CB-Note 122

Calibration Constants Database Software Ver. 2.01
LEAR Crystal Barrel Experiment, PS 197

F.-H. Heinsius

I. Institut für Experimentalphysik
Universität Hamburg

August 1, 1991

Contents

1 Introduction

The management of all calibration data for the Crystal Barrel requires a tool to store, to retrieve and to copy the constants. This is supported by the *Calibration Constants Database Software*. It consists of a package for storing the constants after the calibration of a range of runs and reading the constants according to the run number or the GCB version for Monte Carlo data (Described in chapter 3). Another package is provided for the interactive creation of a database, listing of constants and copying of constants to another database (This is described in chapter 4). The whole package is based on the ZEBRA memory management package and uses extensively the RZ part. ZEBRA/RZ provides the framework for buildung a database to store ZEBRA banks. It also supports the exchange of the database between different computers.

Note: Not all features may be implemented yet.

Big change in version 2.0: The special RUNS directory is not used any more. A directory DEFAULT is foreseen to store default constants for not yet calibrated data e.g. BOL (bicycle online) data. New features of version 1.2: Added export and import of databasefile with KUIP. Command REDUCE added to KUIP to set the last run of a runperiod to a lower value (experts only).

2 Survey to the Database

The basic considerations for the design of the database where:

- Easy access to the calibration constants according to the run numbers.
- Support of the calibration constants for different Monte Carlo versions.
- Splitting of storage of the constants for the different subdetectors.
- Grouping of constants from one run period.
- Storing of different calibration sets for the same run, but having always one default set of constants. This allows reanalyzing of data using earlier calibration data (i.e. the one used for the software trigger).
- Possibility of adding calibration sets for a new subdetector.

All this can be done using the RZ/ZEBRA package. A basic understanding of the RZ-structure is assumed in the following description. Nevertheless it is possible to use the database without knowing of RZ (but MZ/ZEBRA must be known).

The database is organized in RZ-directories, which consists of RZ-keys. Every RZ-key is a pointer to a ZEBRA bank structure. See picture 1 while reading the following description.

The database (top directory name 'CALIN') is splitted into several subdirectories, representing the different runperiods (note that runperiods are named according the date of the first run). Each runperiod subdirectory contains a key (number one) which carries some general information on the runperiod (bank 'CINF': first run number, last run number in this period). In addition it contains a directory for each subdetector used in this runperiod (BE = Barrel Calorimeter Energy Calibration Constants; BL = Barrel Calorimeter Lookuptable).

The calibration constants are stored in the directories of the different subdetectors (i.e. BP, BE, BL,...). Every calibration, which can span many runs, requires only one entry. It is identified by five keys: The first key is the first run number, the second the last run number and the third the

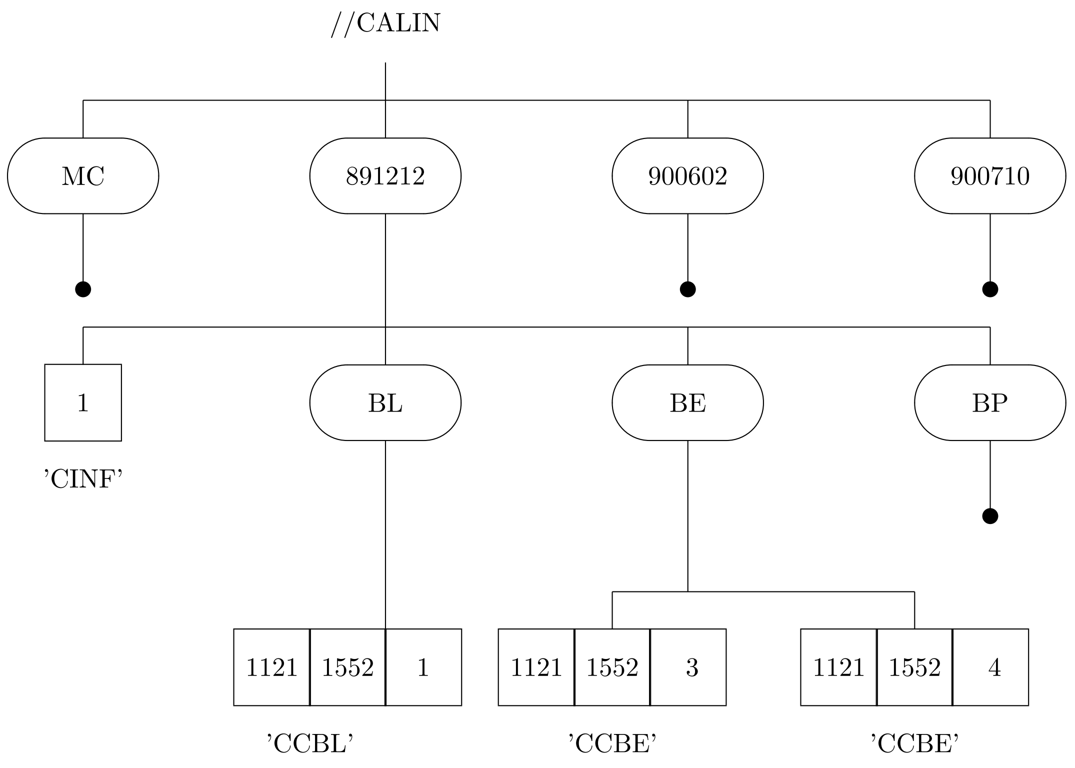

Figure 1: Survey to the database. The ovals represent RZ-directories, the boxes the RZ-keys, the ZEBRA banks are enclosed in quotes. See text for a decription.

calibration method. Each calibration type can be identified by the "method identifier". The method identier is negativ for the default calibration. Different calibration methods can span different runs.

The fourth and fifth key (not shown in picture 1) are for internal use of the database. They hold the type of record (full data, update data or link to other record) and the date of first insertion into the database.

More information on the banks can be found in section 3.1 and in appendix A.

3 Offline Access to the Database

Offline access to the database is possible through the routines CCGETC (get constants) and CC-STOR (store constants). The index for retrieving data is the run number, for storing data the range of runs. As a second index the subdetector is used. Subdetectors can be specified using the parameters in `+SEQ,CCKONS.`, see table 3 on page 19. All data are stored in the structure of ZEBRA banks, as described in the next section. The format of the top bank is special.

