GOOSY
Id.: DBM
Version: 1.0
Date: 14–Jun–1988
Revised: June, 28 1988

GSI Online Offline System

GOOSY Data Base Manager

H.G.Essel, M. Richter

June, 28 1988

GSI, Gesellschaft für Schwerionenforschung mbH
Postfach 11 05 52, Planckstraße 1, D-64220 Darmstadt
Tel. (0 6159) 71–0

List of Figures

Chapter 1

Preface

GOOSY Copy Right

Conventions used in this Document

[Fn], [PFn], [1], [Do], or [Return] key — All key in frame boxes refer to the special keypads on VTx20 compatible terminals like VT220, VT320, VT330, VT340, VT420, VT520, PECAD, PERICOM terminals or DECterm windows under DECwindows/Motif on top or right to the main keyboard, to control characters, or to the delete and return keys of the main keyboard.

<Fn>, <PFn>, <KPn>, <Do>, or <Ctrl>— This is the alternative way of writing the keypad or control keys.

[GOLD], <GOLD>— The [PF1] key is called [GOLD] in most utility programs using the keypad.

PERICOM— On the PERICOM terminal keyboard the function keys are marked opposite to all other terminals, i.e. the 4 [PFn] of the rightmost VTx20 compatible keypad are named [Fn] and the 20 [Fn] keys on the top of each VTx20 compatible keyboard are named [PFn] on a PERICOM.

[Return]— The [Return] is not shown in formats and examples. Assume that you must press [Return] after typing a command or other input to the system unless instructed otherwise.

[Enter]— If your terminal is connected to IBM, the [Enter] key terminates all command lines.

`Ctrl` **key** — The `Ctrl` box followed by a letter means that you must type the letter while holding down the `Ctrl` key (like the `Shift` key for capital letters). Here is an example:

- `Ctrl` `Z` means hold down the `Ctrl` key and type the letter `Z`.

`PFn` **key** — The `PFn` followed by a number means that you must press the `PFn` key and *then* type the number. Here is an example:

- `PF1` `6` press the `PF1` key and then type the number 6 on the main keyboard.

`PFn` **or** `Fn` **keys** — Any `PFn` or `Fn` key means that you just press this key. Here is an example:

- `PF2` means press the `PF2` key.

Examples— Examples in this manual show both system output (prompts, messages, and displays) and user input, which are all written in `typewriter` style. The user input is normally written in capital letters. Generally there is no case sensitive input in GOOSY, except in cases noted explicitly. In UNIX all input and with it user and file names are case sensitive, that means for TCP/IP services like Telnet, FTP, or SMTP mail one has to define node names, user names, and file names in double quotes "name" to keep the case valid for OpenVMS input. Keywords are printed with uppercase characters, parameters to be replaced by actual values with lowercase characters. The computer output might differ depending on the Alpha AXP or VAX system you are connected to, on the program version described, and on other circumstances. So do not expect identical computer output in all cases.

1.1 GOOSY Authors and Advisory Service

The authors of GOOSY and their main fields for advisory services are:

M. Richter GOOSY Data Management, VAX/VMS System Manager (Tel. 2394)

R. Barth GOOSY and PAW software (since 1995) (Tel. 2546)

H.G. Essel (GOOSY 1983-1993) Data Acquisition (Tel. 2491)

N. Kurz Data Acquisition (since 1992) (Tel. 2979)

W. Ott Data Acquisition (since 1994) (Tel. 2979)

People who have been involved in the development of GOOSY.

B. Dechant GOOSY software (1993-1095) (Tel. 2546)

R. S. Mayer Data Acquisition (1992-1995) (Tel. 2491)

R. Fritzsche Miscellanea (1989-1995) (Tel. 2419)

H. Grein Miscellanea (1984-1989)

T. Kroll Miscellanea, Printers (1984-1988)

R. Thomitzek Miscellanea, Printers, Terminals (1988-1989)

W. Kynast GIPSY preprocessor (1988)

W.F.J. Müller GOONET networking, Command interface (1984-1985)

H. Sohlbach J11, VME (1986-1989)

W. Spreng Display, Graphics (1984-1989)

K. Winkelmann GOOSY Data Elements, IBM (1984-1986)

1.2 Further GOOSY Manuals

The GOOSY system is described in the following manuals:

- GOOSY Introduction and Command Summary
- GOOSY Data Acquisition and Analysis
- GOOSY Data Management
- GOOSY Data Management Commands

- GOOSY Display
- GOOSY Hardware
- GOOSY DCL Procedures. GOOSY Error Recovery
- GOOSY Manual
- GOOSY Commands

Further manuals are available:

- GOOSY Buffer structures
- GOOSY PAW Server
- GOOSY LMD List Mode Data Generator
- SBS Single Branch System
- TCP-Package
- TRIGGER Bus
- VME Introduction
- OpenVMS Introduction

1.3 Intended Audience

This manual is written for GOOSY users. It assumes that the reader is familiar with most VAX-VMS concepts and commands. It provides all information necessary to use the Data Base Manager. For VAX beginners the 'VMS Introduction' is recommended. For GOOSY beginners the 'GOOSY Introduction' is recommended.

1.4 Overview

- Section 2:
  Introduction of special terms. Brief description of command inteface.
- Section 3:
  Data Base Manager program.
- Section 4:
  Data Base Manager commands.
- Section 5:
  GOOSY Data Elements.

The author would be grateful for any critical comment or any suggestion about this manual.

Chapter 2

Introduction

2.1 Data Base Organization

In a software system for data acquisition and data analysis a large number of differently structured Data Elements has to be handled. Those Data Elements could be simple variables like calibration parameters or complex structures like a spectrum. Manipulations like create, delete, modify, copy, and show are required for those data objects. The Data Elements must be sharable in memory, i.e. several programs must access the same Data Element at the same time. Any Data Element must be accessible by programs and by commands. For example, a spectrum must be filled by an analysis program, displayed by a display program and shown by a command. Therefore the Data Elements are collected in Data Bases.

The **Data Bases** of GOOSY are implemented on VAX computers as structured Global Sections. The **Data Elements** are stored in sections of the Data Base called **Data Area**. Each Data Area is a cluster of contiguous pages which will be mapped into a program's address space. Data Areas with similar mapping attributes are collected in **Data Pools**. Information about Data Elements is kept in **Data Element Directories**.

The Data Base is organized internally in a hierarchical manner. Each component of the organization can be addressed by name. All names are collected in Directories, the Directory names in a Master Directory. The data regions are split in Data Areas, the Areas are bundled in Data Pools. The names of the Data Areas and the Data Pools are collected in the Area Directory and the Pool Directory, respectively. A Home Block keeps the entry information about the main Directories and the storage information for the Data Areas, i.e. the Data Base usage. Figure 2.1 on page 11 gives a simple overview about the Data Base Organization.

In the following, the objects of a Data Base are described briefly. The list begins with the smallest entity which can be referenced:

- **Member Value**

  This is the smallest entity, which can be **located** and **accessed** via the Data Management. A Member Value is a simple variable of a specific Data Type. The supported Data Types are:

```
BIN FIXED(7)
BIN FIXED(15)
BIN FIXED(31)
BIN FLOAT(24)
BIN FLOAT(53)
BIT(*)  ALIGNED
CHARACTER(*)
CHARACTER(*) VAR
OFFSET
UNKNOWN
```

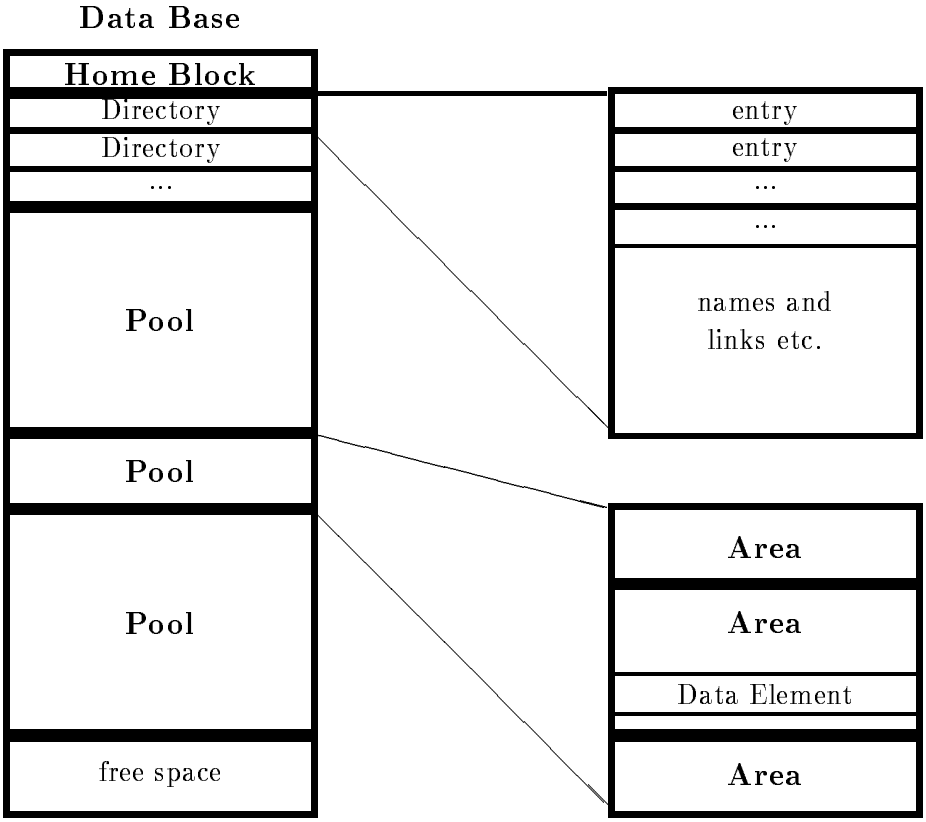

Figure 2.1: The simplified structure of a GOOSY Data Base.

- **Element Member**

  An Element Member is a Member Value or a 1 to 8 dimensional array of Member Values.

- **Data Element**

  A Data Element is the basic entity which can be manipulated by the Data Management. The structure of a Data Element is defined by a Data Type.

  The three basic Data Element forms are:

  – simple Data Element
    Containing only one Member Value or Element Member and corresponding to a simple variable or a simple array of PL/I.

  – complex Data Element
    Containing several Element Members and corresponding to a structure of PL/I.

– indexed Data Element (name arrays)
This is an array of Data Elements corresponding to a pointer array of PL/I referencing structures of the same Data Type.

- **Data Area**

A Data Area is the smallest entity which can be made accessible (mapped) to a program. This is normally invisible to the user.

- **Data Pool**

A Data Pool is the collection of Data Areas which requires the same specific combination of **protection** attributes. All Data Areas of a Data Pool are logically linked together. If a Data Area is short of space, a new Data Area can be created with the same attributes as the previous one. This new Data Area will be linked to the same Data Pool. A user normally knows the Data Pool only, not the Data Area.

- **Data Base**

All information associated with one application is normally collected in one Data Base. A Data Base is a storage area, in which Data Pools can be created.
Every Data Base has the following **protected system Data Areas**:

Home Block: The Home Block is a specific part of the Data Base. The Home Block is always located at the begin of every Data Base and contains all information to **locate** other Data Areas, especially the **Directories**. It also keeps an allocation bit map of the whole Data Base and general informations like the section file name, creation date, and time etc.

Area Directory: This Directory contains the relative addresses, the length in pages (512 bytes), the allocation cluster size, and the names of all Data Areas for the whole Data Base. It also keeps the Data Pool backward-link information for each Data Area.

Pool Directory: This Directory contains the names of all Data Pools for the whole Data Base and the minimum size in bytes of any Data Area in this Data Pool. It also keeps the link information for the first Data Area of each Data Pool.

Master Directory: This Directory contains the names of all Data Element Directories and the relative addresses of the Directory Areas in the Data Base.

Data Element Directories: Each of these Directories contains the names of all Data Elements, the relative addresses of their Data Areas, the relative addresses within the Data Areas, and other Data Element information for all Data Elements of one Data Element Directory. This information is called **Directory Entry**.

There are three additional extensions possible for each entry in a Data Element Directory:

– Any extension of the Data Element Descriptor. These extensions may be of any length. They are characterized by an extension type. All extensions of a Data

Element Descriptor are linked. The character string of the name of a Data Element is located within such an extension.

- A queue of Data Elements of the same Data Element Directory, i.e. in the same Data Element Directory. This allows to bind unnamed Data Elements to named Data Elements, e.g. a named spectrum header and its unnamed spectrum limit definitions.
- A link to Data Elements of any Data Element Directory. This allows logical correlations of Data Elements, e.g. conditions linked to a spectrum.

Type Directory: The Type Directory is a specific Data Element Directory which contains the Data Types declarations of all Data Elements in the Data Base. Each new Data Type must be inserted in the Type Directory before it can be used to create a Data Element.

2.2 Glossary

Data Base A formatted VMS global section. The Data Base name is a logical name of a VMS global section or the global section name itself.

Global Section Part of memory which can be shared by several processes. Provided by VMS.

Global Section File Each global section is created as a file. Parts of the section can be mapped into a process virtual address space. Global section pages are paged to the global section file.

Data Base Area Contiguous number of pages in the Data Base (global section file).

Data Base Pool Composed of several Areas. Smallest entity which can be mapped by a process. One Pool has for one process one access mode (Write or read only). If a Pool runs out of space, one more Area is chained to that Pool.

Data Element Piece of data in the Data Base. It has a name which is kept with other information in a Directory. The data part is kept in a Pool (Area). The structure of a Data Element is described by a PL/I structure declaration.

Data Element Member Member of a Data Element structure.

Data Element Array Data Elements can be indexed in up to two dimensions. Each element of such an array has its own slot in the Directory. Therefore the data structures could be different.

Data Element Type A Data Element describing a PL/I structure. It is created from a PL/I structure declaration. Data Elements are created with that data structure by referring to the corresponding Type.

Data Element Directory Information about Data Elements, Areas, Pools, Data Types and Directories is kept in Directories.

Mount/Dismount Mount a Data Base means to create a global section. The global section file (Data Base) must exist.

Attach Attach a Data Base, Pool or Directory means to map the appropriate parts of the Data Base into the memory of the attaching process. The resulting pointers are kept in a local mapping context structure normally invisible to the user. This operation is needed for programmed access to Data Elements.

Locate Locate a Data Base, Pool, Directory or Data Element means to get the pointer to the object and/or get an identification number. This number is valid and unique during the lifetime of the object. It can be used for a fast attach or locate to get the pointer. This operation is needed for programmed access to Data Elements.

All these functions can be done by commands executed by the Data Base Manager. The commands are described in the following chapters.

2.3 GOOSY Command Interface

2.3.1 Line Input

The GOOSY command syntax is similar to the DCL syntax. A command is composed of several **keywords**. **Positional parameters are delimited by spaces**. In addition to DCL, however, they have names and may be specified in any order by `name=value`. The names are found in the help description of the command or in the menu. **Qualifiers** are preceded by a slash (`/qualifier`). Different to DCL they cannot specify a value. Qualifiers can in most cases be negated by `/NOqualifier`. Qualifier sets are mutually exclusive qualifiers. In the menu display the positional parameters are specified on top together with their names, followed by the qualifiers. At last, the qualifier sets follow. An example is shown below.

2.3.2 Command Procedures

The GOOSY command interface is able to process command files. These files must contain GOOSY commands lines. The default file type is .GCOM. The commands in these files are executed by the '@' command:

```
DBM> @file
GOOSY> @file
```

Comment lines may be written starting the line with an exclamation point (!). Lines can be continued by ending up with an hyphen (-). Arguments can be passed to the procedure. These arguments replace in the procedure text the strings &P1 (first argument) ... &P8. Example: Procedure CS.GCOM contains a command to create a spectrum:

```
CREATE SPECTRUM &P1 L &P2 /DIG
```

This procedure is called by

```
DBM> @CS SP1 0,4096
```

2.3.3 Defaults

In some cases default values are provided for command parameters. Some defaults are replaced by the input values, if specified. These parameters are called **replaceable**. **Global parameters** are valid for several commands, e.g. the Data Base name. Global parameters can be preset in the profiles. Some parameters are **required**, i.e. they are prompted if not specified. In the command descriptions these attributes are described for each parameter.

2.3.4 Conventions in the Data Base Manager

Defaults

Most of the commands of the Data Base Manager concern Data Elements. The full specification of a Data Element would be:

```
  node::base:[directory]name(index)
for example
   E::DB:[DATA]PAR(5)
   DBM> SHO SPEC MVIIA::TEST:[$SPECTRUM]S5
```

(Remote access is not yet implemented!) Data Element Members have to be specified as

```
   node::base:[directory]name(index).member
for example:
   DBM> SHO MEMBER E::DB:[DATA]PAR(5).CAL(3).ENER
```

In this example `PAR` is an indexed Data Element. To provide a more convenient specification the names for the node, base and Directory can be specified by separate parameters. These parameters are replaced by the specified values and thus defaulted in subsequent commands, as shown in the following command sequence:

```
   DBM> SHO SPEC s1 SPEC_DIR=test BASE=db
   DBM> SHO SPEC s2        ! uses Directory 'test' in Data Base 'db'
   DBM> SHO ELEM event DIR=data BASE=db
   DBM> SHO ELEM newevent  ! uses Directory 'data' in Data Base 'db'
   DBM> SHO ELEM [para]p1  ! uses Directory 'para' in Data Base 'db'
   DBM> SHO ELEM newevent  ! uses again Directory 'data' in Data Base 'db'
```

If these replaceable parameters are in addition global, i.e. valid for more then one command, they are preset in **profiles**. The profile is accessed by a logical name GOO$PROFILE. The standard GOOSY profile is GOO$EXE:PROFILE.PROF. It sets the following parameter values for the Data Base manager:

```
BASE=DB
BASEFILE=DB
AREA=
POOL=DATA
DIR=DATA
DYN_LIST=L1
DYN_TYPE=SPECTRUM
NODE=*
NAME=
LINK_FROM=
```

```
LINK_TO=
DYN_DIR=$DYNAMIC
COND_DIR=$CONDITION
SPEC_DIR=$SPECTRUM
PIC_DIR=$PICTURE
POLY_DIR=$POLYGON
CAL_DIR=$CALIB
DYN_POOL=$DYNAMIC
COND_POOL=$COND_POOL
SPEC_POOL=$SPEC_POOL
PIC_POOL=$PIC_POOL
POLY_POOL=$PIC_POOL
CAL_POOL=$CAL_POOL
PAR_DIR=DATA
```

These initial values can be reset by the commands

```
DBM> $ RESET DEFAULT
GOOSY>$DBM>$ RESET DEFAULT
```

depending if the Data Base Manager was started stand alone or as component in a GOOSY environment.

Array Values

Specification of **value arrays** for a parameter is done as a list of values separated by commas and optionally enclosed in parenthesis. E.g. spectrum limits:

```
DBM> CRE SPEC s L (0,1023,0,1023) BINS=4,8
```

Wildcards

Many commands, especially SHOW commands, provide wildcard facilities. Data Element names can be wildcarded with asterisks:

```
*   all names
x*  all names starting with string 'x'
*x  all names ending with string 'x'
x*y all names starting with string 'x' and ending with string 'y'
*x* all names containing string 'x'
```

The asterisk matches any string inclusive null string.

2.3.5 Command Example

The command `CREATE SPECTRUM` has several positional parameters and qualifiers.

```
CREATE SPECTRUM name type limits binsize
                /[NO]ERRV
                /[NO]ERRH
                /ANALOG/DIGITAL
```

The following are valid commands:

```
CRE SPEC s1 L (0,1000) 1 /DIG
CRE SPEC s1(5) L (0,1000) 1 /DIG SPEC_DIR=test
CRE SPEC [test]s1 L (0,1000,0,1000) 1 /DIG
CRE SPEC /DIG s1 L 0,1000 1
CRE SPEC s2 LIM=0,100 TYPE=R /ANA /ERRV
```

Some **invalid** commands:

```
CRE SPEC LIM=0,100 s1   ! invalid positional after named reference
CRE SPEC s1 LIM=(0 100) ! invalid space invalid delimiter in lists
CRE SPEC s1 /ANA/DIG    ! invalid exclusive qualifiers
```

2.3.6 GOOSY Menu

GOOSY menus are specific to GOOSY components. The command `MENU` enters the GOOSY component menu. Some keys are defined for a shorter menu access: As input to a GOOSY component prompt, press NEXT SCREEN. Example:

```
MDBM                       ! start Data Base Manager
SUC: DBM> <NEXT SCREEN>    ! pressing this key enters the menu.
```

There are two types of menus, one for commands and one for the command parameters which is entered when you reach a full command (commands may be composed of several keywords like in DCL). The command menu looks like

```
keyword
Subcommands available ================================================
keyword   * :  * :list of available subcommands
keyword     : short description
************** End of list ******************* End of list ***
```

The first line type is for a command which needs more subcommands, the second is the layout of an executable command. Entering a keyword of the first type, the next menu command level is displayed. Entering a keyword of the second type, the parameter menu of this command is displayed. Several keywords may be entered at once. The parameter menu looks like:

```
keyword keyword...
short description
Positional parameter list ===================================
parameter name |type|short description       =default
parameter name |type|short description       :default
Qualifier list -------------------------------------------
qualifier      |SWI |short description       :default
Qualifier set list ---------------------------------------
set name       |SET|list of possible values  :default
********* End of list ****************** End of list ********
```

In this menu one moves the cursor around using the arrow keys and can overwrite the displayed default values. Note that qualifiers always must be preceeded by a slash (`/qualifier`). An `=` sign before the default marks required parameters. Press the `KP_PF2` key to get a keypad layout. Press the `HELP` key or `KP_PF1` and `KP_PF2` to get help information for the present level. All defaults presently active are displayed in the menu. If the command is executed (by `RETURN`), the actual command line is displayed on top of the screen. You leave the component or prompter or menu by (several) `CTRL Z`. To return to previous command level without executing the current menu command, press `KP_PF4`.

2.3.7 ALIAS Names

GOOSY commands including parameter specifications can be replaced by alias names. These names are defined on two levels: global and environment. The environment level names are searched first. They are activated by the `CRENVIR` command and deactivated by the `DLENVIR` command. If no environment is active, only the global alias names are valid. Alias names cannot be abbreviated. They are implemented on GOOSY command level and DCL command level (DCL symbols). All alias names are deleted during logout. Therefore it is recommended to create alias names in the LOGIN.COM procedure using the DCL command `ALIAS CREATE`.

