GOOSY
Id.: ANAL
Version: 1.0
Date: 14–Jun–1988
Revised: May, 20 1992

GSI Online Offline SYstem

GOOSY Data Acquisition

H.G.Essel

May, 20 1992

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

List of Figures

2.1 Transport Manager I/O Channels 8

C.1 Message Control Block Structure 230
C.2 GOOSY File Structure 233
C.3 Buffer Header Structure 234
C.4 Buffer Element Header Structure 236
C.5 File Header Structure 241
C.6 Event structure type 3 (compressed) 245
C.7 Event structure type 4, subtype 1 (block) 246
C.8 Event structure type 4, subtype 2 (no zeros's) 246
C.9 Event structure type 1 (SILENA) 247
C.10 Event structure type 5 (LRS FERA) 249
C.11 Event structure type 6 (MBD buffer type 6) 250
C.12 Event structure type 7 (MBD buffer type 7) 252
C.13 Event Structure 253
C.14 CAMAC Subevent Structure 253
C.15 Fastbus Subevent Structure 254
C.16 Fastbus Module header 254
C.17 Fastbus Data Word 255
C.18 Buffer Element structure type 9000, Time Stamp 256
C.19 Data structure SILENA ADC 257
C.20 Data structure LRS FERA 258

Chapter 1

Preface

GOOSY Copy Right

Conventions used in this Document

$\boxed{\texttt{Fn}}$, $\boxed{\texttt{PFn}}$, $\boxed{\texttt{1}}$, $\boxed{\texttt{Do}}$, **or** $\boxed{\texttt{Return}}$ **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.

$\boxed{\texttt{GOLD}}$, **<GOLD>**— The $\boxed{\texttt{PF1}}$ key is called $\boxed{\texttt{GOLD}}$ 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 $\boxed{\texttt{PFn}}$ of the rightmost VTx20 compatible keypad are named $\boxed{\texttt{Fn}}$ and the 20 $\boxed{\texttt{Fn}}$ keys on the top of each VTx20 compatible keyboard are named $\boxed{\texttt{PFn}}$ on a PERICOM.

$\boxed{\texttt{Return}}$— The $\boxed{\texttt{Return}}$ is not shown in formats and examples. Assume that you must press $\boxed{\texttt{Return}}$ after typing a command or other input to the system unless instructed otherwise.

$\boxed{\texttt{Enter}}$— If your terminal is connected to IBM, the $\boxed{\texttt{Enter}}$ key terminates all command lines.

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

- $\boxed{\texttt{Ctrl}}$ `Z` means hold down the $\boxed{\texttt{Ctrl}}$ key and type the letter `Z`.

$\boxed{\texttt{PFn}}$ **key** — The $\boxed{\texttt{PFn}}$ followed by a number means that you must press the $\boxed{\texttt{PFn}}$ key and *then* type the number. Here is an example:

- $\boxed{\texttt{PF1}}$ **6** press the $\boxed{\texttt{PF1}}$ key and then type the number 6 on the main keyboard.

$\boxed{\texttt{PFn}}$ **or** $\boxed{\texttt{Fn}}$ **keys** — Any $\boxed{\texttt{PFn}}$ or $\boxed{\texttt{Fn}}$ key means that you just press this key. Here is an example:

- $\boxed{\texttt{PF2}}$ means press the $\boxed{\texttt{PF2}}$ 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

Chapter 2

GOOSY Transport Manager

2.1 Introduction

The Transport Manager program (TMR) is a GOOSY component and is created in a GOOSY environment. In terms of VMS it runs in a subprocess. The TMR executes several commands concerning the data acquisition and data dispatch, and the control of CAMAC equipment. Presently it supports the VME frontend system, the MBD and the J11 single crate system. The event data are collected by the frontend processors in formatted data buffers which are input to the TMR and dispatched to several output channels. The TMR supports presently seven types of input channels and three types of output channels. Five of the input channels are exclusive. Only one exclusive input channel can be activated at the same time but several output channels. A more detailed description follows in the next sections.

Input Channels are:

1. VME exclusive
   Data buffers are read from VME-subsystem. The event format is 10,n.
2. MBD exclusive
   Data buffers are read from MBD. The event format depends on the J11 programs.
3. J11 exclusive
   Data buffers are read from J11 via DECnet. A standard buffer and event format is generated by the J11.
4. File exclusive
   Data buffers are read from a disk or tape file (Standard RMS format).
5. Foreign exclusive
   This input channel requires some programming work to support other frontend systems.

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

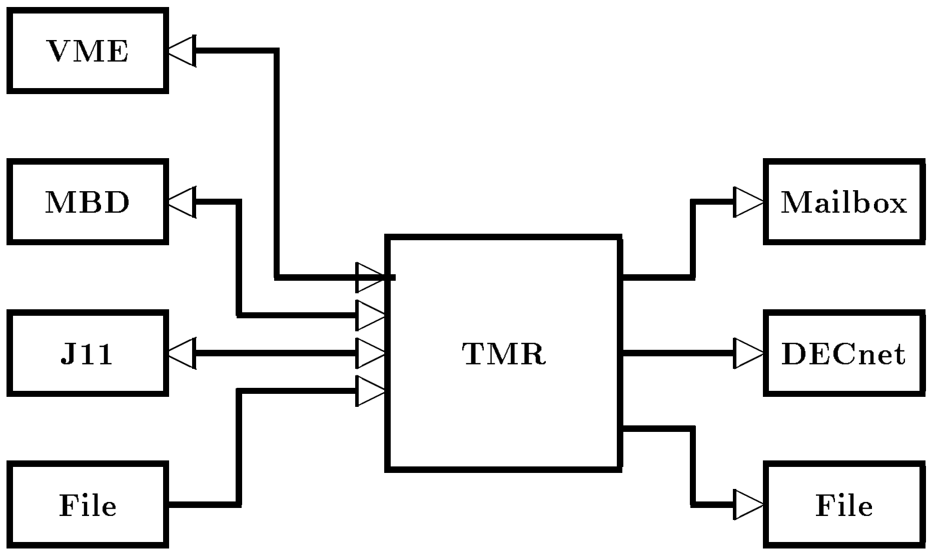

Figure 2.1: The input and output channels of the Transport Manager

6. Mailbox
   At any time a GOOSY buffer may be sent to this channel. It is processed like other buffers.
7. DECnet
   At any time a GOOSY buffer may be sent to this channel. It is processed like other buffers.

The exclusive input channel is selected by the `INITIALIZE ACQUISITION` command. The mailbox and DECnet channels are opened by the sending programs.

Output Channels are:

1. File
   Data buffers are written to a disk or tape file (standard RMS format). This channel

is opened and closed by commands. It always synchronizes the input.

2. Mailbox
   Data buffers are written to mailboxes. There are three mailbox channels. They are filled if they have been read by some program, normally an analysis program. The first mailbox channel optionally synchronizes the input.
3. DECnet
   Data buffers are written to DECnet. Up to 20 DECnet links can be established. The TMR sends buffers to all programs having established a link and having acknowledged the previous buffer. These channels optionally synchronize the input.

Parallel to the data stream dispatching the TMR executes commands controlling the experimental setup, i.e. downline loading programs to frontend processors or executing CAMAC commands. Optionally the data buffer structure is checked and contents may be displayed to the terminal.

2.2 Startup the TMR

The TMR must be started in a GOOSY environment. This can be done by

```
$ CRENVIR environment /$TMR
```

or if the environment exists already :

```
$ GOOSY
GOOSY> CREATE PROCESS TMR $TMR
GOOSY> DELETE PROCESS $TMR
```

The process is started by default with priority 3. Specify another priority by `CREATE PROCESS ... PRIO=p`. The second command deletes the TMR. This does not affect other components of the environment, except that analysis programs cannot get data any more and DECnet links are aborted. It is, however, a good praxis to STOP the acquisition before deleting the TMR component. If the TMR component is created, the TMR must be initialized. At this point one must know about the data input. Therefore the data input is described in the next section.

2.3 Input Channels

2.3.1 MBD Input

Before you can access an MBD, you must specify which MBD you want to use. This is done by the DCL command `SELECT_MBD`. On the cluster VAXes DONALD and EMMA there are two MBD's labeled 'A' and 'B'. The branch cables are labeled as 'DA' and 'DB' or 'EA' and 'EB', respectively.

INITIALIZE ACQUISITION

Using the MBD the TMR must be initialized by

```
   GOOSY> INITIALIZE ACQUISITION mailbox size /MBD
```

'Mailbox' is a name used for the creation of the mailboxes. If not specified, the name of the environment is used (this is recommended). The mailbox names are then

```
   GOOSY_mailbox_1, GOOSY_mailbox_2, GOOSY_mailbox_3
```

'Size' specifies the buffer size in bytes. The default size is 8192. The minimum size is 512 bytes (for file header). A multiple of 512 should be used. A channel is assigned to device 'MBD', the internal buffer queues are created and the mailboxes are created.

LOAD MBD

For a description of the MBD and J11 hardware and programming see the GOOSY Hardware Manual. The MBD code must be loaded to the MBD, the J11 code to the J11's (one per crate). This is done for a two crate system by

```
   GOOSY> LOAD MBD GOO$IO:EXEC/EXEC
   GOOSY> LOAD MBD GOO$IO:ESONE
   GOOSY> LOAD MBD GOO$IO:C2             ! for 2 crates
   GOOSY> LOAD STARBURST file1 1 23/BOOT  ! boot crate one
   GOOSY> LOAD STARBURST file2 2 23/BOOT  ! boot crate two
```

The MBD code is in file GOO$IO:EXEC.BDO, the ESONE code in GOO$IO:ESONE.BDO, standard MBD programs for n crates are GOO$IO:Cn. The J11 crate controller programs are loaded by the last command for each crate. For the programming of the J11's see GOOSY Hardware Manual. The CAMAC station number of the J11 is always 23. Now the MBD should be ready to send data.
If the MBD could not be loaded because of an internal loop or any MBD hangup you may try to reset the MBD by the TMR command

```
   GOOSY> RESET MBD
```

Be shure what you are doing when loading or resetting an MBD and check that you have selected the right MBD on the right VAX. Otherwise you may destroy running experiments.

2.3.2 VME Input

Before one can proceed with the following commands, the VME-hardware must be set up properly. This is described in the hardware manual. The communication processor (E5 or E6) must run the program `net_slave`. On the VAX some definitions have to be done by DCL command ETHDEF:

```
$ ETHDEF processor device
```

where 'processor' is the name of the E5, i.e. E5ELXA and 'device' is the bus where the ethernet interface is plugged in (QB, UB, BI or WS). This command also defines the two logical names EB_EXEC and FEP_EXEC to the standard FIC software, when they are not yet defined. These names can be used in the setup files (see below).

VME setup files

The next things needed are the text files to describe the VME processors and the readout tables. Examples can be found on directory GOO$EXAMPLES (file types .VMEP and .VMET). The syntax is described with the LOAD VME commands. The equipment controlled by one frontend processor is called a branch. The branch is specified by the VME crate number and the memory offset of the processor indicated by LEDs on the frontpanel. This offset can be set by a switch on the platine. For flexibility an ID number (between 1 and 16k) can be defined by the user for these two numbers. This is done together with the processor specification in the setup file. The ID is written into the subevent header of the branch. Therefore programs may be sensitive to that ID, but the processor offsets may be changed independently. Normally there is one top file specifying only the processors and their ID's. In addition in that file the files specifying the readout should be invoked by @file. This allows to keep the files modular. It is recommended to keep the readout tables for each processor in separate files. The top file can be given to the LOAD VME PROGRAM or TABLE commands described later. The DCL command `VMETREE` displays the whole tree of a top file.

INITIALIZE ACQUISITION

Using the VME frontend the TMR must be initialized by

```
GOOSY> INITIALIZE ACQUISITION mailbox size /VME
```

'Mailbox' is a name used for the creation of the mailboxes. If not specified, the name of the environment is used (this is recommended). 'Size' specifies the buffer size in bytes. The default is 16 kB. The frontend system must respond with "Link open acknowledged". Unlike with the other frontends, the command can be repeated, i.e. if the link to the frontends has been broken.

When the communication processor (E5 or E6) had to be restarted, the transport manager must first close the link by

```
   GOOSY> $ CLOSE ETHERNET
```

When the communication processor does not respond, one has to wait for timeout. Then the `INIT AC /VME` command can be executed.

LOAD VME frontend processors

Now the programs must be loaded to the VME processors. This is done by

```
   GOOSY> LOAD VME PROGRAM @file /TABLE
```

The file must contain a list describing the processors in the VME crate. Default file type is .VMEP. An example can be found on GOO$EXAMPLES:VME_SETUP.VMEP. With the optional qualifier `/TABLE` the readout tables are loaded also (see below). This makes sense only, if the readout files are called in the specified file with the @file option. Example of output (1 CAMAC crate):

```
SUC: GOOSY> LOAD VME PROG @ISETUPC_E6
Load programs from EE$ROOT:[SCHALL.GOOSY]ISETUPC_E6.VMEP;
-> ISETUPC_E6
--> ICAM1.VMET;11
 Crate=1 Processor=4 ID=100 Index=0 Subcrate=0 Control=1 EB EB EB_EXEC
 Crate=1 Processor=5 ID=10 Index=1 Subcrate=0 Control=3 FEP CAV FEP_EXEC
Load system file VME$EXE:EB_ROOT.EX20
Load system file VME$EXE:FEP_ROOT.EX20
 Message from crate 1, offset 4, subcrate 0, control 1.
 > EB GEB 3.0 with 1[512Kb] started, 15 buffers [16 Kb]. FEPs: 1 sync. 0 async.
 Message from crate 1, offset 5, subcrate 0, control 3.
 > FEP GFP 3.0 with 2[512Kb] started. 32 buffers [24026 b] Type=3 Number=1.
```

Once loaded the VME system runs independently of the Transport Manager. When the TMR has to be restarted for some reason, the VME system need not to be loaded again. The TMR must know, however, the VME setup. To achieve that, type

```
SUC: GOOSY> LOAD VME PROG @ISETUPC_E6 /NOLOAD
```

There comes a lot of output which can be ignored.

LOAD VME readout tables

Now the readout tables must be loaded to the VME processors. This is done by

```
   GOOSY> LOAD VME TABLE @file trigger
```

The file is normally the same as above. It contains lines @file referencing file which contain a list describing the modules in the CAMAC and Fastbus crates. A trigger number may be specified between 1 and 15. Default is 1. The lists for different triggers cannot be reloaded separately. If one already loaded list is reloaded, the lists for all other triggers must be reloaded too. Example (1 CAMAC crate):

```
SUC: GOOSY> LOAD VME TABLE @ISETUPC_E6
Load tables from   EE$ROOT:[SCHALL.GOOSY]ISETUPC_E6.VMEP;
-> EE$ROOT:[SCHALL.GOOSY]ISETUPC_E6.VMEP;
--> ICAM1.VMET;11
 Message from crate 1, offset 6, subcrate 0, control 3.
 > Readout tables for all triggers reset!
 Message from crate 1, offset 6, subcrate 0, control 3.
 > Trigger  1: Readout for Crate 1, Length 1 loaded.
```

Examples of reaout tables can be found on GOO$EXAMPLES:*.VMET.

SHOW VME SETUP

When the programs and tables are loaded, the command

```
   GOOSY> SHOW VME SETUP
```

displays a list of processors and loaded files. Example:

```
SUC: GOOSY> SHO VME SETUP
------------------------------------------------------------------------
Setup file  : EE$ROOT:[SCHALL.GOOSY]ISETUPC_E6.VMEP
Number FEPs: 1 synchron 0 asynchron
-#---id---cr-pr-sc-cl-mode-cl--ty---------system program----------------
  0   100  1  4  0  1 sync EB  EB  VME$ROOT:[VMEMAN.VMELIB.EXE]EB_ROOT
  1    10  1  5  0  3 sync CAV FEP VME$ROOT:[VMEMAN.VMELIB.EXE]FEP_ROOT
------------------------------------------------------------------------
    Trigger   1 EE$ROOT:[SCHALL.GOOSY]ISETUPC_E6.VMEP;
------------------------------------------------------------------------
SUC: GOOSY>
```

Select Data Transport

There are two ways to get the data from the VME system into the VAX. The faster one is to use a parallel interface (HVR). The second is to use ethernet. The GOOSY event builer must be setup to know which channel to use. This is done by command

```
   GOOSY> SET VME INPUT /NET/HVR/OFF
```

Of cause, only one of the options can be selected. When `/NET` or `/OFF` is selected, you must use the `/NET` qualifier with `START ACQUISITION`. When you select the NET option, the ethernet communication must be set to stream mode (not yet implemented):

```
GOOSY> $ SET GNA ETHERNET /ISTREAM
```

Commands

All other commands given to the VME frontend system have to specify the destination processor(s). This can be done in different ways:

- Specify the processor id with ID=number. This number is defined in your setup files. Only one destination is possible.
- Specify a list of VME-crate/offset pairs of the processors by PROCESSOR=list
- Select processors by type, i.e. /EB or /FEP or /ALL.
- Select processors by type of readout, i.e. /CVI or /CAV or /AEB.

Setup Readout buffers

The frontend processors provide per default 32 subevent buffers of fixed length. The length is calculated from the memory available. The number of buffers or the buffer size may be changed by command

```
GOOSY> SET VME BUFFER BUFFERS=b SIZE=s ...
```

Only one, number of buffers or size [Bytes], can be specified. The other is calculated from the memory available. There is, however, a maximum number of buffers. The number of buffers need not be the same on all processors. The size must be the maximum subevent size. The command answers with a message telling the buffer setup. Note that you must specify the destination in the way described above.

SHOW VME CONTROL

When the programs and tables are loaded, the command

```
GOOSY> SHOW VME CONTROL ...
```

displays buffers, events, and tables loaded by the FEP's. Example:

```
  GOOSY> SHOW VME CONTROL /ALL
Message from crate 1, offset 5, subcrate 0, control 1.
> Bufind(FEP)=0 Events=0 Reject=0 Buffers=0
Message from crate 1, offset 6, subcrate 0, control 3.
> Bufferindex=-1 Events=0 Reject=0 Buffers=0
```

```
 Message from crate 1, offset 6, subcrate 0, control 3.
 > INIT list : Crate 1, Address 20012990, Length 4
 Message from crate 1, offset 6, subcrate 0, control 3.
 > Trigger  1: Crate 1, Address 2002A998, Length 1
 Message from crate 1, offset 6, subcrate 0, control 3.
 > Trigger  2: Crate 1, Address 2002A99C, Length 2
```

SET Trigger

The trigger modules are set by command

```
   GOOSY> SET VME TRIGGER ID=id /RESET/ENABLE/MASTER FASTCLEAR=f CONVERSION=c
```

Only arguments specified are changed in the trigger module. The module responds with a message displaying status and time settings. A more detailed description of the trigger handling is in the hardware manual. The trigger of FEP 1 must be set to `/MASTER`. All trigger modules must be reset and enabled after power up or hardware problems. Note that you must specify the destination in the way described above. An example for setting up trigger modules can be found in GOO$EXAMPLES:VME_TRIGGER.GCOM. Example:

```
     GOOSY> SET VME TRIG ID=10 /RES/ENAB/MAST FAST=10 CONV=20
 Message from crate 1, offset 6, subcrate 0, control 3.
 > cnt=1 st=00000120 ctrl=00000005 fclr=10 cvt=20 100ns
```

Debug Output

When you have terminals (or windows) connected to your frontend processors, you can enable some output to these windows.

```
   GOOSY> SET VME CONTROL FIC_DEBUG 1 ...
```

A value of 0 disables output. Note that you must specify the destination in the way described above.

Initialize CAMAC

In the readout tables CNAFs can be specified for readout or initialization. The initialization CNAFs are executed by command:

```
   GOOSY> INITIALIZE CAMAC ...
```

Note that you must specify the destination in the way described above.

VME messages

Messages delivered by the frontend processors are written to terminal. A line displaying the source is written first, e.g.

```
Message from crate 1, offset 4, subcrate 0, control 1.
> EB GEB 3.0 with 1[512Kb] started, 15 buffers [16 Kb]. FEPs: 1 sync. 0 async.
Message from crate 1, offset 5, subcrate 0, control 3.
> FEP GFP 3.0 with 2[512Kb] started. 32 buffers [24026 b] Type=3 Number=1.
```

CNAF

There are two ways to execute CNAFs. The first is the standard ESONE mechanism. One has to define the logical name `CAMAC_BRANCH_n` in a similar way as with the MBD or J11 systems. There is one extension to specify the VME branch where to execute the CNAF. The VME destination is specified by `crate:offset`. Offset 30 means the communication processor itself which may have a VSB branch connected.

```
$ DEFINE/JOB CAMAC_BRANCH_0 1:3  ! For CAMAC CNAF command in TMR
$ DEFINE/JOB CAMAC_BRANCH_0 1:30 ! For CAMAC CNAF command in TMR
GOOSY> CAMAC CNAF ...
$ DEFINE/JOB CAMAC_BRANCH_0 node::env____$TMR(GN_ESONE)1:3
```

The last definition is used by programs other than the TMR. The ESONE buffers are routed through the TMR to communication processor to destination processor and back. 'env' is the environment name of the TMR.

The other way is the command

```
GOOSY> CNAF VME ...
```

which works only in the TMR. This command should be used for single CNAFs mainly. It responds with a message. No definitions are necessary.

START/STOP ACQUISITION

When the VME system is set up it is ready for getting data. The `START ACQUISITION` command resets the buffer queues and sets the GO bit in the master trigger module. Note, that you must use the `/NET` qualifier when transfer is switched off or set to `/NET` (SET VME INPUT command). The `STOP ACQUISITION` command clears the GO bit and the master FEP (the first FEP in the setup file which controls the master trigger module) marks a last event. When the event builder encounters the last event, it delivers a message and marks the last buffer. When the TMR encounters a last buffer, it delivers a message similar to the one from the event builder. Up to that time a `START ACQUISITION` command is blocked. If no data is sent to the VAX, e.g. if VME INPUT is switched OFF, the qualifier `/RESET` is needed with the `START ACQUISITION` command.

TYPE EVENT

The command displays event data on the terminal if CHECK mode is active (see also below).

```
GOOSY>TYPE EVENT ID=id
```

The specification of an ID applies for VME data only. Only subevent data from subevents of the specified branch are displayed. From the other events only the subevent headers are displayed. Specifying a nonexisting ID results in a very compact output.

Example VME session with one CAMAC crate

```
T:GOOFY$A3$ ethdef e6elxb qb
Ethernet device DEV_QB = XQA0:, Protocol TMR = 5380X, Data interface UUA0:
Communication processor E6ELXB with address 00-00-5B-00-03-64
Event builder program EB_EXEC  is VME$EXE:EB_ROOT.EX20
Frontend  program     FEP_EXEC is VME$EXE:FEP_ROOT.EX20
T:GOOFY$A3$ crenv susi/$tmr
HVR connected from MVIIH::SUSI____$TMR(GN_XX_PRCTRL)
GOOSY environment SUSI created
-----------------------------------------------------
GOOSY environment process                           Node: MVIIH           Process name: GN_SU
GOOSY component process
GOOSY component process There are 3 processes in this job:
GOOSY component process
GOOSY component process   GN_SUSI (*)
GOOSY component process     GN_SUSI____$SVR
GOOSY component process     GN_SUSI____$TMR
T:GN_SUSI$ GOOSY
SUC: GOOSY> ini ac/vme
Link open acknowleged
SUC: GOOSY> load vme p @test
Load programs from EE$ROOT:[GOOFY.G.VME]TEST.VMEP;
-> TEST
--> TEST11.VMET
 Crate=1 Processor=5 ID=100 Index=0 Subcrate=0 Control=1 EB EB EB_EXEC
 Crate=1 Processor=6 ID=10 Index=1 Subcrate=0 Control=3 FEP CAV FEP_EXEC
Load system file VME$EXE:EB_ROOT.EX20
Load system file VME$EXE:FEP_ROOT.EX20
 Message from crate 1, offset 5, subcrate 0, control 1.
 > EB GEB 3.0 with 2[512Kb] started, 47 buffers [16 Kb]. FEPs: 1 sync. 0 async.
 Message from crate 1, offset 6, subcrate 0, control 3.
 > FEP GFP 3.0 with 1[512Kb] started. 32 buffers [7578 b] Type=3 Number=1.
SUC: GOOSY> loa vme t test1 1
Load tables from   EE$ROOT:[GOOFY.G.VME]TEST1.VMEP;
-> EE$ROOT:[GOOFY.G.VME]TEST1.VMEP;
--> TEST11.VMET
 Message from crate 1, offset 6, subcrate 0, control 3.
 > Readout tables for all triggers reset!
 Message from crate 1, offset 6, subcrate 0, control 3.
 > Trigger  1: Readout for Crate 1, Length 1 loaded.
SUC: GOOSY>
```

```
SUC: GOOSY> loa vme t test2 2
Load tables from   EE$ROOT:[GOOFY.G.VME]TEST2.VMEP;
-> EE$ROOT:[GOOFY.G.VME]TEST2.VMEP;
--> TEST12.VMET
 Message from crate 1, offset 6, subcrate 0, control 3.
 > Trigger  2: Readout for Crate 1, Length 2 loaded.
SUC: GOOSY> ini cam/cav
SUC: GOOSY> set vme inp/hvr
SUC: GOOSY> sho vme set
--------------------------------------------------------------------------------
Setup file  : EE$ROOT:[GOOFY.G.VME]TEST.
Number FEPs: 1 synchron 0 asynchron
-#---id---cr-pr-sc-cl-mode-cl--ty---------system program-------------------
  0   100  1  5  0  1 sync EB  EB  VME$ROOT:[VMEMAN.VMELIB.EXE]EB_ROOT
  1    10  1  6  0  3 sync CAV FEP VME$ROOT:[VMEMAN.VMELIB.EXE]FEP_ROOT
--------------------------------------------------------------------------------
    Trigger   1 EE$ROOT:[GOOFY.G.VME]TEST1.VMEP;
    Trigger   2 EE$ROOT:[GOOFY.G.VME]TEST2.VMEP;
--------------------------------------------------------------------------------
SUC: GOOSY> sho vme con /all
 Message from crate 1, offset 5, subcrate 0, control 1.
 > Bufind(FEP)=0 Events=0 Reject=0 Buffers=0
 Message from crate 1, offset 6, subcrate 0, control 3.
 > Bufferindex=-1 Events=0 Reject=0 Buffers=0
 Message from crate 1, offset 6, subcrate 0, control 3.
 > INIT list : Crate 1, Address 20012990, Length 4
 Message from crate 1, offset 6, subcrate 0, control 3.
 > Trigger  1: Crate 1, Address 2002A998, Length 1
 Message from crate 1, offset 6, subcrate 0, control 3.
 > Trigger  2: Crate 1, Address 2002A99C, Length 2
SUC: GOOSY>
SUC: GOOSY> set vme trig id=10 /res/enab/mast fast=10 conv=20
 Message from crate 1, offset 6, subcrate 0, control 3.
 > cnt=1 st=00000120 ctrl=00000005 fclr=10 cvt=20 100ns
SUC: GOOSY> sta ac
SUC: GOOSY> ty ev

********************* Buffer ***************************************************
Buffer type =   10,      Buffer number =        6,      Total length =     8168
Subtype =        1,      Events  =            510,      Data  length =     8160
No spanning events!                                     29-JUL-1992 18:06:47.81
********************************************************************************
```

```
==================== Event 1    =============================================
Type =   10, Subtype =    1, Length =     12, Trigger =  1, Event =        2551
Type =   10, Subtype =    1, Length =      4, Crate =    1, ID =    10, Ctrl CAV
1        :32768              :                  :                 :

SUC: GOOSY>
SUC: GOOSY> stop ac
GN_SUSI____$TMR: ===>  Acquisition stopped
 Message from crate 1, offset 5, subcrate 0, control 1.
 Event builder sent last buffer
 > EB:  Events=7210, Last buffer=15, evts=70, words=1120, t=0, e=
   TMR: Events=7210, Last buffer=15
GN_SUSI____$TMR: STOP ACQUISITION finished.
SUC: GOOSY> $ cl eth
Link aborted to Frontend.
SUC: GOOSY> $ EXIT
T:GN_SUSI$ dlenv
Subprocess deleted: GN_SUSI____$TMR
Subprocess GN_SUSI____$TMR has completed
Subprocess GN_SUSI____$SVR has completed
```

2.3.3 J11 Input

INITIALIZE ACQUISITION

Using the J11 the TMR must be initialized by

```
   GOOSY> INITIALIZE ACQUISITION mailbox size /J11 NODE=j11_node
```

'Mailbox' is a name used for the creation of the mailboxes. If not specified, the name of the environment is used (this is recommended). 'Size' specifies the buffer size in bytes. The default is 8 kB. 'J11_node' is the DECnet node name of the J11. Two links are opened to the specified DECnet node, one for commands, one for data, the internal buffer queues are created and the mailboxes are created.

LOAD J11

Now the CAMAC setup file must be loaded to the J11. This is done by

```
   GOOSY> LOAD J11 file
```

The file must contain a list describing the modules in the crate. For details see GOOSY Hardware Manual. Now the J11 should be ready to send data. The J11 may be initialized several times and can be loaded several times. The acquisition must be stopped to do that.

2.3.4 File Input

INITIALIZE ACQUISITION

Using the file input the TMR must be initialized by

```
GOOSY> INITIALIZE ACQUISITION mailbox size /FILE
```

'Mailbox' is a name used for the creation of the mailboxes. If not specified, the name of the environment is used (this is recommended). 'Size' specifies the buffer size in bytes. The default is 8 kB. The buffer size equals the file record size. The internal buffer queues and the mailboxes are created.

OPEN-CLOSE FILE

Now the input file must be opened. This is done by

```
GOOSY> OPEN FILE file
```

Now the TMR is ready to read data from the file.

2.3.5 Foreign Input

INITIALIZE ACQUISITION

This input channel can be used to support other frontends. The routines to control the frontend must be provided by the user.

2.3.6 START-STOP ACQUISITION

When the input channel is ready the data taking is started by

```
GOOSY> START ACQUISITION buffers events skip_buf skip_event
```

The optional parameters specify the number of buffers or events to process and the number of buffers or events to skip first. A number of zero means unlimited. This is the default.

```
GOOSY> STOP ACQUISITION
```

stops the data taking. All data in the frontends are read and written to the output file, if a file is open and started. A J11 system can be stopped in the way that all data buffers in the J11 are aborted by:

```
GOOSY> STOP ACQUISITION /ABORT
```

The links terminate. You must INIT the ACQUISITION again to start after a /ABORT. If the data input comes from a file, this file can be closed by

```
GOOSY> STOP ACQUISITION /CLOSE
```

To start the acquisition again (reading from a file) a file must be opened.
NOTE: The output file is not closed by STOP ACQUIS.

