GOOSY
Id.: INTRO
Version: 3.0
Date: September, 3 1987
Revised: December, 1 1993

GSI **O**nline **O**ffline **S**Y**stem**

GOOSY Introduction

H.G.Essel

December, 1 1993

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.

Registered Trademarks are not explicitly noted.

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 introduction is written for GOOSY beginners. It assumes that the reader is familiar with basic VAX-VMS concepts and commands. For VAX beginners the 'VMS Introduction' is recommended. The 'GOOSY Introduction' is no reference manual. It provides all information necessary to use GOOSY. Not all features of GOOSY are described in detail but rather a standard subset. A detailed description can be found in the GOOSY manuals.

1.3.1 Overview

- Section 2.1:
  A compact global description of GOOSY. An overview about its structure and capabilities. For readers who want to know something about GOOSY before using it. Not required for the following.
- Section 2.2:
  Describes what to do once before using GOOSY.
- Section 2.3:
  GOOSY command syntax for line and menu input. Not necessary to understand the following but to do something by yourself.
- Sections 2.4, 2.5, 2.6:
  The general ideas about the data management should be known. The more detailed command descriptions can be read later if needed.
- Sections 2.9, 2.10, 2.11, 2.12:
  These sections describe how to use GOOSY.
- Section 2.14:
  People who learn from examples may begin here. The examples can be executed as described.

Chapter 2

GOOSY Introduction

2.1 GOOSY Summary

2.1.1 Summary

GOOSY is a PL/1 based nuclear data acquisition and analysis system implemented on VAX/AXP OpenVMS computers. It gets input from MBD, CAMAC processors (J11, CVC), or VME processors reading out CAMAC modules and/or Fastbus subsystems. The analysis is done by user written routines and/or command controlled analysis tables. Network capabilities support several analysis programs on different DECnet nodes and several displays. A data management allows for comfortable control of PL/1 structured Data Elements. Among others, the main features of GOOSY are:

1. GOOSY is composed of loosely coupled programs like Transport Manager, Display, Analysis, and Data Manager. This structure provides high security and flexibility because the data acquisition is independent of the analysis.
2. The programs may run on different VAX/AXP computers connected via DECnet. The experiment can be controlled from several terminals.
3. The analysis is independent of input which may come from a Transport Manager (CAMAC) via mailbox or DECnet, another analysis via DECnet, or disk or tape.
4. The analysis can be done by user written routines utilizing macros and/or by analysis tables which can be modified and/or (de)activated by interactive commands. The analysis tables control the following actions:
    - check of conditions (window, multiwindow, pattern, boolean, polygon and user defined function conditions) or condition arrays,
    - accumulation of histograms (1- and 2-dimensional, real, integer, and bit spectra with condition filters) or histogram arrays controlled by multiwindow conditions or condition arrays,
    - user written procedures filtered by conditions,
    - density distributions of pairs of any parameters filtered by conditions.
5. The Transport Manager presently supports the parallel CAMAC branch driver MBD and a J11 based auxiliary crate controller CES-2180 Starburst with an Ethernet/DECnet connection. It supports a VME based multiprocessor frontend system controlling CAMAC, Fastbus or foreign subsystems. A CAMAC computer board (CVC) providing a VSB bus to connect CAMAC and/or Fastbus will also be supported. A standard program interface is provided to support other frontend equipment.
6. The GOOSY display provides various tools to handle complex display pictures composed of histograms and distributions.

7. All data of GOOSY (e.g. spectra, conditions, analysis tables, pictures, event data, user defined Data Elements) are stored in Data Bases preserved independently of programs even in case of crashes. They are handled by a Data Manager Program.
8. The analysis of one data stream can be done by several analysis programs running on different DECnet nodes in parallel.

2.1.2 GOOSY Subsystems

Data Management

In a software system for data acquisition and data analysis a large number of differently structured data objects has to be handled. Those data objects could be simple variables like coordinates or calibration parameters or complex structures like a spectrum or a picture. Therefore a Data Base system has been developed and implemented. Data Bases are organized in Directories, similar to OpenVMS, the VAX operating system. All Data Elements have names which are inserted in Directories. Various commands are provided to handle any kind of Data Elements, as CREATE, DELETE, SHOW, COPY, SET. Data Element specifications have the general form:

```
node::base:[directory]name(index).member(index)
```

The node name will be used in the future for remote Data Base access. All Data Elements are structures like PL/1 variables. In addition - and in contrast to a file system - a Data Element name can be indexed. Each member of such an array has the same data structure. The smallest referable entity of a Data Element is any member of its structure.

The Data Base is split into subdivisions called Pools. A Pool is the smallest part of a Data Base which can be mapped into a program's address space where it is contiguous. The mapping specifies the program's access rights to the Pool, e.g. read only or read/write. A Pool is a chain of Data Areas. Areas are contiguous in the file. If the space in a Pool is exhausted, one more Area is added to the chain thus expanding the Pool. All Data Elements are allocated in Pools. All allocations and releases in a Data Base are logged in bit maps. Thus released space can be allocated again.

Any Data Element in the Data Base is described by a Type descriptor. This is used to handle the Data Element, e.g. for CREATE, COPY and SHOW commands. The Type descriptor is generated from a PL/1 structure declaration which can be included in a procedure to access the Data Element. Thus the consistency of the access is provided. The local pointer to the Data Element is returned by locating procedures or macros. After locating the Data Element is protected against deletion.

Several programs can work independently with the same Data Base thus sharing data. Therefore some actions are locked to ensure the integrity of the Data Base and the local mapping contexts in all attached programs.

Command Interface

The GOOSY command syntax is similar to DCL. The commands are not implemented directly in DCL because of conflicting command keywords. They are rather implemented like commands available in DEC utility programs. They can be used in DCL procedures. Through a prompter program commands can be dispatched to subprocesses where they are executed. The commands are defined by subroutine calls. Each command is executed by a PL/1 or FORTRAN routine. Commands are composed of several keywords, e.g.

```
START INPUT NET
START INPUT MAILBOX
START OUTPUT FILE
```

A convenient command input is provided by a menu option. The menu is generated automatically from the command definition. On each menu level interactive help is available. The command parameters can be input in a positional order, by name reference or as qualifiers:

```
CREATE SPECTRUM/DIGITAL LIMITS=(1,1024) DATATYPE=L NAME=name
CREATE SPECTRUM name L (1,1204) /DIGITAL
```

By a logical name mechanism the user can assign names to parts of a command line thus simplifying input of often used commands. Certain parameter values can be preset in profiles. The parameter defaults are optionally updated. Other options include command files executed by the interface and recalling of command lines.

The interface module may be called several times in user written subsystems.

Menu Driven Guide

For users who want to use GOOSY without using manuals or HELP facilities a menu driven GOOSY guide provides a very easy to use interface to GOOSY. Most DCL procedures also have a menu interface.

Graphic System

The GOOSY graphics system is realized with GKS (Graphical Kernel System) which allowed to design the graphic package independently of the graphical output devices. Supported device types are among others Motif windows, postscript files and LN03-PLUS laser printers. A device independent plotfile can be generated which can be re-displayed on any graphical device. This file could be sent to IBM to be plotted on all plotters available.

To control complex experimental set-ups with many detector sub-systems, as planned for SIS/ESR experiments it is necessary to display the spectra for several of those sub-systems at the same time. Therefore the GOOSY graphic system provides a tool to summarize up to 64 frames for spectra and/or scatterplots in pictures. The size of the spectra and scatterplots in the frames and the organization of the frames on the screen can be defined by the user.

The spectra in the frames can be displayed as histograms, in vector or marker mode (for 1-dimensional spectra), and as contour, isometric, or cluster-plots (for 2-dimensional spectra) in linear, logarithmic or square-root mode on any axis. For the contour and cluster-plots arbitrary numbers of cuts are definable. The isometric plot of a two-dimensional spectrum can be rotated in any direction.

Some other features provided by the GOOSY graphics system are:

1. PLOT commands to send the actual picture on the screen or any created metafile to the plotters at VAX or IBM.
2. Update of each frame on the screen in an arbitrary time interval.
3. Projection of two-dimensional spectra.
4. Fit of background and gaussian peaks.
5. Overlays of spectra or additional scatterplot parameters in the corresponding frames.
6. Zooming of a single frame out of the actual picture on the screen.
7. Calibrations of spectra or axis labelling.

2.1.3 GOOSY Components

GOOSY components are programs executing commands. These programs can be started from the VAX/OpenVMS command language DCL or run in subprocesses. In the first case, they get the commands from the terminal, in the second case from a prompter program. Thus several components can be controlled from one terminal. The set of components controlled from one terminal is called 'environment'. Each environment has a supervisor program to dispatch messages between the prompter program and the components.

GOOSY Prompter

This program is started to control components running in an environment. It executes the following functions:

- Create/Delete environment.
- Create/Delete processes.
- Dispatch Commands.

Supervisor and components are started as subprocesses. GOOSY commands are dispatched via the supervisor to the subprocess where the command is executed. The prompter hibernates until it gets an acknowledge message from the supervisor. Therefore the subprocess executing the command can use the terminal for I/O.

Transport Manager

This is the central data dispatcher. It controls the frontend equipment, gets data buffers and dispatches them to DECnet channels, mailboxes and files (disk or tape). Presently it supports CAMAC by the MBD branch driver or by a single crate controller J11. A new CAMAC board utilizing a fully equipped 68030 processor can control other CAMAC or Fastbus crates through a differential VSB bus. A powerful VME frontend system allows to control CAMAC, Fastbus and other subsystems. If other frontend equipment will be needed in the future, this is the only interface which has to be extended. A standard program interface is provided to support other frontend equipment. Because this program runs independently of other programs, data writing is not interrupted by errors occuring in analysis or display programs.

Data Base Manager

This program executes all commands to maintain Data Bases and Data Elements, like CREATE, DELETE, SHOW, MODIFY, COPY etc.

Display

This program executes all display commands. It allocates the output device. It gets data from analysis programs via DECnet for scatter plots.

Analysis

This user specific program analyzes the data. It may get data buffers from DECnet, Mailbox or files (disk or tape). It may output data buffers to DECnet and files. During an experiment several analysis programs may run on different nodes getting the same data from one Transport Manager. Analysis programs can also be chained. The analysis program need not to be modified for online or offline mode.

Others

More programs may be started at any time. They may execute commands and control Data Elements or hardware components.

2.1.4 GOOSY Data Elements

As mentioned above, all data are located in Data Bases. There are some Data Elements defined by GOOSY which are handled by special commands. These are described in the next sections.

Condition

All kinds of conditions can be executed through analysis tables (see below). They may then be used as filters for spectrum accumulation and/or scatter plots. In an analysis routine they are executed by the macro $COND. Each condition has TRUE and FALSE counters and bits for freeze, execute, result, and preset. The different kinds are:

- Window Condition
- Multiwindow Condition
- Pattern Condition
- Function Condition
- Polygon Condition
- Composed (boolean) Condition

Polygon

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.

Spectrum

Spectra may be of the type integer or float. The dimensionality can be one or two. Spectra can be filled through analysis tables or by the macro $ACCU.

Calibration

Calibrations are tables used to calibrate the spectra data when displaying them. Each calibration can be connected to several spectra by the `CALIBRATE SPECTRUM` command (see section 2.12.3 on page 72).

Picture

Pictures keep information to display up to 64 frames containing spectra or scatterplots on one screen.

User Defined 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. To access the Data Element in a program, include the library module declaring its structure and call $LOC macro to receive the pointer to the Data Element.

2.1.5 Supported Hardware

Presently two CAMAC setups are supported by the Transport Manager:

1. Up to seven CAMAC crates each controlled by J11-based auxiliary crate controllers connected to a VAX through an MBD (Microprogrammed branch driver, BiRa Systems Inc.)
2. Single CAMAC crate system controlled by a J11 (PDP-11/73 in a CAMAC module, CES 2180 Starburst) connected to a VAX/AXP through DECnet.

The two subsystems are replaced by a VME based frontend system and a new CAMAC computer system:

1. VME frontend system. Utilizes 68020 processors, each controlling a VSB branch. The VSB branches connect to CAMAC, Fastbus or other subsystems. One processor collects the data and builds the events, another one does all the communication. The event builder transfers the data to a VAX through a parallel interface or through ethernet.
2. CAMAC computer. The board is fully equipped with a 68030 processor, ethernet, SCSI and VSB. Similar to the VME system, different kinds of subsystems may be connected to the VSB. The data is supposed to be written to a local tape.

With all setups, CAMAC commands can be executed (ESONE calls).

J11

For smaller experiments with lower data rates a CAMAC single crate system is supported by GOOSY. A J11 running RSX-11S reads out the CAMAC modules, fills compressed events into buffers which are sent via DECnet to a Transport Manager on a VAX/AXP. The CAMAC setup is described by a text file provided by the user. No programming work is required.

MBD

Each J11 crate controller reads out the CAMAC modules of one crate. The MBD copies the compressed subevents from the J11's to event buffers which are then sent to the VAX memory. Programs for the MBD are provided for standard setups. The stand alone software for the J11's can be generated from CAMAC description files and/or written and modified by the user. In the analysis program the event unpack routine may be written by the user as well.

This setup provides high flexibility and performance.

VME

The VME based frontend system is a multiprocessor system allowing very complex structures. Various kinds of subsystem can be connected through a VSB interface. The readout and processing is done in parallel. The system is fully controlled by the GOOSY transport manager.

CAMAC CVC

This is the latest development. The CAMAC board runs Lynx on a 68030 processor. Via VSB other subsystems can be connected. This system will be able to run standalone. A tape can be connected to the SCSI bus. For monitoring purposes selected data can be sent via ethernet (TCP) to GOOSY and/or PAW analysis sessions.

2.1.6 GOOSY Analysis

Event Loop

GOOSY processes experimental data structured in events. The routine executing the event loop is provided by GOOSY. It executes the following steps:

- Call unpack routine (Get new buffer if no more event is found).
- Call analysis routine (optional).
- Call analysis table executor (optional).
- Call standard routine to pack event into output buffer (Output buffer if it is filled) (optional).

The user provides the unpack routine and optionally an analysis routine.

User Written Analysis Routine

GOOSY provides PL/1 macros to locate Data Elements, accumulate spectra and check conditions. The user may write his own analysis routine using these macros. Any Data Element in a Data Base can be accessed. They are declared as PL/1 based structures. The pointers to the structures are set by the macros.

Dynamic Analysis

To improve the flexibility of the analysis GOOSY provides an analysis table executor. It executes condition checks, spectrum accumulations and scatter plots controlled by analysis tables. Commands are provided to edit the analysis tables. Each Entry in an analysis table is composed of the name of a subject Data Element and a list of names of object Data Elements, i.e. a spectrum name as subject and a member of a Data Element structure as object (optionally an additional condition name). The Entries can be added, deleted or modified without effect to a running analysis program. The modified table can then be activated in the analysis program by a command. Of course, spectra and conditions used in analysis tables can be created independently of any analysis program. The analysis table executor presently processes the following types of Entries, in the order as listed:

Function A user written function is called.

Function Condition The condition value is evaluated by a user written function.

Window Condition The values of the specified objects are checked versus the limits of the window.

Multiwindow Condition The value of the specified object is checked versus the limits of the window.

Pattern Condition The value of the specified object is checked versus the pattern of the condition.

Composed Condition This is the result of a boolean expression of other conditions.

Accumulation Supported are 1- and 2-dimensional spectra of type INTEGER or FLOAT.

Scatter Plots The Entry for scatter plots is added to the analysis table by the DISPLAY command, if the displayed picture contains any scatter plot frame.

Any of the conditions may be used as filters for either spectrum accumulation or for individual scatter plot frames. A condition can be inserted as master condition. If a master condition is false, the table execution is terminated.

2.2 Prerequisites for GOOSY

If you never have used GOOSY before, some preparations have to be done. You must be familiar with the basic VAX/OpenVMS concepts and commands (see VAX/OpenVMS Introduction manual).

2.2.1 Account

Ask the GOOSY group if your account is sufficiently priviledged to run GOOSY.

2.2.2 Information Mailer and VAX notes

Ask the GOOSY group to be included in the GOOSY information mailer list. You can list all GOOSY mails on the terminal by

```
$ GNEWS
```

All mails also are written to VAX notes. The users also can write comments to VAX notes. Thus, VAX notes should be used for communication amongst the users and the GOOSY group. A simple interface to VAX notes is provided by command

```
$ GNOTES ?    ! Enter menu
$ GNOTES WRITE "Titel" /USER/KEY=ERROR/FILE=report.txt
```

The second line inserts the content of the file into the conference GOOSY-USER. See `HELP GNOTES` for more information.

2.2.3 Creating Libraries

Create the following libraries by the commands (after you logged in):

```
$ LIB/CRE/TEXT PRIVLIB.TLB
$ LIB/CRE/HELP PRIVLIB.HLB
$ LIB/CRE/OBJ  PRIVLIB.OLB
```

2.2.4 LOGIN Procedure

In the LOGIN.COM procedure some setups for GOOSY must be done.
Copy GOO$EXE:USER_LOGIN.COM to LOGIN.COM and insert your specific statements at the marked places. (Execute the procedure after this).

```
$ SET NOVERIFY
$! Define names for libraries:
$ DEFINE/JOB PLI$LIBRARY SYS$LOGIN:privlib.TLB
$ DEFINE/JOB tpriv SYS$LOGIN:privlib.TLB
$ DEFINE/JOB LNK$LIBRARY SYS$LOGIN:privlib.OLB
$ DEFINE/JOB opriv SYS$LOGIN:privlib.OLB
$ DEFINE/JOB HLP$LIBRARY SYS$LOGIN:privlib.HLB
$ DEFINE/JOB hpriv SYS$LOGIN:privlib.HLB
$! Define names for profiles:
$ DEFINE/JOB GOO$PROFILE GOO$EXE:PROFILE.PROF
$ DEFINE/JOB GOO$INI_ALL GOO$EXE:INI_ALL.COM
$ DEFINE/JOB GOO$INI_TPO GOO$EXE:INI_TPO.COM
$! Set DCL function keys (Press PF2 for help):
$ PFKEY
$! Set system prompt to "node:user$ ":
$ @GOO$EXE:SETPROMPT.COM
$! Define all GOOSY stuff:
$ @GOO$EXE:GOOLOG.COM
$! Establish a Data Base for analysis control:
$ GOOCONTROL
$!
$! Add here user specific statements:
$!
$ IF F$MODE() .NES. "INTERACTIVE" THEN GOTO G_BATCH
$!
$! Add here statements to be executed interactively only:
$!
$!
$ G_BATCH:
$ IF F$MODE() .NES. "BATCH" THEN EXIT
$ WRITE SYS$OUTPUT "******** Starting user batch procedure ********"
$!
$! Add here statements to be executed in batch only:
$!
$ SET VERIFY
$ EXIT
```

2.2.5 Profiles

In profiles one may set default values for certain command parameters. We recommend to use the default profiles. Define in your LOGIN procedure (as shown above):

```
$ DEFINE/JOB GOO$PROFILE GOO$EXE:PROFILE.PROF
$ DEFINE/JOB GOO$INI_ALL GOO$EXE:INI_ALL.COM
$ DEFINE/JOB GOO$INI_TPO GOO$EXE:INI_TPO.COM
```

You may, however, copy these files to your Directory, modify them and define the logical names to your private profiles.

2.3 GOOSY Command Interface

2.3.1 GOOSY Guide

The easiest way to start with GOOSY is to use the guide. The guide is invoked by

```
$ GUIDE GOOSY
```

The $\boxed{\texttt{KP\_0}}$ on the keyboard calls the guide, too. The guide displays menus with brief descriptions of tasks. These tasks are executed by entering the task number as displayed. The guide is completely self describing. The actual GOOSY commands executed are displayed. So one can learn from the guide by using it. All steps except the programming of analysis routines and the prerequisites can be done in the guide. One may leave and reenter the guide at any time. The top level menu looks like this:

```
     Guide to GOOSY: path = ,  last topic =

     ---- GOOSY top menu  ----------------------------------------------

        1- Preparations needed before using GOOSY
        2- GOOSY operating. Everything to control a running GOOSY.
        3- Mostly used SHOW commands
        4- Spectrum handling
        5- Picture handling
        6- Condition handling
        7- Scatter plot handling
        8- Mostly used display commands
        9  Overview of GOOSY commands (HELP)
       10  Reset defaults to values from profile
       11- Enter menus of GOOSY components
       12  Give an overview over the current GOOSY
       13  Shutdown GOOSY

 <RET>, <CTRL>Z, B: Previous menu  <CTRL>Y, E: Exit  ?, H: Help  R: Refresh

  Enter number to select topic :
```

The topic numbers with a hyphen (-) indicate other menus, all others execute DCL procedures prompting for all necessary information.

2.3.2 Line Input

A GOOSY command line can be given in the following situations (environment and components are described later in this introduction):

1. Input to a GOOSY component prompt, e.g. `SUC: DBM>`. The component must have been started 'stand alone' on DCL level, e.g. by `$ MDBM`.

   ```
   $ MDBM
   SUC: DBM> command
   ```

2. Input to the GOOSY prompter (`SUC: GOOSY>`). A GOOSY environment and the appropriate component must have been created.

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

3. Input to DCL prompt. The GOOSY command must be preceded by the component name, e.g. `MDBM` or by `GOOSY`. The first case is equivalent to (1), the second to (2). The difference is that the command interface returns to DCL level. In addition it is possible to use DCL symbols in the command line.

   ```
   $ MDBM command      ! execute one Data Base Manager command
   $ GOOSY command     ! Excute one GOOSY command
   ```

4. In a GOOSY menu after pressing $\boxed{\texttt{KP\_PF3}}$ key.

Parameters and Qualifiers

The GOOSY command syntax is similar to the DCL syntax. A command is composed of several keywords. Positional parameters are delimited by spaces. String parameters containing delimiters must be enclosed in quotes ("string"). Integer parameters can be entered in decimal, hexadecimal (%Xn), octal (%On), or bit (%Bn) format. In addition to DCL, however, the parameters 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`). Differently to DCL they cannot specifiy a value. In most cases qualifiers can be negated by `/NOqualifier`. Qualifier sets are mutually exclusive qualifiers. In the menu the positional parameters are specified on top together with their names, followed by the qualifiers. At last, the qualifier sets follow together with their qualifiers. An example is shown below.

Command Procedures

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

```
$ GOOSY @file
$ MDBM @file /LOG
```

Comment lines may be written starting the line with an exclamation point. Lines can be continued by ending with a hyphen (-). Optional arguments replace strings &P1...&P8 in the command lines.

Defaults

In some cases default values are provided for command parameters. Some parameter values are stored as new defaults, if specified. Global defaults are valid for several commands, e.g. the Data Base name. Note, however, that the scope of global defaults is limited to a component (see 2.9.3). Only global defaults can be preset in the profiles. Some parameters are required, i.e. they are prompted if not specified. In the command description these attributes are described for each parameter:

```
parameter  default: value
parameter  replace
parameter  global replace
parameter  required
```

Command Example

The following command has two positional parameters, one switch and one qualifier set. Command description:

```
SET ACQUISITION in_buffers out_buffers
                /[NO]SYNC
                /MAILBOX/NET
```

Valid commands are then

```
SET ACQ 4 2 /SYNC/NET  ! in_buffers=4, out_buffers=2
SET ACQ out=2 /NOSYNC  ! out_buffers=2, in_buffers=default
SET ACQ /SYNC/MAIL     ! out_buffers=default, in_buffers=default
```

An **invalid** command would be

```
SET ACQ /MAIL/NET      ! illegal exclusive qualifiers
```

2.3.3 GOOSY Menu

GOOSY menus are specific to GOOSY components (see 2.9.3). 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 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 $\boxed{=}$ sign in front of the default marks a required parameter. Some more keys on the separate keypad are activated as shown in figure 2.1 on page 26

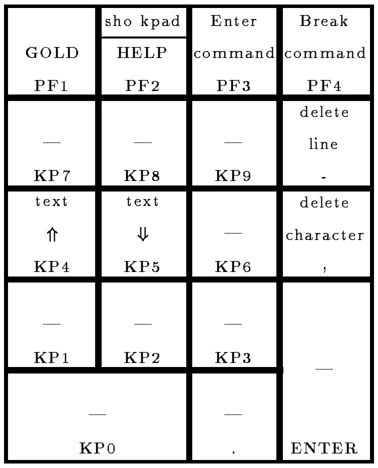
Press the $\boxed{\text{KP\_PF2}}$ key to get a keypad layout. Press the $\boxed{\text{HELP}}$ key or $\boxed{\text{KP\_PF1}}$ and $\boxed{\text{KP\_PF2}}$ to get help information for the present level. At command level the following keys are activated:

```
<HELP> or <KP_PF1><KP_PF2> ! enter HELP for present level
key<RET>                   ! enter next level
<RET> or <KP_PF4>          ! return to previous level
<KP_4>                     ! Scroll to top (courtesy IBM)
<KP_5>                     ! Scroll to bottom  (courtesy IBM)
<KP_PF3>                   ! prompt for command line, return to menu
                           ! after execution
```

At command parameter level the following keys are activated:

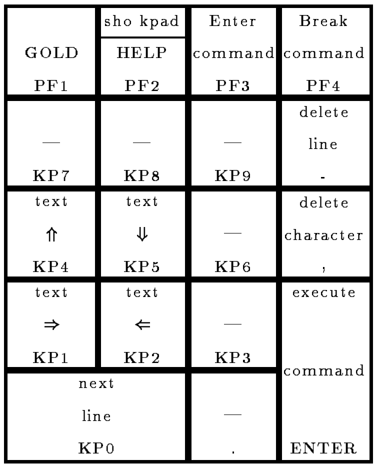
```
<HELP> or <KP_PF1><KP_PF2> ! enter HELP for present level
```

Ctrl Z: Leave GOOSY menu



GOOSY command prompter

Ctrl A: toggle insert/overstrike mode
Ctrl E: move cursor to end of line
Ctrl H: move cursor to begin of line
Ctrl Z: Leave GOOSY Prompter



GOOSY parameter prompter

Figure 2.1: The Special Keypad Layout for the GOOSY command (left) and parameter (right) menu.

```
<RET> or <KP_ENTER>        ! execute current command
<KP_PF4>                   ! return to previous level
<KP_4>                     ! Scroll to top (courtesy IBM)
<KP_5>                     ! Scroll to bottom  (courtesy IBM)
<KP_0> or arrow down       ! line down
arrow up                   ! line up
<DEL>                      ! delete character left of cursor
<KP_COMMA>                 ! delete character of cursor
<KP_1>                     ! shift text right (courtesy IBM)
<KP_5>                     ! shift text left (courtesy IBM)
<KP_MINUS>                 ! delete from cursor to end of line
<KP_PF3>                   ! prompt for command line, return to menu
                           ! after execution
<CTRL>A                    ! toggle between insert/overstrike mode
```

All defaults presently active are displayed in the menu. If the command is executed, the actual command line is displayed on top of the screen. You leave the component or prompter or menu by (several) $\boxed{\text{CTRL}}$Z.