3.1 Description of the Common Calibration Banks Format

All banks of calibration constants, stored in the calibration database, have a supporting top bank, with the format as shown in table 1. The first two letters of this bank are 'CC' the last two are he abbreviation of the subdetector (see table 3 on page 19). The format and number of the subbanks is free and can be totally different for each subdetector. But to ensure correct handling in interactive accesses to the database every subbank should have no next or down links and the recommended types are all integer or all real.

Offset	TYPE	*Quantity*
-K	DOWN LINK	*Pointer to bank with constants*
⋮	⋮	⋮
-1	DOWN LINK	*Pointer to bank with constants*
LQ(↑)		
IQ(↓)		
+1	INTEGER	*first run*
+2	INTEGER	*last run*
+3	INTEGER	*calibration method*
+4	INTEGER	*record type (internal use only)*
+5	INTEGER	*date of first insertion (internal use only)*
+6	INTEGER	*date of calibration*
+7	HOLLERITH	*computer reconstruction done on* (see appendix C)
+8	INTEGER	*calibration program version*
+9	INTEGER	*(for use in the future)*

Table 1: Data stored in the **CCxx** top banks

3.2 How to get the Constants from the Database

At the initialization of the offline program a call to CCINIT is needed to open the direct access file. Monte Carlo data are distinguished from real data at the start of a run by a call to CCISMC.

For Monte Carlo:

```
CALL CCISMC(.TRUE.,IGCB)
```

IGCB is the integer (major version number * 100).

For real data:

```
CALL CCISMC(.FALSE.,0)
```

To get the constants call CCGETC

```
CALL CCGETC(IRUN,ITYPE,KPART,LINK,IERR)
```

With IRUN equal the run number, ITYPE the calibration (0=default method), KPART the parameter for the specified subdetector (see table 3 on page 19), LINK returns the datastructure including the calibration banks and IERR tells any error condition (if negative LINK is invalid, if +1 old constants remains valid). If no constants are returned you can set the run number IRUN to zero, then you get constants from the *DEFAULT* directory, which should be approximately correct.

Once before the end of the program call CCLAST.

3.2.1 Description of the subroutines for retrieving data

This section gives a brief survey of each subroutine.

An asterisk (*) in front of a call argument indicates that this variable is changed by the subroutine. Unless specifically mentioned all variable types correspond to the FORTRAN defaults.

3.2.2 SUBROUTINE CCINIT

Author: F.-H. Heinsius
Creation Date: 18 March, 1989
References:
Call Arguments: (*IERR)
Common Blocks Used: CBLINK, CBUNIT, CCLINK, CCWORK, CCKONS, QUEST
Subroutines Referenced: CCDATA, CCOPEN, (ZEBRA) RZFILE, RZCDIR, RZRDIR

This subroutine initializes the commons /CCWORK/, /CCKONS/ and opens an RZ file for reading the calibration constants. It has to be called once at the beginning of the program. Sets real data input to default.

3.2.3 SUBROUTINE CCISMC

Author: F.-H. Heinsius
Creation Date: 14 May, 1990
References:
Call Arguments: (QISMC, IGCB)
Common Blocks Used: CCWORK
Subroutines Referenced: none.

This subroutine selects, whether the constants are for real data (QISMC is false) or for Monte Carlo (QISMC is true and IGCB is the major GCB version * 100 in integer).

3.2.4 SUBROUTINE CCGETC

Author: F.-H. Heinsius
Creation Date: 18 March, 1989
References:
Call Arguments: (IRUN, ITYPE, KPAR, *LINK, *IERR)

Common Blocks Used: CBLINK, CCLINK, CCWORK, CCKONS, QUEST
Subroutines Referenced: (ZEBRA) MZDROP, RZCDIR, RZRDIR, RZIN

Call the routine CCGETC to get the calibration constants from the database into a zebra structure, which is accessible through the zebra pointer LINK. Specify the run number IRUN, calibration method ITYPE (default is zero) and subdetector KPAR (index from common /CCKONS/). In case of errors (constants not found etc.) IERR is negative and the value of LINK is not valid. IERR is 0 if new constants are loaded and 1 if the constants of the last call remain valid. If no constants are returned you can set the run number IRUN to zero, then you get constants from the *DEFAULT* directory, which should be approximately correct.

For Monte Carlo data the run number is not used, the constants are selected by the GCB version, which is set with routine CCISMC.

3.2.5 SUBROUTINE CCLAST

Author: F.-H. Heinsius
Creation Date: 18 March, 1989
References:
Call Arguments: none.
Common Blocks Used: CBUNIT, CCWORK
Subroutines Referenced: (ZEBRA) RZEND

This subroutine closes the database (RZ-file) opened by CCINIT.

3.3 How to Store Calibration Constants into the Database

For a calibration two database files are needed. One with the old constants (if starting values are needed) and another, which should be a copy of the first database, where the new calibration data are added. You declare this database to the program with a call to CCOLDF (Arguments are described in the next section). If you have no old calibration database you can create a new one using CCNEWF. If the calibration covers a new runperiod (or you have a new database created using CCNEWF) you have to initialize it with a call to CCPERI. This could also be done interactive (see chapter 4). Remember to call CCDATA and CCISMC to have everything properly initialized.

For every range of runs which are calibrated, a bank structure must be created according to the description in section 3.1. (Monte Carlo constants are handled in a special way independent of run numbers, according to the GCB version). The top bank of this structure can be created and filled with subroutine CCBOOK. Assuming that LINK points to the top bank and KBE is the identifier for the calibrated subdetector you call CCSTOR

```
        CALL CCSTOR (LINK, '//CALIB', KBE, ' ')
```

to store the new constants in the database. '//CALIB' is the topdirectory specified by the call to CCOLDF or CCNEWF (for details see next section).

Once at the end you call CCENDF to close the new database. If the new calibration is accepted you can delete the old database and rename the new one to the name of the old one.

3.3.1 Description of callable routines

This section gives a brief survey of each subroutine.

An asterisk (*) in front of a call argument indicates that this variable is changed by the subroutine. Unless specifically mentioned all variable types correspond to the FORTRAN defaults.

