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

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

GOOSY Commands

H.G.Essel, GOOSY Authors

June, 28 1988

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

Chapter 1

GOOSY 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

$ COMMENT

```
$ COMMENT LINE1="first line of comments"
          LINE2="second line of comments"
          LINE3=l3 LINE4=l4 LINE5=l5
          LINE6=l6 LINE7=l7 LINE8=l8
        /TERM /ERRL /SLOG /GLOG /PLOG
```

PURPOSE Write comments to SYS$OUTPUT and log files.

PARAMETERS

LINE1..8 Lines of comment to be written to the destinations defined by the qualifiers. Note, that the comments must be enclosed in quotes if they contain blanks or other delimiters (as usually the case).
Up to eight line may be specified, blank lines will not be written.

/TERM Signals, that the comments are to be written to the SYS$OUTPUT stream.

/ERRL Signals, that the comments are to be written to the error log file.

/SLOG Signals, that the comments are to be written to the session log file.

/GLOG Signals, that the comments are to be written to the global log file.

/PLOG Signals, that the comments are to be written to the process log file. NOTE, that more than one qualifier may be specified. The default if none is specified is /TERM/SLOG.

FUNCTION This command writes up to 8 lines of comments to SYS$OUTPUT (usually terminal or batch log file) and/or to the session, global and process log file.

EXAMPLE $ COMMENT "This is a comment" /SLOG

Action rout. C$UCCOM

Author Walter F.J. Mueller

DESCRIPTION

CALLING

```
@CALL C$UCCOM(C_LINE1,C_LINE2,C_LINE3,C_LINE4,
              C_LINE5,C_LINE6,C_LINE7,C_LINE8,
              C_TERM,C_C_ERRL,C_SLOG,C_GLOG,C_PLOG);
```

ARGUMENTS

C_LINE1..8 CHAR(*) VAR [INPUT]
Up to 8 comment lines to be written, blank lines will be ignored.

C_TERM CHAR(*) VAR [INPUT]
/TERM qualifier, is equivalent to U$M_PRTTERM.

C_ERRL CHAR(*) VAR [INPUT]
/ERRL qualifier, is equivalent to U$M_PRTERRL.

C_SLOG CHAR(*) VAR [INPUT]
/SLOG qualifier, is equivalent to U$M_PRTSLOG.

C_GLOG CHAR(*) VAR [INPUT]
/GLOG qualifier, is equivalent to U$M_PRTGLOG.

C_PLOG CHAR(*) VAR [INPUT]
/PLOG qualifier, is equivalent to U$M_PRTPLOG.

FUNCTION Writes up to 8 lines to SYS$OUTPUT and log files with U$PRTCL. The procedure builds the U$PRTCL mode bit pattern from the passed qualifiers and calls U$PRTCL for each non blank line.

REMARKS -

EXAMPLE

```
@CALL C$UCCOM('Line 1',,,,,,,,,
       '/TERM',,,,,);
```

$ DCL

$ DCL "DCL command line"

PURPOSE Execute a single DCL command line.

PARAMETERS

COMMAND Command line to be executed.

FUNCTION This command executes the specified command line in the context of a spawned process.

EXAMPLE $ DCL "SH DEV /MOUNTED"

Action rout. C$UCDCL

Author Walter F.J. Mueller

Remarks

File name C$UCDCL.PPL

Dataset -

REMARKS -

Description

CALLING @CALL C$UCDCL(C_COMMAND)

ARGUMENTS

C_COMMAND CHAR(*) VAR [INPUT]
DCL command line to be executed.
The command is executed in the context of a spawned subprocess. Therefore all logical names and DCL symbols known while executing the command line.

FUNCTION The LIB$SPAWN service is used to spawn a subprocess for the execution of a single command line.

REMARKS -

$ DEBUG

$ DEBUG

PURPOSE	Enter DEBUGger.
PARAMETERS	-
FUNCTION	This command issues the a debug activation signal to start or enter the DEBUGger. The action is equivalent to ^ Y DEBUG for an image running in an unspawned DCL process but works also for spawned processes.
Action rout.	C$UCDBG
Author	Walter F.J. Mueller
EXAMPLE	$ DEBUG

Remarks

File name	C$UCDBG.PPL
Dataset	-
REMARKS	-

Description

CALLING	@CALL C$UCDBG
ARGUMENTS	-
FUNCTION	The DEBUG activation signal SS$_DEBUG is signaled in order to start or activate the debuger.
EXAMPLE	@CALL C$UCDBG();
REMARKS	-

$ DEFINE KEY

$ DEFINE KEY KEY=k EQUIVALENCE=e
IF_STATE=i SET_STATE=s
/NOECHO/LOCK_STATE/TERMINATE/PROTECT

PURPOSE Associates an equivalence string and a set of attributes with a key on the terminal keyboard.

PARAMETERS

KEY=k Name of the key to be defined. the following table lists the definable keys:

PF1,....,PF4
KP0,....,KP9
PERIOD, COMMA, MINUS, ENTER
LEFT, RIGHT, UP, DOWN
E1,....,E6
HELP, DO
F6,....,F20
CTRLA,....,CTRLZ

Note, that not all keys are available on all key boards and that not all keys are enabled all the time. The terminal has to be in APPLICATION mode to use KP0,....,KP9 and PERIOD,....,ENTER. The terminal has to be in /NOLINE_EDIT mode to use LEFT,...,DOWN, this however is strongly disencouraged.

EQUIVALENCE=e Specifies the string which is to be processed when the key is pressed. If the string contains any spaces or other delimiters, enclose the equivalence string in quotation marks.

IF_STATE=i Specifies the state, which must be in effect for the key definition to be in effect. If this parameter is omitted, the current state is used. The state name is an alphanumeric string. States are established with the /SET_STATE qualifier.

SET_STATE=s Causes the specified state-name to be set when the key is pressed. The state name can be any alphanumeric string. If the SET_STATE parameter is omitted, the current state that was locked remains in effect.

/NOECHO Determines whether or not the equivalence string is displayed on the screen after the key has been pressed. The default is to echo. You cannot use /NOECHO without the /TERMINATE qualifier.

/LOCK_STATE Specifies that the state set by the SET_STATE parameter remains in effect until explicitly changed. Otherwise, the state set by SET_STATE is in effect only for the next defineable key that is pressed or for the next read terminating character typed. The /LOCK_STATE qualifier can only be specified with the SET_STATE parameter.

/TERMINATE Specifies whether or not the current equivalence string is to be terminated (that is, processed) when the key is pressed. The default is not to terminate, which allows to press other keys before the equivalence string is processed. Pressing RETURN has the same effect as using /TERMINATE.

/PROTECT The key is protected against redefinition. Any attempt to redefine or delete this key will be rejected and results in an error message.
This option should be used only in rare cases. Its intended use is to protect keys which change the state (SET_STATE=s) because their redefinition could affect a lot of other key definitions and result in considerable confusion. The PF1 key, used as a general GOLD prefix is protected for that reason.

FUNCTION This command defines a key in the process key translation table. The function is analoguous to the DCL DEFINE/KEY command, look in this description.

EXAMPLE $ DEFINE KEY KP1 "$SHO COM" /TERM

Action rout. C$UCDFK

Author Walter F.J. Mueller

Remarks

File name C$UCDFK.PPL

Dataset -

REMARKS -

Description

CALLING @CALL C$UCDFK(C_KEY,C_EQUI,C_IFSTATE,C_SETSTATE,
C_ECHO,C_LOCKSTATE,C_TERMINATE,C_PROTECT);

ARGUMENTS

C_KEY CHAR(*) VAR [INPUT]
Name of key to be defined.

C_EQUI CHAR(*) VAR [INPUT]
Equivalence string.

C_IFSTATE CHAR(*) VAR [INPUT]
States which must be in effect for the key definition to be in effect.

C_SETSTATE CHAR(*) VAR [INPUT]
Specifies the state name to be set when the key is pressed.

C_ECHO CHAR(*) VAR [INPUT]
/ECHO/NOECHO qualifier, determines whether or not the equivalence string is displayed on your screen after the key has been pressed.

C_LOCKSTATE CHAR(*) VAR [INPUT]
/LOCKSTATE qualifier, specifies the state C_SETSTATEto be in effect until explicitly changed.

C_TERMINATE CHAR(*) VAR [INPUT]
/TERMINATE qualifier, specifies whether or not the current equivalence string is to be terminated when the key is pressed.

C_PROTECT CHAR(*) VAR [INPUT]
/PROTECT qualifier, specifies whether or not the the key is protected against redefinition.

FUNCTION This procedure defines a key. A full discussion is in the command description.

REMARKS -

EXAMPLE @CALL C$UCDFK('KP1','$SHO COM','DEFAULT',",
",",'/TERMINATE',");

$ RECALL

$ RECALL KEY1=k KEY2=ke KEY3=ke KEY4=ke /ALL

PURPOSE Recall, list or file previous commands.

PARAMETERS

KEY1=k Number or a leading substring of the command to be recalled.

KEY2..4=ke Extensions of the leading substring.

/ALL Signals, that the recallable commands are to be listed.

FUNCTION This procedure implements three different functions:
recall a line by line number or verb
list the recall buffer
present a full screen menu

recall The recall function is activated by the

$ RECALL key

command and is similar to the DCL RECALL command. The 'key' specifies the number or leading substring of the command you wish to recall. Command numbers range from 1 up. The most recently entered command is number 1. The term "leading substring" refers to the first several characters in a command string. The substring can be as short as a single character. If the substring is not unique, RECALL recalls the most recently issued command line that matches the substring.

list The list function is activated by the

$ RECALL /ALL

command and is similar to the DCL RECALL/ALL command. It will display all the commands in the recall buffer, along with their command numbers and a line containing the exit status message the execution was not successful.

screen The full screen mode is activated by the

$ RECALL

command (without parameters and qualifiers). It will present a menu containing the recallable command lines and supports five subfunctions:

scroll
select/deselect
prompt
execute
file

scroll The cursor can be moved with the up and down arrow function keys and the window can be scrolled up and down with the KP4 and KP5 keys respectively.

select The POINT function key select the current and all subsequent cursors line until the POINT function key is pressed again. The KP0 function key deselects lines in a similar way. The selected lines are highlighted in reverse video. Note that this select mechanism differs from the one used in the DEC editors and allows you to select a noncontiguous range of command lines.

prompt The prompt function (PF3) opens a window and prompts for a command to be executed. The window is initialized with the command in the cursor line which may be edited.

execute The execute function (ENTER) executes either all selected command lines or the cursor line if nothing has been selected.

file The file function (PF1) writes either all selected command lines or the cursor line in a file. The file name is prompted in window.

DESCRIPTION

CALLING @CALL C$UCREC(C_KEY1,C_KEY2,C_KEY3,C_KEY4,C_ALL);

ARGUMENTS

C_KEY1 CHAR(*) VAR [INPUT]
Number or leading substring of the command to be recalled.

C_KEY2..4 CHAR(*) VAR [INPUT]
Extension of the leading substring. C_KEY1..4 are concatinated with a single delimiting blank to form the effective leading substring.

C_ALL	CHAR(*) VAR [INPUT] /ALL qualifier, signals that the recallable commands are to be listed.
FUNCTION	This procedure implements three different functions: recall a line by line number or verb list the recall buffer present a full screen menu The full screen mode supports the recall and the filing of of a single or a group of command lines. For details look in the command description.
REMARKS	-
EXAMPLE	@CALL C$UCREC('1',",",",",",);

$ REPEAT

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

PURPOSE Repeat a list of commands.

PARAMETERS

COUNT Number of times the following commands are to be repeated.

LINE1..8 Up to eight command lines which are repeated. Note that the command has to be enclosed in quotes if it includes any delimites (as usually the case).

WAIT=w Time interval in seconds to be waited between repetitions. Note, that the default is 0.0 indicating no wait time.

/LOG Signals, that all commands are logged to SYS$OUTPUT before they are executed. The command line is prefixed by the iteration number and a ">".

/NOLOG Signals, that the commands are not to be logged. This is the default.

/ONWARNING Signals, that the repetition is aborted if a command returns with a severity of warning or worse.

/ONERROR Signals, that the repetition is aborted if a command returns with a severity of error or worse.

/ONFATAL Signals, that the repetition is aborted if a command returns with a severity of fatal. The default is /ONWARNING.

FUNCTION This command repeats COUNT times the command lines LINE1...8. The repetition is aborted if a command exits with a status equal or more severe than the abort threshold given by the /ONxxx qualifiers. NOTE, that prompting is disabled during the execution of a command procedure. A missing required parameter or a conversion error results in an error abort of that command and thus of the procedure.

EXAMPLE $ REPEAT 100 "$ SHO TIM"

Action rout. C$UCREP

Author Walter F.J. Mueller

DESCRIPTION

CALLING

```
@CALL C$UCREP(L_COUNT,
        C_LINE1,C_LINE2,C_LINE3,C_LINE4,
        C_LINE5,C_LINE6,C_LINE7,C_LINE8,
        R_WAIT,C_LOG,C_ABORT);
```

ARGUMENTS

L_COUNT BIN FIXED(31) [INPUT]
Repeat count, the command lines C_LINE1..8 will be executed L_COUNT times.

C_LINE1..8 CHAR(*) VAR [INPUT]
Commands to be executed repetively.

R_WAIT BIN FLOAT(24) [INPUT]
Time interval in seconds between repetitions.

C_LOG CHAR(*) VAR [INPUT]
Log qualifier set, either /LOG,/NOLOG or ''.

C_ABORT CHAR(*) VAR [INPUT]
Abort severity qualifier set, either /ONWARNING, /ONERROR, /ONFATAL or ''.

FUNCTION The commands given by C_LINE1..8 are executed L_COUNT times or until an error occurs. The error abort threshold is given by C_ABORT and can be warning, error or fatal.

REMARKS -

EXAMPLE

```
@CALL C$UCREP(100,'$ SHO TIME',,,,,,,,,
        0.0,,'/ONERROR');
```

$ RESET DEFAULT

$ RESET DEFAULT profile

PURPOSE Reset default parameters from profile (=same values than after startup).

PARAMETERS

profile Profile to read global parameter values

FUNCTION During program startup the profile is read and the values of global parameters are set to the values specified in the profile. These values can be changed by

$ SET DEFAULT parname=value.

where parname is the name of the parameter. It is a good practice to use the menu for that, because one can see all global parameters and their current values. The initial values as read from the profile are restored by command

$ RESET DEFAULT profile

where profile is the name of the profile.

NOTE Both commands must be prefixed by the component name if given in a GOOSY environment, e.g.

$ GOOSY> $ANL> $ RES DEF

to reset defaults for the analysis component $ANL.

Defined by C$UTLCM

Action rout. C$GTCPD

Description

CALLING STS$value=C$GTCPD(CV_profile,B_crecom) This description is only for the module.

ARGUMENTS

CV_profile Logical name of the profile to be used for input for the defaults. Presently C$GTCPD is called in C$DSPMN with the name 'GOO$PROFILE'

It has the form:
! comment
<program>:$INIT=<filespec>
<program>:<name>=<value>
A profile template may be created with the
$ SET DEFAULT command
The $INIT keyword defines a initialization CMD file
which is executed. In this file key definitions can be done.

B_crecom If set, the command $ SET DEFAULT is created and. executed. Otherwise it is executed only. In C$GTCPD this parameter must be '1'B to create the command $ SET DEFAULT. If called by command $ RESET DEFAULT B_crecom must be '0'B thus executing $ SET DEF with parameters from profile.

FUNCTION The module may be called at the end of all command definitions in a program (NOT in a procedure creating commands!). It collects all global replaced parameters and generates a parameter definition string for these. This string is used to create a command $ SET DEF which allows to set all the global parameters. The command is executed by C$EXECM using values read from the users profile.
The profile can be created by calling
the command explicitely an specifying a file name. All defaults can be specified, are set and are written to the file. The next time the program is started, the values from the profile are used to preset the defaults. If a profile alreday exists, a new created file must be included manually The profile has the format:
! comment
<program>:$INIT=<filespec>
<program>:<name>=<value>

REMARKS Utility commands C$UTLCM must be available before calling C$GTCPD.

EXAMPLE -

$ SET DEFAULT

$ SET DEFAULT profile program ....

PURPOSE Set default parameters.

PARAMETERS

profile If specified a profile written is into this file

program If specified, this name is written into the profile. Default is the current image name.

.... The other arguments depend on the program executing the command.

Defined by C$GTCPD

Action rout. C$GTCPD_DEF

Description

FUNCTION During program startup the profile is read and the values of global parameters are set to the values specified in the profile. These values can be changed by

$ SET DEFAULT parname=value.

where parname is the name of the parameter. It is a good practice to use the menu for that, because one can see all global parameters and their current values. The initial values as read from the profile are restored by command

$ RESET DEFAULT profile

where profile is the name of the profile.

NOTE Both commands must be prefixed by the component name if given in a GOOSY environment, e.g.

$ GOOSY> $ANL> $ RES DEF

to reset defaults for the analysis component $ANL.

$ 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 COMMAND

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

PURPOSE Show command definitions

PARAMETERS

tokeni Up to 4 command tokens may be specified. If none is given, all declared commands will be listed. If the given tokens refer to a subcommand tree, this will be listed.

output Optional output file for /NAME list

/BRIEF Signals, that only the command tokens and a short one line help is listed. This is the default if more than one command matches the given tokens.

/PARAM Signals, that all parameters are listed for the selected commands. The parameter name and either the qualifier set for SET's or a one line help is listed. This is the default, if only one command matches the given tokens.
NOTE: Pseudo parameters ($AD,$CL,$DP types) are not listed, use the FULL qualifier to see them.

/FULL Signals, that the command attributes, and all parameters with their attributes are listed. The pseudo parameters are listed.

/NAME The command keywords are output concatenated with underscores. The list of these names can be used to extract the command descriptions from a text lib.

Caller -

EXAMPLE $ SHOW COMM

DESCRIPTION

CALLING @CALL
C$UCSHC(C_TOKEN1,C_TOKEN2,C_TOKEN3,C_TOKEN4,
C_OUTPUT,C_MODE)

ARGUMENTS

C_TOKENi CHAR(*) VAR [INPUT]
Tokens of the commands to be listed.
If all strings are zero length, all commands will be listed.

C_OUTPUT CHAR(*) VAR [INPUT]
Optional output file name for /NAME

C_MODE CHAR(*) VAR [INPUT]
Mode qualifier set, may be zero length or
/BRIEF /PARAMETERS /FULL

FUNCTION This procedure lists the contents of the command dispatcher command definition tables. For details look in the command description.

REMARKS -

EXAMPLE @CALL C$UCSHC(",",",",",",",",");

$ SHOW GNA COMPONENTS

$ SHOW GNA COMPONENTS LINK=link NAME=name

PURPOSE Show status of GNA components.

PARAMETERS

LINK=l Specifies the object(s) whose status will be shown. The following input formats are supported and result in the listed status displays:

""	executing object (if omitted).
*	all objects with open connections.
name	all objects with the given name and an open connection.
node	name: this object, a temporary connection is opened if not currently connected.

NAME=n Name of the component to be displayed, all will be shown if omitted.

FUNCTION This command displays the contents of one or all CCB (component control blocks). If no component name has been specified of if a full wildcard is entered, all CCB's are displayed.

Caller N$GNUCM

EXAMPLE $ SH GNA COMP

CALLING @CALL N$GNUSC_RP(S_RCV,P_INPUT,L_LGTI);

PURPOSE Show status of GNA components.

ARGUMENTS

S_RCV Special mode RPC receive descriptor (SN$RSRCV).

P_INPUT Input argument structure, here component name.

L_LGTI Number of bytes in input argument structure.

Return type BIN FIXED(31)

Status codes	XNET_NOINIT XNET_ILLWLDCHR XNET_BADSTRUCSIZE
Initialize	-
Include name	GOOINC(N$GNUSC)

Description

FUNCTION	This procedure is just a stub to interface the command dispatcher and N$GNURP, which performs a local or remote call of N$GNUSC_RP.
File name	N$GNUSC.PPL
Dataset	-
Version	2.01
Author	Walter F.J. Mueller
Last Update	1-MAY-1987

$ 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

$ SHOW GNA LINKS

$ SHOW GNA LINKS LINK=link NAME=name

PURPOSE Show status of GNA links.

PARAMETERS

LINK=l Specifies the object(s) whose status will be shown. The following input formats are supported and result in the listed status displays:

""	executing object (if omitted).
*	all objects with open connections.
name	all objects with the given name and an open connection.
node	name: this object, a temporary connection is opened if not currently connected.

NAME=n Name of the link to be displayed.

FUNCTION This command displays all or one LCB (Link control block).

Caller N$GNUCM

EXAMPLE $ SH GNA LINK

CALLING @CALL N$GNUSL_RP(S_RCV,P_INPUT,L_LGTI);

PURPOSE Show status of GNA links.

ARGUMENTS

S_RCV Special mode RPC receive descriptor (SN$RSRCV).

P_INPUT Input argument structure, here component name.

L_LGTI Number of bytes in input argument structure.

Return type BIN FIXED(31)

Status codes XNET_NOINIT
XNET_ILLWLDCHR
XNET_BADSTRUCSIZE

Initialize	-
Include name	GOOINC(N$GNUSL)

Description

FUNCTION	This procedure is just a stub to interface the command dispatcher and N$GNURP, which performs a local or remote call of N$GNUSL_RP.
File name	N$GNUSL.PPL
Dataset	-
Version	2.01
Author	Walter F.J. Mueller
Last Update	1-MAY-1987

$ SHOW GNA MCBS

$ SHOW GNA MCBS LINK=link
/PAQ/LCB/ALL /BRIEF/FULL

PURPOSE Show status of MCB's.

PARAMETERS

LINK=l Specifies the object(s) whose status will be shown. The following input formats are supported and result in the listed status displays:

""	executing object (if omitted).
*	all objects with open connections.
name	all objects with the given name and an open connection.
node	name: this object, a temporary connection is opened if not currently connected.

/PAQ Will show all MCB's in the buffer pool pre allocation queues.

/LCB Will show all MCB's queued to LCB's, that are MCBs currently involved in input, output or outstanding transactions.

/ALL Tries to show 'all' MCB, including those which are currently in NO queue. This can be very useful to diagnose 'lost buffers'.
NOTE however some implementation restrictions:
Only the first 128 MCB's with a size below 64 kbyte will be shown. Some nonsense may be listed for MCBs which have been physically freed and reused.

/BRIEF Shows a one line info per MCB. Listed are:

MCB address (in hexadecimal)
Link index (to compare with $ SHO GNA LINK)
Queue status
Allocation size and data size
Message type and subtype
The first 20 data bytes in ASCII

/FULL	Shows a more comprehensive info. (NOT YET DONE).
FUNCTION	This command displays all or some MCB's.
Caller	N$GNUCM
EXAMPLE	$ SH GNA MCB
CALLING	@CALL N$GNUSM_RP(S_RCV,P_INPUT,L_LGTI);
PURPOSE	Show status of MCB's.
ARGUMENTS	
S_RCV	Special mode RPC receive descriptor (SN$RSRCV).
P_INPUT	Input argument structure, here component name.
L_LGTI	Number of bytes in input argument structure.
Return type	BIN FIXED(31)
Status codes	XNET_NOINIT XNET_ILLWLDCHR XNET_BADSTRUCSIZE
Initialize	-
Include name	GOOINC(N$GNUSM)

Description

FUNCTION	This procedure is just a stub to interface the command dispatcher and N$GNURP, which performs a local or remote call of N$GNUSM_RP.
File name	N$GNUSM.PPL
Dataset	-
Version	2.01
Author	Walter F.J. Mueller
Last Update	17-JAN-1989

$ SHOW GNA PROCESS

$ SHOW GNA PROCESS LINK=link

PURPOSE Show process status of GNA object.

PARAMETERS

LINK=l Specifies the object(s) whose status will be shown. The following input formats are supported and result in the listed status displays:

""	executing object (if omitted).
*	all objects with open connections.
name	all objects with the given name and an open connection.
node	name: this object, a temporary connection is opened if not currently connected.

FUNCTION This command displays the process status of a GNA object, mainly quota's, memory and workingset usage,,,,,,,,,,,,, priviledges and access rights. It focuses on items which might influence the execution of networking functions.

Caller N$GNUCM

EXAMPLE $ SH GNA PROCESS

CALLING @CALL N$GNUSP_RP(S_RCV);

PURPOSE Show the process status of a GNA object.

ARGUMENTS

S_RCV Special mode RPC receive descriptor (SN$RSRCV).

Return type BIN FIXED(31)

Status codes XNET_NOINIT

Initialize -

Include name GOOINC(N$GNUSP)

Description

FUNCTION	This procedure is just a stub to interface the command dispatcher and N$GNURP, which performs a local or remote call of N$GNUSP_RP.
File name	N$GNUSP.PPL
Dataset	-
Version	2.01
Author	Walter F.J. Mueller
Last Update	1-MAY-1987

$ SHOW GNA RPC

$ SHOW GNA RPC LINK=link NAME=name

PURPOSE Show status of GNA RPC handlers.

PARAMETERS

LINK=l Specifies the object(s) whose status will be shown. The following input formats are supported and result in the listed status displays:

""	executing object (if omitted).
*	all objects with open connections.
name	all objects with the given name and an open connection.
node	name: this object, a temporary connection is opened if not currently connected.

NAME=n Name of the component to be displayed.

FUNCTION This command displays the RPC call statistic of message handlers declared as RPC dispatchers in all or one component.

Caller N$GNUCM

EXAMPLE $ SH GNA RPC

CALLING @CALL N$GNUSR_RP(S_RCV,P_INPUT,L_LGTI);

PURPOSE Show status of GNA RPC handlers.

ARGUMENTS

S_RCV Special mode RPC receive descriptor (SN$RSRCV).

P_INPUT Input argument structure, here component name.

L_LGTI Number of bytes in input argument structure.

Return type BIN FIXED(31)

Status codes	XNET_NOINIT XNET_ILLWLDCHR XNET_BADSTRUCSIZE
Initialize	-
Include name	GOOINC(N$GNUSR)

Description

FUNCTION	This procedure is just a stub to interface the command dispatcher and N$GNURP, which performs a local or remote call of N$GNUSR_RP.
File name	N$GNUSR.PPL
Dataset	-
Version	2.01
Author	Walter F.J. Mueller
Last Update	1-MAY-1987

$ SHOW GNA STATUS

$ SHOW GNA STATUS LINK=link

PURPOSE Show global GNA status.

PARAMETERS

LINK=l Specifies the object(s) whose status will be shown. The following input formats are supported and result in the listed status displays:

""	executing object (if omitted).
*	all objects with open connections.
name	all objects with the given name and an open connection.
node	name: this object, a temporary connection is opened if not currently connected.

FUNCTION This command displays the most important status information of the GNA subsystem. The following items are shown:

* Process/Object Identity
The GNA network object name, the process name and VMS process ID, the running image and the user name are listed.

* Buffer System Status
The number of free buffers and the quota as well as the number of allocates and frees is listed for all pre allocation queues, the number of allocates and frees for large (>64k) buffers and the number of LIB$ calls to get/free virtual memory is shown too. A summary line lists the number and size of all buffers allocated and of all buffers in use and the 'thruput', which is the total size of all buffers allocated so far.

* Component Status
Lists all declared components, the total and current number of links accepted and requested by each component and the number of bytes (on the physical layer) read and written on links communicating with each component.

*	Link Status Lists the currently open links with local component, link partner address, link direction and number of bytes read and written so far.
Caller	N$GNUCM
EXAMPLE	$ SH GNA STATUS
CALLING	@CALL N$GNUSS_RP(S_RCV);
PURPOSE	Show global GNA status.
ARGUMENTS	
S_RCV	Special mode RPC receive descriptor (SN$RSRCV).
Return type	BIN FIXED(31)
Status codes	XNET_NOINIT
Initialize	-
Include name	GOOINC(N$GNUSS)

Description

FUNCTION	This procedure is just a stub to interface the command dispatcher and N$GNURP, which performs a local or remote call of N$GNUSS_RP.
File name	N$GNUSS.PPL
Dataset	-
Version	2.01
Author	Walter F.J. Mueller
Last Update	23-JAN-1989

$ SHOW KEY

$ SHOW KEY KEY=k STATE=k
/FULL/BRIEF /DIRECTORY

PURPOSE	List key definitions.
PARAMETERS	
KEY=k	Name of key to be listed.
STATE=s	Name of state to be listed. All key definitions are shown if both KEY and STATE are omitted. Otherwise the definitions for a single state and/or key are shown.
/BRIEF	Only the equivalence string is listed. This is the default.
/FULL	All key attributes are shown.
/DIRECTORY	A short directory of all key definitions is shown.
FUNCTION	This command lists the key definitions which have been made with a $DEFINE KEY command.
EXAMPLE	$ SHOW KEY STATE=DEFAULT
Action rout.	C$UCSHK
Author	Walter F.J. Mueller

Remarks

File name	C$UCSHK.PPL
Dataset	-
REMARKS	-

Description

CALLING @CALL C$UCSHK(C_KEY,C_STATE,C_MODE,C_DIRECTORY);

ARGUMENTS

C_KEY CHAR(*) VAR [INPUT]
Name of key to be listed.

C_STATE CHAR(*) VAR [INPUT]
Name of state to be listed.

C_MODE CHAR(*) VAR [INPUT]
List verbosity qualifier set, may be:
/FULL/BRIEF.

C_DIRECTORY CHAR(*) VAR [INPUT]
/DIRECTORY qualifier, specifies that only a state directory is to be shown.

FUNCTION This procedure lists the key definitions. A full discussion is in the command description.

REMARKS -

EXAMPLE @CALL C$UCSHK('','DEFAULT','/FULL','');

$ SHOW MEMORY

$ SHOW MEMORY

PURPOSE Show memory usage.

PARAMETERS

EXAMPLE $ SHOW MEM

Action rout. C$UCSHM

Author H.Essel

Remarks

File name C$UCSHM.PPL

created by C$UTLCM.PPL +Command

Module Command : $ SHOW PROCESS PROCESS=p
/BRIEF

PURPOSE Show process information.

PARAMETERS

PROCESS Name or PID of the process whose attributes are to be shown. Either a valid 8 digit process ID or a process name may be given. If omitted, the current process is taken.

/BRIEF Only the image name, process name, PID, user and the list of owner processes are displayed.

EXAMPLE $ SHOW PROCESS /BRIEF

Action rout. C$UCSHP

Author Walter F.J. Mueller

DESCRIPTION

CALLING @CALL C$UCSHP(C_PROC);

ARGUMENTS

C_PROC CHAR(*) VAR [INPUT]
Name or ident of the process.
See description of U$CVPID for further description of the process name syntax. If a zero length string is specified, the own process identity is shown.

FUNCTION -

REMARKS -

EXAMPLE @CALL C$UCSHP(");

$ SHOW TIMER

$ SHOW TIMER

PURPOSE Show statistics of run time library functions.

PARAMETERS -

Caller -

EXAMPLE $SH TIMER

DESCRIPTION

CALLING @CALL C$UCSHL();

ARGUMENTS

FUNCTION The LIB$SHOW_VM and LIB$SHOW_TIMER services are used show statistic information.

REMARKS -

EXAMPLE @CALL C$UCSHL();

ALLOCATE DEVICE

ALLOCATE DEVICE name type xsize ysize
/[NO]MAIN

PURPOSE Allocate a graphical device

PARAMETERS

name Logical device name
use TT for current terminal.
default is DECW$DISPLAY

type Device type. Allowed devices:

MG600	Monterey MG600 and MG620
PECAD	Pecad terminal
TEK4014	Tektronix 4014
TEK41xx	Tektronix 4107, 4109, 4111, 4115
VWS,MV,GPX	MICRO-VAXII graphic Terminal with VWS
DECWINDOW,MOTIF	DEC window / motif
VT240	graphic VT240 and REGIS terminals
VT340	graphic VT340 and REGIS terminals
LN03	LN03-PLUS laser printer file (.LN3)
SIXEL	sixel output file (.SIX)
POST	Postscript file (.PS)
COLOR	Color Postscript file (.PS)
LJ250	Inkjet file (.LJ250)
HP7550A4	HP7550 pen plotter, DINA4 sheets (.HP4)
HP7550A3	HP7550 pen plotter, DINA3 sheets (.HP3)
METAOUT	Metafile output (.MET) MOTIF

xsize Window x-size in Meter (VWS)

	DECwindow	Size in units of screen. i.e. 0.9
ysize	Window y-size in Meter (VWS)	
	DECwindow	Size in units of screen. i.e. 0.9
	For DECwindows/motif the size can be changed using the cursor!	
/[NO] MAIN	Allocate the specified device as the GOOSY-main device.	
Caller	MDISP,MGOODISP,D$DSPCM	
Author	W. Spreng	

Example

1.) ALLOCATE DEVICE txnn
txnn is allocated as MONT600
2.) ALLOCATE DEVICE txnn TEK4014/MAIN
txnn is allocated as Tektronix 4014 and as a GOOSY-main device.
3.) ALLOCATE DEVICE plotter ln03
A plotfile PLOTTER.LN3 will be generated which could be printed on any LN03-Laser printer by the GOOSY command PLOT PICTURE.
4.) ALLOCATE DEVICE scatter.met metaout
generates the metafile SCATTER.MET
5.) ALLOCATE DEVICE goosy VWS
On a MICRO-VAX II graphic Terminal under VWS a Window with the name "goosy" is created and the graphical output is sent to that window
6.) DECwindow:

Create a virtual display by (/NODE is optional):
$ SET DISPLAY/CREATE/NODE=node::0.0
$ SHO DISPLAY
The SHOW command outputs the name of the virtual display, i.e. WSA8:
GOOSY> ALLOCATE DEVICE WSA8: DECW .9 .9

Remarks

File name	D$ALLOC.PPL
Created by	GOO$DISP:D$DSPCM.PPL

Description

CALLING STS=D$ALLOC(CV_name,CV_type,r_xsize,R_ysize,CV_main)

COMMAND ALLOCATE DEVICE name type xsize ysize
/[NO]MAIN

NAME

Routine par. Input CHAR(*) VAR

Command arg. String required, def=DECW$DISPLAY

Logical or physical device name. For a graphical Terminal it is the VMS-Terminal address; e.g.:

TXA6	for a terminal connected directly to one VAX.
LTA999	for a terminal connected to a LAT server.
TT	for the terminal of the session.

For spooled devices (all supported plotters) you can specify a file name with or without a file type.

For Metafiles a file name or a logical name which refers to a file is required.

For MICRO-VAX II and GPX devices under VWS the device name is arbitrary, it is displayed at the generated graphic window.

For DECwindow a virtual display must be created by DCL command SET DISP/CREATE/NODE=node::0.0 where /NODE= is optional (remote display). The DCL command SHO DISP then outputs the virtual display device name WSAnn:

Normally one can use DECW$DISPLAY witch is the default.

TYPE

Routine par. Input CHAR(*) VAR

Command arg. String required default=MOTIF

The type of the device to be allocated has to be specified with this parameter. The following device types are supported:

MG600	Monterey MG600 and MG620
PECAD	Pecad terminal

TEK4014	Tektronix 4014
TEK41xx	Tektronix 4107, 4109, 4111, 4115
VWS,MV,GPX	MICRO-VAXII graphic Terminal with VWS
DECWINDOW,MOTIF	DEC window
LN03	LN03-PLUS laser printer
SIXEL	sixel output
POST	Postscript output format
COLOR	Postscript output format
HP7550A4	HP7550 pen plotter, DINA4 sheets
HP7550A3	HP7550 pen plotter, DINA3 sheets
LJ250	Inkjet output format
VT240	graphic VT240 and REGIS terminals
VT340	graphic VT340 and REGIS terminals
METAOUT	Metafile output

For all supported plotters the plot-file 'name' is generated which could be plotted out later. If no file-type is specified the default plotfile extensions are:

LN03 generates a plotfile name.LN3
SIXEL generates a plotfile name.SIX
POST generates a plotfile name.PS
COLOR generates a plotfile name.PS
LJ250 generates a plotfile name.LJ250
HP7550A3 generates a plotfile name.HP3
HP7550A4 generates a plotfile name.HP4

You can plot them with the PLOT PLOTFILE command.