START-STOP Routines

It is possible to load private routines into the Transport Manager which are then called during START and STOP ACQUIS. See the `LOAD MODULE ACQUISITION` command in section 2.6 for that. These routines must be linked into a sharable image by the DCL command `LSHARIM` (see section **??** page **??**.

2.4 CAMAC Spectra

CAMAC spectra can be incremented in a MR2000 CAMAC memory. The spectra must be created as normal GOOSY spectra with additional information about the crate, station and offset of the spectrum in the CAMAC memory. Commands are provided to copy or add the contents from CAMAC memory into the GOOSY spectrum or to copy or add the contents of the GOOSY spectrum into the CAMAC memory. The accumulation of the CAMAC spectra is controlled by special `START` and `STOP` commands. Note that these commands are executed in the Data Base Manager which must therefore be running.

```
GOOSY> START MR2000 branch crate station /INIT
GOOSY> STOP  MR2000 branch crate station
GOOSY> CLEAR CAMAC SPECTRUM name /CAMAC/SPECTRUM
GOOSY> READ  CAMAC SPECTRUM name /ADD
GOOSY> WRITE CAMAC SPECTRUM name /ADD
GOOSY> SHOW  CAMAC SPECTRUM name       !contents of MR2000
```

2.5 Output Channels

The data buffers collected from the input channel are dispatched to one or more output channels. Normally analysis programs connect to these channels to get the data buffers. The TMR input may be synchronized by an analysis program. This mode must be enabled by a command (see `SET ACQUISITION` command). By default the output channels are filled only with samples, that means if the receiver has acknowledged a buffer. The file output channel synchronizes the input always.

2.5.1 Mailbox Output

Three mailboxes are scanned for output. One buffer is written to each mailbox after starting the acquisition. The second buffer is written to those mailboxes read by an analysis program. If a mailbox has not been read, no more buffers are written to this mailbox until it was read. The first mailbox can be used to synchronize input. This mode is enabled by

```
GOOSY> SET ACQUISITION /SYNC/MAIL
```

Then the TMR waits for the readout of this mailbox before initiating a new input. **NOTE: Mailbox input should be started first before setting synchronous mode**. An analysis program starts mailbox input by

```
GOOSY> START INPUT MAIL mailbox number
```

where 'mailbox' must be the same name as specified in the `INITIALIZE ACQUISITION` command and 'number' is 1,2 or 3.

2.5.2 DECnet Output

DECnet output channels are opened for analysis programs by

```
GOOSY> START INPUT NET node environment
```

where 'node' is the VAX node of the TMR and 'environment' is the name of the environment of the TMR. All opened DECnet channels are scanned for output. One buffer is written to each channel after it has been opened. The second buffer is written to those channels which received an acknowledge from the connected analysis program. DECnet channels can be used to synchronize input. This mode is enabled by

```
GOOSY> SET ACQUISITION /SYNC/NET
```

Then the TMR waits until the buffer is acknowledged by any of the connected analysis programs before initiating a new input. **NOTE: DECnet input should be started first before setting synchronous mode**. The same buffer may be written to several channels unless exclusive mode is enabled by

```
GOOSY> SET ACQUISITION /EXCLUSIVE
```

Then a buffer is sent only to one (free) channel. If only one channel is open, all buffers input to the TMR are analyzed. If several channels are open, all buffers are analyzed but by several analysis programs. NO buffer is sent to more than one analysis.

2.5.3 File Output

This channel synchronizes the TMR input. It 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 file can be on disk or on tape. A tape to be used must be mounted (see tape handling on page 27). This output channel 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 one types

```
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 file header can be copied to/from a disk text file, it can be modified using text editors or by prompting.

```
GOOSY> START OUTPUT FILE HEADEROUT=file /PROMPT
```

The information for the file header is prompted. The optional HEADEROUT specifies a text file to which the header is copied.

```
GOOSY> START OUTPUT FILE HEADERIN=file /EDIT
```

Once a header has been written to a file, it can be used again. The /EDIT qualifier calls first the editor to modify the file. You may also use the /PROMPT qualifier here.

Naming Conventions for IBM

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 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 VMS 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
```

2.6 Loading Private Routines

After the startup of the Transport Manager one can load private routines to be called by the `START-STOP ACQUISITION` commands.

2.6.1 LOAD MODULE ACQUISITION

These modules must be loaded by the command

```
GOOSY>LOAD MODULE ACQUISITION image module init /START
GOOSY>LOAD MODULE ACQUISITION image module init /STOP
```

The optional init parameter is the name of an initialization entry. This entry is called immediately by the command. All these modules must be linked in a sharable image. This is done very convenient by the DCL command `LSHARIM` (see section **??** on page **??**). When the modules are loaded, they are called by the `START-STOP ACQUISITION` commands. Their calling can be switched OFF and ON by the `SET ACQUISITION` command.

2.6.2 Enable/Disable Calling

Enable/disable the calling of loaded start or stop modules is done by

```
GOOSY> SET ACQUISITION /START
GOOSY> SET ACQUISITION /NOSTART
GOOSY> SET ACQUISITION /STOP
GOOSY> SET ACQUISITION /NOSTOP
```

2.7 Acquisition Synchronization

After startup the TMR only writes samples into the mailbox and DECnet channels. If a file channel is started, this channel synchronizes the input. Sometimes it is necessary to analyze 100% of the incoming data. Then the TMR must be synchronized with the analysis. Enable/disable mailbox synchronization is done by

```
GOOSY> SET ACQUISITION /SYNC/MAIL
GOOSY> SET ACQUISITION /NOSYNC
```

If mailbox synchronization is enabled, the TMR waits for the readout of the first mailbox. Then it fills the open DECnet channels if they are acknowledged and the output file. Enable/disable DECnet synchronization is done by

```
GOOSY> SET ACQUISITION /SYNC/NET
GOOSY> SET ACQUISITION /NOSYNC
```

If net synchronization is enabled, the TMR waits for any acknowledge of a DECnet channel. Then it fills the mailboxes if they have been read and the output file. Enable/disable exclusive output mode is done by

```
GOOSY> SET ACQUISITION /EXCLUSIVE
GOOSY> SET ACQUISITION /NOEXCLUSIVE
```

If exclusive mode is enabled the TMR writes one buffer only in one DECnet channel. Together with `/SYNC/MAIL` this means that only mailbox 1 is filled. Together with `/SYNC/NET` it means that all buffers are analyzed, each by one analysis program.

2.8 Miscellaneous Commands

Besides the commands described above the following commands are available.

2.8.1 Data Checking

After startup the TMR checks the buffer structure of each input buffer. This checking may be disabled. Enable/disable data checking is done by

```
GOOSY> SET ACQUISITION /CHECK       ! default
GOOSY> SET ACQUISITION /NOCHECK
```

If checking is enabled the TMR analyzes the data buffers and counts the events. If checking is disabled, the `TYPE EVENT-BUFFER` commands cannot work.

2.8.2 Compress Mode

Selecting J11 compress mode is done by

```
GOOSY> SET ACQUISITION /COMPRESS
GOOSY> SET ACQUISITION /NOCOMPRESS  ! default
```

For J11 input these commands control the data format written by the J11. If compression is enabled, the J11 writes no zeros, but the module number followed by nonzero value. If disabled, the J11 writes fixed length events. In both cases the appropriate unpack routine is provided and selected by the Analysis Manager. The events in the Data Base are the same.

2.8.3 SHOW ACQUISITION

This command gives an overview over the TMR activities and modes. A typical output looks like

```
GOOSY> SHOW ACQUISITION

 ------  Status of Data Acquisition: -----------  8-MAR-1988 17:11:47.23
Buffer size   :  8192    Count: 12
Queues: File:     4  LMD:  0  Current: FREE:   12  File:   0  LMD:   0
File input:  CLOSED
Acquisition : STOPPED,  List mode dump: STOPPED,   File: CLOSED
Buffer- and Event-Statistic:
    FILE reads:          0 buffers          0 events since clear
                         0 buffers          0 events since start
    LMD write:           0 buffers since clear
                         0 buffers since start
                         0 buffers since open
```

```
----------------------------------------------------------------------
  Online Analysis statistic:
  GOOSY_SUSI_1 buffers:         0 100%,          0 100%  since clear
  GOOSY_SUSI_2 buffers:         0 100%,          0 100%  since clear
  GOOSY_SUSI_3 buffers:         0 100%,          0 100%  since clear
Enabled:  buffer check -
Disabled: MBX synchronization - NET synchronization - exclusive -
----------------------------------------------------------------------

GOOSY>
```

The output may be directed to a file and optionally printed by

```
   GOOSY>SHOW ACQUISITION file /PRINT
```

The buffer and event counters can be cleared by

```
   GOOSY>SHOW ACQUISITION /CLEAR
```

Only brief information is displayed by

```
   GOOSY>SHOW ACQUISITION /BRIEF
```

The current file headers are displayed by

```
   GOOSY>SHOW ACQUISITION /INFILE/OUTFILE
```

On a separate terminal one can display a continuously updated overview by DCL command

```
    $ GSTAT env /$TMR
```

2.8.4 TYPE EVENT-BUFFER

These commands display buffer or event data on the terminal if CHECK mode is active.

```
   GOOSY>TYPE BUFFER number
   GOOSY>TYPE EVENT number
   GOOSY>TYPE EVENT number /SAMPLE
   GOOSY>TYPE EVENT number ID=id
```

'Number' specifies the number of buffers or events to be typed. The `/SAMPLE` switch works for VME, J11 and MBD buffers. It means that one event per buffer is typed. The specification of an ID applies for VME data only. Only subevent data from subevents of the specified branch are displayed. From the other events only the subevent headers are displayed. Specifying a nonexisting ID results in a very compact output.

2.9 Controlling the Acquisition

When the GOOSY environment and the TMR is created, the following strategy should be used to check if everything works as expected.

2.9.1 Checking Incoming Data

First start the acquisition. Then control if buffers are delivered to GOOSY. Use the `SHOW` command several times, especially if the data rates are low. Inspect the events written by the J11's.

```
GOOSY> START AC   ! start CAMAC readout
GOOSY> SHO ACQ    ! show if data buffers are read
GOOSY> TYPE EV 10 ! inspect 10 events
```

2.9.2 Analyze Data

The next step would be to send data to the analysis to see scatterplots and spectra. It is recommended to use a simple analysis first. You may create a dynamic list doing simple accumulations. This list can be deactivated later. Disable the analysis routine.

```
GOOSY> SET ANAL/NOANAL   ! Disable analysis routine
GOOSY> START INP MAIL    ! Start data input TMR to ANL
GOOSY> SHO ANAL/BR       ! Look if buffers are read
GOOSY> ATT DYN LIST accu ! Activate dynamic list, e.g. accu
GOOSY> ATT DYN LIST $scatter! Activate dynamic list for scatter
GOOSY> SHO ANAL/BR       ! Look if buffers are read
```

Before you attach the dynamic list you may check that data buffers are read into the analysis. Then inspect your data by looking to scatterplots and single spectra as accumulated in the dynamic list.

2.9.3 Modifying Hardware Setup

When you have to adjust the electronic setup, stop the acquisition first, make the electronic ready, clear the spectra, display a new scatterplot and start again.

```
GOOSY> STOP ACQUIS   ! stop CAMAC readout
! Do the modifications in the electronics
GOOSY> CLEAR SPEC *  ! clear spectra
GOOSY> DISP PI       ! display new scatterplot
GOOSY> START ACQ     ! Start CAMAC readout again
GOOSY> TYPE EVENT    ! Inspect incoming data
```

2.9.4 Writing to Tape

When the data looks OK, you may want to start writing to tape. One should always stop the acquisition before starting or stopping the output. Similar, one should always start the output first and then start the acquisition.

```
GOOSY> STOP AC                        ! Stop to reset hardware scalers
GOOSY> MOUNT TAPE M3: label /INIT     ! Mount and initialize the tape
GOOSY> START OUT FILE M3:filename     ! start data writing to tape
GOOSY> START ACQUIS                   ! Start CAMAC readout
GOOSY> SHO ACQUIS                     ! Check that buffers are written
GOOSY> STOP ACQ                       ! Stop CAMAC readout
GOOSY> STOP OUT FILE                  ! Stop and close output file
GOOSY> START OUT FILE M3:filename     ! start data writing to next file
GOOSY> START ACQUIS                   ! Start CAMAC readout
GOOSY> SHO ACQUIS                     ! Check that buffers are written
```

2.9.5 Full Analysis

The data acquisition and output is not affected by the analysis program. To enable the full analysis, one may detach the dynamic list, enable the private analysis routine, clear the spectra. The analysis input may be stopped first.

```
GOOSY> STOP INP MAIL       ! optional stop input
GOOSY> DET DYN LI accu     ! Disable dynamic list. e.g. accu
GOOSY> SET ANAL/ANAL       ! Enable calling of analysis routine
GOOSY> CLEAR SP *          ! Clear spectra
GOOSY> START INP MAIL      ! Start data input
GOOSY> SHO AN              ! Check that data buffers are read
GOOSY> DISP PIC xx         !
```

Appendix A

Transport Manager Commands

$ CLOSE ETHERNET

$ CLOSE ETHERNET /SHOW

PURPOSE	Close Ethernet link.
PARAMETERS	
/SHOW	Show statistics.
EXAMPLE	$ CL ETH

Description

FUNCTION	-
File name	I$ACVME.PPL
Action rout.	I$ACVME_CLO_ETH
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	20-Jul-1989

$ SET GNA ETHERNET

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

PURPOSE	SET Ethernet modes.
PARAMETERS	
Link	Character def="0 :30:10" Delta time for link check, i.e. 3 sec = '0 ::3'
Read	Character def="0 :00:30" Delta time for read timeout, i.e. 3 sec = '0 ::3' This time should be larger than write timeout times number of write retries times maximum number of physical buffers per logical buffer.
Write	Character def="0 :00:05" Delta time for write timeout, i.e. 3 sec = '0 ::3'
acknow	Integer Number of acknowledges to sent back for each buffer.
Wretry	Integer Number of write retries.
Lretry	Integer Number of ignored link checks.
Output	Character Optional output file.
/SWAP	Enable swapping. Def=/SWAP
/ISTREAM	Enable stream input. Def=/NOISTREAM
/OSTREAM	Enable stream output. Def=/NOOSTREAM
/DEBUG	Enable debug output. Def=/NODEBUG
EXAMPLE	$ SET GNA ETH READ="0 :00:50" /ISTR

Description

FUNCTION	-
File name	I$ACVME.PPL
Action rout.	I$ACVME_SET_GNA
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	20-Jul-1989

$ SHOW GNA ETHERNET

$ SHOW GNA ETHERNET /FULL /CLEAR

PURPOSE	Show Ethernet information.
PARAMETERS	
/FULL	Show full information.
/CLEAR	Clear counter.
EXAMPLE	$ SHO GNA ETH /F

Description

FUNCTION	-
File name	I$ACVME.PPL
Action rout.	I$ACVME_SHO_GNA
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	20-Jul-1989

CALCULATE FASTBUS PEDESTAL

```
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]
```

PURPOSE Set fastbus pedestal subtraction on/off.

PARAMETERS

loop integer (def=100)
Number of events to measure.

throff integer (def=0)
Threshold offset [channels]to add to mesured value

thrfact real (def=1.)
Factor for mesured threshold.

pedoff integer (def=100)
Pedestal offset [channels]to add to mesured value.

pedfact real (def=1.)
Factor for mesured pedestal.

sample integer (def=100)
Sample interval [events]. After "sample" events one event will be not compressed.

trigger integer (def=1)
Trigger number

VMEcrate,processor List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1

ID integer
Processor ID

dummy	NOT used
crate	Crate number
node	NET node
/ON/OFF	Switch compression ON or OFF
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
EXAMPLE	SET VME TRIG

Description

FUNCTION	Measure and calculate pedestals and thresholds.
File name	I$ACV_CALC_PED.PPL
Action rout.	I$ACV_CALC_PED
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

CAMAC CLEAR

CAMAC CLEAR C=c

PURPOSE	Generate Dataway clear
PARAMETERS	
C=c	Number of the crate, in which the dataway clear has to be performed.
EXAMPLE	CAMAC CLEAR 1
Action rout.	I$MCCCC
Author	Walter F.J. Mueller

Remarks

File name	I$MCCCC.PPL
Dataset	-
REMARKS	-

Description

CALLING	@CALL I$MCCCC(I_C);
ARGUMENTS	
I_C	BIN FIXED(15) [INPUT] Crate number between 1 and 7 in which the dataway clear has to be executed.
FUNCTION	This procedure calls I$CCCC to perform a dataway clear (C) in the specified crate.
REMARKS	-
EXAMPLE	@CALL I$MCCCC(1);

CAMAC CNAF

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

PURPOSE	Perform a single CAMAC action
PARAMETERS	
C=c	Crate number, between 1 and 7. Replaceable default = 1
N=n	Station number, between 1 and 31. Replaceable default = 1
A=a	Subaddress, between 0 and 15. Replaceable default = 0
F=f	Function code, between 0 and 31. Replaceable default = 0
DATA=d	Data word. Should be specified if the function code is between 16 and 23 and will be ignored otherwise. NOTE: The data word may be entered also in binary (e.g. %B101), octal (e.g. %O123), or hexadezimal (e.g. %XA1B) if convienient. Replaceable default = 0
Branch=b	Branch number. Used for the translation of CAMAC_BRANCH_b
EXAMPLE	CAMAC CNAF C=1 N=12 A=3 F=16 DATA=%O777
Action rout.	I$MCCNA
Author	Walter F.J. Mueller

Remarks

File name	I$MCCNA.PPL
REMARKS	-

Description

CALLING @CALL I$MCCNA(I_C,I_N,I_A,I_F,L_DATA);

ARGUMENTS

I_C BIN FIXED(15) [INPUT]
Crate number, between 1 and 7.

I_N BIN FIXED(15) [INPUT]
Station number, between 1 and 31.

I_A BIN FIXED(15) [INPUT]
Subaddress, between 0 and 15.

I_F BIN FIXED(15) [INPUT]
function code, between 0 and 31.

L_DATA BIN FIXED(31) [INPUT]
Data word to be written if function is between 16 and 23, ignored otherwise.

I_B BIN FIXED(15) [INPUT]
Branch number. This number is used for the translation of CAMAC_BRANCH_b.

FUNCTION This procedure calls I$CFSA to execute the given CAMAC action and displays the result.

REMARKS -

EXAMPLE @CALL I$MCCNA(1,1,0,16,1);

CAMAC DEMAND

```
CAMAC DEMAND C=c
   /ENABLE/DISABLE/TEST
```

PURPOSE	Enable, disable or test Crate Demand
PARAMETERS	
C=c	Number of the Crate, for which the demand has to be enabled, disabled or tested.
/ENABLE	The demand will be enabled and the status displayed.
/DISABLE	The demand will be disabled and the status displayed
/TEST	Only the status will be displayed, this is the default. It is signaled, whether demands are present in this crate.
EXAMPLE	CAMAC DEMAND 1
Action rout.	I$MCCCD
Author	Walter F.J. Mueller

Remarks

File name	I$MCCCD.PPL
Dataset	-
REMARKS	The /ENABLE and /DISABLE functions may interfere with other running MBD programs, be carefull !!!

Description

CALLING	@CALL I$MCCCD(I_C,C_MODE);
ARGUMENTS	

I_C	BIN FIXED(15) [INPUT] Number of the crate, in which the demand has to be enabled, disabled or tested.
C_MODE	CHAR(*) VAR [INPUT] Mode qualifier, may have the values:

/ENABLE	Demand will be enabled
/DISABLE	Demand will be disabled
/TEST	Demand enablement and the presents of demands will be tested.

FUNCTION If C_MODE is '/ENABLE' or '/DISABLE', the demand in the specified crate is enabled or disabled by a call of I$CCCD. Afterwards or if C_MODE is '/TEST' the status of the demand enablement and the presents of demands is tested with a call of I$CTCD and I$CTGL and displayed.

REMARKS -

EXAMPLE @CALL I$MCCCD(1,'/TEST');

CAMAC INHIBIT

CAMAC INHIBIT C=c
/SET/CLEAR/TEST

PURPOSE	Set, clear or test Dataway inhibit
PARAMETERS	
C=c	Number of the crate, in which the inhibit has to be set, cleared or tested.
/SET	The inhibit will be set and the status displayed.
/CLEAR	The inhibit will be cleared and the status displayed
/TEST	The status of the inhibit will only be displayed.
EXAMPLE	CAMAC INHI 1 /CLEAR
Action rout.	I$MCCCI
Author	Walter F.J. Mueller

Remarks

File name	I$MCCCI.PPL
Dataset	-
REMARKS	-

Description

CALLING	@CALL I$MCCCI(I_C,C_MODE);
ARGUMENTS	
I_C	BIN FIXED(15) [INPUT] Number of crate, for which the Inhibit has to be set, cleared or tested.

C_MODE	CHAR(*) VAR [INPUT] Mode qualifier, may have the values:

/SET	Will set inhibit
/CLEAR	Will clear inhibit
/TEST	Will test and display inhibit status

FUNCTION	This procedure calls I$CCCI to set or clear the inhibit in the selected crate and I$CTCI to test the inhibit status.
REMARKS	-
EXAMPLE	@CALL I$MCCCI(1,'/TEST');

CAMAC INITIALIZE

CAMAC INITIALIZE C=c

PURPOSE	Generate Dataway Initialize
PARAMETERS	
C=c	Number of crate, in which the dataway initialize is to be executed.
EXAMPLE	CAMAC INIT 1
Action rout.	I$MCCCZ
Author	Walter F.J. Mueller

Remarks

File name	I$MCCCZ.PPL
Dataset	-
REMARKS	-

Description

CALLING	@CALL I$MCCCZ(I_C);
ARGUMENTS	
I_C	BIN FIXED(15) [INPUT] Crate number between 1 and 7 in which the dataway initialize has to be performed.
FUNCTION	This procedure calls I$CCCZ to do a dataway init in the specified crate.
REMARKS	-
EXAMPLE	@CALL I$MCCCZ(1);

CAMAC SCAN

CAMAC SCAN C=c N=n F=f /CRATE/STATION/ADDRESS

PURPOSE Perform a crate scan

PARAMETERS

C=c Number of the Crate to be inspected.

N=n Number of the Station to be inspected for a station or address scan.

F=f Function code to be used for an address scan.

/CRATE Signals crate scan. Will try in the specified crate all station - subaddress combinations for function code F=0 and reports for each station:
the highest subaddress with an Q or X response
the 'strongest' response (X or Q) found

/STATION Signals station scan, this is the default. Will try in the specified crate and station all subaddress - function combinations and reports the Q and X response for each combination:
- indicates no X, no Q
X indicates X but no Q
q indicates Q but no X (strange combination !!!)
Q indicates X and Q

/ADDRESS Signals address scan. Will read out in the specified crate all subaddresses of either all or the specified station with the given function code and reports the value if there was a Q response.

Action rout. I$MCSCN

Author Walter F.J. Mueller

Remarks

File name	I$MCSCN.PPL
Dataset	-
REMARKS	-
EXAMPLE	-

Description

CALLING @CALL I$MCSCN(I_C,I_N,I_F,C_MODE,B_DP);

ARGUMENTS

I_C BIN FIXED(15) [INPUT]
Crate to be scanned, must be between 1 and 7.

I_N BIN FIXED(15) [INPUT]
Station to be scanned for a station scan, must be between 1 and 31.

I_F BIN FIXED(15) [INPUT]
Function code to be used for a address scan, must be between 0 and 31.

C_MODE CHAR(*) VAR [INPUT]
Scan mode, valid are:

/CRATE	Perform crate scan, will use I_C.
/STATION	Perform station scan, will use I_C, I_N.
/ADDRESS	Perform address scan, will use I_C, I_F.

B_DP BIT(*) ALIGNED [INPUT]
Default pattern, is used to determine whether I_N was specified in an address scan or not.

FUNCTION For details look in the command description.

REMARKS -

EXAMPLE @CALL I$MCSCN(1,1,0,'/CRATE');

CLOSE FILE

CLOSE FILE

PURPOSE	Close data input file.
EXAMPLE	CLOSE FILE
NOTE	The acquisition must be initialized with /FILE

Description

FUNCTION	Closes a file for data input. The input is stopped by STOP ACQUIS.
File name	I$ACQ_CLO_FIL.PPL
Action rout.	I$ACQ_CLO_FIL
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	20-AUG-1987

CLOSE OUTPUT FILE

CLOSE OUTPUT FILE

PURPOSE	Close list mode dump file
PARAMETERS	
EXAMPLE	CLOSE O F
NOTE	This command is called by STOP OUT FILE which should be used. Actually the output must be stopped anyway.

Description

FUNCTION	Close list mode file.
File name	I$ACQ_CLO_LMD.PPL
Action rout.	I$ACQ_CLO_LMD
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

CNAF VME

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

PURPOSE	Execute CNAF
PARAMETERS	
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
C	Crate number
N	Station
A	Subaddress
F	Function
times	Repetition
data	Data
ID	integer Processor ID
dummy	NOT used
node	optional node name of NET
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
EXAMPLE	CNAF VME 1,1 1 1 2 24

Description

FUNCTION	Execute CNAF.
File name	I$ACV_CNAF_VME.PPL
Action rout.	I$ACV_CNAF_VME
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

CREATE PROGRAM

```
CREATE PROGRAM file structure
             /MBD /J11
             /STRUCTURE
             /[NO]PROGRAM
             /COMPILE
```

PURPOSE	Generates J11 stand alone programs and event data element declarations from CAMAC description file. (called in MUTIL)
PARAMETERS	
file	CAMAC description file. File type should be .CAM.
structure	Optional name for structure of event data element. This name is used as filename for a declaration file and as structure name. If not specified, the name of the input file postfixed by '_DCL' is used. Filetype is .TXT.
/MBD/J11	Default = /MBD Create declaration for a MBD or J11 event.
/STRUCTURE	Create declaration file.
/PROGRAM	Default for /MBD Create J11 programs (only for /MBD) The names of the programs are composed from the name of the CAMAC file postfixed by 'Cn' where n is the crate number. The file type is .MAC.
/COMPILE	Compile and link J11 programs (only for MBD)
EXAMPLE	CRE PROG TEST.CAM X$EVT /MBD/STRUC/PROG/COMP Generates files TESTC0.MAC, TESTC1.MAC,.. and X$EVT.TXT for the declaration of structure X$EVT. The macro files are compiled and linked. CRE PROG TEST.CAM X$EVT /J11/STRUC Generates file X$EVT.TXT for the declaration of structure X$EVT.

Description

FUNCTION	The CAMAC description file as described in the harware manual is used as input to generate declarations for event data elements and J11 stand alone programs for MBD systems.
MBD	A structure declaration matching SA$MBD is generated. Optionally the J11 programs to read out the CAMAC crates is generated and compiled.
J11	For single crate systems a declaration matching SA$EVENT is generated. The module names as specified in the description file are inserted in the structure.

DEBUG VME MEMORY

```
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]
```

PURPOSE	Read/write/copy VME memory.
PARAMETERS	
start	integer (def=0) Start memory address (source)
end	integer (def=0) End memory address (destination)
value	integer (def=0) Value for write
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
crate	Crate number
node	NET node
/READ/WRITE/COPY	Select read or write/read or copy.
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
EXAMPLE	DEB VME MEM

Description

FUNCTION	Read/write/copy VME memory.
File name	I$ACV_DEB_MEM.PPL
Action rout.	I$ACV_DEB_MEM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

DISMOUNT TAPE

```
DISMOUNT TAPE device
                    /[NO]UNLOAD
```

PURPOSE	Dismount tape.
PARAMETERS	-
device	required string global replace Tape device.
/[NO] UNLOAD	switch default=/NOUNLOAD Unload tape after dismount.
EXAMPLE	DISMOUNT TAPE M1 /NOUNL

Description

FUNCTION	Dismount a tape.
File name	I$ACQ_DISMOUNT.PPL
Action rout.	I$ACQ_DISMOUNT
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	04-Jan-1988

DUMP MBD

```
DUMP MBD FROM=f TO=t BDO=bdo BDD=bdd
         /HEXADECIMAL/DECIMAL/OCTAL/BIT
         /INSTRUCTION
```

PURPOSE Format a MBD memory dump

PARAMETERS

FROM=f First address to be dumped, must be between 0 and 4095. NOTE, that you may enter the addresses in any radix, e.g. %O100 specifies 100 octal. The default is 0.

TO=t Last address to be dumped, must be between the value given for FROM and 4095. The default is 4095. The whole memory will be dumped if either FROM or TO is specified, but zero of repeating locations are compacted in the dump.

BDO=bdo MBD object file. If this parameter is given, the specified object file will be read and the given address range is dumped. The default file type is BDO. You may specify any file which is valid for loading with the LOAD MBD command.

BDD=bdd MBD dump file. If this parameter is given, the specified dump file will be read and the given address range is dumped. The default file type is BDD. The dump files are in general created with the STORE MBD command. NOTE: If neither BDO nor BDD are specified, the command will show the memory contents of the currently active MBD.

/HEXADECIMAL Dump the memory contents with hexadecimal radix.

/DECIMAL Dump the memory contents with decimal radix.

/OCTAL Dump the memory contents with octal radix. This is the default.

/BIT Dump the memory contents with binary radix.

/INSTRUCTION Dump the memory contents as disassembled instructions. Each line contains the address of the instruction and the instruction itself in octal

	radix, the CNAF interpretation and the mnemonics of the disassembled instruction. Note, that the CNAF numbers are in decimal radix, whereas all other numbers are in octal radix !!
FUNCTION	This command dumps the contents of the MBD memory, an MBD object file or a MBD memory dump file in either hexadecimal, decimal, octal or binary radix. Note, that the addresses in the first column are always printed in octal radix. Repetive lines are suppressed in order to avoid long output indicating a zeroed memory.
Action rout.	I$MCDMM
Author	Walter F.J. Mueller

Remarks

File name	I$MCDMM.PPL
Dataset	-
REMARKS	-
EXAMPLE	DUMP MBD

Description

CALLING @CALL I$MCDMM(I_FROM,I_TO,C_BDO,C_BDD,C_MODE);

ARGUMENTS

I_FROM	BIN FIXED(15) [INPUT] First address to be dumped, between 0 and 4095.
I_TO	BIN FIXED(15) [INPUT] Last address to be dumped, between I_FROM and 4095.
C_BDO	CHAR(*) VAR [INPUT] Name of a MBD object file, the default file type is .BDO.
C_BDD	CHAR(*) VAR [INPUT] Name of a MBD dump file, the default file type is .BDD.
C_MODE	CHAR(*) VAR [INPUT] Set of mode qualifiers:

/DECIMAL	Write in decimal radix.
/HEXADECIMAL	Write in hexadecimal radix.
/OCTAL	Write in octal radix.
/BIT	Write in binary radix.
/INSTRUCTION	Write intruction mnemonics.

