analyse runs listed here.

3.6 Data Card TIME

Format: TIME par1 par2 **Default:** none

This card allows to set a time limit to an analysis job, ie. the the job will stop before there is no more cpu time available.

par1 type Real
time to leave for termination of the job in seconds (real cpu seconds, not accounting time, varies with computer).

par2 type Integer
check for this time limit every par2's event.

3.7 Data Card USER

Format: USER par1 par2 ... par50 **Default:** none

These parameters are for free use in the user routines. The values are stored in the COMMON/USKEYS/ to be included with Patchy +SEQ,USKEYS. The parameters are available in the equivalenced arrays KEYSUS and IKEYUS for real and integer parameters, respectively.

3.8 Data Card CHAM

Format CHAM par1 par2 ... par10 **Default:** do (or redo) the charged reconstruction

This cards allows to turn on selectively parts of the code for reconstruction of charged particles in the PWC/JDC.

The ordering of the parameters is irrelevant, as this card accepts keywords. Possible keywords are :

'TRAK' turn on track reconstruction.

'GPWC' include the PWC data in reconstruction

'RAWS' Unpack the raw **jdc** data, TJDCGT.

'PATT' Perform pattern recognition on the data, TCPATT.

'RADO' Use fuzzy radon pattern recognition in $r - \phi$ instead of tcpatt.

'RASH' Use extended fuzzy radon pattern recognition ($r - \phi - z$) to look for very short tracks if exactly one track has been found already (In-flight analysis of two-prong data).

'FFUZ' Use Fast Fuzzy pattern recognition (from Locater 1.97/00). This card will dominate the 'RADO' key, as it does not make sense to use both!

'CIRC' Perform a circle fit to the pattern recognized tracks, TCCIRC.

'HELX' Perform a helix fit to the found tracks, TCHELX.

'VERT' Perform a single vertex fit to the found tracks, TCVERT.

'VER2' Fit primary and secondary vertices TCVER2. This is NOT default and NOT yet used in dst production. You should set this card if you want to look for Kshort's. TCVER2 requires the existence of the TCTR bank.

'RTRK' Completely redo charged tracking of data.

'NONE' Completely turn off charged tracking.

The defaults is to do everything, as long as the output banks do not exist. Normal pattern recognition will be used. On a RDT, this would be full processing. On a DST the results can be random depending on which banks are found on tape. No checking is done to insure that the data cards given are consistent. To perform any reconstruction, one must specify 'TRAK'. To use fuzzy radon pattern recognition one has to set both keywords, 'PATT' and 'RADO'. The special short track algorithm (keyword 'RASH') may be applied even if 'RADO' is not set. To turn reconstruction of charged tracks completely off use the keyword 'NONE' and no other keyword. To fully retrack a DST, use
CHAM 'TRAK' 'RTRK' 'GPWC' 'RAWS' 'PATT' 'CIRC' 'HELX' 'VERT'
CAVEAT: presently only 10 keywords are accepted and the 11th will be ignored..

3.9 Data Card XTAL

Format XTAL par1 par2 ... par10
Default: RDT → DST : do everything
Default: DST → DST' : do nothing (unless RTRK is set)

This cards allows to turn on selectively parts of the code for reconstruction of crystal data.

The ordering of the parameters is irrelevant, as this card accepts keywords. Possible keywords are :

'TRAK' turn on crystal reconstruction

'DECF' decode FERA counts

'DECL' decode 2282 counts

'ALCE' do BCALCE

'CLST' do BCLUST

'PEDS' do BCPEDS

'PDRG' use center of gravity method

'PDSM' use PED smoothing method

'RTRK' remove all existing BCTRACK banks (in order to retrack)

'NONE' completely turn off crystal reconstruction

The default is to do everything that hasn't been done before, exept 'PDSM', which is incompatible with 'PDRG'. The center-of-gravity method (PDRG card) has been used since quite some years for the standard production. Therefore, from version 1.25/07 on it has become default.

To include any reconstruction routine one has to specify 'TRAK' plus the other keywords. To turn reconstruction of the crystal data completely off use the keyword 'NONE' and no other keyword in addition. To retrack, specify 'RTRK' along with all other keywords (i.e. partial reconstruction is not possible):
XTAL 'TRAK' 'RTRK' 'DECF' 'DECL' 'ALCE' 'CLST' 'PEDS' 'PDRG'
Please note that it is not recommended to use the PED smoothing and the center-of-gravity subroutines simultaneously. Therefore in your **XTAL** card you should specify either **'PRDG'** or **'PDSM'**. See the BCTRAK manual for more information!

3.10 Data Card GLOBal

Format XTAL par1 par2 ... par10
Default: RDT $\rightarrow$ DST : do everything
Default: DST $\rightarrow$ DST' : do nothing

This cards allows the user to selectively turn on parts of the global tracking software. The ordering of the parameters is irrelevant, as this card accepts keywords. The possible keywords are:

'TRAK' turn on global tracking.

'MTCH' perform matching between tracks and PED's.

'RTRK' to reinitialize global tracking for retracking. The other two cards are still obligatory for global tracking to actually retrack.

The default is to do everything/nothing. The given data cards are **not checked** for consistency. To perform global tracking, one has to specify 'TRAK' plus any other keywords. To turn off global tracking, use only the keyword 'NONE'.
GLOB 'TRAK' 'MTCH' 'RTRK'

3.11 Data Card CALI

Format: CALI par1 par2 ... par20
Default: none

This card controls reading of parameters from the database.To inhibit reading a set of constants from the database give the keyword directly followed by a $-$ sign, e.g. to veto use of BE constants specify 'BE-'.

'BE' Energy constants for the barrel.

'BG' default barrel gains.

'BL' Look–up table for the barrel (used until 1990).

'BP' Pedestal table for the barrel (used until 1991).

'BX' Bad or unused crystals in the barrel.

'BS' Ped smoothing.

'BR' 2282 to FERA ratios.

'JC' JDC input card parameters (ANGL, OPWC, BXT0, SY02, SYDF, DELZ, IAMP, ZOFF)

'JE' dE/dx–calibration of the JDC.

'JL' deviation from JDC stagger for each wire.

'JP' polynomial coefficients for dEdx probabilities.

'JS' stagger constants for the JDC wires.

'JW' Bad wires in the JDC.

'JT' oldJDC time-to-xy-position table (now Garfield table)

'JU' newJDC Garfield table, must be read from data base and is not accepted in the CALI card! (only experts should know how to override ...)

'JZ' z–calibration of the JDC.

'CB' magnetic field for each run.

'CG' 'global geometry' constants (JDC-Barrel angle, offset of downstream and upstream barrel half, xyz vertex used for neutal particles).

'CP' pressure for each run.

'CR' run quality. (good/bad)

'PW' PWC data (obsolete).

'LG' Light–pulser relative gains.

'LP' Light–pulser filter transmissions.

'LX' Crystals used for light–pulser.

3.12 Data Card BANK

The user may select which set of data banks should appear on the output device. At present, there are five different possibilities:

'ALLB' all banks are written to the output device. This is the default.

'RAWB' only the raw data banks will be written to unit 20. The RJDF/RJDG banks (unprocessed JDC data) will be kept if they were available as input.

'GLOB' all banks holding global tracking results plus the raw data banks, execpt RJDF/RJDG banks, are kept.

'DSTN' as above, plus TCTR bank, to allow running global tracking without retracking of the JDC. This correponds to the dst production of NEUTRAL data.

'DSTC' same as 'DSTN', but the crystal banks TBEN, TBCL and TBTK are saved as well. This corresponds to the dst production of CHARGED data.

Of course, only one of the above keywords an be selected.

3.13 Data Card DSTP

Data card to set up all defaults for the DST production mode There is one integer argument which can be any value between 0 and 4.

DSTP 0 old bctrak cuts for ECLUBC=20, EPEDBC=20.

DSTP 1 dst bctrak cuts: ECLUBC=4, EPEDBC=10 MeV. please note that this has become default since CBOFF version 1.27/00

DSTP 2 as before, and LTERM=LERR=LDBG=6 (logical units for terminal output, error messages and debug messages)

DSTP 3 as before, plus setup as BANK keyword 'DSTC'

DSTP 4 as DSTP 2, plus setup as BANK keyword 'DSTN'

3.14 Data Card STOP

This cards has no parameters. It tells FFREAD to stop reading cards.

4 File Definitions

To run an analysis job, one has to supply a set of file definitions for input and output files. These clearly depend on the operating system used. I will list here the necessary definitions for IBM-VM, the VAX-VMS and the decstations.