2.3.4 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
MDISP> DELETE ALIAS name
```

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

2.3.5 User Defined Commands

Besides the commands provided by GOOSY the user may create his own commands. The command is created by routine `C$CRECM` which declares the parameter list and specifies the action routine to be called by the command. Routine `C$DSPMN` is called to prompt for command input. An example can be found in the file GOO$CMD: MCMD.PPL.

2.4 Data Management

A Data Base is located in a file and has a Data Base name. It is strongly recommended to use the same name for the file and the Data Base. The file type should be .SEC. A logical name may be created 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.

The data region of a Data Base is splitted into **Pools**. All Data Elements are stored in Pools. A Pool (and all Data Elements in the 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. Similar to a OpenVMS disk, Data Elements of a GOOSY Data Base are organized in **Directories**. The user must create Directories to use them. A diagram of the simplified Data Base structure is shown in figure 2.2 on page 30.

Data Elements can be of atomic Types (scalars or arrays), or of 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. Similar to PL/1, the variables in a structure are called members. All GOOSY Data Elements like conditions, spectra, pictures, Dynamic Lists, etc. are kept in Data Bases.

Normally one Data Base is adequate for one analysis. The Data Base and its Data Elements are created by commands. **Presently all Data Elements must be created before starting an analysis**. A DCL command procedure should be written to do this. An example of such a DCL command procedure can be copied from the file GOO$EXAMPLES:DB.COM and adapted to the needs of the experiment. Presently at least one condition and one spectrum must be created in a Data Base used for analysis.

2.4.1 Create Data Bases

A new Data Base can be created and preformatted by the DCL command:

```
$ CREDB ?
$ 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
```

The quotation mark enters a little menu. 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`. For additional information for this command use DCL `HELP CREDB`. An example is shown in section 2.14 on page 78. Data Bases are accessible on one node by all programs started by the same OpenVMS user on that node.

2.4.2 Mounting and Dismounting Data Bases

Before accessing a Data Base, it must be mounted (already done by `CREDB`). This is done by issuing one of the DCL commands:

```
$ MDBM MOUNT BASE basename filename
$ MOBASE basename filename
```

GOOSY should then tell you that the Data Base `basename` has been mounted as a global section and is opened and ready. `filename` is the Data Base file name (a VAX/AXP-OpenVMS global section file). It is strongly recommended that the Data Base name is the same as the file name and that the filename has the type .SEC. The Data Base remains mounted until the node is rebooted or the Data Base is dismounted explicitely by one of the DCL commands:

```
$ MDBM DISMOUNT BASE basename
$ DMBASE basename
```

NOTE, however, that the Data Base remains mounted as long as there are programs active using it, even if the message says that there is 'no such (global) section'.

Only processes running on the same node of a cluster are allowed to share Data Bases. If you need a Data Base already mounted on a different node, change either to that node or dismount the Data Base on the other node and mount it on your node. A list of mounted Data Bases is displayed by command

```
$ SSEC
```

One can define logical names for Data Base names. They should be defined with DEFINE/JOB to be known for all subprocesses.

For a detailed description refer to the manual 'GOOSY Data Management'.

2.4.3 Compressing and Decompressing Data Bases

If a Data Base is not used for some time, it can be compressed. The dismounted and compressed Data Base requires in most cases much less disk space. It must be decompressed before it can be mounted again.

```
$ MUTIL COMPRESS BASE basename file
$ MUTIL DECOMPRESS BASE file basename basefile
```

'Basename' and 'basefile' are the names of the Data Base and Data Base file. 'File' is the name of the file for the compressed Data Base. The default file type is .CSEC. The base must be mounted to be compressed.

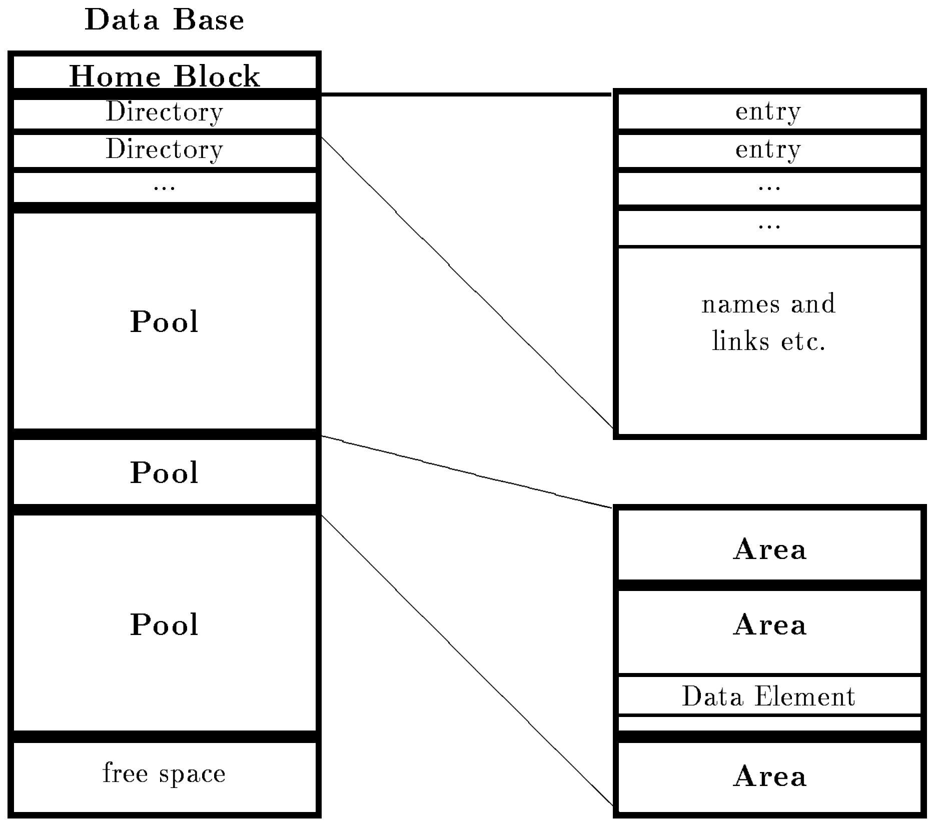


Figure 2.2: The simplified structure of a GOOSY Data Base.

2.4.4 Copying and Expanding Data Bases

A data base can be expanded only when it is copied. To copy a mounted base and to expand it to a bigger size, use

```
    $ MUTIL COPY BASE basename file SIZE=size
```

'Basename' is the name of the Data Base. 'File' is the name of the file for the new Data Base. The default file type is .SEC. The base must be mounted to be copied.

2.4.5 Save Data Bases

Data Bases are global sections. If they are in use by any program, parts of the Data Base are kept in the memory. When, due to a powerfail, the VAX crashes, the parts of the Data Base in the memory cannot be saved into the disk file. To force a dump of the Data Base from memory into the disk file one can use the command `UPDATE BASE`:

```
$ MDBM UPDATE BASE basename
```

2.5 GOOSY Data Elements

As mentioned above, all data is located in Data Bases. Data Bases are organized in Directories, similar to OpenVMS. All Data Elements have names which are inserted in a Directory. Various commands are provided to handle any Type of Data Elements, as CREATE, DELETE, SHOW, COPY, SET. In commands Data Element specifications have the general form:

```
node::base:[directory]name(index).member(index)
```

The node name is used in the future for remote Data Base access. It is presently not used. An asterisk or empty string denotes the current (local) node. All Data Elements may be structured like in PL/1. In addition - and in contrast to a file system - Data Elements may be composed of an array of Data Elements. We call such an array **name array** to distinguish it from an array inside the data. E.g. a spectrum consists of some header information and a data array, but the spectrum name may be indexed. Each member of such a name array keeps the same information as a single spectrum. The smallest referable entity of a Data Element is called a **member**.

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

2.5.1 Conditions

By default, conditions are kept in the Directory $CONDITION. GOOSY conditions are independent of spectra or coordinates (parameters), as opposed to SATAN analyzers. All kinds of conditions are executed as specified in a Dynamic List (see below). They may then be used as filters for spectrum accumulation and/or scatter plot. In an analysis routine they are executed by the macro $COND . Each condition has TRUE and FALSE counters and freeze, result, and preset bits. The following different condition types are implemented:

Window Conditions

A window condition keeps n window limits. Up to eight may be used in a Dynamic List. Each limit pair may be applied to a different object. An 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.

Multiwindow Conditions

The difference to normal window conditions is that there is one result bit for each subwindow. In a Dynamic List any number of subwindows is processed. All subwindows are applied to the same object. The result bits can be used as filters for spectrum array accumulations. 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).

Pattern Conditions

Similar to the windows, the pattern conditions may keep n subpatterns. Up to eight may be checked in a Dynamic List. 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.

Function Conditions

The user may write special routines for more complex conditions. These routines must be linked in a sharable image 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.

Polygon Conditions

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. The execution time is similar to window conditions.

Composed Conditions

This may be any boolean expression of other conditions.

2.5.2 Polygons

By default, polygons are kept in the Directory $POLYGON. 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.

2.5.3 Spectra

By default, spectra are kept in the Directory $SPECTRUM. A spectrum is composed of several Data Elements. 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 .

Spectra are created as digital 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.
- 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.

GOOSY Spectra can be extracted from a Data Base and sent to the IBM to be processed by SATAN. This is done by:

```
$ MDBM DUMP SPECTRUM name OUTPUT=file
$ SIBMSPEC file VSAMlib /RUNID=id
```

The spectra dumped into 'file' are inserted into a SATAN VSAM library.

2.5.4 Calibrations

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.

2.5.5 Pictures

By default, pictures are kept in the Directory $PICTURE. Similar to spectra pictures are sets of several Data Elements. They keep information to display several frames containing spectra or scatterplots. Up to 64 frames may be displayed on one screen.

2.5.6 User Defined 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/1 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 (should be STATIC).

2. Create a Directory in a Data Base (optional)
3. Create a Pool in a Data Base (optional)
4. Create the Data Element Type, using the new PL/1 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 a program, include the library module declaring its structure and call $LOC macro to receive the pointer to the Data Element. An example is shown in section 2.14 on page 78.

2.5.7 Buffer and Event Types

GOOSY listmode data files contain buffers. The buffer and event headers are SA$bufhe and SA$evhe, respectively. The declarations can be found in the text library GOOINC. The third and fourth word in a buffer or event header specify the type and subtype of the buffer or event, respectively. Presently the following buffer types are defined. The numbers are defined in the text library GOOINC($BUFREP). Note the difference of the event structure in a buffer or in a data base.

1. **Buffer type=2, subtype=1, event type=1, subtype=1**
   Buffers formerly used by the J11 based single CAMAC crate system. The J11's write now buffers of type 4, subtype 1 (see below). There is an unpack routine A$UPJ11 for these buffers. The structure of the event data element is SA$e1_1, which is found in the text library GOOTYP. These buffers are presently 8192 bytes long.

2. **Buffer type=3, subtype=1,2, event type=3, subtype=1,2**
   Buffers from analysis output in /COMPRESS mode. On output events are compressed by A$PACMP. The subtype specifies the compression mode. On input the events are expanded by A$UPCMP. The structure of the event data element is free. The pack routine stores the full data element into the output buffer behind an event header. On input the unpack routine copies the event without header from the buffer to the event data element.

3. **Buffer type=4, subtype=1, event type=4, subtype=1,2**
   This type is used by the J11 single crate system and by analysis output in /COPY mode. The event data element structure is SA$EVENT, which is found in text library GOOTYP. The subtype 1 stands for uncompressed events, subtype 2 for zero suppressed events. Analysis output event data elements are copied by A$PAEVT into the output buffer. These events are unpacked during input by A$UPEVT. The structure of the event data element is free, but must have a standard event header. These buffers are presently 8192 bytes long.

4. **Buffer type=6, subtype=1, event type=6, subtype=1**
   This type is used by the standard MBD system. The event data element structure is

SA$MBD, which is found in text library GOOTYP. These buffers are presently 8192 bytes long.

5. **Buffer type=10, subtype=1, event type=10, subtype=1,2,2..**
   This type is used by the standard VME system. The event data element structure is SA$VE10_1, which is found in text library GOOVME. The events are composed by subevents (Structure SA$VES10_1 in GOOVME). CAMAC and Fastbus structures are provided. These buffers are presently 16384 bytes long.

In the `START INPUT MAIL` command the appropriate buffer size must be specified! The default of 8192 matches the current default systems.

2.6 Data Base Manager

As an example for GOOSY commands and to get familiar with Data Elements, we will describe in more detail the Data Base Manager. It is invoced 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. The first 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
DEFINE       :  * :PICTURE
DELETE *     :  * :ALIAS,CONDITION,ELEMENT,LINK,OVERLAY,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:
Help: Help, PF2: Keypad, PF3: Enter command, PF4: Break, ENTER: Prev.menu
Subcommand :
```

One should work with the menu to get familiar with it. If you execute a command, the full command line will be displayed on top of the screen.

In the following we show some often used commands and their most important arguments. Examples can be found in section 2.14 on page 78. 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.

2.6.1 CREATE Commands

Create Directories

For creating Directories one should know that each name array member 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.

```
CRE DIRECTORY directory 100 base       ! 100 entries
```

Create Pools

All Data Elements are allocated in Pools. Normally the default Pools created by command `CREDB` are adequate. One may, however, create additional private Pools. The poolsize is not a limit of the Pool, because it is extended automatically. One should at least specify the size of the largest Data Element to be allocated in the Pool.

```
CRE POOL pool 8192 base                ! size 8192 bytes
```

Create Data Element Types

To create a Data Element, one must specify the structure declaration. This is done by a PL/1 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, respectively. It must be made known to the Data Base. This is done by `CREATE TYPE`:

```
CRE TYPE @library(module) base         ! Declaration from library
CRE TYPE @filename base                ! Declaration from file
```

Create Data Elements

```
CRE ELEMENT [directory]name pool type  ! Pool, Type, and dir. must exist
```

Create Polygons

Each polygon takes two entries in Directory $POLYGON. The default Pool is $PIC_POOL.

```
CRE POLYGON name points
```

The number of points of the polygon are extended automatically if needed. The polygon can be modified by the `REPLACE POLYGON` command:

```
REPLACE POLY name X=(x1,x2,x3,...) Y=(y1,y2,y3,...)
REPLACE POLY name frame /MODIFY
```

In the second command the polygon is displayed in the specified frame and can be edited by cursor input. Using the `/DELETE` qualifier instead of `/MODIFY` existing points are deleted first. The polygon can be displayed by command `DISPLAY POLYGON`.

Create Conditions

Each condition takes two entries in Directory $CONDITION, except composed conditions which take three. The default Pool is $COND_POOL

```
CRE COND WINDOW c (1,1000) 1         ! 1 subwindow
CRE COND WINDOW c (1,1000) 2         ! 2 subwindows
                                     ! both (1,1000)
CRE COND WINDOW c(10) (1,100)        ! 10 cond., 1 subw.
CRE COND WINDOW c (1,100,1,200)      ! 2 subwindows
CRE COND MULTI  c (1,1000) 100       ! 100 subwindows
CRE COND PATTERN c '1'B              ! 1 subpattern
                                     ! padded right with 0
CRE COND PATTERN c '1'B 2            ! 2 subpatterns
CRE COND PATTERN c '1'B INV='1'B /ANY ! invert first bit
CRE COND PATTERN c '11111'B /IDENT   ! identical match
CRE COND COMP c "a | (x & y)"        ! a, x, y must exist
CRE COND POLYGON c poly              ! polygon poly must exist
```

Window and pattern conditions are set by commands `SET CONDITION PATTERN` and `SET CONDITION WINDOW`, respectively. Window conditions can be set by cursor input with command `REPLACE CONDITION WINDOW`.

Create Spectra

Each spectrum takes four entries in Directory $SPECTRUM. Default Pool is $SPEC_POOL.

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

Create Dynamic Lists

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

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

Create Dynamic Entries

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 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 should be declared BASED for references of the Data Element in PL/1 programs. 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.

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

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

Create Pictures

Pictures take one entry in the $PICTURE Directory. Each frame takes one more entry. The default Pool is $PIC_POOL. First, a picture is created. Then the frames are specified.

```
CRE PICTURE pict 6 /NOPROMPT            ! 6 frames
MOD FRAME SCATTER pict 1 [data]evt.geli(1) [data]evt.naj(1)
                                        ! frame one: scatter
MOD FRAME SCATTER pict 2 [data]evt.geli(2) [data]evt.naj(2)
                                        ! frame two: scatter
MOD FRAME SPECTRUM pict 3 [$SPECTRUM]s
                                        ! frame three: spectrum
```

Create Calibrations

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 (for detail description see page 72):

```
CREATE CALIBRATION FIXED cal_1 1024   ! a calibration table with
                                      ! 1024 entries and fixed
                                      ! stepwidth between the
                                      ! uncalibrated values is
                                      ! created
CALIBRATE SPECTRUM s cal_1            ! spectrum "s" is calibrated
                                      ! with "cal_1"
```

2.6.2 SHOW Commands

The output of most SHOW commands can be stopped by $\boxed{\texttt{CTRL}}$Z.

```
SHOW CONDITION *                       ! list of all conditions
SHOW CONDITION c /FULL                 ! full information
SHOW CONDITION * /WINDOW               ! window cond. only
SHOW SPECTRUM *                        ! list of all spectra
SHOW SPECTRUM * /FULL                  ! full information
SHOW SPECTRUM s /DATA                  ! spectrum data
SHOW DYNAMIC LIST list *               ! all entries
SHOW DYNAMIC LIST list SPECTRUM        ! all spectrum entries
SHOW PICTURE *                         ! list of all pictures
SHOW PICTURE pict /FULL                ! full information
SHOW ELEMENT [data]*                   ! list of Elements in [data]
SHOW ELEMENT [data]EVT /DAT            ! contents of [data]EVT
SHOW TABLE                             ! Show all counters of
                                       ! spectra and conditions
SHOW MEMBER [data]EVT.GELI(1)          ! show contents
```

2.6.3 CLEAR Commands

```
CLEAR SPECTRUM *                       ! Clear all spectra
CLEAR SPECTRUM s                       ! Clear spectrum s
CLEAR SPECTRUM s*                      ! Clear all spectra s*
CLEAR CONDITION COUNTER *              ! Clear all test/true counters
```

2.6.4 DELETE commands

```
DELETE SPECTRUM
DELETE CONDITION
DELETE DYNAMIC ENTRY
DELETE PICTURE
DELETE POLYGON
DELETE ELEMENT
```

Data Elements which are in use by any program cannot be deleted, i.e. the analysis program protects all Data Elements it references. The `DETACH ANALYSIS` command releases this protection.

2.6.5 Miscellaneous Commands

```
COPY SPECTRUM
COPY ELEMENT
COPY MEMBER
CALCULATE SPECTRUM
MODIFY DIRECTORY
SET MEMBER
```

2.7 Data Base Access

Besides the methods described in section 2.9.2 on page 48 it is easy to write a program or routine to access data elements in a data base. First one has to attach the data base. This is done by macro `$ATTACH`:

```
$ATTACH(BASE,base,W);
```

Then each data element to be accessed must be located by macro `$LOC`. This macro returns the pointer to the data element. The structure declarations of GOOSY data elements are included by

```
@INCLUDE $MACRO($MACRO);
```

The data base can be detached by

```
$DETACH(BASE,base);
```

An example can be found on GOO$EXAMPLES:TBASE.PPL.

2.8 Utility Commands

Some commands are executed in a separate program called MUTIL.

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

The first menu level looks like:

```
Subcommands available ==================================================
$ *             :  * :ATTACH,CALL,DCL,DEBUG,DEFINE,DIRECTORY,EXECUTE,EXIT,...
COMPRESS *      :  * :BASE
COPY *          :  * :BASE,FILE
CREATE *        :  * :ALIAS,PROGRAM
DECOMPRESS *    :  * :BASE
DELETE *        :  * :ALIAS
HELP            : Access VMS HELP facility
MENU            : Enter a menu driven command dispatcher
SET *           :  * :LOCK
SHOW *          :  * :ALIAS,LOCKS
TYPE *          :  * :FILE
*************** End of list ***************** End of list *********
Command:
Help: Help, PF2: Keypad, PF3: Enter command, PF4: Break, ENTER: Prev.menu
Subcommand :
```

2.9 Starting GOOSY

2.9.1 Steps

To start GOOSY one must perform the following steps (See examples in section 2.14 on page 78):

1. Prepare the hardware and the online programs for the frontend processors. See the 'GOOSY Hardware Manual' for this purpose.
2. Create a Data Base and all Data Elements like spectra, conditions, pictures, Dynamic Lists, and Dynamic List Entries. This should be done in a DCL procedure. An example is found in the file GOO$EXAMPLES:DB.COM.
3. Prepare your analysis program. (see below).
4. Create an environment (see below).
5. Initialize the acquisition and activate Dynamic Lists, if needed.
6. Start data taking or analyzing.

These steps are described in more detail in the next sections.

2.9.2 Analysis Program

Event Loop

The event loop performs the following steps:

1. Clear execution bits.
2. Call unpack routine depending on buffer type.
3. Call user analysis routine (optional).
4. Call Dynamic List Executor (optional).
5. Fill output event into output buffer (optional).

Figure 2.3 on page 46 shows the steps performed in the event loop. First an event is copied from the input buffer to the Data Base. It may be expanded during this step. Then a user analysis routine is called (optionally) which may access the event in the Data Base and other Data Elements. It may write new values into a Data Element for output. Then the Dynamic List Executor is called. Then (optionally) the output Data Element is copied into the output buffer which is delivered to the output channels.

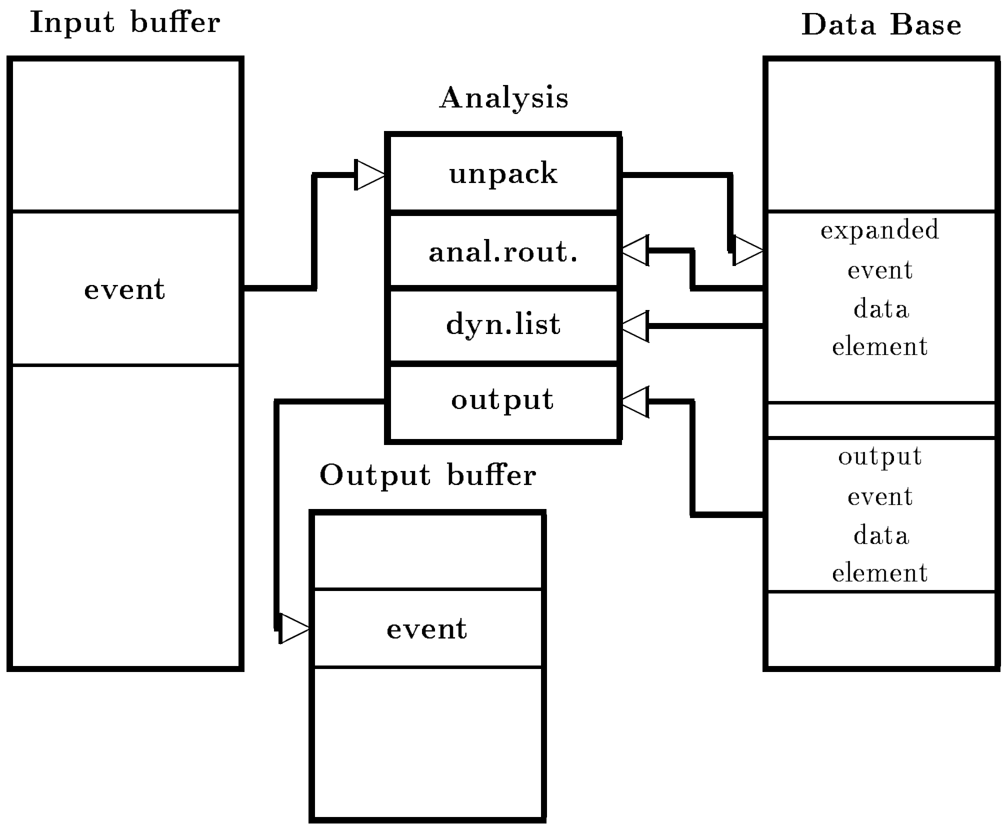


Figure 2.3: Moving events between I/O buffers and data bases in the event loop.

User Analysis Program

If you need an analysis routine, you must link your specific analysis program by

```
LANL anal           !  standard unpack
```

The object file 'anal' must contain an analysis routine called X$ANAL. The unpack routines for J11 generated data, standard MBD data, and analysis output data are provided and linked automatically. See HELP LANL for more information.

Standard Event Unpack Routines

For all standard GOOSY event types unpack routines are provided: Events written by the J11 based single CAMAC crate system are copied from buffers into a Data Element DB:[DATA]EVENT of Type SA$EVENT declared in the text library GOOTYP. If the events should be copied to another Data Element, it can be specified by the `SET EVENT INPUT` command.

Events written by the MBD controlled CAMAC system are copied from buffers into a Data Element DB:[DATA]EVENT of Type SA$MBD declared in the text library GOOTYP. If the events should be copied to another Data Element, it can be specified by the `SET EVENT INPUT` command.

Events written by an analysis are copied from buffers into a Data Element DB:[DATA]EVENT of any Type (Type must match the original event Data Element Type). If events should be copied to another Data Element, it can be specified by the `SET EVENT INPUT` command.