Metafiles could be sent to all supported plotters with the PLOT METAFILE command or they could be displayed later with DISPLAY METAFILE.

But in any case do not forget to give the DEALLOCATE DEVICE command first to close the metafile or plotfile properly!

XSIZE

Routine par.	Input BIN FLOAT(24)
Command arg.	FLOAT, default = 0.0

For window orietated devices, e.g. VAXstations, this parameter specifies the window size in X-direction. The size has to be specified in METERS, except for DECwindow, where it must be specified in units of the screen, i.e. 0.9 is 90% of screen. For DECwindows/motif the size can be changed using the cursor. The window origin is always at the lower left screen edge.

YSIZE

Routine par. Input BIN FLOAT(24)

Command arg. FLOAT, default = 0.0

For window orietated devices, e.g. VAXstations, this parameter specifies the window size in Y-direction. The size has to be specified in METERS, except for DECwindow, where it must be specified in units of the screen, i.e. 0.9 is 90% of screen. For DECwindows/motif the size can be changed using the cursor. The window origin is always at the lower left screen edge.

MAIN

Routine par. Input CHAR(*) VAR

Command arg. Set, default = /NOMAIN

Allocate the specified device as the GOOSY-main device. All displays are then adjusted to make optimal use of the hardware facilities of the specific graphical device. Futhermore the GOOSY-main device is used for all graphical inputs. If not specified the first device which is allocated is used as the GOOSY-main device.

For a detail description of the GOOSY-main device see the GOOSY Display manual.

Function

The specified device is allocated as a workstation in a GKS-session.

An entry in the device description table is created and all hardware facilities of the device are stored in this table. If /MAIN is specified the GOOSY-main device type is defined by the specified device. All pictures produced in the display process are then adjusted to make optimal use of the facilities of the main device. Futhermore this device is additionally used for all graphical inputs.

In principle each device can be allocated as a GOOSY main device, but do not wonder if you try this with a plotter and no graphical inputs are possible!

If /MAIN is not specified, the procedure checks if a main device is defined. If not the type of the first allocated output device is used as the default main device.

If a METAFILE should be allocated the device name has to be a file name or a logical name which references to a file.

ATTACH ANALYSIS

ATTACH ANALYSIS

PURPOSE	Reinitialize analysis after DETACH ANALYSIS
NOTE	Dynamic lists must be attached again.

Description

FUNCTION	This command calls $IBUFFER to reinitialyze the analysis. The data base is detached and attached.
File name	I$ANACM.PPL
Action rout.	I$ANACM_ATT
Version	1.01
Author	H.G.Essel
Last Update	12-APR-1985

ATTACH BASE

ATTACH BASE base node
/READ

PURPOSE	Attach data base.
PARAMETERS	
base	Data Base name required common default
node	optional node optional
/READ	Map readonly
Caller	M$DMCMD
Author	H.G.Essel
File name	M$AATDB.PPL
EXAMPLE	ATT BASE mybase

Remarks

REMARKS	-

Description

CALLING	STS=M$AATDB(CV_BASE,CV_NODE,I_READ)
ARGUMENTS	
CV_BASE	Data Base name CHAR(*) VAR
CV_NODE	optional node CHAR(*) VAR

I_READ	Map readonly BIN FIXED(15)
FUNCTION	Map total bata base.
REMARKS	Module is an action routine.
EXAMPLE	-

ATTACH DYNAMIC LIST

ATTACH DYNAMIC LIST dyn_list dyn_dir base node
/FAST

PURPOSE Attach dynamic list

PARAMETERS

dyn_list Dynamic list name specification
required common default

dyn_dir Default directory
common default:'$DYNAMIC'

base Default data base name
common default:'DB'

node Default node name
common default:'E'

/FAST No execution or freeze bits are checked. Execution bit is set, however. No counters are incremented. Only the following data types are supported:
BIN FLOAT(24) for window, spectra objects
BIT(32) ALIGNED for patterns.
Spectrum type must be R, increment is always 1.
Master entries, scatter plots and user functions are not affected.

EXAMPLE -

Caller M$DLCMD

Author H.G. Essel

File name M$AATDL.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$AATDL(CV_DYN_LIST,CV_DYN_DIR,
CV_BASE,CV_NODE,I_FAST)

ARGUMENTS

CV_DYN_LIST Dynamic list name specification

CV_DYN_DIR Default Directory

CV_BASE Default Data Base name

CV_NODE Default node name

I_FAST Fast execution
IF 1, no execution or freeze bits are checked.
No counters are incremented. Only the following data types are supported:
BIN FLOAT(24) for window, spectra objects
BIT(32) ALIGNED for patterns.
Spectrum type must be R, increment is always 1.

FUNCTION Attach dynamic list

REMARKS Module is an action routine.

EXAMPLE -

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

CALCULATE SPECTRUM

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

PURPOSE Perform spectrum arithmetic operations.

PARAMETERS

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

operand_1 First spectrum operand.

operand Aritmetic operation which should be performed.

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

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

spec_dir Default spectrum directory.

base Default data base name.

node Default node.

/CONSTANT If specified the OPERAND_2 is interpreted as a costant value.

/[NO] KEEP Keep context of all data bases.

Caller MDBM,MGOODBM

Author W. Spreng

Example

1.) CALCULATE SPECTRUM [test]a b + [$spectrum]c
The spectra b and [$spectrum]c are added
and the result is stored in [test]a.
2.) CALCULATE SPECTRUM a b * 2.75 /constant
The contents of spectrum b is increased by the
factor 2.75 and the result is stored in spectrum
a.
3.) CALCULATE SPECTRUM a b + c 2.0
The performed operation for each bin is:
a = b + c*2.0

Remarks

REMARKS Up to now only bins are handled, the spectrum limits are ignored! Therefore be carful to mix spectra with different limits and binsizes!

File name GOO$DE:E$CACSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$CACSP(CV_RESULT,CV_OPERAND_1,
            CV_ACTION,CV_OPERAND_2,R_FACTOR,
            CV_spec_dir,CV_base,CV_node,I_constant,
            I_KEEP)
```

COMMAND

```
CALCULATE SPECTRUM
    result operand_1 action operand_2 factor
    spec_dir base node
    /CONSTANT
    /[NO]KEEP
  Argument and parameter description
```

RESULT

Routine arg. Input CHAR(*) VAR

Command par. String

Name of spectrum which contains the result. If the spectrum does not exist, it will be created with the attributes of OPERAND_1

OPERAND_1

Routine arg. Input CHAR(*) VAR

Command par. String

First spectrum operand.

ACTION

Routine arg. Input CHAR(*) VAR

Command par. String

Aritmetic operation which should be performed. Should be one of the following symbols:

+ addition
- subtraction
* multiplication
: division

OPERAND_2

Routine arg. Input CHAR(*) VAR

Command par. String

Second operand. It could be a spectrum or a constant value; in that case /CONSTANT have to be set.

FACTOR

Routine arg. Input CHAR(*) VAR

Command par. String

Factor to scale Operand_2. This factor is used to increase the channel contents of operand 2. It is ignored if /CONSTANT is specified.

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replacable default="$SPECTRUM"

Default spectrum directory.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replacable default="DB"

Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replacable default="*"

Default node.

/CONSTANT

Routine arg. Input BIN FIXED(15), valid inputs are 0 and 1

Command par. Switch

Flag to mark OPERAND_2 as a constant. If set the value specified in OPERAND_2 is converted to the same data type as OPERAND_1.

/KEEP

Routine arg. Input BIN FIXED(15), valid inputs are 0 and 1

Command par. Switch

If this flag is set the context of all Data bases will be kept.

Function

For each bin the specified aritmetic operation is performed. The result is stored in the specified spectrum.

The binsize of "operand_1" and "operand_2" must be equal. The binsize of the "result" spectrum must be an integer mulitple of the operand spectra. The limits of the spectra could be different; they are ignored. The datatyps of the operands could be different they will be correctly converted. Only conversions to data types of higher precision are allowed:

Result spectrum of type R
Operand 1 spectrum type L

Operand 2 spectrum type R
===> is allowed!
Result spectrum of type L
Operand 1 spectrum type L
Operand 2 spectrum type R
===> is not allowed!

Operand_2 could be interpreted as a constant value, in that case the flag i_const has to be set.

If the "result-spectrum" does not exist, it will be created with the binsize, the spectrum limits of "operand_1". The data type is the type of the operand with the highest precision.

If the switch /CONSTANT is set OPERAND_2 is interpreted as a constant. The specified Value is converted to the same data Type as the specified OPERAND_1.

CALIBRATE SPECTRUM

CALIBRATE SPECTRUM spectrum calibration spec_dir cal_dir base node

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

Example

CALIBRATE SPECTRUM spec cal
The calibration "CAL" is connected to the spectrum "SPEC".

Description

CALLING

STS=E$CALSP(CV_SPECTRUM,CV_CALIBRATION,CV_SPEC_DIR,
CV_CAL_DIR,CV_BASE,CV_NODE)

COMMAND CALIBRATE SPECTRUM spectrum calibration spec_dir cal_dir base node

SPECTRUM

Routine arg. CHAR(*) VAR

Command arg. String required

Name of spectrum which should be connnected with a calibration.

CALIBRATION

Routine arg. CHAR(*) VAR

Command arg. String required

Name of the calibration. The calibration must exist but it is not necessary to set the calibration before connecting it to the spectrum.

SPEC_DIR

Routine arg. CHAR(*) VAR

Command arg. String global replaceable default=$SPECTRUM

Default Directory name for spectra.

CAL_DIR

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=$CALIB

Default Directory for calibration Data Elements.

BASE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=*

Node name for Data Base section file.

Function

FUNCTION

The specified spectrum is connected to a calibration Data Element. It is allowed to calibrate several spectra with the same calibration, but it is impossible to connect the spectra with several calibrations.

An additional Data Element is queued to the spectrum header which stores the Data Element indices for the calibration. Additionaly a bit will be set in the spectrum header to mark the spectrum calibrated. A link between the spectrum and the calibration Data Element is established to guaranty that a calibration could not be deleted if it is connected to at least one spectrum. To delete a calibration all spectra connected to it have to be uncalibrated by the command UNCALIBRATE SPECTRUM.

If the calibration is set, it is possible to display the spectrum in calibrated units.

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');

CLEAR CAMAC SPECTRUM

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

PURPOSE clear one (or all) spectrum

PARAMETERS

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

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

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

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

/LOG Switch default: ”
Output list of cleared spectra

/CAMAC Switch default: ”
Clear spectrum in CAMAC.

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

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

Caller mdbm

Author H.G.Essel

Example

```
CL SP A(1,2) /SPEC
CL SP A(*) /CAM
CL SP A* /CAM/SPEC
CL SP A (clear first spectrum only)
```

Remarks

File name E$ACCSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING
```
STS=E$ACCSP(CV_NAME,CV_SPEC_DIR,CV_BASE,CV_NODE,
            I_LOG,I_CAMAC,I_SPECTRUM,I_KEEP_MAP)
```

COMMAND
```
CLEAR CAMAC SPECTRUM name spec_dir base node
            /CAMAC
            /SPECTRUM
            /LOG
            /[NO]KEEP_MAP
```
Argument description

NAME

Routine arg. Input CHAR(*) VAR

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

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Name of Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Name of Node

LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Output list of cleared spectra

CAMAC

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
CLear spectrum in MR2000.

SPECTRUM

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Clear spectrum in data base.

KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”
Inhibit the unmap (detach) of the Data Base

Function

The data part of the spectrum is set to zero
and the range in the MR2000 is cleared.

CLEAR CONDITION COUNTER

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

PURPOSE clear condition counters specified by name

PARAMETERS

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

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

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

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

/LOG Switch default: none
Output list of cleared conditions

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

Caller MDBM

Author K.Winkelmann

Example

```
CLEAR COND COU A* clear all members
CLEAR COND COU A(3:6) clear four members
CLEAR COND COU A(10) clear one member
CLEAR COND CO A(*) clear all members
CLEAR COND COU A clear all members
```

Remarks

File name GOO$DE:E$CLCO.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$CLCO(CV_NAME,CV_DIR,CV_BASE,CV_NODE,
                I_LOG,I_KEEP_MAP,B_MASK)
```

COMMAND

```
CLEAR CONDITION COUNTER name cond_dir base node
        /[NO]KEEP_MAP
        /LOG
Argument / Parameter description
```

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”

Name expression of condition. Wildcards are accepted in the form:

* x* *x *x* x*y

Name array are supported. Index may be wildacrd. This is assumed, if name is wildcarded. Arrays without index are cleared totally. For one dimensional arrays a range may be specified like x(3:5). No wildcard is allowed in this case. Single members of one and twodimensional arrays may be cleared by specifying the index: X(7) or Y(4,5).

DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'

Type	Input CHAR(*) VAR Name of condition directory

BASE

Routine arg.	Input CHAR(*) VAR
Command par.	string global replace default: 'DB'
Type	Input CHAR(*) VAR Name of Data base

NODE

Routine arg.	Input CHAR(*) VAR
Command par.	required string global replace default: 'E'
Type	Input CHAR(*) VAR Name of Node

LOG

Routine arg.	Input BIN FIXED (15) valid values 0 or 1
Command par.	switch default: "
Type	Input BIN FIXED(15) Output list of cleared conditions

KEEP_MAP

Routine arg.	Input BIN FIXED (15) valid values 0 or 1
Command par.	switch default: "
Type	Input BIN FIXED(15) Inhibit unmap of data Base

Function

The condition counters are cleared

Remarks

The Module is an action routine.

Example

```
STS=E$CLCO('C1','$CONDITION','DB','E',1,1)
```

CLEAR DEVICE

CLEAR DEVICE

PURPOSE	Clear all active workstations
Caller	MDISP,MGOODISP
Author	W. Spreng

Description

CALLING	STS=D$CLRWK;
COMMAND	CLEAR DEVICE

Function

This procedure clears all active workstations defined during a GKS-session.

All the following actions are executed:

- the display surface is cleared
- all segments are deleted from the workstation state list
- all actions pending on the workstation are updated, e.g. changes of the workstation window, workstation viewport etc.

CLEAR ELEMENT

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

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

PARAMETERS

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

dir string global replace default: 'DATA'
Default directory

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

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

/LOG switch default: ”
Displays cleared Data Element.

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

Caller M$DMCMD

Author Th. Kroll

EXAMPLE

CLEAR ELEMENT DB:[DATA]ADAM /LOG Set the Data Element ADAM to Zero.

Remarks

File name M$ACLDE.PPL

Ceated by GOO$DM:M$DMCMD.PPL

Description

CALLING

```
STS=M$ACLDE(CV_NAME,CV_DIR,CV_BASE,CV_NODE,
            I_LOG,I_KEEP_MAP)
```

COMMAND

```
CLEAR ELEMENT name dir base node
   /LOG
   /[NO]KEEP_MAP
Argument /Parameter description
```

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of Dataelement to be cleared like Node::base:[dir]name(i)->type(i). Node, base and directory are defaulted from the according parameters NODE, BASE and DIR.

DIR

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: 'DATA'
Default directory of the dataelement.

BASE

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: 'DB'
Default data base name

NODE

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: 'E'
Default node name

LOG

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: ”
/LOG	Display cleared Data Elements.

KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/KEEP_MAP'
/NOKEEP_MAP	The data base is detached after the command.
/KEEP_MAP	The data base remains attached. This is recommended.

FUNCTION

Clear data element. Reset the dataelement to zero.

CLEAR PICTURE

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

PURPOSE Clear spectra defined in a picture data element

PARAMETERS

picture Picture data element name

frame Frame specification.

pic_dir Default directory name

base Default data base name.

node Default node name

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

/[NO] LOG List names of all pictures found and of all spectra which have been cleared.

Example

1.) CLEAR PICTURE a 3
Contents of spectrum in frame 3 of picture "A" will be cleared.
2.) CLEAR PICTURE a 1:3
Spectra in frames 1 to 3 are cleared
3.) CLEAR PICTURE a *
All spectra in picture a are cleared.
4.) CLEAR PICTURE * *
All spectra in all pictures in the default picture directory are cleared.
5.) CLEAR PICTURE [*]* *
All spectra in all pictures found in any directory are cleared.

Remarks

Created by E$DECMD.PPL

File name D$CLRPI.PPL

Description

CALLING STS=D$CLRPI(CV_PICTURE,CV_FRAME,CV_PIC_DIR,
CV_BASE,CV_NODE,I_keep_map,i_log)

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

PICTURE

Routine arg. CHAR(*) VAR

Command par. String required

Name of Picture Data Element. Wildcards for directory and picture data element name are allowed. In a wildcard loop the default picture GOOSY_SPECTRUM used by DISPLAY SPECTRUM will be excluded.

FRAME

Routine arg. CHAR(*) VAR

Command par. String required default=*

Frame specification. The spectra in the specified frames will be deleted. Valid inputs are:

n - single number
n:m - range of frames
* - all frames

PIC_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default = $PICTURE

Default Picture Directory.

BASE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default = DB
Default Data Base name

NODE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default = $PICTURE
Default node name

/KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch negatable default=/KEEP_MAP
Switch to prevent the dettachment of the Data Base

/NOKEEP_MAP	Dettach Data Base.
/KEEP_MAP	Keep the whole Data Base mapping context

LOG

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch negatgable default=/NOLOG
Controls if the specification of all spectra and pictures are listed.
If you use the /LOG switch the following information is displayed:
(1) the name of the picture which will be handled.
(2) all spectra in that picture which are cleared.

Function

Clear the spectra in the frames of the specified pictures. For the picture names and picture directories wildcards are supported.

The frame specification could be a single frame a range of frames (n:m) or "*" for all frames. Only spectrum frames are handled, if a scatter frame is specified an error message is produced, but no interrupt occurs. The default picture GOOSY_SPECTRUM used by command DISPLAY SPECTRUM is excluded from wildcard loops.

CLEAR SPECTRUM

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

PURPOSE clear one (or all) spectrum

PARAMETERS

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

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

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

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

/LOG Switch default: ”
Output list of cleared spectra

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

Caller mdbm

Author K.Winkelmann

Example

CL SP A(3,9) clear one spectrum
CL SP A(3) clear one spectrum
CL SP A(3:9) clear seven spectra
CL SP A(*) clear all members
CL SP A* clear all members
CL SP A clear all members

Remarks

File name E$CLSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

STS=E$CLSP(CV_NAME,CV_SPEC_DIR,CV_BASE,CV_NODE,
I_LOG,I_KEEP_MAP)

COMMAND

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

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of Spectrum (wildcard) to be cleared
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.
Arrays without index are cleared totally.
For one dimensional arrays a range may be
specified like x(3:5). No wildcard is allowed in
this case. Single members of one and twodimensional
arrays may be cleared by specifying the index:
X(7) or Y(4,5).

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Name of Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Name of Node

LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Output list of cleared spectra

KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Inhibit the unmap (detach) of the Data Base

Function

The data part of the spectrum is set to zero

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

COMPRESS BASE

COMPRESS BASE base file
/DISMOUNT

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

PARAMETERS

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

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

/DISMOUNT switch default:
Dismount data base after copy.

Caller MDBCOPY

Author H.G.Essel

Example

```
MDBCOPY COMP BASE db db.cmp
```

Remarks

File name GOO$DM:M$ACMPB.PPL

Created by GOO$DM:M$DMCMD.PPL

Description

CALLING STS=M$ACMPB(CV_base,CV_file,I_dismount)

COMMAND COMPRESS BASE base file
/DISMOUNT
Argument /parameter description:

BASE

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”
Name of the data base to be compressed and copied.

FILE

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”
File name of output file. Default file type is .CSEC.

/DISMOUNT

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default:
The source data base is dismounted after the copy.

Function

Compress and copy a bata base. The compressed base must be decompressed before it can be used. Note that the output file will be written as a global section. Therefore an existing output file would be overwritten! Therefore in this case an error message is given and no copy is performed. The data base must be mounted and will be dismounted when /DISMOUNT is given.

CONVERT BASE

CONVERT BASE base file size

PURPOSE Convert data base.

PARAMETERS

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

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

Caller MDBCOPY,MUTIL

Author H.G.Essel, B.Dechant

Example

```
$ MDBCOPY CONVERT BASE db db1.sec 16000
$ MUTIL CONVERT BASE db db1.sec 16000
```

Remarks

File name GOO$DM:M$ACOAL.PPL

Created by GOO$DM:M$COBCM.PPL

Description

CALLING STS=M$ACOAL(CV_base,CV_file,l_sizedb)

COMMAND CONVERT BASE base file size Argument /parameter description:

BASE

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”
Name of the data base to be converted

FILE

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”
File name of output file. Default file type is .SEC.

FILE

Routine arg. Input BIN FIXED(31)

Command par. integer
New size of output file in pagelets.

Function

Convert a data base. Note that the output file will be written as a global section. Therefore an existing output file would be overwritten! Therefore in this case an error message is given and no convert is performed. The source data base must be mounted. The new data base is not mounted.

COPY BASE

COPY BASE base file size area
/DISMOUNT

PURPOSE Expand and copy data base.

PARAMETERS

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

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

size integer
Optional size of new base in Kbytes.

area integer
Optional new number of entries in area directory.

/DISMOUNT switch default:
Dismount data base after copy.

Caller MDBCOPY,MUTIL

Author H.G.Essel

Example

```
$ MDBCOPY COPY BASE db db1.sec
$ MUTIL COPY BASE db db1.sec
```

Remarks

File name GOO$DM:M$ACODB.PPL

Created by GOO$DM:M$COBCM.PPL

Description

CALLING STS=M$ACODB(CV_base,CV_file,L_size
,L_area,I_dismount)

COMMAND COPY BASE base file size area
/DISMOUNT
Argument /parameter description:

BASE

Routine arg. Input CHAR(*) VAR

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

FILE

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”
File name of output file. Default file type is
.SEC.

AREA

Routine arg. Input BIN FIXED(31)

Command par. integer
Optional new size of area directory (entries). The maximum of this parameter and the old size is taken.

SIZE

Routine arg. Input BIN FIXED(31)

Command par. integer
Optional new base size in Kbytes. The maximum of this parameter and the old size is taken.

/DISMOUNT

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default:
The source data base is dismounted after the copy.

Function

Expand and copy a bata base. Note that the output file will be written as a global section. Therefore an existing output file would be overwritten! Therefore in this case an error message is given and no copy is performed. The source data base must be mounted and will be dismounted when /DISMOUNT is given. The new data base is not mounted.

COPY CONDITION

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

PURPOSE Copy source Condition to destination Condition

PARAMETERS

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

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

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

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

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

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

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

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

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

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

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

[NO] KEEP_MAP Switch default: /KEEP_MAP

Caller MDBM,MGOODBM

Author Th.KROLL

Example

EXAMPLE

```
COPY CONDITION E::DB:[$CONDITION]S1
                    E::NEWDB:[$CONDITION]NEW_S1/NOCONFIRM
```
Creates the condition NEW_S1 on Database NEWDB and copies the data from condition S1 from Database DB to condition NEW_S1.

```
COPY CONDITION E::DB:[$CONDITION]S(1)
                    E::NEWDB:[$CONDITION]NEW_S2/NOCONFIRM
```
Creates the Condition NEW_S2 on Database NEWDB and copies the data from Condition S(1) from Database DB to Condition NEW_S2.

```
COPY CONDITION E::DB:[$CONDITION]S(1:5)
                    E::NEWDB:[$CONDITION]NEW_S(3:7)/NOCO
```
Creates the conditionarray NEW_S(3:7) on Database NEWDB and copies the data from conditionarray S from Database DB to conditionarray NEW_S.

```
COPY CONDITION E::DB:[$CONDITION]S(*)
                    E::NEWDB:[$CONDITION]NEW_S(*)/NOCO
```
Creates the conditionarray NEW_S(*) on Database NEWDB with the same limits as source Conditions and copies the data from Conditionarray S from Database DB to Conditionarray NEW_S.

```
COPY CONDITION E::DB:[$CONDITION]S(*)
```

E::NEWDB:[$CONDITION]NEW_S(*)/REPLACE
Replaces the conditionarray NEW_S(3:7) on Database NEWDB and copies the data from conditionarray S from Database DB to Conditionarray NEW_S.

COPY CONDITION E::DB:[$CONDITION]S(2)
E::NEWDB:[$CONDITION]S_COPY(4)/NOCON
Creates the Condition S_COPY(4) on Database NEWDB and copies the data from Condition S(2) from Database DB to Condition S_COPY(4).

COPY CONDITION E::DB:[$CONDITION]S(2)
E::NEWDB:[$CONDITION]S_COPY(4)/REPLACE
Copies the Condition S(2) from database DB to the existing Condition S_COPY(4) on destination Database NEWDB (a replace is done).

COPY CONDITION E::DB:[$CONDITION]S(1:5)
E::NEWDB:[$CONDITION]S_COPY(1:5)/REPLA
Copies the Condition S(1:5) from database DB to the existing Condition S_COPY(1:5) on destination Database NEWDB (a replace is done).

COPY CONDITION E::DB:[$CONDITION]S(1)
E::NEWDB:[$CONDITION]S_TEST /REPLACE
Copies the Condition S(2) from database DB to the existing Condition S_TEST on destination Database NEWDB (a replace is done).

COPY CONDITION E::DB:[$CONDITION]S_ORIGNAL
E::NEWDB:[$CONDITION]S_COPY(1)/REPLACE
Copies the Condition S_ORIGINAL from database DB to the existing Condition S_COPY(1) on destination Database NEWDB (a replace is done).

COPY CONDITION E::DB:[$CONDITION]S_ORIGNAL
E::NEWDB:[$CONDITION]S_COPY/REPLACE
Copies the Condition S_ORIGINAL from database DB to the existing Condition S_COPY on destination Database NEWDB (a replace is done).

Remarks

File name E$COCO.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$COCO(CV_SRC_CONAME,CV_DST_CONAME,
           CV_SRC_DIR,CV_DST_DIR,
           CV_SRC_BASE,CV_DST_BASE,
           CV_SRC_NODE,CV_DST_NODE,
           I_REPLACE,I_NOCONFIRM,I_LOG,I_KEEP_MAP)
```

COMMAND

```
COPY CONDITION name destname
                cond_dir destcond_dir
                base destbase node destnode
               /REPLACE
               /CONFIRM
               /LOG
               /[NO]KEEP_MAP
```

Argument / Parameter description.

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Source condition name, may be wildcarded.

DESTNAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Destination condition name, may be wildcarded.

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Source Condition directory name, may be wildcarded.

DESTCOND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Destination Condition directory name, may be wildcarded.

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Source condition data base.

DESTBASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$DB'
Destination condition data base.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Source condition node name.

DESTNODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Destination condition node.

/REPLACE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: " If a condition exists already and /REPLACE is not set not replace will be performed.

/CONFIRM

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/CONFIRM' If a new condition should be created and /NOCONFIRM is set no prompt for creation will follow. If a new condition should be created and /NOCONFIRM is NOT set a prompt for creation will follow.

/LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: " If /LOG is set the copy command displays the condition specifications of each condition copied.

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

/KEEP_MAP The data base remains attached. This is recommended.

Function

Copy condition to another or the same Data Base,
but only on the same Node. If a condition doesn't
exists the switch /NOCONFIRM has to be set. If not,
the command will prompt for creation.
Wildcard (*) for condition and directories and
Conditionsarray's are supported.

Remarks

Module is an action routine.

Example

```
STS$VALUE=E$COCO('S(1)','NEW_S1','$CONDITION',
                 '$OTTO','DB','NEWDB','E','E',
                 0,1,1,1)
```

COPY ELEMENT

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

PURPOSE	Copy source Dataelement to destination Dataelement
PARAMETERS	
name	required string global replace default: " Source Dataelement name
destname	required string global replace default: " Destination Dataelement name
dir	required string global replace default: 'DATA' Default source directory
destdir	required string global replace default: 'DATA' Default destination directory
destpool	required string global replace default: 'DATA' Default destination pool
base	required string global replace default: 'DB' Default source database
destdb	required string global replace default: 'DSTDB' Default destination database
node	required string global replace default: 'E' Default source node
destnode	required string global replace default: 'E' Default destination node
I_REPLACE	switch default: " Replace deastination dataelement
I_ALL	switch default: " Replace or copy a complete name array
I_NOCONFIRM	switch default: " Don't ask to confirm creation of a new Dataelement
I_KEEP_MAP	switch default: /KEEP_MAP Inhibit unmap of Data Base

Caller MDBM,MGOODBM

Author Th. Kroll

EXAMPLE

Following types of copy are supported :
Dataelement EMIL and ADAM defined as scalar
Dataelement ILSE and MARIE defined as name arrays
Dataelement OTTO and EVA defined as
queue name arrays
Dataelement KLAUS and WILLI defined as
queued member

COPY ELEM DB:[dir]EMIL DB:[dir]ADAM
The existing DE ADAM will be deleted and a
new DE will be created and then the data will be
copied.

COPY EL DB:[dir]ILSE(n) DB:[dir]ADAM
The existing DE ADAM will be deleted and a
new DE will be created and then the data will be
copied.

COPY EL DB:[dir]ILSE DB:[dir]MARIE /REPLACE/ALL
DE MARIE must exist, the data of MARIE will be
overwritten.

COPY EL DB:[dir]ILSE(n) DB:[dir]MARIE(m)/REPLACE
DE MARIE must exist, member (m)
will be replaced

COPY EL DB:[dir]EMIL DB:[dir]MARIE(m)/REPLACE
DE MARIE must exist, member(m) will be replaced

COPY EL DB:[dir]ILSE(n:m) DB:[dir]MARIE(o:p)/REPLA
DE MARIE must exist, members (o:p) will
be replaced. The limits of the array must match.

COPY EL DB:[dir]OTTO DB:[dir]EVA/REPLACE
DE EVA must exist, Queue header will be repl.

COPY EL DB:[dir]OTTO->type(i) DB:[dir]EVA->type(j)
/REPLACE
DE EVA must exist, queued name array member(j)
will be replaced

COPY EL DB:[dir]OTTO->type DB:[dir]EVA->type
/ALL/REPLACE
DE EVA must exist, complete queued DE wil be
copied.

COPY EL DB:[dir]KLAUS->type DB:[dir]WILLI->type
/NOCONFIRM
If DE WILLI doesn't exist, the DE WILLI will be created and if no QH exist, the QH will be copied from KLAUS .

File name M$ACODE.PPL

Created by GOO$DM:M$DMCMD.PPL

Description

CALLING

```
STS=M$ACODE(CV_SRC_ELEMENT,CV_DST_ELEMENT,
            CV_SRC_DIR,CV_DST_DIR,CV_DST_POOL,
            CV_SRC_DB,CV_DST_DB,
            CV_SRC_NODE,CV_DST_NODE,
            I_REPLACE,I_ALL,I_NOCONFIRM,I_KEEP_MAP)
```

COMMAND

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

rguments / Parameter description.

ELEMENT

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: " Source Dataelement name

DEST_ELEMENT

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: " Destination Dataelement name

DIR

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'DATA' Source Dataelement directory name

DESTDIR

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'DATA' Destination Dataelement directory name

DESTPOOL

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'DATA' Destination Dataelement pool name

BASE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'DB' Source Dataelement database name

DESTBASE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'DSTDB' Destination Dataelement database name

NODE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'E' Source Dataelement node name

DESTNODE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'E' Destination Dataelement node name

/REPLACE

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: ”
/REPLACE	Replace Destination Dataelement

/ALL

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: ”
/I_ALL	Replace or copy complete name array

/NOCONFIRM

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: ”
/NOCONFIRM	Don’t ask to confirm creation of a new

/KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: ’/KEEP_MAP’
/NOKEEP_MAP	Unmap of Data Base
/KEEP_MAP	Inhibit unmap of Data Base

Function

Copy Dataelements to another directory or database, but only on the same Node. If a Dataelement doesn’t exists the switch I_NOCONFIRM has to be set. If not, the command will be aborted. In case of a complete name array copie switch I_ALL and I_REPLACE are to be set.

COPY FILE

COPY FILE file outfile skip buffers

PURPOSE Output GOOSY list mode data file (called in MUTIL).

PARAMETERS

file required string replace
List mode data file.

output string replace
Required output file.

skip integer default=0
Optional number of buffers to skip.

buffers integer default=100000000
Optional number of buffers to output

Caller MUTIL

Author H.G.Essel

Example

```
$ MUTIL COPY FIL X.LMD OUT=X.HEAD
All buffers from X.LMD written into Y.LMD.
$ MUTIL COPY FIL X.LMD Y.LMD 0 10
10 buffers from X.LMD written into Y.LMD.
$ MUTIL TYPE FIL X.LMD Y.LMD 10 1
Write 11th buffer of X.LMD to Y.LMD.
```

Remarks

File name I$FILCM.PPL

Created by I$FILCM.PPL

Description

CALLING STS=I$FIL_C(CV_file,CV_outfile,L_skip,L_buffers)

COMMAND COPY FILE file outfile skip buffers
Arguments/Parameters description

FILE

Routine arg. Input CHAR(*) VAR

Command par. required string replace
File name for input.

OUTFILE

Routine arg. Input CHAR(*) VAR

Command par. string replace
File name for output.

SKIP

Routine arg. Input BIN FIXED(31)

Command par. integer default=0
Optional number of buffers to be skipped.

BUFFERS

Routine arg. Input BIN FIXED(31)

Command par. integer default=100000000
Optional number of buffers to be output.

Function

Read specified input file and output GOOSY file header and data. Only standard GOOSY data formats are supported.

COPY MEMBER

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

PURPOSE Copy Data Element member to another Data Element member

PARAMETERS

member required string global replace default: " Default source member name Node::base:[dir]name(i)->type(i).member

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

dir required string global replace default: 'DATA' Default source Directory

destdir required string global replace default: 'DATA' Default destination Directory

base required string global replace default: 'DB' Default source Data Base

destbase required string global replace default: 'DSTDB' Default destination Data Base

node required string global replace default: 'E' Default source node

destnode required string global replace default: 'E' Default destination node

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

Caller MDBM,MGOODBM

Author Th. KROLL

Example

COP MEM [ADAM]ABEL.PART [DATA]KAIN.PIECE

Remarks

File name M$ACOME.PPL

Created by GOO$DM:M$DMCMD.PPL

Description

CALLING

```
STS=M$ACOME(CV_MEMBER,CV_DST_MEMBER,
            CV_SRC_DIR,CV_DST_DIR,CV_SRC_DB,
            CV_DST_DB,CV_SRC_NODE,CV_DST_NODE,
            I_KEEP_MAP)
```

COMMAND

```
COPY MEMBER element destelement dir destdir
            base destbase node destnode
            /[NO]KEEP_MAP
```

Argument / Parameter description.

MEMBER

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of source Data Element Member
Node::base:[dir]name(i)->type(i)

DESTMEMBER

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of destination Data Element Member
Node::base:[dir]name(i)->type(i)

DIR

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ’DATA’
Default source directory

DESTDIR

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'DATA'
Default source directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'DB'
Default source Data Base

DESTBASE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'DSTDB'
Default destination Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'E'
Default source Node name

DESTNODE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: 'E'
Default destination Node name

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. sswitch default: '/KEEP_MAP'

/KEEP_MAP Inhibit unmap of Data Base

/NOKEEP_MAP Unmap of Data Base

Function

Source and target Data Element must exist. M$LOMEM is called to get pointer to member for source DE and destination DE. Some conversion are done like PL/I :

BF(7) -> BF(7)
BF(7) -> BF(7)
BF(15) -> BF(15)
BF(31) -> BF(31)
BF(7) -> BF(31)
BF(7) -> BF(15)
BF(15) -> BF(31)
BF(7) -> BFL(24)
BF(7) -> BFL(53)
BF(15) -> BFL(53)
BF(31) -> BFL(24)
BF(31) -> BFL(53)
BFL(24) -> BFL(53)
BFL(24) -> BFL(24)
BFL(24) -> BF(31)
bit -> bit
character -> character
character var -> character var
character var -> character
character -> character var

COPY Polygon

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

PURPOSE Copy source Polygon to destination Polygon

PARAMETERS

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

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

POLY_DIR String global replace default: '$Polygon'
Source Polygon directory name, may be wildcarded.

ESTPOLY_DIR String replace default: '$Polygon'
Destination Polygon directory name, may be wildcarded.

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

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

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

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

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

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

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

[NO] KEEP_MAP Switch default: /KEEP_MAP

Caller MDBM,MGOODBM

Author Th.KROLL

Example

EXAMPLE

```
COPY Polygon E::DB:[$Polygon]S1
                    E::NEWDB:[$Polygon]NEW_S1/NOCONFIRM
  Creates the Polygon NEW_S1 on Database NEWDB
  and copies the data from Polygon S1 from
  Database DB to Polygon NEW_S1.
COPY Polygon E::DB:[$CONDITION]S(1)
                    E::NEWDB:[$CONDITION]NEW_S2/NOCONFIRM
  Creates the Condition NEW_S2 on Database NEWDB
  and copies the data from Condition S(1) from
  Database DB to Condition NEW_S2.
COPY CONDITION E::DB:[$CONDITION]S(1:5)
                    E::NEWDB:[$CONDITION]NEW_S(3:7)/NOCO
  Creates the conditionarray NEW_S(3:7) on Database
  NEWDB and copies the data from conditionarray S
  from Database DB to conditionarray NEW_S.
COPY CONDITION E::DB:[$CONDITION]S(*)
                    E::NEWDB:[$CONDITION]NEW_S(*)/NOCO
  Creates the conditionarray NEW_S(*) on Database
  NEWDB with the same limits as source Conditions
  and copies the data from Conditionarray S from
  Database DB to Conditionarray NEW_S.
COPY CONDITION E::DB:[$CONDITION]S(*)
```

E::NEWDB:[$CONDITION]NEW_S(*)/REPLACE
Replaces the conditionarray NEW_S(3:7) on Database NEWDB and copies the data from conditionarray S from Database DB to Conditionarray NEW_S.

COPY CONDITION E::DB:[$CONDITION]S(2)
E::NEWDB:[$CONDITION]S_COPY(4)/NOCON
Creates the Condition S_COPY(4) on Database NEWDB and copies the data from Condition S(2) from Database DB to Condition S_COPY(4).

COPY CONDITION E::DB:[$CONDITION]S(2)
E::NEWDB:[$CONDITION]S_COPY(4)/REPLACE
Copies the Condition S(2) from database DB to the existing Condition S_COPY(4) on destination Database NEWDB (a replace is done).

COPY CONDITION E::DB:[$CONDITION]S(1:5)
E::NEWDB:[$CONDITION]S_COPY(1:5)/REPLA
Copies the Condition S(1:5) from database DB to the existing Condition S_COPY(1:5) on destination Database NEWDB (a replace is done).

COPY CONDITION E::DB:[$CONDITION]S(1)
E::NEWDB:[$CONDITION]S_TEST /REPLACE
Copies the Condition S(2) from database DB to the existing Condition S_TEST on destination Database NEWDB (a replace is done).

COPY CONDITION E::DB:[$CONDITION]S_ORIGNAL
E::NEWDB:[$CONDITION]S_COPY(1)/REPLACE
Copies the Condition S_ORIGINAL from database DB to the existing Condition S_COPY(1) on destination Database NEWDB (a replace is done).

COPY CONDITION E::DB:[$CONDITION]S_ORIGNAL
E::NEWDB:[$CONDITION]S_COPY/REPLACE
Copies the Condition S_ORIGINAL from database DB to the existing Condition S_COPY on destination Database NEWDB (a replace is done).

Remarks

File name E$COPO.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$COPO(CV_SRC_PONAME,CV_DST_PONAME,
           CV_SRC_DIR,CV_DST_DIR,
           CV_SRC_BASE,CV_DST_BASE,
           CV_SRC_NODE,CV_DST_NODE,
           I_REPLACE,I_NOCONFIRM,I_LOG,I_KEEP_MAP)
```

COMMAND

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

Argument / Parameter description.