The logical unit numbers used by CBOFF are 4, 6, and 8 for output, unit 5 (or 99 on the decstations) for reading the data cards, unit 17 is reserved for reading calibration data from the CB data base. Data are read from unit 21, and are written to unit 20, but this I/O is via IOPACK on IBM. Units 81 and 82 are no longer used.

4.1 FILEDEF for IBM-VM

The following FILEDEF have to be given in the EXEC file before running CBOFF. For more information, see the help files for FILEDEF and SETUP available on VM.

```
'FILEDEF 04 DISK NORMALOG OUTPUT  A (LRECL 133 '
'FILEDEF 05 DISK CARD     INPUT   A '
'FILEDEF 06 DISK TERMINAL OUTPUT  A (LRECL 133 '
'FILEDEF 08 DISK ERRORMSG OUTPUT  A (LRECL 133 '
'FILEDEF 17 DISK CBBASE   DATA    G (XTENT 1000000 '
%'FILEDEF 81 DISK CARDJDC  INPUT   G '
%'FILEDEF 82 DISK GAIN     TABLE   G '
```

One also has to provide a FILEDEF for input data, and, if data card FZOUT is given, for output data. There are several choices :

- **data in eXchange format**
  The value for the parameter BLKSIZE normally is 23040, the CB default for data on tape using for ZEBRA a recordlength of 5760 words. It also can be 3600, for the default ZEBRA record length of 900 words.

  - Read from disk.
    The *filename filetype filemode* given below is an example, you may use a different name.

    ```
    'FILEDEF IOFILE21 DISK   INPUT    DATA  A ( BLKSIZE 23040'
    ```

  - Write to disk.
    This is identical to reading from disk. You might want to specify the additional parameter DISP MOD to add data to a file.

    ```
    'FILEDEF IOFILE20 DISK   OUTPUT   DATA  A ( BLKSIZE 23040 '
    ```

  - Read from tape(s), no output tape.
    You have to give a *tapenumber*, the type of tape *label* (`SL` or `NL`) and the *density* of the tape (6250 for round tape, 38K for IBM-cartridge, ??? for exabyte)

'SETUP TAPE 181 *tapenumber label density* NORING (END'
'FILEDEF IOFILE21 TAP1 *tapenumber* 1 (RECFM U BLKSIZE 23040 LRECL 23040'

For reading more than one tape one has to repeat the SETUP for every tape. Give the parameter END only for the last SETUP statement. You also have to add the following line to USINIT

```
CALL IOOPTN(21,'CON',IRC)
```

– Write to tape.
This is similar to reading a tape and has the same parameters, except for NORING is replaced by RING, the device number is now 182 and TAP2, and the unit number used to write data to is IOFILE20. Again give the parameter END only on the last setup, regarless of input or output tape.

'SETUP TAPE 182 *tapenumber label density* RING (END'
'FILEDEF IOFILE20 TAP2 *tapenumber* 1 (RECFM U BLKSIZE 23040 LRECL 23040'

If you are using different media, i.e. tape and cartridge, the device numbers have to be 289 and TAP9 for the cartridge.

4.2 FILEDEF for VAX-VMS

The following DEFINE statements have to be given in the .COM file before running CBOFF. For more information, see the help files for DEFINE, ALLOCATE and MOUNT available on VMS.

```
$ define for004 normalog.lis
$ define for006 terminal.lis
$ define for008 errormsg.lis
$ define for005 card.input
$ define for017 CB_OFF:cbbase.dat
%$ define for081 CB_OFF:cardjdc.input
%$ define for082 CB_OFF:gain.table
```

One also has to DEFINE the units for input data, and, if data card FZOUT is given, for output data. There are several choices :

- **data in eXchange format**
The value for the parameter BLKSIZE normally is 23040, the CB default for data on tape using for ZEBRA a recordlength of 5760 words. It also can be 3600, for the default ZEBRA record length of 900 words.
  - Read from disk

    ```
    $ define for021 input.dat
    ```

  - Write to disk

```
$ define for020 output.dat
```

– Using tapes
The description given here is valid only on a private VAX like VSXTAL. You mount your tape (or EXABYTE-tape) on the tapedrive *device* (normally MUA0:, MUB0:,..) and then give the commands

```
$ allocate device
$ mount/foreign/blocksize=23040/recordsize=23040 device
$ define for021 device
```

Reading several tapes is currently not supported by CBOFF, (but see Michael Doser about it if you really need to do this).

– Write to tape.
This is similar to reading a tape and has the same parameters, except for the Fortran unit now being 20.

```
$ allocate device
$ mount/foreign/blocksize=23040/recordsize=23040 device
$ define for020 device
```

4.3 FILEDEF for decstations

The following *setenv* statements have to be given in the script file before running CBOFF. For more information, see the man pages for SETENV available on UNIX (type 'man setenv').

```
$ setenv FORT04 normalog.lis
$ setenv FORT06 terminal.lis
$ setenv FORT08 errormsg.lis
$ setenv FORT99 card.input
$ setenv FORT17 $CBDAT/cbbase.dat
```

One also has to define the units for input data, and, if data card FZOUT is given, for output data. There are several choices :

- **data in eXchange format**
The value for the parameter BLKSIZE normally is 23040, the CB default for data on tape using for ZEBRA a recordlength of 5760 words. It also can be 3600, for the default ZEBRA record length of 900 words.

  – Read from disk

  ```
  $ setenv FORT21 input.dat
  ```

  – Write to disk

```
$ setenv FORT20 output.dat
```

– Using tapes (IBM cartridges or exabytes)
The description given here is valid only on our private decstations. There are cartridge drives on two decstations (cbdec5 and cbdec6), and exabyte drives on all decstations. Find a free drive for your kind of tape, load the tape, and then log onto the decstation which owns this drive.

You must define the input file via

```
$ setenv FORT21 /dev/nrmti h
```

where i is the number on the drive. Reading several tapes is supported by CBOFF (see the discussion in the section on USRUN).

– Write to tape.
This is similar to reading a tape and has the same parameters, except for the Fortran unit now being 20.

```
$ setenv FORT20 /dev/nrmti h
```

If you do not wish to keep the events that are written out, you can also write them into the null device by defining the output file to be

```
$ setenv FORT20 /dev/null
```

5 Compiling, linking and submitting the program

Most of the collaboration now uses CMZ for program maintenance, but some users also directly use patchy. I will describe the procedure to follow for both cases.

6 Retracking of data

Although this information is mostly available in other parts of this manual, collecting it in one spot may be useful.

Warning:
The event header is filled during production, and is not refilled during retracking unless global tracking is redone.

6.1 Which banks are at minimum necessary for a complete reconstruction

- Full reconstruction of the charged data via Locater requires either the RJDF/RJDG (in which case pulse recognition is redone) or the RJDC/RJDD banks (in which pulse recognition is not redone).
- Full reconstruction of the barrel information requires either the RBCL/RBCF or the TBEL/TBEF (pedestal subtracted ADC counts) banks.

- Reconstruction of the barrel information with a different energy threshold/split-off recognition requires at least the TBEN bank.
- Global tracking requires the presence of the following banks for track matching : LTBTK, LTBCL, LTCTR and LTCVT

Banks which you do not want to appear in the output stream may be assigned to the IXEVNT division in the routine USINIT. For example setting

```
IXTBEF = IXEVNT
IXTBEL = IXEVNT
```

will not write out the FERA and 2282 decoded banks.

6.2 How to find out which versions were used in the previous job

The version numbers of the software used to track data is always written into the master banks corresponding to the different parts of reconstruction and can be found at the following addresses :

CBOFF	IQ(LEVHD+1)
BCTRAK	IQ(LHTBC+1)
LOCATER	IQ(LHTJD+1)
GTRACK	IQ(LHTRK+1)

Aternatively, one can

```
CALL CBVERS(LLOG)
```

to obtain a printout of both the current versions of software, and whatever versions were used to process the data.

6.3 How to selectively reprocess an event

Selective reprocessing of only parts of an event is rarely necessary or even useful. If you have made changes to calibration constants or processing cuts (i.e. minimum PED energy), it is far more sensible to fully reprocess the event. Be warned that the global track banks will not be automatically updated, and that if you do enable global tracking as well, that matching of tracks to PEDs will not reflect the reprocessed data unless charged tracking is redone as well. Nevertheless, if you really want to reprocess only part of an event, you can selectively enable this via data cards (see below).

6.3.1 Locater

The following input card will force retracking of the chamber information:
CHAM 'TRAK' 'RTRK' 'GPWC' 'RAWS' 'PATT' 'CIRC' 'HELX' 'VERT'

6.3.2 Barrel

The following input card will force retracking of the crystal information:
XTAL 'TRAK' 'DECF' 'DECL' 'ALCE' 'CLST' 'PEDS' 'PDRG' 'RTRK'

6.3.3 Global tracking

The following input card will force rematching of the crystal information with the charged track information:
GLOB 'TRAK' 'MTCH' 'RTRK'

6.4 Trouble Shooting

Quite often people report inconsistencies between their retracking and the central dst production at CERN. I tried to summarize the most frequent errors in a questionary. items.