DCL Commands

DCL commands to handle alias names:

```
$ ALIAS CREATE name string environment /GLOBAL
$ ALIAS DELETE name environment /GLOBAL
$ ALIAS SHOW name environment /GLOBAL/ACTIVE
```

To create global alias names, omit the environment or use the `/GLOBAL` qualifier. Examples:

```
$ ALIAS CRE ANA "CREATE PROC MGOOANL"      !global alias ANA is created
$ ALIAS CRE INAC "INI ACQ /J11 NODE=" SUSI !envir.alias INAC is created
$ ALIAS SHO ENV=SUSI                       !envir.alias names of SUSI
$ ALIAS SHO                                !All active alias names
```

```
$ ALIAS SHO /GLOB                       !Global alias names only
$ ALIAS SHO INAC                        !Alias INAC
```

GOOSY Commands

Sometimes it is useful to create alias names from GOOSY command level. GOOSY commands to create, delete and show alias names are implemented in the GOOSY prompter, in the standalone Data Base Manager and the stand alone display. The arguments are the same as shown above, but the commands begin with the verb to fit into the GOOSY commands.:

```
DBM> CREATE ALIAS name "string"
GOOSY> SHOW ALIAS
DISP> DELETE ALIAS name
```

Alias names created by GOOSY commands are not defined as DCL symbols.

Chapter 3

Data Base Manager

3.1 Data Base Manager Program

The Data Base Manager is invoked stand alone by the DCL command:

```
$ MDBM                      ! start DBM
SUC:DBM> <NEXT SCREEN>      ! pressing this key enters the menu.
```

A Data Base should have been created already, e.g. by CREDB.

3.1.1 Data Base Manager Menu

The top menu level looks like:

```
Subcommands available ================================================
$ *          :  * :ATTACH,CALL,DCL,DEBUG,DEFINE,DIRECTORY,EXECUTE,EXIT,..
CALCULATE *  :  * :SPECTRUM
CALIBRATE *  :  * :SPECTRUM
CLEAR *      :  * :CAMAC,CONDITION,ELEMENT,PICTURE,SPECTRUM
COPY *       :  * :CONDITION,ELEMENT,MEMBER,POLYGON,SPECTRUM
CREATE *     :  * :ALIAS,AREA,BASE,CALIBRATION,CONDITION,DIRECTORY,...
DECALIBRATE  :  * :SPECTRUM
DELETE *     :  * :ALIAS,CONDITION,DYNAMIC,ELEMENT,LINK,POOL,..
DISMOUNT *   :  * :BASE
DUMP *       :  * :SPECTRUM
FREEZE *     :  * :CONDITION,SPECTRUM
HELP         : Access VMS HELP facility
LOCATE *     :  * :BASE,DIRECTORY,ELEMENT,ID,POOL,QUEUEELEMENT,TYPE
MENU         : Enter menu
MODIFY *     :  * :DIRECTORY,FRAME,TABLE
MOUNT *      :  * :BASE
READ *       :  * :CAMAC
SET *        :  * :CONDITION,LETTERING,MEMBER,SPECTRUM
SHOW *       :  * :ALIAS,AREA,CALIBRATION,CAMAC,CONDITION,DIRECTORY,...
START *      :  * :MR2000
STOP *       :  * :MR2000
SUMUP *      :  * :SPECTRUM
UNFREEZE *   :  * :CONDITION,SPECTRUM
UPDATE *     :  * :BASE,DYNAMIC
WRITE *      :  * :CAMAC
************** End of list ******************* End of list ***
Command:
 PF2: Help, PF3: Enter command, PF4: Break, ENTER: Previous menu
Subcommand :
```

3.2 Data Base Manager Component

3.2.1 The GOOSY Environment

As shown above, the Data Base Manager may be executed standalone on DCL level. It may, however, also be bundled together with other GOOSY programs in an **environment**. In an environment several components are started together. Commands to be executed by environment components are dispatched by the GOOSY prompter from one terminal. Therefore, on each terminal one has to create an environment. Commands given from that terminal are executed by the components running in that environment. To create an environment with optional components, use the DCL command `CRENVIR`. The Data Base Manager is started by the qualifiers `/ONLINE` or `/OFFLINE` or `/$DBM`.

```
$ CRENVIR ?                       ! Enter menu
$ CRENVIR environment /ONLINE     ! All GOOSY components
$ CRENVIR environment /OFFLINE    ! All except Transport Manager
$ CRENVIR environment /$DBM       ! Data Base Manager only
```

The environment name must be unique within a user group on one VAX node. It can be one to four characters long.
Similar, the DCL command

```
$ DLENVIR
```

deletes the present environment including all components (subprocesses).

3.2.2 The GOOSY Prompter

The GOOSY prompter is entered by the DCL command `GOOSY`:

```
$ GOOSY
SUC:GOOSY>
```

Now you can enter any GOOSY command. You leave the GOOSY prompter by typing CTRL Z. Single GOOSY commands can be given under DCL by a preceding `GOOSY` or just `G`. This allows to execute all GOOSY commands as DCL commands and to replace DCL symbols in the GOOSY command line.

The GOOSY prompter interprets some special function keys. The definitions are made in GOO$EXE: INI_TP0.COM. A help setup is displayed by the KP_PF2 key.

Now, entering the GOOSY prompter, the menu for the Data Manager component is entered by:

```
<KP_MINUS> ! $DBM : Data Base Manager commands menu
```

Similar, the Data Manager component can be deleted and created by special keys:

```
<PF1><KP_COMMA>  ! delete $DBM : Data Base Manager
<KP_COMMA>       ! create $DBM : Data Base Manager
```

The Data Base Manager is not necessary needed to acquire or analyze data. It may be restarted at any time without affecting other GOOSY components in the environment.

Chapter 4

Data Management Commands

4.1 Data Base Commands

In the following some commands used for general purpose are discussed. The commands can be given to the DBM> prompt or to the GOOSY> prompt if an environment with $DBM component is created. Note that in the following descriptions lower case names have to be replaced by meaningfull values. Uppercase names are keywords and must be typed as shown.

4.1.1 Data Base

Any Data Base must be created once. During the creation of a Data Base the Data Base file is created. In addition, a Data Base is a VMS global section. Therefore the file must be made a global section file. This is called **mounting** the Data Base. The reverse is to **dismount** the Data Base. This means that the file is no longer a global section file, i.e. it can be deleted or copied. The Data Base must be mounted before any of the commands described in the following is executed. It can be mounted only on one VMS node, because a Data Base can be shared in memory by several programs. The Data Base file must be accessible on the node where it is mounted. In a VAX cluster that means that the Data Base must be dismounted on one node before it can be mounted on another node. The file is accessible from all cluster nodes and needs not to be copied. All Data Bases remain mounted until they are dismounted or the VAX operating system is rebooted.

Create/Delete a Data Base

To create a Data Base one may use the command

```
DBM> CREATE BASE base basefile ade mde pde tde basepages
```

After that the Data Base is mounted. It is recommended to give the file the same name as the Data Base with the file type .SEC. For the Data Base name a VMS logical name may be created and used in all other commands. The four parameters **ade, mde, pde, tde** specify the number of Areas, Directories, Pools and Data Types which can be created in the Data Base. The parameter basepages specifies the size of the Data Base in pages (512 bytes). These parameters cannot be extended later. A DCL command is provided to create GOOSY Data Bases:

```
$ CREDB ?             ! Enter menu
$ CREDB basename filename size[KB]
       /SPECTRA=s     ! maximum number of spectra
       /PICTURE=p      ! maximum number of picture frames
       /CONDITIONS=c   ! maximum number of conditions
       /DIRECTORIES=d ! maximum number of Directories
       /POOLS=p        ! maximum number of Pools
       /POLYGON=p      ! maximum number of polygons
       /DYNLIST=d      ! maximum number of Dynamic Lists
```

This command creates the Directories $SPECTRUM, $CONDITION, $PICTURE, $POLYGON, $DYNAMIC, DATA and the Pools $SPEC_POOL, $COND_POOL, $PIC_POOL, $DYNAMIC and DATA. The command procedure creating the GOOSY Data Base can be saved by qualifier `/SAVE=file`. This file may be edited and used later instead of `CREDB`.

Mount/Dismount a Data Base

When a Data Base has been created, it is mounted. A Data Base is dismounted by

```
   DBM> DISMOUNT BASE base
or in DCL by
   $ DELGS name
```

After that the Data Base file can be copied, deleted or dumped, but the Data Base cannot be accessed by commands. To mount the Data Base again, use

```
   DBM> MOUNT BASE base basefile
```

Compress/Expand a Data Base

If a Data Base is not used for some time, it can be compressed to save disk space. The compressed Data Base file contains no zeros. Before a Data Base can be mounted on this file, the file must be expanded.

```
    $ MUTIL COMPRESS BASE base basefile file
```

After that the Data Base is dismounted except the `/MOUNT` qualifier was given. The file 'file' must not exist. To expand a compressed Data Base file use

```
    $ MUTIL DECOMPRESS BASE file base basefile
```

The file 'basefile' must not exist. The Data Base 'base' must not be mounted. After that command the Data Base is not mounted except the `/MOUNT` qualifier was given. The default file type for compressed files is .CSEC.

4.1.2 Data Base Pool

The data part of the Data Elements is allocated in Pools. Pools are composed of chained Areas. Areas are sections of contiguous pages of the Data Base. The Pool size is no limit of the Pool, because it is extended automatically by adding one more Area to the Pool. One should at least specify the size of the largest Data Element to be allocated in the Pool.

```
   DBM> CREATE POOL pool size                ! size in bytes
```

If the Pool size is chosen too small for the Data Elements to be allocated in the Pool, one Area is created for each Data Element. This costs entries in the Area Directory and time to map the Pool. If the size is too big, it wastes space in the Data Base. The command `SHOW POOL` shows a list of existing Pools, the Areas of a Pool and the usage of the Areas.

4.1.3 Data Base Area

As mentioned above, Areas are parts of Pools. The user is normally not involved with Areas. Areas are created automatically. But if Pools are chosen too small, many Areas are created and the Area Directory may be too small. To check that, the command

```
DBM> SHO AREA * /DIR
```

shows a list of all Areas created and

```
DBM> SHO AREA area /FULL
```

shows the usage of an Area.

4.1.4 Data Base Directory

Each Data Element in the Data Base uses one entry slot in a Directory. The information kept in the Directory is used to access the Data Element. The name of the Data Element, the Pool where the data part is allocated and links to other Data Elements are kept in the Directory entry slot.

```
DBM> CREATE DIRECTORY directory entries
```

Creating Directories one should know that each member of a Data Element name array takes one entry in the Directory. Some GOOSY Data Elements take more than one entry, i.e. spectra four, conditions two, composed conditions three and pictures one per picture plus one per frame. The command `SHOW DIRECTORY` shows a list of all Directories and the content and usage of a Directory.

Expand a Data Base Directory

If a Directory has no more free slots, it can be expanded by

```
DBM> MODIFY DIRECTORY directory entries
```

It may happen that there is no more space in the Directory even if there are still free entries. This can be caused by very many links. The columns 'Used bytes for links' and 'Free bytes' in the output of `SHOW DIRECTORY /FULL` show this situation. If there are less than 100 bytes free, the Directory should be modified to a bigger size. A Directory can be deleted only if there are no used entries in it.

4.1.5 Data Element Declarations

Before creating a Data Element, one must specify the structure declaration. This is done by a PL/I structure declaration. This declaration must be in a file or text library module. The name of the file or library module must be the name of the structure. It must made known to the Data Base. This is done by `CREATE TYPE`:

```
DBM> CREATE TYPE @library(module) ! Declaration from library
DBM> CREATE TYPE @filename        ! Declaration from file
```

The same file or library module may be included in a PL/I program to access the Data Element. Note that the structure should be declared as `BASED`. The default base pointer may be declared in the same file or library module. In PL/I programs the pointer to the Data Element is returned by a Data Management routine or macro. A Type declaration can be deleted only if there are no Data Elements referring to that type. The Type declarations known can be obtained by

```
DBM> SHOW DIRECTORY $TYPE /FULL
```

Note the default declarations for 'simple' Data Elements:

```
H  for  BIN FIXED(7)
I  for  BIN FIXED(15)
L  for  BIN FIXED(31)
R  for  BIN FLOAT(24)
D  for  BIN FLOAT(53)
C  for  CHARACTER
V  for  CHARACTER VAR
B  for  BIT ALIGNED
```

These Types may be used to create simple Data Elements composed of one member or array. A specific declaration can be output by

```
DBM> SHOW TYPE type
```

where 'type' is the name of the structure.

4.1.6 Data Elements

Create Data Elements

There are two kinds of Data Elements: **simple** and **complex**. Simple Data Elements are like non structured PL/I variables or arrays. Complex Data Elements are like PL/I structures. Simple Data Elements can be created without Type declarations. The Type parameter specifies the Data Type (H,I,L,R,D,C,V,B) and optionally the array dimensions:

```
DBM> CREATE ELEMENT name pool DIR=directory TYPE=datatype
DBM> CREATE ELEMENT e1 POOL=data DIR=data TYPE=L
DBM> CREATE ELEMENT e2 POOL=data DIR=data TYPE=L(16)
```

Note that Directory and Pool must exist. Complex Data Elements are described by Type declarations which must be created in the Data Base. The declarations used in the `CREATE ELEMENT` command must not contain REFER values.

```
DBM> CREATE TYPE @library(struc)       ! Create Type declaration
DBM> CREATE ELEMENT e1 POOL=data DIR=data TYPE=struc
```

The space allocated in the Pools by Data Elements is controlled by bitmaps. Each bit in the bitmap represents a contiguous number of bytes. This number can be specified together with the `CREATE ELEMENT` command.

```
DBM> CREATE ELEMENT ... CLUSTER=bytes
```

This number should be a fraction of the smallest Data Element allocated in the same Pool. If the number is too big, that means that space is wasted because it is the smallest entity allocatable in the Pool. If it is too small space is wasted because of the size of the bitmap. The number need not to be the same for all Data Elements in the Pool. Only Data Elements with the same clustersize, however, can be allocated in the same Area, i.e. there are at least as many Areas in a Pool as Data Elements with different cluster sizes.

Data Element Name Arrays

Data Elements can be indexed, i.e. to build arrays of Data Elements. The index is added to the Data Element name.

```
DBM> CREATE ELEMENT ea1(20) TYPE=x ...
```

creates 20 Data Elements 'ea1' of Type 'x' using 20 entries in the Directory. All array members have the identical Data Type and Pool during the creation of the Data Element array. To vary single array members use the `CREATE` command with the `/REPLACE` option after the whole Data Element name array was created, e.g.

```
DBM> CREATE ELEMENT ea1(3) POOL=data TYPE=type /REPLACE
```

A single element of such a Data Element name array can be referenced in SHOW commands, but not in DELETE commands. One can only delete the whole data element name array.

Delete Data Elements

Data Elements can be deleted if there are no links on the Data Element (see below) and if no program has access to the Data Element. In the second case an error message will be displayed telling that there are 'locks' for that Data Element. Specifying the name of a Data Element name array the whole array is deleted.

```
DBM> DELETE ELEMENT name DIR=directory
DBM> DELETE ELEMENT [directory]name
```

When the Directory is specified as a separate parameter it is used in subsequent commands as default. When specified as part of the name it is used only temporary.

Data Element Links

Data Element links are used by the data management to protect Data Elements referring each other. In the future they may be used to execute commands for arbitrary groups of Data Elements linked together. A link is always directed from one Data Element to another Data Element which could be the same. As long as there are outgoing or incoming links a Data Element cannot be deleted. The user is normally not involved in the link management. The `SHOW LINKS` command can be used to see the links of a Data Element. One never should delete links if one does not understand very well the situation. If Data Elements cannot be deleted because of links there is normally a reason for that.

4.1.7 Data Base Usage

As mentioned in the previous sections there are many `SHOW` commands to get information about the usage of a Data Base. These are discussed in more detail in the following sections.

SHOW HOMEBLOCK

All pages in the Data Base are controlled by a main bitmap. For each page allocated by an Area one bit in the bitmap is set. Thus freed pages can be used again and the total usage and fragmentation of the Data Base can be determined at any time. The command

```
DBM> SHOW HOMEBLOCK
```

shows the overall usage of the Data Base.

SHOW AREA

Similar the usage of each Area is controlled by Area bitmaps. A list of all Areas and detailed information about an Area is obtained by

```
DBM> SHOW AREA * /DIR    ! Usage of Area Directory
DBM> SHOW AREA *         ! Usage of known Areas
DBM> SHOW AREA xyz /FULL ! Show information and bitmap of Area xyz
```

SHOW POOL

This command outputs a list of all known Pools or detailed information about one specific Pool

```
DBM> SHOW POOL /DIR      ! Usage of Pool Directory
DBM> SHOW POOL /DIR /FULL! Usage of Pool Directory with all entries
DBM> SHOW POOL *         ! Usage of known Pools
DBM> SHOW POOL xyz       ! Show Areas of Pool 'xyz'
DBM> SHOW POOL xyz /FULL ! Show full information about
                         !  all Areas of Pool 'xyz'
```

SHOW DIRECTORY

This command outputs a list of all known Directories or detailed information about one Directory:

```
DBM> SHOW DIRECTORY *          ! list of known Directories
DBM> SHOW DIRECTORY xyz /FULL ! Show entries of Directory xyz
```

The entries of a Directory may show Data Elements without names (name=***). These Data Elements are queued to a master Data Element (Queue Header). They can be accessed only via their queue header. All GOOSY Data Elements like spectra and conditions use this mechanism. A spectrum is composed of three Data Elements queued to the spectrum header Data Element. The data part of queued Data Elements can be allocated in different Pools. Thus the different parts of a spectrum could be accessed in a program by different access modes, i.e. read only or read/write.

Another feature can be seen by this command. Data Elements can be created as name arrays, i.e. the Data Element name is indexed. For each element of such an array one entry in the Directory is used.

The order in the output of the `SHOW DIRECTORY` command is not alphabetically. To get an alphabetic order one should use the command `SHOW ELEMENT`.

SHOW ELEMENT

Any Data Element, regardless of its structure (Type), can be displayed. The information stored in the Directory entry or the information stored in the data part can be selected:

```
DBM> SHOW ELEMENT name
DBM> SHOW ELEMENT [directory]* ! Data Element list
DBM> SHOW ELEMENT name /FULL   ! Directory information
DBM> SHOW ELEMENT name /DATA   ! Data part of the Data Element
```

SHOW MEMBER

Similar to PL/I the smallest entity of a structure is called a Member. A Data Element Member can be set to a value and the current value can be output. This is done by the commands:

```
DBM> SET  MEMBER [directory]elementname.member value
DBM> SHOW MEMBER [directory]elementname.member
```

The expression 'member' may be a list of several Members separated by periods depending on the structure.

Chapter 5

GOOSY Data Elements

There are some GOOSY Data Elements which are handled by special commands. These are described in the next sections.

5.1 Conditions

By default, conditions are kept in the Directory $CONDITION. GOOSY conditions are independent of spectra or coordinates (parameters). All kinds of conditions can be executed in a Dynamic List. In an analysis routine they are executed by the macro $COND. They may then be used as filters for spectrum accumulation and/or scatter plots. Each condition has TRUE and FALSE counters and freeze, result, and preset bits.

5.1.1 Related Commands

```
CREATE   CONDITION type
DELETE   CONDITION
SHOW     CONDITION
CLEAR    CONDITION COUNTER
COPY     CONDITION
SET      CONDITION type
FREEZE   CONDITION
UNFREEZE CONDITION
```

Each condition takes two entries in Directory $CONDITION, except composed conditions which take three. The first Data Element is a condition header common to all conditions, the second keeps specific information. The default Pool is $COND_POOL.

5.1.2 Condition Header Data Element

The declaration of the header is kept in library GOOTYP(SE$COHE).

```
/* ----------------- start of SE$COHE ---------------------------*/
DCL 1 SE$COHE BASED,                          /* condition header  */
      2 BE$COHE_ATTR BIT(32) ALIGNED,         /* flags             */
      2 CVE$COHE_NAME CHAR(30) VAR ,          /* name redundant    */
      2 CE$COHE_TYPE CHAR(1),                 /* summarizes Type   */
      2 CVE$COHE_DATE_TIME_CRE CHAR(23) VAR,/* creation date     */
      2 CVE$COHE_CREATOR CHAR(63) VAR,        /* Creating program  */
      2 LE$COHE_COTAB_DIR_ID BIN FIXED(31), /* table Directory   */
      2 LE$COHE_COTAB_DE_ID BIN FIXED(31),  /* table index       */
      2 LE$COHE_ID_IN_COTAB BIN FIXED(31);  /* cond.table index  */
/* ------------------- end of SE$COHE ---------------------------*/
```

The different kinds of conditions are:

5.1.3 Window Conditions

Data Elements

Window conditions are implemented in two Data Elements. The declarations are kept in library GOOTYP(SE$COHE) and GOOTYP(SE$COWI).

```
/* ----------------- start of SE$COWI ---------------------------*/
DCL 1 SE$COWI BASED,                          /* window condition  */
      2 LE$COWI_TRUE_CT BIN FIXED(31) ,       /* true count        */
      2 LE$COWI_FALSE_CT BIN FIXED(31) ,      /* test count        */
      2 LE$COWI_DIM BIN FIXED(31) ,           /* dimension         */
      2 RE$COWI_LIMITS_LOW(1 REFER (LE$COWI_DIM)) BIN FLOAT(24),
    /* limits            */
      2 RE$COWI_LIMITS_UP(1 REFER (LE$COWI_DIM)) BIN FLOAT(24);
/* ------------------- end of SE$COWI ---------------------------*/
```

A window condition keeps n window limits (subwindows) forming an n-dimensional cube. Points inside the cube are true, outside false. Up to eight subwindows may be used in a Dynamic List, up to four in macro $COND. Each limit pair may be applied to a different object. Object may be any Member of a Data Element which is a BIN FLOAT(24), BIN FIXED(31) or BIN FIXED(15) number. The condition is TRUE if all subwindows are TRUE.

Create Window Conditions

```
DBM> CRE COND WINDOW c (1,1000) 1          ! 1 subwindow
DBM> CRE COND WINDOW c (1,100,1,200)       ! 2 subwindows
DBM> CRE COND WINDOW c (1,1000) 2          ! 2 subwindows
                                           ! both (1,1000)
DBM> CRE COND WINDOW c(10) (1,100)         ! 10 cond., 1 subw. each
```

Set Window Condition Limits

In the Data Base Manager the limits of window conditions can be set only by values. To specify the limits by graphic input, use the command `REPLACE CONDITION WINDOW` which is executed in the display program.

```
DBM> SET CONDITION WINDOW condition limits dimension
DBM> SET COND WINDOW c (1,1000) 1          ! 1. subwindow
DBM> SET COND WINDOW c 1,100 2             ! 2. subwindows
```

5.1.4 Multiwindow Conditions

Multiwindows are composed of the same Data Elements as normal windows. The difference to normal window conditions is that there is one object for all subwindows and one result bit for each subwindow. In a Dynamic List or macro any number of subwindows is processed. The result bits can be used as filters for spectrum array accumulation. The number of the last TRUE subwindow may be used to select a spectrum array member for accumulation (See MULTIWINDOW and INDEXEDSPECTRUM in Dynamic Lists). In macro $COND two execution modes can be selected: $COND(MW,...) executes all subwindows returning all bits and the index of the last true subwindow. $COND(MWI,...) stops execution after the first true subwindow check. If only the index is needed this is a faster mode.

