# Update of database Server and the databases

P. de Simone, F. Pelucchi

December 14, 1995

## 1 Revision of the database server for the KLOE offline

Since beginning of december 1995, the master machine for the database server has been moved from AXKLO1 to HPSERVER. The slave machines are now : axklo1 vaxlnf vxrm70 axpik1 axple1 axrm05.

The principe of the server is already described in the KLOE note No 123 ([1])

### 1.1 Configuration of the server

The database server is using a configuration file called hepdb.names which resides on each top directory of the database area.

In this file the server programs find the information about the databases and the machines to be contacted. Two HEPDB programs are running for the master database server, CDSERV and CDMOVE, and only CDSERV is running for each slave database server. Another program DBMENU makes the interface between the user and the database server to ask for updates.

The principe is that the CDMOVE program scans the directories called "queue" to see if there are files to transfer, in both direction, master to slaves or slave to master. If existing, it transfers the files in the "todo" directories of the machines and deletes them from where it took them. Those files are put there by a user request (via DBMENU program) in case of a slave machine or by the CDSERV program in case of the master machine. The CDSERV programs processes locally the update files put in the "todo" directory by CDMOVE or directly by the user request on the master machine, and in case of the master server redistributes the files with a unic stamp into the slave queue directories.

The hepdb.names file is defined in the case of the master machine (HPSERVER), as follows:

```
:nick.config
        :list.ct rt gt ge ! list of all the database served
        :log./u/kloe/kloedb/log
        :queue./u/kloe/kloedb/todo
        :todo./u/kloe/kloedb/todo
        :save./u/kloe/kloedb/save
        :bad./u/kloe/kloedb/bad
```

```
        :loglevel.3
        :wakeup.30
        :servers.axklo1 axpik1 vxrm70 axple1 vaxlnf

:nick.ct
        :file./u/kloe/kloedb/database/rtcal.dbs
        :desc.Calibration Database for Test Beam 1994
        :servers.axklo1 vaxlnf vxrm70 axpik1 axple1 axrm05
        :write.patrizia frede kloe

:nick.rt
        :file./u/kloe/kloedb/database/rtrnc.dbs
        :desc.Run Condition Database for Test Beam 1994
        :servers.axklo1 vaxlnf vxrm70 axpik1 axple1 axrm05
        :write.patrizia frede kloe
:nick.gt
        :file./u/kloe/kloedb/database/rtgeo.dbs
        :desc.Geometry Database for Test Beam 1994
        :servers.axklo1 vxrm70 axpik1 axrm05
        :write.patrizia frede kloe

:nick.ge
        :file./u/kloe/kloedb/database/geomg.dbs
        :desc.Geometry and magnet fieldDatabase.
        :servers.axklo1 vaxlnf vxrm70 axpik1 axple1 axrm05
        :write.patrizia frede kloe

:nick.axklo1
        :userid.KLOEDB
        :node.AXKLO1
        :poll.yes
        :receive.yes
        :remoteq.DISK$MF:[KLOEDB.TOHPSERV]
        :localq./u/kloe/kloedb/toaxklo1
        :queue.DISK$MF:[KLOEDB.TODO]

:nick.vaxlnf
        :userid.kloedb
        :node.vxlnfa
        :poll.no
        :receive.yes
        :localq./u/kloe/kloedb/tovaxlnf
        :remoteq.DISK$MF:[KLOEDB.TOAXLNF1]

:nick.vxrm70
        :userid.kloe_offline
        :node.vxrm70.roma1
        :poll.no
```

```
        :receive.NO
        :localq./u/kloe/kloedb/tovxrm70
        :queue.[kloe.kloedb.todo]
        :remoteq.[kloe.kloedb.toaxklo1]

:nick.axpik1
        :userid.kloedb
        :node.axpik1.pi.infn.it
        :poll.no
        :receive.NO
        :localq./u/kloe/kloedb/toaxpik1
        :remoteq.disk$mf:[kloedb.toaxklo1]

:nick.axple1
        :userid.kloe
        :node.axple1.le
        :poll.no
        :receive.yes
        :localq./u/kloe/kloedb/toaxple1
        :queue.users2:[kloe.kloedb.todo]
        :remoteq.users2:[kloe.kloedb.toaxklo1]

:nick.axrm05
        :userid.kloedb
        :node.axrm05
        :poll.no
        :receive.yes
        :localq./u/kloe/kloedb/toaxrm05
        :remoteq.disk$mf:[kloedb.toaxklo1]
```

The list of the machines given in the :servers tag in the :nick.config is used by CDMOVE, and the description of the machines is done in the tag :nick.machine. The list given in the :servers. tag in each database description (:nick.ge for instance) is used by CDSERV, the program which processes locally the updates in the databases. This permits to define a different list of remote systems for each database.

In the description of the machines (slaves in the case of the hepdb.names file on the master), the tag :queue points to the remote directory where CDMOVE has to put the update file once the master has processed it (by default it points to a directory called "todo"), the tag :remoteq points to the remote queue directory where CDMOVE has to look into to see if there is a request of update from the slave (by default points to the "queue" directory) and the tag :localq points to the local directory where CDSERV put the updates to be distributed to the slaves and where CDMOVE looks into to know if there are some updates to transfer to the slave (by default point to "tomachine" directory).