User Event Unpack Routine

User unpack routines can be loaded dynamically and are then called instead of the standard routines. Unpack routines get passed the pointer to a data buffer. They control by the return code what GOOSY will do after the call:

```
RETURN(1);               Process event.
RETURN(XIO_NOMOREEVENT); Provide a new buffer.
RETURN(XIO_SKIPEVENT);   Skip the event.
RETURN(XIO_NOOUTPUT);    Suppress the output of this event
                         (ignored, if output is not enabled).
```

The return codes must be declared by

```
@DCL_MSG(XIO_NOMOREEVENT);
@DCL_MSG(XIO_SKIPEVENT);
@DCL_MSG(XIO_NOOUTPUT);
```

The unpack routine copies one event from the buffer to a GOOSY Data Element which must exist in a Data Base. It must have an initialization entry. This entry is called during startup of the analysis program and must locate all Data Elements it needs. Use the $LOC macro for this purpose. One pointer and one Longword is passed. Both are zero for normal startup. The `SET EVENT INPUT` command calls the initialization entry and passes the pointer and length of the specified Data Element. The macros must be declared by

```
@INCLUDE $MACRO($MACRO);
```

A template is available from the file GOO$TEST:X$EVENT.PPL.

User Analysis Routine

Besides the dynamic analysis controlled by commands the user may write an analysis routine doing some calculations, checking conditions and accumulate spectra. This routine must be named **X$ANAL**. It is called without arguments. It must provide an entry **$XANAL** doing all necessary initializations. This entry is called during the startup of the analysis program.

Typically one calls the $LOC macros to locate conditions, spectra or Data Elements to be used in the routine. See the HELP for detailed description. In the X$ANAL routine one uses $ACCU and $COND macros to check conditions and accumulate spectra. The macros must be declared by

```
     @INCLUDE $MACRO($MACRO);
```

A template is available from the file GOO$TEST:X$ANAL.PPL. An example is described in section 2.14 on page 78 and is available from the file GOO$EXAMPLES:X$ANAL.PPL.

The execution of the routine can be disabled and enabled by command:

```
   SET ANALYSIS /NOANAL   ! disable
   SET ANALYSIS /ANAL     ! enable
```

Analysis Output

Sometimes it is useful to output list mode data from an analysis program. This output is started by the command:

```
   START ANALYSIS OUTPUT
```

Then the Data Element DB:[DATA]NEWEVENT is packed into output buffers. Two packing modes are selected with the command qualifiers `/COMPRESSED` or `/COPY`. For `/COMPRESSED` mode the Type (structure) of DB:[DATA]NEWEVENT can be chosen freely by the user. For `/COPY` mode the Data Element must have an event header like GOOINC(SA$EVHE). The Data Element must exist in the Data Base. If the another data should be copied, it can be specified by the `SET EVENT OUTPUT` command. The buffers are written to DECnet (can be read by a second analysis on a different node) and optionally to a file. Output is stopped by the command:

```
   STOP ANALYSIS OUTPUT
```

If another Data Element should be used for output, it can be specified by the command:

```
   SET EVENT OUTPUT
```

Output is done event by event. If the analysis routine returns status XIO_NOOUTPUT, no output event is written.

Macros

The macro calls for analysis routines can be inserted by the LSEDIT editor using the $\boxed{\texttt{F\_8}}$ key.

$LOC macro to locate spectra, conditions and Data Elements. For a one dimensional name array use `$LOC1`, for a two dimensional `$LOC2` macro. You must include a check on the successful execution just behind each $LOC macro. The type argument must be I,L,R or S for spectra and WC,PC,MW or POLY for conditions. Examples (W means write access):

```
$LOC(SPEC,base,$SPECTRUM,spectr,W,type);/* locate spectrum in Data Base  */
$LOC(COND,base,$CONDITION,cond,W,type); /* locate condition in Data Base */
$LOC(DE,base,directory,name,W,type);    /* locate Data Element           */
   Declares and returns a pointer P$_base_dir_name.
   Declares and returns a length  L$_base_dir_name.
$LOC1(SPEC,base,$SPECTRUM,spectrum,1,5,W,t);/* locate spectrum array      */
$LOC1(COND,base,$CONDITION,cond,1,8,W,t);/* locate condition array       */
$LOC1(DE,base,directory,name,1,2,W,t);  /* locate Data Element array     */
   Declares and returns a pointer array P1$_base_dir_name.
   Declares and returns a length array L1$_base_dir_name.

IF ^STS$success THEN @RET(STS$value);   /* Check execution success       */
```

$DE macro to reference GOOSY Data Element members in a program.

```
X=$DE(base,directory,name,member);   /* Copy value to local var. X */
X=$DE(DB,data,EVENT,SA$EVENT.IA$EVENT(I));
```

The Data Element must be located by `$LOC` macro.

$COND macro to execute condition checks. This macro executes window, multiwindow and pattern conditions. Pattern conditions can be of Type `ANY, IDENT, EXCL or INCL`. For a one dimensional name array condition use `$COND1`, for two dimensional the `$COND2` macro. Examples:

```
$COND(WC,base,$CONDITION,cond,result,1,x1);             /* one subwindow  */
$COND(WC,base,$CONDITION,cond,result,2,x1,x2);          /* two subwindows */
$COND(ANY,base,$CONDITION,cond,result,1,x1);            /* one subpattern */
$COND(MW,base,$CONDITION,cond,result,1,x);              /* multi window   */
$COND(MWI,base,$CONDITION,cond,result,1,x);             /* multi window   */
$COND(POLY,base,$CONDITION,cond,result,2,x,y,poly);     /* polygon        */
```

In the following calls `index` is the condition name index.

```
$COND1(WC,base,$CONDITION,cond,index,result,1,x1);    /* one subwindow  */
$COND1(WC,base,$CONDITION,cond,index,result,2,x1,x2); /* two subwindows */
$COND1(ANY,base,$CONDITION,cond,index,result,1,x1);   /* one subpattern */
```

$ACCU macro to accumulate one or two dimensional spectra. For a one dimensional name array spectrum use `$ACCU1`, for two dimensional the `$ACCU2` macro. Examples:

```
$ACCU(L,base,$SPECTRUM,spec,incr,1,x1);                 /* one dimension  */
$ACCU(R,base,$CONDITION,spec,incr,2,x1,x2);             /* two dimensions */
```

In the following calls `index` is the spectrum name index.

```
$ACCU1(L,base,$SPECTRUM,spec,index,incr,1,x1);       /* one dimension  */
$ACCU1(R,base,$CONDITION,spec,index,incr,2,x1,x2);   /* two dimensions */
```

$SPEC macro to fill one or two dimensional spectra. The value is copied into the spectrum channel. For a one dimensional name array spectrum use `$SPEC1`, for two dimensional the `$SPEC2` macro. Examples:

```
$SPEC(L,base,$SPECTRUM,spec,value,1,x1);               /* one dimension  */
$SPEC(R,base,$CONDITION,spec,value,2,x1,x2);           /* two dimensions */
```

In the following calls `index` is the spectrum name index.

```
$SPEC1(L,base,$SPECTRUM,spec,index,value,1,x1);        /* one dimension  */
$SPEC1(R,base,$CONDITION,spec,index,value,2,x1,x2);    /* two dimensions */
```

2.9.3 The GOOSY Environment

As shown above, the Data Base Manager and the Display Program may be executed standalone on DCL level or bundled together with other GOOSY programs in an environment. The Transport Manager and Analysis components can run ONLY in an environment. In an environment several components are started together. Figure 2.4 on page 51 shows the communication between environment components. Commands 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. The Data Bases, however, are shared between environments. Therefore the display may run in a different environment than the analysis. To create an environment with optional components, use the DCL command `CRENVIR`:

```
$ CRENVIR ?
$ CRENVIR environment myanal /ONLINE
$ CRENVIR environment myanal /OFFLINE
$ CRENVIR environment /ONLINE/DEFAULT
$ CRENVIR environment /OFFLINE/DEFAULT
```

The quotation mark enters a little menu. The environment name must be unique within a user group on one VAX node. It can be one to four characters long. The difference between `/ONLINE` and `/OFFLINE` is that with `/ONLINE` the transport manager will be started. By the qualifier `/DEFAULT` the standard GOOSY analysis program will be used. Otherwise the program specified by 'myanal' (default is MGOOANL) will be started.
The analysis is started by default with priority 3. Specify another priority by `CRENVIR ... /PRIO=p`. Similar, the DCL command

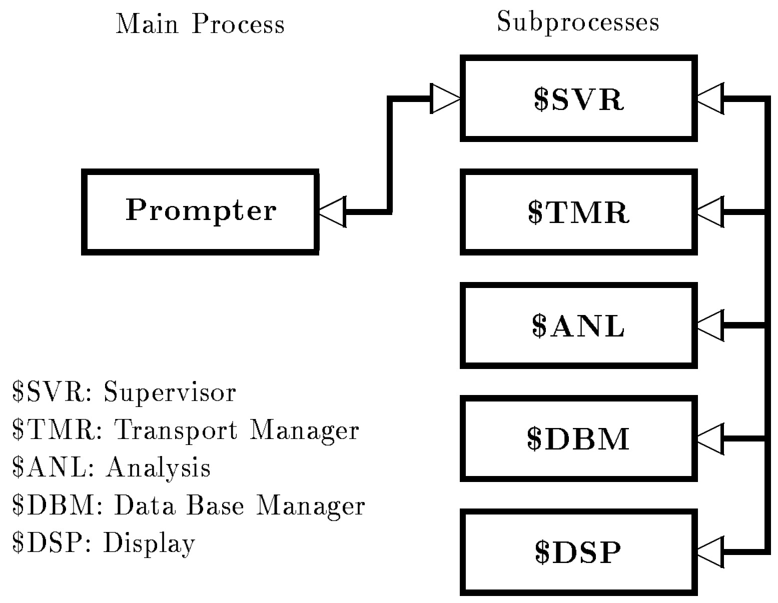


Figure 2.4: GOOSY components shown as VMS processes.

```
   $ DLENVIR
```

deletes the present environment including all components (subprocesses).

2.9.4 The GOOSY Prompter

The GOOSY prompter can only be used after an environment was created! The GOOSY prompter is entered by the DCL command `GOOSY`:

```
   $ GOOSY
it prompts with
   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 or in DCL command procedures which means also in batch jobs. Vice versa, a single DCL command can be executed by the GOOSY prompter:

```
   $ G SHOW TP KEYPAD              !Show keypad layout
```

The upper key values are the simple key hits, the lower are entered with a preceding GOLD(=PF1)-key hit.

CREATE	DELETE	SHOW
ENVIR	ENVIR	KEY
E1	E2	E3
SHOW	SHOW	
COMM	ENVIR	MENU
E4	E5	E6
	—	
—	—	—

	sho kpad	sho proc	—
GOLD	—	—	—
PF1	PF2	PF3	PF4
ANL menu	DSP menu	TMR menu	DBM menu
$ANL>	$DSP>	$TMR>	$DBM>
KP7	KP8	KP9	-
CRE ANL	CRE DSP	CRE TMR	CRE DBM
DEL ANL	DEL DSP	DEL TMR	DEL DBM
KP4	KP5	KP6	,
DISP SPE	DISP SCA	DISP PIC	—
—	—	—	
KP1	KP2	KP3	—
RECALL		CRE J11	
RECAL/ALL		—	
KP0		.	ENTER

Ctrl Z: Leave GOOSY Prompter

Figure 2.5: The Special Keypad Layout for the GOOSY prompter.

```
$ GOOSY                          !Enter GOOSY prompter
SUC:GOOSY> $ DCL "DCL-command"   !Execute DCL command
SUC:GOOSY> <Ctrl>Z               !Leave GOOSY prompter
$
```

Note, however, that "DCL-command" executes in a separate (spawned) temporary process. The GOOSY prompter menu is displayed by pressing the NEXT SCREEN key of your terminal key board.

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. Figure 2.5 on page 52 shows the keypad layout.

Now, entering the GOOSY prompter, the menus for the different components are entered by:

```
<KP_7>      ! $ANL : Analysis commands menu
<KP_8>      ! $DSP : Display commands menu
<KP_9>      ! $TMR : Transport Manager commands menu
<KP_MINUS>  ! $DBM : Data Base Manager commands menu
```

Similar, single components can be deleted and created by special keys:

```
[<PF1>]<KP_4>     ! [delete]create $ANL : Analsysis
[<PF1>]<KP_5>     ! [delete]create $DSP : Display
[<PF1>]<KP_6>     ! [delete]create $TMR : Transport manager
[<PF1>]<KP_COMMA> ! [delete]create $DBM : Data Base Manager
```

TMR: Transport Manager Program
MBD: Microprogrammed Branch driver
J11: Auxiliary crate controller
VME: VME frontend system

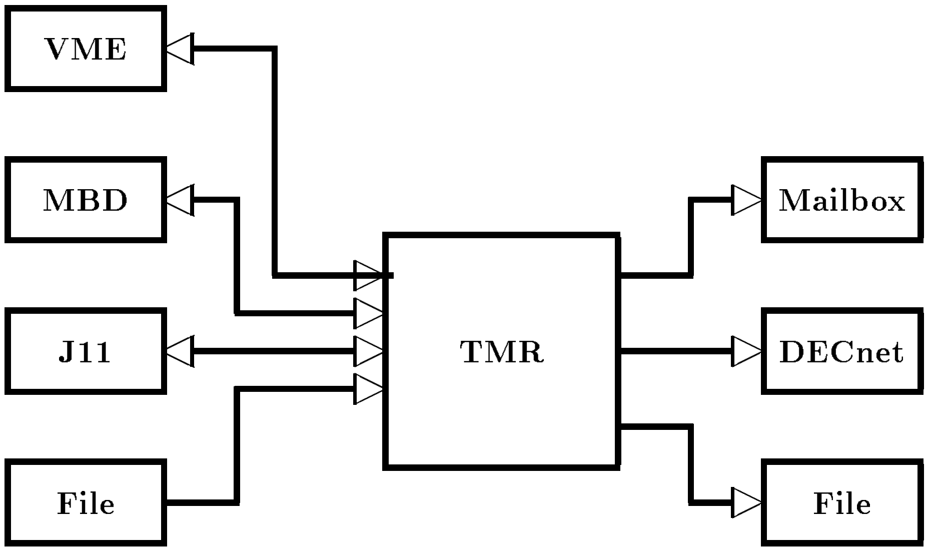


Figure 2.6: The input and output channels of the Transport Manager

Of cause, all GOOSY commands can be entered by line directly to the GOOSY prompter.

2.9.5 GOOSY Components

The following sections describe the GOOSY components running in an environment.

The Transport Manager

The Transport Manager $TMR is the interface to the frontend systems and/or disk or tape files and dispatches the data buffers. It may get input from an MBD, J11, CVC, VME system, mailbox or a file and may write buffers to disk, tape, mailbox, and/or DECnet. Actually the mailbox and DECnet are filled anytime they are read out. Writing to tape or disk is controlled by commands. There is a menu of Transport Manager commands available which can be activated by:

SUC: GOOSY> $\boxed{\texttt{KP\_9}}$

The menu is self-explanatory and contains short descriptions of the available commands.

The GOOSY PAW server

PAW clients may connect to this server. The server reads data from the Transport Manager through a mailbox, filters them and sends an event stream to the connected clients.

The Data Base Manager

The Data Base Manager $DBM executes all commands to maintain GOOSY Data Bases and Data Elements, i.e. Dynamic Lists, spectra, conditions etc.. The menu is activated by:

SUC: GOOSY> $\boxed{\texttt{KP\_MINUS}}$

The Data Base Manager may be started directly under DCL. In this case it is called by the DCL command

```
$ MDBM
```

```
it answers with
```

```
SUC: DBM>
```

Now the $\boxed{\texttt{NEXT SCREEN}}$ key will enter the menu.

The Display

The display menu is activated by:

SUC: GOOSY> $\boxed{\texttt{KP\_8}}$

You can start one more display in the same environment by the DCL command:

```
$ GOOSY CREATE PROCESS DISP name
```

where name is a 4 character name. However, if you control two displays from one terminal, you have to prefix all display commands by `name`> ($DSP is the default display):

SUC: GOOSY> `$DSP`> `DISPLAY PICTURE`
SUC: GOOSY> `name`> `DISPLAY PICTURE`

The process is started by default with priority 3. Specify another priority by `CREATE PROCESS ... PRIO=p`. The display may be started directly under DCL. In this case it is called by

AMR: Analysis Manager Program

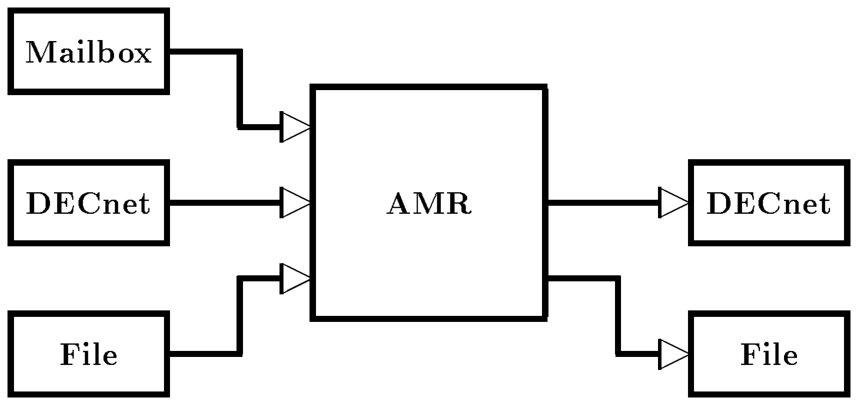


Figure 2.7: The input and output channels of the Analysis Manager (analysis program)

```
$ MDISP
```

`it answers with`

```
SUC: DISP>
```

Now the NEXT SCREEN key will enter the menu. However, no scatter plots can be displayed in this 'stand alone' mode.

The Analysis Program

The program MGOOANL is normally linked by the user with the DCL command `LANL`. It is started as $ANL component. The $ANL menu is activated by:

SUC: GOOSY> KP_7

For the correct function of the analysis routines one must make sure that the Data Base has already been mounted (so that $ANL understands the names of variables, their structures and where they reside). A default analysis program MGOOANL is provided on GOO$EXE. If GOOSY displays an error message saying that the Data Base could not be attached (or global section not found) then one has mount the Data Base and initialize the analysis by the command:

```
$ GOOSY MOUNT BASE base file
$ GOOSY INITIALIZE ANALYSIS
```

The standard analysis program for data input from a single CAMAC crate J11 system, an MDB system, or from analysis output is started by `KP_PERIOD` or the `/DEFAULT` qualifier of the `CRENVIR` command.

2.10 Dynamic Analysis

2.10.1 Activating Dynamic Lists

Dynamic Lists are Data Elements in a Data Base. Each condition check, spectrum accumulation, or scatter plot is an Entry in a Dynamic List. The creation of Dynamic Lists and Entries should be done in the DCL command procedure building the Data Base. Dynamic List Entries are executed per event and may be created and deleted dynamically (parallel to a running analysis).

Several Dynamic Lists may be executed in one analysis program. Dynamic List execution is activated by attaching to it.

```
SUC:GOOSY> ATT DYN LIST d1
```

There may be up to ten different lists attached at the same time. If it is desired to stop the execution of one list and to start the execution of another one, one would type e.g.:

```
SUC:GOOSY> DETACH DYN LIST d1
SUC:GOOSY> ATTACH DYN LIST d2
```

Already attached lists can also be stopped and started by

```
SUC:GOOSY> STOP DYN LIST d3
SUC:GOOSY> START DYN LIST d3
```

Note that the execution order is the order of attachment. This order may be changed with the `DETACH/ATTACH` commands, but not with the `STOP/START` commands. Furthermore, the `STOP/START` sequence is much faster. The following `SHOW` command gives you information about the Dynamic Lists actually executing:

```
SUC:GOOSY> SHO DYN ATT * * !Show all Dynamic Lists of all types
```

There is a "top" command to disable and enable Dynamic List execution at all:

```
SUC:GOOSY> SET ANALYSIS /NODYN   ! disable all Dynamic Lists
SUC:GOOSY> SET ANALYSIS /DYN     ! enable all Dynamic Lists
```

2.10.2 Related Commands

```
CREATE DYNAMIC LIST      listname
DELETE DYNAMIC LIST      listname
SHOW   DYNAMIC LIST      listname
CREATE DYNAMIC ENTRY type listname
DELETE DYNAMIC ENTRY type listname
ATTACH DYNAMIC LIST      listname (for Analysis program only)
DETACH DYNAMIC LIST      listname (for Analysis program only)
SHOW   DYNAMIC ATTACHED  listname (for Analysis program only)
STOP   DYNAMIC LIST      listname (for Analysis program only)
START  DYNAMIC LIST      listname (for Analysis program only)
```

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

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

`/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 condition's 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
```

2.10.3 Execution

Note that for conditions, spectra and picture frames specific freeze bits may be set or cleared by commands. This disables/enables the execution of individual Dynamic List Entries without modifications of the Dynamic List itself.

The Dynamic List is executed in the following order (the `CREATE DYNAMIC ENTRY` subcommand keys are given in parenthesis):

1. Master procedures (PROCEDURE /MASTER)
   Call specified user written procedures (modules in sharable images).
2. Master pattern conditions (PATTERN /MASTER)
   Execute pattern condition test, return if false.
3. Master window conditions (WINDOW /MASTER)
   Execute window condition test, return if false.
4. Master function conditions (FUNCTION /MASTER)
   Call specified user function, return if false.
5. Master polygon conditions (POLYGON /MASTER)
   Check polygon, return if false.

6. Master composed conditions (COMPOSED /MASTER)
   Execute composed condition test, return if false.
7. Procedures (PROCEDURE)
   Call specified user written procedures (modules in sharable images).
8. Pattern conditions (PATTERN)
   Execute pattern condition test.
9. Multi Window conditions (MULTI)
   Execute multi window condition test.
10. Window conditions (WINDOW)
    Execute window condition test.
11. Function conditions (FUNCTION)
    Call specified function (module in sharable image).
12. Polygon conditions (POLYGON)
    Check polygon.
13. Composed conditions (COMPOSED)
    Execute composed condition test.
14. Spectrum accumulation (SPECTRUM)
    Accumulates 1-2 dimensional spectra of type L,R.
15. Spectrum accumulation indexed (INDEXEDSPECTRUM)
    Accumulates 1-2 dimensional spectra of type L,R.
16. Bit spectrum accumulation (BITSPECTRUM)
    Accumulates 1 dimensional bit spectra of type L,R.
17. Scatter plots (SCATTER)
    Send buffered scatter parameter data to displays.

2.10.4 Arrays

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

```
CRE DYN ENTRY WINDOW dlist [d]e_recoil(1:5)
          PARA=[d]$event.ener
CRE DYN ENTRY SPECTRUM dlist [d]ener1(2:4)
```

```
           PARA=[d]$event.e(2:4) CONDI=[d]de_window
CRE DYN ENTRY SPECTRUM dlist [d]ede(1:4)
           PARA=([d]$event.e,$event.de)
CRE DYN ENTRY INDEXED dlist [d]ede(1:7)
           PARA=([d]$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). Multi windows 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 overlapp, in the second case this makes normally no sense.

```
CRE DYN ENTRY SPECTRUM list [d]ener1(2:4)
           PARA=[d]$event.e(2:4) CONDI=[d]m_window
! three spectrum Entries are executed
CRE DYN ENTRY INDEXEDSPECTRUM list [d]ener(2:4)
           PARA=[d]$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.

2.10.5 Entry Types

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.

2.11 GOOSY Control

An example of an ONLINE session is given in section 2.14 on page 78).

2.11.1 Commands

We give a short summary of most often used commands to control the data taking and data analysis. First some data taking control commands:

```
LOAD MBD or J11 or STARBURST          ! Load frontends
INIT ACQUIS /J11 or /MBD or /FILE   ! Select input stream
START or STOP ACQUISITION   !
START or STOP OUTPUT FILE   ! Writing to tape/disk from data acquisition
SHOW ACQUISITION            !
SHOW STARBURST              !
TYPE BUFFER                 ! Stop output by <CTRL>O
TYPE EVENT n                ! Type n events on the screen
```

Here some analysis control commands:

```
START,STOP INPUT FILE    ! Read offline from file (disk or tape)
START,STOP INPUT MAILBOX ! Read online from Transport Manager (CAMAC)
                         ! The buffer size must be specified!
START,STOP INPUT NET     ! Read online from a Transport Manager on a remote VAX
START,STOP ANALYSIS OUTPUT  ! Write to disk or tape
START,STOP SCATTER       !
START,STOP DYNAMIC LIST  !
SET SCATTER BUFFER       ! Set scatter buffer size in display points
SHOW ANALYSIS            !
SHOW DYNAMIC ATTACHED    ! Actual executing lists ! analysis
```

Now some more DBM commands:

```
SHOW DYNAMIC LIST        ! As defined in Data Base
SHOW TABLE               ! Condition and spectra counters
SHOW GOOSY STATUS
SHOW CONDITION
SHOW SPECTRUM
SHOW ELEMENT
SHOW MEMBER
[UN]FREEZE SPECTRUM      ! Inhibit the accumulation into a spectrum
[UN]FREEZE CONDITION     ! Inhibit the execution (set) of a condition
```

2.11.2 Batch and DCL Procedures

All GOOSY commands may be executed in DCL procedures mixed with DCL commands. Because any DCL procedure may run as a batch job, there are no special precautions to run GOOSY in batch. Note, that the environment name must be unique on each VAX node within a user group. There is one DCL implemented command needed to synchronize an executing analysis with the GOOSY prompter, because the analysis executes in parallel to the main session. This command is `MGOOWAIT process`.

```
$ GOOSY ATTACH DYNAMIC LIST l1  ! activate Dynamic List
$ GOOSY START INP FILE xxx      ! start analysis
$ MGOOWAIT GN_env____$anl       ! Wait
$ GOOSY SHOW SPECTRUM /TABLE    ! Show spectrum contents
```

`MGOOWAIT` hibernates until the analysis stops by end of file.

2.11.3 File Output

Writing list mode data from Transport Manager to a file must be explicitly started and stopped by commands. After opening an output file, file writing can be started and stopped without closing the file. The acquisition can be started and stopped without affecting the file status. The file can be on disk or on tape. A tape to be used must be mounted. The file is filled regardless of analysis programs.