Create Multiwindow Conditions

```
DBM> CRE COND MULTI  c (1,1000) 100        ! 100 subwindows
```

Set Multiwindow Condition Limits

In the Data Base Manager the limits of window conditions can be set only by values. To specify the limits by graphic input, use the command `REPLACE CONDITION WINDOW` which is executed in the display program.

```
DBM> SET CONDITION WINDOW condition limits dimension
DBM> SET COND WINDOW c (1,1000) 1          ! 1. subwindow
DBM> SET COND WINDOW c 1,100 2             ! 2. subwindows
```

5.1.5 Pattern Conditions

Data Elements

Pattern conditions are implemented in two Data Elements. The declarations are kept in library GOOTYP(SE$COHE) and GOOTYP(SE$COPA).

```
/* ------------ start of SE$COPA ---------------------------------- */
DCL 1 SE$COPA BASED ,                       /* pattern condition     */
      2 LE$COPA_TRUE_CT BIN FIXED(31) ,     /* true count            */
      2 LE$COPA_FALSE_CT BIN FIXED(31) ,    /* test count            */
      2 BE$COPA_MODE BIT(16) ALIGNED,       /* Mode                  */
      2 LE$COPA_DIM  BIN FIXED(31),         /* dimension             */
      2 SE$COPA_PAT (1 REFER(LE$COPA_DIM)),
        3 BE$COPA_PATT BIT(32) ALIGNED,     /* test pattern          */
        3 BE$COPA_INV  BIT(32) ALIGNED;     /* invert pattern        */
/* -------------- end of SE$COPA ---------------------------------- */
```

Similar to the windows, the pattern conditions may keep n subpatterns. Up to eight may be checked in a Dynamic List and up to four in macro $COND. Each subpattern is compared to a different object which can be any Data Element Member of Type BIT(16) or BIT(32) ALIGNED. The condition is TRUE if all subpatterns match. There are four matching modes:

1. IDENT
   Pattern and object must be identical.
2. ANY
   Pattern and object must have at least one common bit set.
3. INCL
   TRUE if all bits set in the pattern are set in the object (like IDENT inclusive additional bits set only in the object).
4. EXCL
   TRUE if all bits set in the object are set in the pattern (like ANY exclusive additional bits set only in the object).

In addition single bits in the objects can be inverted before testing.

Create Pattern Conditions

```
DBM> CRE COND PATTERN name pattern dimension INVERT=pattern
DBM> CRE COND PATTERN c '1'B               ! 1 subpattern
                                            ! padded right with 0
DBM> CRE COND PATTERN c '1'B 2             ! 2 subpatterns 32 bit each
DBM> CRE COND PATTERN c '1'B INV='1'B /ANY ! invert first bit
DBM> CRE COND PATTERN c '11111'B /IDENT    ! identical match
```

Set Pattern Condition Patterns

```
DBM> SET COND PATTERN name pattern invpat index
DBM> SET COND PATTERN c '1'B INDEX=2        ! 2. subpattern
DBM> SET COND PATTERN c '1'B '1'B           ! invert first bit
```

5.1.6 Function Conditions

Data Elements

Function conditions are implemented in two Data Elements. The declarations are kept in library GOOTYP(SE$COHE) and GOOTYP(SE$COFU).

```
/* ------------ start of SE$COFU -------------------------------------*/
DCL 1 SE$COFU BASED, /* function condition counters and attributes    */
        2 LE$COFU_TRUE_CT BIN FIXED(31) , /* true count */
        2 LE$COFU_FALSE_CT BIN FIXED(31) , /* test count */
        2 CVE$COFU_IMAGE CHAR(14) VAR,  /* sharable image logical name*/
        2 CVE$COFU_MODULE CHAR(30) VAR, /* Module name                */
        2 LE$COFU_DIM  BIN FIXED(31),   /* number of arguments        */
        2 SE$COFU_ARGS (1 REFER(LE$COFU_DIM)),
          3 CVE$COFU_BASE CHAR(14) VAR,    /* Base of argument */
          3  LE$COFU_DIR_ID BIN FIXED(31), /* Directory index  */
          3  LE$COFU_ARG_ID BIN FIXED(31), /* index            */
          3  LE$COFU_ARG_VER BIN FIXED(31);/* version          */
/* -------------- end of SE$COFU ----------------------------- */
```

The user may write his own routines for more complex conditions. These routines must be linked in a sharable image (DCL command `LSHARIM`) and can then be dynamically loaded. In the Dynamic List any members of Data Elements may be specified as arguments for these routines. The first argument, however, must be a BIT(8) ALIGNED returning the result.

Create Function Conditions

```
DBM> CRE COND FUNC c
```

NOTE: presently the image, module and the arguments can be specified only in the Dynamic List Entry, but not in the condition.

5.1.7 Polygon Conditions

Data Elements

Polygon conditions are implemented in two Data Elements. The declarations are kept in library GOOTYP(SE$COHE) and GOOTYP(SE$COPO).

```
/* ------------- start of SE$COPO ---------------------------------- */
DECLARE
 1 SE$COPO BASED,                    /* polygon condition attribute  */
   2 LE$COPO_TRUE_CT BIN FIXED(31) , /* true count                   */
   2 LE$COPO_FALSE_CT BIN FIXED(31) ,/* test count                   */
```

```
   2 CVE$COPO_BASE CHAR(14) VAR,     /* base of polygon              */
   2 RE$COPO_FACTOR BIN FLOAT(24),   /* factor for objects           */
   2 RE$COPO_OFFSET BIN FLOAT(24),   /* offset for objects           */
   2 RE$COPO_BINSIZE BIN FLOAT(24),  /* binsize for table            */
   2 LE$COPO_DIR_ID BIN FIXED(31),   /* polygon Directory index      */
   2 LE$COPO_POLH_ID BIN FIXED(31),  /* polygon header index         */
   2 LE$COPO_POLD_ID BIN FIXED(31);  /* polygon data index           */
/* -------------- end of  SE$COPO  -------------------------------- */
```

A polygon is created and modified independent of polygon conditions. Therefore several polygon conditions may reference the same polygon, but with different objects (coordinates). The polygon and objects are bound to the condition either by creation or by inserting in the Dynamic List or by macro call. The execution time is similar to window conditions.

Create Polygon Conditions

```
   DBM> CRE COND POLY c polygon        ! polygon must exist
```

The polygon referenced by a condition can be modified by command `REPLACE POLYGON` which is executed in the display program.

5.1.8 Composed Conditions

Data Elements

Composed conditions are implemented in three Data Elements. The declarations are kept in library GOOTYP(SE$COHE), GOOTYP(SE$COCO) and GOOTYP(SE$COCOT).

```
/* ------------- start of  SE$COCO ------------------------------- */
DECLARE
 1 SE$COCO BASED,                     /* composed condition         */
   2 LE$COCO_TRUE_CT BIN FIXED(31) ,  /* true count                 */
   2 LE$COCO_FALSE_CT BIN FIXED(31) , /* test count                 */
   2 CVE$COCO_SPEC CHAR(62) VAR,      /* Condition spec.string      */
   2 LE$COCO_DIM BIN FIXED(31),       /* number of conditions       */
   2 SE$COCO_LIST (1 REFER(LE$COCO_DIM)),
     3 CVE$COCO_BASE CHAR(14) VAR,    /* base of condition          */
     3  LE$COCO_DIR_ID BIN FIXED(31), /* Directory index            */
     3  LE$COCO_CON_ID BIN FIXED(31), /* condition index            */
     3  LE$COCO_CON_VER BIN FIXED(31); /* condition version         */
/* -------------- end of  SE$COCO -------------------------------- */
/* ------------- start of SE$COCOT ------------------------------- */
DECLARE
 1 SE$COCOT BASED,                   /* result table               */
```

```
   2 LE$COCOT_TABSIZE BIN FIXED(31),  /*  2**(LE$COCO_NO)-1          */
   2 BE$COCOT_TABLE (0:1 REFER(LE$COCOT_TABSIZE)) BIT(1) ALIGNED;
                                  /* look-up table for complex cond.
                                       has 2**(LE$COCO_DIM) entries */
/* -------------- end of  SE$COCOT ---------------------------------- */
```

This may be any boolean expression of other conditions.

Create Composed Conditions

```
DBM> CRE COND COMP c "a | (x & y)"  ! conditions a, x, y must exist
```

5.2 Polygons

5.2.1 Related Commands

```
CREATE  POLYGON
DELETE  POLYGON
SHOW    POLYGON
COPY    POLYGON
REPLACE POLYGON
DISPLAY POLYGON
```

5.2.2 Data Elements

Polygons are implemented in two Data Elements. The declarations are kept in library GOOTYP(SE$POHE) and GOOTYP(SE$PODAT).

```
/* --------------- begin of SE$POHE  ------------------------ */
DCL 1 SE$POHE BASED,
    2 LE$POHE_VERSION BIN FIXED(31), /* version number        */
    2 LE$POHE_DATA_ID BIN FIXED(31), /* index of SE$PODAT     */
    2 LE$POHE_POINTS  BIN FIXED(31), /* number of points      */
    2 LE$POHE_USED    BIN FIXED(31), /* number of used points */
    2 BE$POHE_FLAGS   BIT(32) ALIGNED; /* flags               */
/* ---------------- end of  SE$POHE ------------------------- */
/* --------------- begin of SE$PODAT ------------------------ */
DCL L_SE$PODAT_points BIN FIXED(31);
DCL 1 SE$PODAT BASED,
    2 RE$PODAT_xmin  BIN FLOAT(24), /* range                  */
    2 RE$PODAT_xmax  BIN FLOAT(24), /* range                  */
    2 RE$PODAT_ymin  BIN FLOAT(24), /* range                  */
    2 RE$PODAT_ymax  BIN FLOAT(24), /* range                  */
    2 LE$PODAT_points BIN FIXED(31),/* number of points       */
    2 LE$PODAT_used  BIN FIXED(31), /* number of used points  */
    2 RE$PODAT_x(L_SE$PODAT_points   REFER(LE$PODAT_points)) BIN FLOAT(24),
    2 RE$PODAT_y(L_SE$PODAT_points   REFER(LE$PODAT_points)) BIN FLOAT(24),
    2 HE$PODAT_key(L_SE$PODAT_points REFER(LE$PODAT_points)) BIN FIXED(7);
/* --------------- end of SE$PODAT -------------------------- */
```

Polygons may be created, displayed, modified, copied and deleted. They can be specified by graphic input or numerically. They are used by one or more conditions.

5.2.3 Create Polygons

```
DBM> CREATE POLYGON polygon points
```

The parameter 'points' defines the initial size of the polygon. If the polygon is modified, this size is extended automatically if needed.

5.2.4 Modify Polygons

Polygons can be modified only in the display program. If a two-dimensional spectrum or a scatterplot is displayed, the polygon can be edited by cursor input.

```
DISP> REPLACE POLYGON polygon frame
```

where 'frame' must be specified, if there are more than one frame on the screen. The points of the polygon can be entered by values, too.

```
DISP> REPLACE POLYGON polygon X=(x1,x2,x3,...) Y=(y1,y2,y3,...)
```

5.3 Spectra

5.3.1 Related Commands

```
CREATE        SPECTRUM
DELETE        SPECTRUM
SHOW          SPECTRUM
COPY          SPECTRUM
SUMUP         SPECTRUM
DISPLAY       SPECTRUM
FREEZE        SPECTRUM
UNFREEZE      SPECTRUM
CALCULATE     SPECTRUM
[DE]CALIBRATE SPECTRUM
CLEAR         SPECTRUM
DUMP          SPECTRUM
FIT           SPECTRUM
```

5.3.2 Data Elements

Spectra are implemented in four Data Elements. The declarations are kept in library GOOTYP(SE$SPHE), GOOTYP(SE$SPHED), GOOTYP(SE$SPDTT), and GOOTYP(SE$SPDTD) where t is I,L or R and d is 1 or 2.

```
/* -------------------- start of SE$SPHE ------------------------ */
DCL 1 SE$SPHE BASED,                    /* spectrum header        */
      2 BE$SPHE_ATTR BIT(32) ALIGNED,   /* attribute flags        */
      2 CVE$SPHE_NAME CHAR(30) VAR,     /* name                   */
      2 LE$SPHE_DIM BIN FIXED(31) ,     /* no of dimensions       */
      2 CE$SPHE_DTYPE CHAR(1),          /* Data Type of spectrum  */
      2 CVE$SPHE_DATE_TIME_CRE CHAR(23) VAR, /* creation date     */
      2 CVE$SPHE_CREATOR CHAR(62) VAR,
      2 LE$SPHE_SPTAB_DIR_ID BIN FIXED(31), /* table Directory    */
      2 LE$SPHE_SPTAB_DE_ID BIN FIXED(31),  /* table index        */
      2 LE$SPHE_ID_IN_SPTAB BIN FIXED(31);  /* spectrum index     */
/* ---------------------- end of SE$SPHE ------------------------ */
/* -------------------- start of SE$SPHED ----------------------- */
DCL 1 SE$SPHED BASED,
      2 BE$SPHED_ATTR BIT(32) ALIGNED;/* attribute flags longword */
/* ---------------------- end of SE$SPHED ----------------------- */
/* -------------------- start of SE$SPDTT ----------------------- */
DCL 1 SE$SPDTT BASED,
```

```
      2 BE$SPDTT_FLAGS BIT(32) ALIGNED,     /* flag table         */
      2 CVE$SPDTT_LETTERING_0 CHAR(80) VAR, /* first lettering    */
      2 IE$SPDTT_DIM_DATA BIN FIXED(15) ,   /* dimension          */
      2 SE$SPDTT_DATA (1 REFER(IE$SPDTT_DIM_DATA)),
        3 CVE$SPDTT_LETTERING CHAR(80) VAR,     /* lettering      */
        3 RE$SPDTT_LIMITS_LOW BIN FLOAT(24),    /* lower limit    */
        3 RE$SPDTT_LIMITS_UP BIN FLOAT(24),     /* upper limit    */
        3 LE$SPDTT_NO_BINS BIN FIXED(31),       /* no of bins     */
        3 RE$SPDTT_LIMITS_BIN BIN FLOAT(24),    /* bin size       */
        3 RE$SPDTT_FACTOR  BIN FLOAT(24),       /* coord. to ch.  */
        3 RE$SPDTT_OFFSET  BIN FLOAT(24);       /* coord. to ch.  */
/* ------------------- end  of SE$SPDTT ------------------------- */
/* ------------------ start of SE$SPDL1     --------------------- */
/* example of data structure of                                  */
/*  one dimensional FOUR bytes integer spectrum                  */
/* ------------------------------------------------------------- */
DCL 1 SE$SPDL1 BASED, /* one dimensional scalar spectrum         */
      2 LE$SPDL1_COUNTS BIN FIXED(31),            /* sum of counts      */
      2 DE$SPDL1_CONTENTS BIN FLOAT(53),          /* contents           */
      2 LE$SPDL1_OUTLIM_UP_COUNTS BIN FIXED(31),  /* overflow counts    */
      2 LE$SPDL1_OUTLIM_LOW_COUNTS BIN FIXED(31), /* underflow counts   */
      2 DE$SPDL1_OUTLIM_UP_CONTENTS BIN FLOAT(53), /* overflow contents  */
      2 DE$SPDL1_OUTLIM_LOW_CONTENTS BIN FLOAT(53),/* underflow contents */
      2 LE$SPDL1_DIM1 BIN FIXED(31),              /* channels dim 1     */
      2 LE$SPDL1_DATA (1 REFER(LE$SPDL1_DIM1)) BIN FIXED(31);
/* -------------------- end of SE$SPDL1    ---------------------- */
```

By default, spectra are kept in the Directory $SPECTRUM. Each spectrum takes four entries. The user need not be concerned with that, but in a SHOW DIRECTORY command these Data Elements will be listed. Spectra may be BIN FIXED(15), BIN FIXED(31) or BIN FLOAT(24). The dimensionality can be up to two. Spectra may be filled in a Dynamic List Entry or by macro $ACCU. Default Pool is $SPEC_POOL.

Spectra are created as digital, CAMAC or analog spectra.

- Digital spectra are used to accumulate integer or bit variables. The integer binsize specifies the number of input bins to be incremented in one spectrum bin. Bit spectra should be dimensioned (1,16) or (1,32), respectively, with binsize 1.
- CAMAC spectra are incremented in CAMAC memory (MR2000). The contents of the CAMAC memory must be copied into GOOSY spectra. These GOOSY spectra are created as digital spectra with binsize 1.
- Analog spectra are used to accumulate float variables. The binsize specifies an interval. The lower limit of the interval is inclusive, the upper limit exclusive. Therefore the upper

spectrum limit is exclusive.

5.3.3 Create Spectra

```
DBM> CRE SPEC s L (0,1023) 10 /DIGITAL      ! BIN FIXED(31), binsize=10
DBM> CRE SPEC s R (0,1023,0,255) (10,10)    ! BIN FLOAT(24), 2-dim.
DBM> CRE SPEC s(10) L (-10,15) 0.1 /ANALOG  ! name array, binsize 0.1
```

5.3.4 CAMAC Spectra

CAMAC spectra are normal digital GOOSY spectra. They keep additional information about the location of the CAMAC spectrum data in the CAMAC crate. This information must be specified during creation of the spectrum.

Related Commands

Some additional commands are provided to access CAMAC spectra, i.e. to copy the data from/to CAMAC and to clear the CAMAC data. The accumulation of the CAMAC spectra is controlled by special `START` and `STOP` commands. Note that these commands are executed in the Data Base Manager which must therefore be running.

```
DBM> CRE SPEC c L (0,1023) BRANCH=0 CRATE=1 STATION=20 OFFSET=1024/CAMAC
DBM> START MR2000 branch crate station /INIT
DBM> STOP  MR2000 branch crate station
DBM> CLEAR CAMAC SPECTRUM name /CAMAC/SPECTRUM
DBM> READ  CAMAC SPECTRUM name /ADD
DBM> WRITE CAMAC SPECTRUM name /ADD
DBM> SHOW  CAMAC SPECTRUM name       !contents of MR2000
```

CAMAC Spectrum Data Element

The additional information is stored in an additional Data Element GOOTYP(SA$SPCAM).

```
/* -------------------- begin of SE$SPCAM     ---------------- */
DCL 1 SE$SPCAM BASED,
    2 BE$SPCAM_mask    BIT(16) ALIGNED, /* bit mask           */
    2 IE$SPCAM_branch  BIN FIXED(15),   /* branch             */
    2 IE$SPCAM_crate   BIN FIXED(15),   /* crate              */
    2 IE$SPCAM_station BIN FIXED(15),   /* station            */
    2 LE$SPCAM_start   BIN FIXED(31),   /* offset in bytes    */
    2 LE$SPCAM_length  BIN FIXED(31);   /* length in longwords */
/* -------------------- end of   SE$SPCAM     ---------------- */
```

The spectra are referenced as normal spectra.

5.3.5 Spectrum Calibration

See also section 5.4.

Related Commands

A calibration is connected/disconnected to a spectrum by

```
DBM> CALIBRATE SPECTRUM spectrum calibration
DBM> DECALIBRATE SPECTRUM spectrum
```

Spectrum Calibration Data Element

One more Data Element is queued to the spectrum header:

```
/*--------------- start of SE$SPCAL  ----------------------*/
DCL P_SE$SPCAL POINTER;
DCL 1 SE$SPCAL BASED (P_SE$SPCAL),
      2 L_SE$SPCAL_DIM BIN FIXED(31),
      2 SE$SPCAL_DIM(1 REFER (L_SE$SPCAL_DIM)),
        3 L_SE$SPCAL_DIR_ID BIN FIXED(31),
        3 L_SE$SPCAL_POOL_ID BIN FIXED(31),
        3 L_SE$SPCAL_DE_ID BIN FIXED(31);
/*--------------- end of    SE$SPCAL ----------------------*/
```

5.3.6 Dump Spectra for IBM

GOOSY spectra can be dumped to a VAX disk file and then copied into an IBM VSAM library to be read later into SATAN programs.

```
DBM> DUMP SPECTRUM name OUTPUT=file
```

Several spectra specified by wildcards can be dumped and transferred at once. The spectrum files are transferred to the IBM by the DCL-command

```
$ SIBMSPEC file VSAMlib
```

The VSAM library must exist on the IBM!

5.4 Calibrations

5.4.1 Related Commands

```
CREATE  CALIBRATION type
SHOW    CALIBRATION
SET     CALIBRATION type (display program)
DISPLAY CALIBRATION (display program)
```

5.4.2 Calibration Data Elements

By default, calibrations are kept in the Directory $CALIB. Similar to spectra calibrations are sets of several Data Elements. They keep a calibration table which is used to calibrate the spectra data when displaying them. Each calibration can be connected to an arbitrary number of spectra.

```
/* -------------------- begin of SE$CAHE     ---------------- */
DCL P_SE$CAHE   POINTER;
DCL 1 SE$CAHE BASED(P_SE$CAHE),
      2 L_SE$CAHE_POINTS BIN FIXED(31),
      2 L_SE$CAHE_USED BIN FIXED(31),
      2 L_SE$CAHE_CAL_ID BIN FIXED(31),
      2 B_SE$CAHE_MASK BIT(32) ALIGNED;
/* -------------------- end of   SE$CAHE     ---------------- */
/* -------------------- begin of SE$CADA     ---------------- */
DCL P_SE$CADA POINTER;
DCL 1 SE$CADA BASED(P_SE$CADA),
      2 B_SE$CADA_MASK  BIT(32) ALIGNED,
      2 CV_SE$CADA_UNITS CHAR(78)VAR,
      2 L_SE$CADA_USED  BIN FIXED(31),
      2 L_SE$CADA_UNCAL  BIN FIXED(31),
      2 L_SE$CADA_CAL  BIN FIXED(31),
      2 R_SE$CADA_UNCAL (1 REFER(L_SE$CADA_UNCAL))
                         BIN FLOAT(24),
      2 R_SE$CADA_CAL (1 REFER(L_SE$CADA_CAL))
                         BIN FLOAT(24);
/* -------------------- end of   SE$CADA     ---------------- */
```

Each calibration takes two entries in Directory $CALIB. One for the main Data Element and one entry for the calibration table contents. First a calibration is created. Then it is connected to several spectra.

5.4.3 Fixed Calibrations

The calibration table contains calibrated values with a fixed step width for the uncalibrated values. The calibrated values are specified by `SET CALIBRATION FIXED` command. They can be specified by values, parameters for a polynomial, read from a file or calculated by a user written procedure.

Create Fixed Calibrations

```
DBM> CREATE CALIBRATION FIXED name entries
```

The parameter 'entries' specifies the number of calibrated values in the table.

Set Fixed Calibration Parameters

```
DBM> SET CALIBRATION FIXED name
```

A more detailed description of this command is found in the GOOSY Display Manual.

5.4.4 Float Calibrations

The calibration table contains uncalibrated and calibrated values. These values are specified by the `SET CALIBRATION FLOAT` command. They can be read from a file or calculated by a user written procedure.