The flag :write allows the responsible to restrict the write access to certains account. The equivalent flag exists for the read access but is not used.

The :receive tag tells CDMOVE if this remote machine is allowed to send request to the master for an update, the :poll tag tells CDMOVE to contact the remote system regarless of whether there are any updates pending for the remote node in question, otherwise it contacts the remote machine only when the master has some update to send.

On each slave machine there is also a hepdb.names file to describe the databases and the master system.

### 1.2 Remarks on CDSERV and CDMOVE program

Due to the big size of certain database banks (specially the drift chamber wires description), the program CDSERV has to be recompiled with a bigger size for the ZEBRA array (PARAMETER (MAXGCB = 8000000) is enough for the ZEBRA common "COMMON /GCBANK/ FENCE(22),LQ(MAXGCB"). To do this the cdserv.cra file is available in the CERN_ROOT:[SRC.CRA] directory on VMS machines or $CERN_ROOT/src/cra on UNIX machines. For the HEPDB.CAR, it is better to take the last version from the asisftp machine at CERN, because the HEPDB code is still in development. The hpserver and the vxlnfa machines are registered to do so.

On Unix machine the CDSERV code has also been changed to have the log file written each time that an update file is processed by closing it instead of keeping it open during all the time that CDSERV is running. To launch CDSERV on hpserver a shell script is existing called *cdstart* in /u/kloe/kloedb directory, on the VMS machine, a command file is also existing CDSERV.COM in disk$mf:[kloedb.server]

For the CDMOVE program, it can be used as it is in the cern area CERN_ROOT:[EXE]CDMOVE.EXE for VMS but it has been changed on the HP machine because of problems in length file names. All the program to rebuild CDSERV and CDMOVE are in /u/kloe/kloedb/cern directory.

From the last version of HEPDB.CAR, a new CDLIB.olb library must be done to have a consistent set of HEPDB routines. The cdlib.cra is in the same area as for CDSERV.

## 2 Database changes

In this section we describe the recent changes of the KLOE Database, that concern mainly the banks that contain informations about the drift chamber cell response parametrization.

### 2.1 New database directory structures

First we moved the six DPS$n$ ($n = 1, ..., 6$) banks from the Geometry Database, defined by the index `'GE'`, to the Calibration Database, defined by the index `'CT'`. Here we show the new directory structures of each Database.
Calibration Database directory structure:

```
//CDCT
      /HELP
      /DICTIONARY
      /CALIBRATION
                  /DRCH_LT
                  /DRCH_MT
                          /PARAM
                          /DCD_SIG
                  /DRCH_ST
                  /CALO_LT
                  /CALO_MT
```

```
                  /CALO_ST
                  /TRIG_LT
                  /TRIG_MT
                  /TRIG_ST
```

Geometry Database directory structure:

```
//CDGE
      /HELP
      /DICTIONARY
      /GEOMETRY
               /DRCH
                    /DIM
                    /POSITION
               /CALO
                    /DIM
                    /POSITION
               /MAG_YOKE
               /INTER_ZONE
                          /QUAD
                          /BEAM_PIPE
               /VXCH
                    /DIM
                    /POSITION
      /MAGNET_FIELD
                   /GEN_MAP
                   /DRCH
                   /QUAD
                   /CALO
                   /VXCH
      /MATERIALS
```

Run Condition Database directory structure:

```
//CDRT
      /HELP
      /DICTIONARY
      /RUN_COND
               /TRIGGER_ST
               /TRIGGER_LT
               /TRIGGER_MT
               /DRCH_LT
               /DRCH_MT
               /DRCH_ST
               /CALO_LT
               /CALO_MT
               /CALO_ST
               /BEAM_LT
               /BEAM_MT
               /BEAM_ST
```

In the two new subdirectories

```
//CDCT/CALIBRATION/DRCH_MT/PARAM/
//CDCT/CALIBRATION/DRCH_MT/DCD_SIG/
```

there are the banks that contain the coefficients for the fine and the raw cell response parametrizations, and the respective sigmas to assign to the reconstructed drift distances.

## 2.2 New banks

⊙ **Fine cell response parametrizations banks**

We changed the content and the structure of the DPS$n$ ($n = 1, ..., 6$) banks.

The most recent parametrization of the cell response has been obtained from the data collected with the small prototype 03.

We remind the reader that the banks of the fine parametrization are 6, one bank per *reference cell*, and for each reference cell the time space relations are parametrized in 36 $\phi$-sections of the cells. The new parametrization describe the time-space relations with a Chebychev polynomial of the fourth order (instead of the third order as it was before), it follows that the parameters for each $\phi$-section of the cells are 5 plus one effective radius.