START-STOP OUTPUT FILE

To start file output to a new file and close a file one types

```
GOOSY> START OUTPUT FILE file size number /OPEN/AUTOMATIC
GOOSY> STOP OUTPUT FILE /CLOSE
```

Then a new file is opened. 'Size' specifies the intended size of the file. When the file is filled it is automatically closed and the acquisition is stopped. All data from the frontends are transferred to the file. Then the file is closed. 'Number' is optionally used together with the `/AUTO` switch. It means that 'number' files of size 'size' are automatically opened, filled, and closed. A running number is added to the file name in this case. The `/OPEN` and `/CLOSE` qualifiers are default. To stop file writing and to start writing the same file type

```
GOOSY> STOP OUTPUT FILE /NOCLOSE
GOOSY> START OUTPUT FILE /NOOPEN
```

The STOP OUTPUT FILE commands does not stop the acquisition.

GOOSY File Header

A GOOSY file header is written to each file after it had been opened. The header information can be prompted or read from a text file (see command description of `START OUTPUT FILE`).

File Names

If one wants to send the output files to the IBM, the filenames must follow some conventions:

1. Maximal length 25 char (including type)
2. Maximal 8 char or 7 digits between two underscores (No $).
3. File type must be .LMD

Tape Handling

Writing to tape requires some additional operations. If the tape is new, it must first be initialized and then mounted. The initialization and the mount should be done within the TMR.

```
GOOSY> MOUNT TAPE tape-device tapename /INIT
```

With these commands the tape density and the size of the tape records can be specified. The defaults should be adequate. The name of the tape is used for the (optional) initialization. After that the tape file will be opened and started like a disk file. You may specify the device together with the file name or as a separate parameter. In the last case the device will be defaulted for following commands. **Use STOP ACQUISITION before STOP OUTPUT FILE and START OUTPUT FILE before START ACQUISITION to make sure that all data sent from CAMAC are written to the file!** If a file size limit is specified, the acquisition is stopped automatically early enough to write all buffers to the file.

To dismount the tape issue the GOOSY command:

```
GOOSY> STOP OUTPUT FILE /CLOSE
GOOSY> DISMOUNT TAPE device:
```

End of Tape

When the tape runs out of space, a STOP ACQUISITION is executed and the file is closed. All data from the frontends are transferred to the file! However, when the tape was not empty at the beginning, the TMR cannot know the space available on the tape. In this case it may happen, that the tape end is reached. Then OpenVMS rewinds the tape and requires a continuation tape. Mount the next tape on the device and look to the VAX operator console for the number of your tape request. Then type on your terminal in DCL:

```
$ NEXTTAPE number
```

2.11.4 Analysis Control

Independent of a Transport Manager the analysis may be started an stopped. The analysis gets data via a mailbox, a DECnet channel or from a file. It also may write output events to a file and/or DECnet.

```
START INPUT MAIL
START INPUT NET node envir component
START INPUT FILE filespec.
```

A more detailed description can be found in the 'GOOSY Data Acquisition and Analysis' manual.

2.11.5 CAMAC Esone Calls

The Transport Manager provides some commands to execute CAMAC functions and to test specific CAMAC modules. The commands are:

```
CAMAC CLEAR
CAMAC CNAF
CAMAC DEMAND
CAMAC INHIBIT
CAMAC INITIALIZE
CAMAC SCAN
TEST module
```

If using a single crate system, a logical name `CAMAC_BRANCH_0` must be defined:

```
$ DEFINE CAMAC_BRANCH_0 "J11x::GESONE()"
```

where 'x' is the node specific part of the node name of the J11. A more detailed description can be found in the 'GOOSY Hardware' manual.

2.12 GOOSY Display

2.12.1 Display Devices

Before displaying spectra or pictures it is necessary to connect a graphical output device to the display process. This could be done with the `ALLOCATE DEVICE` GOOSY display command (**not** the DCL command ALLOCATE):

```
ALLOCATE DEVICE name type /[NO]MAIN
```

Each device has an identifying name which is the physical device address. If it is your current command terminal, the '`name`' is **TT**. The '`type`' is a GOOSY alias name specifying the terminal type. The following terminal types are supported:

MG600 Monterey MG600 and MG620 terminals (default)

PECAD PECAD terminals

TEK4014 TEKTRONIX 4014 or compatibles

TEK4107 TEKTRONIX 4107 or compatibles

TEK4109 TEKTRONIX 4109 or compatibles

TEK4111 TEKTRONIX 4111 or compatibles

TEK4115 TEKTRONIX 4115 or compatibles

Motif VAX/AXP workstations or X-terminals.

MV,VS,GPX Micro VAX VAXstation display (black and white, GPX) a new window will be created with the window-name 'name'.

VT240 DEC VT240, VT241, VT330, VT340 or compatible Regis terminals.

LN03 LN03-PLUS laser-printer. A temporary plotfile `name`.LN3 is generated. The file can be printed later under DCL, e.g. by `$ PCTEK name.LN3`.

POST Postscript. A temporary plotfile `name`.PS is generated. The file can be printed later under DCL.

HP7550A4 Pen-plotter HP7550 with DIN A4 sheets. A temporary plotfile `name`.HP4 is generated. The plotfile can be copied later to a VAX-HP-plotter.

HP7550A3 Pen-plotter HP7550 with DIN A3 sheets. A temporary plotfile `name`.HP3 is generated. The plotfile can be copied later to a VAX-HP-plotter.

METAOUT Generates a device independent plotfile name.MET. The metafile can be plotted later by the DCL command `$ PLOTMET`, see also Help and GOOSY command `PLOT METAFILE`.

In GOOSY up to 10 different graphical devices could be allocated simultanously. Therefore one GOOSY main device must be specified from which all graphical **inputs** are performed. This is done with the `/MAIN` switch. As default the first allocated device is used as the GOOSY main device. All devices allocated simultanously will get the identical graphics output. You cannot use them independently. To do this, you must start a second display program on another command terminal.

- **Examples:**

```
ALLOCATE DEVICE txa2            ! terminal at physical address TXA2: is
                                ! allocated as a MONTEREY MG600
                                ! or a MG620 (default).
ALLOCATE DEVICE plotter ln03    ! A plotfile PLOTTER.LN3 is generated
                                ! which could be printed on any
                                ! LN03-PLUS laser printer.
ALLOCATE DEVICE scatter metaout ! A device independent plotfile
                                ! SCATTER.MET is created.
```

2.12.2 Displaying Pictures, Spectra and Scatterplots

As mentioned above Pictures are groups of spectra and scatter plots that are displayed together. Normally, pictures are created during the Data Base creation phase in a DCL procedure. The command syntax for displaying pictures or spectra is very similar

```
DISPLAY PICTURE  name     ! Picture "name" is displayed .
DISPLAY PICTURE  name/LOG ! All frames are displayed in log-mode
DISPLAY PICTURE           ! The last displayed picture is displayed again.
DISPLAY SPECTRUM name     ! Spectrum "name" is displayed.
                          ! "name" could be one or two dimensional.
DISPLAY SPECTRUM name/LOG ! The spectrum is displayed in log-mode.
DISPLAY SPECTRUM          ! The last displayed spectrum is displayed again.
```

A single scatterplot could be displayed with

```
DISPLAY SCATTER x y (xlow,xhigh,ylow,yhigh) cond
```

where `x` and `y` are the parameters displayed on the x and y axis within the limits `xlow` to `xhigh` and `ylow` to `yhigh`. An optional condition `cond` may be specified. The default Directory for the parameters is `data` and may be ommitted in the specification. Similar the default Directory for a condition is `$CONDITION`. The condition must be executed either in the analysis program or in a dynamic list. If the scatter plot should be executed in a dynamic list different than `$SCATTER`, this list must be specified:

```
DISPLAY SCATTER x y (xlow,xhigh,ylow,yhigh) DYN_SCAT=list
```

NOTE that the dynamic list must be attached. **Leave the display menu after activating any scatter plots!**.

For a more convenient way to execute these display commands, there are special keys defined for them:

`KP_1` for DISPLAY SPECTRUM

`KP_2` for DISPLAY SCATTER

`KP_3` for DISPLAY PICTURE

Press these keys followed by parameters as shown above.

2.12.3 Calibrating Spectra

Spectra can be displayed applying any kind of calibration to them. In GOOSY it is possible to draw a calibrated axis, by which the distance between the tics need not be equal. Or the spectrum itself can be calibrated, then the size of the bins may vary.

```
DISPLAY SPECTRUM name /CALAX       ! Draw a calibrated axis
DISPLAY PICTURE name /CALAX        ! for each calibrated spectrum
DISPLAY SPECTRUM name /CALSP       ! Calibrate the spectrum
DISPLAY PICTURE name /CALSP        ! in each frame
DISPLAY SPECTRUM name/CALAX/NOCHAN ! only the calibrated axis is shown
```

The calibrations are kept in tables, i.e. specific Data Elements in the Data Base. Three types of calibrations are supported:

LINEAR calibration with a linear polynom, the polynom parameters are kept in the Data Element.

FIXED calibration table with a fixed step width between the uncalibrated values.

FLOAT calibration table with an arbitrary stepwidth between the uncalibrated entries.

The calibration tables are created and set with the following commands:

```
CREATE CALIBRATION LINEAR name
CREATE CALIBRATION FIXED name 1024 ! FIXED-type calibration with
                                   ! 1024 entries
CREATE CALIBRATION FLOAT name 1024 ! FLOAT-type calibration with
```

```
                                        ! 1024 entries
   SET CALIBRATION LINEAR
   SET CALIBRATION FIXED name unit
   SET CALIBRATION FLOAT name unit
```

A calibration can be connected to any number of spectra with the command

```
   CALIBRATE SPECTRUM spectrum calibration
```

Before a spectrum can be deleted, the links to the calibration must be canceled by the command

```
   DECALIBRATE SPECTRUM spectrum
```

2.12.4 Temporary Display Modes

All switches specified together with display commands are **temporary display modes**. They are available only until the next DISPLAY command is given.

```
   DISPLAY SPECTRUM name/LOG/MARKER
                                ! LOG mode on scaling axis
                                ! spectrum is displayed with markers.
   DISPLAY SPECTRUM name/TEMP   ! The spectrum is displayed with the
                                ! display modes specified in the last
                                ! DISPLAY PICTURE or DISPLAY SPECTRUM
                                ! command that means in LOG and MARKER
                                ! mode
   DISPLAY PICTURE name/TEMP/HISTO
                                ! LOG mode on scaling axis
                                ! But spectra are displayed as histogramms.
```

2.12.5 Global Display Modes

Sometimes it is useful to display all spectra or pictures with the same display modes. For spectrum frames and pictures these **global modes** can be defined separatly

```
   DEFINE DISPLAY PICTURE /LOG/XLOG/SMOOTH/CLUSTER
                               ! The specified switches are stored
                               ! as global parameters for pictures
   DEFINE DISPLAY SPECTRUM /SQRT/XSQRT/VECTOR
                               ! The specified switches are stored as
                               ! as global parameters for spectra
```

These global definitions can be generally (de)activated by

```
DEFINE DISPLAY PICTURE /[NO]ACTIVE
DEFINE DISPLAY SPECTRUM /[NO]ACTIVE
```

The global settings can be activated or suspended for just one command by

```
DISPLAY PICTURE name /[NO]GLOBAL
DISPLAY SPECTRUM name /[NO]GLOBAL
```

If global modes are active, single mode parameters can be overwritten for just one command:

```
DISPLAY SPECTRUM name/LIN ! The global display modes for spectra
                          ! are activated, but the scaling axis is
                          ! displayed in LIN-mode.
```

To save CPU time there are two modes of the display. They are selected by

```
SET DISPLAY MODE /STANDARD
SET DISPLAY MODE /FAST
```

The fast version saves CPU time. If the CPU is busy, i.e. by an analysis, this version will result in a faster display. This version, however, **does not** provide the `PLOT` and `SAVE` commands. The version to be used can be changed at any time using the above commands.

2.12.6 Setting Window Conditions

Condition windows could be set with the `REPLACE CONDITION WINDOW` and `SET CONDITION WINDOW` commands. Only the `REPLACE` command provides cursor input.

```
SET CONDITION WINDOW name (low,up) dim
                      ! Condition limits "low" and "up"
                      ! of condition "name" in dimension
                      ! "dim" are set.
REPLACE CONDITION WINDOW name (low,up) dim
                      ! Condition limits "low" and "up"
                      ! of condition "name" in dimension
                      ! "dim" are set.
REPLACE CONDITION WINDOW name
                      ! Cursor appears to set limits in
                      ! condition "name" in all dimensions.
```

2.12.7 Useful DISPLAY Commands

In the following the most commonly used display commands will be listed and explained briefly.

If the displayed range of a spectrum or the range in one frame in a picture is too large it could be changed by

```
EXPAND low,up frame/CAL ! Frame "frame" is expanded within the
                        ! limits low and up. The limits are given
                        ! in calibrated units.
EXPAND                  ! The cursor appears to get limits in
                        ! one frame.
```

Spectra could be integrated by

```
INTEGRATE low,up frame ! Spectrum in frame "frame" is integrated
                 /CHAN ! within "low" and "up". The limits are given
                       ! in channels.
INTEGRATE              ! The cursor appears to get limits in
                       ! every frame on the screen.
INTEGRATE /LOOP        ! Enter cursor loop to get several integration
                       ! ranges in every frame on screen.
```

Unfortunately, it takes always some time to create the Dynamic List Entries with the `DISPLAY PICTURE` command for scatterplots defined in a picture Data Element. If the same picture should be displayed again, it is faster to refresh the screen with

```
REFRESH frame          ! A single frame is deleted on screen
                       ! and will be redrawn with the old spectrum
                       ! contents. The old scatterplot information
                       ! will be lost.
REFRESH *              ! The whole screen is cleared and all frames are
                       ! redrawn. The scatterplot informations are lost.
```

A two dimensional spectrum is projected to one dimension by

```
PROJECT SPECTRUM s2 s1 (100,200) ! Limits may be specified by
                                 ! cursor or condition
```

The organisation of the picture frames on the screen and the size of the single frames could be changed by two commands

```
DEFINE PICTURE SETUP
DEFINE FRAME SETUP
```

for details look at the command description.

2.12.8 Plotting Pictures, Spectra, and Scatterplots

The following commands work only in the standard display mode, but not in the fast mode. Use the `SET DISPLAY MODE/STANDARD` command if you are in `/FAST` mode and display the picture to be plotted. The final display on the screen could be sent to a plotter with

```
   PLOT PICTURE queue type  ! The displayed picture is plotted
                            ! on the plotter "queue" (e.g. SYS$LN03_C)
                            ! of type "type" (e.g. LN03).
```

This command plots only information kept in the local display memory. Scatterplot points are **not** kept in the memory. That means that scatterframes on the screen are empty on the plotter output.

If you like to plot scatterframes you first have to allocate a plotter (see section 2.12.1 on page 70) or a device independent plotfile, called metafile. All graphical outputs are then sent to these files. To get a scatter picture on laser printer, use the following command sequence:

```
   ALLOCATE DEVICE filename LN03  ! Open printer file
   DISPLAY SCATTER ...            ! Display scatterplot or picture
   DEALLOCATE DEVICE filename     ! Close printer file
   $ PATEX filename               ! Send file to printer LN03_A
```

Plotfiles or metafiles are generated by the `ALLOCATE DEVICE` command. A device dependent plotfile or a device independent metafile can be plotted with

```
   PLOT PLOTFILE file queue
   PLOT METAFILE file queue type
```

where the queue depends on the output device, e.g. 'SYS$LN03_C' or for `PLOT METAFILE` also 'IBM::'. The type is the device name, e.g. LN03 or for IBM the RP01 or RP02:

```
  PLOT METAFILE xxx.MET SYS$LN03_C LN03  !xxx will be plotted on laser printer C
  PLOT METAFILE xxx.MET IBM:: RP02       !xxx will be plotted on IBM rasterplotter
```

Metafiles can also be plotted from DCL by command `PLOTMET`.

2.13 Error Recovery

GOOSY modules use the VAX-OpenVMS error handling utilities. GOOSY error messages are formatted and output as OpenVMS error messages, i.e.:

```
%facility-severity-code, message text
%GOODM-E-ALREX, M$CRESP: spectrum already exist in data base.
```

The 'severity' is a letter indicating the error level of the message. It can be **S**uccess, **I**nformation, **W**arning, **E**rror, or **F**atal. In many cases one may get more information about a GOOSY error by the DCL command:

```
$ HELP @REC facility code
$ HELP @REC GOODM ALREX
```

'facility' and 'code' can be obtained from the message output. There is also a manual containing all recovery information. GOOSY outputs an additional headline for each message containing the module name, user name, node name, date etc.. To make the reading of the error messages more convenient, one may suppress the output of this line and/or the output of the message prefix (%facility-severity-code). This is done by:

```
$ SETMESSAGE GOOSY /NOPREF/NOHEAD
$ SETMESSAGE GOOSY
```

The second line enables full message output again.

2.14 Session Examples

In the following we show a few examples of ONLINE or OFFLINE GOOSY sessions. The examples can be taken literally. Some parameters, however, have to be replaced by actual values because they depend on the CAMAC setup, the DECnet node where the J11, runs and the display device. The environment name must be specified, too. All these names to be adapted are marked by <>. The example files are on Directory GOO$EXAMPLES: and can be copied from there.

2.14.1 Preparations for the Examples

The following three steps are common to all examples. We assume that the LOGIN.COM procedure is already prepared for GOOSY (see section 2.2.4 on page 19).

Creating Data Elements

The standard event Data Element of J11 generated data is provided by GOOSY (library GOOTYP) and has the structure:

```
    /* ================= GSI Event header ======================= */
    DCL P_SA$event       POINTER;
    DCL 1 SA$event         BASED(P_SA$event),
2 IA$event_dlen    BIN FIXED(15),
2 IA$event_tlen    BIN FIXED(15),
2 IA$event_type    BIN FIXED(15),
2 IA$event_subtype BIN FIXED(15),
2 IA$event(512)    BIN FIXED(15);
    /*-------------------------------------------------------------*/
```

The ADC data are stored by the GOOSY unpack routine in IA$EVENT. The first ADC is in IA$EVENT(1:8), the second in IA$EVENT(9:16). A third SILENA would be in IA$EVENT(17:24) and so on.

We want to use a Data Element for calibrated data (See section 2.5 on page 32). The declaration is:

```
/* file s_out.txt:        */
DCL P_out POINTER;
DCL 1 S_out BASED(P_out),
    2 R_energy BIN FLOAT(24),
    2 R_time   BIN FLOAT(24);
```

Note that the name of the file is the name of the structure. We put it into our private text library TPRIV (defined in LOGIN.COM). From there it can be included in PL/1 procedures. It also can be used to create a GOOSY Data Element in the Data Base.

```
   $ LIB/REPL TPRIV S_out.txt    ! store module in a text library
```

Creating the Data Base

Now we have to create a Data Base. We write a command procedure to do that (See sections 2.4 on page 28, 2.5 on page 32, 2.6 on page 37, and 2.10 on page 57).

```
$! file credb_j11.com
$!
$! Data Element: event (Type SA$event)
$!               newevent (Type S_out)
$! Spectra:   single(1:16)
$!            energy
$!            time
$!            window
$!            pattern
$! Condition: peak
$! Dyn.list:  accu
$!
$ ON ERROR THEN CONTINUE
$ CREDB db db_j11.sec 2000 /NEW
$! The Data Base name 'db' is specifed once and
$! defaulted in the following commands.
$ MDBM
CREATE TYPE      "@GOOTYP(SA$event)"
CREATE TYPE      "@TPRIV(S_out)"
CREATE ELEMENT event    TYPE=SA$event DIR=data POOL=data
CREATE ELEMENT newevent TYPE=S_out    DIR=data POOL=data
CREATE SPECTRUM single(16) L (0,2047) /DIGITAL
CREATE SPECTRUM window     L (0,2047) /DIGITAL
CREATE SPECTRUM pattern    L (1,32)   /DIGITAL
CREATE SPECTRUM energy     R (0,2047) /ANALOG
CREATE SPECTRUM time       R (0,2047) /ANALOG
CREATE CONDITION WINDOW peak (1150,1200)
CREATE DYNAMIC LIST accu ENTRIES=20
CREATE DYNAMIC ENTRY WINDOW accu peak PARA=event.IA$event(4)
CREATE DYNAMIC ENTRY SPECTRUM accu single(1:16) PARA=event.IA$event(1:16)
CREATE DYNAMIC ENTRY SPECTRUM accu window PARA=event.IA$event(7) CONDI=peak
CREATE PICTURE raw 8 /NOPROMPT
MODIFY FRAME SPEC raw 1 single(1)
MODIFY FRAME SPEC raw 2 single(2)
MODIFY FRAME SPEC raw 3 single(3)
MODIFY FRAME SPEC raw 4 single(4)
MODIFY FRAME SPEC raw 5 single(5)
MODIFY FRAME SPEC raw 6 single(6)
MODIFY FRAME SPEC raw 7 single(7)
MODIFY FRAME SPEC raw 8 single(8)
DISMOUNT BASE db
$ EXIT
```

Writing the Analysis Routine

Now we write an analysis routine (See section 2.9 on page 45). This routine does the same as the Dynamic List. In addition it fills spectra 'energy' and 'time'. If we were satisfied with the Dynamic List we would not need it.

```
/* file X$ANAL.PPL                          */
/*******************************************/
X$ANAL:@PROCEDURE RETURNS(BIN FIXED(31));
/*******************************************/
@INCLUDE $MACRO(dcl_proc);
@INCLUDE $MACRO(S$mess);
@INCLUDE $MACRO($MACRO);
@INCLUDE $MACRO(SA$event);
@INCLUDE $MACRO(S_out);

DCL P_event    POINTER STATIC;
DCL P_newevent POINTER STATIC;
DCL B_peak     BIT(1) ALIGNED;
DCL L_index    BIN FIXED(31);
DCL L_incr     BIN FIXED(31) STATIC INIT(1);

@ON_ANY_W(U_CLEANUP);           /* Error routine */

P_SA$event = P_event;           /* Get pointer to input event */
P_out      = P_newevent;        /* Get pointer to output event */

STS$value=1;
/* The data is in IA$event(1:16) */
$COND(WC,DB,$CONDITION,peak,B_peak,1,IA$event(4));
DO L_index = 1 TO 16;
    $ACCU1(L,DB,$SPECTRUM,single,L_index,L_incr,1,IA$event(L_index));
    END;
IF B_peak THEN DO;
    S_out.R_energy=(FLOAT(IA$event(4),24) + FLOAT(IA$event(7),24))/4.;
    S_out.R_time=(FLOAT(IA$event(4),24) + FLOAT(IA$event(7),24))/2.;
    $ACCU(L,DB,$SPECTRUM,window,L_incr,1,IA$event(7));
    $ACCU(R,DB,$SPECTRUM,energy,L_incr,1,S_out.R_energy);
    $ACCU(R,DB,$SPECTRUM,time,L_incr,1,S_out.R_time);
    END;

@RET(STS$value);
/****************************************************/
/* Initialization to locate all used Data Elements  */
/****************************************************/
$XANAL:ENTRY RETURNS(BIN FIXED(31));
@INCLUDE $MACRO($SECDEF);

$LOC(DE,db,data,event,W);
IF ^STS$success THEN @RET(STS$value);
P_event=P$_DB_DATA_EVENT;

$LOC(DE,db,data,newevent,W);
IF ^STS$success THEN @RET(STS$value);
P_newevent=P$_DB_DATA_NEWEVENT;

$LOC1(SPEC,DB,$SPECTRUM,single,1,16,W,L);
IF ^STS$success THEN @RET(STS$value);
$LOC(SPEC,DB,$SPECTRUM,energy,W,R);
IF ^STS$success THEN @RET(STS$value);
$LOC(SPEC,DB,$SPECTRUM,time,W,R);
IF ^STS$success THEN @RET(STS$value);
$LOC(SPEC,DB,$SPECTRUM,window,W,L);
IF ^STS$success THEN @RET(STS$value);
$LOC(COND,DB,$CONDITION,peak,W,WC);
IF ^STS$success THEN @RET(STS$value);

@RET(STS$value);
/*************************************************/
/* This routine is called in case of an error    */
/*************************************************/
U_CLEANUP:PROCEDURE;
END U_CLEANUP;
END X$ANAL;
```

We compile the analysis routine and link our analysis program by the DCL commands:

```
$ CHANAL X$ANAL.PPL  !check macro calls
$ COMP X$ANAL        !compile
$ LANL X$ANAL        !make MG00ANL.EXE
```

You may now continue one of the following examples.

2.14.2 OFFLINE Analysis

Instead of getting the data from the TMR, the Analysis Manager may read the data from a file. The Data Base and analysis routine X$ANAL need not to be modified. This example can be executed literally except the environment name which should be different. The startup looks like

Procedures Starting GOOSY

We write a small command procedure to startup GOOSY.

```
$! file anl_startup.com
$ IF p1 .EQS. "" THEN INQUIRE p1 "Enter environment name "
$ MDBM MOUNT BASE db db_j11.sec
$ CRENV 'p1' /$DSP/$DBM/$ANL  ! No TMR
$ GOOSY CLEAR SPECTRUM *
$ EXIT
```

Similar command procedure to shutdown GOOSY:

```
$! file anl_shutdown.com
$ GOOSY
  STOP INPUT FILE/CLOSE
$ DLENV      ! delete environment
$ MDBM DISMOUNT BASE db
$ EXIT
```

Procedures Starting the Analysis

We write a small command procedure to start the analysis:

```
$! file anl_start.com
$ GOOSY
ATT DYN LIST accu           ! activate the Dynamic List 'accu'
SET ANAL/NOANAL             ! disable X$ANAL in the event loop
START INPUT FILE DAY$ROOT:[GOOFY]J11.LMD/OPEN ! start reading file
$ EXIT
```

Similar command procedure to stop the analysis:

```
$! file anl_stop.com
$ GOOSY
STOP INPUT FILE
DET DYN LIST accu
$ EXIT
```

Command Sequences

Now a typical session could proceed as follows:

```
$ @credb_j11         ! create Data Base
$ @anl_startup <env> ! create GOOSY environment
$ @anl_start         ! start GOOSY (dyn.list only)
$ GOOSY              ! enter GOOSY prompter
GOOSY> SHOW ANAL
GOOSY> ALL DEV <dev> PECAD/MAIN   ! <dev> is the display terminal name
GOOSY> DISP PI raw
GOOSY> DISP SP energy
GOOSY> DISP SP time
GOOSY> DISP SP single(1)
GOOSY> REPLACE COND WIND peak ! set condition by cursor
GOOSY> STOP INPUT FILE/CLOSE ! If we don't want to continue
GOOSY> CLEAR SPECTRUM *      ! clear all spectra
GOOSY> DETACH DYN LIST accu  ! disable dyn.list
GOOSY> SET ANAL/ANAL         ! enable X$ANAL
GOOSY> START INP FIL DAY$ROOT:[GOOFY]J11.LMD/OPEN ! open again if we closed it
GOOSY> <CTRL>Z               ! leave GOOSY prompter
$ @anl_stop
$ @anl_shutdown
```

2.14.3 OFFLINE Analysis with Transport Manager

Instead of getting the data from the J11, the Transport Manager may read the data from a file. The Data Base and the analysis routine X$ANAL need not to be modified. To analyze data one does not need necessarily a TMR because the Analysis Manager can read the data from a file directly. The present example can be useful to get familiar with the TMR. It can be executed literally except the environment name which should be different.