Create Float Calibrations

```
DBM> CREATE CALIBRATION FLOAT name entries
```

The parameter 'entries' specifies the number of calibrated and uncalibrated pairs in the table.

Set Float Calibration Parameters

```
DBM> SET CALIBRATION FLOAT name
```

A more detailed description of this command is found in the GOOSY Display Manual.

5.4.5 Linear Calibrations

Linear calibrations do not use a table, but two parameters for a linear polynomial. A table is not needed because linear polynomials can be inverted. The calibration parameters are set by command `SET CALIBRATION LINEAR`.

Create Linear Calibrations

```
DBM> CREATE CALIBRATION LINEAR name
```

Set Linear Calibration Parameters

```
DBM> SET CALIBRATION LINEAR name
```

A more detailed description of this command is found in the GOOSY Display Manual.

5.5 Pictures

5.5.1 Related Commands

```
CREATE PICTURE
DELETE PICTURE
SHOW   PICTURE
CLEAR  PICTURE
MODIFY FRAME
```

5.5.2 Data Elements

By default, pictures are kept in the Directory $PICTURE. They keep information to display several frames containing spectra or scatterplots. Up to 64 frames may be displayed on one screen. Pictures take one entry in the $PICTURE Directory. Each frame takes one more entry. The default Pool is $PIC_POOL.

5.5.3 Create Pictures

When a picture is created, only the number of frames is specified. The frame content must then be specified by subsequent commands `MODIFY FRAME`. The same frame can be modified several times.

```
DBM> CRE PICTURE name frames
DBM> CRE PICTURE pict 6 /NOPROMPT           ! 6 frames
```

Without the `/NOPROMPT` qualifier the command will enter a menu driven prompting loop to specify all frames. Therefore this qualifier must be specified in command procedures!

5.5.4 Modify Picture Frames

Two commands are provided to modify frames, one for scatter frames and one for spectrum frames:

```
DBM> MOD FRAME SCATTER picture frame x y limits
DBM> MOD FRAME SPECTRUM picture frame spectrum limits
DBM> MOD FRAME SCATTER pict 1 [DATA]evt.geli(1) [DATA]evt.naj(1)
                                          ! frame one scatter
DBM> MOD FRAME SPECTRUM pict 3 [$SPECTRUM]s
                                          ! frame three spectrum
```

A more detailed description of this command is found in the GOOSY Display Manual.

5.6 User Data Elements

5.6.1 Related Commands

```
CREATE ELEMENT
DELETE ELEMENT
SHOW   ELEMENT
COPY   ELEMENT
CLEAR  ELEMENT
```

5.6.2 Create Data Elements

Besides the GOOSY Data Elements the user may define and create his own Data Elements. This may be done by GOOSY commands or by subroutine calls in a program. The following steps must be performed:

1. Put the PL/I source declaration of the Data Element in a text library. The name of the structure should be used as the name for the library module. The declaration must declare a based structure. A base pointer may be specified.
2. Create a Directory in Data Base
3. Create a Pool in Data Base
4. Create the Data Element Type, using the new PL/I structure in the text library.
5. Create the Data Element of the new Type.

If the declaration contains REFER members, the Data Element can be created only in a program, because the REFER values must be specified. To access the Data Element in an analysis procedure, include the library module declaring its structure and call $LOC macro to receive the pointer to the Data Element. In stand alone private programs the macro $ATTACH must be called to attach the Data Base first.

5.6.3 Example

Assume we want to create a Data Element like this:

```
DCL P_SX$evt POINTER;
DCL 1 SX$evt BASED(P_SX$evt),
    2 patt     BIT(32) ALIGNED,
    2 geli(10) BIN FIXED(31),
    2 naj(10   BIN FIXED(15);
```

The structure must be BASED in order to use it in a PL/1 program as well. This declaration is in our library TPRIV in module SX$EVT. We create in the following example a Data Element EVT of the above Type.

```
DBM> CRE TYPE @tpriv(SX$evt)                ! Declaration in library
DBM> CRE ELEMENT [DATA]evt evtdata SX$evt    ! Directory DATA and
                                            !  Pool EVTDATA must exist
```

5.7 Dynamic Lists

5.7.1 Related Commands

```
CREATE DYNAMIC LIST
DELETE DYNAMIC LIST
SHOW   DYNAMIC LIST
CREATE DYNAMIC ENTRY type
DELETE DYNAMIC ENTRY type
```

5.7.2 Create Dynamic Lists

Each Dynamic List takes two entries in Directory $DYNAMIC. The default Pool is $DYNAMIC.

```
DBM> CRE DYNAMIC LIST list 100     ! Dynamic List for 100 entries
```

The following switches apply for the `CREATE DYNAMIC ENTRY` commands:

`/UPDATE` The modification becomes active immediately (also for the `DELETE DYNAMIC ENTRY` command) for a running analysis.

`/MASTER` Valid for conditions (except multiwindow) and procedures. Master Functions are executed first of all other Entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the Dynamic List execution is terminated. If the same master condition is in two Dynamic Lists, both lists are skipped, if the condition was false.

In all commands explicit defaults for Data Base, node and Directories can be specified. These parameters are not included in the following descriptions:

```
DYN_DIR=default Directory of Dynamic List
COND_DIR=default Directory of condition
SPEC_DIR=default Directory of spectrum
PAR_DIR=default Directory of parameters
POLY_DIR=default Directory of polygon
BASE=default Data Base
NODE=default node
```

5.7.3 Arrays

Spectra or conditions may be arrays. In this case an index range must be specified. All additional Data Elements must be either scalar or indexed by the same range. Ranges are specified by (lower limit : upper limit).
Examples:

```
DBM> CRE DYN ENTRY WINDOW dlist [d]e_recoil(1:5) -
          PARA=[d]$event.ener
DBM> CRE DYN ENTRY SPECTRUM dlist [d]ener1(2:4) -
           PARA=[d]$event.e(2:4) CONDI=[d]de_window
DBM> CRE DYN ENTRY SPECTRUM dlist [d]ede(1:4) -
           PARA=([d]$event.e,$event.de)
DBM> CRE DYN ENTRY INDEXED dlist [d]ede(1:7) -
           PARA=([d]$event.e,$event.de) -
           INDEX=[d]a.b(1)
```

[d] is the Directory specification
The difference between windows and multiwindows is that multiwindows have only one object for all subwindows, but one result bit for each, whereas windows need one object per subwindow, but have only one result bit (set, if all subwindows are true). Multiwindows may be used as filters for spectrum array accumulation. The internal dimension of the window must match the specified index range. It may also be used for 'indexed' spectrum accumulation. Then the index of the last matching subwindow is used to select the spectrum member. In the first case, the subwindows may overlap, in the second case this makes normally no sense.

```
DBM> CRE DYN ENTRY SPECTRUM list [d]ener1(2:4) -
           PARA=[d]$event.e(2:4) CONDI=[d]m_window
! three spectrum Entries are executed
DBM> CRE DYN ENTRY INDEXEDSPECTRUM list [d]ener(2:4) -
           PARA=[d]$event.e(1) INDEX=[d]m_window
! One spectrum Entry is executed
```

[d] is the Directory specification
In both cases 'm_window' must have 3 subwindows.

5.7.4 Creating Dynamic List Entries

PROCEDURE

Command to insert an entry with a user specified procedure call:

```
CRE DYN ENTRY PROCEDURE listname MODULE=image(module)
                                 PARAMETER=(argument list)
                                 CONDITION=cond
                                 /MASTER
MODULE        module specification as 'image(module)'. Module
              must be linked in sharable image
image         logical name of shar.image
PARAMETER     arg.list of DE-members
CONDITION     name of condition (optional)
/MASTER       master Entry
```

This Entry will call a module from a sharable image. The pointers to the Data Elements specified in the argument list are passed to the procedure.
Example:

```
CRE DYN ENTRY PROCEDURE dlist
              MOD=privshar(x$loop)
              PARA=([d]$event.z4.de(5),$event.z5)
              /MASTER
```

[d] is the Directory specification
The X$LOOP declaration must be:
`ENTRY(POINTER,POINTER) RETURNS(BIN FIXED(31))`
The sharable image must be linked by the DCL command `LSHARIM`.

FUNCTION

Command to insert an entry with a user specified condition function call:

```
CRE DYN ENTRY FUNCTION listname condition MODULE=image(module)
                                  PARAMETER=(argument list)
                                  /MASTER
MODULE        module specification as 'image(module)'. Module
              must be linked in sharable image. Is used and required only
              if not specified in condition.
image         logical name of shar.image
PARAMETER     arg.list of DE-members
/MASTER       master entry
```

This entry will call a module from a sharable image. The pointers to the Data Elements specified in argument list are passed to the procedure. The first argument, a BIT(1) ALIGNED, returns the condition result.
Example:

```
CRE DYN ENTRY FUNCTION dlist [d]cond
              MOD=privshar(x$cond)
              PARA=([d]$event.z4.de(5),$event.z5)
```

[d] is the Directory specification
The X$COND declaration must be:
`ENTRY(BIT(1) ALIGNED,POINTER,POINTER) RETURNS(BIN FIXED(31))`
The sharable image must be linked by the DCL command `LSHARIM`.

PATTERN

Command to insert a pattern condition entry:

```
CRE DYN ENTRY PATTERN listname cond PARAMETER=object
                                    /MASTER
PARAMETER     DE-members
/MASTER       Master entry
```

The entry will check a specified Data Element member versus a pattern. Note that four test modes can be specified with the pattern condition (IDENT, ANY, EXCL, INCL). The values of the Data Element members can be inverted bitwise. Up to 8 internal dimensions. Objects can be of type BIT(16), BIT(32), BIN FIXED(15), or BIN FIXED(31).
Example:

```
CRE DYN ENTRY PATTERN dlist [d]main_pat
         PARA=[d]$event.pat
         /MASTER
```

[d] is the Directory specification

WINDOW

Command to insert a window condition entry:

```
CRE DYN ENTRY WINDOW listname cond PARAMETER=object
                                   /MASTER
PARAMETER     DE-members
/MASTER       Master entry
```

This entry will check a specified Data Element member versus window limits. Up to 8 internal dimensions. The objects may be BIN FLOAT(24), BIN FIXED(15), or BIN FIXED(31).
Example:

```
CRE DYN ENTRY WINDOW dlist [d]e_recoil
         PARA=[d]$event.ener
```

[d] is the Directory specification

MULTIWINDOW

Command to insert a multiwindow condition entry:

```
CRE DYN ENTRY MULTIWINDOW listname cond PARAMETER=object

PARAMETER     DE-member
```

This entry will check a specified Data Element member versus all window limits. For each check a result bit is set, which may be used to increment a spectrum array member. In addition, the number of the last matching window may be used as the index of a spectrum array member (see INDEXEDSPECTRUM). The object may be BIN FLOAT(24), BIN FIXED(15), or BIN FIXED(31).
Example:

```
CRE DYN ENTRY MULTI dlist [d]e_recoil
         PARA=[d]$event.ener
```

[d] is the Directory specification

POLYGON

Command to insert a polygon condition entry:

```
CRE DYN ENTRY POLYGON listname cond PARAMETER=(x,y)
                                    POLYGON=name
                                    /MASTER
PARAMETER     DE-members for X and Y. Used and required only
              if not specified in condition.
POLYGON       Name of polygon. Used and required only
              if not specified in condition.
/MASTER       Master entry
```

It is checked whether the point X,Y is inside (true) or outside (false) the polygon. Objects may be BIN FLOAT(24), BIN FIXED(15), or BIN FIXED(31).
Example:

```
CRE DYN ENTRY POLY dlist [d]poly_1
         PARA=([d]$event.de,[d]$event.ener)
         POLYG=poly
```

[d] is the Directory specification.

COMPOSED

Command to insert a composed condition entry:

```
CRE DYN ENTRY COMPOSED listname cond /MASTER

/MASTER       Master entry
```

A boolean expression of conditions is executed. The expression is specified in the corresponding condition Data Element.
Example:

```
CRE DYN ENTRY COMPOSED dlist [d]all_ok /MASTER
```

[d] is the Directory specification

SPECTRUM

Command to insert a spectrum entry:

```
CRE DYN ENTRY SPECTRUM listname spectrum PARAMETER=object
                                         CONDITION=cond
                                         INCREMENT=incr
PARAMETER     DE-members for coordinates
CONDITION     condition for spectrum (optional)
INCREMENT     DE-member for increment (optional) (BIN FLOAT(24))
```

Supports spectra of Type BIN FIXED(15), BIN FIXED(31) or BIN FLOAT(24) with up to 2 dimensions. Coordinates can be BIN FIXED(15), BIN FIXED(31) or BIN FLOAT(24).

Examples:

```
CRE DYN EN SPECTRUM dlist [d]ener1
           PARA=[d]$event.e(1) CONDI=[d]de_window
CRE DYN EN SPECTRUM dlist [d]ede
           PARA=([d]$event.e,$event.de)
```

[d] is the Directory specification

INDEXEDSPECTRUM

Command to insert an indexed spectrum entry:

```
CRE DYN ENTRY INDEXEDSPECTRUM listname spectrum(l:u) PARAMETER=object
                                                     INDEX=index
                                                     INCREMENT=incr
                                                     CONDITION=cond
PARAMETER     DE-members for coordinates
INDEX         DE-member (BIN FIXED(31)) or multiwindow to specify
              spectrum member
CONDITION     condition for spectrum (optional)
INCREMENT     DE-member for increment (optional) (BIN FLOAT(24))
```

Supports spectra of Type BIN FIXED(15), BIN FIXED(31) or BIN FLOAT(24) with up to 2 dimensions. Coordinates can be BIN FIXED(15), BIN FIXED(31) or BIN FLOAT(24). Specified spectrum must be an array. Specification of index is used to select the spectrum member to be incremented. This could be either a parameter Data Element or a multiwindow.
Examples:

```
CRE DYN EN INDEXED dlist [d]ener(1:10)
           PARA=[d]$event.e(1) INDEX=[d]m_window
CRE DYN EN INDEXED dlist [d]ede(1:5)
           PARA=([d]$event.e,$event.de) INDEX=[d]a.b
```

[d] is the Directory specification

BITSPECTRUM

Command to insert a bitspectrum entry:

```
CRE DYN ENTRY BITSPECTRUM listname spectrum PARAMETER=object
                                              CONDITION=cond
PARAMETER     DE-members for coordinates
CONDITION     condition for spectrum (optional)
```

Supports one dimensional spectra, Type BIN FIXED(31). Coordinates must be BIT(32), BIT(16), BIN FIXED(15), or BIN FIXED(31).
Example:

```
CRE DYN ENTRY BIT dlist [d]patt
          PARA=[d]$event.pat(1)
```

[d] is the Directory specification

SCATTER

This entry is inserted by the `DISPLAY PICTURE` command, if scatter frames are in the picture, or by the `DISPLAY SCATTER` command. The name of the list can be specified optionally with these commands. The default is $SCATTER. Note that this list must be attached to be active. It should be attached as the last list. Scatter Entries are deleted only by the creating display process. This may lead to 'dead' scatter Entries, i.e. if the environment name is no longer used. Attaching the list in this case a message is displayed that a link could not be opened. Then one should delete all scatter Entries of all types by the command:

```
DELETE DYNAMIC ENTRY SCATTER list * * /UPDATE
```

No scatter plot should be active during that command.

5.7.5 Delete Entries

The commands to delete Dynamic Entries are similar to the create commands.

```
DBM> DELETE DYNAMIC ENTRY type list entry
```

where 'type' is the same keyword as the fourth key in the create commands, 'list' is the Dynamic List name and 'entry' the name of the Data Element, e.g. the spectrum or condition. Different to the create commands the 'type' and 'entry' parameter may be wildcarded by an asterisk to delete all Entries or all Entries of a specific Type.

5.7.6 Examples

For Dynamic List Entries the objects for spectrum accumulation and condition checks, the spectrum increment and the index must be Members of GOOSY Data Elements created already in the Data Base. Assume we have created a Data Element like this:

```
DCL 1 SX$evt,
    2 patt     BIT(32) ALIGNED,
    2 geli(10) BIN FIXED(31),
    2 naj(10   BIN FIXED(15);
```

This declaration is in our library TPRIV in module SX$EVT. We refer in the following examples to Data Element EVT of the above Type. We assume that conditions c,w,a(1:10) and spectra s and s2 already exist.

```
DBM> CRE TYPE @tpriv(SX$evt)                ! Declaration in library
DBM> CRE ELEMENT [DATA]evt evtdata SX$evt    ! Directory DATA and
                                            !  Pool EVTDATA must exist

DBM> CRE DYNAMIC LIST list ENTRIES=100     ! Dynamic List for 100 Entries
DBM> CRE DYN ENTRY PATTERN list c PARAMETER=evt.patt
DBM> CRE DYN ENTRY WINDOW  list w PARAMETER=evt.geli(3)
DBM> CRE DYN ENTRY WINDOW  list a(1:10) PARA=evt.geli(1:10)
                                       ! condition name array!
DBM> CRE DYN ENTRY SPECTRUM list s PARAMETER=[DATA]evt.naj(1)
DBM> CRE DYN ENTRY SPECTRUM list s2 -
             PARAMETER=([DATA]evt.naj(1),[DATA]evt.geli(1)) -
             CONDITION=c
                                       ! 2-dim. spectrum
```

Appendix A

Command Description

CALCULATE SPECTRUM

```
CALCULATE SPECTRUM
      result operand_1 operand operand_2 factor
      spec_dir base node
      /CONSTANT
      /[NO]KEEP
```

PURPOSE Perform spectrum arithmetic operations.

PARAMETERS

result Name of spectrum which contains the result. If the spectrum does not exist, it will be created.

operand_1 First spectrum operand.

operand Aritmetic operation which should be performed.

operand_2 Second operand. It could be a spectrum name or a a constant value; in that case /CONSTANT must be specified.

factor Factor to scale the channel contents of operand 2. It is ignored if /CONSTANT is specified.

spec_dir Default spectrum directory.

base	Default data base name.
node	Default node.
/CONSTANT	If specified the OPERAND_2 is interpreted as a costant value.
/[NO] KEEP	Keep context of all data bases.
Caller	MDBM,MGOODBM
Author	W. Spreng

CALIBRATE SPECTRUM

CALIBRATE SPECTRUM spectrum calibration spec_dir cal_dir base node

PURPOSE	Connect calibration to a spectrum.
PARAMETERS	
spectrum	Name of spectrum
calibration	Name of calibration
spec_dir	Default spectrum directory.
cal_dir	Default directory for calibrations.
base	Deafult Data Base name.
node	Node name for Data Base file
Caller	MDBM,MGOODBM
Author	W.Spreng

CLEAR CAMAC SPECTRUM

CLEAR CAMAC SPECTRUM name spec_dir base node
/CAMAC
/SPECTRUM
/LOG
/[NO]KEEP_MAP

PURPOSE clear one (or all) spectrum

PARAMETERS

NAME required string global replace default: ”
Name of Spectrum (wildcard) to be cleared
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.

SPEC_DIR String global replace default: '$SPECTRUM'
Default spectrum directory.

BASE String global replace default: 'DB'
Name of Data Base

NODE String global replace default: 'E'
Name of node

/LOG Switch default: ”
Output list of cleared spectra

/CAMAC Switch default: ”
Clear spectrum in CAMAC.

/SPECTRUM Switch default: ”
Clear spectrum in data base.

[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller mdbm

Author H.G.Essel

CLEAR CONDITION COUNTER

CLEAR CONDITION COUNTER name cond_dir base node
/[NO]KEEP_MAP
/LOG

PURPOSE clear condition counters specified by name

PARAMETERS

name required string global replace default: ”
Name expression of condition. Wildcards are accepted in the form:
* x* *x *x* x*y
Name arrays are supported. Index may be wildcarded. This is assumed, if name is wildcarded.

cond_dir string global replace default: '$CONDITION'
Name of condition Directory

base String global replace default: 'DB'
Name of Data Base

node String global replace default: 'E'
Name of node

/LOG Switch default: none
Output list of cleared conditions

/[NO] KEEP_MAP Set replace default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base

Caller MDBM

Author K.Winkelmann

CLEAR ELEMENT

CLEAR ELEMENT name dir base node
/LOG
/[NO]KEEP_MAP

PURPOSE Clear Element.Set values of a Data Element to zero.

PARAMETERS

name required string global default: ”
Name of Element to be cleared in the of
Node::base:[dir]name(i)->type(i)

dir string global replace default: 'DATA'
Default directory

base string global replace default: 'DB'
Default Data Base name

node string global replace default: 'E'
Default node name

/LOG switch default: ”
Displays cleared Data Element.

[NO] KEEP_MAP switch default: '/KEEP_MAP'
Inhibit the unmap (detach) of the whole Data Base.

Caller M$DMCMD

Author Th. Kroll

CLEAR PICTURE

CLEAR PICTURE picture frame pic_dir base node
/[NO]KEEP_MAP
/[NO]LOG

PURPOSE Clear spectra defined in a picture data element

PARAMETERS

picture	Picture data element name
frame	Frame specification.
pic_dir	Default directory name
base	Default data base name.
node	Default node name
/[NO] KEEP_MAP	Inhibit the unmap of the whole Data Base.
/[NO] LOG	List names of all pictures found and of all spectra which have been cleared.

CLEAR SPECTRUM

CLEAR SPECTRUM name spec_dir base node
/LOG
/[NO]KEEP_MAP

PURPOSE clear one (or all) spectrum

PARAMETERS

NAME required string global replace default: ”
Name of Spectrum (wildcard) to be cleared
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.

SPEC_DIR String global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE String global replace default: 'DB'
Name of Data Base

NODE	String global replace default: 'E' Name of node
/LOG	Switch default: " Output list of cleared spectra
[NO] KEEP_MAP	Switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
Caller	mdbm
Author	K.Winkelmann

COMPRESS BASE

COMPRESS BASE base file
/DISMOUNT

PURPOSE	Compress and copy data base (the copy cannot be mounted as GOOSY data base!). Command is executed in MUTIL.

PARAMETERS

base	required string default: " Name of the data base to be compressed and copied.
file	required string default: " Name of output file. File must not exist!
/DISMOUNT	switch default: Dismount data base after copy.
Caller	MDBCOPY
Author	H.G.Essel

CONVERT BASE

CONVERT BASE base file size

PURPOSE Convert data base.

PARAMETERS

base required string default: ”
Name of the data base to be expanded and copied.

file required string default: ”
Name of output file. File must not exist!

Caller MDBCOPY,MUTIL

Author H.G.Essel, B.Dechant

COPY BASE

COPY BASE base file size area
/DISMOUNT

PURPOSE Expand and copy data base.

PARAMETERS

base required string default: ”
Name of the data base to be expanded and copied.

file required string default: ”
Name of output file. File must not exist!

size integer
Optional size of new base in Kbytes.

area integer
Optional new number of entries in area directory.

/DISMOUNT switch default:
Dismount data base after copy.