Finally, the format of the new DPS$n$ banks correspond to a vector with dimension 219, where the first 3 components are the header of the vector, while the others contain the parameters as shown by the following scheme:

| Vector Index | Word Type | Description |
|---|---|---|
| DPSHDS | R | header size = 3 |
| DPS1DI | R | number of parameters: NP = 6 |
| DPS2DI | R | number of $\phi$-sections per reference cell: I = 36 |
| DPS2DI+1 | R | 1st parameter for the 1st $\phi$-section |
| DPS2DI+2 | R | 2nd parameter for the 1st $\phi$-section |
| DPS2DI+3 | R | 3rd parameter for the 1st $\phi$-section |
| DPS2DI+4 | R | 4th parameter for the 1st $\phi$-section |
| DPS2DI+5 | R | 5th parameter for the 1st $\phi$-section |
| DPS2DI+6 | R | effective radius for the 1st $\phi$-section |
| . | R | . |
| . | R | . |
| . | R | . |
| DPS2DI+J+NP*(I-1) | R | Jth parameter (J=1,NP) for the Ith $\phi$-section |
| . | R | . |
| . | R | . |
| . | R | . |

⊙ **Raw cell response parametrization bank**

In the directory

```
//CDCT/CALIBRATION/DRCH_MT/PARAM/
```

together with the DPS$n$ ($n = 1, ..., 6$) banks, there is one new small bank, DPRW, that contains the parabola coefficients which provide the raw parametrization of the time space relations.

The format of the DPRW bank is a vector with dimension 5:

| Vector Index | Word Type | Description |
|---|---|---|
| DRWHDS | R | header size = 2 |
| DRW1DI | R | number of parameters: NP = 3 |
| DRW1DI+1 | R | drift time at $x = 0$ [ns] |
| DRW1DI+2 | R | $x$ coefficient [ns/cm] |
| DRW1DI+3 | R | $x^2$ coefficient [ns/cm$^2$] |

All the DPS$n$ banks and the DPRW bank have the `KEY14` (see [2]) different from zero; this key permit us to select between the parametrizations for the $2 \times 2$ cm$^2$ cells (`KEY14`= 2) and the parametrizations for the $3 \times 3$ cm$^2$ cells (`KEY14`= 3).

⊙ **Resolution curve bank**

The bank DCSF, placed in the directory

`//CDCT/CALIBRATION/DRCH_MT/DCD_SIG/`

contains the coefficients that parametrize the resolution curve as a function of the drift distance. Also this bank has a format of a vector with dimension 5:

| Vector Index | Word Type | Description |
|---|---|---|
| DSFHDS | R | header size = 2 |
| DSF1DI | R | number of parameters: NP = 3 |
| DSF1DI+1 | R | parameter $K$ in [cm] that describe the diffusive contribution $\sigma_{diff} = \sqrt{2Kr_d}$ |
| DSF1DI+2 | R | $\lambda/2$ in [cm] that is the half average distance between two clusters |
| DSF1DI+3 | R | $\sigma_{t_d}$ in [ns] that is the experimental resolution on drift time |

⊙ **Raw sigmas table bank**

In the same directory of the DCSF bank there is also the last new bank, DCSR, that contains a table of the raw errors on the reconstructed drift distances as a function of the drift distance itself. The format of the DCSR bank is a vector with dimension 17:

| Vector Index | Word Type | Description |
|---|---|---|
| DSRHDS | R | header size = 2 |
| DSR1DI | R | number of components: NP = 15 |
| . | R | |
| . | R | |
| DSR1DI+(2N+1) | R | lower value in [cm] of the drift distance interval within which is valid $\sigma_{raw}(2N)$ |
| DSR1DI+(2N) | R | $\sigma_{raw}(2N)$ in [$\mu$m] with N= 2, ..., 8 |
| DSR1DI+(2N-1) | R | upper value in [cm] of the drift distance interval within which is valid $\sigma_{raw}(2N)$ |
| . | R | |
| . | R | |

The DCSR bank has the `KEY14` different from zero:

- `KEY14 = 2` selects the table ($\sigma_{raw}$ .vs. $r_d$) for the $2 \times 2\text{cm}^2$ cells
- `KEY14 = 3` selects the table ($\sigma_{raw}$ .vs. $r_d$) for the $3 \times 3\text{cm}^2$ cells

Finally we give the present scheme of the database directories and banks.
Calibration Database directory structure:

```
//CDCT
      /HELP
      /DICTIONARY
      /CALIBRATION
                  /DRCH_LT
                  /DRCH_MT
                          /PARAM
                                - DPS1, DPS2, DPS3, DPS4, DPS5, DPS6 :
                                  one bank per reference cell, each
                                  bank containing 5 Chebychev
                                  parameters plus an effective radius
                                  obtained from the response
                                  parametrizations of 36 phi-sections
                                  inside the reference cells
                                - DPRW : contains the parabola
                                  coefficients that give a raw
```

```
                  parametrization of the time space
                  relations
          /DCD_SIG
                - DCSF : contains the coefficients that
                  parametrize the resolution curve as a
                  function of the drift distance
                - DCSR : contains a raw sigma table as
                  a function of the drift distance
  /DRCH_ST
               - DTOS : T0s for the Drift Chamber TDCS
                  Not yet filled
  /CALO_LT
          - CATL : Attenuation Lengths in the Calorimeter
            cells.
            Used in the reconstruction program
            with fixed value for the MC.
            Existed also with values computed from the
            test beam data at PSI 1994
          - CVEL : Velocity of the light in the Calorimeter
            fiber.
            Used in the reconstruction program
            with fixed value for the MC.
            Existed also with values computed from the
            test beam data at PSI 1994
          - CPMG : PMs Gains for the Calorimeter
            Filled from test beam data of PSI 1994
  /CALO_MT
          - CNPE : Number of Photo-electron for the
            Calorimeter PMs.
            Used in the reconstruction program
            with fixed value for the MC.
            Existed also with values computed from the
            test beam data at PSI 1994
  /CALO_ST
          - CMIP : Minimum Ionising Peaks bank of the
            Calorimeter fibers.
            Filled from test beam data of PSI 1994
          - CTWA : Time Walk Correction for the
            Calorimeter TDC
            Filled from test beam data of PSI 1994
          - CZET
          - CTOF
          - CTOS : T0 values of the Calorimter TDC
            Filled from test beam data of PSI 1994
  /TRIG_LT
  /TRIG_MT
  /TRIG_ST
```