Procedures Starting GOOSY

We write a small command procedure to startup GOOSY.

```
$! file off_tmr_startup.com
$ IF p1 .EQS. "" THEN INQUIRE p1 "Enter environment name "
$ MDBM MOUNT BASE db db_j11.sec
$ CRENV 'p1' /$TMR/$DSP/$DBM/$ANL
$ GOOSY INI ACQUIS /FILE       ! read data from file instead of from J11
$ GOOSY CLEAR SPECTRUM *       ! clear all spectra
$ EXIT
```

Similar command procedure to shutdown GOOSY:

```
$! file off_tmr_shutdown.com
$ DLENV      ! delete environment
$ MDBM DISMOUNT BASE db
$ EXIT
```

Procedures Starting the Analysis

We write a small command procedure to start the acquisition:

```
$! file off_tmr_start.com
$ GOOSY
START INPUT MAIL             ! Start reading data from mailbox
ATT DYN LIST accu            ! activate the Dynamic List 'accu'
SET ANAL/NOANAL              ! disable X$ANAL in the event loop
SET ACQUIS/SYNC/MAIL         ! Synchronize input with analysis
OPEN FILE DAY$ROOT:[GOOFY]J11.LMD ! open file for TMR input
START ACQUIS                 ! Start reading file
$ EXIT
```

Similar command procedure to stop the acquisition:

```
$! file off_tmr_stop.com
$ GOOSY
STOP ACQUIS
CLOSE FILE
STOP INPUT MAIL
DET DYN LIST accu
$ EXIT
```

Command Sequences

Now a typical session could proceed as follows:

```
$ @credb_j11                ! create Data Base
$ @off_tmr_startup <env>    ! create GOOSY environment
$ @off_tmr_start            ! start GOOSY (dyn.list only)
$ GOOSY                     ! enter GOOSY prompter
GOOSY> SHOW ACQUIS
GOOSY> SHOW ANAL
GOOSY> TYPE EVENT 5/SAMPLE
GOOSY> ALL DEV <dev> PECAD/MAIN   ! <dev> is the display terminal name
GOOSY> DISP PI raw
GOOSY> DISP SP energy
GOOSY> DISP SP time
GOOSY> DISP SP single(1)
GOOSY> REPLACE COND WIND peak ! set condition by cursor
GOOSY> STOP ACQUIS
GOOSY> CLOSE FILE           ! if we don't want to continue
GOOSY> CLEAR SPECTRUM *     ! clear all spectra
GOOSY> DETACH DYN LIST accu ! disable dyn.list
GOOSY> SET ANAL/ANAL        ! enable X$ANAL
GOOSY> OPEN FILE DAY$ROOT:[GOOFY]J11.LMD ! open again if we closed it
GOOSY> START ACQUIS
GOOSY> <CTRL>Z              ! leave GOOSY prompter
$ @off_tmr_stop
$ @off_tmr_shutdown
```

2.14.4 ONLINE with a Single CAMAC Crate

We show an example of a trivial experiment using a single CAMAC crate with two SILENA ADC's. The CAMAC is controlled by a J11 auxiliary crate controller. The hardware setup for such a system is described in detail in the GOOSY Hardware Manual.

CAMAC Setup File

First we have to write a CAMAC setup file to describe the modules in the crate. This file must be adapted to the actual CAMAC setup. The example describes two SILENA ADC's in station N8 and N9, respectively.

```
! file j11c.cam, the crate number is not yet used
C=0, N=8, A=0, TY=STANDARD
C=0, N=8, A=1, TY=STANDARD
C=0, N=8, A=2, TY=STANDARD
C=0, N=8, A=3, TY=STANDARD
C=0, N=8, A=4, TY=STANDARD
C=0, N=8, A=5, TY=STANDARD
C=0, N=8, A=6, TY=STANDARD
C=0, N=8, A=7, TY=STANDARD
C=0, N=9, A=0, TY=STANDARD
C=0, N=9, A=1, TY=STANDARD
C=0, N=9, A=2, TY=STANDARD
C=0, N=9, A=3, TY=STANDARD
C=0, N=9, A=4, TY=STANDARD
C=0, N=9, A=5, TY=STANDARD
C=0, N=9, A=6, TY=STANDARD
```

```
C=0, N=9, A=7, TY=STANDARD
```

Procedures Starting GOOSY

We write a small command procedure to startup GOOSY. The J11 is on a specific DECnet node, e.g. J11C:

```
$! file onl_tmr_startup.com
$ IF p1 .EQS. "" THEN INQUIRE p1 "Enter environment name "
$ IF p2 .EQS. "" THEN INQUIRE p2 "Enter node of J11      "
$ MDBM MOUNT BASE db db_j11.sec
$ DEFINE CAMAC_BRANCH_0 "''p2'::GESONE()" ! for ESONE commands
$ CRENV 'p1' /$TMR/$DSP/$DBM/$ANL
$ GOOSY INI ACQUIS NODE='p2' /J11
$ GOOSY LOA J11 j11c.cam          ! load the readout NAF list
$ GOOSY CLEAR SPECTRUM *          ! clear all spectra
$ EXIT
```

Similar command procedure to shutdown GOOSY:

```
$! file onl_tmr_shutdown.com
$ GOOSY STOP ACQUIS/ABO    ! break link to J11
$ DLENV                    ! delete environment
$ MDBM DISMOUNT BASE db
$ DEAS CAMAC_BRANCH_0
$ EXIT
```

Procedures Starting the Acquisition and Analysis

We write a small command procedure to start the acquisition:

```
$! file onl_tmr_start.com
$ GOOSY
START INPUT MAIL SIZE=8192  ! Size for J11 data 8K, MBD 4K
ATT DYN LIST accu           ! activate the Dynamic List 'accu'
SET ANAL/NOANAL             ! disable X$ANAL in the event loop
SET ACQUIS/SYNC/MAIL        ! Synchronize input with analysis
START ACQUIS
$ EXIT
```

Similar command procedure to stop the acquisition:

```
$! file onl_tmr_stop.com
$ GOOSY
STOP ACQUIS
STOP INPUT MAIL
DET DYN LIST accu
$ EXIT
```

Command Sequences

Now a typical session could proceed as follows:

```
$ @credb_j11                       ! create Data Base
$ @onl_tmr_startup <env> <node>    ! create GOOSY environment
$ @onl_tmr_start                   ! start GOOSY (dyn.list only)
$ GOOSY                            ! enter GOOSY prompter
GOOSY> SHOW ACQUIS
GOOSY> SHOW ANAL
GOOSY> TYPE EVENT 5             ! look if events are OK
GOOSY> ALL DEV <dev> PECAD/MAIN ! <dev> is the display terminal name
GOOSY> DISP PI raw
GOOSY> DISP SP energy
GOOSY> DISP SP time
GOOSY> DISP SP single(1)
GOOSY> REPLACE COND WIND peak   ! set condition by cursor
GOOSY> STOP ACQUIS
GOOSY> CLEAR SPECTRUM *         ! clear all spectra
GOOSY> DETACH DYN LIST accu     ! disable the Dynamic List 'accu'
GOOSY> SET ANAL/ANAL            ! enable X$ANAL
GOOSY> START ACQUIS
GOOSY> <CTRL>Z                  ! leave GOOSY prompter
$ @onl_tmr_stop
$ @onl_tmr_shutdown
```

2.14.5 Random Generator

The transport manager provides a random event generator. This generator is activated by the `/FOREIGN` qualifier:

```
GOOSY> INIT ACQUIS /FOR
```

There are 4 parameter events of type 4,1 generated.

2.14.6 Command Examples

The following examples cannot be taken literally but show the usage of some important commands. For more lucidity the parameters which have to be replaced by actual values are enclosed in brackets <>. We use the standard analysis program.

The first example is for a **J11** based single CAMAC create system.

```
$ CRENV <env> /$TMR/J11/$DBM         ! Create environment with standard
                                     ! analysis
$ GOOSY                              ! Enter GOOSY
GOOSY> INIT ACQUIS /J11 NODE=<node>  ! Initialize acquisition for J11
GOOSY> LOAD J11 j11c.cam             ! Load CAMAC table to J11
GOOSY> START ACQUISITION             ! Start data taking
```

The next example controls an **MBD** system using a private analysis program.

```
$ CRENV <env> /$TMR/$ANL/$DBM        ! Create environment with MGOOANL
$ GOOSY                              ! Enter GOOSY
GOOSY> INIT ACQUIS /MBD              ! Initialize acquisition for MBD
GOOSY> LOAD MBD GOO$IO:EXEC.BDO/EXEC ! Load exec code into the MBD
GOOSY> LOAD MBD GOO$IO:ESONE.BDO     ! Load ESONE code into the MBD
GOOSY> LOAD MBD <mycode>.BDO         ! Load user code into the MBD
GOOSY> LOAD STAR <s1>.EXE 1 23 /BOOT ! Load and boot J11 of crate one
GOOSY> LOAD STAR <s2>.EXE 2 23 /BOOT ! Load and boot J11 of crate two
GOOSY> START ACQUISITION             ! Start data taking
```

From here use the same commands for both examples:

```
GOOSY> SHOW ACQUIS                   ! Look what happens
GOOSY> STOP ACQUIS                   ! Stop data taking
GOOSY> START OUTPUT FILE <file.lmd>  ! Open file for output
GOOSY> START ACQUIS
GOOSY> STOP ACQUIS
GOOSY> STOP OUTPUT FILE /CLOSE       ! Close file

GOOSY> ATTACH DYNAMIC LIST accu      ! Activate the Dynamic List 'accu'
GOOSY> START INPUT MAILBOX SIZE=8192 ! Open input channel
                                      (8192 for J11, 4096 for MBD)
GOOSY> START ACQUIS
GOOSY> SHOW ANALYSIS                 ! View analysis
GOOSY> STOP ACQUIS                   ! Stop data taking
GOOSY> STOP INPUT MAILBOX            ! Stop analysis
GOOSY> DETACH DYNAMIC LIST accu      ! No longer execute the Dynamic List 'accu'
GOOSY> DETACH ANALYSIS               ! Free Data Bases for modifications

GOOSY> ATTACH ANALYSIS               ! Reinitialize analysis
GOOSY> ATTACH DYNAMIC LIST accu      ! Activate Dynamic List 'accu'
GOOSY> START INPUT MAILBOX SIZE=8192 ! Start analysis input
                                      (8192 for J11, 4096 for MBD)
GOOSY> START ACQUIS                  ! Start again
GOOSY> STOP ACQUIS
GOOSY> STOP INPUT MAILBOX
GOOSY> <CTRL>Z                       ! Leave GOOSY
$ DLENV                              ! Delete environment
```

Appendix A

GOOSY Command Summary

$ CLOSE ETHERNET /SHOW
Close Ethernet link.

$ COMMENT LINE1="first line of comments"
LINE2="second line of comments"
LINE3=l3 LINE4=l4 LINE5=l5
LINE6=l6 LINE7=l7 LINE8=l8
/TERM /ERRL /SLOG /GLOG /PLOG
Write comments to SYS$OUTPUT and log files.

$ DCL "DCL command line"
Execute a single DCL command line.

$ DEBUG –
Enter DEBUGger.

$ DEFINE KEY KEY=k EQUIVALENCE=e
IF_STATE=i SET_STATE=s
/NOECHO/LOCK_STATE/TERMINATE/PROTECT
Associates an equivalence string and a set of attributes with a key on the terminal keyboard.

$ RECALL KEY1=k KEY2=ke KEY3=ke KEY4=ke /ALL
Recall, list or file previous commands.

$ REPEAT COUNT=c LINE1=l1 LINE2=l2 ... LINE8=l8
WAIT=w /LOG/NOLOG
/ONWARNING/ONERROR/ONFATAL
Repeat a list of commands.

$ RESET DEFAULT profile
Reset default parameters from profile (=same values than after startup).

$ SET DEFAULT profile program ....
Set default parameters.

$ SET GNA ETHERNET Link Read Write Acknow
Wretry Lretry output
/[NO]SWAP
/[NO]ISTREAM
/[NO]OSTREAM
/[NO]DEBUG
SET Ethernet modes.

$ SHOW COMMAND token1 token2 token3 token4 output
/BRIEF/PARAM/FULL/NAME
Show command definitions

$ SHOW GNA COMPONENTS LINK=link NAME=name
Show status of GNA components.

$ SHOW GNA ETHERNET /FULL /CLEAR
Show Ethernet information.

$ SHOW GNA LINKS LINK=link NAME=name
Show status of GNA links.

$ SHOW GNA MCBS LINK=link
/PAQ/LCB/ALL /BRIEF/FULL
Show status of MCB's.

$ SHOW GNA PROCESS LINK=link
Show process status of GNA object.

$ SHOW GNA RPC LINK=link NAME=name
Show status of GNA RPC handlers.

$ SHOW GNA STATUS LINK=link
Show global GNA status.

$ SHOW KEY KEY=k STATE=k
/FULL/BRIEF /DIRECTORY
List key definitions.

$ SHOW MEMORY –
Show memory usage.

$ SHOW TIMER –
Show statistics of run time library functions.

ALLOCATE DEVICE name type xsize ysize
/[NO]MAIN
Allocate a graphical device

ATTACH ANALYSIS –
Reinitialize analysis after DETACH ANALYSIS

ATTACH BASE base node
/READ
Attach data base.

ATTACH DYNAMIC LIST dyn_list dyn_dir base node
/FAST
Attach dynamic list

CALCULATE FASTBUS PEDESTAL loop throff thrfact
pedoff pedfact sample trigger
VMEcrate,processor ID dummy crate node
/ON/OFF [=ONOFF]
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Set fastbus pedestal subtraction on/off.

CALCULATE SPECTRUM –
result operand_1 operand operand_2 factor
spec_dir base node
/CONSTANT
/[NO]KEEP
Perform spectrum arithmetic operations.

CALIBRATE SPECTRUM spectrum calibration spec_dir cal_dir base node
Connect calibration to a spectrum.

CAMAC CLEAR C=c
Generate Dataway clear

CAMAC CNAF C=c N=n A=a F=f DATA=d Branch=b
Perform a single CAMAC action

CAMAC DEMAND C=c
/ENABLE/DISABLE/TEST
Enable, disable or test Crate Demand

CAMAC INHIBIT C=c
/SET/CLEAR/TEST
Set, clear or test Dataway inhibit

CAMAC INITIALIZE C=c
Generate Dataway Initialize

CAMAC SCAN C=c N=n F=f /CRATE/STATION/ADDRESS
Perform a crate scan

CLEAR CAMAC SPECTRUM name spec_dir base node
/CAMAC
/SPECTRUM
/LOG
/[NO]KEEP_MAP
clear one (or all) spectrum

CLEAR CONDITION COUNTER – name cond_dir base node
/[NO]KEEP_MAP
/LOG
clear condition counters specified by name

CLEAR DEVICE –
Clear all active workstations

CLEAR ELEMENT name dir base node
/LOG
/[NO]KEEP_MAP
Clear Element.Set values of a Data Element to zero.

CLEAR PICTURE picture frame pic_dir base node
/[NO]KEEP_MAP
/[NO]LOG
Clear spectra defined in a picture data element

CLEAR SPECTRUM name spec_dir base node
/LOG
/[NO]KEEP_MAP
clear one (or all) spectrum

CLOSE FILE –
Close data input file.

CLOSE OUTPUT FILE –
Close list mode dump file

CNAF VME VMEcrate,processor C N A F times data ID
dummy node
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Execute CNAF

COMPRESS BASE base file
/DISMOUNT
Compress and copy data base (the copy cannot be mounted as GOOSY data base!). Command is executed in MUTIL.

CONVERT BASE base file size
Convert data base.

COPY BASE base file size area
/DISMOUNT
Expand and copy data base.

COPY CONDITION name destname dir dest_dir
base destbase node destnode
/REPLACE
/CONFIRM
/LOG
/[NO]KEEP_MAP
Copy source Condition to destination Condition

COPY ELEMENT name destname dir destdir destpool
base destbase node destnode
/REPLACE
/ALL
/NOCONFIRM
/[NO]KEEP_MAP
Copy source Dataelement to destination Dataelement

COPY FILE file outfile skip buffers
Output GOOSY list mode data file (called in MUTIL).

COPY MEMBER member destmember dir destdir
base destbase node destnode
/[NO]KEEP_MAP
Copy Data Element member to another Data Element member

COPY Polygon name destname poly_dir destpoly_dir
base dest_base node destnode

/REPLACE
/CONFIRM
/LOG
/[NO]KEEP_MAP
Copy source Polygon to destination Polygon

COPY SPECTRUM name dest_name spec_dir destspec_dir
base destbase node destnode
/REPLACE
/CONFIRM
/LOG
/[NO]KEEP_MAP
Copy source Spectrum to destination Spectrum

CREATE ALIAS name string environment
/GLOBAL
Create alias name (GOOSY, MDBM, MDISP, MUTIL).
In DCL use command ALIAS CREATE or CRALI
Please use DCL-command CRALI to create alias.

CREATE AREA area base pool areabytes cluster
/[NO]KEEP_MAP
Create an Area in a Data Base

CREATE BASE – base basefile adentries mdentries pdentries tdentries basepages
/PERMANENT/TEMPORARY
/GLOBAL_SEC/SYSTEM_GLOBALSEC
Create a new Data Base (section)

CREATE CALIBRATION FIXED name entries cal_dir
cal_pool base node
/[NO]KEEP_MAP
Create a Data Element for fixed calibration.

CREATE CALIBRATION FLOAT name entries cal_dir
cal_pool base node
/KEEP_MAP
Create Data Element for float calibration.

CREATE CALIBRATION LINEAR name cal_dir cal_pool
base node
/[NO]KEEP_MAP
Create Data Element for linear calibration.

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

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

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

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
create a pattern condition

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

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

CREATE DIRECTORY dir dedentries base

/[NO]KEEP_MAP
Create a Data Element Directory in a Data Base

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
Create a spectrum dynamic list entry

CREATE DYNAMIC ENTRY COMPOSED dyn_list condition dyn_dir cond_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Create a composed condition dynamic list entry

CREATE DYNAMIC ENTRY FUNCTION dyn_list condition module parameter dyn_dir par_dir cond_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Create a function condition dynamic list entry

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
Create an indexed spectrum dynamic list entry

CREATE DYNAMIC ENTRY MULTIWINDOW dyn_list condition parameter dyn_dir par_dir cond_dir base node
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Create a multiwindow condition dynamic list entry

CREATE DYNAMIC ENTRY PATTERN dyn_list condition parameter dyn_dir base node

/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Create a pattern condition dynamic list entry

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
Create a polygon condition dynamic list entry

CREATE DYNAMIC ENTRY PROCEDURE dyn_list module parameter condition
dyn_dir par_dir cond_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Create a procedure call dynamic list entry

CREATE DYNAMIC ENTRY SCATTER dyn_list picture process condition dyn_dir
base node
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Create a scatter plot dynamic list entry

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
Create a spectrum dynamic list entry

CREATE DYNAMIC ENTRY WINDOW dyn_list condition parameter dyn_dir
par_dir cond_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Create a window condition dynamic list entry

CREATE DYNAMIC LIST dyn_list entries dyn_dir pool
buffer base node
/[NO]KEEP_MAP
Create a dynamic list in a Data Base

CREATE ELEMENT name pool typename refer datalength
cluster queuehead dir base node
/[NO]PROTECT
/[NO]REPLACE
/[NO]KEEP_MAP
Create a Data Element in a Data Base

CREATE ENVIRONMENT environment
Create a GOOSY environment

CREATE LINK link_from link_to dir base node
/MULTIPLE
/[NO]KEEP_MAP
Create a link between two Data Elements

CREATE OVERLAY picture frame spectrum xparam yparam
trans=(xfactor,xoffset,yfactor,yoffset) dynshift node base pic_dir
spec_dir par_dir
/[NO]KEEP_MAP
Add spectrum or scatterparameter to frame of a picture data element

CREATE PICTURE name frames condition object
pic_dir pic_pool cond_dir par_dir
base node
/[NO]PROMPT
/[NO]KEEP_MAP
Create a picture Data Element

CREATE POLYGON polygon points
poly_dir poly_pool base node
/[NO]KEEP_MAP
Create a polygon in a Data Base

CREATE POOL pool areaminsize base
/[NO]KEEP_MAP
Create a Pool in a Data Base

CREATE PROCESS S image type
input output command priority
/DEBUG/RESTART/DUMP

Create a component in GOOSY environment.

CREATE PROGRAM file structure
/MBD /J11
/STRUCTURE
/[NO]PROGRAM
/COMPILE
Generates J11 stand alone programs and event data element declarations from CAMAC description file. (called in MUTIL)

CREATE SESSION environment session
Create a GOOSY session

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
create a spectrum

CREATE TABLE CONDITION name entries
directory pool base node
/[NO]KEEP
create condition bit table

CREATE TABLE SPECTRUM name entries
directory pool base node
/[NO]KEEP
create spectrum bit table

CREATE TYPE typefilename base
/COMPILE/REFERFILE/COMPREF/NOCOMPNOREF
/[NO]KEEP_MAP
Create a Type descriptor from a PL/I structure.

DEALLOCATE DEVICE device

Deallocate a graphical device

DEBUG VME MEMORY start end value
VMEcrate,processor ID dummy crate node
/READ/WRITE/COPY [=OPER]
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]

Read/write/copy VME memory.

DECALIBRATE SPECTRUM spectrum spec_dir base node

Disconnect a spectrum from its calibration.

DECOMPRESS BASE file base basefile
/MOUNT

Restores compressed and copied data base. Command is executed by MUTIL.

DEFINE DISPLAY HEADER string

Define display header line.

DEFINE DISPLAY PICTURE update refresh
/LIN/LOG /SQRT [SCALE]
/XLIN/XLOG/XSQRT [=X_SCALE]
/YLIN/YLOG/YSQRT [=Y_SCALE]
/ZLIN/ZLOG/ZSQRT [=Z_SCALE]
/[NO]ACTIVE
/[NO]ERROR
/[NO]WINDOW
/[NO]LIFE
/[NO]ROTATE
/[NO]SMOOTH
/[NO]LETTER
/[NO]NUMBER
/NOCAL/CALAX/CALSPEC [=CALIB]
/[NO]CHANNELS
/FULL/LAST/ACTUAL [=RANGE]
/AUTOSCALE/SCALE [=SCALE_RANGE]
/HISTO/VECTOR/MARKER [=STYLE]
/CONTOUR/ISO/CLUSTER/SCATTER

Set Spectrum parameter

DEFINE DISPLAY SPECTRUM limits scalim update refresh
/LIN/LOG /SQRT [SCALE]
/XLIN/XLOG/XSQRT [=X_SCALE]
/YLIN/YLOG/YSQRT [=Y_SCALE]
/ZLIN/ZLOG/ZSQRT [=Z_SCALE]
/[NO]ACTIVE
/[NO]ERROR
/[NO]WINDOW
/[NO]LIFE
/[NO]ROTATE
/[NO]SMOOTH
/[NO]LETTER
/[NO]NUMBER
/NOCAL/CALAX/CALSPEC [=CALIB]
/[NO]CHANNELS
/FULL/LAST/ACTUAL [=RANGE]
/AUTOSCALE/SCALE [=SCALE_RANGE]
/HISTO/VECTOR/MARKER [=STYLE]
/CONTOUR/ISO/CLUSTER/SCATTER
Set Spectrum parameter

DEFINE FRAME SETUP tic_number text_font linewidth xyratio channels
/[NO]GRID
/[NO]TICOUTSIDE
/[NO]INFO [=TYPE]
Define the design of picture frames

DEFINE PICTURE SETUP picture rows frames pic_dir base node
/EQUAL
Define frame organization on screen for one picture

DELETE ALIAS name environment
/GLOBAL
Delete alias name (in MUTIL, GOOSY, MDBM, MDISP). Please use DCL-command DLALI to delete alias.

DELETE CALIBRATION calibration node base cal_dir
/[NO]LOG
/[NO]KEEP_MAP
Delete a calibration Data Element.

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

delete a condition

DELETE DYNAMIC ENTRY dyn_type dyn_list
namelist aux dyn_dir base node
/UPDATE
/[NO]KEEP_MAP
Delete a Dynamic List entry

DELETE DYNAMIC LIST dyn_list dyn_dir base node
/[NO]KEEP_MAP
Delete a Dynamic List

DELETE ELEMENT name dir base node
/UNPROTECT
/[NO]KEEP_MAP
Delete a Data Element

DELETE ENVIRONMENT –
Delete a GOOSY environment

DELETE LINK link_from link_to dir base node
/ALL
/MATCH/IN/OUT
/[NO]KEEP_MAP
Delete Data Element link(s)

DELETE OVERLAY picture frame node base pic_dir
/[NO]KEEP_MAP
Delete the defined overlayed spectra or scatterplot parameters for the specified frames.

DELETE PICTURE picture node base pic_dir
/[NO]LOG
/[NO]KEEP_MAP
Delete a Picture Data Element.

DELETE POLYGON – name poly_dir base node
/[NO]CONFIRM
/[NO]KEEP_MAP
delete a polygon

DELETE POOL pool base
/[NO]KEEP_MAP
Delete a Data Base Pool

DELETE PROCESS type
Delete a GOOSY subprocess

DELETE SECTION base
Delete Global Section attributes

DELETE SPECTRUM – name spec_dir base node
/CAMAC
/[NO]CONFIRM
/[NO]KEEP_MAP
delete a spectrum

DETACH ANALYSIS –
Detach data base of analysis.

DETACH BASE base node
Detach data base.