Caller MDBCOPY,MUTIL

Author H.G.Essel

COPY CONDITION

COPY CONDITION name destname dir dest_dir
base destbase node destnode
/REPLACE
/CONFIRM
/LOG
/[NO]KEEP_MAP

PURPOSE Copy source Condition to destination Condition

PARAMETERS

NAME String replace default: ”
Source Condition name, may be wildcarded.

DESTNAME String replace default: ”
Destination Condition name, may be wildcarded.

COND_DIR String global replace default: '$CONDITION'
Source Condition directory name, may be wildcarded.

ESTCOND_DIR String replace default: '$CONDITION'
Destination Condition directory name, may be wildcarded.

BASE String global replace default: 'DB'
Source Condition data base.

DESTBASE String replace default: 'DB'
Destination Condition data base.

NODE String global replace default: 'E'
Source Condition node name.

DESTNODE String global replace default: 'E'
Destination Condition node.

/REPLACE	Switch default: none If a existing Condition will be replaced the switch has to be set.
/CONFIRM	Switch default: /CONFIRM If a new Condition should be created and /NOCONFIRM is set no prompt for creation will follow. If a new Condition should be created and /NOCONFIRM is NOT set a prompt for creation will follow.
/LOG	Switch default none If /LOG is set the copy commands displays the file specifications of each file copied.
[NO] KEEP_MAP	Switch default: /KEEP_MAP
Caller	MDBM,MGOODBM
Author	Th.KROLL

COPY ELEMENT

COPY ELEMENT name destname dir destdir destpool
base destbase node destnode
/REPLACE
/ALL
/NOCONFIRM
/[NO]KEEP_MAP

PURPOSE	Copy source Dataelement to destination Dataelement
PARAMETERS	
name	required string global replace default: " Source Dataelement name
destname	required string global replace default: " Destination Dataelement name
dir	required string global replace default: 'DATA' Default source directory
destdir	required string global replace default: 'DATA' Default destination directory

destpool	required string global replace default: 'DATA' Default destination pool
base	required string global replace default: 'DB' Default source database
destdb	required string global replace default: 'DSTDB' Default destination database
node	required string global replace default: 'E' Default source node
destnode	required string global replace default: 'E' Default destination node
I_REPLACE	switch default: " Replace deastination dataelement
I_ALL	switch default: " Replace or copy a complete name array
I_NOCONFIRM	switch default: " Don't ask to confirm creation of a new Dataelement
I_KEEP_MAP	switch default: /KEEP_MAP Inhibit unmap of Data Base
Caller	MDBM,MGOODBM
Author	Th. Kroll

COPY MEMBER

COPY MEMBER member destmember dir destdir
base destbase node destnode
/[NO]KEEP_MAP

PURPOSE Copy Data Element member to another Data Element member

PARAMETERS

member	required string global replace default: " Default source member name Node::base:[dir]name(i)->type(i).member
destmember	required string global replace default: " Default destination member name Node::base:[dir]name(i)->type(i).member
dir	required string global replace default: 'DATA' Default source Directory
destdir	required string global replace default: 'DATA' Default destination Directory

base	required string global replace default: 'DB' Default source Data Base
destbase	required string global replace default: 'DSTDB' Default destination Data Base
node	required string global replace default: 'E' Default source node
destnode	required string global replace default: 'E' Default destination node
[NO] KEEP_MAP	switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
Caller	MDBM,MGOODBM
Author	Th. KROLL

COPY Polygon

COPY Polygon name destname poly_dir destpoly_dir
base dest_base node destnode
/REPLACE
/CONFIRM
/LOG
/[NO]KEEP_MAP

PURPOSE	Copy source Polygon to destination Polygon
PARAMETERS	
NAME	String replace default: '' Source Polygon name, may be wildcarded.
DESTNAME	String replace default: '' Destination Polygon name, may be wildcarded.
POLY_DIR	String global replace default: '$Polygon' Source Polygon directory name, may be wildcarded.
ESTPOLY_DIR	String replace default: '$Polygon' Destination Polygon directory name, may be wildcarded.

BASE	String global replace default: 'DB' Source Polygon data base.
DESTBASE	String replace default: 'DB' Destination Polygon data base.
NODE	String global replace default: 'E' Source Polygon node name.
DESTNODE	String global replace default: 'E' Destination Polygon node.
/REPLACE	Switch default: none If a existing Polygon will be replaced the switch has to be set.
/CONFIRM	Switch default: /CONFIRM If a new Polygon should be created and /NOCONFIRM is set no prompt for creation will follow. If a new Polygon should be created and /NOCONFIRM is NOT set a prompt for creation will follow.
/LOG	Switch default none If /LOG is set the copy commands displays the file specifications of each file copied.
[NO] KEEP_MAP	Switch default: /KEEP_MAP
Caller	MDBM,MGOODBM
Author	Th.KROLL

COPY SPECTRUM

COPY SPECTRUM name dest_name spec_dir destspec_dir
base destbase node destnode
/REPLACE
/CONFIRM
/LOG
/[NO]KEEP_MAP

PURPOSE Copy source Spectrum to destination Spectrum

PARAMETERS

NAME String replace default: ”
Source Spectrum name, may be wildcarded.

DESTNAME String replace default: ”
Destination Spectrum name, may be wildcarded.

SPEC_DIR String global replace default: '$SPECTRUM'
Source Spectrum directory name, may be wildcarded.

DESTSPEC_DIR String replace default: '$SPECTRUM'
Destination Spectrum directory name, may be wildcarded.

BASE String global replace default: 'DB'
Source spectrum data base.

DESTBASE String replace default: 'DB'
Destination spectrum data base.

NODE String global replace default: 'E'
Source spectrum node name.

DESTNODE String global replace default: 'E'
Destination spectrum node.

/REPLACE Switch default: none
If a existing Spectrum will be replaced the switch has to be set.

/CONFIRM Switch default: /CONFIRM
If a new spectra should be created and /NOCONFIRM is set no prompt for creation will follow. If a new spectra should be created and /NOCONFIRM is NOT set a prompt for creation will follow.

/LOG Switch default none
If /LOG is set the copy commands displays the file specifications of each file copied.

[NO] KEEP_MAP Switch default: /KEEP_MAP

Caller MDBM,MGOODBM

Author Th.KROLL

CREATE AREA

CREATE AREA area base pool areabytes cluster
/[NO]KEEP_MAP

PURPOSE Create an Area in a Data Base

PARAMETERS

area Area name
required common default

base Data Base name
required common default

pool Pool name
required common default

areabytes Size in bytes
replace default:'100'

cluster Cluster size in bytes
replace default:'4'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CREA AREA ALPHA DB BETA 10240 16
create the Area ALPHA in Pool Beta of Data Base DB
with 10KBytes and a cluster size of 16 bytes.

Caller M$DMCMD

Author M. Richter

File name M$ACRAR.PPL

Dataset -

CREATE BASE

CREATE BASE base basefile adentries mdentries pdentries tdentries basepages
/PERMANENT/TEMPORARY
/GLOBAL_SEC/SYSTEM_GLOBALSEC

PURPOSE Create a new Data Base (section)

PARAMETERS

base Data Base name required common replaced default

basefile Data base file name required common replaced default

adentries Number of entries for Area Directory
replace default:'100'

mdentries Number of entries for Master Directory
replace default:'100'

pdentries Number of entries for Pool Directory
replace default:'100'

tdentries Number of entries for Type Directory
replace default:'100'

basepages Size in pages
replace default:'500'

bytesbit Bytes per Bit in Home Block Bit Map (IN PAGELETS)
replace default:'0'

/PERMANENT/TEMPORARY Section permanence
default:'/PERMANENT'

/GLOBAL_SEC/SYSTEM_GLOBALSEC Section scope
default:'/GLOBAL_SEC'

EXAMPLE CREAT BASE DB DB 500 800 200 200 32000 32 create the Data Base DB as a Global Section with the section file DB.SEC under the default VMS directory. The new Data Base of 16 MByte size (32000 pages with 512 bytes) will allow up to 500 Areas, 800 Data Element Directories, 200 Pools, and 200 Data Types. It will be a permanent Group Global Section. Bytes per bit will be:
32 pagelets * 512 bytes/pagelet = 16364 bytes

Caller M$DMCMD

Author M. Richter

File name M$ACRDB.PPL

Dataset -

CREATE CALIBRATION FIXED

CREATE CALIBRATION FIXED name entries cal_dir
cal_pool base node
/[NO]KEEP_MAP

PURPOSE Create a Data Element for fixed calibration.

PARAMETERS

name Name of calibration Data Element.

entries Number of uncalibrated/calibrated points.

cal_dir Directory for calibration

cal_pool Pool for calibration.

base Data Base name

node Node name for Data Base file

/[NO] KEEP_MAP Inhibit the unmap of the whole Data Base.

Caller MDBM,MGOODBM

Author W.Spreng

CREATE CALIBRATION FLOAT

CREATE CALIBRATION FLOAT name entries cal_dir
cal_pool base node
/KEEP_MAP

PURPOSE Create Data Element for float calibration.

PARAMETERS

name Name of calibration Data Element.

entries Number of uncalibrated/calibrated points.

cal_dir Directory for calibration

cal_pool Pool for calibration.

base Data Base name

node node name for Data Base file

/[NO] KEEP_MAP Inhibit the unmap of the whole Data Base.

Caller MDBM,MGOODBM

Author W.Spreng

CREATE CALIBRATION LINEAR

CREATE CALIBRATION LINEAR name cal_dir cal_pool
base node
/[NO]KEEP_MAP

PURPOSE Create Data Element for linear calibration.

PARAMETERS

name Name of calibration Data Element.

cal_dir Directory for calibration

cal_pool Pool for calibration.

base Data Base name

node node name for Data Base file

/[NO] KEEP_MAP Inhibit the unmap of the whole Data Base.

Caller MDBM,MGOODBM

Author W.Spreng

CREATE CONDITION COMPOSED

CREATE CONDITION COMPOSED name expression
cond_dir cond_pool base node
/[NO]DYNAMIC
/[NO]DOCUMENT
/[NO]KEEP_MAP

PURPOSE create a composed condition

PARAMETERS

name required string replace default: ”
name of the condition

expression string replace default: ”
Boolean expression of conditions.

cond_dir string global replace default: '$CONDITION'
default directory

cond_pool string global replace default: '$COND_POOL'
default pool

base string global replace default: 'DB'
default data base

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

/[NO] DOCUMENT switch default: /NODOCUMENT
if on documentation will be held

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE CONDITION FUNCTION

CREATE CONDITION FUNCTION name function
cond_dir cond_pool base node
/[NO]DYNAMIC
/[NO]DOCUMENT
/[NO]KEEP_MAP

PURPOSE create a function condition

PARAMETERS

name required string replace default: ''
name of the condition

function string replace default: ''
specification of function by image(module).

cond_dir string global replace default: '$CONDITION'
default directory

cond_pool string global replace default: '$COND_POOL'
default pool

base string global replace default: 'DB'
default data base

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

/[NO] DOCUMENT switch default: /NODOCUMENT
if on documentation will be held

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE CONDITION MULTIWINDOW

CREATE CONDITION MULTIWINDOW name limits dimension
cond_dir cond_pool base node
/[NO]DYNAMIC
/[NO]DOCUMENT
/[NO]KEEP_MAP

PURPOSE create a multiwindow condition

PARAMETERS

name required string replace default: ''
name of the condition

limits string replace default: '(0,4096)'
specification of limits.

dimension integer default: 1
Internal dimension

cond_dir string global replace default: '$CONDITION'
default directory

cond_pool string global replace default: '$COND_POOL'
default pool

base string global replace default: 'DB'
default data base

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

/[NO] DOCUMENT switch default: /NODOCUMENT
if on documentation will be held

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE CONDITION PATTERN

CREATE CONDITION PATTERN name pattern dimension
cond_dir cond_pool base node invert
/IDENT/INCL/ANY/EXCL (=checkmode)
/[NO]DYNAMIC
/[NO]DOCUMENT
/[NO]KEEP_MAP

PURPOSE create a pattern condition

PARAMETERS

name required string replace default: ”
name of the condition

pattern string replace default: '0'
specification of test pattern.

dimension integer default: 1
Internal dimension

cond_dir string global replace default: '$CONDITION'
default directory

cond_pool string global replace default: '$COND_POOL'
default pool

base string global replace default: 'DB'
default data base

node string global replace default: 'E'
default node

invert string default: '0'
inversion pattern

/[NO] DYNAMIC switch default: /DYNAMIC'

/[NO] DOCUMENT switch default: /NODOCUMENT
if on documentation will be held

checkmode set default: /IDENT valid values are:

/IDENT	true if patt=object
/INCL	true if patt&object=patt
/EXCL	true if patt&object=object
/ANY	true if patt&object true (any 1 occurs)

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE CONDITION POLYGON

CREATE CONDITION POLYGON name polygon dimension
cond_dir poly_dir cond_pool base node
/[NO]DYNAMIC
/[NO]DOCUMENT
/[NO]KEEP_MAP

PURPOSE create a polygon condition

PARAMETERS

name required string replace default: ”
name of the condition

polygon string replace default: ”
Name of polygon.

dimension integer default: 1
Internal dimension (valid value = 1)

cond_dir string global replace default: '$CONDITION'
default condition directory

poly_dir string global replace default: '$POLYGON'
default polygon directory

cond_pool string global replace default: '$COND_POOL'
default pool

base string global replace default: 'DB'
default data base

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

/[NO] DOCUMENT switch default: /NODOCUMENT
if on documentation will be held

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE CONDITION WINDOW

CREATE CONDITION WINDOW name limits dimension
cond_dir cond_pool base node
/[NO]DYNAMIC
/[NO]DOCUMENT
/[NO]KEEP_MAP

PURPOSE create a window condition

PARAMETERS

name required string replace default: ”
name of the condition

limits string replace default: '(0,4096)'
specification of limits.

dimension integer default: 1
Internal dimension

cond_dir string global replace default: '$CONDITION'
default directory

cond_pool string global replace default: '$COND_POOL'
default pool

base string global replace default: 'DB'
default data base

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

/[NO] DOCUMENT switch default: /NODOCUMENT
if on documentation will be held

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DIRECTORY

CREATE DIRECTORY dir dedentries base
/[NO]KEEP_MAP

PURPOSE Create a Data Element Directory in a Data Base

PARAMETERS

dir Data Element Directory name
required, common default

dedentries Number of entries for Data Element Directory
replace default:'100'

base Data Base name
required, common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CREA DIR PARAM 300 DB
create the Data Element Directory 'PARAM' in the
Data Base 'DB' with space for 300 Data Elements.

Caller M$DMCMD

Author M. Richter

File name M$ACRDI.PPL

Dataset -

CREATE DYNAMIC ENTRY BITSPECTRUM

CREATE DYNAMIC ENTRY BITSPECTRUM dyn_list
spectrum parameter increment condition
dyn_dir par_dir cond_dir spec_dir base node
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create a spectrum dynamic list entry

PARAMETERS

dyn_list required string global replace default: ”
Dynamic list name specification.

spectrum required string replace default: ”
Name specification of spectrum array.

parameter required string replace default: ”
Data element member of type
BIT(16) or BIT(32) ALIGNED.

increment string default: ”
Data element member to be used as increment.

condition string default: ”
Name of a condition. If specified the spectrum is filled only, if the condition was true.

dyn_dir string global replace default: '$DYNAMIC'
Default directory

par_dir string global replace default: 'DATA'
Default directory

cond_dir string global replace default: '$CONDITION'
Default directory

spec_dir string global replace default: '$SPECTRUM'
Default directory

base string global replace default: 'DB'
Default data base name

node string global replace default: 'E'
Default node name

/UPDATE switch default: ''
Update dynamic list (then it becames valid for a running analysis immediately.

/[NO] CHECK switch default: /CHECK
Do dynamic list checking by attaching it

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base.

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DYNAMIC ENTRY COMPOSED

CREATE DYNAMIC ENTRY COMPOSED dyn_list condition dyn_dir cond_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create a composed condition dynamic list entry

PARAMETERS

dyn_list required string global replace default: ''
Dynamic list name specification.

condition required string replace default: ''
Name specification of composed condition.

dyn_dir string global replace default: '$DYNAMIC'
Default directory

cond_dir string global replace default: '$CONDITION'
Default directory

base string global replace default: 'DB'
Default data base name

node string global replace default: 'E'
Default node name

/MASTER switch default: "
Master procedure (executed first)

/UPDATE switch default: "
Update dynamic list (then it becames valid for a running analysis immediately.

/[NO] CHECK switch default: /CHECK
Do dynamic list checking by attaching it

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base.

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DYNAMIC ENTRY FUNCTION

CREATE DYNAMIC ENTRY FUNCTION dyn_list condition module parameter dyn_dir par_dir cond_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create a function condition dynamic list entry

PARAMETERS

dyn_list required string global replace default: "
Dynamic list name specification.

condition required string replace default: "
Name specification of condition.

module string replace default: ”
Specification of module as image(module).

parameter string replace default: ”
List of data element name specifications. The pointers to these data elements are passed to the procedure.

dyn_dir string global replace default: '$DYNAMIC'
Default directory of dynamic list

par_dir string global replace default: 'DATA'
Default parameter directory

cond_dir string global replace default: '$CONDITION'
Default directory of condition

base string global replace default: 'DB'
Default data base name

node string global replace default: 'E'
Default node name

/MASTER switch default: ”
Master procedure (executed first)

/UPDATE switch default: ”
Update dynamic list (then it becames valid for a running analysis immediately.

/[NO] CHECK switch default: /CHECK
Do dynamic list checking by attaching it

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base.

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DYNAMIC ENTRY INDEXEDSPECTRUM

CREATE DYNAMIC ENTRY INDEXEDSPECTRUM dyn_list
spectrum parameter index increment condition
dyn_dir par_dir cond_dir spec_dir base node
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create an indexed spectrum dynamic list entry

PARAMETERS

dyn_list required string global replace default: ”
Dynamic list name specification.

spectrum required string replace default: ”
Name specification of spectrum array.

parameter required string replace default: ”
List of data element members. The number must match

index required string replace default: ”
Data element members or multiwindow for index.
Specify a multi window with directory.

increment string default: ”
Data element member to be used as increment.

condition string default: ”
Name of a condition. If specified the spectrum is
filled only, if the condition was true.

dyn_dir string global replace default: ’$DYNAMIC’
Default directory

par_dir string global replace default: ’DATA’
Default directory

cond_dir string global replace default: ’$CONDITION’
Default directory

spec_dir string global replace default: ’$SPECTRUM’
Default directory

base string global replace default: ’DB’
Default data base name

node string global replace default: 'E'
Default node name

/UPDATE switch default: ''
Update dynamic list (then it becames valid for a running analysis immediately.

/[NO] CHECK switch default: /CHECK
Do dynamic list checking by attaching it

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base.

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DYNAMIC ENTRY MULTIWINDOW

CREATE DYNAMIC ENTRY MULTIWINDOW dyn_list condition parameter dyn_dir par_dir cond_dir base node
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create a multiwindow condition dynamic list entry

PARAMETERS

dyn_list required string global replace default: ''
Dynamic list name specification.

condition required string replace default: ''
Specification of module as image(module).

parameter required string replace default: ''
Data element member specification.

dyn_dir string global replace default: '$DYNAMIC'
Default directory

par_dir string global replace default: 'DATA'
Default directory

cond_dir string global replace default: '$CONDITION'
Default directory

base string global replace default: 'DB'
Default data base name

node string global replace default: 'E'
Default node name

/UPDATE switch default: ''
Update dynamic list (then it becames valid for a running analysis immediately.

/[NO] CHECK switch default: /CHECK
Do dynamic list checking by attaching it

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base.

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DYNAMIC ENTRY PATTERN

CREATE DYNAMIC ENTRY PATTERN dyn_list condition parameter dyn_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create a pattern condition dynamic list entry

PARAMETERS

dyn_list required string global replace default: ''
Dynamic list name specification.

condition required string replace default: ”
Pattern condition name specification.

parameter required string replace default: ”
List of data element name specifications. The
number of elements must match the internal dimension of the condition.

dyn_dir string global replace default: '$DYNAMIC'
Default directory

par_dir string global replace default: 'DATA'
Default directory

cond_dir string global replace default: '$CONDITION'
Default directory

base string global replace default: 'DB'
Default data base name

node string global replace default: 'E'
Default node name

/MASTER switch default: ”
Master procedure (executed first)

/UPDATE switch default: ”
Update dynamic list (then it becames valid for a
running analysis immediately.

/[NO] CHECK switch default: /CHECK
Do dynamic list checking by attaching it

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base.

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DYNAMIC ENTRY POLYGON

CREATE DYNAMIC ENTRY POLYGON dyn_list condition parameter polygon dyn_dir par_dir cond_dir poly_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create a polygon condition dynamic list entry

PARAMETERS

dyn_list required string global replace default: ”
Dynamic list name specification.

condition required string replace default: ”
Name specification of polygon condition.

parameter required string replace default: ”
List of two data element members.

polygon string replace default: ”
Name of a polygon.

dyn_dir string global replace default: '$DYNAMIC'
Default directory

par_dir string global replace default: 'DATA'
Default directory

cond_dir string global replace default: '$CONDITION'
Default directory

poly_dir string global replace default: '$POLYGON'
Default directory

base string global replace default: 'DB'
Default data base name

node string global replace default: 'E'
Default node name

/MASTER switch default: ”
Master procedure (executed first)

/UPDATE switch default: ”
Update dynamic list (then it becames valid for a running analysis immediately.

/[NO] CHECK switch default: /CHECK
Do dynamic list checking by attaching it

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base.

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DYNAMIC ENTRY PROCEDURE

CREATE DYNAMIC ENTRY PROCEDURE dyn_list module parameter condition dyn_dir par_dir cond_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create a procedure call dynamic list entry

PARAMETERS

dyn_list required string global replace default: ”
Dynamic list name specification.

module required string replace default: ”
Specification of module as image(module).

parameter string replace default: ”
List of data element name specifications. The pointers to these data elements are passed to the procedure.

condition string default: ”
Name of a condition. If specified the procedure is called only, if the condition was true.

dyn_dir string global replace default: '$DYNAMIC'
Default list directory

par_dir string global replace default: 'DATA'
Default parameter directory

cond_dir string global replace default: '$CONDITION'
Default condition directory

base string global replace default: 'DB'
Default data base name

node string global replace default: 'E'
Default node name

/MASTER switch default: "
Master procedure (executed first)

/UPDATE switch default: "
Update dynamic list (then it becames valid for a running analysis immediately.

/[NO] CHECK switch default: /CHECK
Do dynamic list checking by attaching it

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base.