FUNCTION This procedure dumps either the memory of the MBD directly or the contents of an object or dump file. If both C_BDO and C_BDD are empty strings, the current MBD memory will be shown. If one of the file names is given, it will be read and dumped.

REMARKS -

EXAMPLE @CALL I$MCDMM(0,4095,",",'/DECIMAL');

DUMP STARBURST

DUMP STARBURST FROM=f TO=t C=c N=n TSK=file /HEXADECIMAL/DECIMAL/OCTAL/BIT

PURPOSE Format a STARBURST memory dump through MBD

PARAMETERS

FROM=f First address to be dumped, must be between 0 and and the memory size. The address must be specified as a byte address (as all addresses in a PDP-11), but it is always rounded to a word boundary. NOTE, that you may enter the addresses in any radix, e.g. %O100 specifies 100 octal. The default is 0.

TO=t Last address to be dumped, must be between the value given for FROM and the memory size.

C=c Crate number of the STARBURST to be accessed. The default is Crate 1.

N=n Station number of the STARBURST to be accessed. The default is Station 23.

TSK=file Task image file. If this parameter is given, the specified image file will be read and the given address range is dumped. The default file type is TSK. You may specify any file which is valid for loading with the LOAD STARBURST command. NOTE: If TSK is not specified, the command will show the memory contents of the STARBURST in crate C and station N.

/HEXADECIMAL Dump the memory contents with hexadecimal radix.

/DECIMAL Dump the memory contents with decimal radix.

/OCTAL Dump the memory contents with octal radix. This is the default.

/BIT Dump the memory contents with binary radix.

FUNCTION This command dumps the contents of a STARBURST memory or a image file either hexadecimal, decimal, octal or binary radix. Note, that the addresses in the first column are always printed in octal radix. Repetive lines are suppressed in order to avoid long output indicating a zeroed memory.

EXAMPLE	DUMP STARBURST
Action rout.	I$MCDMS
Author	Walter F.J. Mueller

Remarks

File name	I$MCDMS.PPL
Dataset	-
REMARKS	-

Description

CALLING @CALL I$MCDMS(L_FROM,L_TO,I_C,I_N,C_TSK,C_MODE);

ARGUMENTS

L_FROM BIN FIXED(31) [INPUT]
Lower dump address limit, between 0 and the memory size.

L_TO BIN FIXED(31) [INPUT]
Upper dump limit, between L_FROM and the memory size.

I_C BIN FIXED(31) [INPUT]
Crate number of STARBURST to be accessed.

I_N BIN FIXED(31) [INPUT]
Station number of STARBURST to be accessed.

C_TSK CHAR(*) VAR [INPUT]
Name of Task image file to be dumped. If omitted, the CAMAC module specified with I_C and I_N will be accessed.

C_MODE CHAR(*) VAR [INPUT]
Mode qualifier set:

/DECIMAL	Write in decimal radix.

/HEXADECIMAL	Write in hexadecimal radix.
/OCTAL	Write in octal radix.
/BIT	Write in binary radix.

FUNCTION This procedure dumps either the memory of a STARBURST directly or the contents of an image file. If both C_TSK is an empty strings, the current STARBURST memory will be shown. Otherwise the image file is read and dumped.

REMARKS -

EXAMPLE @CALL I$MCDMS(0,4095,1,23,'','/DECIMAL');

EXECUTE VME

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

PURPOSE	Execute command on remote VME processor
PARAMETERS	
command	command string (counted ASCII, zero terminated)
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
subcrate	subcrate
node	optional node name of NET
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
EXAMPLE	EXE VME "START RUN" /AEB

Description

FUNCTION	Execute command on remote VME processor
File name	I$ACV_EXE_VME.PPL
Action rout.	I$ACV_EXE_VME

Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

FOREIGN ACQUISITION

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

PURPOSE	Whatever
PARAMETERS	
string	string replace default=" description
longword	integer default=8192 range=(1,24576) description
QUAL	qualifier default=/X

/X	decription
/Y	decription
/Z	decription

/SWITCH	negatable switch default=/SWITCH description +Command

Module

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

PURPOSE	Init data taking
PARAMETERS	
mailbox	string replace String to build mailbox name: GOOSY_mailbox_1,2,3

	Three mailboxes are created. Default is the environment name. The name must be unique. This is ensured by using the environment name.
size	integer replace default=8192 Size of listmode data buffers in bytes, must be between 512 and 28672. The size should be a multiple of 512 or even better a multiple of 4096. Together with the /PAGE /KBYTE qualifier this quantity can be specified in different units than bytes. If zero, defaults to 16384 for VME, 8192 others.
count	integer default=8 Number of listmode data buffers, must be between 4 and 32. The default is normally adequate.
in_buffers	integer default=4 Number of buffers queued to MBD. The default is normally adequate.
out_buffers	integer default=0 Number of buffers hold before writing to the output device is started. The default is normally adequate.
node	string replace Node of J11 (only needed for /J11)
command	string replace default=CMDERR Command component of J11 (/J11 only) The default is adequate.
data	string replace default=TMR11S Data component of J11 (/J11 only) The default is adequate.
/MBX	Open input mailbox. String to build mailbox name: GOOSY_mailbox_I1
/MBD	CAMAC interface is MBD
/J11	CAMAC interface is J11
/FILE	Input from file (use OPEN-CLOSE FILE)
/FOREIGN	Input from unknown source.
/PAGE	Buffer size in pages (512 bytes)

/BYTE	Buffer size in bytes
/KBYTE	Buffer size in Kbytes (1024 bytes)
EXAMPLE	INIT ACQU ANNA SIZE=8192 COUNT=12 IN_BUF=6 creates mailboxes GOOSY_ANNA_1,2,3

Description

FUNCTION	This procedure initializes the buffer pool used for listmode data. The given number of buffers is allocated and locked in the workingset. NOTE: The number and size of buffers cannot be changed afterwards whereas the queuing parameters may be modified. In J11 mode the links to the J11 are opened. he mailboxes are created. The size of the mailboxes ust be 8192 for the J11 input. This value is efaulted with the /J11 qualifier. With /MBD the ailbox size default is 4096.
File name	I$ACQ_INI_ACQ.PPL
Action rout.	I$ACQ_INI_ACQ
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

INITIALIZE CAMAC

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

PURPOSE	Initialize CAMAC
PARAMETERS	
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
node	optional node name of NET
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
EXAMPLE	

Description

FUNCTION	Send INIT CAMAC command .
File name	I$ACV_INI_CAM.PPL
Action rout.	I$ACV_INI_CAM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

LOAD J11

```
LOAD J11 file events
           /KEEP /COMPRESS
```

PURPOSE	Load CAMAC module table into J11
PARAMETERS	
file	string replace default=J11.CAM File containing module specifications. See Hardware Manual for description.
events	integer replace default=0 Maximum number of events to be filled in a buffer. A value of 0 (=default) means as many as possible. Acquisition must be initialized. STOP the ACQUISITION before loading the J11.
/COMPRESS	Suppress zeros.
/KEEP	Keep buffers in J11
EXAMPLE	LOAD J11 mymod.CAM

Description

FUNCTION	Loads module description table to J11.
File name	I$ACQ_LOA_J11.PPL
Action rout.	I$ACQ_LOA_J11
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	24-feb-1987

LOAD LRS_2365

```
LOAD LRS_2365 file C=c N=n
    /[NO]LOG
    /[NO]DUMP
```

PURPOSE Load definitions in a LRS 2365 logic matrix

PARAMETERS

file	Name of a definition file, the default file type is DAT.
C=c	Crate number of the module to be loaded.
N=n	Station number of the module to be loaded.

/[NO] LOG Logs the contents of the definition file on SYS$OUTPUT. The definition file is just copied to SYS$OUTPUT while they are read and parsed.
/NOLOG: Don't log definition file, this is the default.

/[NO] DUMP Dumps the contents of the LRS 2365 module found after the down load. The programming words PW 0 to 17 are shown in binary radix. For their interpretation look in the LRS manual.
/NODUMP: Don't dump programming words, this is the default.

FUNCTION This procedure reads the definition file and generates the programming words needed to load the LRS 2365 logic matrix. This unit is an eight fold programmable overlap coinzidence with 16 inputs. Each input can be enable, disabled or inverted for each of the 8 channels, the outputs may be inverted. This allows to perform AND as well as OR operations. The definition file must have the format: The character of the line must be '!','A','O','T'.
A line beginning with a '!' is a comment and will be ignored.
A line beginning with an 'A' or 'O' indicates an AND or OR operation. There may be up to 8 lines beginning with 'A' or 'O', one for each output channel. The rest of the line may contain up to 16 '+',

'-' or 'X' indicating whether an input is to be used directly or inverted or to be ignored.
A line beginning with a 'T' defines the TEST partern. The rest of the line may contain up to 16 '0' or '1' seperated by blanks. A sample input file may look like:

```
! Some comments
!
! 1 1 1 1 1 1 1
! 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
!
A + + X X X X X X X X X X X X X X
A X + + X X X X X X X X X X X X X
O + + + X X X X X X X X X X X X X
!
A X X X + - X X X X X X X X X X X
A X X X X + X X X X X X X X X X X
A X X X X X + X X X X X X X X X X
A X X X X X X + X X X X X X X X X
A X X X X X X X + X X X X X X X X
```

This results in:

```
Out(0) = In(0) .AND. In(1)
Out(1) = In(1) .AND. In(2)
Out(2) = In(0) .OR. In(1) .OR. In(2)
Out(3) = In(3) .AND. .NOT. In(4)
Out(4) = In(4)
Out(5) = In(5)
Out(6) = In(6)
Out(7) = In(7)
```

EXAMPLE LOAD LRS2365 MYCOINZ 1 1

Action rout. I$LDLRS_2365

Author Walter F.J. Mueller

Remarks

File name I$LDLRS_2365.PPL

Dataset -

REMARKS -

Description

CALLING @CALL I$LDLRS_2365(C_FILE,I_C,I_N,C_LOG,C_DUMP);

ARGUMENTS

C_FILE CHAR(*) VAR [INPUT]
Name of the definition file.

I_C BIN FIXED(15) [INPUT]
Crate number of the module to be loaded.

I_N BIN FIXED(15) [INPUT]
Station number of the module to be loaded.

C_LOG CHAR(*) [INPUT]
LOG qualifier set:

/LOG	Log definition file.
/NOLOG	Don't log definition file.

C_DUMP CHAR(*) [INPUT]
DUMP qualifier set:

/DUMP	Dump programming words.
/NODUMP	Don't dump programming words.

FUNCTION Reads the definition file and load a LRS 2365 module. For details look in the command description.

REMARKS -

EXAMPLE @CALL I$LDLRS_2365('MYDEF',1,1,'','');

LOAD MBD

```
LOAD MBD file
   /EXECUTIVE
```

PURPOSE Load microcode in MBD, either executive or a channel program.

PARAMETERS

file Name of the compiled MBD program. The default file type is BDO (for Branch driver object code). The program must have been written with the MBD macro sets MBDGLO and MBD2PG and must have been compiled with the VMS/RSX cross assembler. It must furtheron contain the propper header indicating start address, length and channel. For details look in the function description.

/EXECUTIVE Enables loading of the executive. The executive is the code for the MBD channel 7. It performs common functions like load of other channels and read back of the MBD memory. The executive code must be loaded before any other code after a boot, but a reload of it later on will erase all other channel codes.

FUNCTION This procedure loads the MBD memory with program code which has been compiled with the VMS/RSX compatibility mode assembler in absolute mode. The channel code must have been compiled with the MBDGLO and MBD2PG macro sets in absolute mode. The program must be prefixed by a header which contains information about:

the program length (in 16 bit words)
the load and init start address
the channel, for which the program should execute

A sample program may look as follows:

```
    .TITLE CNAF-PROGRAM
    .LIST TTM
    .ENABLE ABSOLUTE
    .MCALL MBDGLO,MBD2PG
    MBDGLO
    MBD2PG
```

```
        .MCALL TPAGE,ENTRY,CONST,VARIA,COFCNA,FILL
        .MCALL
CAMWRT,CAMWRC,CAMFNC,CAMRD,CAMW,CAMVEC
    ;
    $=400 ; fix the start and init address
    JCSTA=$  ; begin marker
                   ; set up 4 word header block with:
                   ; length (including header!)
                   ; a 0 (was function modifier)
                   ; start and init address
                   ; channel number (0..7)
        .WORD JCEND-JCSTA+4 ; length
        .WORD 0
        .WORD JCSTA ; start & init
        .WORD 0 ; channel number 0
        .....
        any MBD code, for an example look in
            GOO$IO:ESONE.MBD
        .....
        JCEND=$  ; end marker
        .END
```

REMARKS If a MBD code will be loaded into a MBD where another MBD code was loaded before, the new code be shorter (or equal) in memory than the old one. Otherwise use the RELEASE MBD CHANNEL command before loading the new code.

EXAMPLE LOAD MBD MYPROG

Action rout. I$MCLDM

Author Walter F.J. Mueller

Remarks

File name I$MCLDM.PPL

Dataset -

Description

CALLING @CALL I$MCLDM(C_FILE,C_MODE);

ARGUMENTS

C_FILE	CHAR(*) VAR [INPUT] File name of the compiled MBD program.
C_MODE	CHAR(*) VAR [INPUT] Executive qualifier, enables to load executive.
FUNCTION	Load the MBD executive or channel code.
REMARKS	-
EXAMPLE	@CALL I$MCLDM('MYPROG','');

LOAD MODULE ACQUISITION

LOAD MODULE ACQUISITION image module init /START/STOP [=TYPE]

PURPOSE Load module from sharable image.

PARAMETERS

image string
name of sharable image for routines.

module string
name of routine.

init string
name of initialization entry of routine.

/START switch
The module is called at the beginning of
START ACQUIS

/STOP switch
The module is called at the end of
STOP ACQUIS
The modules must be linked into a sharable image by DCL command LSHARIM. The initialization entry is called.

EXAMPLE LOAD MOD ACQU MYSHARE X$START $XSTART /START

Description

FUNCTION Loads a sharable image and the specified module. The sharable image must be linked by command:
LSHARIM (see help).

File name I$ACQ_LOA_MOD.PPL

Action rout. I$ACQ_LOA_MOD

Dataset -

Version	1.01
Author	H.D.Essel
Last Update	12-APR-1985

LOAD STARBURST

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

PURPOSE Load a system or task image in the STARBURST memory.

PARAMETERS

file Name of a RSX11-M task or system image. The default file type is TSK.

C=c Crate number of the STARBURST to be loaded.

N=n Station number of the STARBURST to be loaded.

/[NO] HALT The CPU is explicitely halted before any load operation is done. This is the default.
/NOHALT: The CPU is not halted before the load operation. this may be usefull for the load of a data partition or of a task partition.

/BOOT The CPU is booted after the load at the start address found in the task image label. This is done by writing a JMP @#<startaddress> instruction in the memory locations 0 and 2, setting the powerup interrupt address to 0 (in location 24,26) and by simulating a power failure.

/INIT The CPU is halted after the load and put into console ODT mode. Type <address>G on the local console to start program execution. This can be used to start at an address different from the system start address.

FUNCTION This procedure loads a system or task image into the memory of a STARBURST processor. The processor is halted (if /NOHALT is not specified) before the memory is accessed in order to avoid race conditions with programs still running in the STARBURST. Than the image file is loaded and read back to verify a propper load. Note, that only first 128 kbyte are accessible for this command.
The CPU is either put in console ODT mode (/INIT) or booted at the start address found in the image label (/BOOT). A STARBURST

bootable system image should be task builded with the following options:

```
$ MCR TKB
<image>.TSK/-HD,<image>/-SP=<object modules>
/
STACK=0
//
```

The DCL command LINKJ11 may be used to link a J11 task program.

EXAMPLE LOAD STARBURST MYPROG C=1 N=23 /BOOT

Action rout. I$MCLDS

Author Walter F.J. Mueller

Remarks

File name I$MCLDS.PPL

Dataset -

REMARKS -

Description

CALLING @CALL I$MCLDS(C_FILE,I_C,I_N,C_HALT,C_START);

ARGUMENTS

C_FILE CHAR(*) VAR [INPUT]
File name of the compiled MBD program.

I_C BIN FIXED(15) [INPUT]
Crate number of the STARBURST to be loaded.

I_N BIN FIXED(15) [INPUT]
Station number of the STARBURST to be loaded.

C_HALT CHAR(*) VAR [INPUT]
Halt qualifier set:

/HALT	Halt CPU before load.
/NOHALT	Load without halting CPU.

C_START CHAR(*) VAR [INPUT]
Boot qualifier qualifier set:

/INIT	Init CPU, activate console ODT.
/BOOT	Boot CPU via power fail simulation.

FUNCTION This procedure loads a system image into a STARBURST processor and boots it.

REMARKS -

EXAMPLE @CALL I$MCLDS('MYPROG',1,23,'','');

LOAD VME PROGRAM

```
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
```

PURPOSE	Load programs into VME processors
PARAMETERS	
file	string File containing exec code produced by PCP. or @file containing description file. File type VMEP is defaulted. For this case only qualifier /USER/SYSTEM is used.
procrate	integer VME processor crate NOT USED
processor	integer Processor offset 0-13 NOT USED
id	integer NOT USED Processor ID
node	J11 node (default=J11B) NOT USED
/TABLE	Load readout tables from same file.
/LOAD	Load modules (=default)
/RESET	Call I$ACV_RES_VME first to reset processor. The following qualifiers are not yet active:
/FEP	Processor is FEP

/EB	Processor is EB
/ROP	Processor is ROP
/CVI	Processor controls CAMAC crate with CVI processor
/CVC	Processor controls CAMAC crate with CVC processor
/CAV	Processor controls CAMAC crate with controller
/AEB	Processor controls Fastbus crate with AEB
/EBI	Processor controls EBI memory with foreign equipm.
/VME	Processor controls VME crate.
/SYNC	FEP runs in asynchron mode.
/USER	Load user routine specified in file.
/SYSTEM	Load system executive.
EXAMPLE	LOAD VME PROG @setup

Description

FUNCTION	Loads programs to VME processors. When the file is the exec program, either processor ID and subcrate or VME crate, offset and subcrate must be specified. These values must match with previously loaded processor.
File name	I$ACV_LOA_VME_P.PPL
Action rout.	I$ACV_LOA_VME_P
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

Syntax

The description file contain lines with the following syntax:

lines	<object> <specification>,<specification>,...
object	PROCESSOR — MODULE
specification	<key>=<value>
value	string — number — (<spec>,<spec>,...) — (number,number,...) Lines can be continued by a hyphen at the end. Comments are preceded by exclamation mark. Other files are included using a line @ <newfile>

PROCESSOR	Processor specification lines specify
	1. The software to be loaded
	2. The Bus and processor types
	BRANCH=(CRATE=c,OFFSET=p,ID=id),
	CONTROL=CAV—CVI—CVC—AEB—VME—EBI,
	TYPE=FEP—EB—ROP,
	MODE=SYNC—ASYNC, default=SYNC
	CRATE=c, not used
	SYSTEM=file, should be logical name
	USER=file not used
	All asynchron FEP's must be behind the synchron FEP's. First FEP is master and sets/clears GO bit in trigger module.

MODULE	Module specification lines specify 1. the location of a hardware module 2. the readout, reset or init information. These lines are ignored.

LOAD VME TABLE

```
LOAD VME TABLE file trigger VMEcrate,processor ID
               subcrate node log
               /[NO]LOAD
               /LOG
               /OVER
               /FEP/EB
               /ROP/ROC/AEB/VME
```

PURPOSE Load tables into VME processors. See also I$VMETAB

PARAMETERS

file	string File containing description file.
trigger	integer Trigger number for the table.
VMEcrate,processor	integer array VMEcrate,Processor offset 0-13
ID	integer Processor ID
node	ethernet node (default=ETH_O2)
logfile	optional log file
/LOG	Output tables contents
/LOAD	Load tables (=default) The following qualifiers are not yet active:
/OVER	Overwrite processor crate, offset in file by values specified in command.
/FEP	Processor is FEP
/EB	Processor is EB
/ROP	Processor controls CAMAC crate with processor

/ROC	Processor controls CAMAC crate with controller
/AEB	Processor controls Fastbus crate with AEB
/VME	Processor controls VME crate.
EXAMPLE	LOAD VME TAB @MYSETUP.VMEP

Description

FUNCTION	Loads tables to VME processors.
File name	I$ACV_LOA_VME_T.PPL
Action rout.	I$ACV_LOA_VME_T
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

Syntax

The description file contain lines with the following syntax:

lines <object> <specification>,<specification>,...

object PROCESSOR — MODULE

specification <key>=<value>

value string — number — (<spec>,<spec>,...) —
(number,number,...)
Lines can be continued by a hyphen at the end.
Comments are preceded by exclamation mark. Other files are included using a line @ <newfile>

PROCESSOR Processor specification lines specify
1. The software to be loaded
2. The Bus and processor types
3. The branch ID (for crate,offset)
BRANCH=(CRATE=c,OFFSET=p,ID=id),
CONTROL=ROP—ROC—AEB—VME,
Only the branch specification is used.

MODULE	Module specification lines specify
	1. the location of a hardware module
	2. the readout, reset or init information.
	BRANCH=(CRATE=c,OFFSET=p)—(ID=id),
	CONTROL=CAV—EBI—CVI—AEB—VME,
	NAME=name, NOT YET USED
	TYPE=type,CRATE=c,STATION—N=n,SUBADDRESS—A=a,
	INIT=(FUNCTION=f,REPEAT=r,EXEC=e,DATA=d,A=a),
	READ=(FUNCTION=f,REPEAT=r,EXEC=e,DATA=d),
	RESET=(FUNCTION=f,REPEAT=r,EXEC=e),
	CHANNEL=c,DATA=(v,v,...),DATA=@filespec
	For 16 <= functions <= 23 a data value must be
	specified with INIT/READ

FASTBUS_pedestals

Fastbus pedestals may be written in a text file with four numbers per line separated by commas.

low thresh,low ped,high thresh,high ped

This file must be specified with

DATA=@file

in the fastbus module line.

EXECUTION_codes

valid values are

ON*LY	execute only
X	execute and check X-response
Q	execute and check Q-response
Q1	execute until Q=1
Q0	execute until Q=0
XQ	execute and check X- and Q-response
XQ1	execute and check X until Q=1
XQ0	execute and check X until Q=0

CAMAC_types

the TYPE keyword is optional, can be replaced and
overwritten by INIT/READ/WRITE)

ST*ANDARD	for LRS2248a, Ortec AD811, AD1000 without FIFO, Borer scalers, ...
SI*4418	Silena T/V/Q ADC's
AD8*000	same as SI4418
CS*D24	Wenzel counter
AD1*000	FIFO on
PU*1000	same as AD1000 with FIFO on
LRS4302	
LRS4300	Fera 16 channel ADC
LRS2*261	image chamber analyzer

FASTBUS_types

supported are

LRS1872	LeCroy 64-channel TDC (12 bit)
LRS1875	LeCroy 64-channel TDC (12/15 bit)
LRS1882	LeCroy 96-channel ADC (12 bit)
LRS1885	LeCroy 96-channel ADC (12/15 bit)
KSCF432	Kinetic Systems 64-channel TDC
STR136	Struck 64-bit pattern unit
STR200	Struck 32-channel scaler (32 bit)

MOUNT TAPE

```
MOUNT TAPE device label blocksize density
                        /INITIALIZE
                        /DISMOUNT
                        /TK50 /TK70 /EXABYTE
```

PURPOSE	Mount RMS tape.
PARAMETERS	-
device	required string global replace default=" Tape device.
label	string Label of tape used for initialization.
blocksize	integer replaced default=24576 Blocksize should be multiple of buffersize (8192)
density	integer replaced default=6250 Tape density 800, 1600 or 6250 (depends on tape drive).
/INITIALIZE	switch Initialize tape before mounting. Write specified label to tape.
/DISMOUNT	switch Dismount a currently mounted tape and unload it. After that, the new tape must be mounted on the device. A prompt will give you the chance to do that and to continue.
/TK50	switch Tape is TK50.
/TK70	switch Tape is TK70.
/EXABYTE	switch Tape is EXABYTE.
EXAMPLE	MOUNT TAPE M1 xx001 /INIT

Description

FUNCTION	Mount a tape and initialize it optionally. GOOSY writes ANSI labeled tapes. The tape label is written to the tape during the initialization. If the tape is already initialized, the tape label is read and stored internally. It is used as default for GOOSY file headers in the START OUTPUT FILE command. With the /DISMOUNT qualifier a currently mounted tape is dismounted. Then a prompt is output to wait untill the new tape is mounted on the device.
File name	I$ACQ_MOUNT.PPL
Action rout.	I$ACQ_MOUNT
Version	1.01
Author	H.G.Essel
Last Update	04-Jan-1988

OPEN FILE

OPEN FILE filename device directory headerfile /[NO]HEADER

PURPOSE	Open file for data input stream.
PARAMETERS	-
filename	required string replace Name of listmode data file used for input.
device	string replaced Default device.
directory	string replaced Default directory.
headerfile	string Optional file to write file header.
/HEADER	switch default=/HEADER File has GOOSY header.
EXAMPLE	OPEN FILE j11.lmd DEV=M0:
NOTE	The acquisition must be initialized with /FILE

Description

FUNCTION	Opens a file for data input. The input is started by START ACQUIS. The data is processed as from other input channels. This input is used for multiple parallel working analysis programs on different nodes.
NOTE	The acquisition must be initialized with /FILE
File name	I$ACQ_OPE_FIL.PPL
Action rout.	I$ACQ_OPE_FIL
Dataset	-

Version	1.01
Author	H.G.Essel
Last Update	20-AUG-1987

OPEN OUTPUT FILE

```
OPEN OUTPUT FILE file size number
                 device directory
                 headerinput headeroutput
                 /PROMPT
                 /EDIT
                 /AUTOMATIC
                 /ALLOCATE
                 /PAGE /BYTE /KBYTE /BUFFER
```

PURPOSE Open list mode dump file

PARAMETERS

file required string replace
Name of the file to be opened. The default file type is .LMD. Device and directory are defaulted from the parameters DEVICE and DIRECTORY. Keep in mind filename comventions for IBM.

size integer replace default=35000
Size limit of the file to be written in Kbytes.
Other units can be selected by /PAGE/BYTE/BUFFER

number integer replace default=1000
Number of automatically written files (/AUTO)

device string replace
Default device

directory string replace
Default directory

headerinput string replace default=”
Optional file to read file header.

headeroutput string replace default=”
Optional file to store file header.

/PROMPT Prompt file header information

/EDIT	Edit file header (Headerinput required).
/AUTOMATIC	A three digit number is appended to the file name. If the file is filled, a new file is opened. The number is incremented.
/ALLOCATE	Preallocate file (disk only)
/PAGE	File size in pages (512 bytes)
/BYTE	File size in bytes
/KBYTE	File size in Kbytes (1024 bytes =default)
/BUFFER	File size in buffers
EXAMPLE	OPEN OUT FIL RUN026 DEV=M0:
NOTE	This command is called by START OUTPUT FILE which is the command one should use.

Description

FUNCTION	Open list mode file. If one wants to send the output files to the IBM, the filenames must follow some conventions: Maximal length 25 char (including type) Maximal 8 char or 7 digits between two _ File type is .LMD
File name	I$ACQ_OPE_LMD.PPL
Action rout.	I$ACQ_OPE_LMD
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

PATCH MBD

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

PURPOSE	Patch a MBD memory location
PARAMETERS	
ADDRESS=a	MBD memory address to be modified, be between 0 and 4095. NOTE, that you may enter the addresses in any radix, e.g. %O100 specifies 100 octal. This is a required parameter
VALUE=v	New value for memory location ADDRESS.
C=c	Crate number, if new value is in CNAF format.
N=n	Station number, if new value is in CNAF format.
A=a	Subaddress, if new value is in CNAF format.
F=f	Function code, if new value is in CNAF format. NOTE: Either VALUE or C must be specified.
/NOCONFIRM	Will bypass the confirmation question if specified. The default is the show the address, new and old value in decimal and octal radix and to request a confirmation.
FUNCTION	This procedure replaces the value in the MBD memory location ADDRESS by VALUE or the CNAF code specified with C,N,A and F.
EXAMPLE	PATCH MBD 234 12345
Action rout.	I$MCPAM
Author	Walter F.J. Mueller

Remarks

File name	I$MCPAM.PPL
Dataset	-
REMARKS	-

Description

CALLING @CALL I$MCPAM(I_ADDR,I_VALUE,I_C,I_N,I_A,I_F,
C_NOCONF,B_DP);

ARGUMENTS

I_ADDR BIN FIXED(15) [INPUT]
Address to be modified, must be between 0 and 4095.

I_VALUE BIN FIXED(15) [INPUT]
New value, will be used if not defaulted.

I_C BIN FIXED(15) [INPUT]
Crate number for a value specification as CNAF.
Will be used if not defauled. NOTE, that either I_VALUE or I_C must be specified explicitely.

I_N BIN FIXED(15) [INPUT]
Station number for a value specification as CNAF.

I_A BIN FIXED(15) [INPUT]
Subaddress for a value specification as CNAF.

I_F BIN FIXED(15) [INPUT]
Function code for a value specification as CNAF.

C_NOCONF CHAR(*) VAR [INPUT]
/NOCONFIRM qualifier. The confirmation question is skipped if this parameter is nonblank.

B_DP BIT(*) ALIGNED [INPUT]
Default pattern as passed by the $DP pseudo argument

FUNCTION This procedure replaces the value in the MBD memory location I_ADDR by I_VALUE or the CNAF code specified with I_C,I_N,I_A and I_F.