DETACH BASE base node
Detach data base.

DETACH DISPLAY base

Detach display data base.

DETACH DYNAMIC LIST dyn_list dyn_dir base node
detach dynamic list

DISMOUNT BASE base
Delete Global Section attribute of Data Base

DISMOUNT TAPE device
/[NO]UNLOAD
Dismount tape.

DISPLAY CALIBRATION name cal_dir base node
Display calibration table.

DISPLAY CONDITION condition frame dimension
cond_dir base node
/CHAN/CALIB [=CALIBR]
/DISTRIBUTE
/XAXIS/YAXIS [=AXIS]
Display window condition limits

DISPLAY GRAPH frame file module image
Display user graphics.

DISPLAY METAFILE file directory
Read screen image from file and display it

DISPLAY PICTURE picture condition object dyn_scat
binfactor update refresh
binfactor update refresh
node base pic_dir dyn_dir cond_dir
buffer_size
/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
/SPECIFIED/FULL/LAST/ACTUAL [=RANGE]
/AUTOSCALE/SCALE [=SCALE_RANGE]
/HISTO/VECTOR/MARKER/CONTOUR-
/POINT/ISO/CLUSTER/SCATTER [=STYLE]
/TEMP /GLOBAL [=MODE]
/NOSCATTER
/[NO]ERROR
Display screen image as described by picture

DISPLAY POINT point frame
/LOOP
[CALIBR=]/CHAN/CALIB
Mark point on screen and get channel contents.

DISPLAY POLYGON polygon frame poly_dir base node
/FILL
Display polygon points.

DISPLAY SCATTER xparam yparam limits condition
object dyn_scat xletter yletter
node base par_dir dyn_dir
cond_dir buffer_size
/LAST /ACTUAL [=RANGE]

```
                              /TEMP /GLOBAL [=MODE]
                              /NOSCATTER
```
Display single scatter frame for two parameters.

DISPLAY SPECTRUM spectrum limits scalim

```
    binfactor update refresh cuts theta phi
    node base spec_dir
    /LIN /LOG /SQRT [=SCALE]
    /XLIN /XLOG /XSQRT [=XSCALE]
    /YLIN /YLOG /YSQRT [=YSCALE]
    /ZLIN /ZLOG /ZSQRT [=ZSCALE]
    /[NO]WINDOW
    /[NO]LIFE
    /[NO]ROTATE
    /[NO]SMOOTH
    /[NO]LETTER
    /[NO]NUMBER
    /NOCAL/CALAX/CALSPEC [=CALIB]
    /[NO]CHANNEL
    /FULL/LAST/ACTUAL [=RANGE]
    /AUTOSCALE/SCALE [=SCALE_RANGE]
    /HISTO/VECTOR/MARKER/CONTOUR-
    /POINT/ISO/CLUSTER/SCATTER [=STYLE]
    /TEMP /[NO]GLOBAL [=MODE]
    /[NO]ERROR
```
Display spectrum using default picture

DISPLAY TEXT text frame xposition yposition font size

```
        /CENTER /LEFT /RIGHT [=LOCATE]
        /ABSOLUTE /RELATIVE [=UNIT]
```
Display text into box specified by cursor

DUMP MBD FROM=f TO=t BDO=bdo BDD=bdd

```
      /HEXADECIMAL/DECIMAL/OCTAL/BIT
      /INSTRUCTION
```
Format a MBD memory dump

DUMP SPECTRUM name spec_dir base node output

```
  /[NO]KEEP_MAP
```
dump spectra to file 'outfile' in ASCII fornmat for transfer to SATAN VSAM library on the IBM

DUMP STARBURST FROM=f TO=t C=c N=n TSK=file

```
              /HEXADECIMAL/DECIMAL/OCTAL/BIT
```

Format a STARBURST memory dump through MBD

EXECUTE VME command VMEcrate,processor ID
dummy subcrate node
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Execute command on remote VME processor

EXPAND limits frame
/CHAN/CALIB [=CALIBR]
/LOG/LIN/SQRT [=SCALE]
/XLOG/XLIN/XSQRT [=X_SCALE]
/YLOG/YLIN/YSQRT [=Y_SCALE]
Expand spectrum or scatterplot within specified window.

FIT SPECTRUM frame poly window iter file
/[NO]OUTPUT
/[NO]APPEND
/BACKGROUND
/GAUSS
/SAMEWIDTH
/SHOW
/[NO]ZERO
/[NO]MARK
/NOERROR /STATISTICAL [=ERROR]
Fit spectrum with polynom and/or with gaussian peaks

FOREIGN ACQUISITION string longword
/X /Y /Z [=QUAL]
/[NO]SWITCH
Whatever

FREEZE SPECTRUM – name spec_dir base node
/[NO]KEEP_MAP
freeze a spectrum, inhibit accumulation

INITIALIZE ACQUISITION mailbox size count
in_buffers out_buffers
node command data
/MBD /J11 /FILE/FOREIGN/VME
/PAGE /BYTE /KBYTE
/MBX
Init data taking

INITIALIZE ANALYSIS base1 base2 base3
/[NO]ANALYSIS
/[NO]UNPACK
/[NO]PACK
/[NO]START
/[NO]STOP
/[NO]BASE
Reinitialize analysis (Analaysis must be stopped)

INITIALIZE CAMAC VMEcrate,processor ID dummy node
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Initialize CAMAC

INTEGRATE window frame file
/[NO]OUTPUT
/[NO]APPEND
/CHAN/CALIB [=CALIBR]
/LOOP
Integrate specified window

LOAD J11 file events
/KEEP /COMPRESS
Load CAMAC module table into J11

LOAD LRS_2365 file C=c N=n
/[NO]LOG
/[NO]DUMP
Load definitions in a LRS 2365 logic matrix

LOAD MBD file
/EXECUTIVE
Load microcode in MBD, either executive or a channel program.

LOAD MODULE ACQUISITION image module init
/START/STOP [=TYPE]
Load module from sharable image.

LOAD MODULE ANALYSIS image module init
/START/STOP/UNPACK/PACK/ANAL [=TYPE]
Load module from sharable image.

LOAD STARBURST file C=c N=n
/[NO]HALT

/BOOT/INIT
Load a system or task image in the STARBURST memory.

LOAD VME PROGRAM file procrate processor id subcrate
node
/TABLE
/[NO]LOAD
/RESET
/FEP/EB/ROP
/CVI/CVC/CAV/AEB/VME/EBI
/USER
/[NO]SYNC
/[NO]SYSTEM
Load programs into VME processors

LOAD VME TABLE file trigger VMEcrate,processor ID
subcrate node log
/[NO]LOAD
/LOG
/OVER
/FEP/EB
/ROP/ROC/AEB/VME
Load tables into VME processors. See also I$VMETAB

LOCATE BASE base
/[NO]KEEP_MAP
Locate a Data Base
For test purpose only.

LOCATE DIRECTORY dir base
/[NO]KEEP_MAP
Locate a Data Element Directory
For test purpose only.

LOCATE ELEMENT name
/[NO]KEEP_MAP
Locate a Data Element name array
For test purpose only.

LOCATE ID element dir base
/[NO]KEEP_MAP
Locate a Data Element by Directory index
For test purpose only.

LOCATE POOL pool base
  /[NO]KEEP_MAP
Locate a Pool in a Data Base
 For test purpose only.

LOCATE QUEUEELEMENT element
  /[NO]KEEP_MAP
Locate a queue Data Element name array
 For test purpose only.

LOCATE TYPE name base
  /[NO]KEEP_MAP
Locate a Type descriptor
 For test purpose only.

MODIFY DIRECTORY dir entries base
  /[NO]KEEP_MAP
Modify a Data Element Directory in a Data Base

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

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

MODIFY TABLE CONDITION name entries
directory pool base node
/[NO]KEEP
modify condition bit table

MODIFY TABLE SPECTRUM name entries
directory pool base node
/[NO]KEEP
modify spectrum bit table

MOUNT BASE base basefile
/PERMANENT/TEMPORARY
/GLOBAL_SEC/SYSTEM_GLOBALSEC
Mount an existing Data Base (section)

MOUNT TAPE device label blocksize density
/INITIALIZE
/DISMOUNT
/TK50 /TK70 /EXABYTE
Mount RMS tape.

OPEN FILE filename device directory headerfile
/[NO]HEADER
Open file for data input stream.

OPEN OUTPUT FILE file size number
device directory
headerinput headeroutput
/PROMPT
/EDIT
/AUTOMATIC
/ALLOCATE
/PAGE /BYTE /KBYTE /BUFFER
Open list mode dump file

OVERLAY spectrum xpara ypara binfactor
trans=(xfactor,xoffset,yfactor,yoffset)
frame node base spec_dir par_dir
/ADJUST

/SAVE
/[NO]ERROR
/HISTO/VECTOR/MARKER - =[STYLE]
/ISO/CLUSTER/CONTOUR/SCATTER
Add spectrum to frame

PATCH MBD ADDRESS=a VALUE=v C=c N=n A=a F=f
/NOCONFIRM
Patch a MBD memory location

PATCH STARBURST ADDRESS=a VALUE=v C=c N=n
/NOCONFIRM
Patch a STARBURST memory location

PLOT METAFILE file type command queue copies font
Plot a metafile.

PLOT PICTURE type command queue copies font file
/[NO]FLAG
/[NO]PRINT
Send the current active picture to a plotter

PLOT PLOTFILE file command queue copies
/[NO]DELETE
/[NO]FLAG
Plot device specific plotfile

PRINT command printer form file
/DELETE
Plot device specific plotfile

PROJECT spectrum target window dimension node base spec_dir
/ADD/SUB/CLEAR [=ACTION]
/POLYGON
Project window in 2-dim spectrum

PROTECT SPECTRUM name spec_dir base node
/LOG
/[NO]KEEP_MAP
protect one (or all) spectrum

PROTOCOL line
/SESSION/COMMAND/GLOBAL
Write line(s) into log file.

READ CAMAC SPECTRUM name spec_dir base node
/ADD
/LOG
/[NO]KEEP_MAP
Read spectrum data from MR2000 into GOOSY spectrum

REFRESH frame
/[NO]UPDATE
/[NO]WINDOW
/[NO]OVER
Refresh picture as displayed

RELEASE MBD CHANNEL channel_no
Release MBD channel to allow loading of new code

REPLACE CONDITION WINDOW condition limits dimension cond_dir base node
/CHANN/CALIBR [=CALIBR]
/XAXIS/YAXIS [=AXIS]
Set or replace window condition by cursor

REPLACE POLYGON polygon frame xpoints ypoints
poly_dir base node poly_pool
/DELETE/MODIFY [=MODE]
Set points in polygon

RESET ACQUISITION –
RESET ACQU

RESET CAMAC VMEcrate,processor ID dummy node
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Reset CAMAC

RESET MBD –
Reset MBD and release all active channels

SAVE DISPLAY file directory
Save displayed picture in a metafile.

SEND DATA VMEcrate,processor ID dummy crate node
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Read one subevent.

SET ACQUISITION in_buffers out_buffers events
/[NO]SYNCHRONOUS
/[NO]EXCLUSIVE
/MAILBOX /NET
/[NO]CHECK
/[NO]COMPRESS
/[NO]KEEP
/[NO]START
/[NO]STOP
Set data taking parameters.

SET ANALYSIS –
/[NO]ANALYSIS
/[NO]DYNAMIC
/[NO]SYNCHRON
/[NO]EVENT
/[NO]START
/[NO]STOP
/[NO]FOREIGN
/[NO]TABLES
Set analysis parameters.

SET CALIBRATION FIXED name unit start step input calib uncalib parameters
polynom module image cal_dir base node
/[NO]FILE
/FIT/MODULE/PARAMETER/PROMPT/TABLE [=ACTION]
Set table for a fixed-type calibration.

SET CALIBRATION FLOAT name unit input module image
cal_dir base node
/[NO]FILE
Set table for a float-type calibrations.

SET CALIBRATION LINEAR name unit input parameters
calib uncalib cal_dir base node
/FILE
/FIT/PROMPT/PARAMETER [=/ACTION]
Set parameters for a linear calibration.

SET CONDITION PATTERN – name pattern invpat index cond_dir base node
/[NO]KEEP_MAP
set stored pattern of a pattern condition

SET CONDITION WINDOW condition limits dimension cond_dir base node

/[NO]KEEP_MAP
Set window condition

SET DEVICE COLOR name index r g b
/UPDATE
Allocate a graphical device

SET DISPLAY MODE –
/STANDARD/FAST [=VERSION]
Select display version.

SET DYNAMIC LIST dyn_list dyn_type key value dyn_dir base node
Modify attached dynamic list

SET EVENT INPUT name type directory base
Set input event data element.

SET EVENT OUTPUT name type directory base
Set output event data element.

SET FASTBUS PEDESTAL sample trigger
VMEcrate,processor ID dummy crate node
/ON/OFF [=ONOFF]
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Set fastbus pedestal subtraction on/off.

SET LETTERING – specname dim text spec_dir base node
/[NO]KEEP_MAP
set lettering at display axisses

SET LOCK OUTPUT –
/[NO]PRCID
/[NO]LKID
/[NO]PARID
/[NO]UIC
/[NO]PRCNAM
/[NO]STATE
Set lock output spezification (called in MUTIL)

SET MEMBER mem_spec value dir base node
/[NO]KEEP_MAP
/[NO]LOG
Change value of an Data Element member or a full Data Element member array (wildcarded).

SET MWPC id Number adc_threshold dis_threshold
Reject_pass
/ALL/L/M/N CHAMBER
/X/Y/A LAYER
/[NO]REJECT
/[NO]FASTCLEAR
/SHOW
/RESET
/[NO]LOAD
SET MWPC modes.

SET RANDOM type subtype channels datawords
/COMPRESS
/PRINT
/SPAN
Set some random generator parameters

SET SCATTER BUFFER value dyn_list dyn_dir base node
Set scatter buffer size

SET SPECTRUM POINT spectrum xpoint ypoint file
spec_dir base node
/[NO]KEEP_MAP
Set spectrum channel to specified value.

SET VME BUFFER buffers size VMEcrate,processor ID
dummy node
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
/STOP/RESET [=LAST]
Setup frontend buffers

SET VME CONTROL name value VMEcrate,processor ID
dummy node
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Set value in control structure

SET VME INPUT –
/HVR /NET /TDAS /OFF /CHECK [=PATH]
Select lmd data path

SET VME TRIGGER VMEcrate,processor ID dummy
crate fastclear conversion node
/RESET
/MASTER
/ENABLE/DISABLE [=ENABLE]
/[NO]LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Set trigger module.

SHOW ACQUISITION timer output
/PRINT
/OUTFILE
/INFILE
/CLEAR
/RUN
/SETUP
/BRIEF
/[NO]RATE
Show acquisition status

SHOW ALIAS name environment
/GLOBAL/ACTIVE [=SCOPE]
Show alias name (in MUTIL, GOOSY, MDBM, MDISP).

SHOW ANALYSIS timer output
/PRINT
/BRIEF
/OUTFILE
/INFILE
/CLEAR
/[NO]RATE
Show analysis status

SHOW AREA area base output
/[NO]FULL
/[NO]DIRECTORY
/[NO]KEEP_MAP
/PRINT
Show an Area in a Data Base

SHOW BUFFER DUMP –
Show writing to LMD file.

SHOW CALIBRATION calibration output cal_dir base node
/PRINT
/LINKS
/TABLE
Show calibration information

SHOW CAMAC SPECTRUM name spec_dir
base node output width
/PRINT
/ATTRIBUTES/DATA/ALL/STATUS
/FULL
/MEMBERS
/LOG
/INTEGRAL
/ZERO
/CAMAC
/[NO]KEEP_MAP
show CAMAC spectra

SHOW CONDITION N name cond_dir base node output
/PRINT
/FULL
/MEMBERS
/ATTRIBUTES/COUNTERS/FLAGS/STATUS/ALL
/POLY/WIND/MULTI/PATT/COMP/FUNC/ANY
/[NO]KEEP_MAP
show attributes of a condition

SHOW DEVICES SHOW DEVICE
Show all allocated user devices

SHOW DIRECTORY dir base output
/[NO]FULL
/[NO]DIRECTORY
/[NO]KEEP_MAP
/PRINT
Show Data Elements of a Data Base Directory

SHOW DISPLAY GLOBALS –
/SPECTRUM
/PICTURE
Show global display parameter.

SHOW DYNAMIC ATTACHED dyn_list dyn_type dyn_dir base node output

/PRINT
/[NO]QUEUE
/FULL
Show attached dynamic list

SHOW DYNAMIC LIST dyn_list dyn_type dyn_dir base node output
/[NO]KEEP_MAP
/PRINT
Show Dynamic List (elements)

SHOW ELEMENT name dir base node output
/DECIMAL/HEXADECIMAL/OCTAL/BINARY
/LONGWORD/WORD/BYTE
/[NO]DATA
/[NO]FULL
/[NO]KEEP_MAP
/PRINT
Show a Data Element descriptor and value

SHOW GOOSY STATUS environment p1 p2 p3
/$TMR
/$ANL
GOOSY Status report

SHOW HOME_BLOCK base output
/PRINT
/BITMAP
Show the Home Block of a Data Base

SHOW LINK link_from dir base node
output
/IN/OUT/ALL
/MATCH/TREE
/[NO]KEEP_MAP
/PRINT
Show Data Element link(s)

SHOW LOCKS process lock request granted
/PROCESS
/LOCKS
/EXCLUSIVE
Display system locks

SHOW MAPPING base area
Show mapping of a Data Base

SHOW MAPPING base area
Show mapping of a Data Base

SHOW MAPPING base area
Show mapping of a Data Base

SHOW MEMBER mem_spec dir base node output
/[NO]KEEP_MAP
/PRINT
Show value of a Data Element member or a full Data Element member array (wildcarded).

SHOW PICTURE picture output pic_dir base node
/PRINT
/FULL
/DATA
/[NO]KEEP_MAP
Show picture information

SHOW POLYGON N name poly_dir base node output
/PRINT
/[NO]FULL
/[NO]DATA
/[NO]KEEP_MAP
show attributes of a polygon

SHOW POOL pool base output
/[NO]FULL
/[NO]DIRECTORY
/[NO]KEEP_MAP
/PRINT
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.

SHOW SCATTER BUFFER dyn_list dyn_dir base node
Show scatter buffer size

SHOW SPECTRUM name spec_dir base node output width
/PRINT
/ATTRIBUTES/DATA/ALL/STATUS
/FULL
/MEMBERS

/LOG
/INTEGRAL
/ZERO
/CAMAC
/[NO]KEEP_MAP
show spectra

SHOW STARBURST C=c N=n
Show the execution parameters of a STARBURST.

SHOW TABLE name table tab_dir base node output
/CONDITION /SPECTRUM /ALL
/CONTENT /COUNTS
/PRINT
/[NO]KEEP_MAP
show flag tables from the analysis

SHOW TP0 KEYPAD –
Display of GOOSY auxiliary keypad definition active in GOOSY prompter MGOOTP0 May be called by PF2 key.

SHOW TREE base output
/[NO]KEEP_MAP
/PRINT
Show indices of a Data Base Pool Directory
For test purpose only.

SHOW TYPE type base output
Show a Type descriptor. Wild card are allowed.

SHOW VME CONTROL name VMEcrate,processor ID
dummy node
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Show values in control structure

SHOW VME SETUP file /FULL
Show VME setup.

SLEEP –
Put the prompter in a HIBERNATE state.

START ACQUISITION buffers events
skip_buf skip_event

/CLEAR /NET /STOP /RESET
Start data taking

START ANALYSIS OUTPUT file size buffersize
device directory
type subtype stream
headerinput headeroutput
/PROMPT
/EDIT
/[NO]OPEN
/[NO]SYNCHRON
/COPY/COMPRESS/INPUT
/MBD/J11
/BYTE/KBYTE/PAGE/BUFFER
Start data output from analysis. Output is done to DECnet. If a file is specified, output is written to file too.

START ANALYSIS RANDOM bufevents events
Start analysis for Monte Carlo.

START BUFFER DUMP file buffers size
/ANALYSIS
/KBYTE
Open file and write buffers to LMD file.

START DYNAMIC LIST –
dyn_list dyn_type dyn_dir base node
Start execution of dynamic list

START INPUT FILE file buffers events
skip_buffer skip_event
device directory
/CLEAR
/OPEN
/FOREIGN
/[NO]HEADER
Start data analysis from file at current position. Open it if it was not open.

START INPUT MAILBOX mbx_name mbx_number
buffers events bufevents
skip_buffers size
Open input stream from mailbox

START INPUT NET node environment component
buffers events
/TMR/ANL
/MULTI
Open input stream from network

START MR2000 0 branch crate station
/INITIALIZE
Start/initialize MR2000.

START OUTPUT FILE file size number
device directory
headerinput headeroutput
/PROMPT
/EDIT
/[NO]OPEN
/AUTOMATIC
/ALLOCATE
/PAGE /BYTE /KBYTE /BUFFER
Start list mode dump

START RUN name
Start run.

START SCATTER –
/SYNCHRONOUS /ASYNCHRONOUS [=/MODE]
Start scatter plots for actual picture

START VME VMEcrate,processor ID dummy node
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Send START command to NET

STOP ACQUISITION /ABORT /CLOSE /STOP/RESET
Stop data taking

STOP ANALYSIS OUTPUT /[NO]CLOSE
Stop data output from analysis

STOP ANALYSIS RANDOM –
Close input stream from mailbox

STOP BUFFER DUMP –
Stop writing to LMD file. Close File

STOP DYNAMIC LIST dyn_list dyn_type dyn_dir base node
Stop execution of dynamic list

STOP INPUT FILE –
/CLOSE
Stop reading input file, optional close.

STOP INPUT MAILBOX mbx_num
Close input stream from mailbox

STOP INPUT NET –
Close input stream from DECnet

STOP MR2000 0 branch crate station
Stop MR2000.

STOP OUTPUT FILE –
/[NO]CLOSE
Stop list mode dump

STOP RUN /STOP /ABORT /CLOSE
Stop run.

STOP SCATTER –
Stop scatter plots for actual picture

STOP VME VMEcrate,processor ID dummy node
/LOAD
/ALL/FEP/EB [=DESTINATION]
/CVI/CAV/EBI [=CONTROL]
Send STOP command to NET

STORE LRS_2365 C=c N=n FILE=file
/DUMP/NODUMP
Read back definitions from a LRS 2365 logic matrix and write them formated to a file (or SYS$OUTPUT).

STORE MBD file
Store a MBD memory dump in a file.

SUMUP SPECTRUM spectrum window file spec_dir base node
/CHAN/CALIB [=CALIBR]
/[NO]OUTPUT
/[NO]APPEND
/[NO]KEEP_MAP
Integrate specified window

TEST BOR_1802 –

```
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
              /FULL
```

Perform tests with a BORER 1802 Dataway display

TEST CAMAC B=b C=c N=n TYPE=*

```
  /STOP
  /LIST
```

Common functions for CAMAC tests.

TEST GSI_IOL –

```
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
    /LOOPBACK
```

Test a GSI I/O LAM (IOL) module.

TEST LRS_2228 –

```
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
    /VALUE
```

Test a LRS 2228 TDC module.

TEST LRS_2249 –

```
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
    /PEDESTAL
```

Test a LRS 2249 ADC module.

TEST LRS_2551 –

```
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
```

Test a LRS 2551 scaler module.

TEST LRS_4432 –

```
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
```

Test a LRS 4432 scaler module.

TEST LRS_4434 –

B=b C=c N=n
REPEAT=r
/LIST /STOP /RUN /START /LOOP

Test a LRS 4434 scaler module.

TEST MPI_BIT –

B=b C=c N=n
REPEAT=r
/LIST /STOP /RUN /START /LOOP

Test a MPI bit encoder module.

TEST MPI_TDC –

B=b C=c N=n
REPEAT=r
/LIST /STOP /RUN /START /LOOP
/VALUE

Test a MPI slow TDC module.

TEST REGISTER –

B=b C=c N=n
A=a F=f
REPEAT=r
/LIST /STOP /RUN /START /LOOP
WIDTH=w /FULL

Perform tests with a any register in a CAMAC module.

TEST SEN_2047 –

B=b C=c N=n
REPEAT=r
/LIST /STOP /RUN /START /LOOP

Test a SEN 2047 pattern unit.

TEST SEN_2090 –

B=b C=c N=n
REPEAT=r
/LIST /STOP /RUN /START /LOOP

Test a SEN 2090 video display driver.

TYPE BUFFER number /HEADER

Start to type data buffers

TYPE EVENT number id /SAMPLE /HEADER

Start to type events

TYPE FILE file skip buffers events id outfile
/HEADER /DATA
/EVENTHEADER
/SAMPLE
/PRINT
Output GOOSY list mode data file (called in MUTIL).

UNFREEZE CONDITION name cond_dir base node
/[NO]KEEP_MAP
unfreeze a condition, enable the execution of a condition

UNFREEZE SPECTRUM name spec_dir base node
/[NO]KEEP_MAP
unfreeze a spectrum, enable the accumulation

UNPROTECT SPECTRUM name spec_dir base node
/LOG
/[NO]KEEP_MAP
Unprotect one (or all) spectrum

UPDATE BASE base
Update a Data Base, write all pages to Section File

UPDATE DYNAMIC LIST dyn_list dyn_dir base node
/[NO]KEEP_MAP
Update a Dynamic List

UPDATE FRAMES frames seconds
/REFRESH
Update frames in the actual picture on screen

WAIT seconds
Wait n seconds.

WRITE CAMAC SPECTRUM name spec_dir base node
/ADD
/LOG
/[NO]KEEP_MAP
Write spectrum data from GOOSY spectrum into MR2000

ZOOM FRAME frame picture dyn_scat
pic_dir dyn_dir buffer_size base node
Zoom one frame of a picture.