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DYNAMIC ENTRY SCATTER

CREATE DYNAMIC ENTRY SCATTER dyn_list picture process condition dyn_dir base node
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create a scatter plot dynamic list entry

PARAMETERS

dyn_list	required string global replace default: ” Dynamic list name specification.
picture	required string replace default: ” Name specification of picture.
process	required string replace default: ” Name of display process.
condition	string default: ” Name of a condition. If specified the scatter points are sent only, if the condition was true.
dyn_dir	string global replace default: '$DYNAMIC' Default directory
base	string global replace default: 'DB' Default data base name
node	string global replace default: 'E' Default node name
/UPDATE	switch default: ” Update dynamic list (then it becames valid for a running analysis immediately.
/[NO] CHECK	switch default: /CHECK Do dynamic list checking by attaching it
/[NO] KEEP_MAP	switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base.
Caller	MDBM, MGOODBM
Author	H.G.Essel

CREATE DYNAMIC ENTRY SPECTRUM

CREATE DYNAMIC ENTRY SPECTRUM dyn_list
spectrum parameter increment condition
dyn_dir par_dir cond_dir spec_dir base node
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create a spectrum dynamic list entry

PARAMETERS

dyn_list required string global replace default: ”
Dynamic list name specification.

spectrum required string replace default: ”
Name specification of spectrum array.

parameter required string replace default: ”
List of data element members. The number must match the spectrum dimension.

increment string default: ”
Data element member to be used as increment.

condition string default: ”
Name of a condition. If specified the spectrum is filled only, if the condition was true.

dyn_dir string global replace default: '$DYNAMIC'
Default directory

par_dir string global replace default: 'DATA'
Default directory

cond_dir string global replace default: '$CONDITION'
Default directory

spec_dir string global replace default: '$SPECTRUM'
Default directory

base string global replace default: 'DB'
Default data base name

node string global replace default: 'E'
Default node name

/UPDATE switch default: ”
Update dynamic list (then it becames valid for a running analysis immediately.

/[NO] CHECK switch default: /CHECK
Do dynamic list checking by attaching it

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base.

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DYNAMIC ENTRY WINDOW

CREATE DYNAMIC ENTRY WINDOW dyn_list condition parameter dyn_dir par_dir cond_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP

PURPOSE Create a window condition dynamic list entry

PARAMETERS

dyn_list required string global replace default: ”
Dynamic list name specification.

condition required string replace default: ”
name specification of window condition.

parameter string replace default: ”
List of data element member specifications.

dyn_dir string global replace default: ’$DYNAMIC’
Default directory

par_dir string global replace default: ’DATA’
Default directory

cond_dir string global replace default: '$CONDITION'
Default directory

base string global replace default: 'DB'
Default data base name

node string global replace default: 'E'
Default node name

/MASTER switch default: "
Master procedure (executed first)

/UPDATE switch default: "
Update dynamic list (then it becames valid for a running analysis immediately.

/[NO] CHECK switch default: /CHECK
Do dynamic list checking by attaching it

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base.

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE DYNAMIC LIST

CREATE DYNAMIC LIST dyn_list entries dyn_dir pool buffer base node /[NO]KEEP_MAP

PURPOSE Create a dynamic list in a Data Base

PARAMETERS

dyn_list Dynamic list name specification
required common replaced

entries Number of entries
replaced default:'100'

dyn_dir	Default Directory replaced common default:'$DYNAMIC'
pool	Pool name replaced default:'$DYNAMIC'
buffer	Buffer size in bytes replaced default:'0' (means 80*entries)
base	Default Data Base name replaced common default:'DB'
node	Default node name replaced common default:'E'
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
Caller	M$DMCMD
Author	H.G. Essel
File name	M$ACRDL.PPL
Dataset	-
EXAMPLE	CRE DYN LIST L1 200 create dynamic list L1 in Pool $DYNAMIC with 200 entries

CREATE ELEMENT

CREATE ELEMENT name pool typename refer datalength
cluster queuehead dir base node
/[NO]PROTECT
/[NO]REPLACE
/[NO]KEEP_MAP

PURPOSE	Create a Data Element in a Data Base
PARAMETERS	

name Data Element name (with index)
required common replaced default

pool Pool name
required common replaced default

typename Type name
required

refer List of refer values
optional

datalength Size in bytes
replaced default:'0'

cluster Cluster size in bytes
replaced default:'16'

queuehead Queue header specification
optional

dir Default Directory
common replaced default:'DATA'

base Default Data Base name
common replaced default:'DB'

node Default node name
common replaced default:'E'

/[NO] PROTECT Protect deletion of new Data Element
default:'/NOPROTECT'

/[NO] REPLACE Replace old Data Element
default:'/NOREPLACE'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CREA ELE [PARAM]CALIB(1:10) PARA PARTYP 256 CLUST=16
create the Data Element name array CALIB with 10 members in Directory PARAM. The data of 256 bytes each will be in the Pool PARA. The Data type is PARTYP. PARAM, PARA, and PARTYP must exist already. The cluster size is 16 bytes.

Caller M$DMCMD

Author M. Richter

File name M$ACRDE.PPL

Dataset -

CREATE LINK

CREATE LINK link_from link_to dir base node
/MULTIPLE
/[NO]KEEP_MAP

PURPOSE Create a link between two Data Elements

PARAMETERS

link_from Source Data Element name specification
required, common default

link_to Target Data Element name specification
required, common default

dir Default Directory
required, common default

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/MULTIPLE Multiple identical links allowed

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CRE LIN [EVE]KAIN [ADAM]ABEL
create a link between the Data Element KAIN from Directory EVE and the Data Element ABEL from Directory ABEL, both in Data Base DB.

Caller	M$DMCMD
Author	M. Richter
File name	M$ACRLI.PPL
Dataset	-

CREATE OVERLAY

CREATE OVERLAY picture frame spectrum xparam yparam trans=(xfactor,xoffset,yfactor,yoffset) dynshift node base pic_dir spec_dir par_dir
/[NO]KEEP_MAP

PURPOSE Add spectrum or scatterparameter to frame of a picture data element

PARAMETERS

picture	Name of the picture data element
frame	Number of picture frame for which an overlay should be created.
spectrum	Name of new spectrum, the spectrum could be one or two dimensional.
xparam	X-parameter for additional scatterplot parameters
yparam	Y-parameter for additional scatterplot parameters
xoffset	Offset X-direction for one dim. spectra
xfactor	Factor X-direction for one dim spectra
yoffset	Offset Y-direction for one dim spectra
yfactor	Factor Y-direction for one dim spectra
dynshift	Dynamic Y-offset
entries	Number of overlays which should be allocated for each frame.
node	Default node for Data Base

base	Default Data Base
pic_dir	Default Directory for pictures
spec_dir	Default Directory for spectra
par_dir	Default Directory for scatter parameter.
/[NO] KEEP_MAP	Inhibit the unmap of the whole Data Base.
Caller	MDBM,MGOODBM,E$DECMD
Author	W. Spreng

CREATE PICTURE

```
CREATE PICTURE name frames condition object
               pic_dir pic_pool cond_dir par_dir
               base node
   /[NO]PROMPT
   /[NO]KEEP_MAP
```

PURPOSE Create a picture Data Element

PARAMETERS

name	Name specification of the picture Data Element
frames	Number of frames in the picture
condition	Main condition for the picture
object	Parameter-list which should be checked
pic_dir	Default picture Directory
pic_pool	Default picture pool
cond_dir	Default condition Directory
par_dir	Default parameter Directory
base	Default Data Base

node	Default node name
/[NO] PROMPT	Enter interactive prompting menu:
	/NOPROMPT no prompting
	/PROMPT prompting is performed (DEFAULT)
/[NO] KEEP_MAP	Inhibit the unmap of the whole Data Base.
	/NOKEEP_MAP Data Base will be deattached
	/KEEP_MAP Keep Data Base (DEFAULT)

Caller	MDISP
Action rout.	D$CRPI
Author	W. Spreng

CREATE POLYGON

CREATE POLYGON polygon points
poly_dir poly_pool base node
/[NO]KEEP_MAP

PURPOSE Create a polygon in a Data Base

PARAMETERS

polygon polygon name specification
required common replaced

points Number of points
replaced default:'1'

poly_dir Default Directory
replaced common default:'$POLYGON'

poly_pool Pool name
replaced default:'$PIC_POOL'

base Default Data Base name
replaced common default:'DB'

node Default node name
replaced common default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DECMD

Author H.G. Essel

File name E$ACRPO.PPL

Dataset -

EXAMPLE CRE POLY p1 20
create polygon p1 in Pool $PIC_POOL with 20 points.

CREATE POOL

CREATE POOL pool areaminsize base /[NO]KEEP_MAP

PURPOSE Create a Pool in a Data Base

PARAMETERS

pool Pool name
common default:'DATA'

areaminsize Minimum size of Areas in bytes
replace default:'512'

base data
required, common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CRE POOL ALPHA 10240 DB
create the Pool ALPHA in the Data Base DB. The Areas of this Pool will have a size of at least 10 kByte.

Caller M$DMCMD

Author M. Richter

File name M$ACRPO.PPL

Dataset -

CREATE SPECTRUM

CREATE SPECTRUM name type limits binsize
spec_dir spec_pool base node maxspec
branch crate station offset
/CAMAC
/[NO]DOCU
/[NO]MEANVALUES
/[NO]SQW
/[NO]ERRV
/[NO]ERRH
/[NO]VARBINS
/DYNAMIC/STATIC
/ANALOG/DIGITAL
/[NO]KEEP_MAP

PURPOSE create a spectrum

PARAMETERS

name name of spectrum
required

type data type of spectrum, may be S,H,I,L,R,D or N
replace default:'L'

limits limits of the spectrum
replace default:'(0,1023)'

binsize size of bins
replace default:'1'

spec_dir default Directory
replace default='$SPECTRUM'

spec_pool default pool
replace default='$SPEC_POOL'

base default Data Base
replace default='DB'

node default node
replace default='E'

maxspec maximum number of spectra
default=1024

branch Number of CAMAC branch
replace default=0

crate Number of CAMAC crate on branch
replace default=1

station Number of MR2000 in crate
replace default=2

offset Start address of data in MR2000
replace default=0

/[NO] DOCU if on documentation and history are held
replace default:'/NODOCU'

/[NO] MEANVALUES mean values of the bins are held
replace default:'/NOMEANVALUES'

/[NO] SQW if on sum of square weights per bin are held
replace default:'/NOSQW'

/[NO] ERRV if on vertical errors are held
replace default:'/NOERRV'

/[NO] ERRH if on horizontal errors are held
replace default:'/NOERRH'

/[NO] VARBINS if on spectrum has variable binsize
replace default:'/NOVARBINS'

/DYNAMIC/STATIC possibility of modifications of attributes
replace default:'/DYNAMIC'

/ANALOG/DIGITAL /ANALOG for floating point accumulation, /DIGITAL for accumulation of integers.
The difference is that in analog spectra a bin represents an intervall, but in digital spectra a number (or a range of numbers). Therefore digital spectra have one more bin for the last number. For analog spectra the upper limit is excluded from last bin. Digital spectra are displayed shifted by .5 to the left to center the numbers ticks.
replace default:'/ANALOG'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CREATE SPECTRUM emil H (0,2000,1,10) /NODOCU
a two-dimensional spectrum 'emil' is created containing one-byte data in the limits x:0 to 2000, y:1 to 10 . No documentation will be held

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$CRESP.PPL

Dataset -

CREATE TABLE CONDITION

CREATE TABLE CONDITION name entries
directory pool base node
/[NO]KEEP

PURPOSE create condition bit table

PARAMETERS

name string replace default: 'ANCO'
name of the table

entries integer replace default: 1024
Number of entries (conditions)

directory string replace default: '$ANLTABS'
default directory (will be created).

pool string replace default: '$COND_POOL'
default pool

base string global replace default: 'DB'
default data base

node string global replace default: 'E'
default node

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE TABLE SPECTRUM

CREATE TABLE SPECTRUM name entries
directory pool base node
/[NO]KEEP

PURPOSE create spectrum bit table

PARAMETERS

name string replace default: 'ANSP'
name of the table

entries integer replace default: 1024
Number of entries (spectra)

directory string replace default: '$ANLTABS'
default directory (will be created).

pool string replace default: '$SPEC_POOL'
default pool

base string global replace default: 'DB'
default data base

node string global replace default: 'E'
default node

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller MDBM, MGOODBM

Author H.G.Essel

CREATE TYPE

CREATE TYPE typefilename base
/COMPILE/REFERFILE/COMPREF/NOCOMPNOREF
/[NO]KEEP_MAP

PURPOSE Create a Type descriptor from a PL/I structure.

PARAMETERS

typefilename Type descriptor name. A filename with a leading '@' (default file extension = .TXT) or a library module also with a leading '@', lib(module). This string must be given in double quotes "".
required

base Data Base name
required, common default

/COMPILE /REFERFILE /COMPREF /NOCOMPNOREF Test features:
/COMPILE : performs a test compilation of the Type structure

/REFERFILE : Creates a file with the REFER values of Type structure. The name will be identical to the Type structure name, but the first character will be replaced aby an 'R'.
/COMPREF : performs both compilation and REFER file creation.
/NOCOMPNOREF : performs neither a test compilation nor a creation of the REFER file.
default:'/NOCOMPNOREF'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CRE TYPE "@GOOTYP(SM$DL)" DB
Create the Type SM$DL in Data Base from the library GOOTYP.
CRE TYPE DB "@PRIVAT"
Create the Type PRIVAT in Data Base from the callers file PRIVAT.TXT.

Caller M$DMCMD

Author M. Richter

File name M$ACRTY.PPL

Dataset -

DECALIBRATE SPECTRUM

DECALIBRATE SPECTRUM spectrum spec_dir base node

PURPOSE Disconnect a spectrum from its calibration.

PARAMETERS

spectrum Name of spectrum

spec_dir Directory for spectrum Data Element.

base Data Base name

node	Node name for Data Base file
Author	W. Spreng
Caller	MDBM,MGOODBM

DECOMPRESS BASE

DECOMPRESS BASE file base basefile
/MOUNT

PURPOSE	Restores compressed and copied data base. Command is executed by MUTIL.
PARAMETERS	
file	required string default: ” Name of input file containing compressed base.
base	required string default: ” name of the data base to be restored.
basefile	required string default: ” name of data base file.
/MOUNT	switch default: Mount restored data base.
Caller	MDBCOPY
Author	H.G.Essel

DEFINE PICTURE SETUP

DEFINE PICTURE SETUP picture rows frames pic_dir base node
/EQUAL

PURPOSE	Define frame organization on screen for one picture
PARAMETERS	
picture	Picture name specification
rows	Number of frame-rows on screen.
frames	Number of frame in each row.
pic_dir	Picture Directory
base	Data Base name
node	Node name
/EQUAL	The frames should be of equal size
Caller	MDBM, MGOODBM
Author	W. Spreng

DELETE CALIBRATION

DELETE CALIBRATION calibration node base cal_dir
/[NO]LOG
/[NO]KEEP_MAP

PURPOSE	Delete a calibration Data Element.
PARAMETERS	
calibration	Calibration name specification
node	Default node name
base	Default Data Base
cal_dir	Default Directory
/[NO] LOG	Display picture name after deletion
	/NOLOG nothing is displayed.

/LOG Names of deleted pictures are shown.

/[NO] KEEP_MAP Keep data base mapping context.

/NOKEEP_MAP Data Base will be dettached

/KEEP_MAP Keep Data Base (DEFAULT)

Caller MDBM,MGOODBM

Author W. Spreng

DELETE CONDITION

DELETE CONDITION name cond_dir base node
/[NO]CONFIRM
/[NO]KEEP_MAP

PURPOSE delete a condition

PARAMETERS

name String replace default: none Name of condition

cond_dir String replace default: '$CONDITION' Name of condition directory

base String replace default: 'DB' Name of Data Base

node String replace default: '*' Name of Node

/[NO] CONFIRM Switch default : '/NOCONFIRM'
Noconfirm to delete condition

/[NO] KEEP_MAP Switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base

EXAMPLE DEL CONDITION D::DB:[OTTO]S1

Caller MDBM

Author K.Winkelmann

File name GOO$DE:E$DLCO.PPL

Dataset -

DELETE DYNAMIC ENTRY

DELETE DYNAMIC ENTRY dyn_type dyn_list
namelist aux dyn_dir base node
/UPDATE
/[NO]KEEP_MAP

PURPOSE Delete a Dynamic List entry

PARAMETERS

dyn_type Entry type (* = all types)
common default:'SPECTRUM'

dyn_list Dynamic List name specification
required common default

namelist Name list: base:[dir]name,... or * for all entries
default:'*'

aux Used by scatter plot
default:'*'

dyn_dir Default directory for Dynamic List
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/UPDATE Update

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author H.G. Essel

File name M$ADLLE.PPL

Dataset -

EXAMPLE DEL DYN ENT SPEC DYNA1 * /UPD
delete all entries in the Dynaic List DYNA1 with the type SPECTRUM and update the list.

DELETE DYNAMIC LIST

DELETE DYNAMIC LIST dyn_list dyn_dir base node /[NO]KEEP_MAP

PURPOSE Delete a Dynamic List

PARAMETERS

dyn_list Dynamic List name specification
required common default

dyn_dir Default Directory for Dynamic List
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE DEL DYN LIST DYNA1
delete the Dynamic List 'DYNA1' in the last Directory requested.

Caller M$DMCMD

Author H.G. Essel

File name M$ADLDL.PPL

Dataset -

DELETE ELEMENT

DELETE ELEMENT name dir base node
/UNPROTECT
/[NO]KEEP_MAP

PURPOSE Delete a Data Element

PARAMETERS

name Node::base:[dir]name(i)->type(i)
The name array index might be a wild card with the * as all members. If name is a name array but i was not given then all members of the name array will be deleted (like (*)).
required common default

dir Default Directory
common default: 'DATA'

base Default Data Base name
common default: 'DB'

node Default node name
common default: 'E'

/UNPROTECT Overide deletion protection of Data Element

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ADLDE.PPL

Dataset -

EXAMPLE DEL DB:[DATA]ADAM
delete the Data Element 'ADAM' in the Directory 'DATA' of the Data Base 'DB'.

DELETE LINK

DELETE LINK link_from link_to dir base node
/ALL
/MATCH/IN/OUT
/[NO]KEEP_MAP

PURPOSE Delete Data Element link(s)

PARAMETERS

link_from Target Data Element name specification
required common default

link_to Target Data Element name specification
required common default

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/ALL Delete all links selected by /SELECT

/MATCH/IN/OUT Matching, incomming, outgoing links
replaced default:'/MATCH'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ADLLI.PPL

Dataset -

EXAMPLE DEL LINK DB:[EVE]KAIN DB:[ADAM]ABEL
delete link between the Data Element 'KAIN' of the
Directory 'EVE' and the Data Element 'ABEL' of the Directory 'ADAM',
both in Data Base 'DB'.

DELETE OVERLAY

DELETE OVERLAY picture frame node base pic_dir
/[NO]KEEP_MAP

PURPOSE Delete the defined overlayed spectra or scatterplot parameters for the specified frames.

PARAMETERS

picture Name of picture.

frame Number or range of frames.

node default node name.

base Default Data Base name.

pic_dir Default Picture Directory.

/[NO] KEEP_MAP Inhibit the unmap of the whole Data Base.

/NOKEEP_MAP Data Base will be deattached

/KEEP_MAP Keep Data Base (DEFAULT)

Caller MDBM,MGOODBM,E$DECMD

Author W. Spreng

DELETE PICTURE

DELETE PICTURE picture node base pic_dir
/[NO]LOG
/[NO]KEEP_MAP

PURPOSE Delete a Picture Data Element.

PARAMETERS

picture Picture name specification

node Default node name

base Default Data Base

pic_dir Default Directory

/[NO] LOG Display picture name after deletion

/NOLOG	nothing is displayed.
/LOG	Names of deleted pictures are shown.

/[NO] KEEP_MAP Keep data base mapping context.

/NOKEEP_MAP	Data Base will be dettached
/KEEP_MAP	Keep Data Base (DEFAULT)

Caller MDISP,MGOODISP

Author W. Spreng

DELETE POLYGON

DELETE POLYGON name poly_dir base node
/[NO]CONFIRM
/[NO]KEEP_MAP

PURPOSE delete a polygon

PARAMETERS

name String replace default :
name of polygon, may be fully qualified
or wildcarded

poly_dir String replace default : '$POLYGON'
default directory, taken if directory is not
in the name specification

base String replace default : 'DB'
default base, is taken if base name is not in the
name specification

node String replace default : '*'
default node ,...

/[NO] CONFIRM Switch default : '/NOCONFIRM'
Noconfirm to delete polygon

/[NO] KEEP_MAP Switch default : '/KEEP_MAP'
Inhibit the unmap (detach) of the whole Data Base

EXAMPLE DEL POLYGON D::DB:[$polygon]POLY_1

Caller MDE

Author H.G.Essel

File name GOO$DE:E$DLPO.PPL

Dataset -

DELETE POOL

DELETE POOL pool base
/[NO]KEEP_MAP

PURPOSE Delete a Data Base Pool

PARAMETERS

pool Pool name
required common default

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE DEL POOL ADAM DB

Caller M$DMCMD

Author M. Richter

File name M$ADLPO.PPL

Dataset -

DELETE SECTION

DELETE SECTION base

PURPOSE Delete Global Section attributes

PARAMETERS

base Global Section name
required common default

EXAMPLE -

Caller M$DMCMD

Author M. Richter

File name M$ADLGS.PPL

Dataset -

DELETE SPECTRUM

DELETE SPECTRUM name spec_dir base node
/CAMAC
/[NO]CONFIRM
/[NO]KEEP_MAP

PURPOSE delete a spectrum

PARAMETERS

name String replace default :
name of spectrum, may be fully qualified or wildcarded

spec_dir String replace default : '$SPECTRUM'
default directory, taken if directory is not in the name specification

base String replace default : 'DB'
default base, is taken if base name is not in the name specification

node String replace default : '*'
default node ,...

/CAMAC Switch default : "
Delete CAMAC spectra only

/[NO] CONFIRM Switch default : '/NOCONFIRM'
Noconfirm to delete spectra

/[NO] KEEP_MAP Switch default : '/KEEP_MAP'
Inhibit the unmap (detach) of the whole Data Base

EXAMPLE DEL SPECTRUM D::DB:[OTTO]S1

Caller MDE

Author K.Winkelmann

File name GOO$DE:E$DLSP.PPL

Dataset -

DISMOUNT BASE

DISMOUNT BASE base

PURPOSE Delete Global Section attribute of Data Base

PARAMETERS

base name of Data Base
common replaced default

EXAMPLE DISMO DATA DB

Author M.Richter

Caller M$DMCMD

File name M$DMDB.PPL

DUMP SPECTRUM

DUMP SPECTRUM name spec_dir base node output /[NO]KEEP_MAP

PURPOSE dump spectra to file 'outfile' in ASCII fornmat for transfer to SATAN VSAM library on the IBM

PARAMETERS

name	name of spectrum (wildcard) default = '*'
spec_dir	default directory required common default = '$SPECTRUM'
database	default base required common default = 'DB'
node	default node required common default = 'E'
output	output file required common default = 'OUTFILE'
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
Caller	E$DMPCM
Author	D. Schall
File name	E$DMPSP.PPL
Dataset	-
EXAMPLE	DUMP SPEC ALPHA OUTP=ALPHA1.DATA dump the spectrum 'ALPHA' to the file 'ALPHA1.DATA' in ASCII format for transfer to SATAN VSAM library on the IBM

FREEZE CONDITION

FREEZE CONDITION name cond_dir base node
/ON/OFF
/[NO]KEEP_MAP

PURPOSE freeze a condition

PARAMETERS

name	name of condition required
cond_dir	default Directory replace default:'$CONDITION'
base	default Data Base replace default:'DB'
node	default node replace default:'E'
/ON	condition result bit is set ON
/OFF	condition result bit is set OFF
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
EXAMPLE	-
Caller	E$DECMD
Author	K.Winkelmann
File name	GOO$DE:E$FRECO.PPL
Dataset	-

FREEZE SPECTRUM

FREEZE SPECTRUM name spec_dir base node
/[NO]KEEP_MAP

PURPOSE freeze a spectrum, inhibit accumulation

PARAMETERS

name	name of spectrum required

spec_dir default Directory
replace default:'$SPEC'

base default Data Base
replace default:'DB'

node default node
replace default:'E'
replace default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$FRESP

Dataset -

LOCATE BASE

LOCATE BASE base
/[NO]KEEP_MAP

PURPOSE Locate a Data Base
For test purpose only.