- What does the output and the error log file say? Most of the faults can be found in looking carefully at these files.
- Is the event really fully re-tracked? Check the input cards if they are set as described above.
- What are the cutoffs for the Crystal retracking? As it was decided by the collaboration We set ECLUBC=4 and EPEDBC=10 for the dst production. The default, however, is 20 MeV for both variables.
- Which version do we use? See section above to get the software version used to track the event.
- Which splitoff-recognintion do we use? The dst production program has calls to DOLBY-C and SMART. The relevant information is stored in the corresponding words in the TTKS bank. The event header word IEHDCB(12) is overwritten by the number of gammas accepted by Dolby-C.
- Which calibration is stored in the database used for retracking? The log file tells you date and method of the calibration for each run. Check if you got the one used at production time. For the improved locater (version 1.90 and higher) you should have also a new JDC calibration.
- more hints from the users are welcome!

7 User callable routines

7.1 subroutine ERRLOG

Author: Curtis Meyer
Creation Date: Oct, 1991
References:
Call Arguments: (*IERR, TEXT).
Common Blocks Used: None.
Subroutines Referenced: None.

This routine is called to log error messages. The first time it is called, the user should pass the value of IERR as 0. The routine will then return an error number for this error type, and remember the passed text, (up to the first 80 characters), as well as the run number and event number of the first and last occurance of this error. All subsequent calls with the given error code will be logged, but the text string will be ignored. At the any time, a report can be printed by calling the routine with the entry ERREPT.

Warning, it is necessary that the calling routine remembers the passed error code. The best way to do this is to define the error in the calling routine as follows:

```
INTEGER  IERBAD
SAVE     IERBAD
DATA     IERBAD /0/
...
CALL ERRLOG(IERBAD,' <ROUTINE> X FAILS')
```

8 Description of Subroutines

sorry, still not much documentation yet - volunteers are welcome!

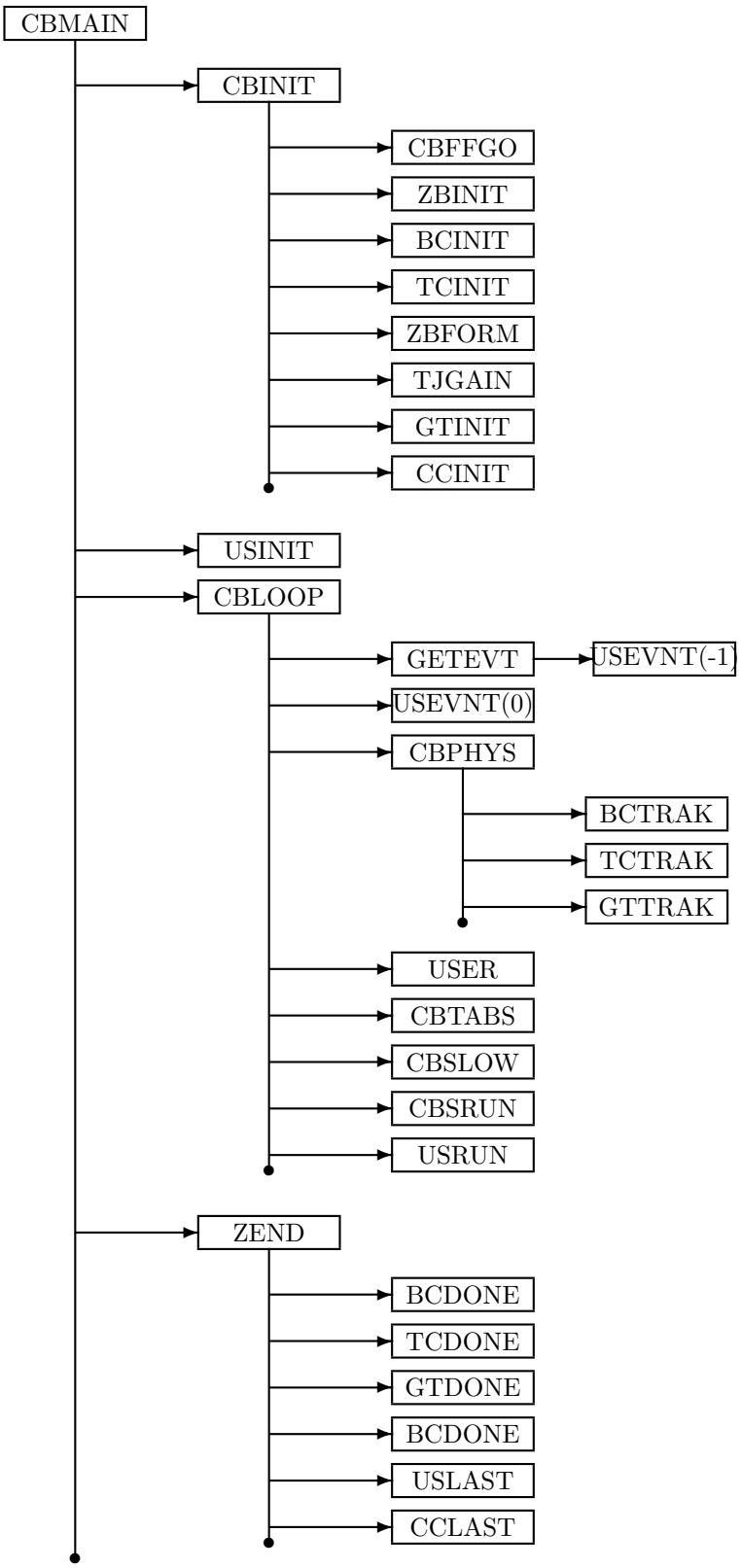

Figure 1: The calling sequence for the offline analysis code.

9 Description of Common Blocks

Although most of the data are kept in Zebra banks, some few are in common blocks. Other common blocks serve to comunicate between user routines and the offline code, and common blocks which are internal. The contents of common blocks, except internal ones, will be given. More information can be found in [3, 1].

9.1 Common blocks to access the ZEBRA banks

9.1.1 Common /CBXDIV/

Format

```
 COMMON /CBXDIV/IXCOMP,IXFRST,
&   IXEVHD,IXHRAW,IXHTJD,IXHTBC,IXHTRK,IXHFIT,IXHUSR,
&   IXRTRG,IXRSFT,IXRPWC,IXRSCL,IXRJDC,IXRJDF,IXRBCF,IXRBCL,
&   IXRBCR,IXRLAT,IXRLPR,IXRMON,IXRENO,IXRMCB,
&   IXDIST,IXSWIT,IXGEOM,IXVERT,IXKINE,IXPHYS,
&   IXMCIN,IXRTBF,IXRTBL,
&   IXTJDC,IXTPWC,IXTCHT,IXTCTK,IXTCTR,IXTCHX,IXTCVX,
&   IXTBEF,IXTBEL,IXTBEN,IXTBCL,IXTBTK,
&   IXTVTX,IXTTKS,IXKRES,
&   IXUSR1,IXUSR2,IXUSR3,IXUSR4,IXUSR5,
&   IXSPR1,IXSPR2,IXSPR3,IXSPR4,IXSPR5,
&   IXCALB,
&   IXTJCE,IXTJCT,IXTJT0,IXTJCZ,IXTJZ0,IXTJZL,IXTJST,IXTJCP,
&   IXTJSL,IXTJSR,IXTJSZ,
&   IXTJWR,IXTJRF,IXCJTF,IXCJNF,IXCJMX,
&   IXCLGE,IXCBEF,IXCBEL,IXCBPF,IXCBPL,IXCBTF,IXCBXF,IXCBXL,
&   IXCBCL,IXCBCF,IXCBRF,IXCBPC,
&   IXCCRN,IXCCBF,IXCCPR,IXCCKL,
&   IXWORK,IXEVNT,IXCONS,IXSAVE,
&   IXSTOR
 INTEGER        IXCOMP,IXFRST,
&   IXEVHD,IXHRAW,IXHTJD,IXHTBC,IXHTRK,IXHFIT,IXHUSR,
&   IXRTRG,IXRSFT,IXRPWC,IXRSCL,IXRJDC,IXRJDF,IXRBCF,IXRBCL,
&   IXRBCR,IXRLAT,IXRLPR,IXRMON,IXRENO,IXRMCB,
&   IXDIST,IXSWIT,IXGEOM,IXVERT,IXKINE,IXPHYS,
&   IXMCIN,IXRTBF,IXRTBL,
&   IXTJDC,IXTPWC,IXTCHT,IXTCTK,IXTCTR,IXTCHX,IXTCVX,
&   IXTBEF,IXTBEL,IXTBEN,IXTBCL,IXTBTK,
&   IXTVTX,IXTTKS,IXKRES,
```

```
&   IXUSR1,IXUSR2,IXUSR3,IXUSR4,IXUSR5,
&   IXSPR1,IXSPR2,IXSPR3,IXSPR4,IXSPR5,
&   IXCALB,
&   IXTJCE,IXTJCT,IXTJT0,IXTJCZ,IXTJZ0,IXTJZL,IXTJST,IXTJCP,
&   IXTJWR,IXTJRF,IXCJTF,IXCJNF,IXCJMX,
&   IXTJSL,IXTJSR,IXTJSZ,
&   IXCLGE,IXCBEF,IXCBEL,IXCBPF,IXCBPL,IXCBTF,IXCBXF,IXCBXL,
&   IXCBCL,IXCBCF,IXCBRF,IXCBPC,
&   IXCCRN,IXCCBF,IXCCPR,IXCCKL,
&   IXWORK,IXEVNT,IXCONS,IXSAVE,
&   IXSTOR
```