Appendix B

GOOSY Macro Summary

$ACCU	(type,base,dir,name,incr,dim,x1,x2,...) Accumulate spectrum
$ACCU1	(type,base,dir,name,ind,incr,dim,x1,x2,...) Accumulate 1-dim. indexed spectrum
$ACCU2	(type,base,dir,name,i1,i2,incr,dim,x1,x2,...) Accumulate 2-dim. indexed spectrum
$ATTACH	(type,base,access) Attach data base items
$COND	(type,base,dir,name,result,dim,x1,x2,...) Executes condition and returnes result.
$COND1	(type,base,dir,name,ind,result,dim,x1,x2,...) Executes 1-dim. indexed condition and returnes result
$COND2	(type,base,dir,name,i1,i2,result,dim,x1,x2,..) Executes 2-dim. indexed condition and returnes result
$DE	(base,dir,name,member) Data elements reference
$DE1	(base,dir,name,index,member) Data elements reference
$DE2	(base,dir,name,i1,i2,member) Data elements reference (2-dim)
$DETACH	(type,base) Detach data base items

$LOC	(type,base,dir,name,access,descr) Locate data elements for analysis
$LOC1	(type,base,dir,name,ll,ul,access,descr) Locate 1-dim. data element arrays for analysis
$LOC2	(type,base,dir,name,l1,u1,l2,u2 ,access,descr) Locate 2-dim. data element arrays for analysis
$MACRO	@INCLUDE $MACRO($MACRO) Initialize analysis macros
$SPEC	(type,base,dir,name,value,dim,x1,x2,...) Set spectrum channel
$SPEC1	(type,base,dir,name,ind,value,dim,x1,x2,...) Set channel in 1-dim. indexed spectrum
$SPEC2	(type,base,dir,name,i1,i2,value,dim,x1,x2,...) Set channel in 2-dim. indexed spectrum
ADD_MSG	@ADD_MSG(errorcode,arg1,arg2,arg3) accomplish the error message belonging to the errorcode by the specified arguments and write the message on the internal error- message stack.
BYTE	@BYTE(integer) returns the ASCII(EBCDIC) character whose code is equivalent to the given integer.
CALL	@CALL procedure performs a function call
CLR_MSG	@CLR_MSG clear the internal message stack on
DCL_MSG	@DCL_MSG(errorname); declaration of error
DMP_CLR_MSG	@DMP_CLR_MSG write the internal message stack on the screen
ENTRY	label: @ENTRY remember name of entry
INCLUDE	@INCLUDE lib(member) include PPL code

LOCAL_ERROR	@LOCAL_ERROR() resignals errors from lower procedure levels
ON_ANY_E	@ON_ANY_E(u_cleanup) catches all signaled errors, calls <u_cleanup> before resignaling the error
ON_ANY_F	@ON_ANY_F(u_cleanup) catches all signaled errors, calls <u_cleanup> before resignaling the error
ON_ANY_W	@ON_ANY_W(u_cleanup) catches all signaled errors, calls <u_cleanup> before resignaling the error
PROCEDURE	<label>:@PROCEDURE remembers the name of the current module
PUT_CLR_MSG	@PUT_CLR_MSG write the internal mesage stack on the screen
RANK	@RANK(char) returns a BIN FIXED (15) number which corresponds to the input character <char>.
REPEAT	@REPEAT(cv,i_repeat) return the string cv concatenated to cv i_repeat times
RET	@RET(errorcode) returns the error code to the calling procedure,
RET_ADD_MSG	@RET_ADD_MSG(errorcode,arg1,arg2,arg3) return and write specified message onto error stack
RET_SET_MSG	@RET_SET_MSG(errorcode,arg1,arg2,arg3) return and write specified message onto error stack
SIZE	@SIZE(reference) returns number of bytes allocated to the referenced variable
STORAGE	@STORAGE(reference) returns number of bytes allocated to the referenced variable
TRACE_MSG	@TRACE_MSG(errorcode,arg1,arg2,arg3) Write e trace message to the internal error stack. The errorcode is normally returned by another routine signaling an error.
TRIM	@TRIM(cv_string,cv_lead,cv_trail) remove leading and/or trailing characters from a string

Appendix C

GOOSY DCL Procedure Summary

acctng	@accttng p1 p2 p3	Takes the ACCOUNTNG file of VMS and do selective inquiries using ACCOUNTING (the VMS utility).
ALIAS	command arguments	Handle GOOSY alias command names
ATENVIR	ATENV*IR environment	Attach environment.
BFXT	@GOO$EXE:BFXT inirep action file /ID=MYDATA/MODE=CHAR /FILELN=fileln/MSGLEVEL=msglevel/NOEXECUTE	send data to the IBM via BFX/NETEX
CADDRESS	infile outfile /PRINTER=p	Create and optional print address files..
CADJUST	infile outfile /ADJUST/TAB	Adjust GOOSY documentation headers.
CALLEX	library filespec GL-switches	Extract Calling sequence from PLI source.
CALLTREE	input /MODULE= /OUTPUT= /TT /EXCLUDE=	Makes a calling tree out of a cross reference
CBACKUP	P1 P2 ... P8	Execute the VMS BACKUP utility and return an exit status based on the analysis of all errors.
CBATCHDCL	"dcl_line1" "dcl_line2" "dcl_line3" "dcl_line4"	

	/LOG_FILE=file /DEFAULT=dir /QUEUE=q /AFTER=time /NAME=n /CHARACTERISTICS=c/CPUTIME=t/RESTART /NODE=n /VERIFY/ACCOUNTING /WARNING/ERROR/FATAL Executes up to 4 DCL command lines in a BATCH job under the current default directory either on the local or an remote node.
CBINHEX	in_file out_file This procedure creates a HEX file containing the FDL description and the contents in hexadecimal format of another file. Those HEX files can be converted back with CHEXBIN and are usefull for sending them over stupid communication links.
CDBLQUOTE	input Replace every quote in a text string by a double quote.
CDCLCOM	inc module performs a compile of PL/I text contained in 'file' within a dummy frame (GOO$DM:U$DCLDU.PPL)
CDCLLIST	dsc p1 p2 p3 p4 p5 p6 p7 This procedure provides a mechanism which allows to pass parameters and qualifiers to DCL procedures in the same way as to DCL commands.
CDELSYM	prefix start end Delete global symbols created by CNAMELIST or CWILDCARD
CDIFFER	file1 file2 /LIST/PARA/DIFFER Allows wildcarded DIFFERENCE with suppressed output.
CDIRO	file-spec. /SIZE /DATE /FULL/TEST sorts a DIR command output
CDOCUM	CDOC module type formatter library Generates documentation from source file and inserts text module to library
CEASYMAIL	@GOO$EXE:CEASYMAIL file recipient /SUBJECT=s/DELETE Send a file as MAIL to a High Energy Physics DECnet user via EARN via the Mailer at CUNYVM.
CEDITDOC	CED file key Calls MGENHEAD for header generation and the VMS editor LSEDIT.

CFILTYPES	CFILT*YPES file switches /D*IRECT Outputs list of all file types on a directory
CHANAL	infile outfile /COPY/FULL Check GOOSY analysis programs.
CHECKDATE	newfile oldfile Checks wether newfile is younger then oldfile (status=1) or not (status=11)
CHEPMAIL	@GOO$EXE:CHEPMAIL file recipient /SUBJECT=s/DELETE Send a file as MAIL to a High Energy Physics DECnet user via EARN via the Mailer at CERNVAX.
CHEXBIN	in_file out_file This procedure processes a HEX file send of a stupid communication network and creates the file described in the HEX file.
CINCHLP	CINC*HLP format incl.libr.(module) help-libr. Generates documentation of include modules.
CINSNUM	CINS input output Inserts line numbers (9 char.) to source files
CLCOUNT	file symbol /NOHEAD Outputs number of lines of source files
CLINK	CL file/switches /switches Link programs. Defaults are updated.
CMAIL	file recipient /ED*IT /DEL*ETE /SUB*JECT=s /NOSE*LF /SPOOL=s Send or spool a MAIL with VMSmail or other mail systems on V8600B only
CMANUAL	CMAN <utility list> <format> <prefix> Composes GOOSY manuals (Programs, modules, procedures)
CMESSAGE	facility [switch] Utility to insert a new message in a message file
CMODMAP	file /COMPILE /OUTPUT=file Outputs list of modules called by module in file. (Only first level calls)
CMSGLIST	CMSG*LIST [directory][module] Generates script and help file of all GOOSY messages

CMTBACK	output Perform an image system backup for all disk devices on a magtape without verification.
CNAMELIST	<namelist> <prefix> /DIR*ECTORY /SHORT /LIB*RARY=<library> /SEL*ECT=<string> /EXC*LUDE=<string> /SEA*RCH=<string> /SIN*CE=<date> /BEF*ORE=<date> /OUT*PUT=<file> Defines a set of symbols with names got from <namelist>.
CNOTICE	recipient/NOSELF "subject" "line 1" ... "line 6" Send a short notice with VMSmail
COMPILE	COM*PILE file /PRE*QUAL=(list)/Q*UALIFIER=(list) /G*IPSY=list/LIBRARY=(list)/DEBUG/KEEP /OLB=library/SINCE=time/BEFORE=time/NEWPLI /FAST/MACRO/CALL/BATCH/COMPILE General compile procedure for all compilers THE PREVIOUS VERSION CAN BE CALLED BY OCOMPILE !
CONCAT	infile outfile /LOG/DELETE/CONFIRM/APPEND Concatenates input files to one output file.
COPTLIST	list keylist Parse IBM styled optional parameter list for DCL procedures.
CPLICOM	PPL_file/switches /switches %DEB %NEWPLI Compile PLI programs including certain text libraries. Defaults are updated.
CPRECOM	CPRE*COM file(switches) Call precompiler
CPRINT	file_list /DESTINATION=d Print a file on a spooled printer.
CPURGE	CPU*RGE filespec [purge qual.] Does a secure purging

CREDB	basename filename size[KB] /DYNLISTS=d /SPECTRA=s /CONDITIONS=c /PICTURES=p /DIRECTORIES=d /POOLS=p /POLYGONS=p /NEW /SAVE=file Create an preformat a new GOOSY data base.
CREMNUM	CREM input output col check Removes line numbers (<col> char.) from source files checking first <check> characters to be digits.
CRENVIR	CRENV*IR environment program component /ONLINE/OFFLINE/DEFAULT /$TMR/$ANL/$DSP/$DBM/J11 /[NO]DECWINDOW /PRIORITY=p/DELETE Creates a GOOSY environment and optional some GOOSY standard components
CREPEAT	CREP*EAT dcl_line Repeats a dcl command continuously, beginning on screen top.
CREPLACE	CREPL*ACE file old new /U*PPER/P*RINT/F*ORMAT/OUT=file Replaces old string by new calling MREPLACE. Documentation headers will be adjusted optionally.
CSYMDIR	@GOO$EXE:CSYMDIR Create symbols for SET DEFAULT from the directory tree
CSYMHLP	CSYM input output Calls MSYMHLP to generate help files from command procedures.
CTEXCOM	file /NOLIST/NOLATEX/PASSES=n/DVI=drv/DELETE=l 'Compile' TEX source using PLAIN TeX ot LaTeX
CTEXMANUAL	manual /REF*ORMAT /REL*EASE=r/VER*SION=v/BR*IEF Create GOOSY manual
CTRL_T	[process][output] Similar to an interactive CTRL_T, but works in DCL procedures.
CVTISOL	<input file>,<output file>[/SHORT] Convert ISOLDE spectra into SATAN readable format.
CWHAT	options arguments Activates the WHAT Utility to display details about the VMS system status.

CWV	– inquires a diary, 'Wiedervorlage'
D0_BACK	@D0_BACK output ALL Perform an image backup for disk device DUA0: on a magtape without verification.
DCFIBM	vaxfile "parm list" Sends VAXfile to IBM DCFTEMP.TEXT and starts batch job calling DCF on IBM
DLENVIR	DLENV*IR Delete current environment and all subprocesses
DOCIBM	DOC vaxfile [dest]PROP TWOPASS Sends a file to IBM DOCTEMP.TEXT(DUMMY) and starts batch job calling DOC
Document	Desciption of the module documentation tools The following commands are provided to generate documentation for VMS help utility and printer output (RNO and SCRIPT).
DTENVIR	DTENV*IR Detach environment.
DVIIBM	dvifile switches Sends DVI-file to IBM. via batch job
DVIPRI	– file(s) /DEVICE=dev /STARTPAGE=spage /PAGES=npage /DELETE /SEP*ARATE/[NO]PR*INT Print TeX's device independant (DVI) file
ECLINE	ECL*INE dcl-line /LIS*T /CONF*IRM /NOASS*IGN /$$1=list /$$2=list /1LIB*RARY=lib /2LIB*RARY=lib /1SIN*CE=date /2SIN*CE=date /1SH*ORT /2SH*ORT /1DIR*ECT /2DIR*ECT /1SEA*RCH=list /2SEA*RCH=list Execute dcl line with dummies replaced from list
EDDT	file /READ/PROFILE=prof Call full screen edit with a profile depending on the file type.

Error_Handling	Error and Message Handling, User's Guide Vers. 1.05 This documentation is available on-line, say 'HELP ERROR_HANDLING'.
ETHDEF	destination ethernet protocol interface Define logicals for ethernet connection to VME.
EXTRCALL	libfile Calls MCALLSYS or MCALLRTL to extract procedure call statments from a listing LIB /EXTRACT from a library
GIPSY	input output /LIST/DELETE Call GIPSY processor.
GLCNVPROJECT	@GOO$EXE:GLCNVPROJECT file project Convert project name within a history file
GLCNVV31	@GOO$EXE:GLCNVV31 file Convert history file format from version 3.1 to 4.0
GLCOMPILE	file Compile a file in the local test environment
GLCREATE	option arguments Creates datasets or entries in the history file directories.
GLDELETE	option arguments Deletes either datasets or information in the history file.
GLDOCUMENT	GLDOC*UMENT filespec /PUT extr.qual form.qual Generate documentation with GLEXTRACT and GLFORMAT.
GLEDIT	dataset /COMMENT=c/NOMARK/NOSUBMIT/NOCONFIRM Edit a dataset.
GLEXTRACT	file_list /MLIB=ml /CLIB=cl /MTLB=mt /MHLB=mh /CTLB=ct /CHLB=ch /TYPE=t/NOSUBMIT /SORT /DIRECT=file /CALL*LIB=tl Extract documentation from source files or libraries and store in libraries or file.
GLFORMAT	input output /SELECT=sl /EXCLUDE=el /TEX/SCRIPT/HELP/PRINT/RNO /BRIEF/COMMAND/TOC/HL2/LEV2 /LIB=l /HLB=ml /NOSUBMIT /LOG/DIAGNOSIS Format extracted documentation headers.

GLGET	dataset file /COMMENT="comm" Copy a dataset from a project directory or library to the current working directory.
GLINFO	key Call GOOSY information system
GLMAIL	"subject" /AUTHOR/USER/PROJECT/KEY=key/CONF=conf Send MAIL to current projects author group or user group and add this message in the appropriate news file.
GLMANAGER	option arguments Performs all project manager functions.
GLPUT	file dataset /COMMENT="comm" /SOURCE/GENERATED/FOREIGN /NOMARK /NOCORRELATE /NOSUBMIT /DOCUMENT /COMPILE /LINK /PROLOGUE /EPILOGUE /UPDATE /NEW/NOCONFIRM/NODELETE /MFORMAT=lib/CFORMAT=lib Store a file from the working directory to a project directory or library.
GLRELEASE	dataset /COMMENT=c /NOINTERLOCK Release a locked dataset without storing a dataset.
GLSATTR	GLSAT*TR type module /UPD*ATE Set attributes for certain types of modules
GLSET	option arguments Defines or changes attributes and characteristics of datasets in the history file or modifies execution characteristics of module management operations.
GLSHOW	option arguments Displays information about the current status of the project and it's modules.
GLTEST	option arguments Collects all operations of the local test environment.
GLTOOL	option arguments This command is a collection is software development tools available under the module managemant system.

GLUPDATE	dataset /NOMARK /NOCORRELATE /NOSUBMIT /DOCUMENT /COMPILE /LINK /PROLOGUE /EPILOGUE /UPDATE Mark a file for update actions.
GNEWS	outfile /SYSTEM/FILE Output all GOOSY mails.
GNOTES	command Read or write notes in one of the GOOSY notebooks.
GOOCONTROL	GOOC*ONTROL [CREATE]or [DISMOUNT] Defines logical name GOO$CONTROL for control data base. The data base is created and mounted optionally.
GUIDE	facility level /INIT=string/BRIEF/LIST/LASER Menu driven guide to use facilities.
HLPSCR	help-library module Extracts modules from a Help library and generates a new output file for SCRIPT
IBMSUBMIT	IBMSUBMIT vaxfilejcl /MAILADDR= /NOANSWER/MSGLEVEL= submit a JCL on the ibm and send optionally back the resulting output to the VAX
LANL	LA*NL obj_list /OLB=objlib/OPT=optfile/CMD=cmdfile /EXE=exefile /MAP=mapfile /DEBUG /SHARE/NOSHARE Link user specific analysis program
LIBCOPY	source_lib source_mod target_lib target_mod /EDIT /GL /LOG Copies text modules from one library to another.
LIBDEL	LIBD*EL library module Handle text library modules
LIBDIFF	source_lib source_mod target_lib target_mod /DIFFER Compare text modules from two libraries.

LIBEXTR	library module output Extract modules from text library.
LIBLIS	library module /FULL/SIN*CE=time /BEF*ORE=time/EXT*RACT/OUT*PUT=file-spec Lists or extracts specified modules of a library
LIBSEARCH	LIBS*EARCH library module list="search args" /SINCE=time/BEFORE=time/FILE Calls SEARCH on modules temporarily extracted from libraries (text only).
LIBTYPE	LIBT*YPE library module /PRINT /EDIT Handle text library modules
LINKJ11	objfile /COMPILE Link a J11 stand alone task
LOADKEYS	– Load F-keys of VT200/VT300.
LSHARIM	module image /GLOBAL=list /SHARE*LOG=name /MAP=mapfile /KEEP /GROUP /DEBUG Link modules into a sharable image.
MESDEF	facility /CLIB=/[NO]NEW/GLPUT/DELETE/LIST Generate message definition file for C programs.
MOVE	filespec dest /CONF /LOG Copies files and deletes them on source directory.
MTAPE	device name /INI*TIALIZE/DENS*ITY=d/BLOCK*SIZE=b/DIS*MOUNT Initialize and mount a GOOSY tape
NEWMOD	* /SINCE=<date> /HELP Outputs a list of all new help modules
NWCOPY	node source dest Copy one or more files to one or more nodes

NWDCL	node dcl_line output Execute a single DCL command line on one or more nodes.
NWDEFINE	user paszwort Define logical names for DECNET functions
NWDIFDIR	node file_spec Compare a directory on a remote node with the local directory and create a list of differences.
NWDIRECT	node dir_spec dir_qual Compares directories on different nodes
NWLIBRARY	node library file qual Perform a library operation on one or more nodes
NWUPDATE	node file_list /DESTINATION=d /EXCLUDE=l /LOG/JOURNAL /SINCE=t/BEFORE=t /MODIFIED/CREATED/EXPIRED/BACKUP /REPLACE/OVERLAY/NEW_VERSION /GENERIC Transfer a set of files to one or more nodes using the BACKUP utility.
OPSER	command Execute priviledged operator commands
PFKEY	– Define terminal auxiliary keypad keys.
PLOTMET	metafile type command plotter /COPIES=c /FONT=f Plot a metafile on specified plotter
PRIL	vaxfile switches Sends VAX-file to IBM PRILTMP1.LIST and starts batch job calling PRIL on IBM (VAX-780 only)
PRILS	file /PROP prints VAX sources on the laser printer
PROMPT	prompt-string default /REQUIRED help Prompt input from SYS$COMMAND

RIBM	ibmds vaxfile /COL=/CHECK=/MEMLIST=/$ALL/INTERACTIVE/NOANSWER Send IBM dataset to VAX file.
RRUN	RR file/switches /switches Runs programs. Defaults are updated.
SCRIBM	SCRI vaxfile [profile][edit] Sends VAX-file to IBM SCRITEMP.TEXT and starts batch job calling SCRIPT on IBM
SELECT_MBD	mbd Select a valid MBD controller on a VAX
SETMESSAGE	facility qualifier Control Message output of GOOSY and VMS
SETSYM	symbol value Checks if symbol already exist and outputs message in this case. Sets symbol to value.
SIBM	vaxfile ibmds /MODE= /MAILADDR=/NOANSWER/INTERACTIVE Sends VAX-file(s) to IBM-dataset(s) and overwrites existing datasets or members
SIBMGKS	metafile /PLOTTER=p Output GKS-Metafile on Plotter RP01,RP02
SIBMSPEC	datafile VSAMlib /RUNID=runid /TIME= /NOL*IST Dump GOOSY spectra to SATAN VSAM library
SSYMBOL	SSYM [<name>][/SEARCH=<string>] Displays symbol translation
SWSIZE	proc.-name /W*SMIN=min/S*TATUS=status/R*EPEAT=n Show processes with their workin set sizes.
TDOCUMENT	TDOC*UMENT file /TOC /INDEX /GOOSY /REP*ORT /REL*EASE=r /VER*SION=v /TITLE=tt /AU*THORS=a /LABEL=lt /LOGO=ll Format TeX source, e.g. produced by GLFORMAT of GLDOC, in the GOOSY document style

TLOCK –
Lock terminal by password

VMESTRUC inputfile /PLI/FOR/C/PLIB=/CLIB=/FLIB=
/GLPUT/DELETE
Generate declarations from language independent source.

VMS_Primer Short introduction for VAX users at GSI.
"Common DEC-IBM PLI Standard" by W.F.J.Mueller and H.G.Essel
"GOOSY Conventions (Standards)" by H.G.Essel

WCLOSE file
Wait for file to be closed.

MADDR –
Format addresses from the source INPUT and writes the result to the destination OUTPUT

MADJUST input[,output]/ADJUST/TAB
Adjusts right margins of documentation headers

MANALCH switches MANALCH
Check GOOSY analysis routines.

MANL RUN GOO$EXE:MANL
Attach and execute dynamic lists

MBASE –
Mount/dism GOOSY data base.

MBINHEX binfile
Convert binary files to ASCII HEX format.

MCALLEX /PPL/PLI
Extract calling sequence.

MCALLRTL –
Extracts call-statements out of a GOORTL listing

MCALLSYS –
Extracts calling-statements out of a listing from PLISTRARLET with the system services

MCALLTREE [module-name]
Makes a calling tree out of a cross reference list

MCMD	R GOO$EXE:MCMD Test and demonstration program for the command dispatcher. This programm is used to validate command dispatcher functions after changes and is part of the command dispatcher tuturial
MCOMHLP	input output [/HELP] Reformats a text module generated by MEXTHEA for command description for MFORMDO.
MCTRL	STS=MCTRL Control inactive users
MDBCOPY	– Compress and decompress GOOSY data bases.
MDBM	– Activate different Data Base Management Activities.
MDCLANAL	MDCL*ANAL <command line> Commands to analyze DCL procedures (Call tree) and generate 'debug' versions.
MDCLDEB	MDCLD <key list> Generates debug versions of command procedures.
MDCLLIST	"dsc" p2 p3 .. p8 This procedure provides a mechanism which allows to pass parameters and qualifiers to DCL procedures in the same way as to DCL commands.
MDISP	or by display commands GOOSY display program
MDVICNVV	RUN GOO$EXE:MDVICNV Convert TeX DVI-file to ASCII Hex-code
MEXTHEA	MEXT Extracts documentation blocks generated by MGENHEAD and generates a text module for MFORMDOC
MFIC_CTRL	– Activate different Activities using the command dispatcher.
MFORMDO	MFORM list Generates SCRIPT and RUNOFF files from textmodules created by MEXTHEA for help and documentation

MGENCIM	– Generate card image file to send to IBM
MGENHEA	MGEN language type mode Generates interactively documentation headers for programs, procedures and command procedures.
MGNS_ESONE	RUN GOO$EXE:MGNS_ESONE Network object to perform remote CAMAC accesses.
MGOOWAIT	process Wait for analysis completion
MGTOOL	GTOOL [LIBRARY—DOCUMENT—MESSAGE] Calls GOOSY tools like code management or documentation tools by menu.
MGUIDE_DISP	RUN MGUIDE_DISP Called in guide.com to display one menu
MHEXBIN	binfile Convert ASCII HEX format file created by BINHEX to binary files.
MHLPSCR	from DCL Procedure HLPSCR Generates a file for SCRIPT output from a Help file
MHLPTEX	from DCL Procedure HLPTEX Generates a file for TEX output from a Help file
MINSNUM	MINS Inserts line numbers to PLI source files
MJCLTRIM	file trimmedfile trim JCL output files from the IBM
MLCOUNT	input /LIST /NOHEAD /PPL/PLI/FOR/MAR/COM Looks for longest record and assignes value to DCL symbols MLCOUNT_MAX_LENGTH and MLCOUNT_LINES.
MLIBWILD	<module spec>[/FULL] Reads input as generated by LIB/LIST and outputs list of matching module names.
MLOCKS	– Show VMS locks.

MMESDEF	<facility> Generate PL1 program to generate file for messages linked.
MMESLIST	MMES <facility> List all messages linked.
MMODMAP	switches Formats output from "SEARCH module.lis ENTRY"
MOPER	– Execute privileged operator commands.
MPFKEY	or <PF2> Display of DCL auxiliary keypad definition
MPLOTMET	GOOEXE:MPLOTMET metafile,type,command, queue,copies,font Send a metafile to a plotter.
MPOSTRIBM	RUN GOO$EXE:MPOSTRIBM file columns removes leading numbers, performs necessary character conversions and chops into several files if PDS format
MPRECOM	MPREC file TAGS(tag list) OUT(out file) Precomposer for PLI programs. Extracts marked lines from master source and outputs PLI source
MPREMES switches	MPREMES Concatenate continuation lines
MREMNUM	MREM trunc,check Removes line numbers from PLI source files (<trunc> char.) (<check> char. are checked to be digits)
MREPLACE	input[,output]/PRINT/FORMAT/UPPER Replaces old string by new string. Adjusts Documentation headers by option.
MSECTION	basename [/version] Check if a data base is mounted.
MSHOSYM	from DCL Procedure CSHOWSYM.COM Display the current value of a global symbol (wildcard)
MSHOW	GOOEXE:MSHOW <command> Show commands for use in spawned processes

MSTATUS	SHOW GOOSY STATUS proc1 [proc2 [proc3]] GSTATUS proc1 [proc2 [proc3]] Activate GOOSY status program. Equivalent to SHOW GOOSY STATUS command.
MSYMHLP	MSYM Reads a command procedure and generates help file.
MTMR	RUN GOO$EXE:MTMR DCL process with a subset of the transport manager functions.
MTRIM	switches Remove trailing spaces from source INPUT and writes result to destination OUTPUT
MTXTSORT	MTXTS*ORT input file,[output file] Sorts 2+ blocks in text file alphabetically.
MUAMODI	MUA <action string> <match string> Runs AUTHORIZE and modifies all user accounts (Called in CUAMODI.COM)
MUDIRO	GOOEXE:MUDIRO.EXE sorts a given directory list
MUTIL	– Activate different Activities using the command dispatcher.
MVMECMD	– VME command executor.
MVOICEX	MVOICE Extracts documentation blocks generated by MGENHEAD and generates a text module for U$TALK.
MWV	– inquires a diary, 'Wiedervorlage'

Appendix D

GOOSY Command Key Summary

GOOSY_keywords

Keywords In the following the GOOSY command keywords are listed with their occurance in the commands.