PARAMETERS

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALODB.PPL

Dataset -

EXAMPLE LOC DATABASE DB
locate the Data Base 'DB'.

LOCATE DIRECTORY

LOCATE DIRECTORY dir base
/[NO]KEEP_MAP

PURPOSE Locate a Data Element Directory
For test purpose only.

PARAMETERS

dir Data Element Directory name
required common default

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALODI.PPL

Dataset -

EXAMPLE LOC DIR EVE DB
locate the Directory 'EVE' in the Data Base 'DB'.

LOCATE ELEMENT

LOCATE ELEMENT name
/[NO]KEEP_MAP

PURPOSE Locate a Data Element name array
For test purpose only.

PARAMETERS

name node::base:[dir]element-name(i)
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALODE.PPL

Dataset -

EXAMPLE LOC ELEM DB:[EVE]ADAM
locate the Data Element 'ADAM' of Directory 'EVE'
in Data Base 'DB'.

LOCATE ID

LOCATE ID element dir base
/[NO]KEEP_MAP

PURPOSE Locate a Data Element by Directory index
For test purpose only.

PARAMETERS

element ID of Data Element
replaced default:'1'

dir ID of Directory
replaced default:'1'
common default

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALOID.PPL

Dataset -

EXAMPLE LOC ID 12 3 DB
locate Data Element with the index '12' in the
Directory with the index '3' of Data Base 'DB'.

LOCATE POOL

LOCATE POOL pool base
/[NO]KEEP_MAP

PURPOSE Locate a Pool in a Data Base
For test purpose only.

PARAMETERS

pool Name of Pool
required common default

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller	M$DMCMD
Author	M. Richter
File name	M$ALOPO.PPL
Dataset	-
EXAMPLE	LOC POOL ADAM DB locate the Pool 'ADAM' in the Data Base 'DB'.

LOCATE QUEUEELEMENT

LOCATE QUEUEELEMENT element
/[NO]KEEP_MAP

PURPOSE	Locate a queue Data Element name array For test purpose only.
PARAMETERS	
element	Node::base:[dir]name(i)->type(i) required
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
Caller	M$DMCMD
Author	M. Richter
File name	M$ALOQE.PPL
Dataset	-
EXAMPLE	LOC QUE DB:[EVE]ADAM->L(2) locate the Data Element with the Data Type 'L' and the name array index '2' queued to the Data Element 'ADAM' of Directory 'EVE' in the Data Base 'DB'.

LOCATE TYPE

LOCATE TYPE name base
/[NO]KEEP_MAP

PURPOSE Locate a Type descriptor
For test purpose only.

PARAMETERS

name Name of Type descriptor
required

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALOTY.PPL

Dataset -

EXAMPLE LOC TYP $SPECTR DB
locate the Type '$SPECTR' in the Data Base 'DB'.

MODIFY DIRECTORY

MODIFY DIRECTORY dir entries base
/[NO]KEEP_MAP

PURPOSE Modify a Data Element Directory in a Data Base

PARAMETERS

dir Directory name
required common default

entries Number of entries for Data Element Directory
replaced default:'100'

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ARNDI.PPL

Dataset -

EXAMPLE MODI DIR $SPECTRUM 2000
modify the Data Element Directory '$SPECTRUM' of Data Base 'DB' to allow up to 2000 entries in it.

MODIFY FRAME SCATTER

MODIFY FRAME SCATTER picture frame xparam yparam limits condition xletter yletter object node base pic_dir par_dir cond_dir
/XLIN /XLOG /XSQRT [=X_SCALE]
/YLIN /YLOG /YSQRT [=Y_SCALE]
/[NO]ROTATE
/[NO]LETTER
/[NO]NUMBER

PURPOSE Modify a single frame in a picture data element.
Specify only the parameters which should be changed

PARAMETERS

picture	Name of picture frame
frame	Frame which should be modified.
xparam	Parameter for x-axis
yparam	Parameter for y-axis
limits	Limits of displayed spectrum or scatter plot.
condition	Main condition for that frame.
object	Parameter-list which should be checked
xletter	X-lettering on scatterplot axis.
yletter	Y-lettering on scatterplot axis.
node	Default node name for all Data Element name specifications.
base	Default Data Base name for all Data Elements
pic_dir	Default Directory for pictures.
par_dir	default Directory for parameter and condition object
cond_dir	default Directory for condition.
X_SCALE	Scaling mode for X-axis
	/XLIN Linear X-axis
	/XLOG Logarithmic X-axis
	/XSQRT Squareroot X-axis
Y_SCALE	Scaling mode for Y-axis
	/YLIN Linear Y-axis
	/YLOG Logarithmic Y-axis
	/YSQRT Squareroot Y-axis
/ROTATE	Rotate displayed spectra ********** not yet implemented *******
/LETTER	Display lettering
	/LETTER Display lettering on axis

	/NOLETTER	Display no lettering
/NUMBER	Display numbering	
	/NUMBER	Display numbers on axis
	/NONUMBER	Display no numbers on axis

Caller MDISP,MGOODISP,D$DSPCM

Author W. Spreng

MODIFY FRAME SPECTRUM

MODIFY FRAME SPECTRUM picture frame spectrum limits scallim scalefactor node base pic_dir spec_dir
/LIN /LOG /SQRT [=SCALE]
/XLIN /XLOG /XSQRT [=X_SCALE]
/YLIN /YLOG /YSQRT [=Y_SCALE]
/ZLIN /ZLOG /ZSQRT [=Z_SCALE]
/[NO]WINDOW
/[NO]LIFE
/[NO]ROTATE
/[NO]SMOOTH
/[NO]LETTER
/[NO]NUMBER
/NOCAL/CALAX/CALSPEC [=CALIB]
/[NO]CHANNELS
/HISTO/VECTOR/MARKER/CONTOUR-
/POINT/ISO/CLUSTER/SCATTER [=STYLE]
/[NO]ERROR

PURPOSE Modify a single frame in a picture Data Element. Specify only the parameters which should be changed.

PARAMETERS

picture Name of picture frame

frame	Number of frame or range of frames
spectrum	Name of spectrum to be displayed in the frame.
limits	Limits of displayed spectrum
scalim	Limits of scaling axis
scalefactor	Scaling factor to increase scaling axis.
node	Default node name.
base	Default Data Base where the picture is assumed.
pic_dir	Default picture Directory name.
spec_dir	Default spectrum Directory name.
scale	Scaling mode for Y- or Z-axis
	/LIN Linear scaling axis
	/LOG Logarithmic scaling axis
	/SQRT Squareroot scaling axis
X_SCALE	Scaling mode for X-axis
	/XLIN Linear X-axis
	/XLOG Logarithmic X-axis
	/XSQRT Squareroot X-axis
Y_SCALE	Scaling mode for Y-axis
	/YLIN Linear Y-axis
	/YLOG Logarithmic Y-axis
	/YSQRT Squareroot Y-axis
Z_SCALE	Scaling mode for Z-axis
	/ZLIN Linear Z-axis
	/ZLOG Logarithmic Z-axis
	/ZSQRT Squareroot Z-axis
WINDOW	Window switch
	/NOWINDOW Display no windows
	/WINDOW Display all associated windows of spectrum

********** not yet implemented *******

LIFE Life mode switch

/NOLIFE	No life mode
/LIFE	life mode (update event by event)

********** not yet implemented *******

ROTATE Rotate displayed spectra

/NOROTATE	No rotate
/ROTATE	Rotate

********** not yet implemented *******

SMOOTH Binnig mode

/SMOOTH	Smooth binning
/NOSMOOTH	min/max binning

LETTER Display lettering

/LETTER	Display lettering on axis
/NOLETTER	Display no lettering

NUMBER Display numbering

/NUMBER	Display numbers on axis
/NONUMBER	Display no numbers on axis

CALIB Perform calibration

/NOCAL	no calibration performed
/CALAX	Calibrate axis
/CALSPEC	Calibrate spectrum

CHANNELS Display channel numbers

/CHANNELS	Display spectrum channels.
/NOCHANNELS	Display no spectrum channels.

STYLE Define style of displayed spectrum.
For one dimensional spectra:

/HISTO	Draw histograms

/VECTOR	Connect spectrum bins with lines
/MARKER	Sign spectrum contents with markers

For two dimensional spectra:

/CONTOUR	Contour lines are displayed
/CLUSTER	Clusters indicating the count rate
/ISO	Show spectrum as an isometric plot.
/SCATTER	Simulate scatter plot data.

/[NO] ERROR Draw statistical error bars for each bin.

/NOERROR	No error bars drawn.
/ERROR	Error bars are drawn at each bin.

Caller MDBM,MGOODDM

Author W. Spreng

MODIFY TABLE CONDITION

MODIFY TABLE CONDITION name entries
directory pool base node
/[NO]KEEP

PURPOSE modify condition bit table

PARAMETERS

name string replace default: 'ANCO'
name for condition table to be replaced

entries integer required:
number of needed conditions

directory string replace default: '$ANLTABS'
default directory

pool string replace default: '$COND_POOL'
default pool

base string replace default: 'DB'
default data base

node string replace default: 'E'
default node

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole data base

Caller MDBM,MGOODBM

Author Th. KROLL

MODIFY TABLE SPECTRUM

MODIFY TABLE SPECTRUM name entries
directory pool base node
/[NO]KEEP

PURPOSE modify spectrum bit table

PARAMETERS

name string replace default: 'ANSP'
name for spectra table to be replaced

entries integer required:
number of needed spectras

directory string replace default: '$ANLTABS'
default directory

pool string replace default: '$SPEC_POOL'
default pool

base string replace default: 'DB'
default data base

node string replace default: 'E'
default node

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole data base

Caller MDBM,MGOODBM

Author Th. KROLL

MOUNT BASE

MOUNT BASE base basefile
/PERMANENT/TEMPORARY
/GLOBAL_SEC/SYSTEM_GLOBALSEC

PURPOSE Mount an existing Data Base (section)

PARAMETERS

base Data Base name
required common default

basefile Data Base file name
required common default

/PERMANENT/TEMPORARY Section permanence
default:'/PERMANENT'

/GLOBAL_SEC/SYSTEM_GLOBALSE Section scope
default:'/GLOBAL_SEC'

Caller M$DMCMD

Author M. Richter

File name M$AMODB.PPL

Dataset -

EXAMPLE MOU DAT DB DB
mount the Data Base 'DB' using the section file 'DB.SEC' from the default VAX/VMS directory.

PROTECT SPECTRUM

PROTECT SPECTRUM name spec_dir base node
/LOG
/[NO]KEEP_MAP

PURPOSE protect one (or all) spectrum

PARAMETERS

NAME required string global replace default: ''
Name of Spectrum (wildcard) to be protected
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.

SPEC_DIR String global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE String global replace default: 'DB'
Name of Data Base

NODE String global replace default: 'E'
Name of node

/LOG Switch default:
Output names of protected spectra.

[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller mdbm

Author H.G.Essel

READ CAMAC SPECTRUM

READ CAMAC SPECTRUM name spec_dir base node
/ADD
/LOG
/[NO]KEEP_MAP

PURPOSE Read spectrum data from MR2000 into GOOSY spectrum

PARAMETERS

NAME required string global replace default: ”
Name of Spectrum (wildcard) to be copied
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.

SPEC_DIR String global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE String global replace default: 'DB'
Name of Data Base

NODE String global replace default: 'E'
Name of node

/LOG Switch default: ”
Output list of copied spectra

/ADD Switch default: ”
Add spectrum channel contents rather than overwrite.

[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller mdbm

Author H.G.Essel

SET CONDITION PATTERN

SET CONDITION PATTERN name pattern invpat index cond_dir base node
/[NO]KEEP_MAP

PURPOSE set stored pattern of a pattern condition

PARAMETERS

name String default :
Name of condition

pattern String default : 0
Specification of pattern
String default : 0
Specification of inverse pattern
Integer default : 1
Internal index

cond_dir String replace default: '$CONDITION'
Default condition Directory

base String replace default : 'DB'
Default Data Base

node String replcae default : '*'
Default node

/[NO] KEEP_MAP Set default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$SECOP.PPL

Dataset -

SET CONDITION WINDOW

SET CONDITION WINDOW condition limits dimension cond_dir base node
/[NO]KEEP_MAP

PURPOSE Set window condition

PARAMETERS

condition Name of window condition

limits Limits for condition window

dimension Dimensions for multi-dimensional condition windows which should be set.

cond_dir Default condition Directory

base Default Data Base name

node Default node name

Caller MDBM,MGOODBM

Author W. Spreng

SET LETTERING

SET LETTERING specname dim text spec_dir base node
/[NO]KEEP_MAP

PURPOSE set lettering at display axisses

PARAMETERS

specname	name of spectrum Wildcards in the directory, data element name and dataelement index are supported. required
dim	dimension (0 ... 8) required
text	text to be written at the axis
spec_dir	default Directory repalce default:'$SPECTRUM'
base	default Data Base repalce default:'DB'
node	default node repalce default:'E'
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
Caller	E$DECMD
Author	K.Winkelmann
File name	GOO$DE:E$SELET.ppl
Dataset	-

SET MEMBER

SET MEMBER mem_spec value dir base node
/[NO]KEEP_MAP

PURPOSE Change value of an Data Element member or a full Data Element member array (wildcarded).

PARAMETERS

mem_spec	Node::base:[dir]name(i)->type(i) required common default
value	Enter new value for DE member required
dir	Default Directory common default:'DATA'
base	Default Data Base name common default:'DB'
node	Default node name common default:'E'
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
Caller	M$DMCMD
Author	M. Richter
File name	M$ASTME.PPL
Dataset	-
EXAMPLE	SET MEMB DB:[DATA]ADAM.ALPHA.BETA 4711 set the value '4711' in the member 'BETA' of structure 'ALPHA' of Data Element 'ADAM' in Directory 'DATA' of Data Base 'DB'.

SET SPECTRUM POINT

SET SPECTRUM POINT spectrum xpoint ypoint file
spec_dir base node
/[NO]KEEP_MAP

PURPOSE Set spectrum channel to specified value.

PARAMETERS

spectrum	Name of spectrum to be set.
xpoint	List of x-values converted into spectrum channels.
ypoint	List of y-values used as spectrum contents.
file	Name of file, used to read X-Y values.
spec_dir	Default spectrum directory.
base	Default data base name.
node	Default node name.
/KEEP_MAP	Keep Data Base mapping context.
Caller	MDBM,MGOODBM,E$DECMD
Author	W. Spreng

SHOW AREA

SHOW AREA area base output
/[NO]FULL
/[NO]DIRECTORY
/[NO]KEEP_MAP
/PRINT

PURPOSE Show an Area in a Data Base

PARAMETERS

area Area name
required common default

base Data Base name
required common default

output optional output file
optional

/[NO] FULL Show full Area information
default: '/NOFULL' Area information without bit map

/[NO] DIRECTORY Show Area directory
default: '/NODIRECTORY'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHAR.PPL

Dataset -

EXAMPLE SHO AR ALPHA OUT='ALPHA.LIS'
show the Area 'ALPHA' of Data Base 'DB' and write
the output into file 'ALPHA.LIS' of the default VAX/VMS directory.

SHOW CALIBRATION

SHOW CALIBRATION calibration output cal_dir base node
/PRINT
/LINKS
/TABLE

PURPOSE Show calibration information

PARAMETERS

calibration Name of calibration.

output Output file

cal_dir Default calibration directory

base Default data base name

node	Default node name
/PRINT	Output file is printed
/LINKS	Links to all spectra are listed.
/TABLE	Show total table contents.
Caller	MDBM, MGOODBM
Author	W. Spreng

SHOW CAMAC SPECTRUM

SHOW CAMAC SPECTRUM name spec_dir
base node output width
/PRINT
/ATTRIBUTES/DATA/ALL/STATUS
/FULL
/MEMBERS
/LOG
/INTEGRAL
/ZERO
/CAMAC
/[NO]KEEP_MAP

PURPOSE show CAMAC spectra

PARAMETERS

name String default: '*'
name of spectrum. Wildcards are supported:
* x* *x *x* x*y Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

spec_dir String replace default: '$SPECTRUM'
Name of Spectrum directory.

base String replace default: 'DB'
Name of Data Base.

node String replace default: '*'
Name of Node.

output String default: none
optional output file

width Integer global replace default: 80
Line length for histogram (80 or 132)

/PRINT Switch default: none
Print output

/WHAT Set replace default: /ATTRIBUTES
/ATTRIBUTES :Display attributes of spectra
/DATA :Display data
/STATUS :calls SHOW TAB/SP
/ALL :Display all

/FULL Switch default: none
Display full spectrum information

/MEMBERS Switch default: none
Display information for all spectrum members

/LOG Switch default: none
Display data in logarithmic format

/INTEGRAL Switch default: none
Display channel contents integral

/ZERO Switch default: none
Display channel contents including zero channels

/CAMAC Switch default: /CAMAC
Display data of MR2000 memory (=default)

/[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

FUNCTION The specified spectrum is accessed and various parts of it are displayed on the terminal. The data part of the spectrum is fetched from the MR2000 memory optionally.

Caller E$SHECM

Author K.WINKELMANN

File name E$SHCSP.PPL

Dataset GOO$DE:E$SHCSP.PPL

SHOW CONDITION

SHOW CONDITION name cond_dir base node output
/PRINT
/FULL
/MEMBERS
/ATTRIBUTES/COUNTERS/FLAGS/STATUS/ALL
/POLY/WIND/MULTI/PATT/COMP/FUNC/ANY
/[NO]KEEP_MAP

PURPOSE show attributes of a condition

PARAMETERS

name String replace default: '*'
Name of condition. Wildcards are supported:
* x* *x *x* x*y
Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.
default=*

cond_dir String replace default: '$CONDITION'
Default Directory

base String replace default: 'DB'
Default Data Base

node String replace default: '*'
Default node

output String replace default: none
Optional output file

/PRINT Switch default: none
Print output

/FULL Switch default: none
Full output

/MEMBERS Switch default: none
Output all members of indexed conditions.

what Set default: none

/ATTRIBUTES	show all attributes of a condition
/COUNTERS	show all counters of a condition
/FLAGS	show all flags
/FULL	show everything
/STATUS	show bit tables and counters

/all show all of them Show bit tables and counters

TYPE Set default: /ANY

/WIND	show window conditions
/MULTI	show multi window conditions
/PATT	show pattern conditions
/COMP	show composed conditions
/POLY	show polygon conditions
/FUNC	show function conditions
/ANY	show all types of conditions

Show type of conditions

KEEP_MAP switch default: /KEEP_MAP

/NOKEEP_MAP	Unmap the Data Base
/KEEP_MAP	inhibit the unmap (detach) of the Base

FUNCTION Condition 'name' is searched and its attributes are listed, esp. type, window limits, pattern bits, related conditions, condition table name

EXAMPLE SHOW CONDITION emil /ATTR
shows all attributes from condition emil

Caller E$SHECM

Author K.Winkelmann

File name E$SHCO.PPL

Dataset -

SHOW DIRECTORY

SHOW DIRECTORY dir base output
/[NO]FULL
/[NO]DIRECTORY
/[NO]KEEP_MAP
/PRINT

PURPOSE Show Data Elements of a Data Base Directory

PARAMETERS

dir Directory name
required common default

base Data Base name
required common default

output optional output file

/[NO] FULL Show full Directory information
default: '/NOFULL' brief Directory information

/[NO] DIRECTORY Show Master Directory
default: '/NODIRECTORY'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHDI.PPL

Dataset -

EXAMPLE SHO DIR EVE OUTP='EVE.LIS' /FULL
show the full information about the Directory 'EVE'
of the Data Base 'DB' and write the output into the file 'EVE.LIS' of the default VAX/VMS directory.

SHOW DYNAMIC LIST

SHOW DYNAMIC LIST dyn_list dyn_type dyn_dir base node output
/[NO]KEEP_MAP
/PRINT

PURPOSE Show Dynamic List (elements)

PARAMETERS

dyn_list Dynamic List name specification
required common default

dyn_type Type of dynamic sublist
common default:'*'

dyn_dir Default Directory
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

output optional output file
optional

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHDL.PPL

Dataset -

EXAMPLE SHO DYN LIS DYNA1 $DYNAMIC DB OUTP='DYNA1.LIS'
show the full information about the Dynamic List 'DYNA1' in Directory 'DYNAMIC' of the Data Base 'DB' and write the output into the file 'DYNA1.LIS' of the default VAX/VMS directory.

SHOW ELEMENT

SHOW ELEMENT name dir base node output
/DECIMAL/HEXADECIMAL/OCTAL/BINARY
/LONGWORD/WORD/BYTE
/[NO]DATA
/[NO]FULL
/[NO]KEEP_MAP
/PRINT

PURPOSE Show a Data Element descriptor and value

PARAMETERS

name Node::base:[dir]name(i)->type(i)
dir, name, type, and index might be given with wildcards '*'
required common default

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

output optional output file
optional
DECIMAL/HEXADECIMAL/OCTAL/BINARY

: Output radix for Data Types
replaced default:'/DECIMAL'

/LONGWORD/WORD/BYTE output format for atomic Data Type Y
replaced default:'/LONGWORD'

/[NO] DATA Shows all the data of a Data Element
default:'/NODATA'

/[NO] FULL Shows all control information of a Data Element
default:'/NOFULL'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHDE.PPL

Dataset -

EXAMPLE SHO ELEM DB:[EVE]ADAM(17) OUTP='ADAM.LIS' /DATA
show the Data Element with the index '17' of the
name array 'ADAM' in Directory 'EVE' of Data Base 'DB' and write the output to the file 'ADAM.LIS' of the default VAX/VMS directory. The data will be shown but only brief control information.
SHO ELEM DB:[EVE]*A*
show all Data Elements of Directory 'EVE'
containing an 'A' in their names. The output will be without data in short form.
SHO ELEM DB:[EVE]BETA->*
show all queued Data Elements of Data Element
'BETA' in Directory 'EVE' of Data Base 'DB'. The output will be without data in short form.