Geometry Database directory structure:

```
//CDGE
      /HELP
      /DICTIONARY
      /GEOMETRY
               /DRCH
                    /DIM
                             - DGGI : Drift General Geometry Information.
                               Filled from MC data blocks
                               Used in the reconstruction program
                    /POSITION
                             - DCWC : Drift Chamber Wires Coordonates
                               Computed from MC data blocks
                               Used in the reconstruction program
                             - DCLD : Drift Chamber Layer description
                               Computed from MC data blocks
                               Used in the reconstruction program
               /CALO
                    /DIM
                             - CCDI : Calo Cell Dimensions
                               Filled from MC data blocks
                             - CMDI : Calo Module Dimensions
                               Filled from MC data blocks
                             - CGGI : Calo General Geometry Information
                               Filled from MC data blocks
                    /POSITION
                             - CCCP : Calorimeter Cell Center Positions
                               Used in the reconstruction program
                               Filled with nominal values
                             - CCEP : Calo. Cell Edge Positions
                               Not yet filled
                             - CMEP : Calo Module Edge Position
                               Not yet filled
                             - CMCP : Calo Module Center Position
                               Not yet filled
               /MAG_YOKE
               /INTER_ZONE
                          /QUAD
                              - QDES : Quadrupoles Description.
                                Filled with MC data blocks in April 95
                              - QIAD : Quadrupoles Instrumentations and
                                Anticoincidence Detector Description
                                Filled with MC data blocks in April 95
                          /BEAM_PIPE
                              - BPDE : Beam Pipe description
                                Filled with MC data blocks in April 95
               /VXCH
                    /DIM
                              - VCDE : Vertex Chamber Description
```

```
                                  Filled with MC data blocks in April 95
                    /POSITION
                                - VCCO : Vertex chamber CellS Coordonates
                                  Filled with nominal values
      /MAGNET_FIELD
                   /GEN_MAP
                   /DRCH
                   /QUAD
                   /CALO
                   /VXCH
      /MATERIALS
                               - MATE banks, one bank per materials
```

Run Condition Database directory structure:

```
//CDRT
      /HELP
      /DICTIONARY
      /RUN_COND
               /TRIGGER_ST
               /TRIGGER_LT
               /TRIGGER_MT
               /DRCH_LT
               /DRCH_MT
                       - DTHR
               /DRCH_ST
                       - DMAP : Electronics-Detector Map file
                         Not yet filled
                       - DDCH : Dead Channels
                         Not yet filled
                       - DDWR : Dead Wires
                         Not yet filled
                       - DHGV : High Voltage
                         Not yet filled
                       - DGAZ : Gaz mixture
                         Not yet filled
               /CALO_LT
               /CALO_MT
               /CALO_ST
                       - CMAP : Electronics-Detector Map file
                         Filled with test beam data of PSI 1994 or
                         Cosmic Ray Stand test
                       - CPED : Pedestals of the Calorimeter ADC
                         Filled with test beam data of PSI 1994 or
                         Cosmic Ray Stand test
                       - CMOF : Map offline.
                         Used in reconstruction programs.
                       - CDCH : ADC Dead Channels
                         Filled with test beam data of PSI 1994 or
```

```
                    Cosmic Ray Stand test data.
                  - CHVM : Calorimeter High Voltages
                    Not yet filled
                  - CZET : Calorimeter z-calibration banks
                    Filled with test beam data of PSI 1994 or
                    Cosmic Ray Stand test data.
          /BEAM_LT
          /BEAM_MT
          /BEAM_ST
```

### 2.3 Key9 or RUNID

Each bank is defined in the database by a number of keys (see [2]), for example we have already saw that the informations that we retrieve from the DPS$n$ ($n = 1, ..., 6$) banks are different if we select Key14 $= 2$ or Key14 $= 3$. Among the other keys, the Key9 called also RUNID is peculiar; his meaning and the way how to know his value has to be explained in detail.

The key RUNID is a packed number telling the origin and what the informations stored will be used for. In the include file RUNTYP.INC the numerical values related to the origin are listed as Experiment Type Identifiers, while the numerical values related to the use are listed as Run Type Identifiers: for example if the informations stored in one bank has been obtained from offline simulations and then will be used for KLOE physics studies, the correspondent parameters are EXPTYP = EXOFSI, and RUNTYP = RUKPHY.