3.3.2 SUBROUTINE CCNEWF

Author: F.-H. Heinsius
Creation Date: 16 March, 1989
References:
Call Arguments: (IURZ, NREC, TPDIR, CHOPT)
Common Blocks Used: CCKONS, QUEST
Subroutines Referenced: CCOPEN, CCPERI, (ZEBRA) RZMAKE, RZLOGL

This subroutine creates a new direct access file to be used for calibration constants on FORTRAN unit IURZ with the maximum number of records given in NREC. It declares the new file as a RZ-file and creates the top directory with the name TPDIR (character*16). The character variable CHOPT supplies the value to the call to RZMAKE. A default "period" for Monte Carlo constants is created as well.IQUEST(1) is not zero if an error occured.

3.3.3 SUBROUTINE CCOLDF

Author: F.-H. Heinsius
Creation Date: 30 March, 1989
References:
Call Arguments: (IURZ, TPDIR)
Common Blocks Used: CCKONS, QUEST
Subroutines Referenced: CCOPEN, (ZEBRA) RZFILE, RZLOGL

This subroutine opens an old direct access file used for calibration constants on unit IURZ. It declares it as RZ-file with the top directory TPDIR (character*16). IQUEST(1) is not zero if an error occured.

3.3.4 SUBROUTINE CCPERI

Author: F.-H. Heinsius
Creation Date: 16 March, 1989
References:
Call Arguments: (TPATH, CPERI, IFRST, ILAST, IPART, NPART)
Common Blocks Used: CBLINK, CCLINK, CCKONS, QUEST
Subroutines Referenced: (ZEBRA) MZBOOK, RZCDIR, RZMDIR, RZOUT

The subroutine CCPERI creates a new directory for run period CPERI (character*16) which starts with run IFRST and ends with run ILAST. Variables IPART(1), NPART=1) are only held for backward compatibility. It uses the RZ-file with topdirectory TPATH (character). For Monte Carlo constants and *DEFAULT* constants first and last run numbers are 0.

3.3.5 SUBROUTINE CCBOOK

Author: F.-H. Heinsius
Creation Date: 12 June, 1989
References:
Call Arguments: (KPART, NS, IFRST, ILAST, IMETH, IVERS, *LINK)
Common Blocks Used: CBLINK, CCLINK, CCKONS
Subroutines Referenced: CCIDEN, (ZEBRA) MZIOCH, MZBOOK

This subroutine books the top calibration bank at LINK for subdetector KPART (index from CCKONS) and NS down links for the banks containing the calibration constants. It fills the data region with first run IFRST, last run ILAST, calibration method IMETH, calibration program verison IVERS and the current date and computer identification obtained from subroutine CCIDEN.

Note: For Monte Carlo constants you have to give the range of validity of GCB versions instead of run numbers (e.g. IFRST=402,ILAST=410). IFRST=1 means from all previous versions on. ILAST=-1 means for all future versions (until new constants are stored).

To store the constants in the *DEFAULT* directory specify IFRST=0,ILAST=0).

3.3.6 SUBROUTINE CCIDEN

Author: F.-H. Heinsius
Creation Date: 12 June, 1989
References:
Call Arguments: (*IDATE, *ICOMP)
Common Blocks Used: none.
Subroutines Referenced: (KERNLIB) DATIME, UCTOH

Returns the current date (integer) and a computer identification (hollerith). The identification is composed of a three letter abbreviation of the computer (APO, ALT, UNX, NRD, SUN, CRY, IBM, MVS, VAX, DEC) and a one letter code for the place where the calibration has taken place. Both are selected by patchy/CMZ flags. (See source code for available flags.)

3.3.7 SUBROUTINE CCSTOR

Author: F.-H. Heinsius
Creation Date: 16 March, 1989
References:
Call Arguments: (LINK, CDIR, KPART, CHOPT)
Common Blocks Used: CBLINK, CCLINK, CCKONS, CCWORK, QUEST
Subroutines Referenced: CCUPDR, (ZEBRA) RZCDIR, RZRDIR, RZIN, RZOUT

Stores the zebra structure referenced by LINK (format see section 3.1) which contains calibration data for subdetector KPART (index from CCKONS) in the RZ file with the topdirectory given by CDIR (character*16). CHOPT (character) equal 'D' labels a calibration to be the default calibration. CHOPT equal ' ' simply adds the calibration to the database.

For storing Monte Carlo constants you have to call CCISMC before.

If the directory for the specified subdedector does not yet exist it will be created.

3.3.8 SUBROUTINE CCENDF

Author: F.-H. Heinsius
Creation Date: 17 March, 1989
References:
Call Arguments: (IURZ, TPDIR)
Common Blocks Used:
Subroutines Referenced: (ZEBRA) RZEND

This subroutine closes the database (RZ-file) on unit IURZ with the top directory TOPDIR (character*16).

3.4 Description of COMMONs

3.4.1 CCKONS

In the /CCKONS/ common block are some constants stored:

```
      INTEGER    KBE, KBL, KBP, KBX, KBR, KBC,
     &           KJZ, KJE, KJS, KJT, KJW, KJC,KJP,
     &           KPW,
     &           KCR, KCB, KCG,
     &           KLG, KLP, KLX,
     &           MPARCC,
     &           MCYCCC, NRECCC, MDIRCC
      PARAMETER (KBE=1, KBL=2, KBP=3, KBX=4, KBR=5, KBC=6,
     &           KJZ=10, KJT=11, KJE=12, KJS=13, KJW=14, KJC=15,KJP=16,
     &           KPW=19,
     &           KCR=20, KCB=21, KCG=22,
     &           KLG=31,KLP=32,KLX=33,
     &           MPARCC=33,
     &           MCYCCC=99999999, NRECCC=1024, MDIRCC=100)
      CHARACTER*2 PARTCC
      COMMON /CCKONS/PARTCC(MPARCC)
```

- KBE, KBL, KBP, KBX, KBX, KBR, KJZ, KJT, KJE, KJS, KJW, KJC, KJP, KPW, KCR, KCB, KCG, KLG, KLP, KLX Indices to identify the different subdetectors.
- MPARCC Maximum number of subdetectors.
- MCYCCC Maximum number of cycles of RZ-keys.
- NRECCC Recordlength in long words for direct access RZ-file.
- MDIRCC Maximum number of directories in the RZ-file.
- PARTCC Abbreviations of the different subdetectors.