REMARKS -

EXAMPLE @CALL I$MCPAM(243,1234,0,0,0,0,'','00111100'B);

PATCH STARBURST

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

PURPOSE	Patch a STARBURST memory location
PARAMETERS	
ADDRESS=a	Address to be modified. The address is to be specified as a BYTE address but has to be word aligned (an even number) and in the range 0 to 131070 (the lower 64k words). NOTE, that you may enter the addresses in any radix, e.g. %O100 specifies 100 octal. This is a required parameter
VALUE=v	New value for memory location ADDRESS.
C=c	Crate number of the STARBURST to be loaded.
N=n	Station number of the STARBURST to be loaded.
/NOCONFIRM	Will bypass the confirmation question if specified. The default is the show the address, new and old value in decimal and octal radix and to request a confirmation.
FUNCTION	This procedure replaces the value in the STARBURST memory location ADDRESS by VALUE.
EXAMPLE	PATCH STARBURST %O620 1234 1 23
Action rout.	I$MCPAS
Author	Walter F.J. Mueller

Remarks

File name	I$MCPAS.PPL
Dataset	-
REMARKS	-

Description

CALLING @CALL I$MCPAS(L_ADDR,I_VALUE,I_C,I_N,
C_NOCONF);

ARGUMENTS

L_ADDR BIN FIXED(31) [INPUT]
Address to be modified. The address is to be
specified as a BYTE address but has to be word aligned (an even number) and in the range 0 to 131070 (the lower 64k words).

I_VALUE BIN FIXED(15) [INPUT]
New value, will be used if not defaulted.

I_C BIN FIXED(15) [INPUT]
Crate number of the STARBURST to be patched.

I_N BIN FIXED(15) [INPUT]
Station number of the STARBURST to be patched.

C_NOCONF CHAR(*) VAR [INPUT]
/NOCONFIRM qualifier. The confirmation question is
skipped if this parameter is nonblank.

FUNCTION This procedure replaces the value in the STARBURST memory location L_ADDR by I_VALUE. The STARBURST is located in crate I_C and station I_N.

REMARKS -

EXAMPLE @CALL I$MCPAS(610,1234,1,23,'');

RELEASE MBD CHANNEL

RELEASE MBD CHANNEL channel_no

PURPOSE	Release MBD channel to allow loading of new code
PARAMETERS	
CHANNEL_NO	Number of the MBD channel to be released
FUNCTION	Release a MBD channel. If a channel was used already and a certain code was loaded into the MBD for this channel, a new code for this channel can only be loaded if its size is smaller than or equal to the size of the old code. To allow loading of larger code on has to 'release' the channel first.
REMARKS	The command should only be used if a MBD channel code could not be loaded.
EXAMPLE	REL MBD CHAN 4
Action rout.	I$MCRMC
Author	M. Richter

Remarks

File name	I$MCRMC.PPL
Dataset	-

Description

CALLING	@CALL I$MCRMC(I_channel_no);
ARGUMENTS	
I_channel_no	I Number of MBD channel to be released. The channel number might be between 0 and 6 BIN FIXED(15)

FUNCTION Release a MBD channel. If a channel was used already and a certain code was loaded into the MBD for this channel, a new code for this channel can only be loaded if its size is smaller than or equal to the size of the old code. To allow loading of larger code on has to 'release' the channel first.

REMARKS -

EXAMPLE @CALL I$MCRMC(4);

RESET ACQUISITION

```
RESET ACQUISITION
```

EXAMPLE	RESET ACQU

Description

FUNCTION	
File name	I$ACQ_RES_ACQ.PPL
Action rout.	I$ACQ_RES_ACQ
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

RESET CAMAC

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

PURPOSE	Reset CAMAC
PARAMETERS	
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
node	optional node name of NET
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
EXAMPLE	

Description

FUNCTION	Send RESET CAMAC command .
File name	I$ACV_RES_CAM.PPL
Action rout.	I$ACV_RES_CAM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

RESET MBD

RESET MBD

PURPOSE	Reset MBD and release all active channels
PARAMETERS	
FUNCTION	Reset the MBD and release all active channels. All current operations of the MBD are stopped and the active VAX I/O channels using the MBD are aborted and thereby released, i.e. no VMS MWAIT condition should occure.
REMARKS	The command should only be used if the MBD hangs.
EXAMPLE	RESET MBD
Action rout.	I$MCRSM
Author	M. Richter

Remarks

File name	I$MCRSM.PPL
Dataset	-

Description

CALLING	@CALL I$MCRSM;
ARGUMENTS	
FUNCTION	Reset the MBD and release all active channels. All current operations of the MBD are stopped and the active VAX I/O channels using the MBD are aborted and thereby released, i.e. no VMS MWAIT condition should occure.
REMARKS	-
EXAMPLE	@CALL I$MCRSM();

SEND DATA

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

PURPOSE	Read one subevent.
PARAMETERS	
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
crate	Crate number
node	NET node
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
EXAMPLE	SEND DATA

Description

FUNCTION	Read one subevent.
File name	I$ACV_SEND_DATA.PPL
Action rout.	I$ACV_SEND_DATA
Dataset	-
Version	1.01

Author	H.G.Essel
Last Update	19-apr-1991

SET ACQUISITION

```
SET ACQUISITION in_buffers out_buffers events
                /[NO]SYNCHRONOUS
                /[NO]EXCLUSIVE
                /MAILBOX /NET
                /[NO]CHECK
                /[NO]COMPRESS
                /[NO]KEEP
                /[NO]START
                /[NO]STOP
```

PURPOSE Set data taking parameters.

PARAMETERS

in_buffers integer default=4
Number of buffers queued to MBD.
Default should be adequate.

out_buffers integer default=0
Number of buffers hold before writing to the output device is started.
Default should be adequate.

events integer replace default=0
Number of events to be filled into one buffer.
Zero means maximum (J11 only).

/SYNCHRONOUS Sychronous mode is entered. This means the Transport manager waits for the analysis. All buffers accepted by the transport manager are analyzed. The analysis may get the data via the first mailbox channel or via DECnet channel. The other channels are filled too, but not used for synchronization.

/EXCLUSIVE Enter exclusive mode. One buffer is sent only to one mailbox and one DECnet channel. Otherwise, a buffer may be sent to several analysis programs.

/MAILBOX Mailbox 1 is the controlling channel for synchronous mode (=default).

/NET DECnet 1 is the controlling channel for synchronous mode.

/[NO] CHECK Check buffer structure (default=/CHECK) /NOCHECK saves CPU time, but NO TYPE BUFFER or TYPE EVENT is possible.

/[NO] COMPRESS J11 only: events are written in compressed mode. Default is /NOCOMPRESS. Compress mode is useful only if less then 30% of the event parameters are non zero. The buffers and events are marked so that the analysis can use the appropriate unpack routine.

/[NO] KEEP J11 only: Keep data buffers (=default).

/[NO] START (de)activate calling of start module. The module must be loaded first by

LOAD MOD ACQ image module init /START

/[NO] STOP (de)activate calling of stop module. The module must be loaded first by

LOAD MOD ACQ image module init /STOP

The modules must be linked into a sharable image by DCL command LSHARIM.

EXAMPLE SET ACQU IN_BUF=6

Description

FUNCTION	Changes acquisition parameters initially defined by INIT ACQUIS command.
File name	I$ACQ_SET_ACQ.PPL
Action rout.	I$ACQ_SET_ACQ
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

SET FASTBUS PEDESTAL

```
SET FASTBUS PEDESTAL sample trigger
                VMEcrate,processor ID dummy crate node
                /ON/OFF [=ONOFF]
                /LOAD
                /ALL/FEP/EB [=DESTINATION]
                /CVI/CAV/EBI [=CONTROL]
```

PURPOSE Set fastbus pedestal subtraction on/off.

PARAMETERS

sample	integer (def=100) Sample interval [events]. After "sample" events one event will be not compressed.
trigger	integer (def=1) Trigger number
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
crate	Crate number
node	NET node
/ON/OFF	Switch compression ON or OFF
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD

EXAMPLE SET VME TRIG

Description

FUNCTION	Set fastbus pedestal subtraction on/off.
File name	I$ACV_SET_PED.PPL
Action rout.	I$ACV_SET_PED
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

SET RANDOM

```
SET RANDOM type subtype channels datawords
           /COMPRESS
           /PRINT
           /SPAN
```

PURPOSE	Set some random generator parameters

PARAMETERS

type	integer default=4 Data header type
subtype	integer default=1 Data header subtype
channels	integer default=4 channels per event
datawords	integer default=4000 number of data words used in buffer
/COMPRESS	switch Generate compressed events
/PRINT	switch Output events
/SPAN	swicth allow buffer spanning

Description

PURPOSE	Setup random generator of TMR.

SET VME BUFFER

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

PURPOSE	Setup frontend buffers
PARAMETERS	
buffers	Number of buffers.
size	Size of buffers (bytes)
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
node	optional node name of NET
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
/STOP/RESET	Break stopping status. Using VME frontends, the STOP ACQ command will enter stopping status which terminated by last buffer. When the frontend system does not send buffers stopping status would never be terminated. In this case, /STOP or /RESET can be used to terminate this status.

EXAMPLE

```
SET VME BUF 16 ID=20
SET VME BUF 0 16000 ID=20
```

Buffers are allocated in the free memory of the frontend processor. When specifying the number of buffers, the size will be calculated. When specifying the size, the number will be calculated. One of the two, buffers or size, must be zero (default).

Description

FUNCTION	Setup frontend buffers.
File name	I$ACV_SET_VME_BUF.PPL
Action rout.	I$ACV_SET_VME_BUF
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

SET VME CONTROL

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

PURPOSE Set value in control structure

PARAMETERS

name Name of parameter as seen in GOOVME(SS$VMECTRL).

FIC_DEBUG	0 no output 1 informational output 2 debug output

value Integer value.

VMEcrate,processor List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1

ID integer
Processor ID

dummy NOT used

node optional node name of NET

/ALL/FEP/EB Select processor

/CVI/CAV/EBI Select processor by controller

/[NO] LOAD Do [NOT]execute. Default= /LOAD

EXAMPLE SET VME CONT FIC_DEBUG 1 /ALL

Description

FUNCTION	Set value in control structure
File name	I$ACV_SET_VME_CTRL.PPL
Action rout.	I$ACV_SET_VME_CTRL
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

SET VME INPUT

```
SET VME INPUT
              /HVR /NET /TDAS /OFF /CHECK [=PATH]
```

PURPOSE	Select lmd data path
PARAMETERS	
/HVR	Use HVR interface.
/NET	Use NET interface.
/TDAS	Send buffers to TDAS event builder.
/OFF	Switch off
/CHECK	Check buffer structure only.
EXAMPLE	SET VME INP /HVR

Description

FUNCTION	Select lmd data path.
File name	I$ACV_SET_VME_INP.PPL
Action rout.	I$ACV_SET_VME_INP
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

SET VME TRIGGER

```
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]
```

PURPOSE	Set trigger module.
PARAMETERS	
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
crate	Crate number
fastclear	Time for fast clear
conversion	Conversion time
node	NET node
/RESET	Reset trigger module
/MASTER	Set master
/ENABLE	Enable trigger
/DISABLE	Disable trigger
node	optional node name of NET
/ALL/FEP/EB	Select processor

/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD

EXAMPLE	SET VME TRIG

Description

FUNCTION	Set trigger module.
File name	I$ACV_SET_TRIG.PPL
Action rout.	I$ACV_SET_TRIG
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

SHOW ACQUISITION

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

PURPOSE	Show acquisition status
PARAMETERS	
timer	integer optional time intervall [sec]for /RATE default is 5 sec.
output	string optional output file
/PRINT	Print output
/CLEAR	Clear counter
/RUN	Show run information
/SETUP	Show VME setup information
/OUTFILE	Show current output file header
/INFILE	Show current input file header
/BRIEF	Brief output
/[NO] RATE	Show data rate
EXAMPLE	SHO ACQU

Description

FUNCTION	Acquisition parameters are output.
File name	I$ACQ_SHO_ACQ.PPL
Action rout.	I$ACQ_SHO_ACQ
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

SHOW STARBURST

SHOW STARBURST C=c N=n

PURPOSE	Show the execution parameters of a STARBURST.
PARAMETERS	
C=c	Crate number of the STARBURST to be accessed. The default is 1.
N=n	Crate number of the STARBURST to be accessed. The default is 23.

FUNCTION

This command displays the execution parameters of a STARBURST processor running the simple data aquisition executive.
The command reads the vector and communication area located between %O600 and %O700. The first 8 words are used for status variables and pointers shared between MBD and STARBURST:

%O600 :	New status
%O602 :	Old status
%O604 :	Address of a valid event buffer
%O606 :	Subsystem index
%O610 :	Address of the first parameters set
%O612 :	Length of the first parameter set.

The procedure displays the current status, prints a warning if the subsystem index differs from the crate number, prints the last event buffer if there is a valid one and lists the declared parameter sets.

EXAMPLE	DUMP STARBURST C=1 N=23
Action rout.	I$MCSHS
Author	Walter F.J. Mueller

Remarks

File name	I$MCSHS.PPL
Dataset	-
REMARKS	-

Description

CALLING @CALL I$MCSHS(I_C,I_N);

ARGUMENTS

I_C BIN FIXED(15) [INPUT]
Crate number of STARBURST to be accessed.

I_N BIN FIXED(15) [INPUT]
Station number of STARBURST to be accessed.

FUNCTION Show the execution parameters of a STARBURST. It dump some locations in the vector page. For details look in the command description.

REMARKS -

EXAMPLE @CALL I$MCSHS(1,23);

SHOW VME CONTROL

```
SHOW VME CONTROL name VMEcrate,processor ID
                 dummy node
                 /LOAD
                 /ALL/FEP/EB [=DESTINATION]
                 /CVI/CAV/EBI [=CONTROL]
```

PURPOSE	Show values in control structure
PARAMETERS	
name	Name of parameter as seen in GOOVME(SS$VMECTRL). If not specified, the whole structure is displayed.
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
node	optional node name of NET
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
EXAMPLE	SET VME CONT FIC_DEBUG 1 /ALL

Description

FUNCTION	Show values in control structure
File name	I$ACV_SHO_VME_CTRL.PPL
Action rout.	I$ACV_SHO_VME_CTRL

Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

SHOW VME SETUP

SHOW VME SETUP file /FULL

PURPOSE	Show VME setup.
PARAMETERS	
file	Optional output file
/FULL	Show full information.
EXAMPLE	SHO VME SET /F

Description

FUNCTION	-
File name	I$ACVME.PPL
Action rout.	I$ACVME_SHO_VME
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	20-Jul-1989

START ACQUISITION

```
START ACQUISITION buffers events
                  skip_buf skip_event
                  /CLEAR /NET /STOP /RESET
```

PURPOSE Start data taking

PARAMETERS

buffers	integer default=0 Number of buffers to process (0 = infinite) (file input only)
events	integer default=0 Number of events to process (0 = infinite) (file input only)
skip_buf	integer default=0 Number of buffers to skip (file input only)
skip_event	integer default=0 Number of events to skip (file input only)
/NET	switch Start input from net (VME)
/CLEAR	switch Clear counter. Same effect as SHOW/CLEAR. Some counters are cleared by START ACQUISITION, others by SHOW/CLEAR or START ACQ/CLEAR.
/STOP/RESET	switch Break stopping status. Using VME frontends, the STOP ACQ command will enter stopping status which terminated by last buffer. When the frontend system does not send buffers stopping status would never be terminated. In this case, /STOP or /RESET can be used to terminate this status.

EXAMPLE STA ACQU

Description

FUNCTION	This procedure starts the data acquisition. The MBD and all front end processors are started. MBD is initialized by I$ACQ_IMBD. Input is queued by delivering I$ACQ_QMBD on AST level. For a J11 system the J11 is started. The buffer and event counters are cleared.
File name	I$ACQ_STA_ACQ.PPL
Action rout.	I$ACQ_STA_ACQ
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

START OUTPUT FILE

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

PURPOSE	Start list mode dump
PARAMETERS	-
file	required string replace File name for new file, if required. Keep in mind filename comventions for IBM.
size	integer replace default=35000 Size of new file in Kbytes, if required Units can be selected by /PAGE/BYTE/BUFFER.
number	integer replace default=1000 Number of files written automatically. (used for /AUTO)
device	string replace Default device
directory	string replace Default directory
headerinput	string replace default=” Optional file to read file header.
headeroutput	string replace default=” Optional file to store file header.
/PROMPT	Prompt file header information

/EDIT	Edit file header (headerinput required).
/NOOPEN	continue output in current file This can be done only if a previous STOP OUT FILE /NOCLOSE was given. Once a file is closed, it cannot be continued.
/OPEN	open new file (close current) (default)
/AUTOMATIC	A new file is opened, if the previous one is filled. A three digit current number is appended to the filename. The /OPEN switch is required.
/ALLOCATE	Preallocate file (disk only)
/PAGE	File size in pages (512 bytes)
/BYTE	File size in bytes
/KBYTE	File size in Kbytes (1024 bytes =default)
/BUFFER	File size in buffers
EXAMPLE	STA OUT FIL /NOCLOSE (continue current open file) STA OUT FIL x.lmd (start new file) If one wants to send the output files to the IBM, the filenames must follow some conventions: Maximal length 25 char (including type) Maximal 8 char or 7 digits between two underscore No dollar signs. File type is .LMD

Description

FUNCTION	Start list mode dump to file.
File name	I$ACQ_STA_LMD.PPL
Action rout.	I$ACQ_STA_LMD
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

START RUN

START RUN name

PURPOSE	Start run.
PARAMETERS	
name	Run name or @filename containing run information. The file is supposed to be ASCII, 80 char/line.
EXAMPLE	STA RUN @RUN_AU_173.RUN

Description

FUNCTION	This procedure starts a run. The run name or filename is stored in the control structure. All frontend processors get the command. With the SHOQ ACQUIS command the content of the run file may be displayed.
File name	I$ACQ_STA_RUN.PPL
Action rout.	I$ACQ_STA_RUN
Version	1.01
Author	H.G.Essel
Last Update	4-Sep-1991

START VME

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

PURPOSE	Send START command to NET
PARAMETERS	
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
node	optional node name of NET
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
EXAMPLE	START VME 1,1

Description

FUNCTION	Send START command .
File name	I$ACV_STA_VME.PPL
Action rout.	I$ACV_STA_VME
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

STOP ACQUISITION

STOP ACQUISITION /ABORT /CLOSE /STOP/RESET

PURPOSE	Stop data taking
PARAMETERS	-
/ABORT	Delete links to J11 and cleanup (J11 only). All buffers in the J11 are lost.
/CLOSE	Close input file (File input only)
/STOP/RESET	Do not wait for last buffer.
EXAMPLE	STOP ACQU
FUNCTION	With the MBD as frontend all data in the MBD is sent to the VAX. All data is written to file, if output is active. With a single crate J11 system the /ABORT qualifier clears all data in the J11.

Description

FUNCTION	MBD inputs are canceled. J11 is stopped. Mailbox QIO's are canceled.
File name	I$ACQ_STO_ACQ.PPL
Action rout.	I$ACQ_STO_ACQ
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

STOP OUTPUT FILE

```
STOP OUTPUT FILE
     /[NO]CLOSE
```

PURPOSE	Stop list mode dump
PARAMETERS	-
/NOCLOSE	Keep file open to continue with STA OUT FILE /NOOP
/CLOSE	Close file (NO append possible) (default)
EXAMPLE	STOP OUT FILE (file is closed)

Description

FUNCTION	Stop list mode dump to file. Optional the file is closed.
File name	I$ACQ_STO_LMD.PPL
Action rout.	I$ACQ_STO_LMD
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

STOP RUN

STOP RUN /STOP /ABORT /CLOSE

PURPOSE	Stop run.
PARAMETERS	-
/STOP	Stop acquisition first.
/ABORT	With /STOP: Delete links to J11 and cleanup (J11). All buffers in the J11 are lost.
/CLOSE	With /STOP: Close input file (File input only)
EXAMPLE	STOP RUN
FUNCTION	Sends STOP RUN command to frontend systems (VME). When /STOP is given, stops acquisition first and closes output file. The /ABORT and /CLOSE switches are used only with /STOP

Description

FUNCTION	Stops run. Optionally stop acquisition first.
File name	I$ACQ_STO_RUN.PPL
Action rout.	I$ACQ_STO_RUN
Version	1.01
Author	H.G.Essel
Last Update	4-Sep-1991

STOP VME

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

PURPOSE	Send STOP command to NET
PARAMETERS	
VMEcrate,processor	List of processor specifications, i.e. 1,0,1,1,1,2 for processors with offets 0,1,2 in VME crate 1
ID	integer Processor ID
dummy	NOT used
node	optional node name of NET
/ALL/FEP/EB	Select processor
/CVI/CAV/EBI	Select processor by controller
/[NO] LOAD	Do [NOT]execute. Default= /LOAD
EXAMPLE	STOP VME 1,1

Description

FUNCTION	Send STOP command .
File name	I$ACV_STO_VME.PPL
Action rout.	I$ACV_STO_VME
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	16-feb-1989

STORE LRS_2365

```
STORE LRS_2365 C=c N=n FILE=file
               /DUMP/NODUMP
```

PURPOSE	Read back definitions from a LRS 2365 logic matrix and write them formated to a file (or SYS$OUTPUT).
PARAMETERS	
C=c	Crate number of the logic matrix to be accessed.
N=n	Station number of the logic matrix to be accessed.
FILE=file	Definition file to be written. The default file type is DAT. The format is as described for the LOAD LRS_2365 command. The default is SYS$OUTPUT.
/DUMP	Dumps the contents of the LRS 2365 module found after the down load. The programming words PW 0 to 17 are shown in binary radix. For their interpretation look in the LRS manual.
/NODUMP	Don't dump programming words, this is the default.
FUNCTION	This commands reads the setup of a LRS 2365 logic matrix back and generates a definition file. This file is in the format described for the LOAD LRS_2365 command and may be used at a later time with this command. This command also serves as a DUMP type command if the FILE parameter is omitted. The definition file is written to SYS$OUTPUT in this case.
EXAMPLE	STORE LRS_2365 1 1
Action rout.	I$STLRS_2365
Author	Walter F.J. Mueller

Remarks

File name	I$STLRS_2365.PPL
Dataset	-
REMARKS	-

Description

CALLING @CALL I$STLRS_2365(I_C,I_N,C_FILE,C_DUMP);

ARGUMENTS

I_C BIN FIXED(15) [INPUT]
Crate number of the module to be loaded.

I_N BIN FIXED(15) [INPUT]
Station number of the module to be loaded.

C_FILE CHAR(*) VAR [INPUT]
Name of definition file to be written.

C_DUMP CHAR(*) [INPUT]
DUMP qualifier set:

/DUMP	Dump programming words.
/NODUMP	Don't dump programming words.

FUNCTION Reads the setup of a LRS 2365 logic matrix and writes a formated definition file.

REMARKS -

EXAMPLE @CALL I$STLRS_2365(1,1,'SYS$OUTPUT','');

STORE MBD

STORE MBD file

PURPOSE	Store a MBD memory dump in a file.
PARAMETERS	
file	Name of the dump file to be written. The default file type is .BDD. The file will contain 16 records with 512 bytes each with the binary image of the whole MBD memory. This file can be read and dumped with the DUMP MBD command.
FUNCTION	This procedure writes a dump of the whole MBD memory to a file. This may be usefull after an MBD error or other unexpected malfunctions. The file can be read later with the DUMP MBD command and may allow to reconstuct the source of trubble.
EXAMPLE	STORE MBD MYDUMP
Action rout.	I$MCSTM
Author	Walter F.J. Mueller

Remarks

File name	I$MCSTM.PPL
Dataset	-
REMARKS	-

Description

CALLING	@CALL I$MCSTM(C_FILE);
ARGUMENTS	

C_FILE	CHAR(*) VAR [INPUT] Name of the dump file to be written. The default file type is .BDD.
FUNCTION	This procedure reads the whole 4 kwords MBD memory and writes it in a dump file. This file has 16 records with 512 bytes each containing the binary memory image.
REMARKS	-
EXAMPLE	@CALL I$MCSTM('MYDUMP');

TEST BOR_1802

```
TEST BOR_1802
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
                /FULL
```

PURPOSE Perform tests with a BORER 1802 Dataway display

PARAMETERS

B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

/FULL A full loopback test will be done in every test cycles. This is especially usefull for 'online' debugging of branch cables ect. because each bit of the CAMAC system is tested in all cycles thus providing a fast response for intermittend errors.

FUNCTION This test uses a Dataway display (either a BORER 1802 or compatible types) as a loopback mirror and tries to verify the datatransfer thru the branch driver, the branch cables, the crate controller the Dataway and back. This is done by sending test patterns to the Dataway display and reading them back. The full test algorithm works as follows:

Set DWD to online mode to aviod contentions with active auxiliary crate controllers.

Generate a new test pattern (for details of look in the description of I$GTPAT).

Send the test pattern to the DWD, read back, compare and check X-Q responses.

Set DWD back to monitor mode.
If the /FULL qualifier has been specified, a set of 532 test pattern is generated and processed in all test cycle. In this mode tests each bit in the CAMAC system in all cycles.

REMARKS The LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

EXAMPLE TEST BOR_1802 C=1 N=22 /START

Action rout. I$CTBOR_1802

Author Walter F.J. Mueller

Remarks

File name I$CTBOR_1802.PPL

Dataset -

Description

CALLING

```
@CALL I$CTBOR_1802(I_B,I_C,I_N,F_DELTA,
                   C_OPTION,C_FULL);
```

ARGUMENTS

I_B	BIN FIXED(15) [INPUT] Branch number of the module to be tested.
I_C	BIN FIXED(15) [INPUT] Crate number of the module to be tested.
I_N	BIN FIXED(15) [INPUT] Station number of the module to be tested.
F_DELTA	BIN FLOAT(24) [INPUT] Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..
C_OPTION	CHAR(*) VAR [INPUT] Test option qualifier set, expected values are: /LIST/STOP /RUN/START/LOOP
C_FULL	CHAR(*) VAR [INPUT] /FULL qualifier. If specified, all test patters are written and read in every test interation.

FUNCTION This procedure performs Dataway loopback tests with a BORER 1802 Dataway display as mirror. For a more detailed discussion look in the description of the command TEST BOR_1802.

REMARKS -

EXAMPLE @CALL I$CTBOR_1802(0,1,22,0.E0,'/RUN','/FULL');

TEST CAMAC

```
TEST CAMAC B=b C=c N=n TYPE=*
   /STOP
   /LIST
```

PURPOSE	Common functions for CAMAC tests.
PARAMETERS	
B=b	Branch for which tests are to be affected. Tests in all branches are affected if omitted or specified as 0 (default = 0).
C=c	Crate for which tests are to be affected. Tests in all crates are affected if omitted or specified as 0 (default = 0).
N=n	Station number for which test are to be affected. Tests in all stations are affected if omitted or specified as 0 (default = 0).
TYPE=t	Type of tests to be affected. Tests of all types are affected if omitted or specified as '*'.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
EXAMPLE	TEST CAMAC /LIST TEST CAMAC TYPE=BOR_1802 /STOP
REMARKS	The command will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.
Action rout.	I$CTCAMAC
Author	Walter F.J. Mueller

Remarks

File name	I$CTCAMAC.PPL
Dataset	-
REMARKS	-

Description

CALLING @CALL I$CTCAMAC(I_B,I_C,I_N,C_TYPE,C_OPTION);

ARGUMENTS

I_B BIN FIXED(15) [INPUT]
Crate number of the module to be tested.

I_C BIN FIXED(15) [INPUT]
Crate number of the module to be tested.

I_N BIN FIXED(15) [INPUT]
Station number of the module to be tested.

C_TYPE CHAR(*) VAR [INPUT]
Type of the tests to be selected, either a '*' or the name of test.

C_OPTION CHAR(*) VAR [INPUT]
Test option qualifier set, expected values are:
/LIST/STOP

FUNCTION This procedure performs some functions acting on all or a group of CAMAC tests. For a more detailed discussion look in the description of the command TEST CAMAC.

REMARKS -

EXAMPLE @CALL I$CTCAMAC(0,0,0,'*','/STOP');

TEST GSI_IOL

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

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

PARAMETERS

B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

/LOOPBACK	Signals, that the external loopback connections have been established and may be used in the test.
FUNCTION	This test performs some manipulations with a IOL module which allow a test with a scope. The full test algorithm works as follows:
EXAMPLE	TEST GSI_IOL C=1 N=12 /START
REMARKS	The LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.
Action rout.	I$CTGSI_IOL
Author	Walter F.J. Mueller

Remarks

File name	I$CTGSI_IOL.PPL
Dataset	-
REMARKS	-

Description

CALLING @CALL I$CTGSI_IOL(I_B,I_C,I_N,F_DELTA,C_OPTIONS,
C_LOOPBACK);

ARGUMENTS

I_B BIN FIXED(15) [INPUT]
Branch number of the module to be tested.

I_C BIN FIXED(15) [INPUT]
Crate number of the module to be tested.

I_N BIN FIXED(15) [INPUT]
Station number of the module to be tested.

F_DELTA BIN FLOAT(24) [INPUT]
Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..

C_OPTION CHAR(*) VAR [INPUT]
Test option qualifier set, expected values are:
/LIST/STOP /RUN/START/LOOP

C_LOOPBACK CHAR(*) VAR [INPUT]
/LOOPBACK qualifier. Signals, that the external
loopback connections have been established and may be used in the test.

FUNCTION This procedures performs a selftest for a GSI IOL module. For a more detailed discussion look in the description of the command TEST GSI_IOL.

REMARKS -

EXAMPLE @CALL I$CTGSI_IOL(0,1,12,0.E0,'/RUN',");

TEST LRS_2228

```
TEST LRS_2228
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
    /VALUE
```

PURPOSE	Test a LRS 2228 TDC module.
PARAMETERS	
B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

/VALUE The values of the test conversion are written to SYS$OUTPUT.

FUNCTION This test exploits the F(25) self test function of a LRS 2228 TDC to perform a simple go/nogo test. The full test algorithm works as follows:

Clear the module with a F(9) function, read all channels and check that they contain a zero.

Perform a test conversion with a F(25) function, wait for the conversion by doing 60 CAMAC accesses, read all channels with a F(2) function and check that they contains neither a zero nor an overflow value. If the /VALUE qualifier has been specified, the test conversion values are written to SYS$OUTPUT and may be used to check the homogenety of the conversion factor.

Read again all channels and check that the previous F(2)A(7) read has actually zeroed all channels.