In the Analysis-Control KLOE software enviroment the parameters EXPTYP and RUNTYP are stored in the LRID (Logical Record IDentifier) YBOS bank, and are packed with other informations as shown in the following scheme:

```
Bit Specification for LRID Run Type:-
=====================================
Bits  0-11      Run Type
Bits 12-15      Run Modifier
Bits 16-19      Partition Number
Bits 20-23      Spare
Bits 24-31      Experiment Type
```

The user that is developing code inside the Analysis-Control enviroment and wants to retrive informations from one database bank, can take the unpacked parameters EXPTYP and RUNTYP from the common JOBSTA. Then the two previous parameters has to be packed in the key RUNID. To be consistent with Analysis-Control it has been decided to pack EXPTYP and RUNTYP with the same scheme of the LRID bank. Here we give an example of packing:

```
C
$$INCLUDE 'K$INC:JOBSTA.INC'
C
        INTEGER RUNID
C
        RUNID = EXPTYP
        RUNID = ISHFT(RUNID,24)
```

```
        RUNID = IOR(RUNID,RUNTYP)
```

In the following we show how to unpack the key RUNID:

```
C      Local declarations
C      ==================
       INTEGER RUNID
       INTEGER REFEXP,REFRUN
       PARAMETER(REFEXP = '00000FFF'X)
       PARAMETER(REFRUN = 'FF000000'X)
C
       EXPTYP = IAND(REFEXP,RUNID)
       RUNTYP = IAND(REFRUN,RUNID)
```

# 3 RTDB package

To access the interface code written for the database you must first "setup" the package called RTDB, severals option are existing on AXKLO1 depending if you want to point to the general KLOE offline databases or to the Lecce drift chamber prototype databases (3rd option of the setup), and if you want to have all the logicals defined for the package or only the one useful for a normal user (4th option of the setup). The different possiblities are :

**$ setup rtdb** : setup the production version of RTDB package (the only existing for the moment), points to the general KLOE offline database and defines only the logical useful for a normal user. this command is equivalent to

- $ setup rtdb production
- or $ setup rtdb production normal

**$ setup rtdb production normal all** : same as before but with all the possible logicals, only useful for the librarian.

**$ setup rtdb production prototype** : points to the Lecce databases with the logicals for "normal" user

**$ setup rtdb production prototype all** : points to the Lecce databases with all possible logicals.

The first setup is executed by the offline setup. A detailed description of the routines existing to create the intermediate FZ file for the database or to retrieve values from the databases or the FZ files is given in the note No 99 [2].

On Unix machine, for the moment only two options is existing,

**setup rtdb production** to have the "normal user" environment variable defined or

**setup -e rtdb production** to have all the environment variable defined.

### 3.1 Program to manipulate the databases : DBMENU

The program DBMENU in RTDB$WORK on VMS or dbmenu in $RTDB_WORK on Unix is an interactive program to manipulate objects in the databases (Update, Modification of keys or data, Delete) or to dump values of objects coming from the database or from the intermediate FZ file used to fill the database. It does not allow the user to modify the structure of the database itself. To do so, if necessary, a independant fortran program must be written.

Here is given the description of how to use the program for each option available. The text between ** are comments about the program, they are not part of the dbmenu display.

The main menu is the following :

```
--------------------------------------------------------------------------------
              Selection Commands to manipulate database objects

  1  [UPDATE]         UpDate a database
  2  [REPLACE]        Replace an object in a database file
  3  [DELETE]         Delete an object in a database file
  4  [RESTORE]        Restore an object
  5  [PRSTATU]        information print
  6  [MODIFY]         Modify keys of an object in a database file
  7  [QUIT]           Exit the menu

KLHDBU>
```

The 1st option UPDATE must be used for each normal update of the databases. The update will be send only if the requestor is allowed to update the database which means on the VMS system to have the KLOE$DB rights identifier, or on UNIX system to be in the list in the hepdb.names file. If this option is choosen, then you have to answer to the following questions :

```
KLHDBU> 1
Give the file name [ ]: **FZ file name of the user where
                        he put his updated values**
Object number:   1 DataBase: GE   Bank: CMDI
Source: OFF  key 14 =       0 Trigger Number =       0
Run Number for the Validity Start=         1
Run Number for the  Validity End=     999999
** it is the display of the FZ header **
Do you want to update this object ? (Y/N) [No]:
Give the pathname for this object [//CDGE/GEOMETRY/CALO/DIM]:
** type return, the value given between [] is retrieve by the code **
Give Run number for the start of validity of this object [1]:
** the default value is read from the FZ file header, you can change
 it if necessary **
Give the Run Type [0]:
** see previous section for the run and experimental
 type, the default value is read from the FZ file header, you can change
```

```
 it if necessary **
Give the value for the key 14 (if any) [0]:
** the default value is read from the FZ file header, you can change
 it if necessary **
Do you want to select with key 14 ? (Y/N) [No]:
** this key is not always used to select a object in a database even
 if it has some values, for instance, the calorimeter group use this
 key in the calibration database to give the run number with which the
 calibration was done, it is just an information and is not
 taken into account when the calibration object is retrieve, instead
 the drift chamber group used this key to distinguish between the
 parametrization values for the 2*2 and 3*3 cells and so this key
 must be taken into account when the object is retrieve **
```

After this question, the program looks into the database to see if a previous version of the object is already existing or not, if it does not find a previous one then the message "`No old version of object found`" appears and it puts the object in the database with the given start of validity and the end of validity equal to 999999. The definition of "previous" object in this case is an object with the same bank name key, the same source key (offline or online) and if selected, the same key 14, it does not concern the instant of validity. If a previous object exists with the same validity range the message "`existing object with the same validity range`" appears then the question

```
Do you really want to update this object (last chance) ?  (Y/N):
```

If you say yes, the object is put in the database with a end of validity equal to 999999 and the previous version is not touched, only the serial number will increase and can make the difference between this two objects. If you answer no, nothing is done.