3.4.2 CCLINK

This common is the zebra link area used by the database.

```
      INTEGER         LCINF,LCIN2,LCRUN,LCRU2,LCDBL,LPART,LCTMP
      COMMON /CCLINK/ LCINF,LCIN2,LCRUN,LCRU2,LCDBL,LPART,LCTMP
```

- LCINF Structural link to **CINF** bank for CCGETC.
- LCIN2 Temporary structural link to **CINF** bank.
- LCRUN Structural link (unused)
- LCRU2 Temporary structural link to **CRUN** bank (unused).
- LCDBL Structural link used in subroutine DBLIS2/DBLIS3.
- LPART Temporary reference link used only inside one subroutine.
- LCTMP Temporary reference link used only inside one subroutine.

3.4.3 CCWORK/CCWRKC

Working storage for the calibration database software.

```
+SEQ,CCKONS.
      INTEGER         IRUNCC, IFIRCC, ILASCC, IMETCC, ICYCCC,
     &                ITYPCC, IDATCC, IGCBCC
      LOGICAL         QISMCC

      COMMON /CCWORK/ IRUNCC, IFIRCC(MPARCC), ILASCC(MPARCC),
     &                IMETCC(MPARCC), ICYCCC(MPARCC), ITYPCC(MPARCC),
     &                IDATCC(MPARCC), IGCBCC,
     &                QISMCC

      CHARACTER*16 TOPDCC, PERICC

      COMMON /CCWRKC/ TOPDCC,PERICC
```

- IRUNCC Current run number of loaded calibration constants.
- IFIRCC, ILASCC, IMETCC, ITYPCC, IDATCC, ICYCCC Identification of currently loaded calibration constants of given subdetector index.
- IGCBCC Current GCB version number, if Monte Carlo data.
- QISMCC Logical is true for Monte Carlo runs.
- TOPDCC Topdirectory of the calibration database input file.
- PERICC Directory of currently loaded runperiod.

3.5 Description of Internal Subroutines

3.5.1 SUBROUTINE CCOPEN

Author: F.-H. Heinsius
Creation Date: 2 June, 1989
References: Subroutine HROPEN by R.Brun
Call Arguments: (LUN, CFNAM, CHOPT, LRECL, MXREC, *ISTAT)
Common Blocks Used: none.
Subroutines Referenced: (IBM ONLY) FILEINF

This general purpose subroutine opens a direct-access file on unit LUN with filename CFNAM (character) or if blank without filename using the recordlength LRECL (in longwords, normally 4 bytes). The character variable CHOPT specifies 'N' for allocating a new file, 'U' to use an old, already existing file for updating and 'O' (same as ' ') to allocate an old file using only read access. For new files MXREC defines the number of records to be allocated (IBM only). The status of the open statement (IOSTAT=) is returned in the variable ISTAT.

When using this subroutine with file specification no external allocation of the file is needed. With no file specification it is needed to allocate the file outside the program using DEFINE/ASSIGN (VAX), FILEDEF (IBM/VM), ALLOC (IBM/TSO), DD statement (IBM/JCL) etc., see section C.2.

3.5.2 SUBROUTINE CCDATA

Author: F.-H. Heinsius
Creation Date: 7 June, 1989
References:
Call Arguments: none.
Common Blocks Used: CBLINK, CCLINK, CCKONS
Subroutines Referenced: (ZEBRA) MZLINK

This subroutine fills the common /CCKONS/ and declares the ZEBRA link area CCLINK.

3.5.3 SUBROUTINE CCUPDR

Author: F.-H. Heinsius
Creation Date: 17 March, 1989
References:
Call Arguments: (CDIR, KPART, KEY, ICYCL, CHOPT)
Common Blocks Used: CBLINK, CCLINK, CCKONS, CCWORK, QUEST
Subroutines Referenced: (ZEBRA) MZBOOK, MZDROP, MZPUSH, RZCDIR, RZIN, RZOUT, RZPURG

bf Obsolete!
Update the RUNS directory of the RZ-file (top directory CDIR, character*16) for subdetector KPART (index) with the information given in KEY(1:5) and ICYCL. CHOPT (character) equal 'D' labels a calibration to be the default calibration. CHOPT equal ' ' simply adds the calibration to the database.

4 Interactive Access to the Database

4.1 Survey

As the offline access to the database is designed for the main purpose of the database — storing and retrieving calibration constants — it is useful to have an interactive tool to manage the database. Using the KUIP command interpreter it is possible to add new runperiods to the database, to list calibrations and to copy the database either to another database (for adding new calibrations) or to a sequential FZ-file to transport the database to another computer, or import it from a FZ-file written by another computer.

All this can be done in an interactive environment where the KUIP, MACRO and VECTOR commands of PAW are available.

All you have to do is to start the program DBKUIP.

4.2 Introduction to the Command Interpreter KUIP

The purpose of KUIP (Kit for a User Interface Package) as a User Interface is to handle the dialogue between the user and the application program requesting input. It takes care of input and parsing of commands, it verifies their syntactical correctness, and then invokes the appropriate application routines.

Various styles of dialogue are available within KUIP: the default style is the so-called 'command mode' (type HELP SET_SHOW/STYLE for more information). In command mode KUIP writes a prompt to request user input.

HELP KUIP/SYNTAX gives more information and examples.

4.3 DB Commands

4.3.1 Notation used

The information supplied for each command can be obtained by entering 'HELP command'. The text contain the full and not abbreviated command_name followed by the parameter list (if any). The '*' on the left of some command names (which of course must not be typed in) means they are executable (or terminal) commands and not menus. A non terminal command can be only inquired for help, but not executed. For example one can enter 'HELP SET_SHOW' to see information about SET_SHOW, but not just 'SET_SHOW' to execute it; on the other hand one can enter 'HELP SET_SHOW/TIMING' and 'SET_SHOW/TIMING' because SET_SHOW/TIMING is a terminal command.

The command name (but only if executable) may have parameters associated to it. Some of them may be mandatory and the remaining optional. As the meaning of parameters is 'positional', a mandatory parameter cannot appear after an optional parameter. Therefore parameters can be: none, all mandatory, all optional, some mandatory followed by some optional. In the following text the optional parameters are enclosed in square brackets (which again must not be typed in).