The CBXDIV common block provides the divisions for all ZEBRA banks used in the offline software. The division of a bank called WXYZ would be stored in the variable IXWXYZ. Include the division information with +SEQ,CBLINK. The /CBHEAD/ common is also given in this sequence.

9.1.2 Common /CBLINK/

Format

```
 COMMON /CBLINK/ LFRST,
&   LEVHD,LHRAW,LHTJD,LHTBC,LHTRK,LHFIT,LHUSR,
&   LRTRG,LRSFT,LRPWC,LRSCL,LRJDC,LRJDF,LRBCF,LRBCL,
&   LRBCR,LRLAT,LRLPR,LRMON,LRENO,LRMCB,
&   LDIST,LSWIT,LGEOM,LVERT,LKINE,LPHYS,LMCIN,LRTBF,LRTBL,
&   LTJDC,LTPWC,LTCHT,LTCTK,LTCTR,LTCHX,LTCVX,
&   LTBEF,LTBEL,LTBEN,LTBCL,LTBTK,
&   LTVTX,LTTKS,LKRES,
&   LUSR1,LUSR2,LUSR3,LUSR4,LUSR5,
&   LSPR1,LSPR2,LSPR3,LSPR4,LSPR5,
&   LCALB,
&   LTJCE,LTJCT,LTJT0,LTJCZ,LTJZ0,LTJZL,LTJST,LTJCP,
&   LTJSL,LTJSR,LTJSZ,
&   LTJWR,LTJRF,LCJTF,LCJNF,LCJMX,
&   LCLGE,LCBEF,LCBEL,LCBPF,LCBPL,LCBTF,LCBXF,LCBXL,
&   LCBCL,LCBCF,LCBRF,LCBPC,LCBBS,LCBGE,LCBGN,LCBGM,LCBGO,
&   LCCRN,LCCBF,LCCPR,LCCKL,
& LLAST
 INTEGER        LFRST,
&   LEVHD,LHRAW,LHTJD,LHTBC,LHTRK,LHFIT,LHUSR,
&   LRTRG,LRSFT,LRPWC,LRSCL,LRJDC,LRJDF,LRBCF,LRBCL,
```

```
&   LRBCR,LRLAT,LRLPR,LRMON,LRENO,LRMCB,
&   LDIST,LSWIT,LGEOM,LVERT,LKINE,LPHYS,LMCIN,LRTBF,LRTBL,
&   LTJDC,LTPWC,LTCHT,LTCTK,LTCTR,LTCHX,LTCVX,
&   LTBEF,LTBEL,LTBEN,LTBCL,LTBTK,
&   LTVTX,LTTKS,LKRES,
&   LUSR1,LUSR2,LUSR3,LUSR4,LUSR5,
&   LSPR1,LSPR2,LSPR3,LSPR4,LSPR5,
&   LCALB,
&   LTJCE,LTJCT,LTJT0,LTJCZ,LTJZ0,LTJZL,LTJST,LTJCP,
&   LTJSL,LTJSR,LTJSZ,
&   LTJWR,LTJRF,LCJTF,LCJNF,LCJMX,
&   LCLGE,LCBEF,LCBEL,LCBPF,LCBPL,LCBTF,LCBXF,LCBXL,
&   LCBCL,LCBCF,LCBRF,LCBPC,LCBBS,LCBGE,LCBGN,LCBGM,LCBGO,
&   LCCRN,LCCBF,LCCPR,LCCKL,
& LLAST
```

The CBLINK common block provides the addresses for all ZEBRA banks used in the offline software. The address of a bank called WXYZ would be stored in the variable LWXYZ. Include the address variables with +SEQ,CBLINK. The /CBHEAD/ common is also given in this sequence.

9.2 Common blocks containing data

9.2.1 Common /CBHEAD/

Format

```
INTEGER    MAXHD
PARAMETER (MAXHD=30)
COMMON/CBHEAD/NEHDCB,IEHDCB,NRHDCB,IRHDCB
INTEGER NEHDCB,IEHDCB(MAXHD),NRHDCB,IRHDCB(MAXHD)
```

This common is part of the Patchy +SEQ,CBLINK. However, it can be included also seperately with +SEQ,CBHEAD.
The meaning of each parameter in the IEHDCB array is given in table 1. The event header is filled during production, and is not refilled during retracking unless global tracking is redone. If an error is encountered during production, the event header is not filled completely (but the production error code is stored in element 20). It is important to realize that the data given in this header is available before the event has been completely unpacked by Zebra. On the -1 entry to USEVNT, the information given in this common block is available, although the event has not yet been unpacked, allowing a very quick decision on accepting or rejecting the event.

9.2.2 Common /BCENER/

Format

Variable	Contents
NEHDCB	number of words in array IEHDCB
IEHDCB	array of event header words
IEHDCB(1)	event type, values :
	0 - start run
	1 - real data
	2 - table event (lookup table)
	3 - slow control event
	4 - pedestal event (since Aug.90)
	5 - flasher event (since Aug.90)
	6 - cosmic event (since Aug.90)
	7 - Americium event (since Aug.90)
2	trigger
3	momentum of incoming $\bar{p}$ (MeV/c)
4	run number
5	event number
6	time (in hexadecimal (one byte each) : hours, minutes, seconds)
7	date (in hexadecimal : year (two bytes), month (one byte), day (one byte)
8	hardware setup, (bit pattern, –1 for GEANT).
9	number of charged tracks in JDC
10	number of unassociated PEDs (gamma)
11	number of long tracks (≥ 10 hits)
	(+ 100 * number of very long tracks (≥ 18 hits) ; from Cboff version 1.19/04 onwards)
12	number of unassociated PEDs with E> 20 MeV or
	number of 'good gammas' as seen by Dolby-C, if the event is
	tracked in central dst production at CERN.
13	total energy (all charged particles treated as pions)
	(from Cboff version 1.21/02 on, energy is negative if pile-up bit is set)
14	total momentum

Table 1: The definition of all elements of the IEHDCB array in the /CBHEAD/ common block.

15	coded word for vertex quality : number of long ($\geq$ 10 layers) vertex tracks + 100 $\times$ number of short vertex tracks + 10 000 if $\sum Q \neq 0$ for long vertex tracks + 20 000 if $\sum Q \neq 0$ for vertex tracks + 100 000 if $\sum Q \neq 0$ for long helix tracks + 200 000 if $\sum Q \neq 0$ for helix tracks
16	number of charged clusters*101 + number of neutral clusters
17	number of vertex tracks (short+long)
18	Total number of JDC hits in chamber. (+ 1000 $\times$ number of PWC_1 clusters + 100000 $\times$ number of PWC_2 clusters from Locater version 1.93/04 onwards).
19	Number of JDC hits NOT used in tracks
20	Error code from reconstruction : 1'000'000 + error code : BCTRAK error code 2'000'000 + error code : LOCATER error code 3'000'000 + error code : GTRACK error code
21	unused
22	number of PED13 clusters
NRHDCB	number of words in IRHDCB
IRHDCB	array of run header words, undefined yet

Table 2: The definition of all elements of the IEHDCB array in the /CBHEAD/ common block.

```
COMMON /BCENER/ ENERBC(60,26)
REAL   ENERBC
```

Include with +SEQ,BCENER.
Calibrated crystal energies in MeV.