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Source Polygon name, may be wildcarded.

DESTNAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Destination Polygon name, may be wildcarded.

POLY_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: ’$Polygon’
Source Polygon directory name, may be wildcarded.

DESTPOLY_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: ’$Polygon’
Destination Polygon directory name, may be wildcarded.

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Source Polygon data base.

DESTBASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Destination Polygon data base.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Source Polygon node name.

DESTNODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Destination Polygon node.

/REPLACE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: " If a Polygon exists already and /REPLACE is not set not replace will be performed.

/CONFIRM

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/CONFIRM' If a new Polygon should be created and /NOCONFIRM is set no prompt for creation will follow. If a new Polygon should be created and /NOCONFIRM is NOT set a prompt for creation will follow.

/LOG

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: " If /LOG is set the copy command displays the Polygon specifications of each Polygon copied.

/KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/KEEP_MAP'
/NOKEEP_MAP	The data base is detached after the command.
/KEEP_MAP	The data base remains attached. This is recommended.

Function

Copy Polygon to another or the same Data Base,
but only on the same Node. If a Polygon doesn't
exists the switch /NOCONFIRM has to be set. If not,
the command will prompt for creation of polygon.
Wildcard (*) for Polygon and directories and
Polygonsarray's are supported.

Remarks

Module is an action routine.

Example

```
STS$VALUE=E$COPO('S(1)','NEW_S1','$Polygon',
                 '$OTTO"DB','NEWDB','E','E',
                  0,1,1,1)
```

COPY SPECTRUM

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

PURPOSE Copy source Spectrum to destination Spectrum

PARAMETERS

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

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

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

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

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

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

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

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

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

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

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

[NO] KEEP_MAP Switch default: /KEEP_MAP

Caller MDBM,MGOODBM

Author Th.KROLL

Example

EXAMPLE

```
COPY SPECTRUM E::DB:[$SPECTRUM]S1
                  E::NEWDB:[$SPECTRUM]NEW_S1/NOCONFIRM
```
Creates the spectrum NEW_S1 on Database NEWDB and copies the data from spectrum S1 from Database DB to Spectrum NEW_S1.

```
COPY SPECTRUM E::DB:[$SPECTRUM]S(1)
                  E::NEWDB:[$SPECTRUM]NEW_S2/NOCONFIRM
```
Creates the spectrum NEW_S2 on Database NEWDB and copies the data from spectrum S(1) from Database DB to Spectrum NEW_S2.

```
COPY SPECTRUM E::DB:[$SPECTRUM]S(1:5)
                  E::NEWDB:[$SPECTRUM]NEW_S(3:7)/NOCO
```
Creates the spectrumarray NEW_S(3:7) on Database NEWDB and copies the data from spectrumarray S from Database DB to Spectrumarray NEW_S.

```
COPY SPECTRUM E::DB:[$SPECTRUM]S(*)
                  E::NEWDB:[$SPECTRUM]NEW_S(*)/NOCO
```
Creates the spectrumarray NEW_S(*) on Database NEWDB with the same limits as source Spectrum S and copies the data from Spectrumarray S from Database DB to Spectrumarray NEW_S.

```
COPY SPECTRUM E::DB:[$SPECTRUM]S(*)
                  E::NEWDB:[$SPECTRUM]NEW_S(*)/REPLACE
```

Replaces the spectrumarray NEW_S(3:7) on Database
NEWDB and copies the data from spectrumarray S
from Database DB to Spectrumarray NEW_S.

```
COPY SPECTRUM E::DB:[$SPECTRUM]S(2)
                  E::NEWDB:[$SPECTRUM]S_COPY(4)/NOCON
```

Creates the spectrum S_COPY(4) on Database NEWDB
and copies the data from spectrum S(2) from
Database DB to Spectrum S_COPY(4).

```
COPY SPECTRUM E::DB:[$SPECTRUM]S(2)
                  E::NEWDB:[$SPECTRUM]S_COPY(4)/REPLACE
```

Copies the spectrum S(2) from database DB to the
existing Spectrum S_COPY(4) on destination
Database NEWDB (a replace is done).

```
COPY SPECTRUM E::DB:[$SPECTRUM]S(1:5)
                  E::NEWDB:[$SPECTRUM]S_COPY(1:5)/REPLA
```

Copies the spectrum S(1:5) from database DB to the
existing Spectrum S_COPY(1:5) on destination
Database NEWDB (a replace is done).

```
COPY SPECTRUM E::DB:[$SPECTRUM]S(1)
                   E::NEWDB:[$SPECTRUM]S_TEST /REPLACE
```

Copies the spectrum S(2) from database DB to the
existing Spectrum S_TEST on destination
Database NEWDB (a replace is done).

```
COPY SPECTRUM E::DB:[$SPECTRUM]S_ORIGNAL
                  E::NEWDB:[$SPECTRUM]S_COPY(1)/REPLACE
```

Copies the spectrum S_ORIGINAL from database DB to
the existing Spectrum S_COPY(1) on destination
Database NEWDB (a replace is done).

```
COPY SPECTRUM E::DB:[$SPECTRUM]S_ORIGNAL
                  E::NEWDB:[$SPECTRUM]S_COPY/REPLACE
```

Copies the spectrum S_ORIGINAL from database DB to
the existing Spectrum S_COPY on destination
Database NEWDB (a replace is done).

Remarks

File name E$COSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$COSP(CV_SRC_SPECNAME,CV_DST_SPECNAME,
           CV_SRC_DIR,CV_DST_DIR,
           CV_SRC_BASE,CV_DST_BASE,
           CV_SRC_NODE,CV_DST_NODE,
           I_REPLACE,I_NOCONFIRM,I_LOG,I_KEEP_MAP)
```

COMMAND

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

Argument / Parameter description.

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Source Spectrum name, may be wildcarded.

DESTNAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Destination Spectrum name, may be wildcarded.

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: ’$SPECTRUM’
Source Spectrum directory name, may be wildcarded.

DESTSPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Destination Spectrum directory name, may be wildcarded.

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Source spectrum data base.

DESTBASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$DB'
Destination spectrum data base.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Source spectrum node name.

DESTNODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Destination spectrum node.

/REPLACE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: " If a spectrum exists already and /REPLACE is not set not replace will be performed.

/CONFIRM

Routine arg. Input BIN FIXED(15) valid values 0 or 1

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

/LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: " If /LOG is set the copy command displays the spectrum specifications of each spectrum copied.

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

/KEEP_MAP The data base remains attached. This is recommended.

Function

Copy spectrum to another or the same Data Base,
but only on the same Node. If a spectrum doesn't
exists the switch /NOCONFIRM has to be set. If not,
the command will prompt for creation of new
Spectrum.
Wildcard (*) for spectras and directories and
Spectrumarray's are supported.

Remarks

Module is an action routine.

Example

```
STS$VALUE=E$COSP('S(1)','NEW_S1','$SPECTRUM',
                 '$OTTO','DB','NEWDB','E','E'
                 0,1,1,1)
```

CREATE ALIAS

CREATE ALIAS name string environment
/GLOBAL

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

PARAMETERS

name required string default: ”
Name of alias. Valid characters as for logical names.

string required string default: ”
String to replace alias name (in quotes)

environment string replace default: ”
Optional name of environment for which the alias is should be created.

/GLOBAL switch default: ”
Alias name is created global.

NOTE All alias names are deleted during logout.

Caller MUTIL, GOOSY, MDBM, MDISP

Author H.G.Essel

Example

Create global alias:
GOOSY> CRE ALI CS ”CREATE SPECTRUM /DIG” /GLOB
Create environment alias:
GOOSY> CRE ALI CS ”CREATE SPECTRUM /DIG” susi

Remarks

File name U$CRALI.PPL

Created by GOO$UTIL:U$ALICM

Description

CALLING STS=U$CRALI(CV_alias,CV_string,CV_environment
,I_global)

COMMAND CREATE ALIAS name string environment
/GLOBAL
Argument / Parameter description.

ALIAS

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”
Name of alias. Valid characters as for logical names.

STRING

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”
String enclosed in quotes ””. The alias name is replaced by this string before command dispatching.

ENVIRONMENT

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Specifies the environment for which the alias is created. Alias names which are not global, are replaced only when their environment is active. The environment must be created by command CRENVIR If no environment is specified, /GLOBAL is assumed.

/GLOBAL

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: ”
/GLOBAL	The alias is global.

Function

Create a logical name in table LNM$ENV_env or LNM$GOOSY.

CREATE AREA

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

PURPOSE Create an Area in a Data Base

PARAMETERS

area Area name
required common default

base Data Base name
required common default

pool Pool name
required common default

areabytes Size in bytes
replace default:'100'

cluster Cluster size in bytes
replace default:'4'

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

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

Caller M$DMCMD

Author M. Richter

File name M$ACRAR.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ACRAR(CV_AREA,CV_BASE,CV_POOL,
L_AREABYTES,L_CLUSTER,I_KEEP_MAP)

ARGUMENTS

CV_AREA I Area name
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

CV_POOL I Pool name
CHAR(*) VAR

L_AREABYTES I Area size in bytes
BIN FIXED(31)

L_CLUSTER I Cluster size in bytes
BIN FIXED(31)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Create an Area in a Data Base

REMARKS Module is an action routine.

EXAMPLE STS$VALUE=M$ACRAR('ALPHA','DB','BETA',10240,16,1);
create the Area ALPHA in Pool BETA of Data Base DB
with 10 KBytes and a cluster size of 16 bytes.

CREATE BASE

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

PURPOSE Create a new Data Base (section)

PARAMETERS

base Data Base name required common replaced default

basefile Data base file name required common replaced default

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

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

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

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

basepages Size in pages
replace default:'500'

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

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

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

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

Caller M$DMCMD

Author M. Richter

File name M$ACRDB.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING

```
STS=M$ACRDB(CV_BASE,CV_BASEFILE,
      L_ADENTRIES,L_MDENTRIES,L_PDENTRIES,
      L_TDENTRIES,L_BASEPAGES,L_BYTESBIT,CV_PERMANENT,
      CV_GLOBAL)
```

ARGUMENTS

CV_BASE Data Base name
CHAR(*) VAR
Input

CV_BASEFILE Data Base file name
CHAR(*) VAR
Input

L_ADENTRIES Number of entries for Area Directory
BIN FIXED(31)
Input

L_MDENTRIES Number of entries for Master Directory
BIN FIXED(31)
Input

L_PDENTRIES Number of entries for Pool Directory
BIN FIXED(31)
Input

L_TDENTRIES Number of entries for Type Directory
BIN FIXED(31)
Input

L_BASEPAGES Data Base size in pages
BIN FIXED(31)
Input

L_BYTESBIT Bytes per Bit in Home Block Bit Map (IN PAGELETS)
BIN FIXED(31)
Input

CV_PERMANENT Section permanence
CHAR(*) VAR
Input

CV_GLOBAL Section scope
CHAR(*) VAR
Input

FUNCTION Create a new Data Base (section)

REMARKS Module is an action routine.

EXAMPLE -

CREATE CALIBRATION FIXED

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

PURPOSE	Create a Data Element for fixed calibration.
PARAMETERS	
name	Name of calibration Data Element.
entries	Number of uncalibrated/calibrated points.
cal_dir	Directory for calibration
cal_pool	Pool for calibration.
base	Data Base name
node	Node name for Data Base file
/[NO] KEEP_MAP	Inhibit the unmap of the whole Data Base.
Caller	MDBM,MGOODBM
Author	W.Spreng

Remarks

Created by	GOO$DE:E$DECMD.PPL
File name	GOO$DE:E$CRCFI.PPL

Description

CALLING	STS=E$CRCFI(CV_name,L_entries,CV_cal_dir, CV_cal_pool,CV_base,CV_node,I_keep_map)
COMMAND	CREATE CALIBRATION FIXED name entries cal_dir cal_pool base node /[NO]KEEP_MAP

NAME

Routine arg. CHAR(*) VAR

Command par. String required

Name of calibration Data Element table which should be created.

ENTRIES

Routine arg. BIN FIXED(31)

Command par. Integer default=0

Number of entries in the calibration table. Determines how many calibrated values are stored in the table.

CAL_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$CALIB

Directory for calibration Data Element

CAL_POOL

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$CAL_POOL

Pool for calibration Data Element

BASE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default = DB

Default Data Base name

NODE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default = *

Default node name

KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch negatable default=/KEEP_MAP

Switch to prevent the dettachment of the Data Base

/NOKEEP_MAP	Deattach Data Dase.
/KEEP_MAP	Keep the whole Data Base mapping context

Function

A calibration table in the specified directory and pool is generated. If the directory and the pool does not exist they will be created. The calibration table can be set by SET CALIBRATION FIXED and it can be connected to a spectrum by CALIBRATE SPECTRUM. After that it is possible to display a spectrum in calibrated units.

In case of ambigious reverse calibration functions (e.g. polynom of the order of > 1) it is impossible to determine the original spectrum units from the calibrated units. To guarantee an umambigiuos correlation of calibrated an uncalibrated values the functional dependence is kept in a calibration table.

A calibration table of type "FIXED" has a fixed stepwidth in the entries of the uncalibrated values. The table range is therefore determined by the value of it's first entry and by the stepwidth. For each uncalibrated value one table entry for the corresponding calibrated values is generated. The size and the range of the table is fixed by the number of "entries".

CREATE CALIBRATION FLOAT

CREATE CALIBRATION FLOAT name entries cal_dir
cal_pool base node
/KEEP_MAP

PURPOSE	Create Data Element for float calibration.
PARAMETERS	
name	Name of calibration Data Element.
entries	Number of uncalibrated/calibrated points.
cal_dir	Directory for calibration
cal_pool	Pool for calibration.
base	Data Base name
node	node name for Data Base file
/[NO] KEEP_MAP	Inhibit the unmap of the whole Data Base.
Caller	MDBM,MGOODBM
Author	W.Spreng

Remarks

Created by	GOO$DE:E$DECMD.PPL
File name	GOO$DE:E$CRCFL.PPL

Description

CALLING	STS=E$CRCFL(CV_name,L_entries,CV_cal_dir, CV_cal_pool,CV_base,CV_node,I_keep_map)
COMMAND	CREATE CALIBRATION FLOAT name entries cal_dir cal_pool base node /[NO]KEEP_MAP

NAME

Routine arg. CHAR(*) VAR

Command par. String required

Name of calibration Data Element table which should be created.

ENTRIES

Routine arg. BIN FIXED(31)

Command par. Integer default=0

Number of entries in the calibration table. Determines how many calibrated values are stored in the table.

CAL_DIR

Routine arg. CHAR(*) VAR

Command par. String global replacable default=$CALIB

Directory for calibration Data Element

CAL_POOL

Routine arg. CHAR(*) VAR

Command par. String global replacable default=$CAL_POOL

Pool for calibration Data Element

BASE

Routine arg. CHAR(*) VAR

Command par. String global replacable default = DB

Default Data Base name

NODE

Routine arg. CHAR(*) VAR

Command par. String global replacable default = *

Default node name

KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch negatgable default=/KEEP_MAP

Switch to prevent the dettachment of the Data Base

/NOKEEP_MAP Deattach Data Dase.

/KEEP_MAP Keep the whole Data Base mapping context

Function

A calibration table in the specified directory and pool is generated. If the directory and the pool does not exist they will be created. The calibration table can be set by SET CALIBRATION FIXED and it can be connected to a spectrum by CALIBRATE SPECTRUM. After that it is possible to display a spectrum in calibrated units.

In case of ambigious reverse calibration functions (e.g. polynom of the order of > 1) it is impossible to determine the original spectrum units from the calibrated units. To guarantee an umambigiuos correlation of calibrated an uncalibrated values the functional dependence is kept in a calibration table.

A calibration table of type "FLOAT" contains out of a list of uncalibrated values and the corresponding calibrated values. The number of entries in the table is fixed by the parameter "entries".

CREATE CALIBRATION LINEAR

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

PURPOSE Create Data Element for linear calibration.

PARAMETERS

name Name of calibration Data Element.

cal_dir Directory for calibration

cal_pool Pool for calibration.

base Data Base name

node node name for Data Base file

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

Caller MDBM,MGOODBM

Author W.Spreng

Remarks

Created by GOO$DE:E$DECMD.PPL

File name GOO$DE:E$CRCLI.PPL

Description

CALLING STS=E$CRCLI(CV_name,CV_cal_dir,CV_cal_pool,
CV_base,CV_node,I_keep_map)

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

NAME

Routine arg. CHAR(*) VAR

Command par. String required
Name of calibration Data Element table which should be created.

CAL_DIR

Routine arg. CHAR(*) VAR

Command par. String global replacable default=$CALIB
Directory for calibration Data Element

CAL_POOL

Routine arg. CHAR(*) VAR

Command par. String global replacable default=$CAL_POOL
Pool for calibration Data Element

BASE

Routine arg. CHAR(*) VAR

Command par. String global replacable default = DB
Default Data Base name

NODE

Routine arg. CHAR(*) VAR

Command par. String global replacable default = *
Default node name

/KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch negatgable default=/KEEP_MAP
Switch to prevent the dettachment of the Data Base

/NOKEEP_MAP Deattach Data Dase.

/KEEP_MAP	Keep the whole Data Base mapping context

Function

A calibration table in the specified directory and pool is generated. If the Directory and the Pool does not exist they will be created.

Linear calibrations contains the two parameter of a linear polynom.

CREATE CONDITION COMPOSED

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

PURPOSE create a composed condition

PARAMETERS

name required string replace default: ”
name of the condition

expression string replace default: ”
Boolean expression of conditions.

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

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

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

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

CRE COND COMP CC (A&B)—C

Remarks

File name GOO$DE:E$ACRCC.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

STS=E$ACRCC(CV_name,CV_expression,
CV_cond_dir,CV_cond_pool,CV_base,CV_node,
I_dynamic,I_document,I_keep_map,B_mask)

COMMAND

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

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of the condition. This has the standard GOOSY format base:[directory]name. Base and directory are defaulted by the explicit parameters. The values here specified are not replaced as defaults!

EXPRESSION

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Boolean expression of conditions

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
default directory

COND_POOL

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$COND_POOL'
default pool

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
default base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
default node

/DYNAMIC

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

/KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	negatable switch default: /KEEP_MAP
/KEEP_MAP	Do not unmap Data Base
/NOKEEP_MAP	unmap Data Base In a procedure call this argument should be always 1 to prevent an unmap of the data base.

MASK

Routine arg.	Input BIT(*) ALIGNED Bits are set by command interface for specified parameters.

Function

Create a composed condition.

CREATE CONDITION FUNCTION

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

PURPOSE create a function condition

PARAMETERS

name required string replace default: ”
name of the condition

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

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

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

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

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE COND FUNCTION USER MYSHARE(CHECK)
```

Remarks

File name GOO$DE:E$ACRFC.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$ACRFC(CV_name,CV_function,
        CV_cond_dir,CV_cond_pool,CV_base,CV_node,
        I_dynamic,I_document,I_keep_map,B_mask)
```

COMMAND

CREATE CONDITION WINDOW name function
cond_dir cond_pool base node
/[NO]DYNAMIC
/[NO]DOCUMENT
/[NO]KEEP_MAP
Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of the condition. This has the standard GOOSY format base:[directory]name. Base and directory are defaulted by the explicit parameters. The values here specified are not replaced as defaults!

FUNCTION

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Specification of function to be called.
The module must be linked in a sharable image. Both are specified by 'image(module)'.

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
default directory

COND_POOL

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$COND_POOL'
default pool

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
default base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
default node

/DYNAMIC

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

/KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	negatable switch default: /KEEP_MAP
/KEEP_MAP	Do not unmap Data Base
/NOKEEP_MAP	unmap Data Base In a procedure call this argument should be always 1 to prevent an unmap of the data base.

MASK

Routine arg.	Input BIT(*) ALIGNED Bits are set by command interface for specified parameters.

Function

Create a function condition.

CREATE CONDITION MULTIWINDOW

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

PURPOSE create a multiwindow condition

PARAMETERS

name required string replace default: ”
name of the condition

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

dimension integer default: 1
Internal dimension

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

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

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

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

CRE COND MULTI BINS (23,64) 5

Remarks

File name GOO$DE:E$ACRMW.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

STS=E$ACRMW(CV_name,CV_limits,L_dimension,
CV_cond_dir,CV_cond_pool,CV_base,CV_node,
I_dynamic,I_document,I_keep_map,B_mask)

COMMAND

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

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of the condition. This has the standard GOOSY format base:[directory]name. Base and directory are defaulted by the explicit parameters. The values here specified are not replaced as defaults!

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '(0,4096)'
Specification of limits as string. The number of limits specified is used to calculate the internal dimension of the condition. If this value is smaller than the one specified explicitly, than the last limits are used for the following dimensions.

DIMENSION

Routine arg. Input BIN FIXED(31)

Command par. Integer default: 1
Internal dimension (number of pairs of limits).
If there are more limits specified, the number of limits defines the internal dimension. If there were not enough limits specified, the last limits are used for all subwindows.

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
default directory

COND_POOL

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$COND_POOL'
default pool

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
default base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
default node

/DYNAMIC

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

/NOKEEP_MAP unmap Data Base In a procedure call this argument should be always 1 to prevent an unmap of the data base.

MASK

Routine arg. Input BIT(*) ALIGNED
Bits are set by command interface for specified parameters.

Function

Create a window condition.

CREATE CONDITION PATTERN

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

PURPOSE create a pattern condition

PARAMETERS

name required string replace default: ”
name of the condition

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

dimension integer default: 1
Internal dimension

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

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

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

node string global replace default: 'E'
default node

invert string default: '0'
inversion pattern

/[NO] DYNAMIC switch default: /DYNAMIC'

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

checkmode set default: /IDENT valid values are:

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE COND PAT MAINPAT 101010 INV=101
CRE COND PAT SUBPAT 11111
```

Remarks

File name GOO$DE:E$ACRPC.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING
```
STS=E$ACRPC(CV_name,CV_pattern,L_dimension,
        CV_cond_dir,CV_cond_pool,CV_base,CV_node,
        CV_invert,I_dynamic,I_document,
        CV_checkmode,I_keep_map,B_mask)
```

COMMAND
```
CREATE CONDITION PATTERN name pattern dimension
  cond_dir cond_pool base node invert
  /[NO]DYNAMIC
  /[NO]DOCUMENT
  /IDENT/INCL/ANY/EXCL
  /[NO]KEEP_MAP
```
Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of the condition. This has the standard GOOSY format base:[directory]name. Base and directory are defaulted by the explicit parameters. The values here specified are not replaced as defaults!

PATTERN

Routine arg. Input CHAR(*) VAR

Command par. string replace default:
'11111111111111111111111111111111'
specification of test pattern. Can be specified as 010101 or '010101'B. Internally a BIT(32) is used. The pattern is padded with zeros to the right. More than 32 bits are stored in several subpatterns.

DIMENSION

Routine arg. Input BIN FIXED(31)

Command par. Integer default: 1
Internal dimension (number of patterns). For patterns longer than 32 bits.

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
default directory

COND_POOL

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$COND_POOL'
default pool

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
default base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
default node

INVERT

Routine arg. Input CHAR(*) VAR

Command par. string default: '0'
invert pattern used to invert bits of the object pattern before the check.

/DYNAMIC

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

CHECKMODE

Routine arg.	Input CHAR(*) VAR
Command par.	set default: /IDENT
/IDENT	pattern and object must be identical
/EXCL	all bits set in object must be set in pattern (like ANY exclusive additional bits set in object)
/ANY	pattern and object must have at least one common bit set
/INCL	all bits set in pattern must be set in object (like IDENT inclusive additional bits set in object).

/KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	negatable switch default: /KEEP_MAP
/KEEP_MAP	Do not unmap Data Base
/NOKEEP_MAP	unmap Data Base In a procedure call this argument should be always 1 to prevent an unmap of the data base.

MASK

Routine arg.	Input BIT(*) ALIGNED Bits are set by command interface for specified parameters.

Function

Create a pattern condition.

CREATE CONDITION POLYGON

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

PURPOSE create a polygon condition

PARAMETERS

name required string replace default: ”
name of the condition

polygon string replace default: ”
Name of polygon.

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

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

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

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

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

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE COND POLY CPOLY [XX]POL1 1
:sp.
CRE COND POLY C_VEC(6) POLY_VEC(2:7)
:sp.
CRE COND POLY C_VEC(6) POLY_MAT(2,3)
:sp.
CRE COND POLY C_MAT(2,3) POLY_VEC(6)
:sp.
CRE COND POLY C_MAT(2,3) POLY_MAT(2,3)
```

Remarks

File name GOO$DE:E$ACRPL.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$ACRPL(CV_name,CV_polygon,L_dimension,
        CV_cond_dir,CV_poly_dir,CV_cond_pool,
        CV_base,CV_node,
        I_dynamic,I_document,I_keep_map,B_mask)
```

COMMAND

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

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of the condition. This has the standard GOOSY format base:[directory]name. Base and directory are defaulted by the explicit parameters. The values here specified are not replaced as defaults!

POLYGON

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Name specification of a polygon which must exist in the data base. This parameter is required if one wants to use the condition in a macro ($COND). In a dynamic list the polygon can be specified for a polygon condition.

DIMENSION

Routine arg. Input BIN FIXED(31)

Command par. Integer default: 1
Internal dimension (number of polygons). presently only one is supported.

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
default condition directory

POLY_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$POLYGON'
default polygon directory

COND_POOL

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$COND_POOL'
default pool

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
default base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
default node

/DYNAMIC

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

/NOKEEP_MAP unmap Data Base In a procedure call this argument should be always 1 to prevent an unmap of the data base.

MASK

Routine arg. Input BIT(*) ALIGNED
Bits are set by command interface for specified parameters.

Function

Create a polygon condition. If a polygon is specified, this polygon is linked to the condition. If inserted in a dynamic list, a polygon can be specified al well. If the condition is executed in a $COND macro, it must be located first by $LOC. In this case a polygon must be linked to the condition.

CREATE CONDITION WINDOW

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

PURPOSE create a window condition

PARAMETERS

name required string replace default: ”
name of the condition

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

dimension integer default: 1
Internal dimension

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

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

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

node string global replace default: ’E’
default node

/[NO] DYNAMIC switch default: /DYNAMIC’

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

CRE COND WIND ENER (23,64)

Remarks

File name GOO$DE:E$ACRWC.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

STS=E$ACRWC(CV_name,CV_limits,L_dimension,
CV_cond_dir,CV_cond_pool,CV_base,CV_node,
I_dynamic,I_document,I_keep_map,B_mask)

COMMAND

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

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of the condition. This has the standard GOOSY format base:[directory]name. Base and directory are defaulted by the explicit parameters. The values here specified are not replaced as defaults!

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '(0,4096)'
Specification of limits as string. The number of
limits specified is used to calculate the internal dimension of the condition. If this value is smaller than the one specified explicitly, than the last limits are used for the following dimensions.

DIMENSION

Routine arg. Input BIN FIXED(31)

Command par. Integer default: 1
Internal dimension (number of pairs of limits).
If there are more limits specified, the number of limits defines the internal dimension. If there were not enough limits specified, the last limits are used for all subwindows.

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
default directory

COND_POOL

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$COND_POOL'
default pool

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
default base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
default node

/DYNAMIC

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

/NOKEEP_MAP unmap Data Base In a procedure call this argument should be always 1 to prevent an unmap of the data base.

MASK

Routine arg. Input BIT(*) ALIGNED
Bits are set by command interface for specified parameters.

Function

Create a window condition.

CREATE DIRECTORY

CREATE DIRECTORY dir dedentries base
/[NO]KEEP_MAP

PURPOSE Create a Data Element Directory in a Data Base

PARAMETERS

dir Data Element Directory name
required, common default

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

base Data Base name
required, common default

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

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

Caller M$DMCMD

Author M. Richter

File name M$ACRDI.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ACRDI(CV_DIR,L_DEDENTRIES,CV_BASE,I_KEEP_MAP)

ARGUMENTS

CV_DIR I Data Element Directory name
CHAR(*) VAR

L_DEDENTRIES I Number of entries for Data Element Directory
BIN FIXED(31)

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of the Data Base
BIN FIXED (15)

FUNCTION Create a Data Element Directory in a Data Base

REMARKS Module is an action routine.

EXAMPLE STS$VALUE = M$ACRDI('PARAM',300,'DB',1)
create the Data Element Directory 'PARAM' in the Data Base 'DB' with space for 300 Data Elements.