Even if parameters are requested together with a given command_name they can be omitted in the command_line (i.e. only the command_name is entered). All the parameters not entered directly with the command_line will be prompted for, unless they are defined optional.

In the following text, after the command + parameter(s) line there is one line for each parameter, with the following syntax:

```
PNAME TYPE 'PROMPT STRING' D=DEFAULT R=RANGE
```

where PNAME is the parameter name, TYPE is the parameter type ('R' for real, 'I' for integer, 'C' for character), 'PROMPT STRING' is the prompt associated (used as parameter descriptor and printed on the terminal when executing commands without supplying mandatory parameters). D=DEFAULT and/or R=RANGE may be present or not. The default value is that taken when executing commands without supplying optional parameters or entering carriage return in response to mandatory parameters prompts. The range may be defined to bound the parameter to a set of permissible values, like 'ON,OFF,ABC' on a character type parameter or 0:100, 1:, :50, -99.1:103.7 on a numeric type parameter (0:100 means from 0 to 100, 1: means from 1 on, :50 means up to 50, etc.).

The default value of a parameter can also be selected with an exclamation mark at the place of the parameter. This is useful to specify just one optional parameter after a list of other optional parameters. Example:

```
TRACE  !  !  my_prompt
```

is equivalent to

```
TRACE  ON  ' '  my_prompt
```

because the first two parameters of the command TRACE, which are optional, have respectively the default values 'ON' and ' '. When the default value of a given parameter is not defined or when the exclamation mark is enclosed in single quotes, the exclamation mark is taken literally.

4.3.2 OUTPUT opt [file quota]

```
OPT         C 'Option' D='NEW' R='OLD,NEW'
FILE        C 'Filename' D=' '
QUOTA       I 'Maximum number of records' D=10000 R=100:
```

Opens an calibration file as output file. If no filename given the file must be preconnected to unit 18. Default is creating a NEW file, OPT='OLD' opens an already existing file. Quota is used only for new files. You need read *and* write access to open an output file.

4.3.3 INPUT [file]

```
FILE       C 'Filename' D=' '
```

Opens an old calibration file as input file. If no filename given the file must be preconnected to unit 17.

4.3.4 CREATE prdir ifrst ilast

```
PRDIR      C 'Run period'
IFRST      I 'First run' D=1 R=0:
ILAST      I 'Last  run' D=9999 R=0:
```

Create a new directory for a new runperiod in the output file. Name the runperiods according to the date of the first run.

4.3.5 LIST [period part run method]

```
PERIOD     C 'Run period' D=' '
PART       C 'Subdetector' D=' ' R=' ,*,BE,BL,BP,BX,JZ,JT,JE,JS,JW,PW,LG,LP,LX'
RUN        I 'Run number' D=0 R=0:
METHOD     I 'Calibration method' D=0 R=0:
```

List parts of the database, depending on parameters. This command uses the input or the output file depending on the current default, which can be changed by the command DB/SWITCH. With NO parameters: The name and runs of all runperiods are listed. With PERIOD: Lists the calibration dates and types for all subdetectors. With PART: Lists the calibration dates and types for this subdetector. PART=* lists it for all subdetectors. With RUN and METHOD: Copies the calibration constants for this run into vectors of the same name as the calibration banks. If the banks have mixed integer/real format one integer and one real vector is filled.

4.3.6 SWITCH [file]

```
FILE       C 'Input or output file' D=' ' R=' ,?,INPUT,OUTPUT'
```

Switch the default file for LIST between in-/output file (no parameters). Specify INPUT to switch to the input file, output for the output file and '?' to get the current default file.

4.3.7 DELETE prdir [part]

```
PRDIR      C 'Run period'
PART       C 'Subdetector' D=' '
```

Deletes whole directory of calibrations constants for one run-period. If PART given deletes this only for one Subdetector. Works on the output file.

4.3.8 RENAME oldpr newpr

```
OLDPR      C 'Run period'
NEWPR      C 'Run period'
```

Changes the name of one runperiod.

4.3.9 COPY period part runs opt

```
PERIOD     C 'Run period' D=' '
PART       C 'Subdetector' D=' '
RUNS       C 'Run# or vector of run numbers' D='*'
OPT        C 'Option to select methods' D='ALL' R='ALL,DEFAULT,1,2,3,4,'
```

Sorry, does not work yet due to an RZCOPY error! Copies calibration constants from input to output file. PERIOD=* copies all runperiods, PART=* copies all subdetectors. RUNS=* copies all runs, one run can be specified as an integer, more runs as a vector (see VECTOR/CREATE). You can select special calibration methods by the integer number or the with OPT='DEFAULT' the default (primary) calibration method. If the period does not exist in the output file a new one will be created. Otherwise the constants will be added to already existing directories. Up to now PART,RUNS and OPT must be * * ALL.

4.3.10 REDUCE period lastrun

```
PERIOD     C 'Run period' D=' '
LASTRU     I 'Run# of the new last run number of this period'
```

Reduces the range of validity of one run period The LASTRUN given must be less than the old last run. Calibrations which are only valid after this run are deleted. All directories are updated.

4.3.11 MAKE_PRIMARY part method run

```
PART       C 'Subdetector' D=' '
METHOD     I 'Calibration method' D=0
RUN        I 'Run number' D=0
```

Make the calibration METHOD the default calibration for one Subdetector PART and RUN. For authorized users only. Uses the output file. (Not yet coded.)

4.3.12 SHOW_METHODS part

```
PART       C 'Subdetector' D=' '
```

Shows the defined calibration methods for subdetector PART. For introduction of new methods ask the software manager.

4.3.13 QUOTA

Shows the quota of the output file. Use copy into a greater file if quota exceeded.

4.3.14 PURGE

Purge old calibration data in the output file. Not yet implemented.

4.3.15 CHECK

Checks the consistency of the output file. Not yet implemented.

4.3.16 REPAIR

Repairs a corrupted calibration database. Not yet implemented.

4.3.17 SAVE

Saves the current status of the output file. (Use this command before the system breaks down.)