If a previous object is found with a start of validity less than the new one you gave, no message is displayed but a new validity range is computed for the previous object which keeps its original start of validity and gets a new end of validity equal to the start of validity value - 1 of the new object put into the database. Then this new object takes a end of validity equal to 999999.

If a previous object is found with a greater start of validity than the one you gave for the new object, the following message is printed :

```
Start Validity Conflicts

This update is going to change validity range of existing data ...

Do you really want to update this object (last chance) ?  (Y/N):
```

Saying yes, the program goes on and tries to insert the new object between the already existing objects if any (insert in term of validity range). It means it tries to compute all the starts and ends of validity for the objects concerned by the change.

For instance, let suppose that for a given bank, several sets of values were already computed, one set valid from run 1 to run 4, another one from 5 to 9 and a last one from run 15 to 999999. A new set of values is computed with the run 7, then the program will insert this object with a start of validity of 7 and a end of validity equal to the end of validity of the object it replaces, so finally after the update, for this bank, 4 objects are existing, one with the validity range from 1 to 4, then from 5 to 6, then from 7 to 9 and from 15 to 999999. If now a new set is done from the run 10, the program does not find a object valid for the run 10, so it just inserts the new object with a start of validity equal to 10 and ask for the end of validity you want, with the condition that it should be less than 15.

All this gymnastic is done to avoid some overlapping in term of validity range which can invalidate the database. If two objects for a same bank exist with some instant of validity in common, by default the more recent is taken which is not always the good one, to retrieve the previous one, a selection on the serial number must be done.

**Second option REPLACE** This option permits to replace the whole data part of a bank without changing the corresponding keys for the database, specially the validity range of the bank. For this you must also create a FZ file as for the update, then answer the questions :

Give the file name [ ]: ** file name of your FZ file **

Object number: 1 DataBase: GE Bank: CMDI Source: OFF key 14 = 0 Trigger Number = 0 Run Number for the Validity Start= 1 Run Number for the Validity End= 999999 Do you want to Replace by this object ? (Y/N) [No]: y

Give the pathname for this object [//CDGE/GEOMETRY/CALO/DIM]:

Give the Run Number for the start of validity of object to replace [1]: ** concern the object to be replaced **

Give the Run Number for the End of validity of object to replace [999999]: ** concern the object to be replaced **

Give the source of the object to be replaced (OFF or ON) [OFF]: ** concern the object to be replaced **

Give the Run Type [0]: ** concern the object to be replaced **

Give the value for the key 14 (if any) [0]: ** concern the object to be replaced **

If the object is found, no other message is printed but the object is deleted and replaced with the new one which keeps the same keys as the previous one.

**third option DELETE** This option permits to delete a object from the database file if you are sure that this object will never be useful again. To do this you must answer the following questions :

Give the 2 characters Database Index [ ]: ** ct, rt, ge or gt **

Give the 4 characters Bank Identifier [ ]: ** see the list in the previous section **

Give the source of the object to be deleted (OFF or ON) [OFF]:

Give the Run Number for the start of validity of this object [0]:

Give the Run Number for the end of validity of this object (Max=999999) [999999]

Give the Run Type [0]: ** see previous section for the run and experimental type **

Give the value for the key 14 (if any) [0]:

If no object corresponding to those keys is found you have the message " No object match found" otherwise nothing is printed and the deletion is done.

**fourth option RESTORE** You have the possiblity to restore a object that you have deleted before, to do so you have to answer to the same serie of questions as the one in the DELETE option.

**fifth option PRSTATU** is the one to print all kind of information from the databases or from an intermediate user FZ file. Choosing this option you obtain the following menu :

```
KLHDBU> 5

--------------------------------------------------------------------------------
                   Selection Command to Print on the screen

  1  [LSTDATIM]        Print last Update date and time
  2  [VALHIS]          Print the validity history of an object
  3  [LISTOB]          Lists object in directory
  4  [DBDUMP]          Dump banks in database
  5  [DUMPBK]          Dump data structure of an FZ file

  7  [QUIT]            Exit the menu

KLHDBU> 1
Give the ABSOLUT pathname of the directory (must begin by //) [ ]:
** you must type the all path name of the directory between
quotes, see the name list in the previous section **

KLHDBU> 2
Give the 2 characters Database Index [ ]: ** ct, rt, gt, or ge **
Give the 4 characters Bank Identifier [ ]: ** see list in the previous
 section **
Give the source of the object (OFF or ON) [OFF]:
Give the Run ID [0]: ** see previous section for the run
and experimental  type **
Give the instant of validity [999999]:
Do you want to select with key 14 ? (Y/N) [No]:

KLHDBU> 3
Give the 4 characters Bank Identifier (blank for all) [ ]:
Give the pathname of the Directory [ ]: ** give the whole path name
 of the directory between quotes, see the list in the previous section **
Give the serial number (0 for all) [0]:

KLHDBU> 4
Give the 2 characters Database Index [ ]: ** ct, rt, gt or ge **
Give the 4 characters Bank Identifier [ ]: ** see list in the previous
 section **
Give the source of the object (OFF or ON) [OFF]:
Give the Run ID [0]: ** see previous section for the run
```

```
and experimental  type **
Give the instant of validity [999999]:
Do you want to select with key 14 ? (Y/N) [No]:

KLHDBU> 5
Give the file name [ ]:
Do you want to print only header?  (Y/N) [No]:
```