EXAMPLE TEST LRS_2228 C=1 N=12 /START

REMARKS The LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

Action rout. I$CTLRS_2228

Author Walter F.J. Mueller

Remarks

File name I$CTLRS_2228.PPL

Dataset -

REMARKS -

Description

CALLING

```
@CALL I$CTLRS_2228(I_B,I_C,I_N,F_DELTA,C_OPTIONS,
                   C_VALUE);
```

ARGUMENTS

I_B BIN FIXED(15) [INPUT]
Branch number of the module to be tested.

I_C	BIN FIXED(15) [INPUT] Crate number of the module to be tested.
I_N	BIN FIXED(15) [INPUT] Station number of the module to be tested.
F_DELTA	BIN FLOAT(24) [INPUT] Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..
C_OPTION	CHAR(*) VAR [INPUT] Test option qualifier set, expected values are: /LIST/STOP /RUN/START/LOOP
C_VALUE	CHAR(*) VAR [INPUT] /VALUE qualifier. If specified, the values of the test conversion are written to SYS$OUTPUT.

FUNCTION This procedures performs a selftest for a LRS 2228 TDC. For a more detailed discussion look in the description of the command TEST LRS_2228.

REMARKS -

EXAMPLE @CALL I$CTLRS_2228(0,1,12,0.E0,'/RUN','/VALUE');

TEST LRS_2249

```
TEST LRS_2249
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
    /PEDESTAL
```

PURPOSE Test a LRS 2249 ADC module.

PARAMETERS

B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

/PEDESTAL	The values of the pedestal and calibration values of the test conversion are written to SYS$OUTPUT.

FUNCTION This test exploits the F(25) self test function of a LRS 2249 ADC to perform a simple go/nogo test. The full test algorithm works as follows:

Clear the module with a F(9) function, read all channels and check that they contain a zero.

Perform a test conversion with a F(25) function.
This will trigger a conversion with an internally generated gate but no input current thus yielding a pedestal value. Wait for the conversion by doing 60 CAMAC accesses, read all channels with a F(2) function and check that they contain a reasonable pedestal value.

Set the dataway INHIBIT and perform another test conversion with a F(25) function. This will trigger a conversion with an internally generated test current thus yielding a rough estimate of the channel calibration. Again wait for the conversion by doing 60 CAMAC accesses, read all channels with a F(2) function and check that they contain a reasonable value.
If the /PEDESTAL qualifier has been specified, the the pedestal and test conversion values are written to SYS$OUTPUT. Be aware, that the pedestal values are for the internally generated gate with and may different in the experimental setup!

Read again all channels and check that the previous F(2)A(11) read has actually zeroed all channels.

EXAMPLE	TEST LRS_2249 C=1 N=12 /START
REMARKS	The LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.
Action rout.	I$CTLRS_2249
Author	Walter F.J. Mueller

Remarks

File name	I$CTLRS_2249.PPL
Dataset	-

Description

CALLING	@CALL I$CTLRS_2249(I_B,I_C,I_N,F_DELTA,C_OPTIONS, C_PEDESTAL);
ARGUMENTS	
I_B	BIN FIXED(15) [INPUT] Branch number of the module to be tested.
I_C	BIN FIXED(15) [INPUT] Crate number of the module to be tested.
I_N	BIN FIXED(15) [INPUT] Station number of the module to be tested.
F_DELTA	BIN FLOAT(24) [INPUT] Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..
C_OPTION	CHAR(*) VAR [INPUT] Test option qualifier set, expected values are: /LIST/STOP /RUN/START/LOOP
C_PEDESTAL	CHAR(*) VAR [INPUT] /PEDESTAL qualifier. If specified, the pedestal and calibration values of the test conversion are written to SYS$OUTPUT.
FUNCTION	This procedures performs a selftest for a LRS 2249 ADC. For a more detailed discussion look in the description of the command TEST LRS_2249.
REMARKS	-
EXAMPLE	@CALL I$CTLRS_2249(0,1,12,0.E0,'/RUN','PEDESTAL');

TEST LRS_2551

```
TEST LRS_2551
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
```

PURPOSE	Test a LRS 2551 scaler module.
PARAMETERS	
B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

FUNCTION This test exploits the F(25) self test function of a LRS 2551 scaler to perform a simple go/nogo test. The full test algorithm works as follows:

Clear the module with a F(9) function, read all channels and check that they contain a zero.

Issue a test with a F(25) function, read all channels and check whether they have been incremented once.

Issue further 32 tests with a F(25) function, read all channels and check whether they contain now the value 33.

Clear the module with a F(9) function, read all channels and check that they contain again zero.

EXAMPLE TEST LRS_2551 C=1 N=12 /START

REMARKS The LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

Action rout. I$CTLRS_2551

Author Walter F.J. Mueller

Remarks

File name I$CTLRS_2551.PPL

Dataset -

Description

CALLING @CALL I$CTLRS_2551(I_B,I_C,I_N,F_DELTA,C_OPTIONS);

ARGUMENTS

I_B BIN FIXED(15) [INPUT]
Branch number of the module to be tested.

I_C BIN FIXED(15) [INPUT]
Crate number of the module to be tested.

I_N BIN FIXED(15) [INPUT]
Station number of the module to be tested.

F_DELTA	BIN FLOAT(24) [INPUT] Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..
C_OPTION	CHAR(*) VAR [INPUT] Test option qualifier set, expected values are: /LIST/STOP /RUN/START/LOOP

FUNCTION This procedures performs a selftest for a LRS 2551 scaler. For a more detailed discussion look in the description of the command TEST LRS_2551.

REMARKS -

EXAMPLE @CALL I$CTLRS_2551(0,1,12,0.E0,'/RUN');

TEST LRS_4432

```
TEST LRS_4432
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
```

PURPOSE Test a LRS 4432 scaler module.

PARAMETERS

B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

FUNCTION

This test exploits the F(25) self test function of a LRS 4432 scaler to perform a simple go/nogo test. The full test algorithm works as follows:

Clear the module by writing the clear bit in the CSR, than load and read all channels by writing the load and read bit in the CSR and reading 32 words. Check, that all channels are zeroed.

Issue a test by writing the test bit in the CSR, load and read all channels and check whether all bytes of all channels have been incremented once. For details of the test function look in the 4432 manual.

Issue further 254 tests, load and read all channels. Check, the all channels return a word with all 24 bits set. (Each byte has been incremented 255 times and thus should contain '11111111'B).

Clear the module, load and read all channels. Check, that all channels are again zeroed.

EXAMPLE

TEST LRS_4432 C=1 N=12 /START

REMARKS

The LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

Action rout. I$CTLRS_4432

Author Walter F.J. Mueller

Remarks

File name I$CTLRS_4432.PPL

Dataset -

Description

CALLING

@CALL I$CTLRS_4432(I_B,I_C,I_N,F_DELTA,C_OPTIONS);

ARGUMENTS

I_B

BIN FIXED(15) [INPUT]
Branch number of the module to be tested.

I_C	BIN FIXED(15) [INPUT] Crate number of the module to be tested.
I_N	BIN FIXED(15) [INPUT] Station number of the module to be tested.
F_DELTA	BIN FLOAT(24) [INPUT] Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..
C_OPTION	CHAR(*) VAR [INPUT] Test option qualifier set, expected values are: /LIST/STOP /RUN/START/LOOP

FUNCTION This procedures performs a selftest for a LRS 4432 scaler. For a more detailed discussion look in the description of the command TEST LRS_4432.

REMARKS -

EXAMPLE @CALL I$CTLRS_4432(0,1,12,0.E0,'/RUN');

TEST LRS_4434

```
TEST LRS_4434
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
```

PURPOSE Test a LRS 4434 scaler module.

PARAMETERS

B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

FUNCTION	This test exploits the F(25) self test function of a LRS 4434 scaler to perform a simple go/nogo test. The full test algorithm works as follows: Clear the module by writing the clear bit in the CSR, than load and read all channels by writing the load and read bit in the CSR and reading 32 words. Check, that all channels are zeroed. Issue a test by writing the test bit in the CSR, load and read all channels and check whether all bytes of all channels have been incremented once. For details of the test function look in the 4432 manual. Issue further 254 tests, load and read all channels. Check, the all channels return a word with all 24 bits set. (Each byte has been incremented 255 times and thus should contain '11111111'B). Clear the module, load and read all channels. Check, that all channels are again zeroed.
Action rout.	I$CTLRS_4434
Author	Walter F.J. Mueller

Remarks

File name	I$CTLRS_4432.PPL
Dataset	-
REMARKS	-
EXAMPLE	TEST LRS_4434 C=1 N=12 /START

Description

CALLING	@CALL I$CTLRS_4434(I_B,I_C,I_N,F_DELTA,C_OPTIONS);
ARGUMENTS	
I_B	BIN FIXED(15) [INPUT] Branch number of the module to be tested.

I_C BIN FIXED(15) [INPUT]
Crate number of the module to be tested.

I_N BIN FIXED(15) [INPUT]
Station number of the module to be tested.

F_DELTA BIN FLOAT(24) [INPUT]
Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..

C_OPTION CHAR(*) VAR [INPUT]
Test option qualifier set, expected values are:
/LIST/STOP /RUN/START/LOOP

FUNCTION This procedures performs a selftest for a LRS 4434 scaler. For a more detailed discussion look in the description of the command TEST LRS_4434.

REMARKS -

EXAMPLE @CALL I$CTLRS_4434(0,1,12,0.E0,'/RUN');

TEST MPI_BIT

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

PURPOSE Test a MPI bit encoder module.

PARAMETERS

B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

FUNCTION This command performs a simple test for a MPI bit encoder module. The full test algorithm works as follows:

Generate a test pattern and write it to the bit encoder.

Read back the bit multiplicity and 18 bit numbers. Than check, whether the multiplicity and the bit numbers are correct and whether the correct stop word has been generated by the bit encoder.

EXAMPLE TEST MPI_BIT C=1 N=12 /START

REMARKS The LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

Action rout. I$CTMPI_BIT

Author Walter F.J. Mueller

Remarks

File name I$CTMPI_BIT.PPL

Dataset -

Description

CALLING @CALL I$CTMPI_BIT(I_B,I_C,I_N,F_DELTA,C_OPTIONS);

ARGUMENTS

I_B BIN FIXED(15) [INPUT]
Branch number of the module to be tested.

I_C BIN FIXED(15) [INPUT]
Crate number of the module to be tested.

I_N BIN FIXED(15) [INPUT]
Station number of the module to be tested.

F_DELTA BIN FLOAT(24) [INPUT]
Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..

C_OPTION CHAR(*) VAR [INPUT]
Test option qualifier set, expected values are:
/LIST/STOP /RUN/START/LOOP

FUNCTION This procedures performs a selftest for a MPI bit encoder. For a more detailed discussion look in the description of the command TEST MPI_BIT.

REMARKS -

EXAMPLE @CALL I$CTMPI_BIT(0,1,12,0.E0,'/RUN');

TEST MPI_TDC

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

PURPOSE Test a MPI slow TDC module.

PARAMETERS

B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

/VALUE The values of the time intervall between clear and strobe is written to SYS$OUTPUT.

FUNCTION This test exploits the F(16)A(0,1) self test functions of a MPI slow TDC to perform a simple go/nogo test. The full test algorithm works as follows:

Clear the TIME with a F(16)A(1) function.

Strobe the TIME with a F(16)A(0) function.

Read the TIME with a F(0)A(0) function and check whether a reasonable value has been returned. If the /VALUE qualifier has been specified, this time is written to SYS$OUTPUT. Because this is the time between the clear and the strobe action done with a single I$CFGA call, is is independant of the speed and load of the host VAX but a measure of the MBD performance.

EXAMPLE TEST MPI_TDC C=1 N=12 /START

REMARKS The LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

Action rout. I$CTMPI_TDC

Author Walter F.J. Mueller

Remarks

File name I$CTMPI_TDC.PPL

Dataset -

Description

CALLING

```
@CALL I$CTMPI_TDC(I_B,I_C,I_N,F_DELTA,C_OPTIONS,
                  C_VALUE);
```

ARGUMENTS

I_B BIN FIXED(15) [INPUT]
Branch number of the module to be tested.

I_C	BIN FIXED(15) [INPUT] Crate number of the module to be tested.
I_N	BIN FIXED(15) [INPUT] Station number of the module to be tested.
F_DELTA	BIN FLOAT(24) [INPUT] Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..
C_OPTION	CHAR(*) VAR [INPUT] Test option qualifier set, expected values are: /LIST/STOP /RUN/START/LOOP
C_VALUE	CHAR(*) VAR [INPUT] /VALUE qualifier. If specified, the time intervall between clear and strobe is written to SYS$OUTPUT.

FUNCTION This procedures performs a selftest for a MPI slow TDC. For a more detailed discussion look in the description of the command TEST MPI_TDC.

REMARKS -

EXAMPLE @CALL I$CTMPI_TDC(0,1,12,0.E0,'/RUN','/VALUE');

TEST REGISTER

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

PURPOSE	Perform tests with a any register in a CAMAC module.
PARAMETERS	
B=b	Number of the branch of the module to be tested.
C=c	Number of the crate of the module to be tested.
N=n	Station number of the module to be tested.
A=a	Number of the subaddress in the module to be tested.
F=f	Function in the module to be tested.
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed.

WIDTH=w The width of the register to be tested in bits. May be between 2 and 24, the default is 16.

/FULL A full set of test patterns is written and read back in every test cycle. The default is one test pattern per cycle. The /FULL mode is usefull to check for infrequent, intermittent problems.

FUNCTION This test uses any read/write register in a CAMAC module as a loopback mirror and tries to verify the datatransfer thru the branch driver, the branch branch cables, the crate controller, the dataway, the module register and back. This is done by sending test patterns and reading them back. The full test algorithm works as follows:

Generate a new test pattern (for details of look in the description of I$GTPAT).

Send the test pattern to the module, use the specified subaddress and the function code plus 16. Read the pattern back immediately with the same subaddress and the specified function code. Compare send and read pattern and check whether read and write had X and Q response. If the /FULL qualifier has been specified, a full set of test patterns is generated and processed in all test cycles.

Action rout. I$CTREGISTER

Author Walter F.J. Mueller

Remarks

File name I$CTREGISTER.PPL

Dataset -

REMARKS -

EXAMPLE TEST REGISTER C=1 N=22 /START

Description

CALLING

```
@CALL I$CTREGISTER(I_B,I_C,I_N,I_A,I_F,
        F_DELTA,C_OPTION,I_WIDTH,C_FULL);
```

ARGUMENTS

I_B	BIN FIXED(15) [INPUT] Branch number of the module to be tested.
I_C	BIN FIXED(15) [INPUT] Crate number of the module to be tested.
I_N	BIN FIXED(15) [INPUT] Station number of the module to be tested.
I_A	BIN FIXED(15) [INPUT] Subaddress of the module to be tested.
I_F	BIN FIXED(15) [INPUT] Function code in the module to be tested.
F_DELTA	BIN FLOAT(24) [INPUT] Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..
C_OPTION	CHAR(*) VAR [INPUT] Test option qualifier set, expected values are: /LIST/STOP /RUN/START/LOOP
I_WIDTH	BIN FIXED(15) [INPUT] Width of the register to be tested in bits. May be between 2 and 24.
C_FULL	CHAR(*) VAR [INPUT] /FULL qualifier. If specified, all test patters are written and read in every test interation.

FUNCTION This procedure performs dataway loopback tests with a any read/write register in a CAMAC module. For a more detailed discussion look in the description of the command TEST REGISTER.

REMARKS -

EXAMPLE

```
@CALL I$CTREGISTER(0,1,22,0,2,0.E0,
        '/RUN',16,'/FULL');
```

TEST SEN_2047

```
TEST SEN_2047
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
```

PURPOSE	Test a SEN 2047 pattern unit.
PARAMETERS	
B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

FUNCTION This test exploits the F(25) self test function of a SEN 2047 pattern to perform a simple go/nogo test. The full test algorithm works as follows:

Clear the module with a F(9) function, read the pattern and check that it contains a zero.

Issue a test with a F(25) function, read the pattern with a F(0) function and check whether it contains all one's.

Reread the pattern with a F(2) and a F(0) function and check whether is still contains an all one's pattern or is zeroed respectively.

Finally issue another F(0) on even test cycles and a F(25) on odd test cycles. This causes the pattern units LED's to blink and may be used to signal test activities for this crate.

EXAMPLE TEST SEN_2047 C=1 N=12 /START

REMARKS The LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

Action rout. I$CTSEN_2047

Author Walter F.J. Mueller

Remarks

File name I$CTSEN_2047.PPL

Dataset -

Description

CALLING @CALL I$CTSEN_2047(I_B,I_C,I_N,F_DELTA,C_OPTIONS);

ARGUMENTS

I_B BIN FIXED(15) [INPUT]
Branch number of the module to be tested.

I_C BIN FIXED(15) [INPUT]
Crate number of the module to be tested.

I_N	BIN FIXED(15) [INPUT] Station number of the module to be tested.
F_DELTA	BIN FLOAT(24) [INPUT] Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..
C_OPTION	CHAR(*) VAR [INPUT] Test option qualifier set, expected values are: /LIST/STOP /RUN/START/LOOP

FUNCTION This procedures performs a self-test for a SEN 2047 pattern unit. For a more detailed discussion look in the description of the command TEST SEN_2047.

REMARKS -

EXAMPLE @CALL I$CTSEN_2047(0,1,12,0.E0,'/RUN');

TEST SEN_2090

```
TEST SEN_2090
    B=b C=c N=n
    REPEAT=r
    /LIST /STOP /RUN /START /LOOP
```

PURPOSE Test a SEN 2090 video display driver.

PARAMETERS

B=b	Number of the branch of the module to be tested. Replaceable default = 0
C=c	Number of the crate of the module to be tested. Replaceable default = 1
N=n	Station number of the module to be tested. Replaceable default = 1
REPEAT=r	Repetition time in seconds for started tests. Must be between 0. and 3599., the default value is 1 sec.
/LIST	Will list tests of the given type. List all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/STOP	Will abort test of the given type. Abort all tests if no crate or station has been specified or just the tests in a given crate and/or station.
/RUN	Will execute the test on the specified CAMAC address only one time. This is the default if on other qualifier has been specified.
/START	Will start the repetive execution of the test on the specified CAMAC address at time intervalls specified with the REPEAT parameter.
/LOOP	Like /START, but the test is executed as often as possible, limitted only by the CPU and CAMAC speed. This LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

FUNCTION This test writes patterns to a SEN 2090 video display driver which allows a visual go/nogo test. The full test algorithm works as follows:

In the first test cycle the module is initialised by enabling the character mode and the cursor and by clearing the screen.

Than a pattern consisting of all printable characters is written to the screen. The pattern is shifted by one character in each line and in each test cycle.

EXAMPLE TEST SEN_2090 C=1 N=12 /START

REMARKS The LOOP function will only work if the GOOSY command menu is not active. Leave the command menu after submitting of the command.

Action rout. I$CTSEN_2090

Author Walter F.J. Mueller

Remarks

File name I$CTSEN_2090.PPL

Dataset -

REMARKS -

Description

CALLING @CALL I$CTSEN_2090(I_B,I_C,I_N,F_DELTA,C_OPTIONS);

ARGUMENTS

I_B BIN FIXED(15) [INPUT]
Branch number of the module to be tested.

I_C BIN FIXED(15) [INPUT]
Crate number of the module to be tested.

I_N BIN FIXED(15) [INPUT]
Station number of the module to be tested.

F_DELTA BIN FLOAT(24) [INPUT]
Repetition time in seconds for started tests. Must be between 0. and 3599., a recommended value is 1..

C_OPTION CHAR(*) VAR [INPUT]
Test option qualifier set, expected values are:
/LIST/STOP /RUN/START/LOOP

FUNCTION This procedures performs a self test for a SEN 2090 video display driver. For a more detailed discussion look in the description of the command TEST SEN_2090.

REMARKS -

EXAMPLE @CALL I$CTSEN_2090(0,1,12,0.E0,'/RUN');

TYPE BUFFER

TYPE BUFFER number /HEADER

PURPOSE	Start to type data buffers
PARAMETERS	
number	integer default=1 Number of buffers to be typed .
/HEADER	Output header only.
EXAMPLE	TYPE BUF 1
NOTE	This command works only if buffer checking is enabled.

Description

FUNCTION	Sets the number of data buffers to be dumped by I$ACQ_DUMP.
File name	I$ACQ_DMP_BUF.PPL
Action rout.	I$ACQ_DMP_BUF
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

TYPE EVENT

TYPE EVENT number id /SAMPLE /HEADER

PURPOSE	Start to type events
PARAMETERS	
number	integer default=1 Number of events to be typed.
id	integer default=0 Optional subevent id
/SAMPLE	Type one (first) event per buffer.
/HEADER	Only headers are output
EXAMPLE	TYPE EVE 10
NOTE	This command works only if buffer checking is enabled.

Description

FUNCTION	Start event typing.
File name	I$ACQ_DMP_EVT.PPL
Action rout.	I$ACQ_DMP_EVT
Dataset	-
Version	1.01
Author	Walter F.J. Mueller
Last Update	12-APR-1985

Appendix B

DCL Commands

ALIAS

ALIAS command arguments

PURPOSE	Handle GOOSY alias command names
ARGUMENTS	
command	subcommand key: CREATE DELETE SHOW A quotation mark enters menu.
arguments	argument list depending on command.

Description

FUNCTION	Alias names can be defined on two levels: Environment and Global. Alias names defined for a certain environment are activated together with the CRENV command and deactivated with the DLENV command. Global alias names are active anytime. An alias is searched first in the environment table and then in the global table. Alias names are deleted during logout.
Version	1.01
Author	H.G.Essel
Last Update	14-APR-1987

CREATE

CALLING	ALIAS CREATE name string environment /GLOBAL CRALIAS name string environment /GLOBAL
name	Name of the alias. May contain alphanumeric characters including _ $. A quotation mark enters menu.

string	Replacement string for alias. If the string contains delimiters (in terms of DCL arguments) it must be enclosed in quotes.
environment	Optional environment for which the alias will be created. If omitted a global alias is created.
/GLOBAL	Create global alias. Ignore environment parameter.

DELETE

CALLING	ALIAS DELETE name environment /GLOBAL DLALIAS name environment /GLOBAL
name	Name of the alias. May contain alphanumeric characters including _ $. A quotation mark enters menu.
environment	Optional environment for which the alias will be deleted. If omitted a global alias is deleted.
/GLOBAL	Delete global alias. Ignore environment parameter.

SHOW

CALLING	ALIAS SHOW name environment /GLOBAL/ACTIVE SHALIAS name environment /GLOBAL/ACTIVE
name	Optional name of the alias. A quotation mark enters menu.
environment	Optional environment for which the alias will be listed. If not specified, /ACTIVE is assumed.
/GLOBAL	Only global alias names are listed.
/ACTIVE	All active alias names are listed.

ATENVIR

ATENV*IR environment

PURPOSE	Attach environment.
FUNCTION	Defines logical GOOSY_PROMPT to GOO_env_MBX. Sets prompter GOOSY to MGOOTP1. The environment must be started by CRENVIR /MBX at the same node.
Version	1.01
Author	H.G.essel
Last Update	14-APR-1987

CLINK

```
CL file/switches /switches
```

PURPOSE	Link programs. Defaults are updated.
ARGUMENTS	
file	Name of object file (def.=last)
switches	Linker switches. Behind SPACE are not updated

DESCRIPTION

FUNCTION

Two problems are solved:

1. To keep last input as default
2. To inculde standard options.

There are two ways to specify switches. Directly following the file name the switches are kept as default for the next call. After a blank switches are temporary used. If one wants to use the last file, but with additional switches, the file name must be *.

REMARKS	-
File name	CLINK.COM
Dataset	-
Version	1.01
Author	H.Essel
Last Update	20-JAN-1984

COMMENT

COMMENT topic command arguments

PURPOSE Write and read comments for a topic.

ARGUMENTS

topic Name of the COMMENT topic. A logical name must be defined for the topic file:
CMT$topic = filespec

command subcommand key:
CREATE
READ [default]
WRITE
EDIT
KEYS

arguments argument list depending on command.

Description

FUNCTION COMMENT writes and reads a comment file. The file must be created once. A logical name must be assigned in the form
CMT$topic = filespec
Entries are written with date and username.
Entries are displayed in a specified date window.

Version 1.01

Author H.G.Essel

Last Update 10-MAR-1991

CREATE

CALLING	COMMENT topic CREATE
topic	Name of the COMMENT topic. A logical name must be defined for the topic file. Headlines are prompted line by line. These lines are displayed with the READ command.

READ

CALLING	COMMENT topic [READ]keylist begin-date end-date
topic	Name of the COMMENT topic. A logical name must be defined for the topic file.
READ	If no argument follows, this keyword may be omitted.
keylist	List of keys. Only entries marked with specified keys are output. * means all keys and defaults begin-date to 1-MAR-1991. Not specified defaults begin-date to YESTERDAY.
date	Date from/to wich file entries are to be displayed. Defaults depend on keylist. Begin-date default is YESTERDAY when keylist is empty, and 1-MAR-1991 when keylist is not empty (but may be *). End-date defaults to TOMORROW, if not specified.

WRITE-EDIT

CALLING	COMMENT topic WRITE-or-EDIT keylist
topic	Name of the COMMENT topic. A logical name must be defined for the topic file.
WRITE	Lines are prompted and are inserted in the file.
EDIT	The headline is written to a temporary file and the editor in invoked.
keylist	List of keys. Keys mey be specified as filter when reading entries.

KEYS

CALLING	COMMENT topic KEYS
topic	Name of the COMMENT topic. A logical name must be defined for the topic file. List all topics in a topic file.

Format

The file is a normal text file and may be edited. The syntax is:

Lines starting with ! are displayed also outside
time window. Example: header lines.
Lines starting with - are date lines.
All others are text lines.

Example

Creating a new topic file for topic MYCOM:

```
  $ DEFINE CMT$MYCOM SYS$LOGIN:CMT_MYCOM.TXT
  $ COMM MYCOM CREATE
  > ...
  $ ...
 Insert entry:
  $ COMM MYCOM W
  > ...
  $ ...
 Insert entry with key (use editor):
  $ COMM MYCOM EDIT key1,key2
  > ...
  $ ...
 Read entries with key (no date limit):
  $ COMM MYCOM READ key1
  $ ...
 Read all entries:
  $ COMM MYCOM READ *
  $ ...
 Read entries with key since yesterday:
  $ COMM MYCOM READ key1 yesterday
  $ ...
 Read all entries since yesterday:
  $ COMM MYCOM READ
  $ ...
```

CONCAT

CONCAT infile outfile /LOG/DELETE/CONFIRM/APPEND

PURPOSE	Concatenates input files to one output file.
ARGUMENTS	
infile	Input file spec. The version number must be *. All files are concatenated in the correct order (Version n+1 behind version n).
outfile	Output file name. Must be different from input.
/APPEND	If output file already exists, input files are appended.
/LOG	Log operations
/DELETE	Delete copied files
/CONFIRM	Confirm file deletion (not copy).

Description

FUNCTION	When several versions of a file are copied into one file, the highest version is on top. If this is not wanted, CONCAT copies the files in reverse order: Lowest version is on top. Lowest version is assumed to be 1. Highest version is highest on disk. Versions may be missing.
EXAMPLE	X.DAT;5 X.DAT;4 X.DAT;2 CONCAT X.DAT;* Y.DAT copies X.DAT;2 to Y.DAT and appends then X.DAT;4 and X.DAT;5
Version	1.01
Author	H.G.Essel
Last Update	21-OCT-1988

CRENVIR

```
CRENV*IR environment program component
                    /ONLINE/OFFLINE/DEFAULT
                    /$TMR/$ANL/$DSP/$DBM/J11
                    /[NO]DECWINDOW
                    /PRIORITY=p/DELETE
```

PURPOSE	Creates a GOOSY environment and optional some GOOSY standard components
ARGUMENTS	
environment	Name of the environment (max. 4 char)
program	Optional name of private analysis program. If not specified, MGOOANL is assumed. This private program is started by /$ANL or by /ONLINE or /OFFLINE, if no /DEF is specified.
component	Optional component name for analysis program. Default is $ANL.
/ONLINE	Creates TMR, DSP, DBM and analysis program specified by program (default=MGOOANL, if /DEF is not specified). If /DEF is specified, a GOOSY standard analysis program is started. If program is specified, this is used regardless of /DEFAULT.
/OFFLINE	Creates DSP, DBM and analysis program specified by program (default=MGOOANL, if /DEF is not specified). If /DEF is specified, a GOOSY standard analysis program is started. If program is specified, this is used regardless of /DEFAULT.
/DEFAULT	Creates default analysis. Otherwise use specified program, which is the name of your private analysis program (default=MGOOANL). If a program name has been specified, this switch is ignored.
/$TMR	Create Transport Manager $TMR
/$ANL	Create Analysis program $ANL

/$DSP	Create Display $DSP
/NODECWINDOW	Use old version of display (/$DSP must be given) Default is the DEC-GKS version.
/$DBM	Create Data Base Manager $DBM
/J11	Create standard analysis program GOO$EXE:MGOOANL
/PRIORITY=	Specify priority for analysis component (DEF=3). You cannot raise the priority above 4 if you have not the proper privileges.
/DELETE	Delete environment log files.

Description

FUNCTION

A GOOSY environment is created only if it does not already exist.
The components specified by qualifiers are created. You may use this
command to create components in an existing environment. Specifiy
the current name or * for the environment parameter in this case.
A user specific analysis program linked by
command LANL is started from current directory by /$ANL.
GOOSY provides a standard analysis program.
This analysis is started by /J11 or by /DEFAULT.
By default the analysis programs are started
with priority 3 unless /PRIO=p is specified.

NOTE An environment is deleted by command DLENV.

Version 1.01

Author H.G.Essel

Created 14-JAN-1987

Last Update 19-OCT-1987
File creation error handled /HE
18-APR-1990
create enviroment table in this module /RF
23-Jul-1993
DEC-GKS V5.0 version is default. /HE

Examples

```
$ CRENV SUSI /$DBM ! create environment and $DBM
$ CRENV SUSI /$TMR/$ANL ! add $ANL component
$ DLENV ! delete environment
$ CRENV SUSI /ONLINE/DEF ! Create $TMR, $DBM, $DSP, $ANL
                         ! Use GOO$EXE:MGOOANL
$ DLENV ! delete environment
$ CRENV SUSI /OFFLINE ! Create $DBM, $DSP, $ANL
                         ! Use private MGOOANL
$ DLENV ! delete environment
$ CRENV SUSI may5 /OFFL ! Create $DBM, $DSP, $ANL
                         ! Use MAY5.EXE for analysis
$ DLENV ! delete environment
$ CRENV SUSI may5 /DELE ! Create $ANL
                         ! Use MAY5.EXE for analysis
                         ! delete old log files
```

CTRL_T

CTRL_T [process][output]

PURPOSE	Similar to an interactive CTRL_T, but works in DCL procedures.
ARGUMENTS	
process	optional process name
output	optional output (def=SYS$OUTPUT)

Description

FUNCTION	Process information of specified or current process is written to output (SYS$OUTPUT).
Version	1.01
Author	H.G.Essel
Last Update	28-JUL-1986

DEBUG_WINDOW

```
DEBUG_WINDOW window
DEBWIN window
```

PURPOSE Setup windows and inits for debug.