SHOW GOOSY STATUS

SHOW GOOSY STATUS environment p1 p2 p3
/$TMR
/$ANL

PURPOSE	GOOSY Status report
ARGUMENTS	
environment	Optional environment
p1	Process name to be monitored
p2	Process name to be monitored
p3	Process name to be monitored
/$TMR	Monitor $TMR of environment
/$ANL	Monitor $ANL of environment
Information	See HELP MSTATUS, HELP GOOCONTROL
EXAMPLE	GOOSY> SHOW G ST P1=GN_HGE___$TMR $ GSTAT P1=GN_HGE___$TMR $ GSTAT SUSI /$TMR/$ANL
Action rout.	M$GOOST
Author	H.G.Essel

SHOW HOME_BLOCK

SHOW HOME_BLOCK base output
/PRINT
/BITMAP

PURPOSE	Show the Home Block of a Data Base
PARAMETERS	
base	Data Base name required common default
output	optional output file optional

/BITMAP	output bitmap
/PRINT	Print output
Caller	M$SHMCM
Author	M. Richter
File name	M$ASHHB.PPL
Dataset	-
EXAMPLE	SHO HOM DB OUTP='HBDB.LIS' show the Home Block Information of Data Base 'DB' and write it into the file 'HBDB.LIS' of the default VAX/VMS directory.

SHOW LINK

SHOW LINK link_from dir base node
output
/IN/OUT/ALL
/MATCH/TREE
/[NO]KEEP_MAP
/PRINT

PURPOSE	Show Data Element link(s)
PARAMETERS	
link_from	Source Data Element name specification required common default
dir	Default Directory common default:'DATA'
base	Default Data Base name common default:'DB'
node	Default node name common default:'E'

output	optional output file optional
/IN/OUT/ALL	In, out, all links replaced default:'/ALL'
/MATCH/TREE	Matching or total link tree default:'/MATCH'
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
/PRINT	Print output
Caller	M$DMCMD
Author	M. Richter
File name	M$ASHLI.PPL
Dataset	-
EXAMPLE	SHO LINK DB:[EVE]ADAM /OUT OUT='ADAM.LIS' show all outgoing links from Data Element 'ADAM' in Directory 'EVE' of Data Base 'DB' and write the output into the file 'ADAM.LIS' of the VAX/VMS default directory.

SHOW MEMBER

SHOW MEMBER mem_spec dir base node output
/[NO]KEEP_MAP
/PRINT

PURPOSE	Show value of a Data Element member or a full Data Element member array (wildcarded).
PARAMETERS	
mem_spec	Node::base:[dir]name(i)->type(i) required

dir	Default Directory common default:'DATA'
base	Default Data Base name common default:'DB'
node	Default node name common default:'E'
cv_out	optional file for output optional
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
/PRINT	Print output BIN FIXED(15)
Caller	M$DMCMD
Author	M. Richter
File name	M$ASHME.PPL
Dataset	-
EXAMPLE	SHO MEM DB:[EVE]ADAM(12).KAIN OUT='ADAM.LIS' show the member 'KAIN' of the Data Element with the index '12' of the name array 'ADAM' in the Directory 'EVE' of the Data Base 'DB' and write the output into the file 'ADAM.LIS' of the default VAX/VMS directory.

SHOW PICTURE

SHOW PICTURE picture output pic_dir base node
/PRINT
/FULL
/DATA
/[NO]KEEP_MAP

PURPOSE Show picture information

PARAMETERS

picture	Name of picture.
output	Output file
pic_dir	Default picture Directory
base	Default Data Base name
node	Default node name
/PRINT	Output file is printed
/FULL	Short information about all frames is shown.
/DATA	Detailed information about all frames is shown.
/KEEP_MAP	Inhibit the unmapping of the whole Data Base.
Caller	MDBM, MGOODBM
Author	W. Spreng

SHOW POLYGON

SHOW POLYGON name poly_dir base node output
/PRINT
/[NO]FULL
/[NO]DATA
/[NO]KEEP_MAP

PURPOSE show attributes of a polygon

PARAMETERS

name	name of polygon. Wildcards are supported: * x* *x *x* x*y
poly_dir	default Directory replace default='$PICTURE'

base	default Data Base replace default='DB'
node	default node replace default='E'
output	optional output file
/PRINT	print output
/[NO] FULL	show full header
/[NO] DATA	show polygon points
/[NO] KEEP_MAP	inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
EXAMPLE	SHOW POLY emil /DATA shows points of polygon emil
Caller	E$SHECM
Author	H.G.Essel
File name	E$ASHPO.PPL
Dataset	-

SHOW POOL

SHOW POOL pool base output
/[NO]FULL
/[NO]DIRECTORY
/[NO]KEEP_MAP
/PRINT

PURPOSE	Show Areas of a Data Base Pool The name, size, filling level, cluster size, and number of fragments are shown for each Area. The Pool Directory can be shown with /DIRECTORY.

PARAMETERS

pool	Pool name, wild cards are allowed. The Pool name will be ignored for /DIRECTORY. required common default
base	Data Base name required common default
output	optional output file
/[NO] FULL	Show the full information of all Areas of a Pool. Together with the /DIRECTORY option all entries in the Pool Directory are listed. default:'/NOFULL'
/[NO] DIRECTORY	Show information of the Pool Directory Together with the /FULL option all entries in the Pool Directory are listed. With the /DIRECTORY any given Pool name will be ignored. default:'/NODIRECTORY'
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
/PRINT	Print output
Caller	M$DMCMD
Author	M. Richter
File name	M$ASHPO.PPL
Dataset	-
EXAMPLE	SHO POO ADAM OUT='ADAM.LIS' show the Areas of the Pool 'ADAM' of the Data Base 'DB' and write the output into the file 'ADAM.LIS' of the default VAX/VMS directory.

SHOW SPECTRUM

SHOW SPECTRUM name spec_dir base node output width
/PRINT
/ATTRIBUTES/DATA/ALL/STATUS
/FULL
/MEMBERS
/LOG
/INTEGRAL
/ZERO
/CAMAC
/[NO]KEEP_MAP

PURPOSE show spectra

PARAMETERS

name String default: '*'
name of spectrum. Wildcards are supported:
* x* *x *x* x*y Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

spec_dir String replace default: '$SPECTRUM'
Name of Spectrum directory.

base String replace default: 'DB'
Name of Data Base.

node String replace default: '*'
Name of Node.

output String default: none
optional output file

width Integer global replace default: 80
Line length for histogram (80 or 132)

/PRINT Switch default: none
Print output

/WHAT Set replace default: /ATTRIBUTES
/ATTRIBUTES :Display attributes of spectra
/DATA :Display data
/STATUS :calls SHOW TAB/SP
/ALL :Display all

/FULL	Switch default: none Display full spectrum information
/MEMBERS	Switch default: none Display information about all spectrum members.
/LOG	Switch default: none Display data in logarithmic format
/INTEGRAL	Switch default: none Display channel contents integral
/ZERO	Switch default: none Display channel contents including zero channels
/CAMAC	Switch default: none Display CAMAC spectra only
/[NO] KEEP_MAP	Switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
FUNCTION	The specified spectrum is accessed and various parts of it are displayed on the terminal
EXAMPLE	SHOW SPEC OTTO /FULL shows otto and all queued data elements
Caller	E$SHECM
Author	K.WINKELMANN
File name	E$SHSP.PPL
Dataset	GOO$DE:E$SHSP.PPL

SHOW TABLE

SHOW TABLE name table tab_dir base node output
/CONDITION /SPECTRUM /ALL
/CONTENT /COUNTS
/PRINT
/[NO]KEEP_MAP

PURPOSE show flag tables from the analysis

PARAMETERS

NAME String replace default: '*'
Name of items (conditions or spectra)

TABLE String replace default: '*'
Name of table (may be omitted)

TAB_DIR String replace default: '$ANLTABS'
Name of Directory

BASE String replace default: 'DB'
Name of Data Base

NODE String replace default: '*'
Name of node

OUTPUT String replace default: ''
Optional output file

TYPE Set replace default: /ALL /CONDITION : /SPECTRUM : /ALL : Kind of table

CONT Set replace default: /COUNTS /CONTENT : /COUNTS : Display spectrum content or counts

[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

/PRINT Switch default: none
Print output

FUNCTION The contents of the tables (flags for freeze and executed (spectra) or freeze,executed,result and result preset (conditions) are shown.
In addition, condition counters and spectrum
contents and overflows are displayed

EXAMPLE SHO TA /SP

Caller E$SHECM

Author K.Winkelmann

File name GOO$DE:E$SHANT.PPL

Dataset -

SHOW TREE

SHOW TREE base output
/[NO]KEEP_MAP
/PRINT

PURPOSE Show indices of a Data Base Pool Directory
For test purpose only.

PARAMETERS

base Data Base name required common default

output optional output file
optional

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHTR.PPL

Dataset -

EXAMPLE SHO TREE DB
show the name tree of the Pool Directory of the
Data Base 'DB'.

SHOW TYPE

SHOW TYPE type base output

PURPOSE Show a Type descriptor. Wild card are allowed.

PARAMETERS

type required string default: " type descriptor name

base required string global replace default: " Data Base name

output string default: " optional output file

/DATA switch default: /PLI Output of Type descriptor as a PL-I Structure or only the data.

/[NO] KEEP_MAP switch default: /KEEPMAP Inhibit the unmap (detach) of the whole Data Base

/PRINT switch default: Print output on Line Printer.

Caller M$SHMCM

Author M. Richter

START MR2000

START MR2000 branch crate station /INITIALIZE

PURPOSE Start/initialize MR2000.

PARAMETERS

branch Integer replace default: 0
Number of CAMAC branch.

crate Integer replace default: 1
Number of crate on branch.

station	Integer replace default: 2 Station number of MR2000 in crate.
/INITIALIZE	switch default: ” Initialize MR2000 before start.
Caller	mdbm
Author	H.G.Essel

STOP MR2000

STOP MR2000 branch crate station

PURPOSE	Stop MR2000.

PARAMETERS

branch	Integer replace default: 0 Number of CAMAC branch.
crate	Integer replace default: 1 Number of crate on branch.
station	Integer replace default: 2 Station number of MR2000 in crate.
Caller	mdbm
Author	H.G.Essel

SUMUP SPECTRUM

SUMUP SPECTRUM spectrum window file spec_dir base node
/CHAN/CALIB [=CALIBR]
/[NO]OUTPUT
/[NO]APPEND
/[NO]KEEP_MAP

PURPOSE Integrate specified window

PARAMETERS

spectrum name of spectrum

window Limits of integration window

file File for the output of the results.

CALIBR Specifies the unit in which the coordinates are given.

/CHAN	Original spectrum units.
/CALIB	Calibrated units.

/[NO] OUTPUT Output occus additionally onto the specified file.

/[NO] APPEND Results are appended to an existing file.

/[NO] KEEP_MAP Inhibit the unmap of the Data Base.

Caller MDBM,MGOODBM

Author W. Spreng

UNFREEZE CONDITION

UNFREEZE CONDITION name cond_dir base node
/[NO]KEEP_MAP

PURPOSE unfreeze a condition, enable the execution of a condition

PARAMETERS

name	name of condition required
cond_dir	default Directory replace default:'$COND'
base	default Data Base replace default:'DB'
node	default node replace default:'E'
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
EXAMPLE	-
Caller	E$DECMD
Author	K.Winkelmann
File name	GOO$DE:E$UNFCO.PPL
Dataset	-

UNFREEZE SPECTRUM

UNFREEZE SPECTRUM name spec_dir base node
/[NO]KEEP_MAP

PURPOSE unfreeze a spectrum, enable the accumulation

PARAMETERS

name	name of spectrum required
spec_dir	default Directory replace default:'$COND'
base	default Data Base replace default:'DB'

node default node
replace default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$UNFSP.PPL

Dataset -

UNPROTECT SPECTRUM

UNPROTECT SPECTRUM name spec_dir base node
/LOG
/[NO]KEEP_MAP

PURPOSE Unprotect one (or all) spectrum

PARAMETERS

NAME required string global replace default: ''
Name of Spectrum (wildcard) to be unprotected
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.

SPEC_DIR String global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE String global replace default: 'DB'
Name of Data Base

NODE	String global replace default: 'E' Name of node
/LOG	Switch default: Output names of unprotected spectra.
[NO] KEEP_MAP	Switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
Caller	mdbm
Author	H.G.Essel

UPDATE BASE

UPDATE BASE base

PURPOSE	Update a Data Base, write all pages to Section File
PARAMETERS	
BASE	Name of the Data Base common required default
EXAMPLE	UPD DA DB
Caller	M$DMCMD
Author	M. Richter
File name	M$UPDB.PPL
Dataset	-

UPDATE DYNAMIC LIST

UPDATE DYNAMIC LIST dyn_list dyn_dir base node
/[NO]KEEP_MAP

PURPOSE Update a Dynamic List

PARAMETERS

dyn_list Dynamic List name specification
required common default

dyn_dir Default Directory
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author H.G.Essel

File name M$AUPDL.PPL

Dataset -

EXAMPLE UPDAT DYN LI DYNA_1 $DYNAMIC
update the Dynamic List 'DYNA_1' of the Directory '$DYNAMIC' in the Data Base last used.

WRITE CAMAC SPECTRUM

WRITE CAMAC SPECTRUM name spec_dir base node
/ADD
/LOG
/[NO]KEEP_MAP

PURPOSE Write spectrum data from GOOSY spectrum into MR2000

PARAMETERS

NAME required string global replace default: ”
Name of Spectrum (wildcard) to be copied
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.

SPEC_DIR String global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE String global replace default: 'DB'
Name of Data Base

NODE String global replace default: 'E'
Name of node

/LOG Switch default: ”
Output list of copied spectra

/ADD Switch default: ”
Add spectrum channel contents rather than overwrite.

[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller mdbm

Author H.G.Essel

GOOSY Glossary

Analysis Manager ($ANL) Part of the analysis program controlling the data I/O and the event loop.

$ANL The Analysis program as a GOOSY component. Runs in a subprocess named GN_env___$ANL.

$DBM The Data Base Manager as a GOOSY component. Runs in a subprocess named GN_env___$DBM.

$DSP The Display Program as a GOOSY component. Runs in a subprocess named GN_env___$DSP.

$TMR The Transport Manager as a GOOSY component. Runs in a subprocess named GN_env___$TMR.

ATTACH Data Bases, Pools, and Dynamic Lists must be attached before they can be used. The ATTACH operation specifies the protection mode for Data Base Pools.

Branch The CAMAC parallel branch connects up to seven CAMAC crates to a computer Interface, e.g. to the MBD.

Buffer GOOSY buffers have a standard buffer header describing the content of the buffer through type/subtype numbers. A GOOSY buffer may contain list mode data (events) file headers, or other kind of data. Buffers can be sent over DECnet and copied from/to tape and disks. Most GOOSY buffers contain buffer Data Elements.

Buffer Data Element A data structure preceeded by a 4 word header stored in a buffer. The header keeps information about the size and the type of the buffer Data Element.

Buffer Unpack Routine A buffer unpack routine copies one event from the buffer into an event Data Element. It has to control the position of the events in the buffer. It gets passed the pointer to the buffer as argument.

CAMAC **C**omputer **A**utomated **M**easurement **a**nd **C**ontrol. A standard for high-energy physics and nuclear physics data acqusition systems, defined by the ESONE (**E**uropean **S**tandard **O**n **N**uclear **E**lectronics) committee between 1966 and 1969.

CONDITION In contrast to SATAN, GOOSY conditions are independent of spectra. Besides the multi window conditions which are similar to SATAN analyzer conditions, GOOSY provides window-, pattern-, composed- and userfunction-conditions. Each condition has counters associated for true/false statistics. Conditions can be executed in a Dynamic List or by macro the $COND in an analysis routine. Each condition can be used as filter for spectrum accumulation or scatter plots.

CONNECT A calibration can be connected to any number of spectra with the GOOSY command `CALIBRATE SPECTRUM`.

CVC CAMAC VSB Computer. A CAMAC board with a 68030 processor running Lynx, OS9 or pSOS. It can be equipped with ethernet and SCSI and VSB.

Data Base A Data Base is located in a file and has a Data Base name. It is recommended to use the same name for the file and the Data Base. The file type should be .SEC. A logical name may be defined for the Data Base name. To activate a Data Base it must be mounted. It is dismounted during a system shutdown or by command. If a Data Base runs out of space, it can presently NOT be expanded.

Data Base Directory Similar to a VMS disk, GOOSY Data Bases are organized in Directories. They must be created.

Data Base Manager ($DBM) This is a program executing all commands to handle Data Bases. It may run directly in DCL or in a GOOSY environment.

Data Base Pool The storage region of a Data Base is splitted in Pools. All Data Elements are stored in Pools. A Pool can be accessed by a program with READ ONLY protection or with READ/WRITE protection. Pools must be created. They are automatically expanded if necessary, up to the space available in a Data Base.

Data Element A Data Element is allocated in a Data Base Pool. Its name is kept in a Directory. Data Elements can be of atomic Types (scalars or arrays), or of the structure Type (PL/1 structures). Besides the data structure a Data Element can be indexed (one or two dimensional). Such Data Elements are called name arrays. Each name array member has its own data and Directory entry.

Data Element Member Similar to PL/1, the variables in a structure are called members.

Data Element Type GOOSY Data Elements can be PL/1 structures. The structure declarations must be in a file or text library module. They are used to create a Data Element Type in the Data Base and can be included in a program to access the Data Element.

Dynamic List A Dynamic List has several Entries, each specifying an action like condition check or spectrum accumulation. It is executed for each event in the analysis program. The Entries are added or removed by commands even without stopping the analysis.

Dynamic List Entry An Entry in a Dynamic List keeps all information to execute an action. For example, an accumulation Entry contains the spectrum name, an object and optional a condition and an increment parameter.

Dynamic List Executor The part of the analysis program which scans through a Dynamic List for each event executing the actions specified by the Entries.

Environment The Transport Manager and the analysis programs run only in a GOOSY environment which has to be created first. They are started by specific commands. The Display and the Data Base Manager may run under DCL or in a GOOSY environment. The display must run in a GOOSY environment if scatter plots are used. The main difference is that in an environment several programs are 'stand by', whereas in DCL you can run only one program at a time.

Event Packet of data in the input or output stream which is processed by the same program part (see event loop).

Event Buffer Data Element A data structure preceeded by a 4 word header stored in a buffer. The header keeps information about the size and the type of the event buffer Data Element. The event buffer Data Element is copied by unpack routines to event Data Elements.

Event Data Element A Data Element in a Data Base which is used to store events. Event Data Elements are used to copy events from an input buffer into the Data Base or from the Data Base into an output buffer.

Event Unpack Routine An event unpack routine copies one event from the buffer into an event Data Element. Different from a buffer unpack routine, it gets passed the pointer to the event in the buffer as argument.

GOOSY Components GOOSY is composed of components, i.e. programs like the Transport Manager $TMR, the Analysis Program $ANL, the Display $DSP and the Data Base Manager $DBM. Data Base Manager and Display program may be envoced under DCL in a 'stand alone' mode. $TMR and $ANL can run only in a GOOSY environment. Components run in an environment as VAX/VMS subprocesses of the terminal process.

GOOSY Prompter If GOOSY components run in an environment, their commands are the input to the GOOSY prompter. The GOOSY prompter is entered by `GOOSY` and prompts with `SUC: GOOSY>`. Now you can enter GOOSY commands which are dispatched to the appropriate GOOSY components for execution. Single GOOSY commands can be executed from DCL preceding them by `GOOSY`. The prompter exits after the command termination. **The GOOSY prompter can only be used after an environment was created!**

J11 This is an auxiliary crate controller based on a PDP 11/73 processor (type CES 2180 Starburst). Has full PDP instruction set including floating point arithmetic. A J11 running under RSX/11S controls one CAMAC crate and sends the data via DECnet to a VAX.

LAM **L**ook **A**t **M**e. A signal on the CAMAC Dataway, which may request a readout (CAMAC interrupt).

LOCATE In a program, any Data Element must be located, before it can be used. The LOCATE operation returns the pointer to the Data Element. The macro $LOC provides a convenient way to locate spectra, conditions or arbitrary Data Elements.

Mailbox An interprocess communication method provided by VMS. Processes on the same node can send/receive data through mailboxes.

MBD **M**icroprogrammed **B**ranch **D**river from BiRa Systems Inc. supports the protocol of the CAMAC parallel *Branch*, defined by the *CAMAC* standard (GOLDA equivalent: CA11-C). This is an interface between CAMAC and a VAX. It gets data from the crate controllers (J11) and sends them to the transport manager running on a VAX.

MOUNT A GOOSY Data Base must be mounted before it can be accessed. The `MOUNT` operation connects the Data Base name with the Data Base file name.

Object To increment a spectrum or execute a condition, the Dynamic List executor needs a value for the spectrum channel, or a value to compare to window limits. These values are called objects. An object must be a member of a Data Element.

Picture A Picture is a complex display. A picture is a set of up to 64 frames with spectra and/or scatterplots. Once created and specified they remain in a Data Base independent of programs. They are displayed by `DISPLAY PICTURE` command. Pictures are composed of frames.

Picture Frame Each frame is a coordinate system for a spectrum or scatter plot. Up to 64 different frames may inserted to a picture.

Prompter Command interface for GOOSY environment. The GOOSY prompter is called by DCL command `GOOSY`. Then all commands are delivered to the environment components for execution.

Scatter Plot The GOOSY display component can display any pairs of Data Element members event by event in scatter plot mode (live mode). Several scatter plots can be displayed on one screen (pictures). Scatter plots are executed in Dynamic Lists and may be filtered by conditions.

Spectrum A GOOSY spectrum differs from a SATAN analyzer in that there are no windows or conditions associated. A spectrum can be filled in a Dynamic List Entry or in an analysis routine by macro $ACCU.

STARBURST This is an auxiliary crate controller based on a PDP 11/73 processor (type CES 2180 Starburst). Has full PDP instruction set including floating point arithmetic. Each CAMAC crate is controlled by one STARBURST running a standalone program. The STARBURST reads out the crate and sends the data to the MBD.

Supervisor Each environment has a supervisor component. The supervisor dispatches messages between the GOOSY prompter and the environment components.

Transport Manager ($TMR) This program acts as data buffer dispatcher. It gets data buffers from the CAMAC branch (MBD) or via DECnet from a single CAMAC crate (J11) or from a disk/tape file and writes them to disk/tape files, DECnet, and mailboxes. It executes all CAMAC control commands. The $TMR runs only in a GOOSY environment.

Unpack Routine An unpack routine copies one event from the buffer into an event Data Element. There are two types: buffer and event unpack routines. Buffer unpack routines control the whole buffer, event unpack routines only one event.

Index

Contents