$

$ CLOSE ETHERNET
$ COMMENT
$ DCL
$ DEBUG
$ DEFINE KEY
$ RECALL
$ REPEAT
$ RESET DEFAULT
$ SET DEFAULT
$ SET GNA ETHERNET
$ SHOW COMMAND
$ SHOW GNA COMPONENTS
$ SHOW GNA ETHERNET
$ SHOW GNA LINKS
$ SHOW GNA MCBS
$ SHOW GNA PROCESS
$ SHOW GNA RPC
$ SHOW GNA STATUS
$ SHOW KEY
$ SHOW MEMORY
$ SHOW PROCESS
$ SHOW TIMER

1802

TEST BOR 1802

2047

TEST SEN 2047

2090

TEST SEN 2090

2228

TEST LRS 2228

2249

TEST LRS 2249

2365

LOAD LRS 2365
STORE LRS 2365

2551

TEST LRS 2551

4432

TEST LRS 4432

4434

TEST LRS 4434

ACQUISITION

FOREIGN ACQUISITION
INITIALIZE ACQUISITION
LOAD MODULE ACQUISITION
RESET ACQUISITION
SET ACQUISITION
SHOW ACQUISITION
START ACQUISITION
STOP ACQUISITION

ALIAS

CREATE ALIAS
DELETE ALIAS
SHOW ALIAS

ALLOCATE

ALLOCATE DEVICE

ANALYSIS

ATTACH ANALYSIS
DETACH ANALYSIS
INITIALIZE ANALYSIS
LOAD MODULE ANALYSIS
SET ANALYSIS
SHOW ANALYSIS
START ANALYSIS OUTPUT
START ANALYSIS RANDOM
STOP ANALYSIS OUTPUT
STOP ANALYSIS RANDOM

AREA

CREATE AREA
SHOW AREA

ATTACH

ATTACH ANALYSIS
ATTACH BASE
ATTACH DYNAMIC LIST

ATTACHED

SHOW DYNAMIC ATTACHED

BASE

ATTACH BASE
COMPRESS BASE
CONVERT BASE
COPY BASE
CREATE BASE
DECOMPRESS BASE
DETACH BASE
DETACH BASE
DISMOUNT BASE
LOCATE BASE
MOUNT BASE
UPDATE BASE

BIT

TEST MPI BIT

BITSPECTRUM

CREATE DYNAMIC ENTRY BITSPECTRUM

BLOCK

SHOW HOME BLOCK

BOR

TEST BOR 1802

BUFFER

SET SCATTER BUFFER
SET VME BUFFER
SHOW BUFFER DUMP
SHOW SCATTER BUFFER
START BUFFER DUMP
STOP BUFFER DUMP
TYPE BUFFER

CALCULATE

CALCULATE FASTBUS PEDESTAL
CALCULATE SPECTRUM

CALIBRATE

CALIBRATE SPECTRUM

CALIBRATION

CREATE CALIBRATION FIXED
CREATE CALIBRATION FLOAT
CREATE CALIBRATION LINEAR
DELETE CALIBRATION
DISPLAY CALIBRATION
SET CALIBRATION FIXED
SET CALIBRATION FLOAT
SET CALIBRATION LINEAR
SHOW CALIBRATION

CAMAC

CAMAC CLEAR
CAMAC CNAF
CAMAC DEMAND
CAMAC INHIBIT

CAMAC INITIALIZE
CAMAC SCAN
CLEAR CAMAC SPECTRUM
INITIALIZE CAMAC
READ CAMAC SPECTRUM
RESET CAMAC
SHOW CAMAC SPECTRUM
TEST CAMAC
WRITE CAMAC SPECTRUM

CHANNEL

RELEASE MBD CHANNEL

CLEAR

CAMAC CLEAR
CLEAR CAMAC SPECTRUM
CLEAR CONDITION COUNTER
CLEAR DEVICE
CLEAR ELEMENT
CLEAR PICTURE
CLEAR SPECTRUM

CLOSE

$ CLOSE ETHERNET
CLOSE FILE
CLOSE OUTPUT FILE

CNAF

CAMAC CNAF
CNAF VME

COLOR

SET DEVICE COLOR

COMMAND

$ SHOW COMMAND

COMMENT

$ COMMENT

COMPONENTS

$ SHOW GNA COMPONENTS

COMPOSED

CREATE CONDITION COMPOSED
CREATE DYNAMIC ENTRY COMPOSED

COMPRESS

COMPRESS BASE

CONDITION

CLEAR CONDITION COUNTER
COPY CONDITION
CREATE CONDITION COMPOSED
CREATE CONDITION FUNCTION
CREATE CONDITION MULTIWINDOW
CREATE CONDITION PATTERN
CREATE CONDITION POLYGON
CREATE CONDITION WINDOW
CREATE TABLE CONDITION
DELETE CONDITION
DISPLAY CONDITION
FREEZE CONDITION
MODIFY TABLE CONDITION

REPLACE CONDITION WINDOW
SET CONDITION PATTERN
SET CONDITION WINDOW
SHOW CONDITION
UNFREEZE CONDITION

CONTROL

SET VME CONTROL
SHOW VME CONTROL

CONVERT

CONVERT BASE

COPY

COPY BASE
COPY CONDITION
COPY ELEMENT
COPY FILE
COPY MEMBER
COPY POLYGON
COPY SPECTRUM

COUNTER

CLEAR CONDITION COUNTER

CREATE

CREATE ALIAS
CREATE AREA
CREATE BASE
CREATE CALIBRATION FIXED
CREATE CALIBRATION FLOAT
CREATE CALIBRATION LINEAR
CREATE CONDITION COMPOSED
CREATE CONDITION FUNCTION

CREATE CONDITION MULTIWINDOW
CREATE CONDITION PATTERN
CREATE CONDITION POLYGON
CREATE CONDITION WINDOW
CREATE DIRECTORY
CREATE DYNAMIC ENTRY BITSPECTRUM
CREATE DYNAMIC ENTRY COMPOSED
CREATE DYNAMIC ENTRY FUNCTION
CREATE DYNAMIC ENTRY INDEXEDSPECTRUM
CREATE DYNAMIC ENTRY MULTIWINDOW
CREATE DYNAMIC ENTRY PATTERN
CREATE DYNAMIC ENTRY POLYGON
CREATE DYNAMIC ENTRY PROCEDURE
CREATE DYNAMIC ENTRY SCATTER
CREATE DYNAMIC ENTRY SPECTRUM
CREATE DYNAMIC ENTRY WINDOW
CREATE DYNAMIC LIST
CREATE ELEMENT
CREATE ENVIRONMENT
CREATE LINK
CREATE OVERLAY
CREATE PICTURE
CREATE POLYGON
CREATE POOL
CREATE PROCESS
CREATE PROGRAM
CREATE SESSION
CREATE SPECTRUM
CREATE TABLE CONDITION
CREATE TABLE SPECTRUM
CREATE TYPE

DATA

SEND DATA

DCL

$ DCL

DEALLOCATE

DEALLOCATE DEVICE

DEBUG

$ DEBUG
DEBUG VME MEMORY

DECALIBRATE

DECALIBRATE SPECTRUM

DECOMPRESS

DECOMPRESS BASE

DEFAULT

$ RESET DEFAULT
$ SET DEFAULT

DEFINE

$ DEFINE KEY
DEFINE DISPLAY HEADER
DEFINE DISPLAY PICTURE
DEFINE DISPLAY SPECTRUM
DEFINE FRAME SETUP
DEFINE PICTURE SETUP

DELETE

DELETE ALIAS
DELETE CALIBRATION
DELETE CONDITION

DELETE DYNAMIC ENTRY
DELETE DYNAMIC LIST
DELETE ELEMENT
DELETE ENVIRONMENT
DELETE LINK
DELETE OVERLAY
DELETE PICTURE
DELETE POLYGON
DELETE POOL
DELETE PROCESS
DELETE SECTION
DELETE SPECTRUM

DEMAND

CAMAC DEMAND

DETACH

DETACH ANALYSIS
DETACH BASE
DETACH BASE
DETACH DISPLAY
DETACH DYNAMIC LIST

DEVICE

ALLOCATE DEVICE
CLEAR DEVICE
DEALLOCATE DEVICE
SET DEVICE COLOR

DEVICES

SHOW DEVICES

DIRECTORY

CREATE DIRECTORY

LOCATE DIRECTORY
MODIFY DIRECTORY
SHOW DIRECTORY

DISMOUNT

DISMOUNT BASE
DISMOUNT TAPE

DISPLAY

DEFINE DISPLAY HEADER
DEFINE DISPLAY PICTURE
DEFINE DISPLAY SPECTRUM
DETACH DISPLAY
DISPLAY CALIBRATION
DISPLAY CONDITION
DISPLAY GRAPH
DISPLAY METAFILE
DISPLAY PICTURE
DISPLAY POINT
DISPLAY POLYGON
DISPLAY SCATTER
DISPLAY SPECTRUM
DISPLAY TEXT
SAVE DISPLAY
SET DISPLAY MODE
SHOW DISPLAY GLOBALS

DUMP

DUMP MBD
DUMP SPECTRUM
DUMP STARBURST
SHOW BUFFER DUMP
START BUFFER DUMP
STOP BUFFER DUMP

DYNAMIC

ATTACH DYNAMIC LIST
CREATE DYNAMIC ENTRY BITSPECTRUM
CREATE DYNAMIC ENTRY COMPOSED
CREATE DYNAMIC ENTRY FUNCTION
CREATE DYNAMIC ENTRY INDEXEDSPECTRUM
CREATE DYNAMIC ENTRY MULTIWINDOW
CREATE DYNAMIC ENTRY PATTERN
CREATE DYNAMIC ENTRY POLYGON
CREATE DYNAMIC ENTRY PROCEDURE
CREATE DYNAMIC ENTRY SCATTER
CREATE DYNAMIC ENTRY SPECTRUM
CREATE DYNAMIC ENTRY WINDOW
CREATE DYNAMIC LIST
DELETE DYNAMIC ENTRY
DELETE DYNAMIC LIST
DETACH DYNAMIC LIST
SET DYNAMIC LIST
SHOW DYNAMIC ATTACHED
SHOW DYNAMIC LIST
START DYNAMIC LIST
STOP DYNAMIC LIST
UPDATE DYNAMIC LIST

ELEMENT

CLEAR ELEMENT
COPY ELEMENT
CREATE ELEMENT
DELETE ELEMENT
LOCATE ELEMENT
SHOW ELEMENT

ENTRY

CREATE DYNAMIC ENTRY BITSPECTRUM
CREATE DYNAMIC ENTRY COMPOSED
CREATE DYNAMIC ENTRY FUNCTION
CREATE DYNAMIC ENTRY INDEXEDSPECTRUM
CREATE DYNAMIC ENTRY MULTIWINDOW
CREATE DYNAMIC ENTRY PATTERN
CREATE DYNAMIC ENTRY POLYGON

CREATE DYNAMIC ENTRY PROCEDURE
CREATE DYNAMIC ENTRY SCATTER
CREATE DYNAMIC ENTRY SPECTRUM
CREATE DYNAMIC ENTRY WINDOW
DELETE DYNAMIC ENTRY

ENVIRONMENT

CREATE ENVIRONMENT
DELETE ENVIRONMENT

ETHERNET

$ CLOSE ETHERNET
$ SET GNA ETHERNET
$ SHOW GNA ETHERNET

EVENT

SET EVENT INPUT
SET EVENT OUTPUT
TYPE EVENT

EXECUTE

EXECUTE VME

EXPAND

EXPAND

FASTBUS

CALCULATE FASTBUS PEDESTAL
SET FASTBUS PEDESTAL

FILE

CLOSE FILE
CLOSE OUTPUT FILE
COPY FILE
OPEN FILE
OPEN OUTPUT FILE
START INPUT FILE
START OUTPUT FILE
STOP INPUT FILE
STOP OUTPUT FILE
TYPE FILE

FIT

FIT SPECTRUM

FIXED

CREATE CALIBRATION FIXED
SET CALIBRATION FIXED

FLOAT

CREATE CALIBRATION FLOAT
SET CALIBRATION FLOAT

FOREIGN

FOREIGN ACQUISITION

FRAME

DEFINE FRAME SETUP
MODIFY FRAME SCATTER
MODIFY FRAME SPECTRUM
ZOOM FRAME

FRAMES

UPDATE FRAMES

FREEZE

FREEZE CONDITION
FREEZE SPECTRUM

FUNCTION

CREATE CONDITION FUNCTION
CREATE DYNAMIC ENTRY FUNCTION

GLOBALS

SHOW DISPLAY GLOBALS

GNA

$ SET GNA ETHERNET
$ SHOW GNA COMPONENTS
$ SHOW GNA ETHERNET
$ SHOW GNA LINKS
$ SHOW GNA MCBS
$ SHOW GNA PROCESS
$ SHOW GNA RPC
$ SHOW GNA STATUS

GOOSY

SHOW GOOSY STATUS

GRAPH

DISPLAY GRAPH

GSI

TEST GSI IOL

HEADER

DEFINE DISPLAY HEADER

HOME

SHOW HOME BLOCK

HVR

ID

LOCATE ID

INDEXEDSPECTRUM

CREATE DYNAMIC ENTRY INDEXEDSPECTRUM

INHIBIT

CAMAC INHIBIT

INITIALIZE

CAMAC INITIALIZE
INITIALIZE ACQUISITION
INITIALIZE ANALYSIS
INITIALIZE CAMAC

INPUT

SET EVENT INPUT
SET VME INPUT
START INPUT FILE
START INPUT MAILBOX

START INPUT NET
STOP INPUT FILE
STOP INPUT MAILBOX
STOP INPUT NET

INTEGRATE

INTEGRATE

IOL

TEST GSI IOL

J11

LOAD J11

KEY

$ DEFINE KEY
$ SHOW KEY

KEYPAD

SHOW TP0 KEYPAD

LETTERING

SET LETTERING

LINEAR

CREATE CALIBRATION LINEAR
SET CALIBRATION LINEAR

LINK

CREATE LINK
DELETE LINK
SHOW LINK

LINKS

$ SHOW GNA LINKS

LIST

ATTACH DYNAMIC LIST
CREATE DYNAMIC LIST
DELETE DYNAMIC LIST
DETACH DYNAMIC LIST
SET DYNAMIC LIST
SHOW DYNAMIC LIST
START DYNAMIC LIST
STOP DYNAMIC LIST
UPDATE DYNAMIC LIST

LOAD

LOAD J11
LOAD LRS 2365
LOAD MBD
LOAD MODULE ACQUISITION
LOAD MODULE ANALYSIS
LOAD STARBURST
LOAD VME PROGRAM
LOAD VME TABLE

LOCATE

LOCATE BASE
LOCATE DIRECTORY
LOCATE ELEMENT
LOCATE ID
LOCATE POOL
LOCATE QUEUEELEMENT

LOCATE TYPE

LOCK

SET LOCK OUTPUT

LOCKS

SHOW LOCKS

LRS

LOAD LRS 2365
STORE LRS 2365
TEST LRS 2228
TEST LRS 2249
TEST LRS 2551
TEST LRS 4432
TEST LRS 4434

MAILBOX

START INPUT MAILBOX
STOP INPUT MAILBOX

MAPPING

SHOW MAPPING
SHOW MAPPING
SHOW MAPPING

MBD

DUMP MBD
LOAD MBD
PATCH MBD
RELEASE MBD CHANNEL
RESET MBD

STORE MBD

MCBS

$ SHOW GNA MCBS

MEMBER

COPY MEMBER
SET MEMBER
SHOW MEMBER

MEMORY

$ SHOW MEMORY
DEBUG VME MEMORY

MESSAGE

METAFILE

DISPLAY METAFILE
PLOT METAFILE

MODE

SET DISPLAY MODE

MODIFY

MODIFY DIRECTORY
MODIFY FRAME SCATTER
MODIFY FRAME SPECTRUM
MODIFY TABLE CONDITION
MODIFY TABLE SPECTRUM

MODULE

LOAD MODULE ACQUISITION
LOAD MODULE ANALYSIS

MOUNT

MOUNT BASE
MOUNT TAPE

MPI

TEST MPI BIT
TEST MPI TDC

MR2000

START MR2000
STOP MR2000

MULTIWINDOW

CREATE CONDITION MULTIWINDOW
CREATE DYNAMIC ENTRY MULTIWINDOW

MWPC

SET MWPC

NET

START INPUT NET
STOP INPUT NET

OPEN

OPEN FILE
OPEN OUTPUT FILE

OUTPUT

CLOSE OUTPUT FILE
OPEN OUTPUT FILE
SET EVENT OUTPUT
SET LOCK OUTPUT
START ANALYSIS OUTPUT
START OUTPUT FILE
STOP ANALYSIS OUTPUT
STOP OUTPUT FILE

OVERLAY

CREATE OVERLAY
DELETE OVERLAY
OVERLAY

PATCH

PATCH MBD
PATCH STARBURST

PATTERN

CREATE CONDITION PATTERN
CREATE DYNAMIC ENTRY PATTERN
SET CONDITION PATTERN

PEDESTAL

CALCULATE FASTBUS PEDESTAL
SET FASTBUS PEDESTAL

PICTURE

CLEAR PICTURE

CREATE PICTURE
DEFINE DISPLAY PICTURE
DEFINE PICTURE SETUP
DELETE PICTURE
DISPLAY PICTURE
PLOT PICTURE
SHOW PICTURE

PLOT

PLOT METAFILE
PLOT PICTURE
PLOT PLOTFILE

PLOTFILE

PLOT PLOTFILE

POINT

DISPLAY POINT
SET SPECTRUM POINT

POLYGON

COPY POLYGON
CREATE CONDITION POLYGON
CREATE DYNAMIC ENTRY POLYGON
CREATE POLYGON
DELETE POLYGON
DISPLAY POLYGON
REPLACE POLYGON
SHOW POLYGON

POOL

CREATE POOL
DELETE POOL
LOCATE POOL

SHOW POOL

PRINT

PRINT

PROCEDURE

CREATE DYNAMIC ENTRY PROCEDURE

PROCESS

$ SHOW GNA PROCESS
$ SHOW PROCESS
CREATE PROCESS
DELETE PROCESS

PROGRAM

CREATE PROGRAM
LOAD VME PROGRAM

PROJECT

PROJECT

PROTECT

PROTECT SPECTRUM

PROTOCOL

PROTOCOL

QUEUEELEMENT

LOCATE QUEUEELEMENT

RANDOM

SET RANDOM
START ANALYSIS RANDOM
STOP ANALYSIS RANDOM

READ

READ CAMAC SPECTRUM

RECALL

$ RECALL

REFRESH

REFRESH

REGISTER

TEST REGISTER

RELEASE

RELEASE MBD CHANNEL

REPEAT

$ REPEAT

REPLACE

REPLACE CONDITION WINDOW
REPLACE POLYGON

RESET

$ RESET DEFAULT
RESET ACQUISITION
RESET CAMAC
RESET MBD

RPC

$ SHOW GNA RPC

RUN

START RUN
STOP RUN

SAVE

SAVE DISPLAY

SCAN

CAMAC SCAN

SCATTER

CREATE DYNAMIC ENTRY SCATTER
DISPLAY SCATTER
MODIFY FRAME SCATTER
SET SCATTER BUFFER
SHOW SCATTER BUFFER
START SCATTER
STOP SCATTER

SECTION

DELETE SECTION

SEN

TEST SEN 2047
TEST SEN 2090

SEND

SEND DATA

SESSION

CREATE SESSION

SET

$ SET DEFAULT
$ SET GNA ETHERNET
SET ACQUISITION
SET ANALYSIS
SET CALIBRATION FIXED
SET CALIBRATION FLOAT
SET CALIBRATION LINEAR
SET CONDITION PATTERN
SET CONDITION WINDOW
SET DEVICE COLOR
SET DISPLAY MODE
SET DYNAMIC LIST
SET EVENT INPUT
SET EVENT OUTPUT
SET FASTBUS PEDESTAL
SET LETTERING
SET LOCK OUTPUT
SET MEMBER
SET MWPC
SET RANDOM
SET SCATTER BUFFER
SET SPECTRUM POINT

SET VME BUFFER
SET VME CONTROL
SET VME INPUT
SET VME TRIGGER

SETUP

DEFINE FRAME SETUP
DEFINE PICTURE SETUP
SHOW VME SETUP

SHOW

$ SHOW COMMAND
$ SHOW GNA COMPONENTS
$ SHOW GNA ETHERNET
$ SHOW GNA LINKS
$ SHOW GNA MCBS
$ SHOW GNA PROCESS
$ SHOW GNA RPC
$ SHOW GNA STATUS
$ SHOW KEY
$ SHOW MEMORY
$ SHOW PROCESS
$ SHOW TIMER
SHOW ACQUISITION
SHOW ALIAS
SHOW ANALYSIS
SHOW AREA
SHOW BUFFER DUMP
SHOW CALIBRATION
SHOW CAMAC SPECTRUM
SHOW CONDITION
SHOW DEVICES
SHOW DIRECTORY
SHOW DISPLAY GLOBALS
SHOW DYNAMIC ATTACHED
SHOW DYNAMIC LIST
SHOW ELEMENT
SHOW GOOSY STATUS
SHOW HOME BLOCK

SHOW LINK
SHOW LOCKS
SHOW MAPPING
SHOW MAPPING
SHOW MAPPING
SHOW MEMBER
SHOW PICTURE
SHOW POLYGON
SHOW POOL
SHOW SCATTER BUFFER
SHOW SPECTRUM
SHOW STARBURST
SHOW TABLE
SHOW TP0 KEYPAD
SHOW TREE
SHOW TYPE
SHOW VME CONTROL
SHOW VME SETUP

SLEEP

SLEEP

SPECTRUM

CALCULATE SPECTRUM
CALIBRATE SPECTRUM
CLEAR CAMAC SPECTRUM
CLEAR SPECTRUM
COPY SPECTRUM
CREATE DYNAMIC ENTRY SPECTRUM
CREATE SPECTRUM
CREATE TABLE SPECTRUM
DECALIBRATE SPECTRUM
DEFINE DISPLAY SPECTRUM
DELETE SPECTRUM
DISPLAY SPECTRUM
DUMP SPECTRUM
FIT SPECTRUM
FREEZE SPECTRUM
MODIFY FRAME SPECTRUM

MODIFY TABLE SPECTRUM
PROTECT SPECTRUM
READ CAMAC SPECTRUM
SET SPECTRUM POINT
SHOW CAMAC SPECTRUM
SHOW SPECTRUM
SUMUP SPECTRUM
UNFREEZE SPECTRUM
UNPROTECT SPECTRUM
WRITE CAMAC SPECTRUM

STARBURST

DUMP STARBURST
LOAD STARBURST
PATCH STARBURST
SHOW STARBURST

START

START ACQUISITION
START ANALYSIS OUTPUT
START ANALYSIS RANDOM
START BUFFER DUMP
START DYNAMIC LIST
START INPUT FILE
START INPUT MAILBOX
START INPUT NET
START MR2000
START OUTPUT FILE
START RUN
START SCATTER
START VME

STATUS

$ SHOW GNA STATUS
SHOW GOOSY STATUS

STOP

STOP ACQUISITION
STOP ANALYSIS OUTPUT
STOP ANALYSIS RANDOM
STOP BUFFER DUMP
STOP DYNAMIC LIST
STOP INPUT FILE
STOP INPUT MAILBOX
STOP INPUT NET
STOP MR2000
STOP OUTPUT FILE
STOP RUN
STOP SCATTER
STOP VME

STORE

STORE LRS 2365
STORE MBD

SUMUP

SUMUP SPECTRUM

TABLE

CREATE TABLE CONDITION
CREATE TABLE SPECTRUM
LOAD VME TABLE
MODIFY TABLE CONDITION
MODIFY TABLE SPECTRUM
SHOW TABLE

TAPE

DISMOUNT TAPE

MOUNT TAPE

TDC

TEST MPI TDC

TEST

TEST BOR 1802
TEST CAMAC
TEST GSI IOL
TEST LRS 2228
TEST LRS 2249
TEST LRS 2551
TEST LRS 4432
TEST LRS 4434
TEST MPI BIT
TEST MPI TDC
TEST REGISTER
TEST SEN 2047
TEST SEN 2090

TEXT

DISPLAY TEXT

TIMER

$ SHOW TIMER

TP0

SHOW TP0 KEYPAD

TREE

SHOW TREE

TRIGGER

SET VME TRIGGER

TYPE

CREATE TYPE
LOCATE TYPE
SHOW TYPE
TYPE BUFFER
TYPE EVENT
TYPE FILE

UNFREEZE

UNFREEZE CONDITION
UNFREEZE SPECTRUM

UNPROTECT

UNPROTECT SPECTRUM

UPDATE

UPDATE BASE
UPDATE DYNAMIC LIST
UPDATE FRAMES

VME

CNAF VME
DEBUG VME MEMORY
EXECUTE VME
LOAD VME PROGRAM
LOAD VME TABLE
SET VME BUFFER

SET VME CONTROL
SET VME INPUT
SET VME TRIGGER
SHOW VME CONTROL
SHOW VME SETUP
START VME
STOP VME

VOICE

WAIT

WAIT

WINDOW

CREATE CONDITION WINDOW
CREATE DYNAMIC ENTRY WINDOW
REPLACE CONDITION WINDOW
SET CONDITION WINDOW

WRITE

WRITE CAMAC SPECTRUM

ZOOM

ZOOM FRAME

Appendix E

GOOSY Glossary

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 Microprogrammed **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