**Sixth option MODIFY** to modify existing keys of objects without changing the data.

```
KLHDBU> 6
Give the 2 characters Database Index [ ]: ** ct, rt,gt or ge **
Give the 4 characters Bank Identifier [ ]:  ** see list below **
Give the source of the object (OFF or ON) [OFF]:
Give the Run Number for the start of validity of this object [0]:
Give the Run Number for the end of validity of this object (Max=999999)
[999999]:
Give the Run ID [0]: ** see previous section for the run
and experimental type **
Give the value for the key 14 [0]:
Do you really want to modify the keys of this object? (Y/N)[No]: yes
Give the NEW source of the object  (OFF or ON) : ** concerns the NEW keys **
Give the NEW Run Number for the start of validity of this object [0]:
Give the NEW Run Number for the end of  validity of this object (Max=999999):
Give NEW the Run Type [0]:
Give the NEW key 14 [0]:
```

### 3.2 New routines to retrieve values from the databases

Some routines have been written to retrieve values from the geometry and calibration databases. They will be used in the Monte-Carlo but can also be used for real data.

```
C
********************************************************************************
      INTEGER FUNCTION GTDBMC(RUNNUM,EXPTYP,RUNTYP)
********************************************************************************
C
C     Description:
C     ============
C     Function which retrieves the MC data blocks from the databases
C     and fill the COMMON /MCDIMS/ and COMMON /DIGITI/
C     Initialize also the database area in the zebra common,
C     must not be used in A_C environment which initialise already
C     this area.
C
C     Input Paramater
C     ===============
C     INTEGER RUNNUM   ! Run number used as instant of validity for
```

```
C                       ! the object to be retrieve from databases
C     INTEGER EXPTYP    ! Experiment type used as in A_C
C     INTEGER RUNTYP    ! Run type used as in A_C
C                       ! These 2 last information are used to fill
C                       ! the key of the database concerning the trigger
C                       ! type.
C
********************************************************************************
      INTEGER FUNCTION DBCGGI(IDXID,SOURCC,KEY14,RUNNUM,NTRIG)
********************************************************************************
C
C      Description
C      ===========
C      Get the values for the Calorimeter General Geometry Information
C
C      Input Parameter:
C      ===============
C      CHARACTER*2 IDXID  ! Character Index of the database (ge)
C      CHARACTER*4 SOURCC ! Offline source
C      INTEGER     KEY14  ! key 14 for the db
C      INTEGER     RUNNUM ! instant of validity (run number)
C      INTEGER     NTRIG  ! Packed word for the run and experiment type
C                         ! as done in A_C
C
C      Output parameter: none , the commons COMMON /MCDIMS/ and COMMON /DIGITI/
C      ================  are filled (see description in Geanfi)
********************************************************************************
      INTEGER FUNCTION DBCCDI(IDXID,SOURCC,KEY14,RUNNUM,NTRIG)
********************************************************************************
C
C      Description
C      ===========
C      Get the values for the Calorimeter readout Cells DImension
C
C      Input Parameter:
C      ===============
C      CHARACTER*2 IDXID  ! Character Index of the database (ge)
C      CHARACTER*4 SOURCC ! Offline source
C      INTEGER     KEY14  ! key 14 for the db
C      INTEGER     RUNNUM ! instant of validity (run number)
C      INTEGER     NTRIG  ! Packed word for the run and experiment type
C                         ! as done in A_C
C
C      Output parameter: none , the commons COMMON /MCDIMS/ and COMMON /DIGITI/
C      ================  are filled (see description in Geanfi)
********************************************************************************
      INTEGER FUNCTION DBDGGI(IDXID,SOURCC,KEY14,RUNNUM,NTRIG)
********************************************************************************
```

```
C
C      Description
C      ===========
C      Get the values for the Drift Chamber General Geometry Information
C
C      Input Parameter:
C      ===============
C      CHARACTER*2 IDXID  ! Character Index of the database (ge)
C      CHARACTER*4 SOURCC ! Offline source
C      INTEGER     KEY14  ! key 14 for the db
C      INTEGER     RUNNUM ! instant of validity (run number)
C      INTEGER     NTRIG  ! Packed word for the run and experiment type
C                         ! as done in A_C
C
C      Output parameter: none , the commons COMMON /MCDIMS/ and COMMON /DIGITI/
C      ================  are filled (see description in Geanfi)
C
********************************************************************************
      INTEGER FUNCTION DBBPDE(IDXID,SOURCC,KEY14,RUNNUM,NTRIG)
********************************************************************************
C
C      Description
C      ===========
C      Get the values for the Beam Pipe
C
C      Input Parameter:
C      ===============
C      CHARACTER*2 IDXID  ! Character Index of the database (ge)
C      CHARACTER*4 SOURCC ! Offline source
C      INTEGER     KEY14  ! key 14 for the db
C      INTEGER     RUNNUM ! instant of validity (run number)
C      INTEGER     NTRIG  ! Packed word for the run and experiment type
C                         ! as done in A_C
C
C      Output parameter: none , the commons COMMON /MCDIMS/ and COMMON /DIGITI/