9.3 Common blocks with Control Information

9.3.1 Common /CBCNTL/

Format

```
COMMON /CBCNTL/ ACCECB, ACCRCB, REDOCB, RTYPCB
LOGICAL ACCECB,ACCRCB,REDOCB
INTEGER RTYPCB
```

Include with +SEQ,CBCNTL.
Allow the user to stop further reconstruction of this event or run by setting ACCECB or ACCRCB to .FALSE., or to reanalyze the same event by setting REDOCB to .TRUE.. The variable RTYPCB allows the user to select a desired run type.

1 Data run, default.

2 Pedestal run.

4 Light pulser run.

8 Cosmic run.

For a combination of run types, the user should .OR. the codes together. ETYPCB selects the type(s) of events to accept for reconstruction. This can be set in the USINIT routine by setting ETYPCB(I) to .TRUE. to select for event type I.

00 Start of run, default .FALSE.

01 Real data, default .TRUE.

02 Table events, default .FALSE.

03 Slow control events, default .FALSE.

04 Pedestal events, default .FALSE.

05 Flasher events, default .FALSE.

06 Cosmic events, default .FALSE.

07 Americium events, default .FALSE.

08 undefined, default .FALSE.

09 undefined, default .FALSE.

10 undefined, default .FALSE.

9.3.2 Common /CBCHHD/

Format

```
      COMMON /CBCHHD/ HCH4(20),HHEAD,NBANK,ILOC(0:20),
     &                IDIV(0:10),HDIV(10)
      INTEGER         HCH4,HHEAD,NBANK,ILOC,IDIV,HDIV
```

Include with +SEQ,CBCHHD.

9.3.3 Common /CBDBCO/

Format

```
      COMMON /CBDBCO/
      LOGICAL
      INTEGER
```

Include with +SEQ,CBDBC0.
The variables in this common control input constants from the data base.

9.3.4 Common /CBHARD/

Format

```
       COMMON  / CBHARD  / MONTEC, RSV1, SILIQU, SILICE, KENCNT,
     >                     ENDCUP, ENDCDO, JDCTYP, VTXOUT, VTXINN,
     >                     GASLIQ, TGTTYP, MOMENT, FIELD
       INTEGER             MONTEC, RSV1, SILIQU, SILICE, KENCNT,
     >                     ENDCUP, ENDCDO, JDCTYP, VTXOUT, VTXINN,
     >                     GASLIQ, TGTTYP, MOMENT, FIELD
```

Include with +SEQ,CBHARD.
The meaning of the hardware variables is listed in table 3.

9.3.5 Common /CBKEYS/

Format

variable	detector	meaning
MONTEC	data type	0 = real 1 = monte carlo
RSV1	not yet used	
SILIQU	quad sili	0 = thin, 1 = thick, 2 = none
SILICE	center sili	0 = thin, 1 = thick, 2 = none
KENCNT	kens counter	0 = old, 1 = bochum, 2 = none
ENDCUP	endcap upstream	0 = none, 1 = bgo
ENDCDO	endcap downstream	0 = none, 1 = bgo
JDCTYP	jdc type	0 = old, 1 = new, 2 = none
VTXOUT	vertex outer	(0 = PWC, 1 = strip,
VTXINN	vertex inner	(2 = cerenk, 3 = none
GASLIQ	target phase	0 = liquid, 1 = gas
TGTTYP	target type	0 = LH, 1 = LD
MOMENT	pbar momentum	for real data:
		0 = 200 MeV/c, 1 = 105 MeV/c, 2 = 600 MeV/c
		3 = 1200 MeV/c, 4= 1940 MeV/c
		for Monte Carlo data:
		5 = at rest, 6 = in flight
FIELD	magnet field	0 = 0 T, 1 = 1.0 T, 2 = 1.5 T
		3 = other (only Monte Carlo data)

Table 3: The definition of all elements of the /CBHARD/ common block.

```
COMMON /CBKEYS/ FZINCB(4),FZOTCB(4),FZSECB(4),LOUTCB
INTEGER         FZINCB,FZOTCB,FZSECB
LOGICAL         LOUTCB
```

Include with +SEQ,CBKEYS.
Controls ZEBRA input and output. The most important flag for the user is LOUTCB which determines if a data summary tape is written.

9.3.6 Common /CBLDST/

Format

```
COMMON /CBLDST/ LDSTCB
LOGICAL LDSTCB
```

Include with +SEQ,CBLDST.
The logical LDSTCB, if set .TRUE., simulates the complete dst production setup. HOWEVER, THIS IS NOT YET FULLY IMPLEMENTED ...

9.3.7 Common /CBLOUT/

Format

```
COMMON /CBLOUT/ OUTSOR,OUTEOR,OUTSLC,OUTTAB,OUTERR
LOGICAL         OUTSOR,OUTEOR,OUTSLC,OUTTAB,OUTERR
```

Include with +SEQ,CBLOUT.
Controls which event types are written to data summary tape.

- OUTSOR Start of Run event, default .TRUE.
- OUTEOR End of Run event, default .TRUE.
- OUTSLC Slow control event, default .TRUE.
- OUTTAB Table event, default .TRUE.
- OUTERR Events failing analysis, default .TRUE.

9.3.8 Common /CBOUTB/

Format

```
COMMON /CBOUTB/ OUTBCB
INTEGER OUTBCB
```

Include with +SEQ,CBOUTB.
The variable OUTBCB selects the set of data banks which should appear on the output device (logical unit 20). The information is stored as an integer containing four characters. This old-fashioned hollerith string was introduced to ease the input via a data card. See section 3.12 for more details.

9.3.9 Common /CBREAD/

Format

```
      COMMON / CBREAD  /  RDRSV1, RDSLOW, RDLGHT, RDSOFT, RDBC,
     >                    RDJDC, RDPWC
      LOGICAL            RDRSV1, RDSLOW, RDLGHT, RDSOFT, RDBC,
     >                    RDJDC, RDPWC
```

Include with +SEQ,CBREAD.
The logicals are set true if being read out (see table 4).

variable	readout
RDRSV1	not yet used
RDSLOW	slow control
RDLGHT	lightpulser
RDSOFT	software trigger
RDBC	crystals
RDJDC	jdc
RDPWC	pwc

Table 4: logical variables belonging to the /CBHARD/ common block.

9.3.10 Common /CBSLCT/

Format

```
COMMON /CBSLCT/ CHRNCB,BEVTCB,EEVTCB,NSELCB,RSELCB(50)
LOGICAL CHRNCB,BEVTCB,EEVTCB
INTEGER NSELCB,RSELCB
```

Include with +SEQ,CBSLCT.
Allow the user to select certain runs, ranges of runs, events, or ranges of events for analysis.

9.3.11 Common /CBSTAT/

Format

```
COMMON /CBSTAT/ CBSTAT(30),CBTIMU
INTEGER         CBSTAT
REAL            CBTIMU
```

Include with +SEQ,CBSTAT.
The variables in this common contain reconstruction and run statitics (number of events of a given type, number reconstructed, ...).

9.3.12 Common /CBSTOP/

Format

```
COMMON /CBSTOP/ STOPCB
LOGICAL STOPCB
```

Include with +SEQ,CBSTOP.
Allow the user to stop the job by setting STOPCB to .TRUE..

9.3.13 Common /CBTIML/

Format

```
COMMON /CBTIML/ TFINCB,TMEVCB
REAL    TFINCB
INTEGER TMEVCB
```

Include with +SEQ,CBTIML.
Allow the user to limit the amount of time spent in analysis.

9.3.14 Common /CBTYPE/

Format

```
 COMMON /CBTYPE/ LTYPMC, LTYPDT, LTYPPD, LTYPLP, LTYPCS,
&                LTYPBK, LTYPSC, LTYPAM, LTYPTB
 LOGICAL         LTYPMC, LTYPDT, LTYPPD, LTYPLP, LTYPCS,
&                LTYPBK, LTYPSC, LTYPAM, LTYPTB
```

Include with +SEQ,CBTYPE.
These logicals are set true if the event is a Monte Carlo event (LTYPMC), real data (LTYPDT), pedestal event (LTYPPD), light pulser event (LTYPLP), cosmic event (LTYPCS), slow control event (LTYPSC), an americium event (LTYPAM) or a table event (LTYPTB).

9.3.15 Common /CBVRSN/

Format

```
      COMMON  / CBVRSN  / IVERCB,IVERLC,IVERGT,IVERCC,IVERCF,
     >                    IVERPH,IVERCD
      INTEGER             IVERCB,IVERLC,IVERGT,IVERCC,IVERCF,
     >                    IVERPH,IVERCD
```

Include with +SEQ,CBVRSN.
All versions of the software packages currently used are stored in the common / CBVRSN /. If the version is not available, the variable is set to zero. Currently, only the first four variables are filled.