CREATE DYNAMIC ENTRY BITSPECTRUM

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

PURPOSE Create a spectrum dynamic list entry

PARAMETERS

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

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

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

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

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

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

par_dir string global replace default: ’DATA’
Default directory

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN EN BITSPECTRUM dlist [d]patt(1:10)
            PAR=[d]$event.patt(1:10)
[d]is the directory specification
```

Remarks

File name M$ACEBS.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING STS=M$ACEBS(CV_dyn_list,CV_spectrum,
CV_parameter,CV_increment,CV_condition,
CV_dyn_dir,CV_par_dir,CV_cond_dir,CV_spec_dir,
CV_base,CV_node,
I_update,I_check,I_keep_map)

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

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Spectrum name specification. Directory is defaulted by SPEC_DIR. Array limits: [dir]name(ll:ul).

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications separated by commas. As may as spectrum dimension. Type must be BIT(16) or BIT(32) ALIGNED. Directory is defaulted by PAR_DIR.

INCREMENT

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Data element member specification used for increment. If not specified, 1 is used as increment. Directory is defaulted by PAR_DIR.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. string default: ”
If specified, the spectrum is filled only if the condition was true. The condition must be executed explicitely (either in a dynamic list or in the analysis program). Directory is defaulted by PAR_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$DYNAMIC'
Default directory of the dynamic list

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DATA'
Default directory of the parameters

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Default directory of the condition

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Default directory of the spectrum

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Default node.

/UPDATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”

/UPDATE The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

/NOCHECK No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

/KEEP_MAP The data base remains attached. This is recommended.

Function

Create a dynamic list entry for bitspectrum accumulation. Supports spectra of type BIN FIXED(31) with 1 dimension. Coordinates can be BIT(16) or BIT(32) ALIGNED. Specified spectrum can be an array.

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1. **Master procedures (PROCEDURE /MASTER)**
2. **Master pattern conditions (PATTERN /MASTER)**
3. **Master window conditions (WINDOW /MASTER)**
4. **Master function conditions (FUNCTION /MASTER)**
5. **Master polygon conditions (POLYGON /MASTER)**
6. **Master composed conditions (COMPOSED /MASTER)**
7. **Procedures (PROCEDURE)**
8. **Pattern conditions (PATTERN)**
9. **Window conditions (WINDOW)**
10. **Multi Window conditions (MULTI)**
11. **Function conditions (FUNCTION)**
12. **Polygon conditions (POLYGON)**
13. **Composed conditions (COMPOSED)**
14. **Spectrum accumulation indexed (INDEXED)**
15. **Spectrum accumulation (SPECTRUM)**
16. **Bit spectrum accumulation (BITSPECTRUM)**
17. **Scatter plots (SCATTER)**

Arrays

Spectra or conditions may be arrays. In this case an index range must be specified. All additional data elements must be either scalar or indexed by the same range. Ranges are specified by (l:u).

Examples

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

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

```
  CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
                 PARA=[d]$event.e(2:4) CONDI=[d]m_window
  CRE DYN EN INDEXED dlist [d]ener(2:4)
                 PARA=[d]$event.e(1) INDEX=[d]m_window
```

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC ENTRY COMPOSED

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

PURPOSE Create a composed condition dynamic list entry

PARAMETERS

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

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

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

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

CRE DYN EN COMPOSED dlist [d]all_ok /MASTER
[d]is the directory specification
CRE DYN ENTR COMP 1 [$CONDITION]CC

Remarks

File name M$ACECC.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING
STS=M$ACECC(CV_dyn_list,CV_condition,
CV_dyn_dir,CV_cond_dir,CV_base,CV_node,
I_master,I_update,I_check,I_keep_map)

COMMAND
CREATE DYNAMIC ENTRY COMPOSED dyn_list condition
dyn_dir cond_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and
directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specifiaction of composed condition.
Directory is defaulted from COND_DIR.
No array supported.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$DYNAMIC'
Default directory of the dynamic list

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Default directory of the condition

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Default node.

/MASTER

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”

/MASTER Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated. Master condition result bits are checked any time, even if the condition was already executed. So, if the same master condition is in two dynamic lists, both lists are skipped, if the condition was false.

/UPDATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”

/UPDATE The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

/NOCHECK No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

/KEEP_MAP The data base remains attached. This is recommended.

Function

Create a dynamic list entry for composed condition. A boolean expression of conditions is executed. The expression is specified in the condition data el.

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1. **Master procedures (PROCEDURE /MASTER)**
2. **Master pattern conditions (PATTERN /MASTER)**
3. **Master window conditions (WINDOW /MASTER)**
4. **Master function conditions (FUNCTION /MASTER)**
5. **Master polygon conditions (POLYGON /MASTER)**
6. **Master composed conditions (COMPOSED /MASTER)**
7. **Procedures (PROCEDURE)**
8. **Pattern conditions (PATTERN)**
9. **Window conditions (WINDOW)**
10. **Multi Window conditions (MULTI)**
11. **Function conditions (FUNCTION)**
12. **Polygon conditions (POLYGON)**
13. **Composed conditions (COMPOSED)**
14. **Spectrum accumulation indexed (INDEXED)**
15. **Spectrum accumulation (SPECTRUM)**
16. **Bit spectrum accumulation (BITSPECTRUM)**
17. **Scatter plots (SCATTER)**

CREATE DYNAMIC ENTRY FUNCTION

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

PURPOSE Create a function condition dynamic list entry

PARAMETERS

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

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

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

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

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

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

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN ENTRY FUNCTION dlist x$cond
                  MOD=privshar(x$cond)
                  PAR=([d]$event.z4.de(5),$event.z5)
  [d]is the directory specification
  X$COND must be in the form:
  ENTRY(BIT(1) ALIGNED,POINTER,POINTER)
  RETURNS(BIN FIXED(31))
  CRE DYN ENTR FUNC l c_cali MYSHR(CALI)
                    PAR=[DATA]EVENT.RAW
  CALI must be in the form:
  ENTRY(BIT(1) ALIGNED,POINTER)
  RETURNS(BIN FIXED(31))
  Call procedure CALI which is linked in sharable
    image MYSHR. Pass pointer to data element as second argument.
```

Remarks

File name M$ACEFC.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING STS=M$ACEFC(CV_dyn_list,CV_condition,CV_module,
CV_parameter,CV_dyn_dir,CV_par_dir,CV_cond_dir,

CV_base,CV_node,
I_master,I_update,I_check,I_keep_map)

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

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name of the function condition.

MODULE

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Specification of module in the form image(module), where image is the name of the sharable image in which the module must be linked. The first argument of the procedure must be a BIT(1) ALIGNED variable returning the condition value (’1’B for true, ’0’B for false).

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ''
List of data element member name specifications
separated by commas. The pointers to these data element members are passed as arguments to the procedure. Base and directory are defaulted by the values specified by BASE and PAR_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$DYNAMIC'
Default directory of the dynamic list

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DATA'
Default directory of the parameters

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Default directory of the condition

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Default node.

/MASTER

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: "
/MASTER	Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated. Master condition result bits are checked any time, even if the condition was already executed. So, if the same master condition is in two dynamic lists, both lists are skipped, if the condition was false.

/UPDATE

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: "
/UPDATE	The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/CHECK'
/CHECK	The dynamic list is checked for validity.
/NOCHECK	No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/KEEP_MAP'
/NOKEEP_MAP	The data base is detached after the command.
/KEEP_MAP	The data base remains attached. This is recommended.

Function

Create a function condition entry.

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1. **Master procedures (PROCEDURE /MASTER)**
2. **Master pattern conditions (PATTERN /MASTER)**
3. **Master window conditions (WINDOW /MASTER)**
4. **Master function conditions (FUNCTION /MASTER)**
5. **Master polygon conditions (POLYGON /MASTER)**
6. **Master composed conditions (COMPOSED /MASTER)**
7. **Procedures (PROCEDURE)**
8. **Pattern conditions (PATTERN)**
9. **Window conditions (WINDOW)**
10. **Multi Window conditions (MULTI)**
11. **Function conditions (FUNCTION)**
12. **Polygon conditions (POLYGON)**
13. **Composed conditions (COMPOSED)**
14. **Spectrum accumulation indexed (INDEXED)**
15. **Spectrum accumulation (SPECTRUM)**
16. **Bit spectrum accumulation (BITSPECTRUM)**
17. **Scatter plots (SCATTER)**

CREATE DYNAMIC ENTRY INDEXEDSPECTRUM

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

PURPOSE Create an indexed spectrum dynamic list entry

PARAMETERS

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

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

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

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

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

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

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

par_dir string global replace default: ’DATA’
Default directory

cond_dir	string global replace default: '$CONDITION' Default directory
spec_dir	string global replace default: '$SPECTRUM' Default directory
base	string global replace default: 'DB' Default data base name
node	string global replace default: 'E' Default node name
/UPDATE	switch default: " Update dynamic list (then it becames valid for a running analysis immediately.
/[NO] CHECK	switch default: /CHECK Do dynamic list checking by attaching it
/[NO] KEEP_MAP	switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base.
Caller	MDBM, MGOODBM
Author	H.G.Essel

Example

```
CRE DYN EN INDEX dlist [d]ener(1:10)
            PAR=[d]$event.e
            IND=[d]$event.i
CRE DYN EN INDEX dlist [d]ede(1:5)
            PAR=([d]$event.e(1:5),$event.de(1:5))
            IND=[d]$event.i
[d]is the directory specification
```

Remarks

File name	M$ACEIS.PPL
Created by	GOO$DM:M$DMCMD

Description

CALLING

STS=M$ACEIS(CV_dyn_list,CV_spectrum,
CV_parameter,CV_index,CV_increment,
CV_condition,CV_dyn_dir,CV_par_dir,
CV_cond_dir,CV_spec_dir,CV_base,CV_node,
I_update,I_check,I_keep_map)

COMMAND

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

Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Spectrum name specification. Directory is defaulted by SPEC_DIR. Array limits: [dir]name(ll:ul). Spectrum must be array.

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications separated by commas. As may as spectrum dimension. Directory is defaulted by PAR_DIR.

INDEX

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Data element member specification or multiwindow used for index. Directory is defaulted by PAR_DIR. Therefore a multi window must be specified with directory.

INCREMENT

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Data element member specification used for increment. If not specified, 1 is used as increment. Directory is defaulted by PAR_DIR.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. string default: ”
If specified, the spectrum is filled only if the condition was true. The condition must be executed explicitely (either in a dynamic list or in the analysis program). Directory is defaulted by PAR_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$DYNAMIC'
Default directory of the dynamic list

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DATA'
Default directory of the parameters

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Default directory of the condition

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Default directory of the spectrum

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Default node.

/UPDATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''

/UPDATE The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

/NOCHECK No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

/KEEP_MAP The data base remains attached. This is recommended.

Function

Create a dynamic list entry for indexed spectrum accumulation. Supports spectra of type BIN FIXED(31) or BIN FLOAT(24) with up to 2 dimensions. Coordinates can be BIN FIXED(31) or BIN FLOAT(24). Specified spectrum must be an array.

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1.Master procedures (PROCEDURE /MASTER)

2.Master pattern conditions (PATTERN /MASTER)

3.Master window conditions (WINDOW /MASTER)

4.Master function conditions (FUNCTION /MASTER)

5.Master polygon conditions (POLYGON /MASTER)

6.Master composed conditions (COMPOSED /MASTER)

7.Procedures (PROCEDURE)

8.Pattern conditions (PATTERN)

9.Window conditions (WINDOW)

10.Multi Window conditions (MULTI)

11.Function conditions (FUNCTION)

12.Polygon conditions (POLYGON)

13.Composed conditions (COMPOSED)

14.Spectrum accumulation indexed (INDEXED)

15.Spectrum accumulation (SPECTRUM)

16.Bit spectrum accumulation (BITSPECTRUM)

17.Scatter plots (SCATTER)

Arrays

Spectra or conditions may be arrays. In this case an index range must be specified. All additional data elements must be either scalar or indexed by the same range. Ranges are specified by (l:u).

Examples

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

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

```
  CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
                 PARA=[d]$event.e(2:4) CONDI=[d]m_window
  CRE DYN EN INDEXED dlist [d]ener(2:4)
                 PARA=[d]$event.e(1) INDEX=[d]m_window
```

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC ENTRY MULTIWINDOW

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

PURPOSE Create a multiwindow condition dynamic list entry

PARAMETERS

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

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

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

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

par_dir string global replace default: 'DATA'
Default directory

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN EN MULTI dlist [d]e_recoil
            PAR=[d]$event.ener
  [d]is the directory specification
  CRE DYN ENTR MULTI 1 [$CONDITION]M PAR=[DATA]EVT.X
```

Remarks

File name M$ACEMW.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING
STS=M$ACEMW(CV_dyn_list,CV_condition,
CV_parameter,CV_dyn_dir,CV_par_dir,CV_cond_dir,
CV_base,CV_node,
I_update,I_check,I_keep_map)

COMMAND
CREATE DYNAMIC ENTRY MULTIWINDOW dyn_list condition parameter dyn_dir par_dir cond_dir base node
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of the multiwindow condition.
Directory is defaulted from COND_DIR.

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications
separated by commas. The number of elements must match the number of subwindows. The directory is defaulted from PAR_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$DYNAMIC'
Default directory of the dynamic list

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DATA'
Default directory of the parameters

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Default directory of the condition

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Default node.

/UPDATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”

/UPDATE The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

/NOCHECK No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

/KEEP_MAP The data base remains attached. This is recommended.

Function

Create a multiwindow condition dynamic list entry. Check specified member versus all window limits. For each check a result bit is set, which may be used to increment a spectrum array member. In addition, the number of the last matching window may be used as index of a spectrum array member (see INDEXED).

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1.Master procedures (PROCEDURE /MASTER)

2.Master pattern conditions (PATTERN /MASTER)

3.Master window conditions (WINDOW /MASTER)

4.Master function conditions (FUNCTION /MASTER)

5.Master polygon conditions (POLYGON /MASTER)

6.Master composed conditions (COMPOSED /MASTER)

7.Procedures (PROCEDURE)

8.Pattern conditions (PATTERN)

9.Window conditions (WINDOW)

10.Multi Window conditions (MULTI)

11.Function conditions (FUNCTION)

12.Polygon conditions (POLYGON)

13.Composed conditions (COMPOSED)

14.Spectrum accumulation indexed (INDEXED)

15.Spectrum accumulation (SPECTRUM)

16.Bit spectrum accumulation (BITSPECTRUM)

17.Scatter plots (SCATTER)

Arrays

The difference between windows and multiwindows is that multiwindows have only one object for all * subwindows, but one result bit for each, whereas windows need one object per subwindow, but has only one result bit (set, if all subwindows are true). Multi windows may be used as filters for spectrum array accumulation. The internal dimension of the window must match the specified index range. It may also be used for indexed spectrum accumulation. Then

the index of the matching subwindow is used to select the spectrum member. In the first case, the subwindows may overlapp, in the second case this makes normally no sense.

```
CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
            PARA=[d]$event.e(2:4) CONDI=[d]m_window
CRE DYN EN INDEXED dlist [d]ener(2:4)
            PARA=[d]$event.e(1) INDEX=[d]m_window
```

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC ENTRY PATTERN

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

PURPOSE Create a pattern condition dynamic list entry

PARAMETERS

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

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

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

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

par_dir string global replace default: 'DATA'
Default directory

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN EN PATTERN dlist [d]main_pat
            PAR=[d]$event.pat
            /MASTER
  [d]is the directory specification
  CRE DYN ENTR PAT 1 [$CONDITION]P PAR=[DATA]EVENT.B
```

Remarks

File name M$ACEPC.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING
```
STS=M$ACEPC(CV_dyn_list,CV_condition,CV_parameter,
  CV_dyn_dir,CV_par_dir,CV_cond_dir,
  CV_base,CV_node,
  I_master,I_update,I_check,I_keep_map)
```

COMMAND
```
CREATE DYNAMIC ENTRY PATTERN dyn_list condition
parameter dyn_dir par_dir cond_dir base node
   /MASTER
   /UPDATE
   /[NO]CHECK
   /[NO]KEEP_MAP
```
Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of a pattern condition. The directory name is defaulted from COND_DIR.

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications separated by commas. The number of elements must match the internal dimensionality. The members must be BIT(16) ALIGNED or BIT(32) ALIGNED. The directory is defaulted from PAR_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$DYNAMIC'
Default directory of the dynamic list

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DATA'
Default directory of the parameters

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Default directory of the condition

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Default node.

/MASTER

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''

/MASTER Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated. Master condition result bits are checked any time, even if the condition was already executed. So, if the same master condition is in two dynamic lists, both lists are skipped, if the condition was false.

/UPDATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''

/UPDATE The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/CHECK'
/CHECK	The dynamic list is checked for validity.
/NOCHECK	No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/KEEP_MAP'
/NOKEEP_MAP	The data base is detached after the command.
/KEEP_MAP	The data base remains attached. This is recommended.

Function

Create a pattern condition entry. Check specified members versus patterns. Note that four test modes can be specified with the pattern condition DE (ALL, ANY, MATCH, EXCL). The values of the members can be inverted bitwise. Up to 8 internal dimensions.

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1.Master procedures (PROCEDURE /MASTER)

2.Master pattern conditions (PATTERN /MASTER)

3.Master window conditions (WINDOW /MASTER)

4.Master function conditions (FUNCTION /MASTER)

5.Master polygon conditions (POLYGON /MASTER)

6.Master composed conditions (COMPOSED /MASTER)

7.Procedures (PROCEDURE)

8.Pattern conditions (PATTERN)

9.Window conditions (WINDOW)

10.Multi Window conditions (MULTI)

11.Function conditions (FUNCTION)

12.Polygon conditions (POLYGON)

13.Composed conditions (COMPOSED)

14.Spectrum accumulation indexed (INDEXED)

15.Spectrum accumulation (SPECTRUM)

16.Bit spectrum accumulation (BITSPECTRUM)

17.Scatter plots (SCATTER)

Arrays

Spectra or conditions may be arrays. In this case an index range must be specified. All additional data elements must be either scalar or indexed by the same range. Ranges are specified by (l:u).

Examples

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

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

```
  CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
```

PARA=[d]$event.e(2:4) CONDI=[d]m_window
CRE DYN EN INDEXED dlist [d]ener(2:4)
PARA=[d]$event.e(1) INDEX=[d]m_window
[d]is the directory specification
In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC ENTRY POLYGON

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

PURPOSE Create a polygon condition dynamic list entry

PARAMETERS

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

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

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

polygon string replace default: ”
Name of a polygon.

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

par_dir string global replace default: 'DATA'
Default directory

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

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

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

node	string global replace default: 'E' Default node name
/MASTER	switch default: " Master procedure (executed first)
/UPDATE	switch default: " Update dynamic list (then it becames valid for a running analysis immediately.
/[NO] CHECK	switch default: /CHECK Do dynamic list checking by attaching it
/[NO] KEEP_MAP	switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base.
Caller	MDBM, MGOODBM
Author	H.G.Essel

Example

```
CRE DYN ENTR POLY 1 [$CONDITION]poco
      PAR=([DATA]EVENT.RAW(1),[DATA]EVENT.RAW(2))
```

Remarks

File name	M$ACEPL.PPL
Created by	GOO$DM:M$DMCMD

Description

CALLING	STS=M$ACEPL(CV_dyn_list,CV_condition, CV_parameter,CV_polygon, CV_dyn_dir,CV_par_dir,CV_cond_dir,CV_poly_dir, CV_base,CV_node, I_master,I_update,I_check,I_keep_map)
COMMAND	CREATE DYNAMIC ENTRY POLYGON dyn_list condition parameter polygon dyn_dir par_dir cond_dir poly_dir base node /MASTER /UPDATE /[NO]CHECK

/[NO]KEEP_MAP
Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and
directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
name specification of polygon condition.
Directory is defaulted from COND_DIR.

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
List of two data element member names
separated by commas. Directory is defaulted from PAR_DIR.

POLYGON

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Optional specification of a polygon. Normally the
polygon is specified in the polygon condition.
Directory is defaulted from POLY_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: ’$DYNAMIC’
Default directory of the dynamic list

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DATA'
Default directory of the parameters

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Default directory of the condition

POLY_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$POLYGON'
Default directory of the polygon

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Default node.

/MASTER

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”

/MASTER Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated. Master condition result bits are checked any

time, even if the condition was already executed. So, if the same master condition is in two dynamic lists, both lists are skipped, if the condition was false.

/UPDATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”

/UPDATE The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

/NOCHECK No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

/KEEP_MAP The data base remains attached. This is recommended.

Function

Create a dynamic list entry for polygon condition.

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1. **Master procedures (PROCEDURE /MASTER)**
2. **Master pattern conditions (PATTERN /MASTER)**
3. **Master window conditions (WINDOW /MASTER)**
4. **Master function conditions (FUNCTION /MASTER)**
5. **Master polygon conditions (POLYGON /MASTER)**
6. **Master composed conditions (COMPOSED /MASTER)**
7. **Procedures (PROCEDURE)**
8. **Pattern conditions (PATTERN)**
9. **Window conditions (WINDOW)**
10. **Multi Window conditions (MULTI)**
11. **Function conditions (FUNCTION)**
12. **Polygon conditions (POLYGON)**
13. **Composed conditions (COMPOSED)**
14. **Spectrum accumulation indexed (INDEXED)**
15. **Spectrum accumulation (SPECTRUM)**
16. **Bit spectrum accumulation (BITSPECTRUM)**
17. **Scatter plots (SCATTER)**

CREATE DYNAMIC ENTRY PROCEDURE

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

PURPOSE Create a procedure call dynamic list entry

PARAMETERS

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

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

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

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

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

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

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN ENTRY PROCEDURE dlist
                MOD=privshar(x$loop)
                PAR=([d]$event.z4.de(5),$event.z5)
                /MASTER
  [d]is the directory specification
  X$LOOP must be in the form:
  ENTRY(POINTER,POINTER) RETURNS(BIN FIXED(31))
  CRE DYN ENTR PROC l MYSHR(CALI) PAR=[DATA]EVENT.RAW
  Call procedure CALI which is linked in sharable
    image MYSHR. Pass pointer to data element as argument.
  CALI must be in the form:
  ENTRY(POINTER) RETURNS(BIN FIXED(31))
```

Remarks

File name M$ACEPR.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING

STS=M$ACEPR(CV_dyn_list,CV_module,
CV_parameter,CV_condition,
CV_dyn_dir,CV_par_dir,CV_cond_dir,
CV_base,CV_node,
I_master,I_update,I_check,I_keep_map)

COMMAND

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

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and
directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

MODULE

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Specification of module in the form image(module),
where image is the name of the sharable image in which the module must be linked.

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
List of data element member name specifications
separated by commas. The pointers to these data element members are passed as arguments to the procedure. If not specified, base and directory are defaulted from PAR_DIR and BASE.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. string default: ”
If specified, the procedure is called only if the condition was true. The condition must be executed explicitely (either in a dynamic list or in the analysis program. If not specified, base and directory are defaulted from COND_DIR and BASE.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$DYNAMIC'
Default directory of the dynamic list

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DATA'
Default directory of the parameters

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Default directory of the condition

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Default node.

/MASTER

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: "
/MASTER	Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated.

/UPDATE

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: "
/UPDATE	The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/CHECK'
/CHECK	The dynamic list is checked for validity.
/NOCHECK	No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifier.

/KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/KEEP_MAP'
/NOKEEP_MAP	The data base is detached after the command.
/KEEP_MAP	The data base remains attached. This is recommended.

Function

Create a dynamic list entry for procedure call. Calls module from sharable image. Pointers to data elements specified in argument list are passed.

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1.Master procedures (PROCEDURE /MASTER)

2.Master pattern conditions (PATTERN /MASTER)

3.Master window conditions (WINDOW /MASTER)

4.Master function conditions (FUNCTION /MASTER)

5.Master polygon conditions (POLYGON /MASTER)

6.Master composed conditions (COMPOSED /MASTER)

7.Procedures (PROCEDURE)

8.Pattern conditions (PATTERN)

9.Window conditions (WINDOW)

10.Multi Window conditions (MULTI)

11.Function conditions (FUNCTION)

12.Polygon conditions (POLYGON)

13.Composed conditions (COMPOSED)

14.Spectrum accumulation indexed (INDEXED)

15.Spectrum accumulation (SPECTRUM)

16.Bit spectrum accumulation (BITSPECTRUM)

17.Scatter plots (SCATTER)

CREATE DYNAMIC ENTRY SCATTER

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

PURPOSE Create a scatter plot dynamic list entry

PARAMETERS

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

picture required string replace default: ”
Name specification of picture.

process required string replace default: ”
Name of display process.

condition string default: ”
Name of a condition. If specified the scatter points are sent only, if the condition was true.

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN ENTR SCATTER l [$PICTURE]p GN_SUSI___$DSP
```

Remarks

File name M$ACESC.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING
```
STS=M$ACESC(CV_dyn_list,CV_picture,
  CV_process,CV_condition,
  CV_dyn_dir,CV_base,CV_node,
  I_update,I_check,I_keep_map)
```

COMMAND
CREATE DYNAMIC ENTRY SCATTER dyn_list picture process condition dyn_dir base node
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of picture.

PROCESS

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
name of display process.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. string default: ”
If specified, the scatter points are sent only if the condition was true. The condition must be executed explicitely (either in a dynamic list or in the analysis program.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: ’$DYNAMIC’
Default directory of the dynamic list

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: ’DB’
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: ’E’
Default node.

/UPDATE

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: "
/UPDATE	The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/CHECK'
/CHECK	The dynamic list is checked for validity.
/NOCHECK	No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/KEEP_MAP'
/NOKEEP_MAP	The data base is detached after the command.
/KEEP_MAP	The data base remains attached. This is recommended.

Function

Create a dynamic list entry for scatter plot. This entry is created by the DISPL command. Several display processes may create several scatter plot entries.

CREATE DYNAMIC ENTRY SPECTRUM

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

PURPOSE Create a spectrum dynamic list entry

PARAMETERS

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

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

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

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

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

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

par_dir string global replace default: ’DATA’
Default directory

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

spec_dir	string global replace default: '$SPECTRUM' Default directory
base	string global replace default: 'DB' Default data base name
node	string global replace default: 'E' Default node name
/UPDATE	switch default: " Update dynamic list (then it becames valid for a running analysis immediately.
/[NO] CHECK	switch default: /CHECK Do dynamic list checking by attaching it
/[NO] KEEP_MAP	switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base.
Caller	MDBM, MGOODBM
Author	H.G.Essel

Example

```
CRE DYN EN SPECTRUM dlist [d]ener(1:10)
            PAR=[d]$event.e(1:10)
CRE DYN EN SPECTRUM dlist [d]ede(1:5)
            PAR=([d]$event.e(1:5),$event.de(1:5))
[d]is the directory specification
```

Remarks

File name	M$ACESP.PPL
Created by	GOO$DM:M$DMCMD

Description

CALLING

```
STS=M$ACESP(CV_dyn_list,CV_spectrum,
  CV_parameter,CV_increment,CV_condition,
  CV_dyn_dir,CV_par_dir,CV_cond_dir,CV_spec_dir,
  CV_base,CV_node,
  I_update,I_check,I_keep_map)
```

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

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Spectrum name specification. Directory is defaulted by SPEC_DIR. Array limits: [dir]name(ll:ul).

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications separated by commas. As may as spectrum dimension. Directory is defaulted by PAR_DIR.

INCREMENT

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Data element member specification used for increment. If not specified, 1 is used as increment. Directory is defaulted by PAR_DIR.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. string default: ”
If specified, the spectrum is filled only if the condition was true. The condition must be executed explicitely (either in a dynamic list or in the analysis program). Directory is defaulted by PAR_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$DYNAMIC'
Default directory of the dynamic list

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DATA'
Default directory of the parameters

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Default directory of the condition

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Default directory of the spectrum

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Default node.

/UPDATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”

/UPDATE The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

/NOCHECK No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

/KEEP_MAP The data base remains attached. This is recommended.

Function

Create a dynamic list entry for spectrum accumulation. Supports spectra of type BIN FIXED(31) or BIN FLOAT(24) with up to 2 dimensions. Coordinates can be BIN FIXED(31) or BIN FLOAT(24). Specified spectrum can be an array.

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1. **Master procedures (PROCEDURE /MASTER)**
2. **Master pattern conditions (PATTERN /MASTER)**
3. **Master window conditions (WINDOW /MASTER)**
4. **Master function conditions (FUNCTION /MASTER)**
5. **Master polygon conditions (POLYGON /MASTER)**
6. **Master composed conditions (COMPOSED /MASTER)**
7. **Procedures (PROCEDURE)**
8. **Pattern conditions (PATTERN)**
9. **Window conditions (WINDOW)**
10. **Multi Window conditions (MULTI)**
11. **Function conditions (FUNCTION)**
12. **Polygon conditions (POLYGON)**
13. **Composed conditions (COMPOSED)**
14. **Spectrum accumulation indexed (INDEXED)**
15. **Spectrum accumulation (SPECTRUM)**
16. **Bit spectrum accumulation (BITSPECTRUM)**
17. **Scatter plots (SCATTER)**

Arrays

Spectra or conditions may be arrays. In this case an index range must be specified. All additional data elements must be either scalar or indexed by the same range. Ranges are specified by (l:u).

Examples

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

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

```
  CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
                 PARA=[d]$event.e(2:4) CONDI=[d]m_window
  CRE DYN EN INDEXED dlist [d]ener(2:4)
                 PARA=[d]$event.e(1) INDEX=[d]m_window
```

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC ENTRY WINDOW

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

PURPOSE Create a window condition dynamic list entry

PARAMETERS

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

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

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

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

par_dir string global replace default: 'DATA'
Default directory

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN EN WINDOW dlist [d]e_recoil
            PAR=[d]$event.ener
  [d]is the directory specification
  CRE DYN ENTR WIN 1 [$CONDITION]W
                          PAR=[DATA]EVENT.RAW
```

Remarks

File name M$ACEWC.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING
```
STS=M$ACEWC(CV_dyn_list,CV_condition,
  CV_parameter,CV_dyn_dir,CV_par_dir,CV_cond_dir,
  CV_base,CV_node,
  I_master,I_update,I_check,I_keep_map)
```

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

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Dynamic list name specification. Node, base and directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
name specification of window condition. The directory is defaulted from COND_DIR. An array must be specified by [dir]name(ll:ul)

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications separated by commas. The number of elements must match the number of subwindows.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: ’$DYNAMIC’
Default directory of the dynamic list

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: ’DATA’
Default directory of the parameters

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$CONDITION'
Default directory of the condition

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Default node.

/MASTER

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”

/MASTER Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated. Master condition result bits are checked any time, even if the condition was already executed. So, if the same master condition is in two dynamic lists, both lists are skipped, if the condition was false.

/UPDATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”

/UPDATE The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/CHECK'
/CHECK	The dynamic list is checked for validity.
/NOCHECK	No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/KEEP_MAP'
/NOKEEP_MAP	The data base is detached after the command.
/KEEP_MAP	The data base remains attached. This is recommended.

Function

Create a dynamic list entry for window condition. Check specified members versus window limits. Up to 8 internal dimensions.

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1.Master procedures (PROCEDURE /MASTER)

2.Master pattern conditions (PATTERN /MASTER)

3.Master window conditions (WINDOW /MASTER)

4.Master function conditions (FUNCTION /MASTER)

5.Master polygon conditions (POLYGON /MASTER)

6.Master composed conditions (COMPOSED /MASTER)

7.Procedures (PROCEDURE)

8.Pattern conditions (PATTERN)

9.Window conditions (WINDOW)

10.Multi Window conditions (MULTI)

11.Function conditions (FUNCTION)

12.Polygon conditions (POLYGON)

13.Composed conditions (COMPOSED)

14.Spectrum accumulation indexed (INDEXED)

15.Spectrum accumulation (SPECTRUM)

16.Bit spectrum accumulation (BITSPECTRUM)

17.Scatter plots (SCATTER)

Arrays

Spectra or conditions may be arrays. In this case an index range must be specified. All additional data elements must be either scalar or indexed by the same range. Ranges are specified by (l:u).

Examples

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

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

```
  CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
```

PARA=[d]$event.e(2:4) CONDI=[d]m_window
CRE DYN EN INDEXED dlist [d]ener(2:4)
PARA=[d]$event.e(1) INDEX=[d]m_window
[d]is the directory specification
In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC LIST

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

PURPOSE Create a dynamic list in a Data Base

PARAMETERS

dyn_list Dynamic list name specification
required common replaced

entries Number of entries
replaced default:'100'

dyn_dir Default Directory
replaced common default:'$DYNAMIC'

pool Pool name
replaced default:'$DYNAMIC'

buffer Buffer size in bytes
replaced default:'0' (means 80*entries)

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

node Default node name
replaced common default:'E'

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

Caller M$DMCMD

Author H.G. Essel

File name M$ACRDL.PPL

Dataset -

EXAMPLE CRE DYN LIST L1 200
create dynamic list L1 in Pool $DYNAMIC with 200 entries

Remarks

REMARKS -

Description

CALLING STS=M$ACRDL(CV_DYN_LIST,L_ENTRIES,CV_DYN_DIR ,CV_POOL,L_BUFFER,CV_BASE,CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_DYN_LIST I Dynamic list name specification

L_ENTRIES Number of entries

CV_DYN_DIR Default Directory

CV_POOL Pool name

L_BUFFER Buffer size in bytes. 80 bytes/entry should be adequate and is used as default, if L_BUFFER is 0.

CV_BASE I Default Data Base name

CV_NODE I Default node name

I_KEEP_MAP I 1= Inhibit unmap of Data Base

REMARKS Module is an action routine.

FUNCTION Create a dynamic list in a Data Base. Each action in the dynamic list takes one entry slot except scatter plot entries. These need one more slot for each condition specified in the picture. Information under following key words:

Related_commands

CREATE DYNAMIC LIST listname

CREATE DYNAMIC ENTRY PROCEDURE listname entry

CREATE DYNAMIC ENTRY PATTERN listname entry

CREATE DYNAMIC ENTRY WINDOW listname entry

CREATE DYNAMIC ENTRY FUNCTION listname entry

CREATE DYNAMIC ENTRY COMPOSED listname entry

CREATE DYNAMIC ENTRY MULTIWINDOW listname entry

CREATE DYNAMIC ENTRY POLYGON listname entry

CREATE DYNAMIC ENTRY SPECTRUM listname entry

CREATE DYNAMIC ENTRY INDEXED listname entry

CREATE DYNAMIC ENTRY BIT listname entry

CREATE DYNAMIC ENTRY SCATTER istname entry

DELETE DYNAMIC ENTRY listname type entry

SHOW DYNAMIC LIST listname type

ATTACH DYNAMIC LIST listname (Analysis only)

DETACH DYNAMIC LIST listname (Analysis only)

SHO DYNAMIC ATTACHED listname type (Analysis only)

SET DYNAMIC LIST Listname type key value (Analysis only)

STOP DYNAMIC LIST Listname type (Analysis only)

START DYNAMIC LIST Listname type (Analysis only)

listname	Data element name of dynamic list
type	Type of dynamic list entry: PROCEDURE,FUNCTION,PATTERN,MULTI,WINDOW POLYGON,COMPOSED,SPECTRUM,INDEXED,BIT,SCATTER
entry	Name of dynamic list entry = name of data element

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1. **Master procedures (PROCEDURE /MASTER)**
2. **Master pattern conditions (PATTERN /MASTER)**
3. **Master window conditions (WINDOW /MASTER)**
4. **Master function conditions (FUNCTION /MASTER)**
5. **Master polygon conditions (POLYGON /MASTER)**
6. **Master composed conditions (COMPOSED /MASTER)**
7. **Procedures (PROCEDURE)**
8. **Pattern conditions (PATTERN)**
9. **Window conditions (WINDOW)**
10. **Multi Window conditions (MULTI)**
11. **Function conditions (FUNCTION)**
12. **Polygon conditions (POLYGON)**
13. **Composed conditions (COMPOSED)**
14. **Spectrum accumulation indexed (INDEXED)**
15. **Spectrum accumulation (SPECTRUM)**
16. **Bit spectrum accumulation (BITSPECTRUM)**
17. **Scatter plots (SCATTER)**

Arrays

Spectra or conditions may be arrays. In this case an index range must be specified. All additional data elements must be either scalar or indexed by the same range. Ranges are specified by (l:u).

Examples

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

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