4.3.18 TOFZ rzfile fzfile [option]

```
RZFILE     C 'Input  RZ filename' D=' '
FZFILE     C 'Output FZ filename' D=' '
OPTION     C 'FZ file in eXchange or Alpha exchangeformat' D='X' R='X,A'
```

Copies the input RZ file to a FZ exchange file.

4.3.19 FROMFZ fzfile rzfile [option]

```
FZFILE     C 'Input  FZ filename' D=' '
RZFILE     C 'Output RZ filename' D=' '
OPTION     C 'FZ file in eXchange or Alpha exchange format' D='X' R='X,A'
```

Create a new output RZ file from a FZ exchange file.

4.4 Description of COMMONs

4.4.1 DBWORK/DBWRKC

Working storage for the interactive calibration database software.

```
      LOGICAL QINPDB, QOUTDB, QSWIDB
      INTEGER IINPDB, IOUTDB
      COMMON /DBWORK/ IINPDB, IOUTDB, QINPDB, QOUTDB, QSWIDB
      CHARACTER*18 INPFDB, OUTFDB
      COMMON /DBWRKC/ INPFDB, OUTFDB
```

- QINPDB True if input file open.
- QOUTDB True if output file open.
- QSWIDB True for output file selected, false for input file selected.
- IINPDB Logical unit for input file.
- IOUTDB Logical unit for output file.
- INPFDB Top directory of input file.
- OUTFDB Top directory of output file.

4.4.2 PAWC

ZEBRA storage used for KUIP.

```
      INTEGER NWPAW,IXPAWC,IHBOOK,IXHIGZ,IXKUIP,IFENCE,LMAIN,IQP,LQP
      REAL    WS,QP
      DIMENSION IQP(50000 -19),QP(50000 -19),LQP(50000 -10)
      COMMON /PAWC/ NWPAW,IXPAWC,IHBOOK,IXHIGZ,IXKUIP,IFENCE(5),
     +              LMAIN(1), WS(50000 -11)
      EQUIVALENCE (LQP(1),LMAIN(1))
      EQUIVALENCE (IQP(1),LQP(9))
      EQUIVALENCE ( QP(1),LQP(9))
```

- IQP, QP, LQP Integer data, real data, zebra links.

4.5 Description of the Program DBKUIP

Author: F.-H. Heinsius
Creation Date: 12 May, 1989
References:
Call Arguments: none.
Common Blocks Used: CBLINK, CCLINK, CCKONS, DBWORK, PAWC
Subroutines Referenced: CCDATA, DBDEFS, DBEXIT, (ZEBRA) MZEBRA, MZSTOR, MZLINK, MZPAW, (KUIP) KUINIT, KUEXIT, KUEXEC, KUWHAT, VECDEF

This is the user interface program to the database using the KUIP package of PAW. The commands are defined in subroutine DBDEFS. After initialization control is given to KUIP via the routine KUWHAT (to use grafik menus this call has to be replaced with KUWHAG). For every command a subroutine is called, the first two letters are 'DB', the last 4 letters are the first 4 letters of the command. For a description of these subroutines look into the source code.

A Internal Format of the Database

A.1 Description of the Period Banks CINF

In this bank is the valid range of runs stored. For the Monte Carlo period and the *DEFAULT* methods the first and last run is 0. See table 2.

Offset	TYPE	*Quantity*
+1	INTEGER	*first run*
+2	INTEGER	*last run*

Table 2: Data stored in the **CINF** bank

A.2 Description of the Part Banks CCBE, CCBP, ...

These top banks are decribed in section 3.1 on page 6. The down banks are depending on the subdetectors. They are shown in section B.2.

B Current Formats for the Crystal Barrel Database

The *Calibration Constants Database Software* is not limited to the Crystal Barrel. It can be used for every detector with big amount of calibration constants. The detector dependent parts are kept in the two letter abbreviations of the different subdetectors decribed in the following section.

B.1 Current Supported Subdetectors

Note that every part identifies one calibration independent of the other parts. For example barrel pedestals (BP) are first stored. In another calibration the gains are evaluated (BE).

Parameter +SEQ,CCKONS.	Two letter code PARTCC(Parameter) common /CCKONS/	Description
KBE	'BE'	Barrel Energy Constants
KBL	'BL'	Barrel Lookuptable
KBP	'BP'	Barrel Pedestals
KBX	'BX'	Barrel bad/unused crystals
KBX	'BR'	Barrel Ratios FERA/2282
KBC	'BC'	Barrel Pedestal corrections for FERA
KJZ	'JZ'	JDC z calibrations
KJE	'JE'	JDC hit amplitude to energy
KJS	'JS'	JDC stagger constants
KJT	'JT'	JDC r/Φ calibration
KJW	'JW'	JDC information on bad wires
KJC	'JC'	JDC card file
KJP	'JP'	JDC dE/dx calibration polynom
KPW	'PW'	PWC
KCR	'CR'	List of bad runs
KCB	'CB'	B-field
KCG	'CG'	Geometry
KLG	'LG'	Relative gains from lightpulser
KLP	'LP'	Lightpulser filter transmissions
KLX	'LX'	Crystals, used for Lightpulser

Table 3: Abbreviations of subdetectors

B.2 Formats of the Calibration Banks

B.2.1 Part BP (Barrel Pedestal): CBPF, CBPL, CBNF, CBNL

In the **CBPF** and **CBPL** banks are the pedestals for the FERA and for the LeCroy 2282 ADC systems stored, see table 4. In the **CBNF** and **CBNL** banks is the sigma of the pedestals for the FERA and 2282 ADC's stored. Same format as the **CBPF** and **CBPL** banks.

Offset	TYPE	*Quantity*
$+\varphi + (\vartheta - 1) \cdot 60$	REAL	*Pedestal* [ADC-Channel]
$\vdots$	$\vdots$	$\vdots$
+1561	REAL	*Threshold* [ADC-Channel]

Table 4: Data stored in the **CBPL** and **CBPF** bank

B.2.2 Part BE (Barrel Energy): CBEF and CBEL

In the **CBEF** and **CBEL** banks are the energy constants for the FERA and for the LeCroy 2282 ADC systems stored, see table 5. When only the **CBEL** banks are stored use the FERA/2282 ratios and FERA pedestal corrections form 'BR' and 'BC'.