- IVERCB version of CBOFF
- IVERLC version of LOCATER
- IVERBC version of BCTRAK (defined in /BCSTAT/)
- IVERGT version of GTRACK
- IVERCC version of CCDBCB
- IVERCF version of CBKFIT
- IVERPH version of PHYSAC
- IVERCD version of CBDROP

9.3.16 Common /CBVTRK/

Format

```
      COMMON  / CBVTRK  / ITRKCB,ITRKLC,ITRKBC,ITRKGT,ICBGEA,IGEANT
      INTEGER             ITRKCB,ITRKLC,ITRKBC,ITRKGT,ICBGEA,IGEANT
```

Include with +SEQ,CBVTRK.
All versions of the software packages used at tracking time are stored in the common / CBVTRK /. If the version is not available, the variable is set to zero. For monte carlo data the last two variables hold the versions for CBGEANT and GEANT with which the data were produced.

- ITRKCB version of CBOFF
- ITRKLC version of LOCATER
- ITRKBC version of BCTRAK

- ITRKGT version of GTRACK
- ICBGEA version of CBGEANT if monte carlo data
- IGEANT version of GEANT if monte carlo data

9.3.17 Common /CBUNIT/

Format

```
      COMMON /CBUNIT/LDST,LRDT,LLOG,LPRNT,LTERM,LDBG,LERR,LNORM,LHST,
     &               LTIME,LTIML,LJCAL,LJGAIN,LJTOUT,LJGOUT
      INTEGER LRDT,LDST,LLOG,LPRNT,LTERM,LDBG,LERR,LNORM,LHST
      INTEGER LTIME,LTIML,LJCAL,LJGAIN,LJTOUT,LJGOUT
```

Include with +SEQ,CBUNIT.

- LDST Logical unit 20 for outputting the DST.
- LRDT Logical unit 21 for reading the data tape.
- LLOG Logical unit 4, the log file.
- LPRNT Logical unit 4.
- LTERM Logical unit 99 for inputting the card file.
- LDBG Logical unit 7, used for debug output.
- LERR Logical unit 8, error file. Messages should also go to LLOG.
- LNORM Logical unit 9. Available to the user.
- LHST Logical unit 10 for user histograms.
- LTIME Logical unit 30.
- LTIML Logical unit 31.
- LJCAL Logical unit 81. Alternate Locater card file. (obsolete)
- LJGAIN Logical unit 82, gain file for locater. (obsolete)
- LJTOUT
- LJGOUT

Essential to run a job are units where input is expected: 21 and 99. The connection of a logical unit to the file is operating system dependent.

9.4 User common blocks

9.4.1 Comon /USKEYS/

Format

```
      COMMON/ USKEYS / KEYSUS(50)
      REAL KEYSUS
      INTEGER IKEYUS(50)
      EQUIVALENCE (KEYSUS(1),IKEYUS(1))
```

Include with +SEQ,USKEYS.
Allow users to input their own parameters.

9.5 The value of π

For uniform values of π throughout the analysis, use +SEQ,PI2PI to obtain the following parameters in a subroutine.

```
      REAL       PI,PI2,PIHLF
      PARAMETER (PI    = 3.141592653589793)
      PARAMETER (PI2   = 6.283185307179586)
      PARAMETER (PIHLF = 1.570796326794896)
C
      DOUBLE PRECISION DPI,DPI2,DPIHLF
      PARAMETER (DPI   = 3.141592653589793D0)
      PARAMETER (DPI2  = 6.283185307179586D0)
      PARAMETER (DPIHLF= 1.570796326794896D0)
```

10 Description of data banks

All data are accessible as data banks managed by ZEBRA. The banks and the links (or pointers) to the data banks are kept in two common blocks, namely /CBBANK/ for the data banks, and /CBLINK/ for the links. These two common blocks should be included in subroutines with Patchy : +SEQ,CBLINK. The variable name of a link always has 5 characters : a L plus the name of the bank (4 characters). The banks are:

data banks containing raw data; (for a description of the contents see below)

RTRG Trigger information. The trigger word is also stored in the event header.

RSCL Scaler information

RSFT Software trigger information.

RPWC PWC data.

RJDC/RJDD Processed JDC data

RJDF/RJDG Unprocessed JDC data (full flash ADC information)

RBCF FERA ADC data read out via HSM

RBCR FERA ADC data read out via big read module

RBCL LeCroy 2280 ADC system data

RLPR Lightpulser data bank

RLAT Bank containing data form Latches

RMCB Monte-Carlo information

banks created during reconstruction of PWC/JDC data (charged tracking);
(these banks are described in [3])

TPWC Hit information for the PWC

TJTP temporary data bank used in tracking

TCHT reconstructed hit information

TCTK first level reconstruction of tracks, subbanks TCSG

TCTR tracking results, subbanks TCTX

TCHX fit points, subbanks TCHP.

TCVX vertex information, subbanks TCVT and TCVP.

banks created during reconstruction of Crystal data; for a description of the contents see [1]

TBEN Crystal energies in MeV

TBCL cluster information

TBTK Particle Energy Deposit (PED) information

banks created by the global reconstruction; for a description of the contents see [4]

TVTX Vertex information

TTKS Track banks

KRES Kinematic fit results

10.1 Description of bank contents

To access data in the banks use the array Q for real, IQ for integer (and for bits[2]) or LQ for LINKS. For example, if the first data word of the RJDC bank is integer it has to be accessed as IQ(LRJDC+1).

10.1.1 The RTRG bank

The information contained in the trigger bank is described below. The different trigger codes are described in the trigger description files xxxxx.dsc .

Link: LRTRG in COMMON/CBLINK/

For the interpretation of the above information, we show two short code segments. The first example demonstrates how to extract which of the silicium counters fired:

```
* PLU5 settings for Apr93 and Oct93!!
+SEQ,CBLINK.
*
      LOGICAL UPPERL,UPPERR,DOWNL,DOWNR,CENTER
      INTEGER SILI4              ! used to decode quad sili counter info
*
         DOWNR = .FALSE.
         DOWNL = .FALSE.
         UPPERR = .FALSE.
```

[2]bits means that the data are packed. There is usually no sign bit. Bit 1 is the least significant bit.

Offset	Bits	Content
+1	1-32	length of trigger bank (I*4)
+2	1-8	Trigger PLU 5 (includes pile-up bit)
	17-24	Trigger Trigger register input (same pattern as bit 1-8)
+3	1-8	Trigger PLU 7 (timeout/force/x/face/pwc/jdc/monitor/accept)
	17-24	Trigger PLU 6 (face/pwc/jdc/triggercode/accept)
+4	1-16	Trigger counter 1 (time active trigger 125 nsec clock)
	17-24	Trigger PLU 8 (not used)
+5	17-32	Trigger counter 2 (time comp gate open 250 nsec clock)
+6	1-16	JDC Malu 1
	17-32	Face multiplicity
+7	1-16	JDC Malu 3
	17-32	JDC Malu 2
+8	1-16	JDC Malu 5
	17-32	JDC Malu 4
+9	1-16	JDC Malu 7
	17-32	JDC Malu 6
+10	1-16	JDC Malu 9
	17-32	JDC Malu 8
+11	1-16	JDC Malu 11
	17-32	JDC Malu 10
+12	1-4	inverted JDC multiplicity layer 1
	5-8	inverted JDC multiplicity layer 2
	9-12	inverted JDC multiplicity layer 3
	13-16	inverted JDC multiplicity layer 4
	17-24	JDC Malu 12
+13	17-20	inverted JDC multiplicity layer 5
	21-24	inverted JDC multiplicity layer 6

Table 5: The data in the trigger bank.

```
         UPPERL = .FALSE.
         CENTER = JBIT(IQ(LRTRG+2),17)
*
         SILI4  = JBYT(IQ(LRTRG+2),18,3)
         IF(SILI4.EQ.1)  THEN
             DOWNR = .TRUE.
         ELSEIF(SILI4.EQ.2) THEN
             DOWNL = .TRUE.
         ELSEIF(SILI4.EQ.3) THEN
             UPPERR = .TRUE.
         ELSEIF(SILI4.EQ.4) THEN
             UPPERL = .TRUE.
         ENDIF
*
```