```
  CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
                 PARA=[d]$event.e(2:4) CONDI=[d]m_window
  CRE DYN EN INDEXED dlist [d]ener(2:4)
                 PARA=[d]$event.e(1) INDEX=[d]m_window
```

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE ELEMENT

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

PURPOSE Create a Data Element in a Data Base

PARAMETERS

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

pool Pool name
required common replaced default

typename Type name
required

refer List of refer values
optional

datalength Size in bytes
replaced default:'0'

cluster Cluster size in bytes
replaced default:'16'

queuehead Queue header specification
optional

dir Default Directory
common replaced default:'DATA'

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

node Default node name
common replaced default:'E'

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

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

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

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

Caller M$DMCMD

Author M. Richter

File name M$ACRDE.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ACRDE(CV_NAME,CV_POOL,CV_TYPENAME,
LA_REFER,I_DATALENGTH,L_CLUSTER,CV_QUEUEHEAD,
CV_DIR,CV_BASE,CV_NODE,I_PROTECT,
I_REPLACE,I_KEEP_MAP)

ARGUMENTS

CV_NAME I Data Element name (with index)
CHAR(*) VAR

CV_POOL I Pool name
CHAR(*) VAR

CV_TYPENAME	I Type name CHAR(*) VAR
LA_REFER	I Refer values (*) BIN FIXED(31)
I_DATALENGTH	I Size in bytes BIN FIXED(15)
L_CLUSTER	I Cluster size in bytes BIN FIXED(31)
CV_QUEUEHEAD	I Queue header specification CHAR(*) VAR
CV_DIR	I Default Directory CHAR(*) VAR
CV_BASE	I Default Data Base name CHAR(*) VAR
CV_NODE	I Default node name CHAR(*) VAR
I_PROTECT	I Protect deletion of new Data Element BIN FIXED(15)
I_REPLACE	I Replace old Data Element BIN FIXED(15)
I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
FUNCTION	Create a Data Element in a Data Base
REMARKS	Module is an action routine.
CALLING	STS=M$ACRDE(CV_NAME,CV_POOL,CV_TYPENAME, LA_REFER,I_DATALENGTH,L_CLUSTER,CV_QUEUEHEAD, CV_DIR,CV_BASE,CV_NODE,I_PROTECT, I_REPLACE,I_KEEP_MAP)
EXAMPLE	STS$VALUE=M$ACRDE('[PARAM]CALIB(1:10)','PARA', 'PARTYP',0,256,16,",",",",0,0,1); create the Data Element name array CALIB with 10 members in Directory PARAM. The data of 256 bytes each will be in the Pool PARA. The Data type is PARTYP. PARAM, PARA, and PARTYP must exist already. The cluster size is 16 bytes.

CREATE ENVIRONMENT

CREATE ENVIRONMENT environment

PURPOSE Create a GOOSY environment

PARAMETERS

environment name of the environment to be created. Must have 1 to 4 characters. If omitted, the 3 trailing hexadecimal digits of the process identification are used (which are by construction processor wide unique).

FUNCTION This command creates a GOOSY Environment. An environment is the context, in which several sessions each with several processes may execute. Note, that processes communicate only with processes of the same environment. The issuing process will become the master process of the new environment and should not be logged out before all sessions attached to this environment are logged off or deleted. The issuing process will change its name into

GN_<name>

and create the supervisor ($SVR) process.

EXAMPLE CRE ENV TEST

Action rout. U$TCCRE

Author Walter F.J. Mueller

Remarks

File name U$TCCRE.PPL

Dataset -

REMARKS This command is 'more or less' equivalent to the following sequence of DCL and GOOSY commands:

SET PROC /NAME=GN_<name>
GOOSY CREATE PROCESS GOO$EXE:MGOOSV0 $SVR

The expression GOO$EXE:MGOOSV0 is accessed via a logical name SV0.

Description

CALLING @CALL U$TCCRE(C_NAME)

ARGUMENTS

C_NAME CHAR(*) VAR [INPUT]
Name of the environment to be created.

FUNCTION This procedure creates a GOOSY environment. For a detailed description look in the command part.

REMARKS -

EXAMPLE @CALL U$TCCRE('ANL1');

CREATE LINK

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

PURPOSE Create a link between two Data Elements

PARAMETERS

link_from Source Data Element name specification
required, common default

link_to Target Data Element name specification
required, common default

dir Default Directory
required, common default

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/MULTIPLE Multiple identical links allowed

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

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

Caller M$DMCMD

Author M. Richter

File name M$ACRLI.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ACRLI(CV_LINK_FROM,CV_LINK_TO,
CV_DIR,CV_BASE,CV_NODE,I_MULTIPLE,I_KEEP_MAP)

ARGUMENTS

CV_LINK_FROM I Source Data Element name specification
CHAR(*) VAR

CV_LINK_TO I Target Data Element name specification
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_MULTIPLE I Multiple identical links allowed
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Create a link between two Data Elements

REMARKS Module is an action routine.

EXAMPLE -

CREATE OVERLAY

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

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

PARAMETERS

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

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

Caller MDBM,MGOODBM,E$DECMD

Author W. Spreng

Example

1.) CREATE OVERLAY pic 2 spec DYNSHIFT=1.5
In picture "PIC" spectrum "SPEC" should be overlayed in frame 2 with a dynamic shift of 1.5.

2.) CREATE OVERLAY pic 1 xpara=par_x ypara=par_y

The scatterplot parameters par_x,par_y are additionally added to frame 1 in picture "PIC"

Remarks

File name D$DEFOV.PPL

Created by E$DECMD.ppl

REMARKS If a dynamic y-shift is produced a dynamic y-scaling axis could be generated specifying the scaling factor in the MODIFY FRAME SPECTRUM command.

Description

CALLING
```
STS=D$DEFOV(CV_picture,L_frame,CV_spectrum,CV_xparam
            CV_yparam,RA_trans,R_dynshift,L_entries
            CV_node,CV_base,CV_pic_dir,CV_spec_dir,
            CV_par_dir,LA_dim,B_mask)
```

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

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name specification for the picture for which an overlay should be defined.

FRAME

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable

Number of the frame for which the overlays should be defined.

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. String

Name of spectrum which should be added to the specified spectrum frame. The spectrum could be one or two dimensional.

XPARAM,YPARAM

Routine arg. Input CHAR(*) VAR

Command par. String

The X and Y-parameter for additional scatterplot correlations, which should be displayed in the specified scatter frame.

TRANS

Routine arg. Input (*) BIN FLOAT(24)

Command par. Real array default=(1.0,0.0,1.0,0.0)

Defines a linear transformation for overlayed spectra in the horizontal (x) and vertical (y) axis:
(xfactor,xoffset,yfactor,yoffset)
This transformation is applied to one and two dimensional spectra.

DYNSHIFT

Routine arg. Input BIN FLOAT(24)

Command par. Real

If several spectra should be overlayed and shifted into Y-direction it is useful to make this shift dynamicly, if the spectrum contents are changed by aquisition. This could be achieved specifing 'dynshift'. The offset is then determined as :

Y-offset = dynshift * (Maximum channel contents)

ENTRIES

Routine arg. Input BIN FIXED(31)

Command par. Integer default = 1

Specifies how many entries in the dataelement for overlays should be created for each frame. One overlay takes one slot.

This parameter can be used to prevent the replacement of the overlay dataelements, if not enough slots are availible to create a new overlay, because with each replacement free fragments are generated in your database which may not be used futheron. Therefore it is recommended to reduce the number of dataelement replacements by increasing this parameter.

The number of overlays is not limited by this parameter, you can generate more overlays as preliminary specified, if necessary the existing dataelement will be increased.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=*

Default node name

BASE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=DB

Default Data Base name

PIC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=$PICTURE
Default Picture Directory

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=$SPECTRUM
Default spectrum Directory

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=DATA
Default parameter Directory

/KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch negatgable default=/KEEP_MAP
Switch to prevent the dettachment of the Data Base

/NOKEEP_MAP	Deattach Data Dase.
/KEEP_MAP	Keep the whole Data Base mapping context

Function

Several overlays of spectra and/or scatterplot parameter can be defined for each frame of an existing picture. The information about the overlays is kept in additional Data Elements, which are different for spectra and scatterplots. If for 'picture' such an overlay Data Element is created for each frame 'entries' slots are reserved for overlayed spectra or scatterplots, depending on type of the specified frame.

It is obvious that the frame type (spectrum or scatter) must match the specified overlay. Futhermore it is imposssible to generate an overlay of a one and a two dimensional spectrum.

For the overlay of spectra a linear transformation in x- and y-direction can be defined with the help of the transformation parameter:

(xfactor,xoffset,yfactor,yoffset)

If the spectrum contents are changed by the acquisition, overlayed one dimensional spectra can be dynamicly shifted in y-direction, using the parameter "dynshift". The offset in Y-direction is then determined by:

Y-offset = R_dynshift * (Maximum channel contents)

To define a dynamic scaling axis specify the scaling factor in the MODIFY FRAME SPECTRUM command.

The created overlays are not displayed by default with the DISPLAY PICTURE command. To have a look at them give the OVERLAY command without specifying any parameter!

CREATE PICTURE

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

PURPOSE Create a picture Data Element

PARAMETERS

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

/[NO] PROMPT Enter interactive prompting menu:

/NOPROMPT	no prompting
/PROMPT	prompting is performed (DEFAULT)

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

/NOKEEP_MAP	Data Base will be deattached

/KEEP_MAP Keep Data Base (DEFAULT)

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

Example

1.) CREATE PICTURE a 4/noprompt
Picture "A" is created with 4-Frames and no prompting is performed.
2.) CREATE PICTURE b
Picture "B" with 1-Frame is created and the automatic prompting menue will be invoked.

Remarks

File name	GOO$DISP:D$CRPI.PPL
Created by	GOO$DISP:D$DSPCM.PPL

Description

CALLING

```
STS=D$CRPI(CV_pic_name,L_frames,CV_condition,
           CV_object CV_pic_dir,CV_pic_pool,
           CV_COND_DIR,cv_par_dir,CV_base,
           CV_node,CV_prompt,I_keep_map,B_mask)
```

COMMAND

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

NAME

Routine arg.	CHAR(*) VAR
Command par.	String required
	Name of the picture which should be created.

FRAMES

Routine arg. BIN FIXED(31)

Command par. Integer default=1

Number of picture frames. Each frames contains one spectrum or one scatterplot. The maximum number of possible frames is 64.

CONDITION

Routine arg. CHAR(*) VAR

Command par. String

General picture condition for all scatterplots in this picture. The scatter data are show if the result flag of the condition is true.

OBJECT

Routine arg. CHAR(*) VAR

Command par. String

Condition object for the general picture condition. If this parameter is specified the condition is checked against the object. This will destroy the result of an earlier check of the same condition!

PIC_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$PICTURE

Default picture Directory.

PIC_POOL

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$PIC_POOL

Default picture Pool.

COND_DI

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$CONDITION

Default condition Directory.

PAR_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=DATA

Default parameter Directory.

BASE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

PROMPT

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch default = /PROMPT

If set an interactive prompting menue is envoked in which the parameter for each picture frame can be set.

Set /NOPROMPT to prevent this; e.g if pictures are created in command procedures!

KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch default = /KEEP_MAP

If set the mapping context will be kept. If /NOKEEP_MAP specified the Data Base will be dettached after the command completion.

Function

The procedure creates a picture Data Element in the Data Base with the specified number of frames.

The global picture condition is checked for each scatterplot defined in that picture. If the condition flag, which is set in the analysis or in the dynamic list, is true the scatterdata are displayed. If additionally a condition object is specified the global picture condition is checked against this object, but the result flag of an earlier condition check is no longer valid.

If /PROMPT is defined an interactive editing menu is envoked where for each frame all parameters are prompted. It is recommended to use this mode in an interactive session to be shure that all frames are completely specified! The editing menu has the advantage that the default display modes could be displayed.

If /NOPROMPT is specified (e.g. this is necessary for BATCH jobs) the frames has to be specified with the MODIFY FRAME command. In that case take care that all frames will be modified!

CREATE POLYGON

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

PURPOSE Create a polygon in a Data Base

PARAMETERS

polygon polygon name specification
required common replaced

points Number of points
replaced default:'1'

poly_dir Default Directory
replaced common default:'$POLYGON'

poly_pool Pool name
replaced default:'$PIC_POOL'

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

node Default node name
replaced common default:'E'

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

Caller M$DECMD

Author H.G. Essel

File name E$ACRPO.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=E$ACRPO(CV_POLYGON,L_POINTS,
CV_POLY_DIR,CV_POOL,CV_BASE,
CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_POLYGON Polygone name specification

L_POINTS Number of points

CV_POLY_DIR Default Directory

CV_POOL Pool name

CV_BASE Default Data Base name

CV_NODE Default node name

I_KEEP_MAP Inhibit unmap of Data Base

REMARKS Module is an action routine.

FUNCTION Create a polygon in a Data Base. The size of the polygon specified by L_POINTS is dynamically enlarged if necessary.

Related_commands

CREATE POLYGON polygon

SET POLYGON polygon

DELETE POLYGON polygon

SHOW POLYGON polygon

CREATE POOL

CREATE POOL pool areaminsize base
/[NO]KEEP_MAP

PURPOSE Create a Pool in a Data Base

PARAMETERS

pool Pool name
common default:'DATA'

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

base data
required, common default

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

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

Caller M$DMCMD

Author M. Richter

File name M$ACRPO.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ACRPO(CV_POOL,L_AREAMINSIZE,CV_BASE, I_KEEP_MAP)

ARGUMENTS

CV_POOL I Pool name
CHAR(*) VAR

L_AREAMINSIZE I Minimum size of Areas in bytes
BIN FIXED(31)

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Create a Pool in a Data Base

REMARKS Module is an action routine.

EXAMPLE -

CREATE PROCESS

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

PURPOSE Create a component in GOOSY environment.

PARAMETERS

image string
File name of program to be started.

type string
Name used together with the environment name for process name:
GN_env__type.
Maximum of 4 characters.

input string default=SYS$INPUT
Input device.

output string default=SYS$OUTPUT
Output device.

command string
DCL command line to be executed before the image started.

priority integer default=3
Priority of started image.

/DEBUG switch
Image is started with debugger

/DUMP switch
Image is dumped to disk file in case of errors.

/RESTART	switch Image is restarted in case of errors.
EXAMPLE	GOOSY> CRE PROC MYANAL ANA1
Action rout.	U$TCCRP
Author	Walter F.J. Mueller

Remarks

File name	U$TCCRP.PPL
Dataset	-
REMARKS	-

Description

CALLING

```
@CALL U$TCCRP(C_IMAG,C_TYPE,C_INPUT,C_OUTPUT,
              C_COMMAND,C_PRIORITY,
              C_DEBUG,C_RESTART,C_DUMP);
```

ARGUMENTS

C_IMAG	Name of the image to be started.
C_TYPE	Type name of the subprocess to be created.
C_INPUT	Equivalence for SYS$INPUT.
C_OUTPUT	Equivalence for SYS$OUTPUT.
C_COMMAND	Command to be executed before image start.
C_PRIORITY	Priority of spawned process.
C_DEBUG	Run image under DEBUG
C_RESTART	Enable image restart after an abort.
C_DUMP	Enable image dump after an exception.

FUNCTION	-
REMARKS	-
EXAMPLE	-

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.

CREATE SESSION

CREATE SESSION environment session

PURPOSE	Create a GOOSY session
PARAMETERS	
environment	Name of the parent environment.
session	Name of the session to be created.
EXAMPLE	CRE SES TEST
Action rout.	U$TCCRS
Author	Walter F.J. Mueller

Remarks

File name	U$TCCRS.PPL
Dataset	-
REMARKS	-

Description

CALLING	@CALL U$TCCRS(C_ENV,C_SES)
ARGUMENTS	
C_ENV	Name of the parent environment.
C_SES	Name of the session to be created.
FUNCTION	-
REMARKS	-
EXAMPLE	-

CREATE SPECTRUM

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

PURPOSE create a spectrum

PARAMETERS

name name of spectrum
required

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

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

binsize size of bins
replace default:'1'

spec_dir default Directory
replace default='$SPECTRUM'

spec_pool default pool
replace default='$SPEC_POOL'

base default Data Base
replace default='DB'

node default node
replace default='E'

maxspec maximum number of spectra
default=1024

branch Number of CAMAC branch
replace default=0

crate Number of CAMAC crate on branch
replace default=1

station Number of MR2000 in crate
replace default=2

offset Start address of data in MR2000
replace default=0

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

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

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

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

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

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

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

/ANALOG/DIGITAL /ANALOG for floating point accumulation, /DIGITAL for accumulation of integers.
The difference is that in analog spectra a bin represents an intervall, but in digital spectra a number (or a range of numbers). Therefore digital

spectra have one more bin for the last number.
For analog spectra the upper limit is excluded from
last bin. Digital spectra are displayed shifted by
.5 to the left to center the numbers ticks.
replace default:'/ANALOG'

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

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

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$CRESP.PPL

Dataset -

Remarks

REMARKS action routine

Description

CALLING

```
STS=E$CRESP(CV_SPEC,CV_TYPE,CV_LIMITS
      ,CV_BINSIZE,CV_DIR,CV_POOL,CV_BASE,CV_NODE
      ,L_TAB,I_BRANCH,I_CRATE,I_STATION,L_OFFSET
      ,I_CAMAC,CV_DOCU,CV_MEANVALUES,CV_SQW,CV_ERRV
      ,CV_ERRH,CV_VARBINS,CV_DYNAMICSTATIC,
      ,CV_ANALOGDIGITAL,I_KEEP_MAP,B_MASK)
```

ARGUMENTS

CV_SPEC name of spectrum
CHAR(*) VAR
Input

CV_TYPE data type of spectrum may be
N - spec contains no data
S - spec contains scatter data (1 bit per channel)
H - spec contains one byte integer data

I - two bytes integer data
L - four byte integer data
R - four byte floating point data
D - eight byte (double precision) float.pt. data
CHAR(*) VAR
Input

CV_LIMITS limits of the spectrum , limits given in pairs of two per dimension e.g. (0,10,1,100) defines a spectrum where x is between 0 and 10, and y betweeen 1 and 100
CHAR(*) VAR
Input

CV_BINSIZE size of the bin, may be a floating point number , default is 1.0
BIN FLOAT(16)
Input

CV_DIR default Directory
CHAR(*) VAR
Input

CV_POOL default pool
CHAR(*) VAR
Input

CV_BASE default Data Base
CHAR(*) VAR
Input

CV_NODE default node
CHAR(*) VAR
Input

L_TAB Maximum number of spectra.
BIN FIXED(31)
Input
Default=1024

I_BRANCH Number of CAMAC branch for CAMAC spectra.
BIN FIXED(15)
Input
Default=0

I_CRATE Number of CAMAC crate for CAMAC spectra.
BIN FIXED(15)

Input
Default=0

I_STATION Station of MR2000 in CAMAC crate for CAMAC spectra.
BIN FIXED(15)
Input
Default=0

L_OFFSET Start address of CAMAC spectrum in MR2000.
BIN FIXED(31)
Input
Default=0

I_CAMAC If 1, spectrum is CAMAC spectrum.
BIN FIXED(15)
Input
Default=0

CV_DOCU '/DOCU' if documentation and history is wanted
CHAR(*) VAR
Input

CV_MEANVALUES '/MEANVALUES' if mean values per bin are wanted
CHAR(*) VAR
Input

CV_SQW '/CV_SQW' if squares of weights are wanted to be hel
CHAR(*) VAR
Input

CV_ERRV '/ERRV' if vertical errors are wanted to be held
CHAR(*) VAR
Input

CV_ERRH '/ERRH' if horizontal errors are wanted to be held
CHAR(*) VAR
Input

CV_VARBINS '/VARBINS' if sjpectrum has variable bins ehich are then held with the spectrum
CHAR(*) VAR
Input

CV_DYNAMICSTATIC '/DYNAMIC' if all attributes of the spectrum ma be modified

'/STATIC' data type and numbers of dimensions can not be modified
CHAR(*) VAR
Input

CV_ANALOGDIGITAL '/ANALOG' if input is float.pt.number, '/DIGITAL' if input is integer
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

FUNCTION A spectrum is created in the data management including all queued data elements . All structures are initialized according to the attributes given by the command.

REMARKS Action routine

EXAMPLE -

CREATE TABLE CONDITION

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

PURPOSE create condition bit table

PARAMETERS

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

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

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

pool string replace default: '$COND_POOL'
default pool

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

node string global replace default: 'E'
default node

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE TAB COND ENTRIES=512
```

Remarks

File name GOO$DE:E$ACRCT.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

STS=E$ACRCT(CV_name,L_entries,
CV_directory,CV_pool,CV_base,CV_node,
I_keep_map)

COMMAND

CREATE TABLE CONDITION name entries
directory pool base node
/[NO]KEEP
Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'ANCO'
Name specification of the table. This has the standard GOOSY format base:[directory]name. Base and directory are defaulted by the explicit parameters. The values here specified are not replaced as defaults!

ENTRIES

Routine arg. BIN FIXED(31)

Command par. integer replace default: 1024
This defines the maximum number of conditions which can be created. Each condition member of an array occupies one entry.

DIRECTORY

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$ANLTABS'
default directory

POOL

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$COND_POOL'
default pool

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
default base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
default node

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

/NOKEEP_MAP unmap Data Base In a procedure call this argument should be always 1 to prevent an unmap of the data base.

Function

Create a condition table. In this table the bits for freeze, execution, preset and result are allocated.

CREATE TABLE SPECTRUM

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

PURPOSE create spectrum bit table

PARAMETERS

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

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

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

pool string replace default: '$SPEC_POOL'
default pool

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

node string global replace default: 'E'
default node

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE TAB SPEC ENTRIES=512
```

Remarks

File name GOO$DE:E$ACRST.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$ACRST(CV_name,L_entries,
        CV_directory,CV_pool,CV_base,CV_node,
        I_keep_map)
```

COMMAND

CREATE TABLE SPECTRUM name entries
  directory pool base node
  /[NO]KEEP
Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'ANSP'
  Name specification of the table. This has the standard GOOSY format base:[directory]name. Base and directory are defaulted by the explicit parameters. The values here specified are not replaced as defaults!

ENTRIES

Routine arg. BIN FIXED(31)

Command par. integer replace default: 1024
  This defines the maximum number of spectra which can be created. Each spectrum member of an array occupies one entry.

DIRECTORY

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$ANLTABS'
  default directory

POOL

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$SPEC_POOL'
default pool

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
default base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
default node

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

/NOKEEP_MAP unmap Data Base In a procedure call this argument should be always 1 to prevent an unmap of the data base.

Function

Create spectrum table. In this table the bits for freeze, execution are allocated.

CREATE TYPE

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

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

PARAMETERS

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

base Data Base name
required, common default

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

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

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

callers file PRIVAT.TXT.

Caller M$DMCMD

Author M. Richter

File name M$ACRTY.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ACRTY(CV_TYPEFILENAME,CV_BASE,CV_COMPREF,
I_KEEP_MAP)

ARGUMENTS

CV_TYPEFILENAME I Type descriptor file name (def ext = .TXT)
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

CV_COMPREF I Compile and/or write REFER file
/COMPILE : performs a test compilation of the Type structure
/REFERFILE : Creates a file with the REFER values of Type structure. The name will be identical to the Type structure name, but the first character will be replaced aby an 'R'.
/COMPREF : performs both compilation and REFER file creation.
/NOCOMPNOREF : performs neither a test compilation nor a creation of the REFER file. This is the default.
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Create a Type descriptor

REMARKS Module is an action routine.

EXAMPLE -

DEALLOCATE DEVICE

DEALLOCATE DEVICE device

PURPOSE	Deallocate a graphical device
PARAMETERS	
device	Logical device name or * for all devices. TT for login terminal
Caller	MDISP,MGOODISP,D$DSPCM
Action rout.	D$DEALL
Author	W. SPRENG

Examples

1. DEALLOCATE DEVICE txnn
   Device txnn will be freed
2. DEALLOCATE DEVICE *
   All allocated devices are freed.

Remarks

File name	D$DEALL.PPL
created by	GOO$DISP:D$DSPCM.PPL

Description

CALLING	STS=D$DEALL(CV_device)
COMMAND	DEALLOCATE DEVICE device

DEVICE

Routine arg. CHAR(*) VAR

Command par. String required

Logical or physical device name or * for all devices. For plotter use the file name of the generated plotfile!

Function

Deallocate graphical devices. The following actions are performed:

- The device is disconnected from the GKS-session
- The entry in the main device table is deleted
- If there are no devices of the same type allocated, the device description table is deleted
- If the specified device is the GOOSY-main device all actions will be performed, but additionally it will be checked if another device of the same type exists which could be used as GOOSY-main device. If this is not the case a device of the category INPUT/OUTPUT will be searched and will be allocated as the GOOSY-main device for all following actions.

If it is not possible to find an other main main device the request to deallocate the device will be canceled. To deallocate this device you have to deallocate all devices or you must allocate a new device which could be used as a GOOSY-main device! This is necessary to be sure that always a main device exists.

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

DECALIBRATE SPECTRUM

DECALIBRATE SPECTRUM spectrum spec_dir base node

PURPOSE	Disconnect a spectrum from its calibration.
PARAMETERS	
spectrum	Name of spectrum
spec_dir	Directory for spectrum Data Element.
base	Data Base name
node	Node name for Data Base file
Author	W. Spreng
Caller	MDBM,MGOODBM

Remarks

Created by	GOO$DE:E$DECMD.PPL
File name	GOO$DE:E$DECAL.PPL

Description

CALLING	STS=E$DECAL(CV_spectrum,CV_spec_dir,CV_base,CV_node)
COMMAND	DECALIBRATE SPECTRUM spectrum spec_dir base node

SPECTRUM

Routine arg.	Input CHAR(*) VAR
Command par.	String required Name specification of the spectrum which should be disconnected from its calibration. No wildcards in the spectrum specification are supported.

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$SPECTRUM
Default spectrum Directory.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB
Default Data Base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*
Default node name.

Function

FUNCTION The specified spectrum is disconnected from any calibration. No wild-cards in the spectrum specification are supported.

DECOMPRESS BASE

DECOMPRESS BASE file base basefile
/MOUNT

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

PARAMETERS

file required string default: ”
Name of input file containing compressed base.

base required string default: ”
name of the data base to be restored.

basefile required string default: ”
name of data base file.

/MOUNT switch default:
Mount restored data base.

Caller MDBCOPY

Author H.G.Essel

Example

MDBCOPY DECOMP BASE db.cmp db db.sec

Remarks

File name GOO$DM:M$ADECB.PPL

Created by GOO$DM:M$DMCMD.PPL

Description

CALLING STS=M$ADECB(CV_file,CV_base,CV_basefile,I_mount)

COMMAND DECOMPRESS BASE file base basefile
/MOUNT
Argument /parameter description:

FILE

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”
File name of input file containig compressed
data base. This file is generated by command
MDBCOPY COMPRESS BASE base basefile file
Default file type is .CSEC.

BASE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of the data base to be restored.

BASEFILE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
File name of the data base.

/MOUNT

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default:
The restored data base is mounted.

Function

Decompress and copy a bata base. The output file of the data base must not exist. The data base to be restored must not be mounted. The base will be not mounted untill /MOUNT is specified.

DEFINE DISPLAY HEADER

DEFINE DISPLAY HEADER string

PURPOSE	Define display header line.
PARAMETERS	
string	String to be displayed on top of picture. The string is defined as user mode logical name in the process table. It remains valid until program exit or redefinition.
Caller	MDISP,MGOODISP,D$DSPCM
Action rout.	D$SDHD
Author	H.G.Essel

Examples

DEF SP HEAD "This is picture xx"

Remarks

File name	D$SDHD.PPL
created by	GOO$DISP:D$DSPCM.PPL

Description

CALLING	STS=D$SDHD(CV_string)
COMMAND	DEFINE DISPLAY HEADER string

STRING

Routine arg.	CHAR(*) VAR
Command par.	String required String to be displayed on top of picture.

Function

Define string to be displayed on top of picture. The string definition is valid as long as the program is up.

DEFINE DISPLAY PICTURE

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

PURPOSE Set Spectrum parameter

PARAMETERS

update Update time interval.

refresh Refresh time interval.
********** not yet implemented *******

SCALE Scaling mode for Y- or Z-axis for one or two dimensional spectra.

/LIN	Display axis in a linear scale
/LOG	Display axis in a logarithmic scale
/SQRT	Display axis in a square root scale

X_SCALE	Scaling mode for X-axis	
	/XLIN	Display axis in a linear scale
	/XLOG	Display axis in a logarithmic scale
	/XSQRT	Display axis in a square root scale
Y_SCALE	Scaling mode for Y-axis	
	/YLIN	Display axis in a linear scale
	/YLOG	Display axis in a logarithmic scale
	/YSQRT	Display axis in a square root scale
Z_SCALE	Scaling mode for Z-axis	
	/ZLIN	Display axis in a linear scale
	/ZLOG	Display axis in a logarithmic scale
	/ZSQRT	Display axis in a square root scale
/[NO] WINDOW	Window switch ********** not yet implemented *******	
/[NO] LIFE	Activate life display of spectra. ********** not yet implemented *******	
/[NO] SMOOTH	Display the spectrum with smooth binning.	
/[NO] LETTER	Display lettering on axis	
/[NO] NUMBER	Display numbering on axis	
CALIB	Perform calibration	
	/CALAX	Calibrate axis.
	/CALSPEC	Calibrate spectrum data.
	/NOCAL	no calibration is performed.
/[NO] CHANNELS	Display channel numbers	
RANGE	Select display window	
	/FULL	Use full spectrum range.
	/LAST	Use last displayed range.
	/ACTUAL	Use actual displayed range.
SCALE_RANGE	Scaling method	

	/AUTOSCALE	Autoscaling is performed
	/SCALE	Scaling as in MODIFY FRAME command
STYLE	Display mode of spectra. 1. One dimensional spectra	
	/HISTO	histograms are generated
	/VECTOR	spectrum contents are connected with olylines
	/MARKER	spectrum contents are signed with markers
	2. Two dimensional spectra	
	/CONTOUR	Contour lines are displayed.
	/CLUSTER	spectrum contents in cluster mode
	/ISO	pseudo 3D isometric plot is generated.
	/SCATTER	Simulate scatter plot data.

Caller MDISP,MGOODISP,D$DSPCM

Author W. Spreng

Example

DEFINE DISPLAY PICTURE /LOG/SMOOTH/VECTOR

The spectra are displayed with a logarithmic scaling axis (Y for one and Z for two dimensional spectra).The display bins are smoothed and the one dimensional spectra are displayed in VECTOR mode. If the two dimensional spectra should be displayed in CLUSTER mode specify additionally:

DEFINE DISPLAY PICTURE/CLUSTER

Remarks

File name D$SPPAR.PPL

Created by GOO$DISP:D$DSPCM.PPL

Description

CALLING

```
STS=D$SPPAR(L_update,L_refresh,
  CV_scale,CV_Xscale,CV_Yscale,CV_Zscale,I_active,
  I_error,I_window,I_life,I_rotate,I_smooth,
  CV_letter,CV_number,CV_calib,CV_channels,CV_range,
  CV_scale_range,CV_style,B_mask)
```

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

UPDATE

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable

Update interval. If <= 0 , no update.

If > 0 every 'update' seconds the whole picture will be updated with the new channel contents of each spectrum.

REFRESH

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable

Refresh intervall. If <=0 , no refresh.

If > 0 all spectra in the picture will be deleted and redrawn every 'refresh' seconds. The scatter plots will be stored on the display and not deleted if a device with selective erase operation is used (e.g. Tektronix 4115) if not the whole spectrum is deleted and the collected data in scatter plots are lost. Therefore be careful using this parameter.

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

SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Set the display mode on the scaling axis. You can select between three modes:

/LIN	Display the linear count rate.
/LOG	Display the spectrum contents in logarithmic mode.
/SQRT	Display the square root of the spectrum contents.

X_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Display mode of X-axis. You can select between three modes:

/XLIN	Display the axis linear.
/XLOG	Display the axis in logarithmic mode.
/XSQRT	Display the square root of axis.

Y_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Display mode of Y-axis. You can select between three modes:

/YLIN	Display the axis linear.
/YLOG	Display the axis in logarithmic mode.
/YSQRT	Display the square root of axis.

Z_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Display mode of Z-axis. You can select between three modes:

/ZLIN	Display the axis linear.
/ZLOG	Display the axis in logarithmic mode.
/ZSQRT	Display the square root of axis.

ACTIVE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable default=/ACTIVE

Activate or deactivate the global display parameter for spectra. Possible inputs are:

/ACTIVATE	activate display parameter. This is the default. The parameter can be deactivated any time by DEFINE DISPLAY SPECTRUM/NOACTIVE or for a single display by DISPLAY SPECTRUM/NOGLOBAL
/NOACTIVE	deactivate the display parameter. They can be activated later by DEFINE DISPLAY SPECTRUM /ACTIVE or for a single display by DISPLAY SPECTRUM/GLOBAL

ERROR

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable

Display statistical errors at each channel of a one dimensional spectrum.

/ERROR	Statistical errors (the square root of the count rate) are displayed.
/NOERROR	No errors are displayed

WINDOW

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable

********** Not yet implemented **********

LIFE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable

********** Not yet implemented **********

ROTATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable

********** Not yet implemented **********

SMOOTH

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable

Display the spectrum with smooth binning, the mean values of the channel contents about the display binsize will be shown. The effect is that the spectrum looks smoother, but the displayed spectrum contents could be fractional numbers.

/SMOOTH	Display the spectra with smooth binning
/NOSMOOTH	The minimum and maximum contents of the display bins are shown.

LETTER

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Activate or deactivate the lettering on the axis.

/LETTER	The lettering is displayed.
/NOLETTER	The lettering is not displayed.

NUMBER

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Activate or deactivate the numbering on the axis.

/NUMBER	The numbering is displayed.
/NONUMBER	The numbering is not displayed.

CALIB

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Display the spectrum in calibrated units. To do this the displayed spectra has to be connected to an existing calibration. Different calibration modes are available:

/CALAX	The axis is drawn in calibrated units and the spectrum in uncalibrated units. Therefore the distance between two subsequent tics varies.
/CALSPEC	The spectrum is drawn in calibrated units. Then the width of the displayed spectrum bins varies.
/NOCAL	No calibration is performed to the displayed spectra.

In any case the axis with uncalibrated units is displayed, too. To prevent this specify the /NOCHANNELS switch!

CHANNELS

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Activate or deactivate the display of an axis with the original channel units.

/CHANNELS	Display an axis with original units.
/NOCHANNEL	The axis with original units is not displayed.

RANGE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Specify one of the available display windows:

/FULL	The full spectrum range is displayed.
/LAST	The last displayed range is used. This window is indicated be a "#".
/ACTUAL	Use the actual displayed range.

The specified window is used for the display of the spectra.

SCALE_RANGE

Routine arg. Input CHAR(*) VAR

Command par. Set default=/SCALE

Defines the range of the scaling axis.

/AUTOSCALE	Perform autoscaling
/SCALE	Use the scaling range defined in the MODIFY FRAME command.

STYLE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Defines the display style of the spectrum data. For one and two dimensional spectra different styles are implemented.

One dimensional spectra:

/HISTO	Histograms are generated
/VECTOR	The spectrum contents are connected by polylines.
/MARKER	The spectrum contents are indicated by markers.

For two dimensional spectra:

/CONTOUR	Contour lines are displayed.
/CLUSTER	The spectrum count rates are indicated by clusters of a variable size and colour.
/ISO	A 3D isometric plot is generated.
/SCATTER	Scatter plot data are simulated. In each the countrate is indicated by a number of points, randomly distributed in the bin.

Function

Define the global display parameters for the display of pictures. Subsequent definitions of global parameter are cumulative and do not distroy the earlier settings. Therfore at any time additional parameters can be set or existing ones can be modified.

By default they are activated after their definition. But the parameter can be deactivated at any time by DEFINE DISPLAY PICTURE/NOACTIVE

DEFINE DISPLAY SPECTRUM

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

PURPOSE Set Spectrum parameter

PARAMETERS

limits Display limits for spectrum

scalim Limits for scaling axis

update Update time interval.

refresh Refresh time interval.
********** not yet implemented *******

SCALE Scaling mode for Y- or Z-axis for one or two dimensional spectra.

/LIN Display axis in a linear scale

	/LOG	Display axis in a logarithmic scale
	/SQRT	Display axis in a square root scale
X_SCALE	Scaling mode for X-axis	
	/XLIN	Display axis in a linear scale
	/XLOG	Display axis in a logarithmic scale
	/XSQRT	Display axis in a square root scale
Y_SCALE	Scaling mode for Y-axis	
	/YLIN	Display axis in a linear scale
	/YLOG	Display axis in a logarithmic scale
	/YSQRT	Display axis in a square root scale
Z_SCALE	Scaling mode for Z-axis	
	/ZLIN	Display axis in a linear scale
	/ZLOG	Display axis in a logarithmic scale
	/ZSQRT	Display axis in a square root scale
/[NO] WINDOW	Window switch ********** not yet implemented *******	
/[NO] LIFE	Activate life display of spectra. ********** not yet implemented *******	
/[NO] SMOOTH	Display the spectrum with smooth binning.	
/[NO] LETTER	Display lettering on axis	
/[NO] NUMBER	Display numbering on axis	
CALIB	Perform calibration	
	/CALAX	Calibrate axis.
	/CALSPEC	Calibrate spectrum data.
	/NOCAL	no calibration is performed.
/[NO] CHANNELS	Display channel numbers	
RANGE	Select display window	
	/FULL	Use full spectrum range.
	/LAST	use last displayed range.