C      ================  are filled (see description in Geanfi)
C
********************************************************************************
      INTEGER FUNCTION DBQDES(IDXID,SOURCC,KEY14,RUNNUM,NTRIG)
********************************************************************************
C
C      Description
C      ===========
C      Get the values for the Quadrupoles
C
C      Input Parameter:
C      ===============
C      CHARACTER*2 IDXID  ! Character Index of the database (ge)
```

```
C      CHARACTER*4 SOURCC ! Offline source
C      INTEGER     KEY14  ! key 14 for the db
C      INTEGER     RUNNUM ! instant of validity (run number)
C      INTEGER     NTRIG  ! Packed word for the run and experiment type
C                         ! as done in A_C
C
C      Output parameter: none , the commons COMMON /MCDIMS/ and COMMON /DIGITI/
C      ================  are filled (see description in Geanfi)
C
********************************************************************************
      INTEGER FUNCTION DBQIAD(IDXID,SOURCC,KEY14,RUNNUM,NTRIG)
********************************************************************************
C
C      Description
C      ===========
C      Get the values for the Quadrupoles Instrumentation and Anticoincidence
C      detector.
C
C      Input Parameter:
C      ===============
C      CHARACTER*2 IDXID  ! Character Index of the database (ge)
C      CHARACTER*4 SOURCC ! Offline source
C      INTEGER     KEY14  ! key 14 for the db
C      INTEGER     RUNNUM ! instant of validity (run number)
C      INTEGER     NTRIG  ! Packed word for the run and experiment type
C                         ! as done in A_C
C
C      Output parameter: none , the commons COMMON /MCDIMS/ and COMMON /DIGITI/
C      ================  are filled (see description in Geanfi)
C
********************************************************************************
      INTEGER FUNCTION INITST(RUNNUM,EXPTYP,RUNTYP)
********************************************************************************
C
C     Description:
C     ============
C     Function which retrieves the fine cell response
C     parametrizations from the databases
C     and fills the COMMON /PARAMETRI2/ and the COMMON /PARAMETRI3/
C
C     Input Paramater
C     ==============
C     INTEGER RUNNUM   ! Run number used as instant of validity for
C                      ! the object to be retrieve from databases
C     INTEGER EXPTYP   ! Experiment type used as in A_C
C     INTEGER RUNTYP   ! Run type used as in A_C
C                      ! These 2 last information are used to fill
C                      ! the database key14 concerning the RUNID
```

```
C
C     Output parameter : none
C     ================
C
***************************************************************************
      INTEGER FUNCTION INITRAWST(RUNNUM,EXPTYP,RUNTYP)
***************************************************************************
C
C     Description:
C     ============
C     Function which retrieves the raw cell response
C     parametrizations from the databases
C     and fills the COMMON /RAWPAR2/ and the COMMON /RAWPAR3/
C
C     Input Paramater
C     ===============
C     INTEGER RUNNUM   ! Run number used as instant of validity for
C                      ! the object to be retrieve from databases
C     INTEGER EXPTYP   ! Experiment type used as in A_C
C     INTEGER RUNTYP   ! Run type used as in A_C
C                      ! These 2 last information are used to fill
C                      ! the database key14 concerning the RUNID
C
C     Output parameter : none
C     ================
C
***************************************************************************
      INTEGER FUNCTION INITFSIG(RUNNUM,EXPTYP,RUNTYP)
***************************************************************************
C
C     Description:
C     ============
C     Function which retrieves the coefficients
C     to parametrize the resolution curve on drift distance
C     and fills the COMMON /FINESIG/
C
C     Input Paramater
C     ===============
C     INTEGER RUNNUM   ! Run number used as instant of validity for
C                      ! the object to be retrieve from databases
C     INTEGER EXPTYP   ! Experiment type used as in A_C
C     INTEGER RUNTYP   ! Run type used as in A_C
C                      ! These 2 last information are used to fill
C                      ! the database key14 concerning the RUNID
C
C     Output parameter : none
C     ================
C
```

```
******************************************************************************
      INTEGER FUNCTION INITRSIG(RUNNUM,EXPTYP,RUNTYP)
******************************************************************************
C
C     Description:
C     ============
C     Function which retrieves the raw
C     sigma table as a function on drift distance
C     and fills the COMMON /RAWSIG/
C
C     Input Paramater
C     ==============
C     INTEGER RUNNUM   ! Run number used as instant of validity for
C                      ! the object to be retrieve from databases
C     INTEGER EXPTYP   ! Experiment type used as in A_C
C     INTEGER RUNTYP   ! Run type used as in A_C
C                      ! These 2 last information are used to fill
C                      ! the database key14 concerning the RUNID
C
C     Output parameter : none
C     ================
C
******************************************************************************
```

## References


[1] KLOE Database server for test beam, F. Pelucchi

[2] Database User's and reference manuel, F. Pelucchi