B.2.3 Part BL (Barrel Lookuptable): CBLF and CBLL

In the **CBLF** is the data from LOOKBC(3,,) and in **CBLL** is the data from LK22BC stored (see common BCLOOK).

Offset	TYPE	*Quantity*
$+\varphi + (\vartheta - 1) \cdot 60$	REAL	*Slope (Energy/ADC-Channel)* [MeV]
$\vdots$	$\vdots$	$\vdots$

Table 5: Data stored in the **CBEL** and **CBEF** bank

Offset	TYPE	*Quantity*
+(FERA channel)	INTEGER	$\varphi + (\vartheta - 1) \cdot 60$
$\vdots$	$\vdots$	$\vdots$

Table 6: Data stored in the **CBLF** bank

B.2.4 Part BX (Bad crystals): CBXL, CBXF, CBCL, CBCF

The **CBXF** and **CBXL** banks contain a list of the the FERA and LeCroy 2282 ADCs which are not used for energy calculation, see table 8.

In the **CBCF** and **CBCL** banks are correction functions for nonlinear channels of the FERA and LeCroy 2282 ADC channels stored (table 9). For a description of the correction function see subroutine BCLINE of BCTRAK.

B.2.5 Part BR (Barrel Ratios FERA/2282): CBRF/BCRD

B.2.6 Part BC (Barrel Pedestal corrections for FERA): CBPC/BCCF

B.2.7 Part JE (JDC hit amplitude to energy): TJCE

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

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

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

B.2.8 Part JZ (JDC z calibration): TJCZ, TJZ0, TJZL, TJT0

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

The **TJT0** bank contains the time offset to be used for every wire in the JDC. At present, the data is read in from unit 81 with the rest of the gain information. If the information is not found in the file, then the program takes the crate–wise values in the ITFCRJ array found in the /RJPRMS/ common block. These values can be forced by using the **ITFC** control card. If 17=1 is used on this card, then the values from the common block are always used.

The **TJZL** bank contains the length of each wire in the JDC, z_l^i. These are not necessarily the same for every wire as the position of the electrical connection between the wire and crimp pins can

Offset	TYPE	*Quantity*
+(2282 channel)	INTEGER	$\varphi + (\vartheta - 1) \cdot 60$
$\vdots$	$\vdots$	$\vdots$

Table 7: Data stored in the **CBLL** bank

Offset	TYPE	*Quantity*
+1	INTEGER	*Number of bad channels N*
+2	INTEGER	*(φ,ϑ) of first bad channel*
⋮	⋮	⋮
+N+1	INTEGER	*(φ,ϑ) of last bad channel*

Table 8: Data stored in the **CBXL** and **CBXF** bank (Bad unused channels LeCroy, FERA)

Offset	TYPE	*Quantity*
-K	REFERENCE LINK	*Pointer to type of correction function* N
⋮	⋮	⋮
-1	REFERENCE LINK	*Pointer to type of correction function* 1
LQ(↑)		
IQ(↓)		
+1	INTEGER	*Number of correction functions N*
+2	INTEGER	*(φ,ϑ) for first correction function*
⋮	⋮	⋮
+N+1	INTEGER	*(φ,ϑ) for Nth correction function*
+N+2 +0	INTEGER	*type of first correction function*
+N+2 +1	INTEGER	*number of real parameters K*
+N+2 +2	REAL	*parameter 1*
+N+2 +K+2	REAL	*parameter K*
⋮	⋮	next correction functions

Table 9: Data stored in the **CBCL** and **CBCF** bank (Nonlinear channels LeCroy, FERA)

vary by several millimeters at both ends of the chamber. This data is addressed in the same manner as in the previous bank.

The **TJC0** bank contains the z–position at the center of each sense wire, $(z_l/2)$. This can also be different from zero for the same reasons as with the previous data bank. As with the last bank, the addressing is as in the **TJCE** data bank.

B.2.9 Part JS (JDC stagger constants): TJST

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

B.2.10 Part JT (JDC r-phi calibration): TJCP, TJRF

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

000 Use Taylor polynomials for the expansion.

001 Use Hermite polynomials for the expansion.

010 Use Laguerre polynomials for the expansion.

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

Offset	TYPE	*Quantity*
+001	INTEGER	*Calibration Date.*
+002	INTEGER	*Calibration Time.*
+003	REAL	*JDC Set Voltage Chan. 1.*
⋮		
+034	REAL	*JDC Set Voltage Chan. 32.*
+035	REAL	*Standard Deviation of Voltage 1.*
⋮		
+066	REAL	*Standard Deviation of Voltage 32.*
+067	REAL	*JDC True Voltage Chan. 1.*
⋮		
+098	REAL	*JDC True Voltage Chan. 32.*
+099	REAL	*Standard Deviation of Voltage 1.*
⋮		
+130	REAL	*Standard Deviation of Voltage 32.*
+132	REAL	*JDC Isobutane Flow.*
+133	REAL	*JDC CO2 Flow.*
+134	REAL	*JDC Mixed Flow.*
+137	REAL	*Standard Deviation of JDC Isobutane Flow.*
+138	REAL	*Standard Deviation of JDC CO2 Flow.*
+139	REAL	*Standard Deviation of JDC Mixed Flow.*
+140	REAL	*HDC Potential Voltage.*
+141	REAL	*HDC Grid Voltage.*
+142	REAL	*HDC Drift Voltage.*
+145	REAL	*Standard deviation of Potential Voltage.*
+146	REAL	*Standard deviation of Grid Voltage.*
+147	REAL	*Standard deviation of Drift Voltage.*
+150	REAL	*HDC Drift Time 1.*
+151	REAL	*HDC Drift Time 1 Sigma.*
+152	REAL	*HDC Drift Time 2.*
+153	REAL	*HDC Drift Time 2 Sigma.*
+154	REAL	*HDC Drift Time Difference.*
+155	REAL	*HDC Drift Time Difference Sigma.*
+160	REAL	*Standard Deviation of Time 1.*
+161	REAL	*Standard Deviation Time 1 Sigma.*
+162	REAL	*Standard Deviation Time 2.*
+163	REAL	*Standard Deviation Time 2 Sigma.*
+164	REAL	*Standard Deviation Time Difference.*
+165	REAL	*Standard Deviation Time Difference Sigma.*
+170	REAL	*JDC Absolute Pressure.*
+171	REAL	*JDC Differential Pressure.*