	/ACTUAL	Use actual displayed range.
SCALE_RANGE	Scaling method	
	/AUTOSCALE	Autoscaling is performed
	/SCALE	Scaling as in MODIFY FRAME command
STYLE	Display mode of spectra. 1. One dimensional spectra	
	/HISTO	histograms are generated
	/VECTOR	spectrum contents are connected with olylines
	/MARKER	spectrum contents are signed with markers
	2. Two dimensional spectra	
	/CONTOUR	Contour lines are displayed.
	/CLUSTER	spectrum contents in cluster mode
	/ISO	pseudo 3D isometric plot is generated.
	/SCATTER	Scatter plot data are simulated.
Caller	MDISP,MGOODISP,D$DSPCM	
Author	W. Spreng	

Example

DEFINE DISPLAY SPECTRUM 5, /LOG/SMOOTH/VECTOR

The spectra are displayed with a lower window limit of 5. The scaling axis (Y for one and Z for two dimensional spectra) is displayed in a logaritmic scale. The display bins are smoothed on the one dimensional spectra are displayed in VECTOR mode. If the two dimensional spectra should be displayed in CLUSTER mode specify additionally:

DEFINE DISPLAY SPECTRUM /CLUSTER

Remarks

File name	D$SDPAR.PPL
Created by	GOO$DISP:D$DSPCM.PPL

Description

CALLING

STS=D$SDPAR(CV_limits,CV_scallim,
L_update,L_refresh,
CV_scale,CV_Xscale,CV_Yscale,CV_Zscale,I_active,
I_error,I_window,I_life,I_rot,I_smooth,CV_letter,
CV_number,CV_calib,CV_channels,CV_range,
CV_scale_range,CV_style,B_mask)

COMMAND

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

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

Display limits for spectrum. The input of a one or two dimensional window is possible, e.g.:
(100,560) - for a one dimensional window
(100,500,200,400) - for a two dimensional window
Any lower and upper limits in the window specification can be skipt. The missings values are replaced by the limits of the displayed spectrum, e.g: (100,). The lower limit is 100. The upper limits are given by the upper limit of the spectrum, displayed with these window.

SCALIM

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

Specifies limits on the scaling axis; for one dimensional spectra this is the y-axis, for two dimensional spectra the z-axis. The lower and upper limits has to be specified, e.g:

(-100,1000)

UPDATE

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable

Update interval. If <= 0 , no update. If > 0 every 'update' seconds the whole spectrum will be updated with the new channel contens of each spectrum.

REFRESH

Routine arg. Input BIN FIXED(31)

Command par. Integer replacable

Refresh intervall. If <=0 , no refresh.

If > 0 all spectra in the picture will be deleted and redrawn every 'refresh' seconds. The scatter plots will be stored on the display and not deleted if a device with selective erase operation is used (e.g. Tektronix 4115) if not the whole spectrum is deleted and the collected data in scatter plots are lost. Therefore be careful using this parameter.

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

SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Set the display mode on the scaling axis. You can select between three modes:

/LIN	Display the linear count rate.

/LOG	Display the spectrum contents in logarithmic mode.
/SQRT	Display the square root of the spectrum contents.

X_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Display mode of X-axis. You can select between three modes:

/XLIN	Display the axis linear.
/XLOG	Display the axis in logarithmic mode.
/XSQRT	Display the square root of axis.

Y_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Display mode of Y-axis. You can select between three modes:

/YLIN	Display the axis linear.
/YLOG	Display the axis in logarithmic mode.
/YSQRT	Display the square root of axis.

Z_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Display mode of Z-axis. You can select between three modes:

/ZLIN	Display the axis linear.
/ZLOG	Display the axis in logarithmic mode.
/ZSQRT	Display the square root of axis.

ACTIVE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable default=/ACTIVE

Activate or deactivate the global display parameter for spectra. Possible inputs are:

/ACTIVATE	activate display parameter. This is the default. The parameter can be deactivated any time by DEFINE DISPLAY SPECTRUM/NOACTIVE or for a single display by DISPLAY SPECTRUM/NOGLOBAL
/NOACTIVE	deactivate the display paramter. They can be activated later by DEFINE DISPLAY SPECTRUM /ACTIVE or for a single display by DISPLAY SPECTRUM/GLOBAL

ERROR

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable

Display statistical errors at each channel of a one dimensional spectrum.

/ERROR	Statistical errors (the square root of the count rate) are displayed.
/NOERROR	No errors are displayed

WINDOW

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable

********** Not yet implemented **********

LIFE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable

********** Not yet implemented **********

ROTATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable

********** Not yet implemented **********

SMOOTH

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch replaceable negatable

Display the spectrum with smooth binning, that mean mean values of the channel contents about the display binsize willl be shown. The effect is that the spectrum looks smoother, but the displayed spectrum contents could be fractional numbers.

/SMOOTH	Display the spectra with smooth binning
/NOSMOOTH	The minimum and maximum contents of the display bins are shown.

LETTER

Routine arg. Input CHAR(*) VAR

Command par. Switch replaceable negatable

Activate or deactivate the lettering on the axis.

/LETTER	The lettering is displayed.
/NOLETTER	The lettering is not displayed.

NUMBER

Routine arg. Input CHAR(*) VAR

Command par. Switch replaceable negatable

Activate or deactivate the numbering on the axis.

/NUMBER	The numbering is displayed.
/NONUMBER	The numbering is not displayed.

CALIB

Routine arg. Input CHAR(*) VAR

Command par. Switch replaceable negatable

Display the spectrum in calibrated units. To do this the displayed spectra has to be connected to an existing calibration. Different calibration modes are available:

/CALAX	The axis is drawn in calibrated units and the spectrum in uncalibrated units. Therefore the distance between two subsequent tics varies.
/CALSPEC	The spectrum is drawn in calibrated units. Then the width of the displayed spectrum bins varies.
/NOCAL	No calibration is performed to the displayed spectra.

In any case the axis with uncalibrated units is displayed, too. To prevent this specify the /NOCHANNELS switch!

CHANNELS

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Activate or deactivate the display of an axis with the original channel units.

/CHANNELS	Display an axis with original units.
/NOCHANNEL	The axis with original units is not not displayed.

RANGE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable

Specify on of the available display windows:

/FULL	The full spectrum range is displayed.
/LAST	The last displayed range is used. This window is indicated be a "#".
/ACTUAL	Use the actual displayed range.

The specified window is used for the display of the spectra.

SCALE_RANGE

Routine arg. Input CHAR(*) VAR

Command par. Set default=/SCALE

Defines the range of the scaling axis.

/AUTOSCALE	Perform autoscaling
/SCALE	Use the scaling range defined in the MODIFY FRAME command.

STYLE

Routine arg. Input CHAR(*) VAR

Command par. Set replacable

Defines the display style of the spectrum data. For one and two dimensional spectra different styles are implemented.

One dimensional spectra:

/HISTO	Histograms are generated
/VECTOR	The spectrum contents are connected by polylines.
/MARKER	The spectrum contents are indicated by markers.

For two dimensional spectra:

/CONTOUR	Contour lines are displayed.
/CLUSTER	The spectrum count rates are indicated by clusters of a variable size and colour.
/ISO	A 3D isometric plot is generated.
/SCATTER	Scatter plot data are simulated. In each the countrate is indicated by a number of points, randomly distributed in the bin.

Function

Define the global display parameters for the display of spectra. Subsequent definitions of global display parameter are cumulative and do not distroy the earlier settings. Therfore at any time additional parameters can be set or existing ones can be modified.

By default they are activated after their definition. But the parameter can be deactivated at any time by DEFINE DISPLAY SPECTRUM/NOACTIVE

DEFINE FRAME SETUP

DEFINE FRAME SETUP tic_number text_font linewidth xyratio channels
/[NO]GRID
/[NO]TICOUTSIDE
/[NO]INFO [=TYPE]

PURPOSE	Define the design of picture frames
PARAMETERS	
tic_number	Number of tics on axis
text_font	Text font for all descriptions
linewidth	Factor to increase the width of all polylines.
xyratio	X/Y-ratio of the maximum used display surface.
channels	Number of channels which should be displayed for two dimensional spectra.
/[NO] GRID	If set the lines of isometric plots are draw parallel to the x- and y-axis. If not set only parallel to the x-axis.
/TICOUTSIDE	Draw tics at spectrum axis outside the frame.
/[NO] INFO	Frame type

/INFO	Frame with information
/NOINFO	Frame without any information

Caller	MDISP
Author	W. Spreng

Remarks

Created by	GOO$DISP:D$DSPCM.PPL
File name	GOO$DISP:

Examples

DEFINE FRAME SETUP 20
Draw 20 tic at the axis, if that is possible.
DEFINE FRAME SETUP FONT=-3
Change the font for all descriptions.
DEFINE FRAME SETUP CHANNELS=200 /NOGRID
Two dimensional spectra are shown with 200 channels and for isometric plots no grids are produced.
DEFINE FRAME SETUP /NOINFO
Draw frames with a large spectrum area, but without detail spectrum informations.

Description

CALLING

```
STS=D$DEFRA(L_tic_number,L_font,R_linewidth,
            R_xyratio, L_CHAN, I_grid, I_ticoutside,
            cv_type)
```

COMMAND

```
DEFINE FRAME SETUP tic_number text_font linewidth xyratio
channels
  /[NO]GRID
  /[NO]TICOUTSIDE
  /[NO]INFO [=TYPE]
```

TIC_NUMBER

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable default=10

Number of tics which should be drawn at the axis. The real number of tics can be different from this number, because it depends on the range of the axis and the size of the frames. But with this parameter you can increase or decrease the number of tics.

FONT

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable default=0

Font number to change text fonts defined in the GKS-bundle tables which are used in case of hardware character requirement.

NOTE That means, hardware font must be enabled, i.e.:
DEFINE FRAME SETUP 10 0 (is default after start)

GTS-GRAL GKS The following GKS-font numbers are supported:

0	Hardware text font
-1...-11	proportional fonts.
-51	thick proportional font
-101..-111	proportional italics
-151	thick proportional italics
-201..-211	mono spaced fonts
-251	thick mono spaced font
-301..-311	mono spaced italics
-351	thick mono spaced italics

Recommended is font -51.

DEC GKS The standard DEC software fonts are:

0	Hardware text font
-1,1	Standard ISO font (Thin, ugly).
-15	Thick Roman (But underscore is arrow)

Recommended is font -15.
With Postscript format you may use one of the following fonts:

-101..-104	Times (Roman, italic, Bold, Bold italic)
-105..-108	Helvet. (Roman,italic,bold,bold italic)
-109..-112	Courier (Roman,italic,bold,bold italic)
-114..-117	Lubalin (Roman,italic,bold,bold italic)
-118..-121	School (Roman,italic,bold,bold italic)
-122..-125	Av.Garde (Roman,italic,bold,bold italic)
-126..-129	Souvenir (Roman,italic,bold,bold italic)

LINEWIDTH

Routine arg. Input BIN FLOAT(24)

Command par. REAL replaceable default=0 min=0 max=10
Factor to increase the width of all polylines!

XYRATION

Routine arg. Input BIN FLOAT(24)

Command par. REAL replaceable default=0 min=0 max=10

XY-ratio of the maximum display surface used for the display of pictures. If set to 0 the default GOOSY display ratio is used (the whole screen). If you want to draw all pictures to be higher than wideeeee specify a ratio lower than 1.0, then the total height of the screen is used, but the width is reduced.

CHANNELS

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable default=100 min=50 max=500

Number of channels which should be displayed for two dimensional spectra.

GRID

Routine arg. Input BIN FIXED(15)

Command par. Switch replaceable default=/GRID

If set the lines of isometric plots are draw parallel to the x- and y-axis. If not set only parallel to the x-axis.

TICOUTSIDE

Routine arg. Input BIN FIXED(15)

Command par. Switch replaceable default=/NOTICOUTSIDE

The tics at the spectrum axis can be drawn inside or outside the spectrum frame. Defauld is inside.

TYPE

Routine arg. Input CHAR(*) VAR

Command par. Set replaceable default=/INFO

Specifies if all spectrum and scatterplot information should be displayed or if the data area should be increased and a minimum set of information should be displayed.

/INFO	Smaller spectra and large information field to display the spectrum header, integration and fit results etc.
/NOINFO	The spectra appear larger, but only the spectrum name is displayed, no place available for futher information.

Function

FUNCTION

In the GOOSY-Display the user has the possibility to modify the default frame set-up. The number of tic-marks on the axis and the text font could be changed.

The maximum number of channels which should be displayed for two dimensional spectra could be defined. This is sometimes useful if the default display set-up of 100 channels is to rough to make details visible. But be careful to increase this parameter too much, the memory requirements and the CPU-time will be increased with the 2th power of this parameter.

If the two-dimensional isometric plots should be displayed as a grid specify /GRID!

Futhermore two different spectrum sizes are implemented. Small spectra with an information area large enough to display the spectrum header and all intgration or fit results(/INFO). Or large spectra with no information area (/NOINFO) are availible.

DEFINE PICTURE SETUP

DEFINE PICTURE SETUP picture rows frames pic_dir base node /EQUAL

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

Remarks

Created by	GOO$DISP:D$DSPCM.PPL
File name	GOO$DISP:D$PISUP.PPL

Examples

DEFINE PICTURE SETUP picture 3 3,6,2
Arrange frames in 3 rows:
1.row 3 frames.
2.row 6 frames.

3.row 2 frames.

The total number of frames in the picture must be 11. If the size of the frames should be equal specify:

DEFINE PICTURE SETUP picture 3 3,6,2 /EQUAL

Description

CALLING STS=D$PISUP(CV_PICTURE,L_ROWS,LA_FRAMES, CV_PIC_DIR,CV_BASE,CV_NODE,I_equal)

COMMAND DEFINE PICTURE SETUP picture rows frames pic_dir base node /EQUAL

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name of the picture for which the frame arrangement should be modified.

ROWS

Routine arg. Input BIN FIXED(31)

Command par. Integer required

Number of the rows in which the frames should be arranged. This number determines the vertical size of the frames. The number of rows can be any number in the range of 1 to the total number of frames in the picture.

FRAMES

Routine arg. Input (*) BIN FIXED(31)

Command par. Integer array required

Specifies the number of frames in each row. Therefore for each row one value has to be specified and the sum over all values has to be equal to the number of total frames in the picture.

E.g. if the frames should be arranged in 4 rows, 4 values are required, e.g. 3,4,4,1:

3 frames in row 1

4 frames in row 2
4 frames in row 3
1 frames in row 4

---

12 frames in total picture.

PIC_DIR

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=$PICTURE

Default picture Directory.

BASE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

/EQUAL

Routine arg. Input (*) BIN FIXED(15)

Command par. Switch

Normally the frames in each row are spread out over the total horizontal range of the display. Therfore the frames in different rows have different sizes if the amount of frames in the rows are different.

But it is possible to give all frames the same size. This can be done by /EQUAL switch. The size of the frames is determined by the row containing the most frames.

Function

With this command the default frame organization on the screen could be changed separately for each picture. This is possible every time after the creation of the picture.

The frames in the picture could be arranged in an arbitrary number of rows on the screen. This means the y-extents of all frames are identical. For each row the number of frames in that row has to be specified in the parameter 'frames'. The sum of all elements in 'frames' should be identical to the number of total frames in the picture.

Normally the frames in each row are spread out over the total horizontal range of the display. Therfore the frames in different rows have different sizes if the amount of frames in the rows are different.

But it is possible to give all frames the same size. This can be done specifying the /EQUAL switch. The size of the frames is determined by the row containing the most frames.

The frames are arranged row by row on the screen, from the left to the right.

DELETE ALIAS

DELETE ALIAS name environment
/GLOBAL

PURPOSE Delete alias name (in MUTIL, GOOSY, MDBM, MDISP). Please use DCL-command DLALI to delete alias.

PARAMETERS

name required string default: ”
Name of alias. Valid characters as for logical names.

environment string replace default: ”
Optional name of environment for which the alias should be deleted.

/GLOBAL switch default: ”
Delete global alias.

Caller MUTIL, GOOSY, MDBM, MDISP

Author H.G.Essel

Example

Delete global alias CS
DEL ALI CS /GLOB
Delete alias CS for environment SUSI
DEL ALI CS SUSI

Remarks

File name U$DLALI.PPL

Created by GOO$UTIL:U$ALICM

Description

CALLING STS=U$DLALI(CV_alias,CV_environment,I_global)

COMMAND DELETE ALIAS name environment
/GLOBAL
Argument / Parameter description.

ALIAS

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”
Name of alias. Valid characters as for logical names.

ENVIRONMENT

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Specifies the environment in which the alias was valid.

/GLOBAL

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”

/GLOBAL The alias was valid independent of an environment.

Function

Delete a logical name in table LNM$ENV_env or LNM$GOOSY.

DELETE CALIBRATION

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

PURPOSE Delete a calibration Data Element.

PARAMETERS

calibration Calibration name specification

node Default node name

base Default Data Base

cal_dir Default Directory

/[NO] LOG Display picture name after deletion

/NOLOG nothing is displayed.

/LOG Names of deleted pictures are shown.

/[NO] KEEP_MAP Keep data base mapping context.

/NOKEEP_MAP Data Base will be dettached

/KEEP_MAP Keep Data Base (DEFAULT)

Caller MDBM,MGOODBM

Author W. Spreng

Example

1.) DELETE CALIBRATION a
Calibration "A" will be deleted.
2.) DELETE CALIBRATION [*]*test* /LOG
All calibrations containing the string "TEST" in the name are deleted and all calibration found are listed.

Remarks

File name GOO$DISP:E$DELCA.PPL

Created by GOO$DE:E$DECMD.PPL

REMARKS The specified calibrations could not be deleted if links to other Data Element exist. This is the case if one or several spectra are connected to this calibration.

Description

CALLING
```
STS = E$DELCA(CV_calib,CV_node,CV_db,CV_cal_dir,
              I_LOG,I_keep_map)
```

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

CALIBRATION

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name of calibrations which should be deleted. Wildcards in calibration and directory name are allowed.

NODE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

BASE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

CAL_DIR

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=$CALIB

Default calibration Directory.

/LOG

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch default=/NOLOG

List the name of the pictures which has been deleted:

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

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch default=/KEEP_MAP

Switch to keep or to dettach the whole Data Base mapping context:

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

Function

The specified Calibration Data Element is deleted from the specified Data Base in the specified Directory on the section file on the specified Node.

Wildcards in the calibration name and in the directory name are supported.

DELETE CONDITION

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

PURPOSE delete a condition

PARAMETERS

name String replace default: none Name of condition

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

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

node String replace default: '*' Name of Node

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

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

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

Caller MDBM

Author K.Winkelmann

File name GOO$DE:E$DLCO.PPL

Dataset -

Remarks

REMARKS -

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$DLCO(CV_NAME,CV_DIR,CV_BASE,
CV_NODE,I_CONFIRM,I_KEEP_MAP)

Argument description

NAME

Type Input CHAR(*) VAR
Name of condition, may be fully qualified
(or wildcarded (nyi))

DIR

Type Input CHAR(*) VAR
Default directory, taken if directory is not
in the name specification

BASE

Type Input CHAR(*) VAR
Default base, is taken if base name is not in the
name specification

NODE

Type Input CHAR(*) VAR
Default node ,...

CONFIRM

Type Input BIN FIXED(15)

0 Confirm to delete condition

1 Noconfirm to delete condition

KEEP_MAP

Type Input BIN FIXED(15)
Valid values are:

0 Unmap of Data Base

1 Inhibit unmap of Data Base

Function

The header element of a condition is searched. Then a queued data elements are deleted. Existing links to the analysis tables are deleted. Finally the header element is deleted. Wildcards for Condition names are supported.

Remarks

Module is an action routine.

Example

```
STS$VALUE=E$DLCO('OTTO','DATA','DB','/NOCONFIRM',
                 '/KEEP_MAP')
```

DELETE DYNAMIC ENTRY

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

PURPOSE Delete a Dynamic List entry

PARAMETERS

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

dyn_list Dynamic List name specification
required common default

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

aux Used by scatter plot
default:'*'

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

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/UPDATE Update

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

Caller M$DMCMD

Author H.G. Essel

File name M$ADLLE.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=M$ADLLE(CV_DYN_TYPE,CV_DYN_LIST, CV_NAMELIST,CV_AUX,CV_DYN_DIR,CV_BASE, CV_NODE,I_UPDATE,I_KEEP_MAP)

ARGUMENTS

CV_DYN_TYPE I Entry type
CHAR(*) VAR

CV_DYN_LIST I Dynamic List name specification
CHAR(*) VAR

CV_NAMELIST I Name list: base:[dir]name,...
CHAR(*) VAR

CV_AUX I Auxiliary string (used by scatter plot entries)
CHAR(*) VAR

CV_DYN_DIR I Default directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_UPDATE I Update
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Delete a Dynamic List entry

REMARKS Module is an action routine.

EXAMPLE -

DELETE DYNAMIC LIST

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

PURPOSE Delete a Dynamic List

PARAMETERS

dyn_list Dynamic List name specification
required common default

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

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

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

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

Caller M$DMCMD

Author H.G. Essel

File name M$ADLDL.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ADLDL(CV_DYN_LIST,CV_DYN_DIR,CV_BASE, CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_DYN_LIST I Dynamic List name specification
CHARACTER(*) VAR

CV_DYN_DIR I Default Directory
CHARACTER(*) VAR

CV_BASE I Default Data Base name
CHARACTER(*) VAR

CV_NODE I Default node name
CHARACTER(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Delete a Dynamic List entry

REMARKS Module is an action routine.

EXAMPLE -

DELETE ELEMENT

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

PURPOSE Delete a Data Element

PARAMETERS

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

dir Default Directory
common default: 'DATA'

base Default Data Base name
common default: 'DB'

node Default node name
common default: 'E'

/UNPROTECT Overide deletion protection of Data Element

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

Caller M$DMCMD

Author M. Richter

File name M$ADLDE.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=M$ADLDE(CV_NAME,CV_DIR,CV_BASE,C_NODE,
I_UNPROTECT,I_KEEP_MAP)

ARGUMENTS

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

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_UNPROTECT I Override deletion protection of Data Element
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Delete a Data Element or a Data Element name array.

REMARKS Module is an action routine.
Since Data Element name arrays could only be
deleted completely, the name array index must be given as a wild card '*' or without any index.

EXAMPLE -

DELETE ENVIRONMENT

DELETE ENVIRONMENT

PURPOSE	Delete a GOOSY environment
PARAMETERS	
FUNCTION	This command deletes the current GOOSY Environment. All subprocesses named GN_env_x are deleted. The supervisor is deleted. The process name is set to Y_env_n, where n is counted from 1 until the name is unique.
EXAMPLE	DEL ENV
Action rout.	U$TCDEE
Author	H.G.Essel

Remarks

File name	U$TCDEE.PPL
Definition	U$TPRCM.PPL
Dataset	-
REMARKS	Command must be given from master process.

Description

CALLING	@CALL U$TCDEE()
ARGUMENTS	
FUNCTION	This procedure deletes a GOOSY environment. For a detailed description look in the command part.
REMARKS	-
EXAMPLE	@CALL U$TCDEE();

DELETE LINK

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

PURPOSE Delete Data Element link(s)

PARAMETERS

link_from Target Data Element name specification
required common default

link_to Target Data Element name specification
required common default

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/ALL Delete all links selected by /SELECT

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

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

Caller M$DMCMD

Author M. Richter

File name M$ADLLI.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=M$ADLLI(CV_LINK_FROM,CV_LINK_TO, CV_DIR,CV_BASE,CV_NODE,I_ALL,CV_SELECT,I_KEEP_MAP)

ARGUMENTS

CV_LINK_FROM I Source Data Element name specification
CHAR(*) VAR

CV_LINK_TO I Target Data Element name specification
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_ALL I Delete all links selected by /SELECT
BIN FIXED(15)

CV_SELECT I Matching, incomming, outgoing links
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Delete Data Element link(s)

REMARKS Module is an action routine.

EXAMPLE -

DELETE OVERLAY

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

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

PARAMETERS

picture Name of picture.

frame Number or range of frames.

node default node name.

base Default Data Base name.

pic_dir Default Picture Directory.

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

/NOKEEP_MAP Data Base will be deattached

/KEEP_MAP Keep Data Base (DEFAULT)

Caller MDBM,MGOODBM,E$DECMD

Author W. Spreng

Remarks

File name -

Created by E$DECMD.PPL

Example

1.) DELETE OVERLAY a 1
The overlayed spectra/scatterplots for frame number 1 are deleted from picture 'A'.
2.) DELETE OVERLAY A 3:8
The overlays for frame 3 to 8 are deleted from picture 'A'.
3.) DELETE OVERLAY A *
The overlays for each frame are deleted from picture 'A'.

Description

CALLING STS=D$DDEOV(CV_picture,CV_frame,CV_node,CV_base, CV_pic_dir)

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

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of picture for which the overlays should be deleted.

FRAME

Routine arg. Input CHAR(*) VAR

Command par. String required

Number of frame for which the overlayed spectra or additional scatterplot parameters should be deleted. Possible inputs are:

n - for a single frame
n:m - for a range of frames
* - for all frames

If the overlays for all frames should be deleted all Data Elements keeping information about overlays will be deleted from the Data Base. If a single frame or a range offrames has been specified the corresponding Data Elements for the overlays are marked unused, but the Data Element itself will not be changed or deleted!

NODE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=*

Default node name

BASE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=DB

Default Data Base name

PIC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=$PICTURE

Default picture directory

KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch default = /KEEP_MAP

If set the mapping context will be kept. If /NOKEEP_MAP specified the Data Base will be dettached after the command completion.

FUNCTION

The overlays for the specified picture will be deleted or marked unused depending on the specified frames.

If all frames in the picture are specified, the Data Elements keeping information about the overlays will be deleted. If a single frame or a range of frames is specified the corresponding overlays are marked unused, but the Data Element itself remains unchanged. Therefore no free areas are generated in that case!

DELETE PICTURE

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

PURPOSE Delete a Picture Data Element.

PARAMETERS

picture Picture name specification

node Default node name

base Default Data Base

pic_dir Default Directory

/[NO] LOG Display picture name after deletion

/NOLOG nothing is displayed.

/LOG Names of deleted pictures are shown.

/[NO] KEEP_MAP Keep data base mapping context.

/NOKEEP_MAP Data Base will be dettached

/KEEP_MAP Keep Data Base (DEFAULT)

Caller MDISP,MGOODISP

Author W. Spreng

Example

1.) DELETE PICTURE a
Picture "A" will be deleted.
2.) DELETE PICTURE [*]*test* /LOG
All pictures containing the string "TEST" in their name are deleted and all pictures found are listed.
3.) DELETE PICTURE * /LOG
Delete all pictures in the default Directory and print a list of them.

Remarks

File name GOO$DISP:D$DEPIC.PPL

Created by GOO$DE:E$DECMD.PPL

REMARKS The specified picture could not be deleted if links to other Data Element exist. This could happen if scatter-plots are defined in the picture and if this picture has an entry in the dynamic list!

Description

CALLING
```
STS = D$DEPIC(CV_picture,CV_node,CV_db,CV_pic_dir,
              I_LOG,I_keep_map,B_mask)
```

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

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name of pictures which should be deleted. Wildcards in picture and directory name are allowed

NODE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

BASE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

PIC_DIR

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=$PICTURE

Default picture Directory.

/LOG

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch default=/NOLOG

List the name of the pictures which has been deleted:

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

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch default=/KEEP_MAP

Switch to keep or to dettach the whole Data Base mapping context:

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

Function

The specified Picture Data Element is deleted from the specified Data Base in the specified Directory on the section file on the specified Node.

Wildcards in the picture name and in the Directory are supported.

DELETE POLYGON

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

PURPOSE delete a polygon

PARAMETERS

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

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

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

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

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

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

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

Caller MDE

Author H.G.Essel

File name GOO$DE:E$DLPO.PPL

Dataset -

Function

The header element of a polygon is searched. Then a queued data elements are deleted. Finally the header element is deleted.

Example

```
STS$VALUE=E$DLPO('OTTO','$POLYGON','DB','E',
                 0,1,1)
```

Remarks

REMARKS Module is an action routine.

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$DLPO(CV_NAME,CV_DIR,CV_BASE,
           CV_NODE,I_CONFIRM,I_KEEP_MAP)
```

Argument description:

NAME

Type Input CHAR(*) VAR
name of polygon, may be fully qualified (or wildcarded (nyi))

POLY_DIR

Type Input CHAR(*) VAR
default directory, taken if directory is not in the name specification

BASE

Type Input CHAR(*) VAR
default base, is taken if base name is not in the name specification

NODE

Type	Input CHAR(*) VAR default node ,...

CONFIRM

Type	Input BIN FIXED(15)
0	Confirm to delete polygon
1	Noconfirm to delete polygon

KEEP_MAP

Type	Input BIN FIXED(15)
0	Inhibit unmap of Data Base
1	Unmap of Data Base

Function

The header element of a polygon is searched. Then a queued data elements are deleted. Finally the header element is deleted.
Wildcards for polygon names are supported.

Remarks

Module is an action routine.

Example

```
STS$VALUE=E$DLPO('S1_POLY','$POLYGON','DB',0,1)
```

DELETE POOL

DELETE POOL pool base
/[NO]KEEP_MAP

PURPOSE Delete a Data Base Pool

PARAMETERS

pool Pool name
required common default

base Data Base name
required common default

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

EXAMPLE DEL POOL ADAM DB

Caller M$DMCMD

Author M. Richter

File name M$ADLPO.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ADLPO(CV_POOL,CV_BASE,I_KEEP_MAP)

ARGUMENTS

CV_POOL I Pool name
CHAR(*) VAR

CV_BASE	I Data Base name CHAR(*) VAR
I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
FUNCTION	Delete a Data Base Pool
REMARKS	Module is an action routine.
EXAMPLE	-

DELETE PROCESS

DELETE PROCESS type

PURPOSE Delete a GOOSY subprocess

PARAMETERS

type Subprocess type to be deleted. This may be the type name of any currently active GOOSY subprocess of the current session.

EXAMPLE DEL PROC ANA1
This will delete the subprocess ANA1 in the current session.

Action rout. U$TCDEP

Author Walter F.J. Mueller

Remarks

File name U$TCDEP.PPL

Dataset -

REMARKS Note, that this command allows to delete GOOSY system processes like $HVR (Hypervisor) or $SVR (Supervisor). So be carefull and think before.

Description

CALLING @CALL U$TCDEP(C_TYPE)

ARGUMENTS

C_TYPE CHAR(*) VAR [INPUT]
Subprocess type to be deleted. May have 1 to 4 characters.

FUNCTION This procedure deletes a GOOSY subprocess, which has been created with the CREATE PROCESS command and wait untill this process is not longer known to VMS.

REMARKS The process is aborted with a SYS$DELPRC system service call, thus no user mode exit handlers will be executed.

EXAMPLE @CALL U$TCDEP('ANA1');

DELETE SECTION

DELETE SECTION base

PURPOSE Delete Global Section attributes

PARAMETERS

base Global Section name
required common default

EXAMPLE -

Caller M$DMCMD

Author M. Richter

File name M$ADLGS.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ADLGS(CV_BASE)

ARGUMENTS

CV_BASE I Global Section name
CHAR(*) VAR

FUNCTION Delete Global Section attributes

REMARKS Module is an action routine.

EXAMPLE -

DELETE SPECTRUM

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

PURPOSE delete a spectrum

PARAMETERS

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

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

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

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

/CAMAC Switch default : "
Delete CAMAC spectra only

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

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

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

Caller MDE

Author K.Winkelmann

File name GOO$DE:E$DLSP.PPL

Dataset -

Function

The header element of a spectrum is searched. Then a queued data elements are deleted. Existing links to the analysis tables are deleted. Finally the header element is deleted.

Example

```
STS$VALUE=E$DLSP('OTTO','$SPECTRUM','DB','E',
                 0,1,1)
```

Remarks

REMARKS Module is an action routine.

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$DLSP(CV_NAME,CV_DIR,CV_BASE,
CV_NODE,I_CAMAC,I_CONFIRM,I_KEEP_MAP)
Argument description:

NAME

Type Input CHAR(*) VAR
name of spectrum, may be fully qualified
(or wildcarded (nyi))

SPEC_DIR

Type Input CHAR(*) VAR
default directory, taken if directory is not
in the name specification

BASE

Type	Input CHAR(*) VAR default base, is taken if base name is not in the name specification

NODE

Type	Input CHAR(*) VAR default node ,...

CAMAC

Type	Input BIN FIXED(15)
0	Delete all specified spectra
1	Delete CAMAC spectra only

CONFIRM

Type	Input BIN FIXED(15)
0	Confirm to delete spectra
1	Noconfirm to delete spectra

KEEP_MAP

Type	Input BIN FIXED(15)
0	Inhibit unmap of Data Base
1	Unmap of Data Base

Function

The header element of a spectrum is searched. Then a queued data elements are deleted. Existing links to the analysis tables are deleted. Finally the header element is deleted.
Wildcards for spectrum names are supported.

Remarks

Module is an action routine.

Example