ARGUMENTS

window Desired debug window style:

DECW or MOTIF	A setup is generated to use standard DECwindow/Motif debugger windows. If you are not on a workstation screen, the standard screen initialisation is enabled.
VWS or none	On VWS or DECwindow workstations a separate debug window will be opened for the conventional debug in/output. The standard screen initialisation is enabled.
terminal	When a terminal is specified, the debug input and output is directed to this terminal. The standard screen initialisation is enabled.

Description

FUNCTION -

REMARKS -

EXAMPLE

```
$ DEBWIN MOTIF
```

DLENVIR

DLENV*IR

PURPOSE	Delete current environment and all subprocesses
FUNCTION	1.Defines logical LNM$GOOSY_TABLES to LNM$GOOSY. 2.Deletes all GOOSY components. 3.Deletes the environment. 4.Sets process name to Y_env_#.
Version	1.01
Author	H.G.essel
Last Update	14-APR-1987

DTENVIR

DTENV*IR

PURPOSE	Detach environment.
FUNCTION	Deassigne logical GOOSY_PROMPT. Resets prompter GOOSY to MGOOTP0
Version	1.01
Author	H.G.essel
Last Update	6-Feb-1990

ETHDEF

ETHDEF destination ethernet protocol interface

PURPOSE	Define logicals for ethernet connection to VME.
ARGUMENTS	
destination	Destination node (E5ELXA). Valid values: E5ELXA, E5ELXB
ethernet	Interface type. Valid values: Microvaxes:QB V8600A,V8600B:UB V6000a,V780a:BI 4000-90:WZ other workstations: WS
protocol	Protocol type. Valid values: TMR
interface	Parallel interface. Valid values: UUA0:

Description

FUNCTION	Defines logical names for ethernet connection to NET processor in VME.
Version	1.01
Author	H.G.Essel
Last Update	25-APR-1990

GOOCONTROL

GOOC*ONTROL [CREATE]or [DISMOUNT]

PURPOSE	Defines logical name GOO$CONTROL for control data base. The data base is created and mounted optionally.
ARGUMENTS	
CREATE	A new data base is created. The name is composed of GOOCTRL_group_node.SEC
DISMOUNT	The data base is dismounted. This should be done during LOGOUT.

Description

CALLING	GOOC*ONTROL [CREATE]
ARGUMENTS	
CREATE	A new data base is created
FUNCTION	The programs MGOOTMR and MGOOANL keep their control information in a database GOO$CONTROL, if it exists. The command SHOW GOOSY STATUS may be used to display this information on screen. The command GOOCONTROL creates such a data base and defines the logical name GOO$CONTROL. The "physical" data base name is GOOCTRL_<group>_<node>, where node is the one letter node name, group the hex group number. The command should be included in the LOGIN procedure.
REMARKS	On each node there must be a control data base. If the section file does not exist, it is created. If the data base is not mounted, this is done.
EXAMPLE	GOOCON CRE creates a data base GOOCTRL_131_E, if called on EMMA from GOOFY (Group 203).

GUIDE

GUIDE facility level /INIT=string/BRIEF/LIST/LASER

PURPOSE Menu driven guide to use facilities.

ARGUMENTS

facility The name of the facility to be used, e.g. GOOSY If ommitted or asterisk, all available facilities are listed. Then a facility is prompted.

level An optional level specification to enter a specific menu level. The specification is in the form:
n1.n2.n3...
If an asterisk is given, all available levels are listed. Then a level is prompted.

/INIT=string This string will be passed to the initialization procedure as one argument.

/BRIEF Some command procedures display information about the actions to be done. Parts of this output can be suppressed by /BRIEF.

/LIST All menu levels are displayed together with their file names. Note that the 'real' filenames are prefixed by GU_facility_.

/LASER For file output no highlight mode is written. The LN03 is capable to print highlight. To include highlightening in the output file, use /LASER All list output may be directed to a file by
GUIDE/OUTPUT=file ...

Function

FUNCTION A menu driven guide is entered. The menus show topics marked by numbers. If a number is followed by a hyphen(-) the topic points to another menu. If not, it executes a command procedure. The top line shows with "path =" the topic numbers to reach the current menu. It shows with "last topic =" the previous topic after a return.

Next menu Entering a number the menu of the selected topc is displayed or the command procedure is executed, respectively.
  Enter number to select topic : 2 (select topic 2)

Exit a menu The previous menu is entered by typing "B" or "b" or <CTRL>Z or just <RETURN>. The guide is left by "E" or "e" or <CTRL>Y.
  Enter number to select topic : e (exit GUIDE)
  Enter number to select topic : B (prev. menu)

Select menu To enter another menu directly, one can specify a level expression by "=n1.n2....". Then GUIDE enters the menu n1.n2.n3 (counting from the beginning). This is the same as leaving GUIDE and calling again with the level specification.
  Enter number to select topic : =1.2.1 (select menu 1.2.1)
  Enter number to select topic : = (select top menu)
One can jump over menus by input of a level expression "n1.n2.n3". Then these topics are selected beginning from the current one.
  Enter number to select topic : 2.4 (select topic 2.4)

DCL proc. A DCL command line can be entered behind an at (@). The command will be executed in a spawned subprocess. Note that the @ means NOT to execute a DCL procedure! To do that, two @'s are necessary:
  Enter number to select topic : @DIR *.PPL
  Enter number to select topic : @@dclproc

HELP DCL help can be invoced directly by:
  Enter number to select topic : H keywords

Examples

EXAMPLE

```
GUIDE ! display facilities
  GUIDE GOOSY ! enter first level menu
  GUIDE GOOSY 1.2 ! enter menu 1.2
  GUIDE/OUTPUT=GOOSY_GUIDE.LIS GOOSY /LIST
                        ! write all levels into file
  GUIDE GOOSY * ! display all levels and
                        ! prompt for level
Examples for answering the prompt:
  Enter number to select topic : 2 (select topic 2)
  Enter number to select topic : 2.4 (select topic 2.4)
                             starting at current level.
  Enter number to select topic : =1.2.1 (select menu 1.2.1)
                             starting from level 0.
```

```
Enter number to select topic : = (select top menu)
Enter number to select topic : e (exit GUIDE)
Enter number to select topic : @DIR *.PPL
                               execute command DIR *.PPL in spawn.
Enter number to select topic : @@XYZ
                               execute DCL procedure XYZ
Enter number to select topic : ? (HELP GUIDE)
Enter number to select topic : H string (HELP)
```

Guide-Programming

Function

GUIDE processes two kinds of files:

1. Text files with type .TXT
2. DCL procedures with type .COM

The file names are:
GU_facility_menu.TXT or GU_facility_menu.COM
<facility> is specified by calling GUIDE,
<menu> is specifed in the text files for the
menu levels.
The first level file must be GU_facility.TXT

The text files contain the menu information. All text and DCL files must reside on the same directory. This directory is translated from logical name GUIDE$facility. If this logical name is not defined, the files are looked up from the directory of GUIDE. The format is described in the following:

Menu-Design

The menus ore defined in text files
GU_facility_menu.TXT

An exclamation mark (!) at the beginning of a line marks comments. These lines are ignored as well as empty lines.

The first line must be the menu headline preceded
by an !. The next line must be empty.

At the beginning of the line there must be the
menu name for the menu to be called by that line. If the next level is no text, but rather a DCL procedure to be executed, the menu name must be preceded by an "@".

Behind two bars (——) the text to be displayed
follows. This text is used as headline for that topic menu. If no bars are found, the line is displayed as continuation line (double bars are not allowed). Example:

! comment line
XYZ—— menu line (enters next menu)
@ABCDE—— menu line (executes DCL procedure)
continuation without double bars
Here, topic 1 enters the next menu level reading
text from file GU_facility_XYZ.TXT
Topic 2 executes a DCL procedure named
GU_facility_ABCDE.COM The text files should not contain more than 19 true text lines (without comments).

Command-procedures

The command procedures executed by GUIDE should handle CONTROL_Y and ERRORs in a proper way. They should report at the end a success or error. When they display information on the screen they should prompt for a <RET> to continue to give the user the chance to read the output. Prompts from the terminal should be done by

```
$ READ/END=G_cont/PROMPT="string" SYS$COMMAND line
$ G_cont:
```

This avoids the answers to be written in the terminal
recall buffer. The END label is reached by CONTROL_Z. Note that the symbol 'line' is NOT changed in that case. Another way is to use the GUIDE_PROMPT procedure. The symbol PROMPT is defined in GUIDE.COM:

```
$ PROMPT "string" "default"
```

The answer is in global symbol PROMPT_ANSWER.
If the prompt was broken by <CTRL>Z , a 3 is returned in $STATUS and PROMPT_ANSWER is "". The default specification is optional. There can be optionally specified a HELP keyword as P4. Then a ? as input enters HELP with that keyword. The guide procedures may also be call MDCLLIST to get parameters. The procedures are always called with a ? as P1. Then MDCLLIST enters a parameter menu.
GUIDE sets a global symbol GUIDE_VERB (verbosity).
GUIDE_VERB is TRUE by default.
GUIDE_VERB is FALSE if GUIDE is called with /BRIEF.
Using that symbol one can control the verbosity of
output.

GUIDE-initialization

Specific initializations for a guide can be done in a command procedure named
GUIDE$facil:GU_facil_INITIALIZATION
This procedure is called before any menu is entered.
It is not called for listings (/LIST/FILE/MENU).

GUIDE-finish

Specific finish actions for a guide can be done in a command procedure named
GUIDE$facil:GU_facil_FINISH
This procedure is called before leaving guide.
It is not called for listings (/LIST/FILE/MENU).

Guide-guide

There is a guide to write guides. This guide is invoked by
GUIDE GUIDE
The files GU_GUIDE* can be used as example how to
write guide files.

Qualifiers

CALLING	GUIDE facility level /BRIEF/LIST/LASER /FULL/FILES/MENU
/FULL	All sources are included in the output.
/FILES	Outputs a list of all used files.
/MENU	Outputs all menus.

LINKJ11

LINKJ11 objfile /COMPLE

PURPOSE	Link a J11 stand alone task
ARGUMENTS	
objfile	OBJ filename, created by COMPILE file.MAC. Must be on current directory.
/COMPILE	Optional compile file first.

Description

FUNCTION	-
Version	1.01
Author	H. Grein
Last Update	30-APR-1987

LSHARIM

```
LSHARIM module image
          /GLOBAL=list
          /SHARE*LOG=name
          /MAP=mapfile
          /KEEP
          /GROUP
          /DEBUG
```

PURPOSE	Link modules into a sharable image.
ARGUMENTS	
module	Modules to be linked into sharable image: 1. List of file names (wildcards) 2. @file contains file names 3. @library(module name list)
image	Name of generated executable sharable image
/GLOBAL=list	Globals which occur in the modules
/SHARE*LOG=n	Logical name of the sharable image
/MAP=mapfile	Optional map file.
/KEEP	Keep all temporary files
/GROUP	The logical name for the sharable image is entered in the GROUP table instead of the JOB table.
/DEBUG	Link with debugger.

DESCRIPTION

CALLING

```
LSHARIM module image
        /GLOBAL=list
        /SHARE*LOG=name
```

```
/MAP=mapfile
/KEEP
/DEBUG
```

ARGUMENTS

module	Modules to be linked into the sharable image. The following inputs are possible: 1.) A VAX/VMS file specification list with any wildcards; e.g. X$*USER*. The default file type is .OBJ. The specified modules have to be compiled. 2.) A file, which contains the list of modules to be linked into the sharable image. The file has to be specified with a leading "@"; e.g. @list.dat Per line one module name is expected. The modules have to be compiled. 3.) A library specification; e.g @opriv.olb(X$*) @opriv.olb(X$USER_ANAL)
image	Name of generated executable sharable image file. Default type is .EXE.
/GLOBAL=list	All FORTRAN COMMON blocks or PL/I EXTERNAL variables have to be placed in unshared sections. Therefore all globals occuring in your program have to be known. If only one global global parameter occurs define it directly: /GLOBAL=name Futhermore a file containing a list of all used global paramters can be specified,e.g.: /GLOBAL=@list.dat In each line one global parameter name is assumed.
/SHARE*LOG=n	Logical name assigned to the the sharable image in JOB table. If not specified, the sharable image name is used.
/MAP=mapfile	Optional map file.
/GROUP	The logical name for the sharable image is entered in the GROUP table instead of the JOB table. You need GRPNAM priviledge for that. Note that the the logical name is valid for all sessions in the group.
/KEEP	Keep all temporary files
/DEBUG	Link with debugger.

FUNCTION

One or several modules can be linked together into a sharable image. The global parameters refered in the modules can be specified and are placed in unshared sections of the sharable image. For control a linker map file is genereated.

The sharable image can be referenced only by the logical name. This name is defined only during the session or untill next system startup (/GROUP). Therefore one should add the definition in the LOGIN.COM with /JOB qualifier.

Example

LSHARIM x$*.obJ anal.exe /MAP=anal.map/share=usershr
All modules starting with X$ are linked into the generated sharable image "ANAL.EXE". The map file "ANAL.MAP" will be produced and the logical name "USERSHR" is assigned to the sharable image.

LSHARIM x$start,x$stop anal.exe
Modules x$start.obj and x$stop.obj are used building the sharable image "ANAL.EXE".

LSHARIM @file.dat anal.exe
All modules listed in "FILE.DAT" are used building the sharable image "ANAL.EXE".

LSHARIM @opriv(X$*) anal.exe
All modules X$* foung in the obejct library opriv are linked to a sharable image.

MANUAL

MANUAL PRINT
/INTRO/DISPLAY/ANALYSIS/DATABASE/VME/HARDWARE
/BUFFER/VMS/ACQCOM/ANACOM/ALL/ACQUISITION

PURPOSE	Print GOOSY manuals.
ARGUMENTS	
/INTRO	GOOSY introduction
/DISPLAY	GOOSY display including commands
/ANALYSIS	GOOSY data acquisition and analysis including commands and macros
/ACQUISITION	GOOSY data acquisition.
/DATABASE	GOOSY data base manager including commands
/VME	VME frontend system
/HARDWARE	Hardware description (J11 and MBD) and buffer structures
/BUFFER	GOOSY buffer and event structures
/VMS	GSI introduction to VMS
/ACQCOM	GOOSY transport manager commands
/ANACOM	GOOSY analysis manager commands and macros
/ALL	All manuals (Caution: A lot of paper!)

DESCRIPTION

CALLING MANUAL PRINT
/INTRO/DISPLAY/ANALYSIS/DATABASE/VME/HARDWARE
/BUFFER/VMS/ACQCOM/ANACOM/ALL/ACQUISITION

ARGUMENTS

FUNCTION Print specified manuals doublesided on postscript printer in computer center printer room.

MTAPE

```
MTAPE device name
        /INI*TIALIZE/DENS*ITY=d/BLOCK*SIZE=b/DIS*MOUNT
```

PURPOSE	Initialize and mount a GOOSY tape
ARGUMENTS	
device	Logical name of tape unit.
name	Label name of the tape
/INITIALIZE	Initialize tape
/DENSITY=d	Tape density, default=6250
/BLOCKSIZE=b	Blocksize in Kbyte, default=24. Should be multiple of GOOSY blocksize. Default is normally adequate.
/DISMOUNT	A tape already mounted is dismounted first. You must mount the new tape on the device and hit <RETURN> to continue.

Description

FUNCTION	The tape is optionally initialized and then mounted. The density specification is used for initialization, the blocksize for mounting. If /DISMOUNT is specified, the tape presently mounted (if any) is dismounted. Then You must mount the new tape on the device, and enter <RETURN> to continue. Without the /DISMOUNT qualifier it is assumed that the desired tape volume is already mounted on the device.
Version	1.01
Author	H.G.Essel
Last Update	8-OCT-1987

OPSER

OPSER command

PURPOSE Execute priviledged operator commands

ARGUMENTS

command ? -> enter main menu
SEARCH (GOOSY)
DIFFER (GOOSY)
COMPILE (GOOSY)
REPLY
TAPE
SET

Description

FUNCTION The commands supported by OPSER are executed by an operator server.

EXAMPLE To enter main menu:

```
$ OPSER
```

To enter sub menu for REPLY:

```
$ OPSER REPLY
```

PLOTMET

```
PLOTMET metafile type command plotter
          /COPIES=c /FONT=f
```

PURPOSE	Plot a metafile on specified plotter
ARGUMENTS	
metafile	Name of the metafile which should be plotted on the specified queue or physical device.
type	Device type for format. The following types are supported by GOOSY: 1.) LN03 Laser printer (=default) 2.) HP7550A3,HP7550A4 pen plotter 3.) POST postscript
command	Optional print command (enclose in ""). If specified, plotter is ignored.
plotter	Queue name or physical address of the plotter. If a colon (":") is specified at this position it is assumed that a physical adress has beeen specified. Is ignored when a print command is given.
copies	Number of copies which should be printed. If no Printer queue is specified "copies" is ignored. (default=1)
font	Font to be used to modify the default text bundle table. This argument could be used to produce pictures with nicer lettering. (default=0)

Description

FUNCTION This procedure plots the specified metafile on a plotter.

NOTE One may format to POSTscript format, but print on LN03 printers. In this case use the command "P x POST" where x is A,B,C...

Example

```
PLOTMET x.meta POST "PS A POST"
  PLOTMET x.meta POST "P A POST"
  PLOTMET x.meta LN3 "" SYS$LN03_A
```

File name	PLOTMET.COM
Dataset	-
Version	1.01
Author	W. Spreng
Last Update	19-NOV-1986

SELECT_MBD

SELECT_MBD mbd

PURPOSE Select a valid MBD controller on a VAX

ARGUMENTS

mbd I The device code of a MBD (A or B)
A : Normally the only or the first MBD on a VAX.
This is the only one for micro-VAXes.
B : The second MBD on VAX-8600 (DONALD or EMMA).

Description

CALLING SELECT_MBD mbd

ARGUMENTS

mbd I The device code of a MBD (A or B)
A : Normally the only or the first MBD on a VAX.
This is the only one for micro-VAXes.
B : The second MBD on VAX-8600 (DONALD or EMMA).

FUNCTION The command procedure will check any existing MBD, the VAX where the selection is done, and it will define the logical names:
MBDA, MBDB, and MBDC
in the job logical name table of the caller.
The GOOLOG command procedure will define the logical names to be invalid to make shure that the user will call this procedure before he can access any CAMAC function.

REMARKS Must be called once before any CAMAC function can be performed.

EXAMPLE SELECT_MBD B ! Selects the 2nd MBD

Utility UTIL

Home direct. GOO$EXE

File name	SELECT_MBD.COM
Author	M. Richter
Created	15-FEB-1988
Last Update	19-FEB-1988

SETMESSAGE

SETMESSAGE facility qualifier

PURPOSE	Control Message output of GOOSY and VMS
ARGUMENTS	
facility	GOOSY: GOOSY Messages VMS : VMS Messages
qualifier	see below

Description

FUNCTION	Enables or Dissables different levels of messages
Version	1.01
Author	R.Thomitzek
Last Update	20-OCT-1988

GOOSY

CALLING	SETMESSAGE GOOSY /NOHEADER/NOPREFIX/LAST/ON/OFF/SHOW
/NOHEADER	Message Headerline of GOOSY-Messages will be suppressed
/NOPREFIX	Message Prefix of GOOSY-Messages will be suppressed
/LAST	Only last Message will be output
/SHOW	Show message setting
/ON	Full message
/OFF	Same as /NOHEADER/NOPREFIX

Example

```
SETM GOOSY /NOH ! no header
  SETM GOOSY /ON ! full message output
  SETM GOOSY ! full message output
  SETM GOOSY /SHO ! Show message output setting
  SETM GOOSY /OFF ! no header, no prefix
```

VMS

CALLING SETMESSAGE VMS /ON/NOPREFIX

/ON Switch ON all output of VMS Messages

/NOPREFIX Switch OFF all parts of VMS Messages except text.

Example

```
SETM VMS /NOP ! no facility, severity and id
  SETM VMS /ON ! full message output
  SETM VMS ! full message output
```

TLOCK

TLOCK

PURPOSE	Lock terminal by password

Description

FUNCTION	Locks terminal by password. The password is prompted after call. Then it must be reentered to leave the procedure. If you forget the password, the job must be canceled!
Version	1.01
Author	H.G.Essel
Last Update	1-DEC-1988

VMESTRUC

VMESTRUC inputfile /PLI/FOR/C/PLIB=/CLIB=/FLIB=
/GLPUT/DELETE

PURPOSE	Generate declarations from language independent source.
ARGUMENTS	
inputfile	File containing language independent decla- rations. Specify library module as library(module).
/PLI/FOR/C	Controls, which output is generated.
/PLIB/FLIB=	Optional VMS text libraries to store the generated modules.
/CLIB=	Optional directory to store generated modules. I.e. VME$INC:
/GLPUT	Use GLPUT command instead of LIB/REP or COPY
/DELETE	Delete generated files.

Description

CALLING	VMESTRUC inputfile /PLI/FOR/C/PLIB=/CLIB=/FLIB= /GLPUT/DELETE
ARGUMENTS	
inputfile	File containing language independent declarations. Default file type is .VMES. Specify library module as library(module).
/PLI	Output PL/1 include file. Filename is inputfile.PINC.
/FOR	Output FORTARN include file. Filename is inputfile.FINC.
/C	Output C include file. Filename is inputfile..
/PLIB/FLIB=	Optional VMS text libraries to store the generated modules witrh PL/1 or FORTRAN syntax.

/CLIB=	Optional directory to store generated modules with C syntax, i.e. VME$INC:
/GLPUT	Use GLPUT command instead of LIB/REP or COPY
/DELETE	Delete generated files.
FUNCTION	Declarations of constants, variables and structures are translated into the proper PL/1, FORTRAN or C statements. The syntax of the source file is described below. If none of the qualifier is selected, all three output files are generated. The syntax of each include file is checked by a test compilation.
EXAMPLE	VMESTR sysctrl generates sysctr.pinc, sysctrl.finc and sysctrl..

Syntax

The syntax of the source file is:
! comment at any place
DEFINE constant value
PREFIX letter
[EXTERNAL]LONG [POINTER]name[(i1,i2)]
[EXTERNAL]WORD [POINTER]name[(i1,i2)]
[EXTERNAL]BYTE [POINTER]name[(i1,i2)]
[EXTERNAL]BIT [POINTER]name[(i1,i2)]
[EXTERNAL]FLOAT [POINTER]name[(i1,i2)]
[EXTERNAL]STRUCTURE [POINTER]structure name
  the structure must be known.
STRUCTURE [POINTER]structure
structure declarations
ENDSTRUCTURE
SWAP
  lines to be swapped in order in C output
ENDSWAP
%%P this line for PL/1 only
%%C this line for C only
%%F this line for FORTRAN only

Definition values can be specified in decimal, hex (%X...), octal (%O...) string. Character striong must be enclosed in "" All keywords except POINTER may be abbreviated. The array dimensions may be constants previously defined. The variable names for PL/1 and C are prefixed by type letters, i.e. L_ for longword. Structure members are prefixed by type letter, prefix letter and dollar sign. Structures may be nested up to 2 levels:

```
STRUCTURE POINTER X
  STRUCTURE Y
    LONG Y1
  ENDSTR
ENDSTR
```

EXAMPLE

S1

```
Source:
  prefix A
  str pointer x
      long x_1
  swap
      word x_2
      word x_3
  endswap
  %%P word x(LA$x_1-2)
  %%C word x(1)
  endstr
  %%C extern struct x x1(10)
  %%P extern struct x x1(10)
generates PL/1:
  DCL P_SA$x POINTER ;
  DCL 1 SA$x BASED(P_SA$x),
      2 LA$x_1 BIN FIXED(31),
      2 IA$x_2 BIN FIXED(15),
      2 IA$x_3 BIN FIXED(15),
      2 IA$x(LA$x_1-2) BIN FIXED(15);
  DCL 1 SA$x1(10) LIKE(SA$x) EXTERNAL;
and C:
  struct s_x
  {
  long l_x_1;
  short i_x_3;
  short i_x_2;
  short i_x[1];
  } *p_x;
  extern struct s_x s_x1[10];
```

S2

Source:
```
prefix N
%%P long SN$y_1
str pointer y
     long y_1
%%P str y_2(L_SN$y_1 REFER(LN$y_1))
%%P byte y_3
%%P byte y_4
%%P byte y_5
%%P byte y_6
%%P endstr
%%C long y_7(1)
endstr
```
generates PL/1:
```
DCL L_SN$y_1 BIN FIXED(31);
DCL P_SN$y POINTER ;
DCL 1 SN$y BASED(P_SN$y),
     2 LN$y_1 BIN FIXED(31),
     2 SN$y_2(L_SN$y_1 REFER(LN$y_1)) ,
     3 HN$y_3 BIN FIXED(7),
     3 HN$y_4 BIN FIXED(7),
     3 HN$y_5 BIN FIXED(7),
     3 HN$y_6 BIN FIXED(7);
```
and C:
```
struct s_y
{
long l_y_1;
long l_y_7[1];
} *p_y;
```

S3

Source:
```
prefix N
str z
     long z_1
     str z_2(10)
          long z_3
```

```
            long z_4
        endstr
    endstr
generates PL/1:
    DCL 1 SN$z ,
        2 LA$z_1 BIN FIXED(31),
        2 SN$z_2(10) ,
        3 LN$z_3 BIN FIXED(31),
        3 LN$z_4 BIN FIXED(31);
and C:
    struct s_z
    {
    long l_z_1;
    struct s_z_2
    {
    long l_z_3;
    long l_z_4;
    } z_2;
    } z;
```

WCLOSE

WCLOSE file

PURPOSE Wait for file to be closed.

ARGUMENTS

file File to be checked.

Description

FUNCTION Trys to open specified file. If the file is locked, it waits and retries, if not the file is closed. This command should be used to wait for an analysis writing an output file after closing the output file by STOP ANAL OUT /CLOSE because pending buffers are written before the file is closed. If in a DCL procedure the analysis would be deleted after the STOP command, the last buffer cannot be written into the file.

EXAMPLE

```
$ GOOSY START ANAL OUT X.LMD
  $ GOOSY START INPUT FILE Y.LMD
  $ MGOOWAIT ...
  $ GOOSY STOP ANAL OUT /CLOSE
  $ WCLOSE X.LMD
  $ GOOSY DELETE PROCESS $ANL
  $ ...
```

Version 1.01

Author H.G.Essel

Last Update 15-FEB-1989

Appendix C

GOOSY Data Formats

C.1 Introduction

C.1.1 Buffers

The GOOSY dump file format defines the structure of

1. data streams between the processors and processes controlled by GOOSY, e.g. the frontend equipment and the GOOSY processes,
2. dumps of data produced by GOOSY for later analysis or exchange of data between GOOSY and other systems.

The smallest entities of data, which are transported by GOOSY in the sense mentioned above, are called *buffers*. Presently these buffers have a fix length of either 16, 8 or 4 KByte. On disk, the buffers are stored in one RMS record, on tape several buffers can be stored into one tape record.

C.1.2 Buffer Files

If GOOSY buffers are dumped to files, the first buffer may be a file header buffer (see section C.4.2)! If the file is written to a tape, the tape is labled by **ANSI tape labels** as described in the ANSI standard (American National Standard X3.27-1978). In the appendix there is an overview of the implementations of this standard on DEC VAX/VMS and IBM MVS/XA. In general, GOOSY uses DEC's standard RMS file formats. The GOOSY files contain fixed length records.

C.1.3 Message Control Blocks

The GOOSY MCB format defines the structure of

1. control data streams between the processors and processes controlled by GOOSY, e.g. the frontend equipment and the GOOSY processes.

C.1.4 Glossary

byte means: 8–bit–sequence

word means: 2 bytes

longword means: 4 bytes.

buffer element Whole buffer or part of a buffer.

buffer element header unified structure keeping information about the trailing buffer element data.

buffer element data Data of any structure including other buffer elements. Always preceded by a buffer element header.

event Data describing one physical event. Events are buffer elements in standard buffers. There are, however, buffers containig events without headers (nonstandard buffers). Events may be composed of subevents.

If not otherwise stated:

All length fields are given in 16–bit word units. One box line in the structure figures represents a 32 bit word. Offsets are given in bytes. The order of bits, bytes, and words is always from the right to the left, i.e. from the least to the most significant bit or byte, as the VAX processes them. All character string fields are written with 7–bit ASCII coding.

Byte Order: Between machines with different byte ordering a longword swap must be performed. All Structures in this manual refer to the VAX byte ordering (little endian: least significant bit is in byte with lowest address). Big endian machines must use structure declarations with swapped words and bytes.