In the second example we access the PWC and JDC multiplicity:

```
+SEQ,CBLINK.

*   FADC values for JDC layers 1..4, 5..6 and for PWC

      INTEGER JFADC1, JFADC2, JPFADC

*  multiplicity values for face, PWC1, PWC2, JDC (layer1..6)

      INTEGER JFACE1
      INTEGER JPMUL1, JPMUL2
      INTEGER JFMUL1, JFMUL2, JFMUL3, JFMUL4, JFMUL5, JFMUL6

      INTEGER  JBYT
      EXTERNAL JBYT

*  test pointer to bank

      IF (LRTRG.LE.0) STOP 'No trigger bank !'

* get the JDC multiplicity ; FADC1 covers the first four layers,
*                            FADC2 covers the last two layers

      JFADC1 = JBYT(IQ(LRTRG+12),  1, 16)
      JFADC2 = JBYT(IQ(LRTRG+13), 17, 16)
```

```
*
*            none of the following should be larger than 9 ...
*
      JFMUL1 = JBYT(NOT(JFADC1),  1, 4)
      JFMUL2 = JBYT(NOT(JFADC1),  5, 4)
      JFMUL3 = JBYT(NOT(JFADC1),  9, 4)
      JFMUL4 = JBYT(NOT(JFADC1), 13, 4)
      JFMUL5 = JBYT(NOT(JFADC2),  1, 4)
      JFMUL6 = JBYT(NOT(JFADC2),  5, 4)
*
* now the PWC multiplicity
*
      JPFADC = JBYT(IQ(LRTRG+20), 1, 16)
*
      JPMUL1 = MIN( MAX( 0, JBYT(NOT(JPFADC),1,4) ) , 15)
      JPMUL2 = MIN( MAX( 0, JBYT(NOT(JPFADC),5,4) ) , 15)
*
* finally the FACE multiplicity
*
      JFACE1 = JBYT(IQ(LRTRG+6),17,16)
```

10.1.2 The RPWC bank

Raw data from the PWC, format as defined by the LeCroy PCOS electronics. The inner PWC has wire numbers in the range 0–119, the outer 128–307.

Link: LRPWC in COMMON/CBLINK/

Offset	Type	Content
+1	Integer	Number of I*2 words in the bank, this word is counted as 2 I*2
+2	Bits	bits 1-16: wire number of first wire bits 17-32: number of wires fired
+3	Bits	bits 1-16: wire number of third hit bits 17-32: wire number second hit
⋮	⋮	⋮

10.1.3 The RJDC/RJDD banks

The **RJDC/RJDD** 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 identify

what the data format is. The data as shown in in table 6 is the standard format. For more information consult reference [3]. The contents of this data bank can be printed using the routine RJRJDC.

Link: LRJDC in COMMON/CBLINK/

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 6: 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 resolution is 200ps per count, in which case the conversion constant is 5.
- *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 Conv*ersion word.
- *A Left* is the amplitude from the *left* or $+z$ preamplifier.
- *A Right* is the amplitude from the *right* or $-z$ preamplifier.

10.1.4 The RJDF bank

Unprocessed JDC flash ADC data. This bank can be looked at using the subroutine TWRJDF, or printed using the subroutine RJRJDF. There are actually two **RJDF** banks, the first contans the outer layers of the JDC and the second contains the inner layers.
Link: LRJDF in the /CBLINK/ common, IRJDF=LQ(LRJDF).

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 7: The data format in the **RJDF** data bank.

10.1.5 The RBCF bank

FERA ADC system raw data read out via HSM in a very compact format. These data can also be found in the bank TBEF (see ref. [1]) in less compact format.
Link: LRBCF in COMMON/CBLINK/

Offset	Type	Content
+1	Integer	Number of I*2 words in the bank, including these two
+2	Bits	Bits 1-16: number of FERA words of data (I*2 words Bits 17-32: memory mode, possible values (should normally be H0) H0 - zero supressed H1 - not zero supressed, packed H2 - not zero supressed, not packed
+3	Bits	first two I*2 words of FERA data, see FERA manual
⋮	⋮	⋮

10.1.6 The RBCR bank

FERA data read out via big read module. Contents not yet defined.

10.1.7 The RBCL bank

LeCroy 2280 ADC system raw data in a very compact format. The same data can be found in the bank TBEL (see ref [1]) in an easily accessible way.
Link: LRBCL in COMMON/CBLINK/

Offset	Type	Content
+1	Integer	Number of I*2 words in the bank, including these two
+2	Bits	Bits 1-16: number of LeCroy 2280 words of data (I*2 words Bits 17-32: memory mode, possible values (should normally be L2) L0 - pedestal subtracted, zero suppressed packed data L1 - pedestal subtracted, not zero suppressed, not packed data L2 - pedestal not subtracted, not zero suppressed, not packed data
+3	Bits	first two I*2 words of LeCroy data (first word in bits 1...16, second in bits 17...32
⋮	⋮	⋮

10.1.8 The RSFT bank

Information from Software trigger. This bank contains only short integers (16 bit), except the first word. It is therefore easier to first unpack this bank into an array ISOFT (sufficiently dimensioned !) and then look at the contents of this array. You could use the subroutine UPKCH from Cernlib (M427). Define an additional array IPAR(5), set IPAR(1)=16, IPAR(2..5)=0 and the do

```
CALL UPKCH(IQ(LRSFT+2),ISOFT,IQ(LRSFT+1)-2,IPAR))
```

The contents of the array ISOFT is then given in table 8.

10.1.9 The RSCL bank

Data from the trigger scalers. All scalers are 32 bit long, except the *light* and *ameri* scalers.
Link: LRSCL in COMMON/CBLINK/

Word	Content
ISOFT(1)	number of concurrent triggers (maximum 4)
2	trigger 1 , +1 = accepted , 0 = rejected event
3	trigger 2 , +1 = accepted , 0 = rejected event
4	trigger 3 , +1 = accepted , 0 = rejected event
5	trigger 4 , +1 = accepted , 0 = rejected event
6...9	trigger 1...4 energy window in MeV
10...13	trigger 1...4 central energy of window in MeV
14...17	trigger 1...4 cluster mask
18...21	trigger 1...4 π° mask
22...25	trigger 1...4 η mask
26	window for π°, η mass
27	minimum total energy all triggers
28	total energy found
29	N, the number of crystals with energy
30	energy
31	channel (hardware numbering sceme)
$\vdots$	*repeat energy and channel N times*
$30 + 2 \times N$	M, the number of neutrals found
$30 + 2 \times N$+1	position $((\theta - 1) \times 64 + (\phi - 1))$
$30 + 2 \times N$+2	energy in MeV
$30 + 2 \times N$+3	number of crystal in neutral cluster
$\vdots$	*repeat M times*
$31 + 2 \times N + 3 \times M$	L, the number of invariant masses $L = \frac{M(M-1)}{M} + \pi^\circ$–supressed combinations)
$31 + 2 \times N + 3 \times M$+1	invariant mass
$\vdots$	*repeat L times*
$32 + 2 \times N + 3 \times M + L$	number of π° found
$32 + 2 \times N + 3 \times M + L$+1	number of η found

Table 8: Data from the software trigger after unpacking into the array ISOFT (see text).

Offset	Type	Content
+1	Integer	Number of I*2 words in the bank, including these two
+2	Integer	Sili Left Down (lsb only)
+3	Integer	Sili Right Down (lsb only)
+4	Integer	Sili Left Up (lsb only)
+5	Integer	Sili Right Up (lsb only)
+6	Integer	beam (lsb only)
+7	Integer	beam (lsb only)
+8	Integer	central veto (lsb only)
+9	Integer	testpulser ($100\mu sec$, lsb only))
+10	Integer	light 1 (bits $1 \cdots 16$) , americium 1 (bits $17 \cdots 32$)
+11	Integer	light 2 (bits $1 \cdots 16$) , americium 2 (bits $17 \cdots 32$)
+12	Integer	PLU 5/ output 6
+13	Integer	PLU 1/ output 6
+14	Integer	PLU 2/ output 5
+15	Integer	PLU 2/ output 6
+16	Integer	PLU 3/ output 5
+17	Integer	PLU 3/ output 6
+18	Integer	PLU 4/ output 4
+19	Integer	PLU 4/ output 5
+20	Integer	PLU 5/ output 7
+21	Integer	PLU 5/ output 8
+22	Integer	PLU 6/ output 6
+23	Integer	PLU 7/ output 6
+24	Integer	PLU 7/ output 7
+25	Integer	events accepted
+26	Integer	fast resets total
+27	Integer	sys resets total

Table 9: Scaler information (see text).

10.1.10 The RLAT bank

This bank gives the pattern of the firing latches as three 32 bit words per θ ($1 \cdots 26$), for a total of 79 words.

Offset	Content	
+1		
	32 17	16 1
+2	LATCH_θ^1	LATCH_θ^2
+3	LATCH_θ^3	LATCH_θ^4
+4	LATCH_θ^5	LATCH_θ^6
$\vdots$	$\times$ 26	

Table 10: Structure of the Latch bank; there are three 32 bit words per θ, where θ runs from 1 to 26 in the θ ordering $18 \cdots 26$, $10 \cdots 17$ and $1 \cdots 9$.