```
STS$VALUE=E$DLSP('OTTO','DATA','DB',0,1)
```

DETACH ANALYSIS

DETACH ANALYSIS

PURPOSE Detach data base of analysis.

REMARKS Analysis must be stopped. Dynamic lists must be detached.

Description

FUNCTION This command calls $IBUFFER to detach the analysis data base. The initialisation can be explicitely done by command INIT ANA or ATTACH ANA. Any START command calls $IBUFFER after a DETACH command. The analysis must be stopped for DETACH.

File name I$ANACM.PPL

Action rout. I$ANACM_DET

Version 1.01

Author H.G.Essel

Last Update 12-APR-1985

DETACH BASE

DETACH BASE base node

PURPOSE Detach data base.

PARAMETERS

base Data Base name
required common default

node optional node
optional

Caller M$DMCMD

Author H.G.Essel

File name M$ADADB.PPL

EXAMPLE ATT BASE mybase

Remarks

REMARKS -

Description

CALLING STS=M$ADADB(CV_BASE,CV_NODE)

ARGUMENTS

CV_BASE Data Base name
CHAR(*) VAR

CV_NODE optional node
CHAR(*) VAR

FUNCTION Map total bata base.

REMARKS Module is an action routine.

EXAMPLE -

DETACH BASE

DETACH BASE base node

PURPOSE Detach data base.

PARAMETERS

base Data Base name
required common default

node optional node
optional

Caller M$DMCMD

Author H.G.Essel

File name M$ADADB.PPL

EXAMPLE ATT BASE mybase

Remarks

REMARKS -

Description

CALLING STS=M$ADADB(CV_BASE,CV_NODE)

ARGUMENTS

CV_BASE Data Base name
CHAR(*) VAR

CV_NODE optional node
CHAR(*) VAR

FUNCTION Map total bata base.

REMARKS Module is an action routine.

EXAMPLE -

DETACH DISPLAY

DETACH DISPLAY base

PURPOSE	Detach display data base.
PARAMETERS	
base	Data base name or * for all bases.
Caller	MDISP,MGOODISP,D$DSPCM
Action rout.	D$DEALL
Author	W. SPRENG

Examples

DETACH DISPLAY db
Detach data base DB form display process.
DETACH DISPLAY *
Detach all data bases located by the display process

Remarks

File name	D$DETDI.PPL
created by	GOO$DISP:D$DSPCM.PPL

Description

CALLING	STS=D$DEALL(CV_base)
COMMAND	DETACH DISPLAY base

DEVICE

Routine arg. CHAR(*) VAR

Command par. String required

Name of database which should be detached or * if the display process should detach all attached bases.

Function

The specified data base is detached from the display process.

DETACH DYNAMIC LIST

DETACH DYNAMIC LIST dyn_list dyn_dir base node

PURPOSE detach dynamic list

PARAMETERS

dyn_list Dynamic list name specification
common required

dyn_dir Default Directory
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

EXAMPLE -

Caller M$DLCMD

Author H.G. Essel

File name M$ADADL.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ADADL(CV_DYN_LIST,CV_DYN_DIR,
CV_BASE,CV_NODE)

ARGUMENTS

CV_DYN_LIST	Dynamic list name specification CHAR(*) VAR Input
CV_DYN_DIR	Default Directory CHAR(*) VAR Input
CV_BASE	Default Data Base name CHAR(*) VAR Input
CV_NODE	Default node name CHAR(*) VAR Input
FUNCTION	Detach dynamic list
REMARKS	Module is an action routine.
EXAMPLE	-

DISMOUNT BASE

DISMOUNT BASE base

PURPOSE Delete Global Section attribute of Data Base

PARAMETERS

base name of Data Base
common replaced default

EXAMPLE DISMO DATA DB

Author M.Richter

Caller M$DMCMD

File name M$DMDB.PPL

Description

CALLING STS = M$DMDB(cv_db_name)

ARGUMENTS

cv_db_name I Name of the Data Base to be dismounted. A logical name translation will be performed. This name is the name of the (System) Global Section and not of the Section file.
BIN FIXED (31)

FUNCTION The (System) Global Section with the name 'cv_db_name' (a logical name translation will be performed) will be marked for deletion. The actual deletion of the Global Section takes place when all processes that have mapped the Global Section have unmapped all pages. The database is detached.

REMARKS A conversion to upper case characters and a logical name translation will be performed. The name must match with the name known by the VMS system.

EXAMPLE -

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

DISPLAY CALIBRATION

DISPLAY CALIBRATION name cal_dir base node

PURPOSE	Display calibration table.
PARAMETERS	
name	Name of calibration
cal_dir	Directory for calibration Data Elements.
base	Data Base name
node	Node name for Data Base file
Caller	MDISP,MGOODISP
Author	W. Spreng

Remarks

File name	GOO$DISP:D$DCAL.PPL
Created by	GOO$DISP:D$DSPCM.PPL

Description

CALLING	STS=D$DCAL(CV_NAME,CV_CAL_DIR,CV_BASE,CV_NODE)
COMMAND	DISPLAY CALIBRATION name cal_dir base node

NAME

Routine arg.	Input CHAR(*) VAR
Command par.	String required Name of the calibration, which should be displayed.

CAL_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$CALIBRATION
Default Directory for calibrations.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB
Default Data Base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*
Default node name.

Function

The correlation of the calibrated and uncalibrated units is shown graphically. Futhermore it will be checked if the functional dependence is unique. If this is not the case a warning message is produced and the critical points are listed.

DISPLAY CONDITION

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

PURPOSE Display window condition limits

PARAMETERS

condition Name of condition.

frame Number of frame in which the condition should be displayed

dimension Dimension of condition which should be displayed.

cond_dir Default Directory name for condition

base Default Data Base

node default node name

CALIBR Units in which the condition should be displayed.

/CHAN	Original spectrum units.
/CALIB	Calibrated spectrum units.

/DISTRIBUTE Distribute the specified mebers of the condition name array to different frames started with the specified "frame".

AXIS Specifies the axis at which the limits should be marked:

/XAXIS	The limits are marked at the x-axis.
/YAXIS	The limits are marked at the y-axis.

Caller MDISP,MGOODISP,D$DSPCM

Action rout. D$DCWIN

Author W. Spreng

Example

1.) DISPLAY CONDITION WINDOW C(1) 2 3:5
Dimension 3 to 5 of condition array "C(1)" is displayed in frame 2
2.) DISPLAY CONDITION WINDOW c2
Limits of condition C2 is displayed in frame 1 Dimension 1 is used.
3.) DISPLAY CONDITION WINDOW c(2:4) 3 * /DISTRIBUTE
All dimensions of the condition array members c(2), c(3) and c(4) are displayed in different frames:

C(2) in frame 3
C(3) in frame 4
C(4) in frame 5

Remarks

File name D$DCWIN.PPL

Created by GOO$DISP:D$DSPCM.PPL

REMARKS Only one dimensional conditions are supported.

Description

CALLING

```
STS=D$DCWIN(CV_condition,CV_frame,CV_dimension,
            CV_cond__dir,CV_base,CV_node,CV_calibr,
            I_distribute,CV_axis)
```

COMMAND

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

CONDITION

Routine arg. CHAR(*) VAR

Command par. String replaceable

Name of window condition Supported condition name specifications are:

1.) COND_1
2.) COND_ARRAY(1)
3.) COND_ARRAY(2:5)
4.) COND_ARRAY(*)

Supported Condition types are: WINDOW and MULTIWINDOW conditions.

FRAME

Routine arg. CHAR(*) VAR

Command par. String replaceable; default=1

Number of frame in which this condition should be displayed.

DIMENSION

Routine arg. CHAR(*) VAR

Command par. String replaceable; default=*

Dimension of condition limits which should be shown. Possible inputs are:

1. n - single number
2. n:m - range of dimensions
3. * - all condition limits will be shown

COND_DIR

Routine arg. CHAR(*) VAR

Command par. String replaceable global default=$CONDITION

Default Directory for conditions

BASE

Routine arg. CHAR(*) VAR

Command par. String replaceable global default=DB

Default data Base

NODE

Routine arg. CHAR(*) VAR

Command par. String replaceable global default=*

Default node name

CALIBR

Routine arg. CHAR(*) VAR

Command par. Set default=/CHAN

Specified the units in which the condition limits should be displayed

/CHAN	Original spectrum units.
/CALIB	Calibrated spectrum units.

DISTRIBUTE

Routine arg. BIN FIXED(15); valid values 0 and 1

Command par. Switch

Distribute the single members of a name array to subsequent frames. The first frame is "frame".

If the condition array members COND(2:5) and frame 1 is specified the single array members are displayed in subsequent frames in the following manner:

COND(2) into frame 1
COND(3) into frame 2
COND(4) into frame 4
COND(5) into frame 5

AXIS

Routine arg. CHAR(*) VAR

Command par. Switch default="/XAXIS"

Specifies the axis at which the condition limits should be marked.

/XAXIS	The condition limits are marked at the x-axis.
/YAXIS	The condition limits are marked at the y-axis. This is done in any case, even if the spectrum in the frame is one dimensional.

Function

The limits of a specified condition are displayed in the specified frame. For multiwindow conditions the "dimension(s)" which should be shown have to be specified.

The members of condition arrays can be distributed to subsequent frames, if /DISTRIBUTE is specified.

Futhermore the condition limits can be marked at the X- or at the Y-axis.

DISPLAY GRAPH

DISPLAY GRAPH frame file module image

PURPOSE	Display user graphics.
PARAMETERS	
frame	Number of frame into which the graph should be drawn.
file	Specifies a file which can be used to read the graphical x and y vectors.
module	User module which should be dynamicly linked out of a sharable image.
image	Sharable image which contains the specifeid user module.
Caller	MDISP,MGOODISP
Author	W. Spreng

Example

1.) DISPLAY GRAPH 2 test.dat
The graphical data are read from file test.dat and are displayed in frame 2.
2.) DISPLAY GRAPH 5 *::DB:[SPECTRUM]test fit user
The module FIT is called, it has to be linked into the sharable image USER. The specified spectrum name is passed the the user module.

Remarks

Created by	D$DSPCM.PPL
File name	GOO$DISP:D$DGRAP.PPL

Description

CALLING STS=D$DGRAP(L_frame,CV_File,L_index,l_size,
CV_MODULE,CV_IMAGE,CV_type)

COMMAND DISPALY GRAPH frame file index size module image
/MARKER/LINE [=type]

Frame

Routine arg. Input BIN FIXED(31)

Command par. Integer

Number of frame into which the user graphics should be displayed.

File

Routine arg. Input CHAR(*) VAR

Command par. String

Specifies a file from which the X and Y-Vectors of the user graph should be read. If file input is required the module and image name have to be unspecified.

In each record of the file two values are expected. The first is the x-coordinate the second the y-coordinate.

If a module and image is specified this parameter is passed to the user module as a parameter!

INDEX

Routine arg. Input BIN FIXED(31)

Command par. Integer valid values MIN=1, MAX=100

Specifies the linetype index or markerindex which should be used. You can chose a combination of color and type, the index is calculated like:

10*colour+type

The colour representation depends on the output device. The linetypes are the following:

1	solid
2	dashed
3	dotted

4	dashed-dotted
5	long-dashed

It is possible that for serveral devices more linetypes are available. The marker types are:

1	dot
2	plus
3	asterix
4	circle
5	cross
6	square
7	triangle
8	rhomb
9	star
10	large cross

INDEX

Routine arg. Input BIN FIXED(31)

Command par. Integer valid values MIN=1

Specifies trhe linethickness or marker size.

MODULE

Routine arg. Input CHAR(*) VAR

Command par. String

User module which should be linked dynamicly from a user defined sharable image. The user routine must set the calibration table. The user-module is called with the following parameters:

```
   L_status=user(CV_file,L_length,RA_x,RA_y)
CV_file : Input. The string specified in the
          File parameter.
L_length: Used length in the arrays RA_x and RA_y.
          Output.
```

RA_x : Output. X-vector containing the
x-coordinates to be displayed.
RA_y : Output. Y-vector containing the
x-coordinates to be displayed.
A module "user" can be linked into the sharable image "SHARE.EXE" by:

```
LSHARE user.obj share.exe /share=usershr
```

"USERSHR" is the logical image name, which can be inserted into the "IMAGE" parameter of this command.

IMAGE

Routine arg. Input CHAR(*) VAR

Command par. String

Sharable image in which the user module can be found.

Function

A set of user created x/y coordinates can be displayed in the specified frame.

DISPLAY METAFILE

DISPLAY METAFILE file directory

PURPOSE	Read screen image from file and display it
PARAMETERS	
file	Metafile name produced with GKS
directory	VMS-directory for file
Caller	MDISP,MGOODISP,D$DSPCM
Author	W. Spreng

Example

1.) DISPLAY METAFILE file.met [user.metafile]
Metafile "[user.metafile]file.met" is interpreted and displayed on all active devices
2.) DISPLAY METAFILE file.met []
Metafile "[]file.met" is interpreted and displayed on all active devices

Remarks

File name	D$DSAVE.PPL
Created by	D$DSPCM.PPL
REMARKS	The format of the metafile is not fixed in the GKS standard, therefore it is possible that files produced with other GKS-implementation could not be interpreted.

Description

CALLING	STS=D$DSAVE(CV_file,CV_dir)
COMMAND	DISPLAY METAFILE file directory

FILE

Routine arg. Input CHAR(*) VAR

Command par. String required

Metafile name produced with GOOSY, on IBM or with any user software using the GKS-implementation available at GSI!

DIRECTORY

Routine arg. Input CHAR(*) VAR

Command par. String replacable default= SYS$LOGIN:[METAFILE]

VMS-directory where the metafile can be found. Set it "[]" if your file is on the current directory.

Function

Every Metafile produced in GOOSY on IBM or with any user software could be displayed.

To be shure that all information is displayed the workstation viewport has to be set in the metafile. Default is the x/y ration of your main-device to be shure that the whole screen is used.

Metafiles produced with GOOSY are only documented graphical informations! The metafile does not keep any informations about the spectra which are included in your picture. Therefore you can only display the metafile, but it is not possible to manipulate these pictures, e.g. to EXPAND frames, to FIT SPECTRA, to INTEGRATE, etc...

DISPLAY PICTURE

```
DISPLAY PICTURE picture condition object dyn_scat
        binfactor update refresh
        binfactor update refresh
        node base pic_dir dyn_dir cond_dir
        buffer_size
        /LIN /LOG /SQRT [=SCALE]
        /XLIN /XLOG /XSQRT [=X_SCALE]
        /YLIN /YLOG /YSQRT [=Y_SCALE]
        /ZLIN /ZLOG /ZSQRT [=Z_SCALE]
        /[NO]WINDOW
        /[NO]LIFE
        /[NO]ROTATE
        /[NO]SMOOTH
        /[NO]LETTER
        /[NO]NUMBER
        /NOCAL/CALAX/CALSPEC [=CALIB]
        /[NO]CHANNELS
        /SPECIFIED/FULL/LAST/ACTUAL [=RANGE]
        /AUTOSCALE/SCALE [=SCALE_RANGE]
        /HISTO/VECTOR/MARKER/CONTOUR-
        /POINT/ISO/CLUSTER/SCATTER [=STYLE]
        /TEMP /GLOBAL [=MODE]
        /NOSCATTER
        /[NO]ERROR
```

PURPOSE Display screen image as described by picture

PARAMETERS

picture Name of picture

condition Name of condition to be used for all scatter plot frames.

object Object for specified condition

dyn_scat	Name of dynamic listfor the scatter-plot entry
binfactor	Binning Factor
update	Update interval. If <= 0 , no update
refresh	Refresh intervall. If <= 0 , no refresh ******** not yet implemented ********
node	Node name
base	Data Base name
pic_dir	Picture Directory name
dyn_dir	Directory name for dynamic list
cond_dir	Directory name for conditions
buffer_size	Number of scatter points in scatter buffer
SCALE	Scaling mode for Y- or Z-axis
	/LIN Linear scaling axis
	/LOG Logarithmic scaling axis
	/SQRT Squareroot scaling axis
X_SCALE	Scaling mode for X-axis
	/XLIN Linear X-axis
	/XLOG Logarithmic X-axis
	/XSQRT Squareroot X-axis
Y_SCALE	Scaling mode for Y-axis
	/YLIN Linear Y-axis
	/YLOG Logarithmic Y-axis
	/YSQRT Squareroot Y-axis
Z_SCALE	Scaling mode for Z-axis
	/ZLIN Linear Z-axis
	/ZLOG Logarithmic Z-axis
	/ZSQRT Squareroot Z-axis
/[NO] WINDOW	Window switch

	/NOWINDOW	Display no windows
	/WINDOW	Display all associated windows of spectrum

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

/[NO] LIFE Life mode switch

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

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

/[NO] ROTATE Rotate displayed spectra

/NOROTATE	No rotate
/ROTATE	Rotate

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

/[NO] SMOOTH Binnig mode

/SMOOTH	Smooth binning
/NOSMOOTH	min/max binning

/[NO] LETTER Display lettering

/LETTER	Display lettering on axis
/NOLETTER	Display no lettering

/[NO] NUMBER Display numbering

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

CALIB Perform calibration

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

/[NO] CHANNELS Display channel numbers

/CHANNEL	Display spectrum channels.
/NOCHANNEL	Display no spectrum channels.

RANGE Display predefined window

	/FULL	Display full spectrum range
	/LAST	Display last (#) range
	/ACTUAL	Display actual range
	/SPECIFIED	as defined in MODIFY FRAME command.

SCALE_RANGE Scaling method

/AUTOSCALE	Autoscaling is performed
/SCALE	Scaling as in MODIFY FRAME command

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

/HISTO	Draw histograms
/VECTOR	Connect spectrum bins with lines
/MARKER	Signe spectrum contents with markers

For two dimensional spectra:

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

MODE Select temporary or global display modes

/TEMP	Use last display parameters
/GLOBAL	Use global picture parameters
/NOGLOBAL	do not use global parameters.

/NOSCATTER Deactivtivate switch for scatter plots

/NOSCATTER	Do not start scattering.

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

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

Caller MDISP,MGOODISP,D$DSPCM

Author W. Spreng

Examples

1.) DISPLAY PICTURE a
Picture "a" is diplayed as defined in the Data Element.
2.) DISPLAY PICTURE a /global
Picture "a" is diplayed with the switches defined in the SET PICTURE PARAMETER command.
3.) DISPLAY PICTURE a /GLOBAL/LIN
Picture "a" is diplayed with ths switches defined in the SET PICTURE PARAMETER command. but with linear scaling axis.
4.) DISPLAY PICTURE a /YLOG
All y-axis are displayed in "LOG" mode, this is valid for scatterframes too!

Remarks

File name D$DPICT.PPL

Created by GOO$DISP:D$DSPCM.PPL

REMARKS All parameters, qualifiers and switches specified in this command acts on all frames! If overlayed spectra or scatter-plot parameters have been defined with the OVERLAY command and are not saved in the picture Data Element they are lost when this command is given! Only overlays specified in the Data Element will be considered and could be displayed with the OVERLAY command.

Description

CALLING

```
STS=D$DPICT(CV_picture,CV_condition,CV_object,
      CV_dyn_scat,L_binfactor,L_update,L_refresh,
      CV_node,CV_db,CV_pic_dir,CV_dyn_dir,CV_cond_dir
     ,CV_buffer_size,CV_scale,CV_Xscale,CV_Yscale
     ,CV_Zscale,I_window,I_life,I_rotate
     ,I_smooth,CV_letter,CV_number,
      CV_calib,CV_channels,CV_range,
     ,CV_scale_range,CV_style,CV_mode,I_noscatter,
      I_error,B_mask)
```

COMMAND

```
DISPLAY PICTURE picture condition object dyn_scat
        binfactor update refresh
        node base pic_dir dyn_dir cond_dir
        buffer_sizeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

/LIN /LOG /SQRT [=SCALE]
/XLIN /XLOG /XSQRT [=X_SCALE]
/YLIN /YLOG /YSQRT [=Y_SCALE]
/ZLIN /ZLOG /ZSQRT [=Z_SCALE]
/[NO]WINDOW
/[NO]LIFE
/[NO]ROTATE
/[NO]SMOOTH
/[NO]LETTER
/[NO]NUMBER
/NOCAL/CALAX/CALSPEC [=CALIB]
/[NO]CHANNELS
/SPECIFIED/FULL/LAST/ACTUAL [=RANGE]
/AUTOSCALE/SCALE [=SCALE_RANGE]
/HISTO/VECTOR/MARKER/CONTOUR-
/POINT/ISO/CLUSTER/SCATTER [=STYLE]
/TEMP /GLOBAL [=MODE]
/NOSCATTER
/[NO]ERROR

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name specification of Picture Data Element which should be displayed. All frames of the picture has to be defined, before a display is possible!

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. String

General picture condition for all scatterplots in this picture. The scatter data are show if the result flag of the condition is true. If this parameter is specified the global picture condition defined in the CREATE PICTURE command will be overwritten.

OBJECT

Routine arg. Input CHAR(*) VAR

Command par. String

Condition object for the general picture condition. If this parameter is specified the condition is checked against the object. This will destroy the result of an earlier check of the same condition!

DYN_SCAT

Routine arg. Input CHAR(*) VAR

Command par. String global default=$SCATTER

Name of dynamic list for the scatter-plot entry

BINFACTOR

Routine arg. Input BIN FIXED(31)

Command par. Integer, default=0

Binning Factor. If larger than "0" it specifies the number of bins to be summed up in the display. This corresponds to a temporarily modification of the spectrum binsize. E.g. if the binning factor is 2 the count rate in one displayed spectrum bin is given by the sum of two neighbouring bins in the original spectrum!

If =0 an internally calculated display binsize is used to optimize the displayed data.

UPDATE

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable

Update interval. If <= 0 , no update. If > 0 every 'update' seconds the whole spectrum will be updated with the new channel contents of each spectrum.

REFRESH

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable

Refresh intervall. If <=0 , no refresh. If > 0 all spectra in the spectrum will be deleted and redrawn every 'refresh' seconds. The

scatter plots will be stored on the display and not deleted if a device with selctive erase operation is used (e.g. Tektronix 4115) if not the whole spectrum is deleted and the collected data in scatter plots are lost. Therefore be careful using this parameter.
********** Not yet implemented **********

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global default=*
Default node name.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global default=DB
Default Data Base where the picture is assumed.

PIC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global default=$PICTURE
Default Directory name.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global default=$DYNAMIC
Default Directory name for dynamic list

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global default=$CONDITION
Default Directory name for conditions

BUFFER_SIZE

Routine arg. Input CHAR(*) VAR

Command par. String global replace default=1000

Number of scatter points in scatter buffer.

SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Set the display mode on the scaling axis. You can select between three modes:

/LIN	Display the linear count rate.
/LOG	Display the spectrum contents in logarithmic mode.
/SQRT	Display the square root of the spectrum contents.

X_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of X-axis. You can select between three modes:

/XLIN	Display the axis linear.
/XLOG	Display the axis in logarithmic mode.
/XSQRT	Display the square root of axis.

Y_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of Y-axis. You can select between three modes:

/YLIN	Display the axis linear.
/YLOG	Display the axis in logarithmic mode.
/YSQRT	Display the square root of axis.

Z_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of Z-axis. You can select between three modes:

/ZLIN	Display the axis linear.
/ZLOG	Display the axis in logarithmic mode.
/ZSQRT	Display the square root of axis.

WINDOW

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

********** Not yet implemented **********

LIFE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

********** Not yet implemented **********

ROTATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

********** Not yet implemented **********

SMOOTH

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

To optimize the displayed spectrum data, the display reduces the number of displayed bins by an internal display binsize. The spectrum bins can be gathered in to ways:

/SMOOTH	Display the spectra with smooth binning. In that mode the mean values of the channel contents over the display binsize will be shown. The effect is that the spectrum looks smoother, but the displayed spectrum contents could be fractional numbers.
/NOSMOOTH	The minimum and maximum contents of the display bins are shown. In that modes spikes in the spectra are not smoothed out.

LETTER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the lettering on the axis.

/LETTER	The lettering is displayed.
/NOLETTER	The lettering is not displayed.

NUMBER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the numbering on the axis.

/NUMBER	The numbering is displayed.
/NONUMBER	The numbering is not displayed.

CALIB

Routine arg. Input CHAR(*) VAR

Command par. Set

Display the spectrum in calibrated units. To do this the displayed spectra has to be connected to an existing calibration. Different calibration modes are available:

/CALAX	The axis is drawn in calibrated units and the spectrum in uncalibrated units. Therefore the distance between two subsequent tics varies.
/CALSPEC	The spectrum is drawn in calibrated units. Then the width of the displayed spectrum bins varies.
/NOCAL	No calibration is performed to the displayed spectra.

In any case the axis with uncalibrated units is displayed too. To prevent this specify the /NOCHANNELS switch!

CHANNELS

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactivate the display of an axis with the original channel units.

/CHANNELS	Display an axis with original units.
/NOCHANNEL	The axis with original units is not not displayed.

RANGE

Routine arg. Input CHAR(*) VAR

Command par. Set default=/LAST

Specify on of the available display windows:

/FULL	The full spectrum range is displayed.

/LAST	The last displayed range is used. This window is indicated be a "#".
/ACTUAL	Use the actual displayed range.
/SPECIFIED	Use the range as defined in the MODIFY FRAME command.

The specified window is used for the display of the spectra.

SCALE_RANGE

Routine arg. Input CHAR(*) VAR

Command par. Set default=/SCALE

Defines the range of the scaling axis.

/AUTOSCALE	Perform autoscaling
/SCALE	Use the scaling range defined in the MODIFY FRAME command.

STYLE

Routine arg. Input CHAR(*) VAR

Command par. Set

Defines the display style of the spectrum data. For one and two dimensional spectra different styles are implemented.

One dimensional spectra:

/HISTO	Histograms are generated
/VECTOR	The spectrum contents are connected by polylines.
/MARKER	The spectrum contents are indicated by markers.

For two dimensional spectra:

/CONTOUR	Contour lines are displayed.
/CLUSTER	The spectrum count rates are indicated by clusters of a variable size and colour.
/ISO	A 3D isometric plot is generated.

/SCATTER Simulate scatter plot data.

For one dimensional spectra /HISTO is default for two dimensional /CONTOUR is default.

MODE

Routine arg. CHAR(*) VAR

Command par. SET

Use the global or temporary display parameters. The temporary parameters, are the modes specified in the last DISPLAY command. The global parameters are specified with DEFINE DISPLAY PICTURE.

/TEMP Use temporary parameters.

/GLOBAL Use the global parameters.

NOSCATTER

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

Switch to deactivate scatter-plot

/NOSCATTER Scatter plot have to be activated via command

If nothing specified the scatterplot is activated

ERROR

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

Display statistical errors at each channel of a one dimensional spectrum.

/ERROR Statistical errors (the square root of the count rate) are displayed.

/NOERROR No errors are displayed

Function

Display the picture in the modes and limits defined with the MODIFY FRAME commands. All frames have to be defined before the picture can be displayed.

The specifications kept in the frames are changed by the parameter of the DISPLAY PICTURE command, but only for the current display. Futhermore the parameter act on all frames!

The global picture condition is changed permanently if a "condition" with or without an "object" has been specified.

DISPLAY POINT

DISPLAY POINT point frame
/LOOP
[CALIBR=]/CHAN/CALIB

PURPOSE Mark point on screen and get channel contents.

PARAMETERS

point Name of point for input or value

frame Number of frame or spectrum name

/LOOP Enter cursor loop to mark several points

CALIBR Specifies the unit in which the coordinates are given.

/CHAN	Original spectrum units.
/CALIB	Calibrated units.

Caller MDISP,MGOODISP,D$DSPCM

Author W. Spreng

Example

1. DISPLAY POINT (1023) 1 or DISPLAY POINT 1023 1
returns contents of channel 1023 of 1dim. spectrum in frame 1
DISPLAY POINT (100,600) 2
the same for 2 dimensional spectrum in frame 2.
2. DISPLAY POINT 1023
3. DISPLAY POINT frame=1
4. DISPLAY POINT
5. DISPLAY POINT /LOOP
6. DISPLAY POINT frame=1/LOOP

Remarks

File name D$DPOIN.PPL

Created by GOO$DISP:D$DSPCM.PPL

Description

CALLING STS=D$DPOIN(CV_point,CV_frame,CV_calibr)

COMMAND DISPLAY POINT point frame
/LOOP
[CALIBR=]/CHAN/CALIB

POINT

Routine arg. Input CHAR(*) VAR

Command par. String

Coordinates of the point to be marked. The coordinates should be specified like:
(number) for 1.-dim spectra
(n1,n2) for 2.-dim spectra
If no coordinates have been specified the cursor appears to select one or several points.

FRAME

Routine arg. Input CHAR(*) VAR

Command par. String

Number of frame in which the coordinates should be be marked or selected. The spectrum in this frame is used to determine the channel contents. If no frame is specified and if more than one frame is on the screen, the cursor appears to pick the frame.

LOOP

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch

Enter cursor loop until break facility is given. If specified an arbitrary number of points in any frame could be marked.

CALIBR

Routine arg. Input CHAR(*) VAR

Command par. Set default="/CHAN"

If the spectrum in the selected frame is calibrated the input of the coordinates can occur in in calibrated or uncalibrated units:

/CHAN	Coordinate of point is in channels
/CALIB	Coordinates of point are in calibrated units

If the coordinates are specified via cursor input the units are known and this switch has no effect!

FUNCTION

The selected points will be marked in the selected frames and the contents of the spectrum in that frame is displayed.

Different inputs are possible (look at the examples):

1. 'point', 'frame' are specified:
   The specified point is marked on the display and information about the spectrum contents is returned.
2. 'point' specified:
   The cursor appears to select the frame.
3. 'frame' is specified:
   Then only points in that frame could be marked.
4. Nothing specified:
   Cursor appears to select the point and the frame
5. /LOOP is specified :
   The cursor appears to select an arbitrary number of points in different frames. The input will be finished if the break facility is given. The parameter CV_point is ignored.
6. /LOOP and 'frame' is specified:
   An arbitrary number of points could be selected in the specified frame.

The point to be displayed could be specified in channels or in calibrated units, depending on the switches "/CHAN or /CALIB". If calibrated units are given, but the spectrum in the specified frame is not connected to a calibration an error is signaled.

In any case the points are marked on the screen and the spectrum contents is displayed.

DISPLAY POLYGON

DISPLAY POLYGON polygon frame poly_dir base node
/FILL

PURPOSE	Display polygon points.
PARAMETERS	
polygon	Name of polygon.
frame	Frame in which the polygon should be displayed.
poly_dir	Directory for polygons.
base	Data Base name
node	Node name for Data Base file
/FILL	Fill interior of polygon.
Caller	MDISP,MGOODISP
Author	W. Spreng

Remarks

File name	D$DPOLY.PPL
Created by	D$DSPCM.PPL

Description

CALLING	STS=D$DPOLY(CV_polygon,L_frame,CV_poly_DIR, CV_BASE,CV_NODE,I_fill)
COMMAND	DISPLAY POLYGON polygon frame poly_dir base node /FILL

POLYGON

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of the polygon , which should be displayed.

FRAME

Routine arg. Input BIN FIXED(15)

Command par. Integer

Number of the frame in which the polygon should be displayed. If pictures with more than one frame are active, the frame number has to be specified.

POLY_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$POLYGON

Default Directory for polygons.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

FILL

Routine arg. Input BIN FIXED(15) valid inputs are 0 and 1

Command par. Integer

Fill the interior of the polygon to indicate the points inside. If not set only the contour of the is drawn.

Function

The specified polygon is displayed in the specified frame.

DISPLAY SCATTER

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

PURPOSE Display single scatter frame for two parameters.

PARAMETERS

xparam	Name of parameter for x- value
yparam	Name of parameter for y- value
limits	Window limits for X and Y.
condition	Name of condition
object	Parameter object on which the condition acts.
dyn_scat	Name of dynamic list for scatter plot
xletter	X-lettering on scatterplot axis.
yletter	Y-lettering on scatterplot axis.
node	Default node name for all Data element name specifications.
base	Default Data Base name for all Data Elements
par_dir	Default Directory for parameters (objects)
dyn_dir	Default Directory name for dynamic list
cond_dir	Default Directory name for condition
buffer_size	Number of scatter points in scatter buffer

RANGE Display window to be used.

/LAST	Window of last DISPLAY SCATTER command
/ACTUAL	Window of the current active scatterplot

MODE Display modes to be used.

/TEMP	Modes of the last DISPLAY command
/GLOBAL	Global display modes should be used

/NOSCATTER Wait for START SCATTER comand.

Caller MDISP,MGOODISP,D$DSPCM

Author W. Spreng

Remarks

File name D$DSCAT.PPL

Created by GOO$DISP:D$DSPCM.PPL

Description

CALLING

```
STS=D$DSCAT(CV_xparam,CV_yparam,
            CV_window,CV_condition,CV_obj,CV_dyn
            CV_xletter,cv_yletter,
            CV_node,CV_db,CV_par_dir,CV_dyn_dir,
            CV_cond_dir,CV_buffer_size
            CV_range,CV_mode,I_noscatter,B_mask)
```

COMMAND

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

XPARAM

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name of the parameter displayed in x-direction. If no directory the default parameter Directory is used. Valid input, e.g for J11 standard event:

EVENT.IA$EVENT(1)

YPARAM

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name of parameter for y-value. If no Directory is specified the Directory of "xparam" is used as default, e.g a valid input for a standard J11 event is:

EVENT.IA$EVENT(2)

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=(0,1023,0,1023)

Window limits to specify the range of the displayed parameter correlations. Valid inputs are any GOOSY display window specification (see the GOOSY Display Manual), e.g.

(0,100,300,400)

(,,300,400) xmin=0, xmax=1023, ymin=300, ymax=400

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. String

Name of condition which is used to filter the scatter data. If no condition object is specified only the result bit of the condition is checked. If it is true the scatter data are displayed.

OBJECT

Routine arg. Input CHAR(*) VAR

Command par. String

Condition object for the general picture condition. If this parameter is specified the condition is checked against the object. This will destroy the result of an earlier check of the same condition!

DYN_SCAT

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$SCATTER

Name of dynamic list used to create the entries necessary for the scatter plot. The SCATTER entry in this dynamic list is necessary to tell the analysis which scatterplots are requested by your display.

ATTENTION The dynamic list has to be attatched before a scatterplot could be created.

XLETTER,YLETTER

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

X- and Y-lettering for scatter plot axis. If nothing is specified the parameter names are used

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node name for all Data Element name specifications.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name for all Data Elements

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DATA

Default Directory for parameter.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global raplacable default=$DYNAMIC
Default Directory name for dynamic list

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$CONDITION
Default Directory name for conditions

BUFFER_SIZE

Routine arg. Input CHAR(*) VAR

Command par. String global replace default=1000
Number of scatter points in scatter buffer.

RANGE

Routine arg. Input CHAR(*) VAR

Command par. SET
Display window to be used.

/LAST	Window of last DISPLAY SCATTER command
/ACTUAL	Window of the current active scatterplot

MODE

Routine arg. Input CHAR(*) VAR

Command par. SET
Display modes to be used.

/TEMP	Modes of the last DISPLAY command
/GLOBAL	Global display modes should be used

NOSCATTER

Routine arg. Input BIN FIXED(15) valid inputs are 0 and 1

Command par. SWITCH

Do not start with the scatter plot, wait for START SCATTER command.

Function

With this command correlations between two event parameters 'xparam' and 'yparam'could be displayed in live mode. To pass the data for the required scatterplot to the analysis an entry in the dynamic list 'dyn_scat' is created.

The condition is used as a trigger for the scattered correlations. If the condition is specified without any object only the result bit of this condition is checked, therefore take care that the condition has been executed. If an object has been specified the condition will be applied to it, even if the condition has been executed earlier. Therefore take care that the specified condition is not refered twice in the specified dynamic list, the last condition check wins. This is the reason why it is recommended to use a separate dynamic list for all scatter plots.

As default mode the scatter-plot works asynchronously to the analysis so that the analysis does not slow down. If the whole data should be displayed on screen, the scatter-plot have to run synchronously to the analysis. In that case you have to specify /NOSCATTER to prevent the scatter mode running on. After that you have to give the command:

START SCATTER /SYNCHRONOUS

In any case the scatter-plot data are not stored on disk or in the local storage of a display device. So the scatter plot provides a high resolution 2-dimensional display without occupying any computer storage. Therefore it is not possible to send scatter data to a plotter. If you try this the only effect is that you see on your plot a frame without any scatter data in it. This has been done for two reasons:

1. to reduce the used storage and

2. to ensure a scatter rate which is only limited by the physical transmission rate of the terminal line.

To get a nice plot of your scatter data you have to produce a plotfile by allocating a plotter device or by creating a metafile:

ALLOCATE DEVICE scatter ln03
ALLOCATE DEVICE scatter metaout

which can be sent to a plotter by the corresponding PLOT PLOTFILE or PLOT METAFILE commands!