C.2 Message Control Block Structure

Control information between the VAX computers and the VME processors is packed in message control blocks. These are composed of a header and a message field. The message field contains a message header and a GOOSY buffer. The fields in the header are used on the local modules. No

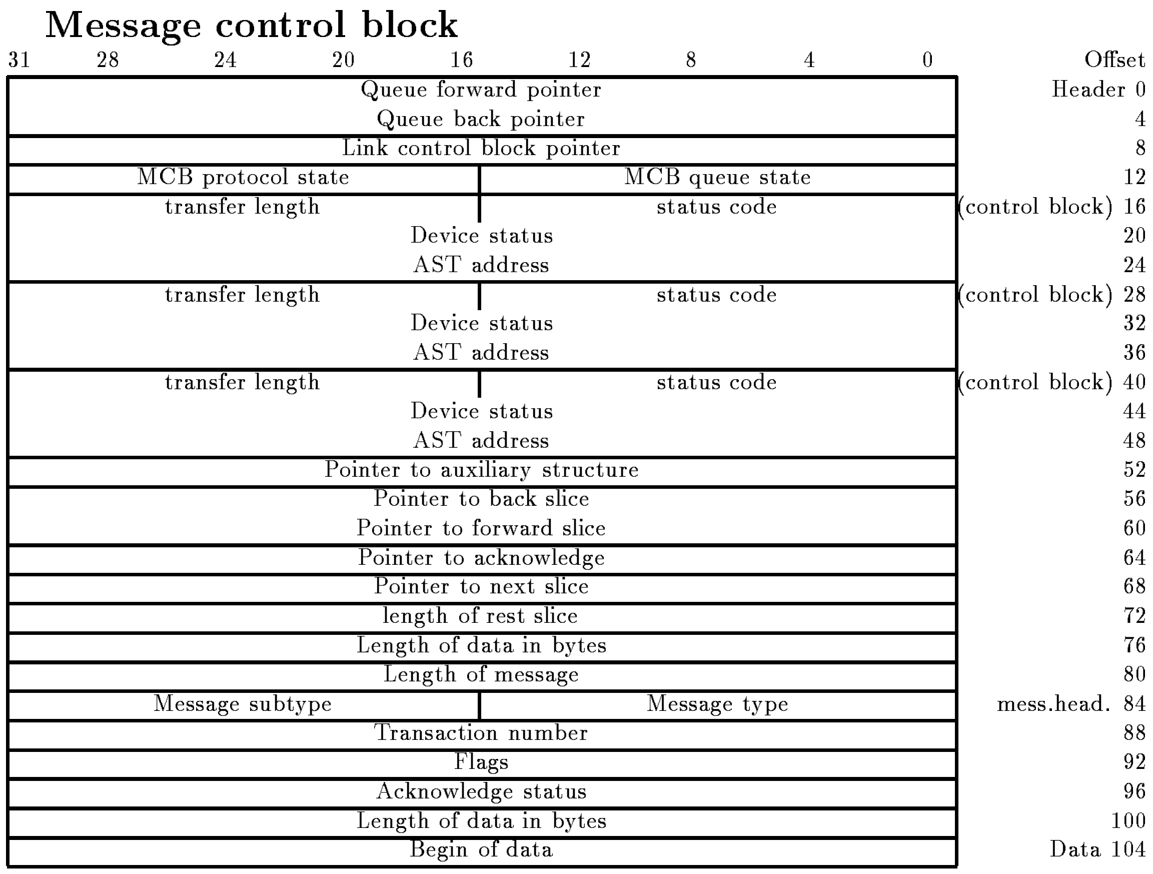

Figure C.1: Message Control Block Structure

information is transferred. The message header contains information which is transferred. The structure is found in GOOINC(SN$MCB):

```
DCL 1 SN$MCB           BASED(P_SN$MCB),
2 SN$MCB_CTL,                           /* Control part */
  3 PN$MCB_NMCB(2)     POINTER,         /* Queue link */
  3 PN$MCB_LCB         POINTER,         /* LCB backpointer */
  3 IN$MCB_QSTATE      BIN FIXED(15),   /* MCB queue state */
  3 IN$MCB_PSTATE      BIN FIXED(15),   /* MCB protocol state */
  3 SN$MCB_PIOSB,                       /* IOSB used for NET-QIO's */
    4 IN$MCB_PIOSB_STAT BIN FIXED(15),  /* Operation status */
    4 IN$MCB_PIOSB_LGT  BIN FIXED(15),  /* Transfer length */
    4 LN$MCB_PIOSB_AUX  BIN FIXED(31),  /* Device specific information */
  3 EN$MCB_PAST         ENTRY(POINTER)  /* Completion AST */
                        RETURNS(BIN FIXED(31))
                        VARIABLE,
  3 SN$MCB_LIOSB,                       /* IOSB used for NET-QIO's */
    4 IN$MCB_LIOSB_STAT BIN FIXED(15),  /* Operation status */
    4 IN$MCB_LIOSB_LGT  BIN FIXED(15),  /* Transfer length */
    4 LN$MCB_LIOSB_AUX  BIN FIXED(31),  /* Device specific information */
  3 EN$MCB_LAST         ENTRY(POINTER)  /* Completion AST */
                        RETURNS(BIN FIXED(31))
                        VARIABLE,
  3 SN$MCB_TIOSB,                       /* IOSB used for NET-QIO's */
    4 IN$MCB_TIOSB_STAT BIN FIXED(15),  /* Operation status */
    4 IN$MCB_TIOSB_LGT  BIN FIXED(15),  /* transfer length */
    4 LN$MCB_TIOSB_AUX  BIN FIXED(31),  /* Device specific information */
  3 EN$MCB_TAST         ENTRY(POINTER)  /* Completion AST */
                        RETURNS(BIN FIXED(31))
                        VARIABLE,
  3 PN$MCB_APPL         POINTER,        /* Pointer to application DSC */
  3 PN$MCB_MCB_BACK     POINTER,        /* MCB backpointer for slicing */
  3 PN$MCB_MCB_FORW     POINTER,        /* MCB forward pointer for slicing */
  3 PN$MCB_MCB_ACKN     POINTER,        /* MCB pointer to acknowledge */
  3 PN$MCB_BUF_PTR      POINTER,        /* Point to next slice */
  3 LN$MCB_BUF_LGT      BIN FIXED(31),  /* Length of rest slice */
  3 LN$MCB_ALLOC_SIZE   BIN FIXED(31),  /* Allocation size */
  3 LN$MCB_MSG_SIZE     BIN FIXED(31),  /* Total message size */
                                        /* Header plus data part send */
2 SN$MCB_MSG,                           /* Total message */
  3 SN$MCB_HDR,                         /* Message header */
    4 IN$MCB_MSG_TYPE   BIN FIXED(15),  /* Message type */
    4 IN$MCB_MSG_SUBTYPE BIN FIXED(15), /* Message sub-type */
    4 LN$MCB_TSN        BIN FIXED(31),  /* Transaction number */
    4 BN$MCB_MODE       BIT(32) ALIGNED,/* Flags */
    4 LN$MCB_STAT_ACKN  BIN FIXED(31),  /* Acknowledge status */
    4 LN$MCB_DATA_SIZE  BIN FIXED(31),  /* Data size */
  3 SN$MCB_DATA,                        /* Message data */
    4 IN$MCB_DATA(1 $MCB_DATA REFER(LN$MCB_ALLOC_SIZE))
                        BIN FIXED(7);   /* Message data array */
```

C.3 Buffer Structure

C.3.1 Standard Buffers

- **Buffer Element**
  A GOOSY buffer contains an arbitrary number of buffer elements. Buffer elements, which are *not* known to GOOSY are invalid and rejected. Any buffer element is composed of two parts:
- **Buffer Element Header**
  Headers work like envelopes for data. Examples for headers are the buffer header (see section C.3.1) and the event header (section C.3.4). The header specifies the type and size of the following data.
- **Buffer Element Data**
  Arbitrary structured data. The structure may contain other buffer elements. The type specified in the buffer element header must always uniquely define the kind of data following.

Examples of buffer elements are the buffer itself, GOOSY events and GOOSY subevents. Others are time stamps, spectra etc.. Figure C.2 shows the buffer structure. One can see the nested structures. The headers always contain a type/subtype number combination and the word length of the following data. The type/subtype numbers are unique for a certain data structure. All modules processing buffers can check if a buffer element has the correct type. If not, it may just skip the element, output messages or skip the buffer.

C.3.2 Nonstandard Buffers

Structures, which are defined by external processors or by the hardware of a frontend system are called *external structures*. In standard buffers external structures are always enveloped by headers. These headers must be added by the frontend processors. An example is the event type 1 as described in section C.5.3, a structure, which is created by the SILENA 4418x ADC–System. If external structures without header are copied directly into a buffer, this buffer has no standard format. Examples of such external structures are the SILENA (section C.7.1) and FERA (section C.7.2) event structures, if they are not preprocessed by a frontend processor adding a header.

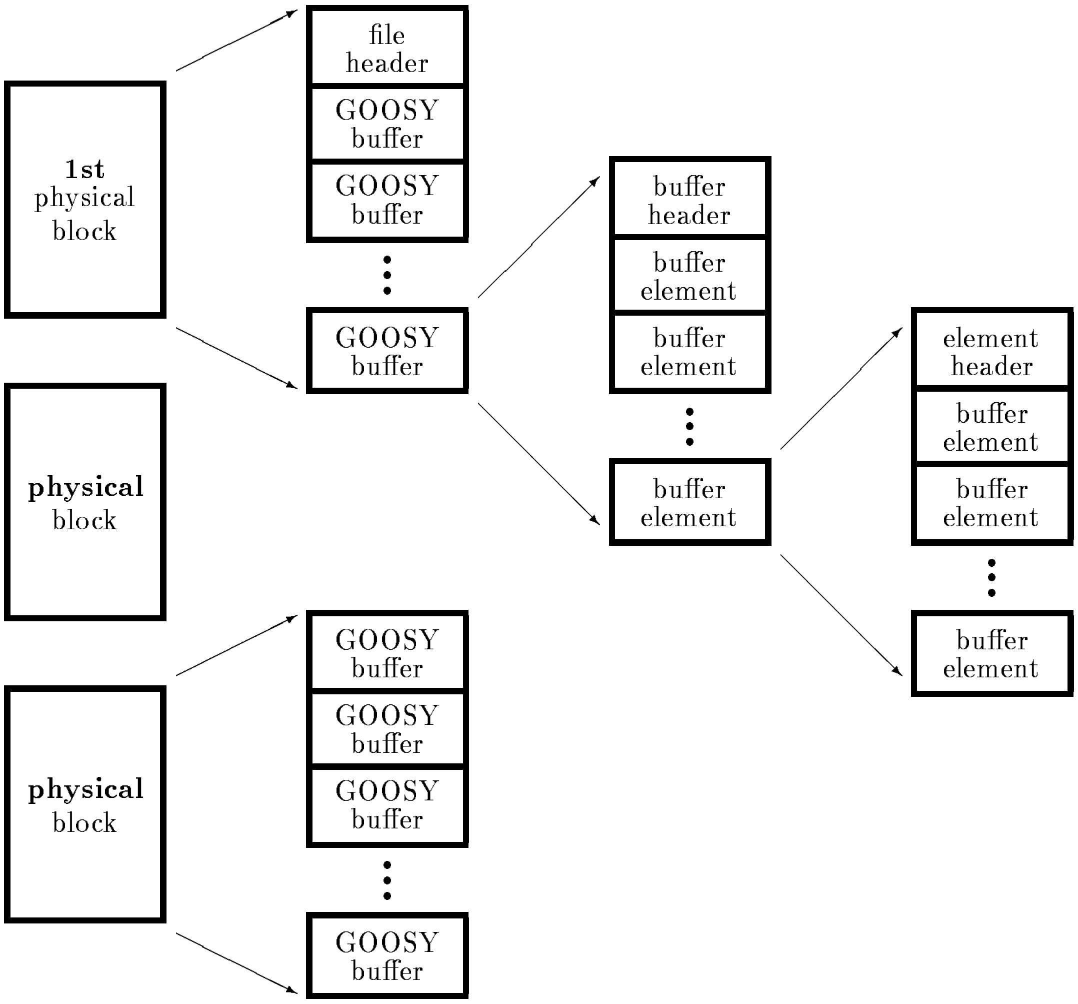

Figure C.2: The GOOSY data structures of a listmode dump file.

C.3.3 Buffer Header

Buffer Header

31 28 24	20 16	12 8	4 0	Offset
Length of data field (buffer without header)				0
Sub Type		Type		4
Fragment begin	Fragment end	Used Length of Data field		8
Buffer Number for this Type - Sub Type				12
Number of Buffer Elements				16
Current Index				20
Time stamp VMS 64 bit format				24
Time stamp VMS 64 bit format				28
Byte order tag				32
Length of last event				32
2 Longwords reserved				32
2nd Data word		1st Data word		48
...				

Figure C.3: Buffer Header Structure

The total length of the buffer header is 48 bytes.

Length of data field Length of the buffer without this buffer header in 16–bit words.

(BIN FIXED (31)).

Type A number specifying the buffer type.

(BIN FIXED (15)).

Sub Type A number specifying the buffer subtype.

(BIN FIXED (15)).

Used Length of Data Field Number of 16–bit words actually used in the Data field in this buffer.

(BIN FIXED (15)).

Fragment begin If this byte is= 1, the buffer contains a fragment (the first part of a buffer element, which is not complete) at the end of the buffer. The fragment is missing its trailing part, which has to be found in the following buffer of the same type and subtype.

(BIT(8)).

Fragment end If this byte is= 1, the buffer contains a fragment (the rest of a buffer element which is not complete) at the begin of the buffer. The fragment is missing its first part, which had to be found in the preceding buffer of the same type and subtype.

(BIT(8)).

Number of buffer elements This number is needed to decide in the case of fragment begin and fragment end, if there are two different fragments or only one fragment. A fragment is counted like a buffer element.

(BIN FIXED (31)).

Buffer Number A current number of buffers of the same type.

(BIN FIXED (31)).

Current Index A longword to store the index of the last processed event. This filed can be used by routines processing the buffer to store the index of the last processed buffer element. If the buffer is stored on disk or tape this field must be zero or 1.

(BIN FIXED (31)).

Time stamp A quadword for the system time in VAX/VMS binary format. This is the number of 100-nanoseconds since 17–Nov–1858 00:00.

(BIT(64)).

Byte order tag The creator of the buffer writes a 1 here. Each program processing the buffer must check this field. If it founds a 1, byte ordering is OK, if not, a longword swap must be performed.

(BIN FIXED (31)).

Length of last event When the last event in the buffer is a fragment, the length field in the event header keeps the size of the fragment. The length of the total event is kept in the buffer header.

(BIN FIXED (31)).

2 Free Longwords Reserved

((2) BIN FIXED(31)).

Data Words The Data Field of the buffer has a length specified by "Length of Data field", where only those words are used for data as specified in "Used Length of Data Field". The structure of the "Data Words" field is specified by buffer type and subtype.

(any).

Structure Declaration

The PL/1 structure mapping this structure is in GOOINC(SA$BUFHE):

```
/* ============= GSI buffer structure ============================*/
DCL P_SA$bufhe          POINTER;
DCL 1 SA$bufhe      BASED(P_SA$bufhe),
    2 IA$bufhe_DLEN     BIN FIXED(15), /* Data length            */
    2 IA$bufhe_TLEN     BIN FIXED(15), /* Spare = 0              */
    2 IA$bufhe_TYPE     BIN FIXED(15), /* Type                   */
    2 IA$bufhe_SUBTYPE  BIN FIXED(15), /* Subtype                */
    2 IA$bufhe_USED     BIN FIXED(15), /* Used length            */
    2 HA$bufhe_END      BIN FIXED(7),  /* first buf.el.is fragment*/
    2 HA$bufhe_BEGIN    BIN FIXED(7),  /* last buf.el.is fragment */
    2 LA$bufhe_BUF      BIN FIXED(31), /* Buffer number          */
    2 LA$bufhe_EVT      BIN FIXED(31), /* number of fragments    */
    2 LA$bufhe_CURRENT_I BIN FIXED(31),/* for unpack             */
    2 LA$bufhe_TIME(2)  BIN FIXED(31), /* time stamp             */
    2 LA$bufhe_FREE(4)  BIN FIXED(31), /* Byte order tag         */
                                       /* Length of last event   */
                                       /* free                   */
                                       /* free                   */
    2 IA$bufhe_DATA(1 REFER(IA$bufhe_DLEN))
                        BIN FIXED(15); /* data field             */
/*---------------------------------------------------------------*/
```

C.3.4 Buffer Element Header

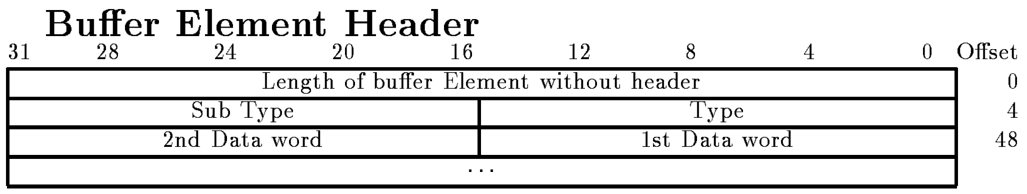

Figure C.4: Buffer Element Header Structure

The total length of the buffer element header is 8 bytes.

Length of buffer element Length of the buffer element without this header in 16–bit words.

(BIN FIXED (31)).

Type	A number specifying the buffer element type. (BIN FIXED (15)).
Sub Type	A number specifying the buffer element subtype. (BIN FIXED (15)).
Data	Any structure of data depending on type and subtype. (any).

Structure Declaration

The PL/1 structure mapping this structure is in GOOINC(SA$EVHE):

```
/* ================ GSI Event header ===================== */
DCL P_SA$evhe       POINTER;
DCL 1 SA$evhe         BASED(P_SA$evhe),
    2 LA$evhe_dlen    BIN FIXED(31), /* data length in words */
    2 IA$evhe_type    BIN FIXED(15), /* type                 */
    2 IA$evhe_subtype BIN FIXED(15), /* subtype              */
    2 IA$evhe_data(1 REFER(IA$evhe_dlen))
                      BIN FIXED(15); /* first data word      */
/*-------------------------------------------------------------*/
```

C.3.5 Event Spanning

Events could sometimes be bigger than a buffer. Therefore an event may span over buffer boundaries. The two bits in the buffer header specify if the first or last element in the buffer are fragments. When the last element is a fragment, the length field keeps the length of the fragment. The total length is in the buffer header. The next buffer contains a fragment at the beginning. This fragment is preceeded by an element header (see above). The length field keeps the length of the fragment, type and subtype are the same as for the first fragment.

NOTE Any software processing buffers must be prepared to get buffers with 'lonely' fragments, i.e. at the beginning of a file there might be a fragment. Similar the last buffer in a file may contain a fragment at the end.

C.4 Buffer Types

C.4.1 Overview

Presently the following buffer types and buffer element types are used

2000,1	File header. Buffer header plus one buffer element data.
3000,1	Acknowledge buffer. This buffer contains no data but marks the end of a buffer stream.
1 ,1	MBD buffer. This is a no standard GOOSY header. The buffer must be processed by user written routines.
2 ,1	Buffer contains J11 generated SILENA formatted events with standard header of type:
	1 ,1 SILENA formatted subevents.
3 ,1	Buffer contains compressed buffer elements of type:
	3 ,1 Compress mode 1
	3 ,2 Compress mode 2
4 ,1	Buffer contains events of type:
	4 ,1 uncompressed events
	4 ,2 compressed events (zeros suppressed)
5 ,1	Buffer contains LRS FERA events with standard header of type:
	5 ,1 FERA formatted subevents
6 ,1	Buffer contains standard MBD events of type:
	6 ,1 standard events with structure defined by J11 programs.
7 ,s	Buffer contains standard MBD events of type:
	7 ,s events with user structure defined by J11 programs.
	The subtype numbers can be specified by the user.
10 ,1	Buffer contains VME formatted events of type 10,1.

10 ,1	standard event written by VME system. Event is composed by subevents of type 10,1 and 10,2.

12 ,1	Buffer contains SILENA formatted events without standard header as stored in FERA memory.
15 ,1	Buffer contains LRS FERA events without standard header as stored in FERA memory.
1000,s	GOOSY Data Element. Type specified by s.

1000,1	GOOSY spectrum
1000,2	GOOSY condition
1000,3	GOOSY picture
1000,4	GOOSY polygon
1000,5	GOOSY calibration
1000,6	GOOSY Data Element (any)

10101,n	External user buffer type (Mainz).
10102,n	External user buffer type (THD).
10103,n	External user buffer type (CAVEB).
any	Any buffer may contain following element types

9000,1	time stamp
2001,1	CAMAC Readout table (initialization)
2001,2	CAMAC Readout table (readout)
2001,3	CAMAC Readout table (reset)
2002,1	Fastbus readout table (init)
2003,1	VME Readout table (init)

C.4.2 File Header Buffer

Figure C.5 shows the GOOSY File header structure. Note, that the File Header Buffer is a standard GOOSY buffer.

Buffer header information:

Length of data field Depends on buffer length.
(BIN FIXED(31))

Type A number specifying the buffer type. For this file header always = 2000.
(BIN FIXED(15))

Subtype A number specifying the buffer subtype. For this file header always = 1.
(BIN FIXED(15))

Used Length of Data Field Depends on length of comment.
(BIN FIXED(15))

Fragment begin This file header buffer contains *never* incomplete buffer elements. This field is always = 0.
(BIN FIXED(7))

Fragment end This file header buffer contains *never* incomplete buffer elements. This field is always = 0.
(BIN FIXED(7))

Number of Buffer Elements For this file header always = 1.
(BIN FIXED(31))

Buffer Number A current number of buffers of the same type.
(BIN FIXED(31))

Current Index A longword not used.
(BIN FIXED (31)).

Time stamp A quadword for the system time in VAX/VMS binary format. This is the number of 100-nanoseconds since 17–Nov–1858 00:00.
(BIT(64)).

4 Free Longwords ((4) BIN FIXED(31)).

File Header Buffer

31 28 24	20 16	12 8 4 0	Offset
Length of buffer without header			0
Buffer Subtype = 1		Buffer Type = 2000	4
Fragment begin= 0	Fragment end= 0	Used Length of Data field = 1000	8
Buffer Number for this Type - Sub Type			12
Number of Buffer Elements or Fragments of Buffer Elements = 1			16
Not used			20
Time stamp VMS 64 bit format			24
Time stamp VMS 64 bit format			28
4 Longwords reserved			32
...			
Tape label(30 char.)		Used length of tape label	48
Tape label continuation			
...			
File name (86 char.)		Used length of File name	80
File name continuation			
...			
User name (30 char.)		Used length of user name	168
User name continuation			
...			
Date "`dd-mmm-yyyy hh:mm:ss.mm`" (24 character)			200
Date continuation			
Run ID (66 char.)		Used length of Run ID	224
Run ID continuation			
...			
Experiment (66 char.)		Used length of Experiment	292
Experiment continuation			
...			
Number of Lines = n			360
Line 1 (78 char.)		Used length of Line 1	364
Line 1 continuation			
Line 2 (78 char.)		Used length of Line 2	
Line 2 continuation			
...			
Line n (78 char.)		Used length of Line n	
Line n continuation			

Figure C.5: File Header Structure

File header specific Information:

Used Length of Tape Label Number of characters used in the next field.
(BIN FIXED(15)).

Tape Label Contains the tape label of the ANSI tape, if the file was created on a tape.
(CHAR(30) VAR).

Used Length of File name Number of characters used in the next field.
(BIN FIXED(15)).

File name Name of file at the time of creation. The used Length is specified by the "Used Length of File name" field. 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

(CHAR(86) VAR).

Used Length of User name Number of characters used in the next field.
(BIN FIXED(15)).

User name User name of the creating VAX/VMS process.
(CHAR(30) VAR).

Date Character string of the creation date in the format "`dd-mmm-yyyy hh:mm:ss.mm` " where `dd` is the day of month, `mmm` is the 3 character abbreviation of the english spelled month (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC), and `yyyy` is the year, `hh` are hours, `mm` minutes, `ss.mm` are seconds, e.g. "21-OCT-1986 14:34:30.10 ". This date string is always padded by a space character.
(CHARACTER(24)).

Used Length of Run Identification Number of characters used in the next field.
(BIN FIXED(15)).

Run Identification Character string to identify the experiment run corresponding to this file. The contents is user defined. The string can have a maximum of 66 characters. The actual length is defined by the "Length of Run Identification" field.
(CHARACTER(66) VAR).

Used Length of Experiment Name Number of characters used in the next field.

(BIN FIXED(15)).

Experiment Name Character string to identify the experiment corresponding to this file. The contents is user defined. The string can have a maximum of 66 characters. The actual length is defined by the "Length of Experimenter Name" field.

(CHARACTER(66) VAR).

Number of Lines Number of 78–character lines following.

(BIN FIXED(31)).

Used Length of line Number of characters used in the next field.

(BIN FIXED(15)).

Comment Lines Character string array to characterize the contents of this file. The lines are user defined. The header can have a maximum of 46 lines. The actual number of lines is defined by the "Number of Lines" field.

((*) CHARACTER(78) VAR).

Structure Declaration

The file header buffer is mapped by the PL/1 structure GOOINC(SA$FILHE):

```
/* ============ GSI file header buffer structure ================*/
DCL L_SA$filhe_lines    BIN FIXED(31);/* number of lines         */
DCL P_SA$filhe          POINTER;
DCL 1 SA$filhe      BASED(P_SA$filhe),
    2 IA$filhe_DLEN     BIN FIXED(15), /* Data length            */
    2 IA$filhe_TLEN     BIN FIXED(15), /* Total length           */
    2 IA$filhe_TYPE     BIN FIXED(15), /* Type                   */
    2 IA$filhe_SUBTYPE  BIN FIXED(15), /* Subtype                */
    2 IA$filhe_USED     BIN FIXED(15), /* Used length            */
    2 HA$filhe_END      BIN FIXED(7),  /* first event is fragment */
    2 HA$filhe_BEGIN    BIN FIXED(7),  /* last event is fragment  */
    2 LA$filhe_BUF      BIN FIXED(31), /* Buffer number          */
    2 LA$filhe_EVT      BIN FIXED(31), /* number of fragments    */
    2 LA$filhe_CURRENT_I BIN FIXED(31),/* for unpack             */
    2 LA$filhe_TIME(2)  BIN FIXED(31), /* time stamp             */
    2 LA$filhe_FREE(4)  BIN FIXED(31), /* free                   */
    2 CA$filhe_label    CHAR(30) VAR,  /* tape label             */
    2 CVA$filhe_file    CHAR(86) VAR,  /* file name              */
    2 CA$filhe_user     CHAR(30) VAR,  /* user name              */
    2 CA$filhe_time     CHAR(24),      /* time and date          */
    2 CVA$filhe_run     CHAR(66) VAR,  /* run id                 */
    2 CVA$filhe_exp     CHAR(66) VAR,  /* experiment             */
    2 LA$filhe_lines    BIN FIXED(31), /* number of lines        */
    2 CVA$filhe_line(L_SA$filhe_lines REFER(LA$filhe_lines))
                        CHAR(78) VAR;  /* comment lines          */
/*-------------------------------------------------------------*/
```

C.4.3 GOOSY Data Element Buffers

These buffers of type 1000 contain GOOSY Data Elements. These are encoded in special structures. The subtype may be used to select different Data Element types. (Not yet impl.)

C.4.4 GOOSY Listmode Data Buffers

Listmode data buffers contain buffer elements called *events*. The different event types are described in the next section.

C.5 Event Structures

C.5.1 Event Type 3 (compressed)

Figure C.6 shows the event structure of type 3. Behind the header there follows one Data Element which is compressed. Two compress modes are supported. One adds a BIT(32) longword for each 32 Longwords. Zero longwords are suppressed and marked in the bitstring. The other adds counter longwords containing the number of following zero or nonzero longwords. **These buffer elements are longword aligned!**

Event Type 3

31 … 16	15 … 0	Offset
Data length		0
Subtype = 1, 2	Type = 3	4
Compression mode		8
Length of uncompressed data		12
First compressed longword		16
…		

Figure C.6: Event structure type 3 (compressed)

Compression mode Two modes are provided: Bit mask mode and counter mode. (BIN FIXED (31)).

Length of uncompressed data Length of the original Data Element. (BIN FIXED (31)).

Usage

The analysis program can output Data Elements event by event. These Data Elements are copied to GOOSY buffers. Two storage modes can be selected: Compress and Copy mode. With compress mode the above structure is copied to the buffer. The original structure of the Data Element is lost. If the buffer is input by another analysis, the compressed buffer element is decompressed and restored. The advantage is that arbitrary data structures can be compressed, the disadvantage, that the compress/decompress procedure is time consuming. The pack routine is X$PACMP, the unpack routine X$UPCMP.

C.5.2 Event Type 4

Event Type 4, Subtype 1 (block)

Figure C.7 shows the event structure of type 4. Behind the header one Data Element follows. The structure is processed as a word array. **These buffer elements are NOT longword aligned!**

Event Type 4, Subtype 1

31 28 24 20 16	12 8 4 0	Offset
Data length		0
Subtype = 1	Type = 4	4
second data word	First data word	8
. . .		

Figure C.7: Event structure type 4, subtype 1 (block)

Event Type 4, Subtype 2 (no zero's)

Figure C.8 shows the event structure of type 4. Behind the header one Data Element follows. The structure is processed as a word array. Each data word is specified by an identification number, e.g. an ADC number. **These buffer elements are longword aligned!**

Event Type 4, subtype 2

31 28 24 20 16	12 8 4 0	Offset
Data length		0
Subtype = 2	Type = 4	4
First data word	First data word id	8
Second data word	Second data word id	12
. . .		

Figure C.8: Event structure type 4, subtype 2 (no zeros's)

Structure Declaration

Both event structures are copied to a Data Element in the Data Base with structure GOOTYP(SA$EVENT):

```
/* ================ GSI Event header 4,1 ====================*/
DCL P_SA$event       POINTER;
DCL 1 SA$event         BASED(P_SA$event),
    2 IA$event_dlen    BIN FIXED(15),  /* data length in words */
    2 IA$event_tlen    BIN FIXED(15),  /* not used =0          */
    2 IA$event_type    BIN FIXED(15),  /* type = 4             */
    2 IA$event_subtype BIN FIXED(15),  /* subtype = 1          */
    2 IA$event(512)    BIN FIXED(15);  /* data.                */
/*---------------------------------------------------------*/
```

Note that this structure contains no REFER because it is used to create the event Data Element in the Data Base. For special purposes the user may create his own event structure. The first four words must be declared as shown above.

Usage

The analysis program can output Data Elements event by event. These Data Elements are copied to GOOSY buffers. Two storage modes can be selected: Compress and Copy mode. With copy mode the above structure (subtype 1) is copied to the buffer. The original structure of the Data Element is lost. If the buffer is input by another analysis, the buffer element is copied back to the Data Element. The advantage is that arbitrary data structures can be copied, the disadvantage that no compression is done. The original Data Element must have a standard header.

Both formats are also used by the CAMAC single crate system controlled by a J11. The zero suppression can be enabled during data acquisition. The unpack routine for these events is X$UPEVT.