+172	REAL	*Gas mixture as fraction Isobutane.*
+175	REAL	*Standard Deviation JDC Pressure.*
+176	REAL	*Standard Deviation JDC Diff. Pres.*
+177	REAL	*Standard Deviation of mixture.*
+180	REAL	*JDC Temperature 1 in Kelvins.*
+181	REAL	*JDC Temperature 2 in Kelvins.*
+182	REAL	*JDC Temperature 3 in Kelvins.*
+183	REAL	*JDC Temperature 4 in Kelvins.*
+184	REAL	*JDC Temperature in Kelvins.*
+185	REAL	*Sigma in Temperature 1.*
+186	REAL	*Sigma in Temperature 2.*
+187	REAL	*Sigma in Temperature 3.*
+188	REAL	*Sigma in Temperature 4.*
+189	REAL	*Sigma in Temperature.*

B.2.11 Part JW (JDC information on bad wires): TJWR

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

B.2.12 Part JC (JDC card file): JCRD

B.2.13 Part JP (JDC dE/dx calibration polynom): JPOL

B.2.14 Part PW (PWC): CPWC

In the **CPWC** bank is the data from the PWC stored. A preliminary description can be found in table 10.

B.2.15 Part CR (List of bad runs): CCRN/CRUN

B.2.16 Part CB (B-field): CCBF/CBMG

B.2.17 Part CG (Geometry): CGEO

B.2.18 Part LP (Lightpulser): CLP1 and CLP2

In the **CLP1** and **CLP1** banks are the transmissions of the different filtercombinations of the two lightpulsers stored.

B.2.19 Part LG (Lightpulser, rel. gains): CLGE

Gain corrections for the barrel calibration from lightpulser results.

Offset	TYPE	*Quantity*
+1	REAL	*threshold*
⋮	⋮	⋮
+10	REAL	*threshold*
+11	REAL	*delay*
⋮	⋮	⋮
+20	REAL	*delay*
+21	REAL	*high voltage PWC 1*
+22	REAL	*high voltage PWC 2*
+23	REAL	*current PWC 1*
+24	REAL	*current PWC 2*
+25	REAL	*gas mixture Argon*
+26	REAL	*gas mixture Ethan*
+27	REAL	*gas mixture Argon/Ethan*
+28	REAL	φ *of wire 0 PWC 1*
+29	REAL	φ *of wire 0 PWC 2*

Table 10: Data stored in the **CPWC** bank

Offset	TYPE	*Quantity*
+1	INTEGER	*first run number*
+2	INTEGER	*last run number N*
+3	INTEGER	*first run: 0 for good, -1 for bad run*
⋮	⋮	⋮
+N+2	INTEGER	*last run: 0 for good, -1 for bad run*

Table 11: Data stored in the **CRUN** bank

B.2.20 Part LX (Lightpulser, crystals): CLXT

C Computer Dependencies

C.1 Computer identifications

C.2 Filehandling

The alloction of the database is handled different on all computers. Examples are given for VAX (DCL), IBM/VM and IBM/MVS (JCL). For read access you have to type the following:

DCL: `$ DEFINE FOR017 dev:[dir]CBBASE.DAT`

Offset	TYPE	*Quantity*
+1	INTEGER	*first run number*
+2	INTEGER	*last run number N*
+3	REAL	*first run: B-field*
⋮	⋮	⋮
+N+2	REAL	*last run: B-field*

Table 12: Data stored in the **CBMG** bank

Offset	TYPE	*Quantity*
+1	INTEGER	*format type (1)*
+2	REAL	*phi-orientation JDC*
+3	REAL	*-z shift barrel*
+4	REAL	*+z shift barrel*

Table 13: Data stored in the **CGEO** bank

Offset	TYPE	*Quantity*
+1	REAL	*transmission for combination 0*
⋮	⋮	⋮
+64	REAL	*transmission for combination 63*

Table 14: Data stored in the **CLP1** and **CLP2** banks

VM: `FILEDEF 17 DISK CBBASE DATA G (PERM XTENT 100000`

JCL: `//GO.FT17F001 DD DISP=SHR,DSN=user.DAT.CBBASE`

To create a new database use DBKUIP or use the following statements:

DCL: `$ DEFINE FOR018 dev:[dir]CBNEW.DAT`

VM:
```
FILEDEF 18 DISK CBNEW DATA A (PERM RECFM F BLKSIZE 4096
     DSORG DA XTENT 100000
```

JCL:
```
//GO.FT18F001 DD UNIT=SYSDA,DISP=NEW,DSN=user.DAT.CBNEW,
     // SPACE=(4096,(quota,0)),DCB=(RECFM=F,BLKSIZE=4096), DSORG=DA
```

To update a database use the following statements:

DCL: `$ DEFINE FOR018 dev:[dir]CBUPDATE.DAT`

VM: `FILEDEF 18 DISK CBUPDATE DATA A6 (PERM XTENT 100000`

JCL: `//GO.FT018F001 DD DISP=OLD,DSN=user.DAT.CBUPDATE`

Patchy/CMZ flag	3 letter identification	Description
AIX	AIX	IBM RISC station
APOLLO	APO	Apollo
ALT	ALT	Alliant
UNIX	UNX	Unix (other than AIX/CRY/...)
NORD	NRD	Nord
SUN	SUN	Sun
CRAY	CRY	Cray
IBM	IBM	IBM/VM
IBM,IBMMVS	MVS	IBM/MVS
DECS	DEC	DECstation Ultrix
VAX	VAX	VAX/VMS

Table 15: Abbreviations of computers

Patchy flag	1 letter identification	Description
BERKELEY	B	Berkeley
BOCHUM	O	Bochum
CERN	C	CERN/Geneva
HAMBURG	H	Hamburg
KARLSRUHE	K	Karlsruhe
LONDON	L	London
MAINZ	M	Mainz
MUENCHEN	U	Muenchen
RUTHERFORD	R	Rutherford
STRASBOURG	S	Strasbourg
ZUERICH	Z	Zürich

Table 16: Abbreviations of locations