The three 32 bit words for each θ are combined into one 64 bit pattern (two 32 bit words):
$\text{LATCH}_\theta^A = 65536 \cdot \text{LATCH}_\theta^2 + \text{LATCH}_\theta^3$ and
$\text{LATCH}_\theta^B = 65536 \cdot \text{LATCH}_\theta^5 + \text{LATCH}_\theta^6$,
which correspond to the firing pattern in ϕ ($1 \cdots 64$) for a given θ ($1 \cdots 26$).

The correspondance between ϕ and bit position is then:

$\phi =$ $1 \cdots 16$ corresponds to bits $17 \cdots 32$ in LATCH_θ^A
$\phi =$ $17 \cdots 32$ corresponds to bits $1 \cdots 16$ in LATCH_θ^A
$\phi =$ $33 \cdots 48$ corresponds to bits $17 \cdots 32$ in LATCH_θ^B
$\phi =$ $49 \cdots 64$ corresponds to bits $1 \cdots 16$ in LATCH_θ^B

10.1.11 The RMCB bank

Exists only for Monte Carlo data
This is a bank supporting GEANT banks containing information on the generated event. The three banks are supported at offsets -1 (VERT) , -2 (KINE) (both GEANT banks) and -4 (MBEN). The contents of the three banks are described in Tables 11, 12 and 13.

10.1.12 The MCIN banks

Exists only for Monte Carlo data.
The MCIN bank is a bank supporting CBGEANT banks containing information on the generated event (the informational banks). Full information on these banks is given in [6]. There are seven linearly linked banks (MCIN, PHYS, KINE, VERT, GEOM, SWIT and DIST) which are accessible via the links LMCIN, LPHYS, ...

Offset	Content
Data words in bank	
IQ(LQ(LRMDB-1)+1)	Number of vertices in event
Content of sub-banks	
Q(LQ(LQ(LRMCB-1)-IVERT)+1)	V_X
Q(LQ(LQ(LRMCB-1)-IVERT)+2)	V_Y
Q(LQ(LQ(LRMCB-1)-IVERT)+3)	V_Z
Q(LQ(LQ(LRMCB-1)-IVERT)+4)	Time of Flight
IQ(LQ(LQ(LRMCB-1)-IVERT)+5)	beam number of vertex origin
IQ(LQ(LQ(LRMCB-1)-IVERT)+6)	target number of vertex origin
IQ(LQ(LQ(LRMCB-1)-IVERT)+7)	number of particles from vertex
IQ(LQ(LQ(LRMCB-1)-IVERT)+8)	particle number in KINE bank

Table 11: Structure of the VERT bank (see routine GSVERT and GEANT manual, page KINE 199).

Offset	Content
Data words in bank	
IQ(LQ(LRMCB-2)+1)	Number of particles in event
Content of sub-banks	
Q(LQ(LQ(LRMCB-2)-IPART)+1)	P_X
Q(LQ(LQ(LRMCB-2)-IPART)+2)	P_Y
Q(LQ(LQ(LRMCB-2)-IPART)+3)	P_Z
Q(LQ(LQ(LRMCB-2)-IPART)+4)	Energy
Q(LQ(LQ(LRMCB-2)-IPART)+5)	particle ID
Q(LQ(LQ(LRMCB-2)-IPART)+6)	number of vertex origin
Q(LQ(LQ(LRMCB-2)-IPART)+7)	number of vertices from this particle
Q(LQ(LQ(LRMCB-2)-IPART)+8)	vertex number in VERT bank

Table 12: Structure of the KINE bank (see routine GSKINE and GEANT manual, pages KINE 191 and 192).

Offset	Content
Data words in bank	
IQ(LRMCB+1)	Number of tracks in event
Content of sub-banks	
IQ(LQ(LRMCB-ITRACK)+0)	NX := Number of crystals hit by track
IQ(LQ(LRMCB-ITRACK)+1)	Total energy deposited by track
IQ(LQ(LRMCB-ITRACK)+2)	Crystal index of first crystal
IQ(LQ(LRMCB-ITRACK)+3)	Energy deposited in first crystal
IQ(LQ(LRMCB-ITRACK)+2*NX)	Crystal index of NX^{th} crystal
IQ(LQ(LRMCB-ITRACK)+2*NX+1)	Energy deposited in NX^{th} crystal

Table 13: Structure of the MBEN bank (see routines BCMBEN in CBGEANT).

10.2 Table events

There are three table events at the beginning of each run. They are decoded in routine CBTABS, but the banks they contain are not exported to the rest of CBOFF. Each of the three events contains essentially one single bank; the first two events contain the RTBF, resp. RTBL banks which give the mapping between channel numbers and crystal θ and ϕ positions, while the third table event contains the DAMO bank, which documents the set-up of the software trigger and of the event builder. The DAMO bank consists of 108 words; its' structure is shown in Table 14.

11 Event display on the decstations

An interactive event display is available on the decstations. As above, the minimum environment variables that must be set are FORT21 and FORT99. The present version only supports X-terminals, and is started by typing 'cbdisp' (if you need a version for Falco terminals, I can perhaps arrange that). Parts of it are still buggy, so if you find a problem, you can earn undying fame by fixing it. The event display setups are taken by default from /cboff/pro/bin/cbdisp_setups.kumac, but you can use your own setup file by declaring it from within cbdisp via

```
DEFAULT_SETUP DEFNAM=your_file.kumac
```

Offset	Content
+1	Status
+2	Error
+3	Run number
+4	Run type
+5	Momentum
+6	Trigger
+7	PWC_1 minimum (bits $1 \cdots 16$), maximum (bits $17 \cdots 32$)
+8	PWC_2 minimum (bits $1 \cdots 16$), maximum (bits $17 \cdots 32$)
+9	JDC minimum
+10	JDC maximum
+11	FACE minimum
+12	FACE maximum
+13	Trigger status (bits $1 \cdots 16$), PWC status (bits $17 \cdots 32$)
+14	JDC status (bits $1 \cdots 16$), BC status (bits $17 \cdots 32$)
+15	Slow control status (bits $1 \cdots 16$), Flasher status (bits $17 \cdots 32$)
+16	JDC2 status (bits $1 \cdots 16$), EVB status (bits $17 \cdots 32$)
+17	Event number
+18	Flasher number (0 = no flasher active, 1 or 2 for barrel half)
+19	Filter number (binary combination)
+20	Flasher sequence ($1 \cdots$ n)
+21	Filter status
+22 – +31	Command line from run control (char)
+32 – +41	Name of trigger preset file (char)
+42 – +51	Name of hardware trigger (char)
+52 – +61	Name of JDC multiplicity file (char)
+62	Slow control data module information
+63	Slow control data module information
+64	FORCE options (bits $1 \cdots 16$), software trigger info (bits $17 \cdots 32$)
+65,+66	Trigger accepted (four 16 bit integers packed into two 32 bit integers)
+67,+68	Energy window (four 16 bit integers packed into two 32 bit integers)
+69,+70	Energy cms (four 16 bit integers packed into two 32 bit integers)
+71,+72	Cluster mask (four 16 bit integers packed into two 32 bit integers)
+73,+74	π^0 mask (four 16 bit integers packed into two 32 bit integers)
+75,+76	η mask (four 16 bit integers packed into two 32 bit integers)
+77	π^0 mass (bits $1 \cdots 16$), energy cut (bits $17 \cdots 32$)
+78	Vax strobe (bits $1 \cdots 16$), vax pad (bits $17 \cdots 32$)
+79	Vax command
+80	Vax status (two 16 bit words)
+81	Vme strobe (bits $1 \cdots 16$), vme pad (bits $17 \cdots 32$)
+82	Vme command
+83	Vme status (two 16 bit words)
+84 – +108	Message (char)

Table 14: Structure of the DAMO bank (see routine CBTABS in CBOFF).

References

[1] F.–H. Heinsius, T. Kiel, Crystal data reconstruction software, CB–note 92

[2] C. A. Meyer, User Guide for LOCATER, CB–note 123

[3] C. A. Meyer, Chamber Reconstruction Software, CB–note 93

[4] M. Burchell, Global Tracking Particle Bank Structure, CB–note 118

[5] F.–H. Heinsius, Calibration Constants Database Software, CB–note 122

[6] R. Bossingham, CBGEANT 4.06/02 manual, CB–note 169

[7] R. Brun *et.al.*, Format Free Input Processing, Cern Computer Center Program Library Long Write Up I302

[8] R. Brun, J. Zoll, ZEBRA user guide (provisional), Cern Computer Center Program Library Long Write Up Q100

[9] J. Zoll, ZEBRA reference manual FZ

[10] J. Zoll, ZEBRA reference manual MZ