C.5.3 Event Type 1 (Buffer Type 2, SILENA)

These events have a standard buffer element header. This header must be generated by the processor reading out the ADC. Otherwise these events are stored without header in buffers of type 12. As shown in figure C.9, the buffer element is composed of a header followed by several Data Elements. These Data Elements are produced by the ADC/TDC modules type SILENA 4418x. **These buffer elements are NOT longword aligned!**

Event Type 1

31 28 24 20 16		12 8	4 0	Offset
Data length				0
Subtype = 1	Type = 1			4
Pattern Word	1	No. of Data Words	Subevent Id.	8
2nd Data Word	1st Data Word			12
. . .				
Pattern Word	1	No. of Data Words	Subevent Id.	
2nd Data Word	1st Data Word			
. . .				

Figure C.9: Event structure type 1 (SILENA)

Sub Event Id A number from 0 to 127 defining the sub event to which the following pattern word belongs. **This byte is NOT longword aligned!**

(BIN FIXED (7)).

Number of Data Words The number of data words (e.g. ADC values) following the pattern word. This number must be identical to the number of bits set in the pattern word.

(BIN FIXED (7) but with the highest bit, the event tag bit set to 0 !!).

Event Tag Bit The bit 2^{15} marks the event tag word. This bit is set in subevent header longwords.

(BIT (1)).

Pattern Word Each bit in the pattern word corresponds to a data word (e.g. ADC value) following this pattern word. The bit 2^0 corresponds to the first word. The number of bits set in the pattern word must be identical to the "Number of Data Words" field of this Simple Event Structure.

(BIT (16)).

Data Words The number of 16 bit data words (e.g. ADC data) is defined by the number of bits set in the pattern word or the identical "Number of Data Words" field in the structure.

((n) BIN FIXED (15)).

Usage

This format is presently not used.

C.5.4 Event Type 5 (LRS FERA)

These events have a standard buffer element header. This header must be generated by the processor reading out the ADC. Otherwise these events are stored without header in buffers of type 15. As shown in figure C.10, the buffer element is composed of a header followed by several Data Elements. These Data Elements are produced by the ADC/TDC modules type LRS 4300 (FERA). **These buffer elements are NOT longword aligned!**

Subevent Id A number defining the sub event to which the following subevent belongs. **This byte is NOT longword aligned!**

(BIN FIXED (7)).

# Data Words The number of data words (e.g. ADC values) following the pattern word.

(BIN FIXED (7) but with the highest bit, the event tag bit set to 0 !!).

Event Tag Bit The bit 2^{15} marks the event tag word. This bit is set in subevent header longwords.

(BIT (1)).

Event Type 5

31 28 24 20 16 12 8 4 0 Offset

									Offset
Data length									0
Subtype = 1			Type = 5						4
0	1st SA	1st data word	1	# Data Words	0	0	0	Subevent Id	8
0	3rd SA	3rd Data Word	0	2nd SA	2nd Data Word				12
...									
0	1st SA	1st data word	1	# Data Words	0	0	0	Subevent Id	
0	3rd SA	3rd Data Word	0	2nd SA	2nd Data Word				
...									

Figure C.10: Event structure type 5 (LRS FERA)

Data Words The number of 11 bit data words (e.g. ADC data) is defined by "number of data words". The source of the data words is specified by the "SA" field.

(BIT(11)).

SA Subaddress of the source of the data word.

(BIT(4)).

Usage

This format is presently not used.

C.5.5 Event Type 6 (MBD buffer type 6)

Figure C.11 shows the event structure of type 6 in buffers of type 6. The subevent structure is produced by the J11 and MBD programs. **These buffer elements are NOT longword aligned!**

Subevent length Length of subevent in words **excluding** header longword. **This word is NOT longword aligned!**

(BIN FIXED (15)).

CAMAC crate The number of the CAMAC crate where the subsequent subevent data came from.

(BIN FIXED (7))

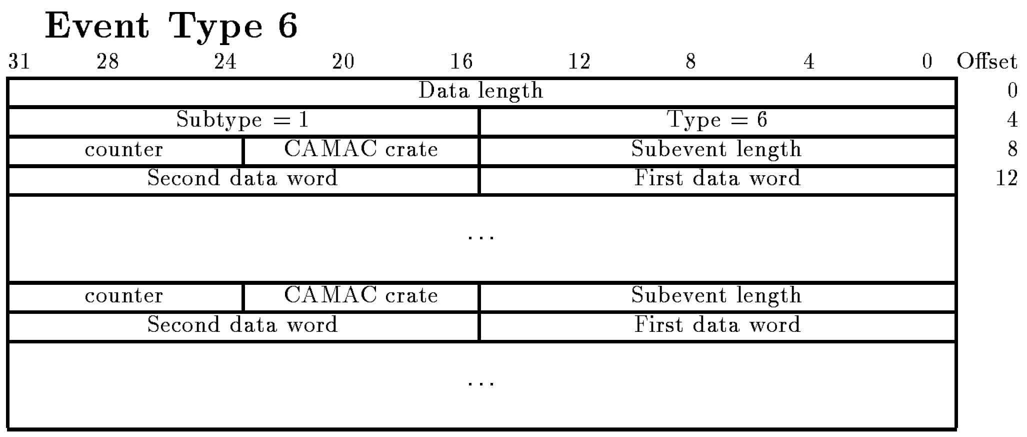

Figure C.11: Event structure type 6 (MBD buffer type 6)

Counter A counter to check correct order of events and subevents.
(BIN FIXED (7))

Data Words Data.
(BIN FIXED (15))

Structure Declarations

The subevent structure is mapped by the PL/1 structure GOOINC(SA$ME6_1):

```
/*======== Declaration of MBD event structure 6,1 ===========*/
  DCL P_SA$ME6_1 POINTER INIT(NULL);
  DCL 1 SA$ME6_1 BASED(P_SA$ME6_1),
 2 IA$ME6_1_slen    BIN FIXED(15), /* subevent length */
 2 HA$ME6_1_crate   BIN FIXED(7),  /* crate           */
 2 HA$ME6_1_event   BIN FIXED(7),  /* event count     */
 2 IA$ME6_1_data(IA$ME6_1_slen)
     BIN FIXED(15), /* data words      */
 2 SA$ME6_1_next,
   3 IA$ME6_1_nslen   BIN FIXED(15),
   3 HA$ME6_1_nscrate BIN FIXED(7);
/*-----------------------------------------------------------*/
```

The event structure is copied to a Data Element in the Data Base with structure GOOTYP(SA$MBD):

```
/* ===== Declaration of MBD event structure 6,1 ===== */
  DCL P_SA$MBD POINTER;
  DCL 1 SA$MBD BASED(P_SA$MBD),
2 IA$MBD_dlen    BIN FIXED(15),
2 IA$MBD_tlen    BIN FIXED(15),
2 IA$MBD_type    BIN FIXED(15),
2 IA$MBD_subtype   BIN FIXED(15),

        2 SA$MBD_C1,
3 IA$MBD_C1_slen   BIN FIXED(15), /* subevent length    */
3 IA$MBD_C1(99)    BIN FIXED(15), /* data words */
        2 SA$MBD_C2,
3 IA$MBD_C2_slen   BIN FIXED(15), /* subevent length    */
3 IA$MBD_C2(99)    BIN FIXED(15), /* data words */
        2 SA$MBD_C3,
3 IA$MBD_C3_slen   BIN FIXED(15), /* subevent length    */
3 IA$MBD_C3(99)    BIN FIXED(15), /* data words */
        2 SA$MBD_C4,
3 IA$MBD_C4_slen   BIN FIXED(15), /* subevent length    */
3 IA$MBD_C4(99)    BIN FIXED(15), /* data words */
        2 SA$MBD_C5,
3 IA$MBD_C5_slen   BIN FIXED(15), /* subevent length    */
3 IA$MBD_C5(99)    BIN FIXED(15), /* data words */
        2 SA$MBD_C6,
3 IA$MBD_C6_slen   BIN FIXED(15), /* subevent length    */
3 IA$MBD_C6(99)    BIN FIXED(15), /* data words */
        2 SA$MBD_C7,
3 IA$MBD_C7_slen   BIN FIXED(15), /* subevent length    */
3 IA$MBD_C7(99)    BIN FIXED(15); /* data words */
```

Note that this structure contains no REFER because it is used to create the event Data Element in the Data Base. For special purposes the user may create his own event structure. The first four words must be declared as shown above. If the length of the subcrate structures are different, a special unpack routine must be provided.

Usage

This will be the standard MBD event structure. The event Data Element with the structure SA$MBD will be filled by a standard unpack routine X$UPMBD.

C.5.6 Event Type 7 (MBD buffer type 7)

Figure C.12 shows the event structure of type 7 in buffers type 7. The subevent structure is provided by the user. **These buffer elements are NOT longword aligned!**

Event Type 7

31 28 24 20 16	12 8 4 0	Offset
Data length		0
Subtype = s	Type = 7	4
Second data word	First data word	8
...		

Figure C.12: Event structure type 7 (MBD buffer type 7)

Type	Must be 7. (BIN FIXED (15)).
Subtype	The subtype can be specified by the user. The buffer subtype must be equal to this event subtype. (BIN FIXED (15)).
Data Words	Data. Contains subevent structures defined by the user. Different structures can be marked by different type/subtype numbers. (BIN FIXED (15))

Usage

This type allows users to write specific applications requiring specific event structures.

C.5.7 Event Type 10 (VME)

This structure is composed by the EB. It is mapped by SA$VE10_1 in library GOOINC.

```
/* ================= GSI VME Event header ====================== */
DCL P_SA$ve10_1       POINTER;
DCL 1 SA$ve10_1         BASED(P_SA$ve10_1),
    2 LA$ve10_1_dlen    BIN FIXED(31),
    2 IA$ve10_1_type    BIN FIXED(15),
    2 IA$ve10_1_subtype BIN FIXED(15),
    2 IA$ve10_1_dummy   BIN FIXED(15),
```

Event Type 10, Subtype 1

31 28 24 20 16	12 8 4 0	Offset
Data length [words]		0
Subtype = 1	Type = 10	4
Trigger	Not used	8
Event counter		12
Subevent 1		16
. . .		
Subevent n		16+x

Figure C.13: Event Structure

```
    2 IA$ve10_1_trigger BIN FIXED(15),
    2 LA$ve10_1_count   BIN FIXED(31),
    2 IA$ve10_1(LA$ve10_1_dlen-4)    BIN FIXED(15),
    2 LA$ve10_1_next    BIN FIXED(31);
/*------------------------------------------------------------------*/
```

CAMAC Subevent Structure 10,1

This subevent structure is written by the ROP or the FEP. It is defined in SA$VES10_1 in library GOOINC.

Subevent Type 10, Subtype 1

31 28 24	20 16	12 8 4 0	Offset
Subevent Data length [words]			0
Subevent subtype = 1		Subevent type = 10	4
Control	subcrate	Processor ID	8
CAMAC value		CAMAC module ID	12
. . .			

Figure C.14: CAMAC Subevent Structure

```
/* ================ GSI VME Subevent header ====================== */
DCL P_SA$ves10_1       POINTER;
DCL 1 SA$ves10_1         BASED(P_SA$ves10_1),
    2 LA$ves10_1_dlen    BIN FIXED(31),
    2 IA$ves10_1_type    BIN FIXED(15),
    2 IA$ves10_1_subtype BIN FIXED(15),
    2 IA$ves10_1_procid  BIN FIXED(15),
    2 HA$ves10_1_subcrate BIN FIXED(7),
    2 HA$ves10_1_control BIN FIXED(7),
    2 IA$ves10_1(LA$ves10_1_dlen-2)    BIN FIXED(15),
    2 LA$ves10_1_next    BIN FIXED(31);
/*-------------------------------------------------------------------*/
```

FASTBUS Subevent Structure 10,2

This subevents are written from the AEB. The header structure is defined in SA$VES10_1 in library GOOINC.

Subevent Type 10, Subtype 2

31 28	24 20	16 12 8 4 0	Offset
Subevent Data length [words]			0
Subevent subtype = 2		Subevent type = 10	4
Control	subcrate	Processor ID	8
Fastbus module header			16
...			

Figure C.15: Fastbus Subevent Structure

The following structure maps to the data field. It is defined in SA$vesfb in library GOOINC.

Fastbus module header

31 28	24 20	16 12 8 4 0	Offset
Longwords	Geo.addr.	Module ID	0
1st data word			4
...			

Figure C.16: Fastbus Module header

```
/* Fastbus module header maps to IA$ves10_2(i) */
DCL P_SA$vesfb       POINTER;
DCL 1 SA$vesfb BASED(P_SA$ves_fb),
    2 IA$vesfb_id BIN FIXED(15),
    2 HA$vesfb_addr BIN FIXED(7),
    2 HA$vesfb_lwords BIN FIXED(7),
    2 LA$vesfb_data(HA$vesfb_lwords)    BIN FIXED(31),
    2 LA$vesfb_next    BIN FIXED(31);
```

Fastbus data word

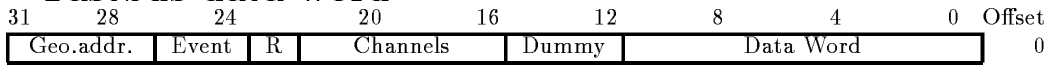

Figure C.17: Fastbus Data Word

One data word looks like

```
/* Structure of data words */
/* Numbers from 1 to 32 can be used in POSINT */
%REPLACE FBDATA_d    BY 1;   %REPLACE FBDATA_d_l  BY 12;
%REPLACE FBDATA_x    BY 13;  %REPLACE FBDATA_x_l  BY 4;
%REPLACE FBDATA_ch   BY 17;  %REPLACE FBDATA_ch_l BY 7;
%REPLACE FBDATA_r    BY 24;  %REPLACE FBDATA_r_l  BY 1;
%REPLACE FBDATA_ev   BY 25;  %REPLACE FBDATA_ev_l BY 3;
%REPLACE FBDATA_ad   BY 28;  %REPLACE FBDATA_ad_l BY 5;
DCL P_SI$FBDATA POINTER; /* maps to LA$vesfb_data(i) */
DCL 1 SI$FBDATA BASED(P_SI$FBDATA),
    2 BI$FBDATA_d  BIT(12) /* data word */
    2 BI$FBDATA_x  BIT(4), /* dummy     */
    2 BI$FBDATA_ch BIT(7), /* channel   */
    2 BI$FBDATA_r  BIT(1), /* range     */
    2 BI$FBDATA_ev BIT(3), /* event     */
    2 BI$FBDATA_ad BIT(5); /* geo addr. */
/*-------------------------------------------------------------------*/
```

C.6 Buffer Element Structures

C.6.1 Buffer Element Type 9000 (Time Stamp)

Figure C.18 shows the Time Stamp structure (buffer element structure type 9000). **These buffer elements are longword aligned!**

Buffer Element Type 9000, Time Stamp

31 28 24 20 16	12 8 4 0	Offset
Data length=10		0
Subtype = 0	Type = 9000	4
Date "`dd-mmm-yyyy hh:mm:ss.mm`" (24 character)		8
Date continuation		

Figure C.18: Buffer Element structure type 9000, Time Stamp

Date Character string of the creation date in the format "`dd-mmm-yyyy hh:mm:ss.mm` " where `dd` is the day of month, `mmm` is the 3 character abbreviation of the english spelled month (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC), and `yyyy` is the year, `hh` are hours, `mm` minutes, `ss.mm` are seconds, e.g. "21-OCT-1986 14:34:30.10 ". This date string is always padded by a space character. (CHARACTER(24)).

Usage

Not yet used.

C.6.2 Buffer with GOOSY Data Elements

Buffer type 1000 contains GOOSY Data Elements. The subtype specifies the kind of Data Element.

GOOSY spectrum

GOOSY condition

GOOSY picture

GOOSY polygon

GOOSY calibration

GOOSY Data Element

C.7 Nonstandard Buffer Structures

C.7.1 Buffer Type 12 (SILENA)

Figure C.19 shows the subevent structure as produced by one ADC/TDC module of type SILENA 4418x. Several modules produce several subsequent structures. **These buffer elements are NOT longword aligned!**

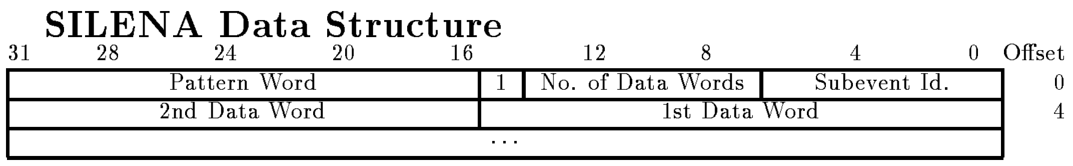

Figure C.19: Data structure SILENA ADC

Subevent Id A number from 0 to 127 defining the subevent to which the following pattern word belongs. **This byte is NOT longword aligned!**

(BIN FIXED (7)).

Number of Data Words The number of data words (e.g. ADC values) following the pattern word. This number must be identical to the number of bits set in the pattern word.

(BIN FIXED (7) but with the highest bit, the event tag bit set to 0 !!).

Event Tag Bit The bit 2^{15} marks the event tag word. This bit is set in the subevent header longword.

(BIT (1)).

Pattern Word Each bit in the pattern word corresponds to a data word (e.g. ADC value) following this pattern word. The bit 2^0 corresponds to the first word. The number of bits set in the pattern word must be identical to the "Number of Data Words" field of this Simple Event Structure.

(BIT (16)).

Data Words The number of 16 bit data words (e.g. ADC data) is defined by the number of bits set in the pattern word or the identical "Number of Data Words" field in the structure.

((n) BIN FIXED (15)).

Usage

Not yet used.

C.7.2 Buffer Type 15 (LRS FERA)

Figure C.20 shows the subevent structure as produced by one ADC/TDC module of type LRS 4300 (FERA). Several modules produce several subsequent structures. **These buffer elements are NOT longword aligned!**

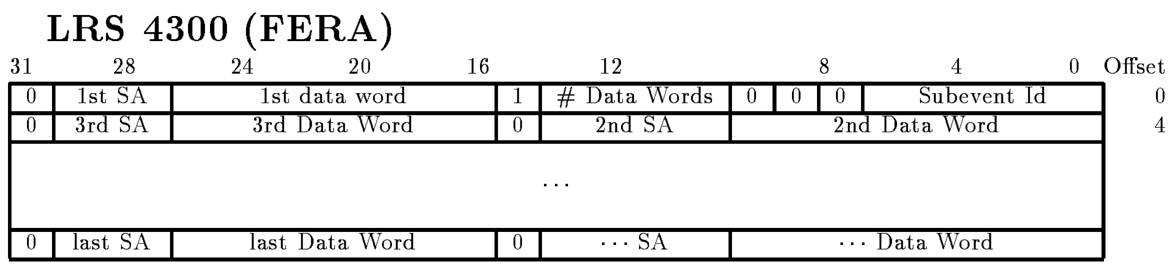

Figure C.20: Data structure LRS FERA

Subevent Id	A number defining the subevent to which the following subevent belongs. **This byte is NOT longword aligned!** (BIN FIXED (7)).
# Data Words	The number of data words (e.g. ADC values) following the pattern word. (BIN FIXED (7) but with the highest bit, the event tag bit set to 0 !!).
Event Tag Bit	The bit 2^{15} marks the event tag word. This bit is set in the subevent header longword. (BIT (1)).
Data Words	The number of 11 bit data words (e.g. ADC data) is defined by "number of data words". The source of the data words is specified by the "SA" field. (BIT(11)).
SA	Subaddress of the source of the data word. (BIT(4)).

Usage

Not yet used.

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 Look At Me. 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 Branch Driver 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

A

Analysis
- Manager 261

ATTACH 261

B

Branch 261
Buffer 228, 261
- Data Element 261
- element 228, 232
  - calibration 256
  - condition 256
  - data 228, 232
  - Data Element 256
  - header 228, 232, 236
  - picture 256
  - polygon 256
  - spectrum 256
  - time stamp 256
- file header 240
- files 228
- header 234
- non standard 232
- size 11, 12, 21, 22
- spanning 237
- structure 232
- subtype 234, 236
- type 234, 236, 238
- Unpack Routine 261

Byte order 229

C

CAMAC 261
Condition 262
CONNECT 262
control key 4
Ctrl
- keys 4

CVC 262

D

Data
- buffer 244
- Element
  - buffer 244

Data Base 262
- Directory 262
- Manager 262
- Pool 262

Data Element 262
- Member 262
- Type 262

Dynamic List 262
- entry 263
- executor 263

E

Endian 229
enter key 3
Environment 10, 263
ESONE
- VME 17

Event 229, 263
- Buffer Data Element 263
- compressed 245
- Data Element 263
- FERA
  - non standard 258

standard 248
J11 245
MBD
standard 249
user defined 252
SILENA
no standard 257
standard 247
spanning 237
types 238, 245, 247, 248, 249, 252, 257, 258
Unpack Routine 263
VME 252
CAMAC 253
FASTBUS 254
word block 245
word block - no zeros 246
Exclusive output 29

F

FASTBUS
event 254
File header 26, 31, 240
File names
IBM 27, 242
Fn keys 4
fragment 234

G

GOLD key 3

I

IBM
file names 27, 242

J

J11 263, 264
compress 30
load 21

K

key
enter 3
GOLD 3
keypad 3
GOLD 3

L

LAM 264
Load routines
acquisition 28
LOCATE 264

M

Mailbox 264
MBD 264
load 11
reset 11
Member of Data Element 262
Message control block 228, 230
MOUNT
Data Base 264

O

Object 264
Output 26

P

PERICOM terminal 3
PFn keys 4
Picture 264
frame 264
Prompter 263, 264

R

Reset MBD 11

S

Scatter plot 264
Spanning 237
Spectrum 264
Starburst 263, 264
Start
acquisition 22
Stop
acquisition 22

Supervisor 265
Synchronization 29
    acquisition 9, 25, 26, 29

T

Tape 27
    dismount 27
    mount 27
Transport Manager 265
    data check 30
    INITIALIZE
        file input 22
        foreign input 22
        J11 21
        MBD 11
        VME 12
    Input Channels 7
    mailbox 11, 12, 21, 22
    output 8
        DECnet 25
        file 26
        mailbox 25
    startup 10
    tape 27
Trigger
    set 16
Type of Data Element 262

U

Unpack
    Routine 265

V

VME
    buffers 15
    CNAF 17
    command 15
    debug 16
    ESONE 17
    event 252
        CAMAC 253
        FASTBUS 254
    files 12
    input 14
    load 13
    setup 12
    show 14, 15
    transport 14

$ANL 261
$DBM 261
$DSP 261
$TMR 261

Contents

1 Preface **3**
1.1 GOOSY Authors and Advisory Service . . . 5
1.2 Further GOOSY Manuals . . . 5

2 GOOSY Transport Manager **7**
2.1 **Introduction** . . . 7
2.2 **Startup the TMR** . . . 10
2.3 **Input Channels** . . . 11
2.3.1 MBD Input . . . 11
INITIALIZE ACQUISITION . . . 11
LOAD MBD . . . 11
2.3.2 VME Input . . . 12
VME setup files . . . 12
INITIALIZE ACQUISITION . . . 12
LOAD VME frontend processors . . . 13
LOAD VME readout tables . . . 13
SHOW VME SETUP . . . 14
Select Data Transport . . . 14
Commands . . . 15
Setup Readout buffers . . . 15
SHOW VME CONTROL . . . 15
SET Trigger . . . 16
Debug Output . . . 16
Initialize CAMAC . . . 16
VME messages . . . 17
CNAF . . . 17
START/STOP ACQUISITION . . . 17
TYPE EVENT . . . 18
Example VME session with one CAMAC crate . . . 19
2.3.3 J11 Input . . . 21
INITIALIZE ACQUISITION . . . 21

LOAD J11 . . . 21
2.3.4 File Input . . . 22
INITIALIZE ACQUISITION . . . 22
OPEN-CLOSE FILE . . . 22
2.3.5 Foreign Input . . . 22
INITIALIZE ACQUISITION . . . 22
2.3.6 START-STOP ACQUISITION . . . 22
START-STOP Routines . . . 23
2.4 **CAMAC Spectra** . . . 24
2.5 **Output Channels** . . . 25
2.5.1 Mailbox Output . . . 25
2.5.2 DECnet Output . . . 25
2.5.3 File Output . . . 26
START-STOP OUTPUT FILE . . . 26
GOOSY File Header . . . 26
Naming Conventions for IBM . . . 27
Tape Handling . . . 27
End of Tape . . . 27
2.6 **Loading Private Routines** . . . 28
2.6.1 LOAD MODULE ACQUISITION . . . 28
2.6.2 Enable/Disable Calling . . . 28
2.7 **Acquisition Synchronization** . . . 29
2.8 **Miscellaneous Commands** . . . 30
2.8.1 Data Checking . . . 30
2.8.2 Compress Mode . . . 30
2.8.3 SHOW ACQUISITION . . . 30
2.8.4 TYPE EVENT-BUFFER . . . 31
2.9 **Controlling the Acquisition** . . . 32
2.9.1 Checking Incoming Data . . . 32
2.9.2 Analyze Data . . . 32
2.9.3 Modifying Hardware Setup . . . 32
2.9.4 Writing to Tape . . . 33
2.9.5 Full Analysis . . . 33

APPENDIX **33**

A Transport Manager Commands **35**
$ CLOSE ETHERNET . . . 36
$ SET GNA ETHERNET . . . 37
$ SHOW GNA ETHERNET . . . 39
CALCULATE FASTBUS PEDESTAL . . . 40
CAMAC CLEAR . . . 42

CAMAC CNAF 43
CAMAC DEMAND 45
CAMAC INHIBIT 47
CAMAC INITIALIZE 49
CAMAC SCAN 50
CLOSE FILE 52
CLOSE OUTPUT FILE 53
CNAF VME 54
CREATE PROGRAM 56
DEBUG VME MEMORY 58
DISMOUNT TAPE 60
DUMP MBD 61
DUMP STARBURST 64
EXECUTE VME 67
FOREIGN ACQUISITION 69
INITIALIZE CAMAC 72
LOAD J11 73
LOAD LRS_2365 74
LOAD MBD 77
LOAD MODULE ACQUISITION 80
LOAD STARBURST 82
LOAD VME PROGRAM 85
LOAD VME TABLE 88
MOUNT TAPE 92
OPEN FILE 94
OPEN OUTPUT FILE 96
PATCH MBD 98
PATCH STARBURST 100
RELEASE MBD CHANNEL 102
RESET ACQUISITION 104
RESET CAMAC 105
RESET MBD 106
SEND DATA 107
SET ACQUISITION 109
SET FASTBUS PEDESTAL 111
SET RANDOM 113
SET VME BUFFER 114
SET VME CONTROL 116
SET VME INPUT 118
SET VME TRIGGER 119
SHOW ACQUISITION 121
SHOW STARBURST 123

SHOW VME CONTROL 125
SHOW VME SETUP 127
START ACQUISITION 128
START OUTPUT FILE 130
START RUN 132
START VME 133
STOP ACQUISITION 134
STOP OUTPUT FILE 135
STOP RUN 136
STOP VME 137
STORE LRS_2365 138
STORE MBD 140
TEST BOR_1802 142
TEST CAMAC 145
TEST GSI_IOL 147
TEST LRS_2228 150
TEST LRS_2249 153
TEST LRS_2551 156
TEST LRS_4432 159
TEST LRS_4434 162
TEST MPI_BIT 165
TEST MPI_TDC 168
TEST REGISTER 171
TEST SEN_2047 174
TEST SEN_2090 177
TYPE BUFFER 180
TYPE EVENT 181

B DCL Commands 183

ALIAS 184
ATENVIR 186
CLINK 187
COMMENT 188
CONCAT 191
CRENVIR 192
CTRL_T 195
DEBUG_WINDOW 196
DLENVIR 197
DTENVIR 198
ETHDEF 199
GOOCONTROL 200
GUIDE 201

LINKJ11 . . . 206
LSHARIM . . . 207
MANUAL . . . 210
MTAPE . . . 212
OPSER . . . 213
PLOTMET . . . 214
SELECT_MBD . . . 216
SETMESSAGE . . . 218
TLOCK . . . 220
VMESTRUC . . . 221
WCLOSE . . . 226

C GOOSY Data Formats **227**
C.1 **Introduction** . . . 228
C.1.1 Buffers . . . 228
C.1.2 Buffer Files . . . 228
C.1.3 Message Control Blocks . . . 228
C.1.4 Glossary . . . 228
C.2 **Message Control Block Structure** . . . 230
C.3 **Buffer Structure** . . . 232
C.3.1 Standard Buffers . . . 232
C.3.2 Nonstandard Buffers . . . 232
C.3.3 Buffer Header . . . 234
Structure Declaration . . . 236
C.3.4 Buffer Element Header . . . 236
Structure Declaration . . . 237
C.3.5 Event Spanning . . . 237
C.4 **Buffer Types** . . . 238
C.4.1 Overview . . . 238
C.4.2 File Header Buffer . . . 240
Structure Declaration . . . 244
C.4.3 GOOSY Data Element Buffers . . . 244
C.4.4 GOOSY Listmode Data Buffers . . . 244
C.5 **Event Structures** . . . 245
C.5.1 Event Type 3 (compressed) . . . 245
Usage . . . 245
C.5.2 Event Type 4 . . . 245
Event Type 4, Subtype 1 (block) . . . 245
Event Type 4, Subtype 2 (no zero's) . . . 246
Structure Declaration . . . 246
Usage . . . 247
C.5.3 Event Type 1 (Buffer Type 2, SILENA) . . . 247

Usage . . . 248
C.5.4 Event Type 5 (LRS FERA) . . . 248
Usage . . . 249
C.5.5 Event Type 6 (MBD buffer type 6) . . . 249
Structure Declarations . . . 250
Usage . . . 251
C.5.6 Event Type 7 (MBD buffer type 7) . . . 252
Usage . . . 252
C.5.7 Event Type 10 (VME) . . . 252
CAMAC Subevent Structure 10,1 . . . 253
FASTBUS Subevent Structure 10,2 . . . 254
C.6 **Buffer Element Structures** . . . 256
C.6.1 Buffer Element Type 9000 (Time Stamp) . . . 256
Usage . . . 256
C.6.2 Buffer with GOOSY Data Elements . . . 256
GOOSY spectrum . . . 256
GOOSY condition . . . 256
GOOSY picture . . . 256
GOOSY polygon . . . 256
GOOSY calibration . . . 256
GOOSY Data Element . . . 256
C.7 **Nonstandard Buffer Structures** . . . 257
C.7.1 Buffer Type 12 (SILENA) . . . 257
Usage . . . 258
C.7.2 Buffer Type 15 (LRS FERA) . . . 258
Usage . . . 259

GOOSY Glossary **261**

Index **266**