DISPLAY SPECTRUM

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

PURPOSE Display spectrum using default picture

PARAMETERS

spectrum Name of spectrum

limits Display limits for spectrum

scalim Limits for scaling axis

binfactor Binning factor.

update Update interval.

refresh	Refresh interval. ******** not yet implemented ********	
cuts	Cuts on z-axis for CONTOUR/CLUSTER plots.	
theta	Rotation angle about x-axis for ISOMETRIC plots.	
phi	Rotation angle about y-axis for ISOMETRIC plots.	
node	Default node name.	
base	Default Data Base where the picture is assumed.	
spec_dir	Default Directory name.	
SCALE	Scaling mode for Y- or Z-axis	
	/LIN	Linear scaling axis
	/LOG	Logarithmic scaling axis
	/SQRT	Squareroot scaling axis
X_SCALE	Scaling mode for X-axis	
	/XLIN	Linear X-axis
	/XLOG	Logarithmic X-axis
	/XSQRT	Squareroot X-axis
Y_SCALE	Scaling mode for Y-axis	
	/YLIN	Linear Y-axis
	/YLOG	Logarithmic Y-axis
	/YSQRT	Squareroot Y-axis
Z_SCALE	Scaling mode for Z-axis	
	/ZLIN	Linear Z-axis
	/ZLOG	Logarithmic Z-axis
	/ZSQRT	Squareroot Z-axis
/[NO] WINDOW	Window switch ********** not yet implemented *******	
/[NO] LIFE	Life mode switch ********** not yet implemented *******	

/[NO] ROTATE Rotate displayed spectra
********** not yet implemented *******

/[NO] SMOOTH Binnig mode

/SMOOTH	Smooth binning
/NOSMOOTH	min/max binning

/[NO] LETTER Display lettering

/LETTER	Display lettering on axis
/NOLETTER	Display no lettering

/[NO] NUMBER Display numbering

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

CALIB Perform calibration

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

/[NO] CHANNELS Display channel numbers

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

RANGE Display predefined window

/FULL	Display full spectrum range
/LAST	Display last (#) range
/ACTUAL	Display actual range

SCALE_RANGE Scaling method

/AUTOSCALE	Autoscaling is performed
/SCALE	Scaling as specified in last command

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

/HISTO	Draw histograms
/VECTOR	Connect spectrum bins with lines

	/MARKER	Signe spectrum contents with markers

For two dimensional spectra:

	/CONTOUR	Contour lines are displayed
	/CLUSTER	Clusters indicating the count rate
	/ISO	Show spectrum as an isometric plot.
	/SCATTER	Scatter plot data are simulated.
MODE	Select temporary or global display modes	
	/TEMP	Use last display parameters
	/GLOBAL	Use global picture parameters
	/NOGLOBAL	do not use global parameters.
/[NO] ERROR	Draw statistical error bars for each bin.	
	/NOERROR	No error bars drawn.
	/ERROR	Error bars are drawn at each bin.
Caller	MDISP,MGOODISP,D$DSPCM	
Author	W. Spreng	

Examples

1.) DISPLAY SPECTRUM spec
"spec" is displayed in default mode
2.) DISPLAY SPECTRUM /TEMP
The last spectrum is displayed with the same modes and within the same range as before.
3.) DISPLAY SPECTRUM two CUTS=0.4,0.5,0.7,0.9
Spectrum "two" is displayed in contour mode with the contourlines at the specified cuts.
4.) DISPLAY SPECTRUM two THETA=0.0 PHI=0.0/iso
Spectrum "two" is displayed in isometric mode. The x-axis is identical to the horizontal screen axis and the y-axis points into the screen, the z-axis points along the vertical screen axis.
5.) DISPLAY SPECTRUM two THETA=90.0 phi=90.0/iso
Now the x-axis of the spectrum is along the vertical screen axis and the y-axis is along the horizontal screen axis (from right to left) the z-axis comes out of your screen. With this set-up you will see only crossing lines.

Remarks

File name D$DSPEC.PPL

Created by D$DSPCM.PPL

Description

CALLING

```
STS=D$DSPEC(CV_spectrum,CV_limits,CV_scalim,
      L_binfactor,L_update,L_refresh,L_NUM_CUTS,
      RA_cuts,CV_theta,CV_phi,
      CV_node,CV_base,CV_spec_dir,
     ,CV_scale,CV_Xscale,CV_Yscale
     ,CV_Zscale,I_window,I_life,I_rotate
     ,I_smooth,CV_letter,CV_number,
      CV_calib,CV_channels,CV_range,
     ,CV_scale_mode,CV_style,CV_mode,LA_array,B_mask)
```

COMMAND

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

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

Name of spectrum which should be displayed

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String

Display limits for spectrum. Valid inputs are any GOOSY display window specification, e.g:

(xmin,xmin) - Minimum and maximum value for x
(xmin,) - Maximum is upper spectrum limit.
(,xmax) - Minimum is lower spectrum limit.
(xmin,xmax,ymin,ymax) -
for a two dimensional window.

SCALIM

Routine arg. Input CHAR(*) VAR

Command par. String

Range of scaling axis. Valid inputs are:
(min,max) - Minimum and maximum limits of scaling axis.
(min,) - upper limits is maximum spectrum contents.
(,max) - lower limits is minimum spectrum contents.

BINFACTOR

Routine arg. Input BIN FIXED(31)

Command par. Integer, default=0

Binning Factor. If larger than "0" it specifies the number of bins to be summed up in the display. This corresponds to a temporarily modification of the spectrum binsize. E.g. if the binning factor is 2 the count rate in one displayed spectrum bin is given by the sum of two neighbouring bins in the original spectrum!

If =0 an internally calculated display binsize is used to optimize the displayed data.

UPDATE

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable

Update interval. If <= 0 , no update. If > 0 every 'update' seconds the whole spectrum will be updated with the new channel contents of each spectrum.

REFRESH

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable

Refresh interval. If <=0 , no refresh. If > 0 all spectra in the spectrum will be deleted and redrawn every 'refresh' seconds. The scatter plots will be stored on the display and not deleted if a device with selctive erase operation is used (e.g. Tektronix 4115) if not the whole spectrum is deleted and the collected data in scatter plots are lost. Therefore be careful using this parameter.

+++++++ not yet implemented ++++++++++++++++++

NUMCUTS

Routine arg. Input BIN FIXED(31)

Command par. Integer, default=0

Defines the number of cuts used for the display of CONTOUR, CLUSTER and SCATTER spectra. If specified as 0 the cuts of the CUTS parameter are used, if larger than 0 the cuts are displayed in a fixed stepwidth and the cuts defined in CUTS are ignored.

CUTS

Routine arg. Input CHAR(*) VAR

Command par. Real arrray default=(0.1,0.2,0.3,0.4,0.5,0.6,0.7, 0.8,0.9,1.0)

Cuts for CLUSTER/CONTOUR plots. The cuts are interpreted in relative units of the difference between the spectrum maximum and the spectrum minimum (e.g if 0.5 is specified the contour line at half between the spectrum maximum and spectrum minimum will be drawn). The actual cuts are calculated like:

ACTUAL = min_spectrum +
cuts*(max_spectrum-min_spectrum)

Therefore the specified values have to be in the range of 0.0<=R_cuts<=1.0.

THETA

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=25.0

Rotation angle about X-Axis. This is the first rotation which is performed clockwise looking in direction of the X-axis.

PHI

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=25.0

Rotation angle about Z-Axis. This is the second rotation which is performed clockwise looking in direction of the Z-axis.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$SPECTRUM

Default Directory name for spectra

SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set default=/LIN

Set the display mode on the scaling axis. You can select between three modes:

/LIN	Display the linear count rate.
/LOG	Display the spectrum contents in logarithmic mode.
/SQRT	Display the square root of the spectrum contents.

X_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set default=/XLIN

Display mode of X-axis. You can select between three modes:

/XLIN	Display the axis linear.
/XLOG	Display the axis in logaritmic mode.
/XSQRT	Display the square root of axis.

Y_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set default=/YLIN

Display mode of Y-axis. You can select between three modes:

/YLIN	Display the axis linear.
/YLOG	Display the axis in logaritmic mode.
/YSQRT	Display the square root of axis.

Z_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set deafult=/ZLIN

Display mode of Z-axis. You can select between three modes:

/ZLIN	Display the axis linear.
/ZLOG	Display the axis in logaritmic mode.
/ZSQRT	Display the square root of axis.

WINDOW

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

********** Not yet implemented **********

LIFE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

********** Not yet implemented **********

ROTATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

********** Not yet implemented **********

SMOOTH

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

To optimize the displayed spectrum data, the display reduces the number of displayed bins by an internal display binsize. The spectrum bins can be gathered in to ways:

/SMOOTH	Display the spectra with smooth binning. In that mode the mean values of the channel contents over the display binsize will be shown. The effect is that the spectrum looks smoother, but the displayed spectrum contents could be fractional numbers.
/NOSMOOTH	The minimum and maximum contents of the display bins are shown. In that modes spikes in the spectra are not smoothed out.

LETTER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the lettering on the axis.

/LETTER	The lettering is displayed.
/NOLETTER	The lettering is not displayed.

NUMBER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the numbering on the axis.

/NUMBER	The numbering is displayed.
/NONUMBER	The numbering is not displayed.

CALIB

Routine arg. Input CHAR(*) VAR

Command par. Set

Display the spectrum in calibrated units. To do this the displayed spectra has to be connected to an existing calibration. Different calibration modes are available:

/CALAX	The axis is drawn in calibrated units and the spectrum in uncalibrated units. Therefore the distance between two subsequent tics varies.
/CALSPEC	The spectrum is drawn in calibrated units. Then the width of the displayed spectrum bins varies.
/NOCAL	No calibration is performed to the displayed spectra.

In any case the axis with uncalibrated units is displayed too. To prevent this specify the /NOCHANNELS switch!

CHANNELS

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactivate the display of an axis with the original channel units.

/CHANNELS	Display an axis with original units.
/NOCHANNELS	The axis with original units is not displayed.

RANGE

Routine arg. Input CHAR(*) VAR

Command par. Set

Specify one of the available display windows:

/FULL	The full spectrum range is displayed.
/LAST	The last displayed range is used. This window is indicated be a "#".

/ACTUAL	Use the actual displayed range.

The specified window is used for the display of the spectra. If no window is given the spectrum is displayed in its full range.

SCALE_RANGE

Routine arg. Input CHAR(*) VAR

Command par. Set

Defines the range of the scaling axis.

/AUTOSCALE	Perform autoscaling
/SCALE	Use the scaling range defined in the last DISPLAY SPECTRUM command.

STYLE

Routine arg. Input CHAR(*) VAR

Command par. Set

Defines the display style of the spectrum data. For one and two dimensional spectra different styles are implemented.

One dimensional spectra:

/HISTO	Histograms are generated
/VECTOR	The spectrum contents are connected by polylines.
/MARKER	The spectrum contents are indicated by markers.

For two dimensional spectra:

/CONTOUR	Contour lines are displayed.
/CLUSTER	The spectrum count rates are indicated by clusters of a variable size and colour.
/ISO	A 3D isometric plot is generated.
/SCATTER	Scatter plot data are simulated. In each the countrate is indicated by a number of points, randomly distributed in the bin.

For one dimensional spectra /HISTO is default, for two dimensional spectra /CONTOUR is default.

MODE

Routine arg. CHAR(*) VAR

Command par. SET

Use the global or temporary display parameters. The temporary parameters, are the modes specified in the last DISPLAY command. The global parameters are specified with DEFINE DISPLAY PICTURE.

/TEMP	Use temporary parameters.
/GLOBAL	Use the global parameters.
/NOGLOBAL	Do not use the global parameters.

ERROR

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

Display statistical errors at each channel of a one dimensional spectrum.

/ERROR	Statistical errors (the square root of the count rate) are displayed.
/NOERROR	No errors are displayed

Function

The specified spectrum is displayed with the specified switches and modes. If 2-dim. spectra should be displayed in cluster or isometric mode the cuts and rotation angles can be modified:

a) in CLUSTER,CONTOUR mode an arbitrary number of cuts could be defined with "CUTS=...". The cuts are interpreted in relative units of the difference between the spectrum maximum and minimum.

b) in ISOMETRIC mode rotaion angles about the x-axis "THETA" and the y-axis "PHI" could be specified. Any values between -360.0 and + 360.0 are allowed.

DISPLAY TEXT

DISPLAY TEXT text frame xposition yposition font size
/CENTER /LEFT /RIGHT [=LOCATE]
/ABSOLUTE /RELATIVE [=UNIT]

PURPOSE Display text into box specified by cursor

PARAMETERS

text String containing text.

frame Frame into which the the text should be placed.

xposition X-position where the text string should be placed.

yposition Y-position where the text string should be placed.

font GKS text font number for text string.

size Factor to increase the text-size.

LOCATE Specifies the location of the text.

/LEFT	Specified position is the lower left
/RIGHT	Specified position is the lower right
/CENTER	Specified coordinate is the center

UNIT Specifies in which units the coordinates are given.

/ABSOLUTE	The coordinates are in absolute units.
/RELATIVE	The coordinates are in relative units.

Caller MDISP,MGOODISP

Author W. Spreng

Example

1.) DISPLAY TEXT "sdf sdf" 1

The cursor appears to specify the reference point in frame 1. The reference point is at the lower left corner of the text.

2.) DISPLAY TEXT "sdf sdf" xpos=100 ypos=200/CEN

The cursor appears to specify the frame. The text is centered.

3.) DISPLAY TEXT "sdf sdf" ypos=200/RIGHT

The cursor appears to specify the frame and the x-coordinate of the reference point. The specified point lies at the lower right edge of the text box.

4.) DISPLAY TEXT "sdf sdf" xpos=0.5 ypos=0.5/REL

The reference point lies in the middle of the physical screen.

Remarks

File name D$DTEXT.PPL

Created by D$DSPCM.PPL

Description

CALLING STS=D$DTEXT(CV_text,L_frame,R_xposition,R_yposition,
L_font,R_size,CV_locate,CV_unit,B_mask)

COMMAND DISPLAY TEXT text frame xposition yposition font size
/CENTER /LEFT /RIGHT [=LOCATE]
/ABSOLUTE /RELATIVE [=UNIT]

TEXT

Routine arg. Input CHAR(*) VAR

Command par. String required

String containing text. If text contains blanks, it has to be enclosed in apostrophes.

FRAME

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable

Frame into which the the text should be displayed. If the text should be placed in absolute units a valid frame number is required, because it is necessary to know in which coordinate system the text position should be interpreted and in which part of the picture it has to be drawn.

XPOSITION

Routine arg. Input BIN FLOAT(24)

Command par. FLOAT

X-position where the text string should be placed.

The X/Y-coordinates could be given in absolute coordinates according to the actual axis in the specified frame. Or in relative coordinates, given in units of the height and width of the total picture. If the coordinates are given in relative units the text could be placed anywhere one the screen, not only in a single frame!

YPOSITION

Routine arg. Input BIN FLOAT(24)

Command par. FLOAT

Y-position where the text string should be placed.

The X/Y-coordinates could be given in absolute coordinates according to the actual axis in the specified frame. Or in relative coordinates, given in units of the height and width of the total picture. If the coordinates are given in relative units the text could be placed anywhere one the screen, not only in a single frame!

FONT

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable default=0

GKS text font number for text string. Supported text-fonts are:

0	Hardwaretext
-1...-11	proportional
-101..-111	proportional italics
-201..-211	mono spaced
-301..-311	mono spaced italics

SIZE

Routine arg. Input BIN FLOAT(24)

Command par. FLOAT default=1.0

Factor to increase the text-size. This factor is relative to the minimum text height in the specified frame. E.g. a size of 2 increases the hight of the text to the double heigth of the lowest text height in the picture.

LOCATE

Routine arg. Input CHAR(*) VAR

Command par. Set default = /LEFT

Specifies the location of the text relativ to the X/Y-coordinates. Possible inputs are:

/LEFT	The specified position is the lower left corner of the text box.
/RIGHT	The specified position is the lower right corner of the text box.
/CENTER	The specified coordinate should be in the middle of the text box.

UNIT

Routine arg. Input CHAR(*) VAR

Command par. Set default = /ABSOLUTE

Specifies in which units the coordinates are given. Possible inputs are:

/ABSOLUTE	The coordinates are in spectrum units as indicated at the actual displayed axis in the specified frame. Therefore absolute units are defined only in one frame!
/RELATIVE	The coordinates are in relative units of the height and width of the total picture. (X,Y)=(0.0,0.0) is the lower left screen edge; (X,Y)=(1.0,1.0) is the upper right screen edge.

The ABSOLUTE units are only defined in one frame which has to be specified. The RELATIVE units describe a position on the total display-screen and are therefore picture independent!

FUNCTION

A text string is written into the picture displayed on the screen. The text string could be placed into a single frame, then the specified coordinates for the text reference point are interpreted in units of the actual axis in the specified frame. This mode is active if /ABSOLUTE has been set. The disadvantage of this mode is its correlation to the actually displayed picture!

But the text position can be specified in relative screen coordinates, if /RELATIVE is set. Then the reference point is interpreted in relative units of the screen height and width! In that mode the frame number is ignored. The relative position of e.g. (x,y) = (0.0,0.0) is the lower left screen edge.

If the frame number and/or the coordinates of the reference point are not specified they are promted via cursor input.

The text could be placed in different modes relative to the reference point:

/LEFT	The reference point is the lower left corner of the text box.
/RIGHT	The reference point is the lower right of the text box.
/CENTER	The reference point lies in the middle of the text box.

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 SPECTRUM

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

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

PARAMETERS

name name of spectrum (wildcard)
default = '*'

spec_dir default directory
required common default = '$SPECTRUM'

database default base
required common default = 'DB'

node default node
required common default = 'E'

output output file
required common default = 'OUTFILE'

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

Caller E$DMPCM

Author D. Schall

File name E$DMPSP.PPL

Dataset -

EXAMPLE DUMP SPEC ALPHA OUTP=ALPHA1.DATA
dump the spectrum 'ALPHA' to the file 'ALPHA1.DATA'
in ASCII format for transfer to SATAN VSAM library on the IBM

Remarks

REMARKS -

Description

CALLING STS=E$DMPSP(spec_name,spec_dir,database,node,
outfile,i_keep_map)

ARGUMENTS

spec_name name of spectrum. Wildcards are supported:
* x* *x *x* x*y
Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

spec_dir default Directory
replace default='$SPECTRUM'

database default Data Base
replace default='DB'

node default node
replace default='E'

outfile output file for spectra

/[NO] KEEP_MAP inhibits unmapping (detach) of database default /KEEP_MAP

FUNCTION the specified spectra are accessed and written to a file 'outfile' in SATAN AREAD input format with IBM Jobcontrol DD statements as separators.The first part of the file contains a list of the DD names of the dumped spectra. The spectra can be written to a SATAN VSAM analyzer library with the DCL command SIBMSPEC .

REMARKS -

EXAMPLE -

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

EXPAND

EXPAND limits frame
/CHAN/CALIB [=CALIBR]
/LOG/LIN/SQRT [=SCALE]
/XLOG/XLIN/XSQRT [=X_SCALE]
/YLOG/YLIN/YSQRT [=Y_SCALE]

PURPOSE Expand spectrum or scatterplot within specified window.

PARAMETERS

limits Limits for expansion range.

frame number of frame to be expanded.

CALIBR Specifies if the limits should be interpreted as spectrum cannels or as calibrated units.

/CHAN	Channels are specified.
/CALIB	Calibrated units are specified.

This two switches are valid inputs.

SCALE Modify the current display mode on the scaling axis.

/LIN	Linear scaling axis.
/LOG	Logarithmic Scaling axis.
/SQRT	Square root scaling axis.

X_SCALE Modify the current display mode on the X-axis.

/XLIN	Linear X-axis.
/XLOG	Logarithmic X-axis.
/XSQRT	Square root X-axis.

Y_SCALE	Modify the current display mode on the X-axis.	
	/YLIN	Linear Y-axis.
	/YLOG	Logarithmic Y-axis.
	/YSQRT	Square root Y/-axis.

Caller	MDISP,MGOODISP,D$DSPCM
Action rout.	D$DEXP
Author	W.Spreng

EXAMPLE

1.) EXPAND (100,300) 3
=> one dim spectrum in frame 3 is expanded within the specified limits. If in frame 3 is a two dimensional spectrum the y-range is unchanged.
2.) EXPAND (100,300,400,800) 3
=> expansion range of x-axis is 100 to 300. Expansion range on y-axis is 400 to 800 if a 2-dim spectrum or a scatterplot is in frame 3. For 1-dim spectra this y-range will be ignored.
3.) EXPAND (,300) 3
=> every value in the array specification could be set to the actual value if it is not specified. In this example the lower x-value is unchanged.
4.) EXPAND (100,,,700) 3
=> in this example x-max and y-min values are unchanged.
5.) EXPAND (100,300,400,800) *
=> ALL Frames on screen should be expanded. expansion range of x-axis is 100 to 300. Expansion range on y-axis is 400 to 800 if a 2-dim spectrum or a scatterplot is in frame 3. For 1-dim spectra this range will be ignored.
Cursor inputs:
6.) EXPAND
=> cursor appears to define expansion limits in one frame. Two limits have to be set.
7.) EXPAND frame=3
=> cursor appears to define expansion limits in frame 3. Two limits have to be set.
8.) EXPAND (0,100)
=> If more than one frame is on screen cursor appears to select the frame.
9.) EXPAND frame=3:6/LOG
=> Cursor appears to specify limits and the frames 3 to 6 are expanded and the scaling axis is displayed in logarithmic mode.
The limits could be specified as channels or as calibrated units. If calibrated units should be used but the spectrum is not connected to a calibration an error is signaled.

Remarks

File name D$DEXP.PPL

Created by GOO$DISP:D$DSPCM.PPL

Description

CALLING STS=D$DEXP(CV_limits,CV_frame,CV_cal,
cv_scale,cv_xscale,cv_yscale,b_mask)

COMMAND EXPAND limits frame
/CHAN/CALIB [=CALIBR]
/LOG/LIN/SQRT [=SCALE]
/XLOG/XLIN/XSQRT [=X_SCALE]
/YLOG/YLIN/YSQRT [=Y_SCALE]

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String

Limits for expansion range. Possible inputs are is any GOOSY display window specification:
E.g. (100,200,400,700) for 2.-dim spectra
(100,300) for 1.- dim spectra
every position could be set to the actual display value by two subsequent commas. If no limits are given the cursor prompts for input. See the command examples.

FRAME

Routine arg. Input CHAR(*) VAR

Command par. String

Number of the frame in which the expansion should be performed. Valid inputs are:
n : a single frame number as shown on screen
n:m : a range of valid frames
* : for all frames on screen
If devices with no local ereasure facility are active (e.g. TEKTRONIX 4014 or any plotter), and the standard GOOSY display is used do not specify a range of frames! Because in that case the whole picture is

completely redrawn for each specified frame. If all frames are specified the whole picture will be deleted and all frames are redrawn at once.

If no frame is specified the cursor appears to select a frame.

CALIBR

Routine arg. Input CHAR(*) VAR

Command par. Set default=/CHAN

Specifies if the limits should be interpreted as spectrum cannels or as calibrated units.

/CHAN	Channels are secified.
/CALIB	Calibrated units are specified

SCALE

Routine arg. Input CHAR(*) VAR

Command par. SET

Changes the display mode of the scaling axis in the expanded frames

/LIN	to LINEAR
/LOG	to LOGARITHMIC
/SQRT	to SQUARE ROOT

X_SCALE

Routine arg. Input CHAR(*) VAR

Command par. SET

Changes the display mode of the X-axis in the expanded frames

/XLIN	to LINEAR
/XLOG	to LOGARITHMIC
/XSQR	to SQUARE ROOT

Y_SCALE

Routine arg. Input CHAR(*) VAR

Command par. SET

Changes the display mode of the Y-axis in the expanded frames

/YLIN	to LINEAR
/YLOG	to LOGARITHMIC
/YSQRT	to SQUARE ROOT

FUNCTION

The spectrum or scatterplot in frame "frame" is expanded within the specified limits. Following inputs are valid:

1. frame = "
=> The cursor appears to specify the frame
2. limits = "
=> the cursor appears to specify the expansion range in frame "cv_frame"
3. limits = " and frame = "
=> the cursor appears to specify the expansion range in one arbitrary frame.
4. limits and frame specified
=> expansion is performed with no futher input

It is possible to expand several or all frames with the limits directly specified or selected by cursor input. A range of several frames is given by N:M all frames are expanded with 'frame'=*.

The limits could be specified as channels or as calibrated units. If calibrated units should be used but the spectrum is not connected to a calibration an error is signaled.

FIT SPECTRUM

```
FIT SPECTRUM frame poly window iter file
  /[NO]OUTPUT
  /[NO]APPEND
  /BACKGROUND
  /GAUSS
  /SAMEWIDTH
  /SHOW
  /[NO]ZERO
  /[NO]MARK
  /NOERROR /STATISTICAL [=ERROR]
```

PURPOSE	Fit spectrum with polynom and/or with gaussian peaks
PARAMETERS	
frame	Frame number to specify the spectrum used for the fit.
poly	Power of background polynom
window	Window to specify data region for gaussian fits.
iter	Maximum number of iterations.
file	File for the output of the results.
/[NO] OUTPUT	Output occus additionally onto the specified file.
/[NO] APPEND	Results are appended to an existing file.
/BACKGROUND	Fit background polynom
/GAUSS	Fit gaussian peaks
/SHOW	Show intermediate results of each iteration.
/SAMEWIDTH	All peaks are assumed to have same width.
/ZERO	Use points close to background too.

/[NO] MARK	Mark grphical input point by polylines.	
	/MARK	All points are indicated.
	/NOMARK	The inputs are not indicated.
ERROR	Error wightening	
	/STATISTICAL	Statistical errors are used.
	/NOERROR	No errors are taken into account.

Caller MDISP

Author W. Spreng

Remarks

File name GOO$DISP:D$FISP.PPL

Created by GOO$DISP:D$DSPCM.PPL

Description

CALLING

```
STS=D$FISP(L_frame,L_poly,CV_window,L_iter,
           CV_file,I_output,I_append,
           I_background,I_gauss,I_samewidth,I_show,
           I_mark,I_zero,CV_error,B_mask);
```

COMMAND

```
FIT SPECTRUM frame poly window iter file
  /[NO]OUTPUT
  /[NO]APPEND
  /BACKGROUND
  /GAUSS
  /SAMEWIDTH
  /SHOW
  /[NO]ZERO
  /[NO]MARK
  /NOERROR /STATISTICAL [=ERROR]
```

FRAME

Routine arg. Input BIN FIXED(31)

Command par. Integer; Default=1

The spectrum in the specified frame is used to perform the required fit.

POLY

Routine arg. Input BIN FIXED(31)

Command par. Integer; Default=1

Specifies the maximum power of the polygon, which is fitted to the spectrum data.

WINDOW

Routine arg. Input CHAR(*) VAR

Command par. String; Default=*

Window specification for the data region used to limit the data used to perform the gaussian fits. Following inputs are possible:

* Use the whole displayed region
? or " Specify window with cursor input.
(n,m) Use data between limit n and m.

ITER

Routine arg. Input BIN FIXED(31)

Command par. Integer; Default=10

Specifies the maximum number of iterations, which should be performed.

FILE

Routine arg. CHAR(*) VAR

Command arg. String replacable default=GOOSY_RESULT.LOG

Name of a file onto which the results should be written. To direct the outputs to that file the /OUTPUT switch is required!

OUTPUT

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch, default = /NOOUTPUT

Write results into the specified file.

/OUTPUT	The output is directed onto the file.
/NOOUTPUT	results are only written to terminal and into the session logfile.

APPEND

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch, default=/APPEND

Append the output to an existing file.

/APPEND	output is appended to an existing file.
/NOAPPEND	A new output file is created.

BACKGROUND

Routine arg. Input BIN FIXED(15); valid inputs are 0 and 1

Command par. Switch

A polygon should be fitted to the spectrum data. The power of the polygon is specified by the "poly" parameter! The cursor appears and up to 100 windows. The spectrum data in these windows are used for the fit.

If additionally gaussian peaks should be fitted, this polygon is used as the spectrum background.

GAUSS

Routine arg. Input BIN FIXED(15); valid inputs are 0 and 1

Command par. Switch

Gaussian peaks should be fitted to the spectrum data. The cursor appears to specify up to 100 data windows, which are used to fit a single gaussian peak. The peaks must be well separated.

SAMEWIDTH

Routine arg. Input BIN FIXED(15); valid inputs are 0 and 1

Command par. Switch

If specified the peak width is prompted only once. That width is used as a first guess for all peaks.

ITER

Routine arg. Input BIN FIXED(15); valid inputs are 0 and 1

Command par. Switch

If set the result of each iteration will be shown.

ZERO

Routine arg. Input BIN FIXED(15); valid inputs are 0 and 1

Command par. Switch

With this switch it is possible to ignore points which are not statistically significant about the background during the fit:

/ZERO	Use all data, even zeros!
/NOZERO	Ignore all points not clearly above the background.

All points in the range

ABS(Background-Data) < SQRT(Background)

are not used in the fit if zero suppression is required.

MARK

Routine arg. Input BIN FIXED(15); valid inputs are 0 and 1

Command par. Switch, default = /MARK

Mark graphical input points by polylines.

/MARK	mark the graphical input points.
/NOMARK	do not mark the inputs.

To control the acceptance of the graphical inputs it is recommended to use the /MARK switch. If nice pictures without any superfluous information should be produced, use the /NOMARK switch.

ERROR

Routine arg. Input CHAR(*) VAR

Command par. Set; Default=/STATISTICAL

Perform an error wightening during the fit. The spectrum data are wightened with their statistical errors.

/NOERROR No errors are taken into account in the fit.

/STATISTICAL The data are wightened with their statistical errors.

Function

Fit spectrum data with a polynom as background and an arbitrary number of well separated gaussian peaks. The spectrum displayed in the specified frame is used for the fit.

If /BACKGROUND is specified the cursor appears to specify background windows until the GKS-break facility is given at the graphical input device. Then a polynom with the power of 'poly' is fitted to the selected data regions, the result is stored in the resulting fit vector and the background is subtracted from the spectrum data.

If /GAUSS is specified the cursor appears again to specify the peak position and the peak width. The number of performed iterations is limited by the specified parameter 'ITER'. The results of each iteration can be controlled with the "/SHOW" switch.

The method applied fitting the background is described in Bevington page ???; for the gaussian peaks see page ???.

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

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

PURPOSE freeze a condition

PARAMETERS

name name of condition
required

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

Remarks

REMARKS	action routine

Description

CALLING	STS=E$FRECO(CV_NAME,CV_DIR,CV_BASE,CV_NODE,CV_ON, CV_OFF,I_KEEP_MAP)
ARGUMENTS	
CV_NAME	default directory CHAR(*) VAR Input
CV_DIR	default directory CHAR(*) VAR Input

CV_BASE default base
CHAR(*) VAR
Input

CV_NODE default node
CHAR(*) VAR
Input

CV_ON if on , result bit is set on
CHAR(*) VAR
Input

CV_OFF if on , result bit is set off, if on is on too, result bit will not be modified
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

FUNCTION The freeze bit in the analysis flag tables ([$ANLTABS]ANCO is the default) in which the specified condition lies, is set. This disables any subsequent execution of the condition by the analysis program or any dynamic list. The result bits are left untouched, except /ON or /OFF are specified. Then the preset values in the analysis tables are set accordingly, as well as the result bits.

REMARKS action routine

EXAMPLE -

FREEZE SPECTRUM

FREEZE SPECTRUM name spec_dir base node
/[NO]KEEP_MAP

PURPOSE freeze a spectrum, inhibit accumulation

PARAMETERS

name name of spectrum
required

spec_dir default Directory
replace default:'$SPEC'

base default Data Base
replace default:'DB'

node default node
replace default:'E'
replace default

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

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$FRESP

Dataset -

Remarks

REMARKS -

Description

CALLING STS=E$FRESP(CV_NAME,CV_DIR,CV_BASE,
CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_NAME name of spectrum
CHAR(*) VAR
Input

CV_DIR name of Directory
CHAR(*) VAR
Input

CV_BASE name of Data Base
CHAR(*) VAR
Input

CV_NODE name of node
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

FUNCTION The freeze bit for the corresponding spectrum in analysis tables is set. This bit is interrrogated each time when accumulation is attempted. If it is on , no accumulation will be done. You can reset this bit by using the command UNFREEZE

REMARKS Module is an action routine.

EXAMPLE -

INITIALIZE ACQUISITION

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 ANALYSIS

INITIALIZE ANALYSIS base1 base2 base3
/[NO]ANALYSIS
/[NO]UNPACK
/[NO]PACK
/[NO]START
/[NO]STOP
/[NO]BASE

PURPOSE Reinitialize analysis (Analaysis must be stopped)

base1 string
Data base to attached.

base2 string
Data base to attached.

base3 string
Data base to attached.

/NOANAL Disable calling of loaded analysis routine.

/ANAL Initialize and enable loaded analysis routine. The module must have been loaded with
LOAD MOD ANAL anal ianal /ANAL

/NOUNPACK Disable calling of loaded unpack routine.

/UNPACK Initialize and enable loaded unpack routine. The module must have been loaded with
LOAD MOD ANAL unpack iunpack /UNPACK

/NOPACK Disable calling of loaded pack routine.

/PACK Initialize and enable loaded pack routine. The module must have been loaded with
LOAD MOD ANAL pack ipack /PACK

/NOSTART Disable calling of loaded START routine.

/START	Initialize and enable loaded start routine. The module must have been loaded with LOAD MOD ANAL start istart /START
/NOSTOP	Disable calling of loaded STOP routine.
/STOP	Initialize and enable loaded stop routine. The module must have been loaded with LOAD MOD ANAL stop istop /STOP
/[NO] BASE	[No]base available

Description

FUNCTION	This command calls I$ANACM_DET to detach the analysis. Then it checks weather the specified modules have been loaded. START and STOP routines are initialized, if entries have been specified. Then I$ANACM_ATT is called to reinitialyze the analysis. The /NO... switches switch off the calling of the loaded module. Then the default modules are called.
File name	I$ANACM.PPL
Action rout.	I$ANACM_INIT
Version	1.01
Author	H.G.Essel
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

INTEGRATE

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

PURPOSE	Integrate specified window
PARAMETERS	
window	Limits of integration window
frame	Frame which should be integrated.
file	File for the output of the results.
/[NO] OUTPUT	Output occus additionally onto the specified file.
/[NO] APPEND	Results are appended to an existing file.
CALIBR	Specifies the unit in which the coordinates are given.
	/CHAN Original spectrum units.
	/CALIB Calibrated units.
/LOOP	Enter cursor loop to mark several points
Caller	MDISP,MGOODISP,D$DSPCM
Author	W. Spreng

Example

1.) INTEGRATE
=> The cursor appears to define integration limits in any frame on the screen.
2.) INTEGRATE fr=5

=> the cursor appears to fix the limits. Inputs are only accepted in frame 5.

3.) INTEGRATE 100,600

=> cursor appears to select frame

4.) INTEGRATE /LOOP

=> Cursor appears to define limits and frames. Inputs are accepted from any frame on screen. The cursor loop interrupts with the GKS-Break facility.

5.) INTEGRATE fr=5/loop

=> Cursor appears to define limits. Inputs are accepted from frame 5. The cursor loop interrupts with the GKS-Break facility.

6.) INTEGRATE 100,600 *

=> All frames on screen are integrated.

Remarks

File name D$INT.PPL

Created by GOO$DISP:D$DSPCM.PPL

Description

CALLING

```
STS=D$INT(CV_window,CV_frame,CV_file,I_output,
          I_append,CV_calibr,I_LOOP)
```

COMMAND

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

WINDOW

Routine arg. Input CHAR(*) VAR

Command par. String

Window which should be used as integration range. Any valid display window specification is allowed (see the GOOSY Display Manual):

(xmin,xmax) for a one dimensional window

(xmin,xmax,ymin,ymax) for a two dimensional window.

If nothing has been specified the cursor appears to specify the window limits.

The upper window limits are exclusive! E.g. for a one dimensional analog spectrum of binsize 1 the window (1,3) yields to an integration of two bins! Although value 3 belongs to the third bin.

FRAME

Routine arg. Input CHAR(*) VAR

Command par. String

Number of frames in which the coordinates should be be marked or selected. The spectrum in this frame is used to determine the channel contents. If no frame is specified and if more than one frame is on the screen, the cursor appears to pick the frame.

Beside the specification of a single frame the following inputs are valid:

N:M - Integrate frame N to M with the specified or selected limits.
* - Integrate all spectra on the screen within the specified or selected window.

FILE

Routine arg. CHAR(*) VAR

Command arg. String replacable default=GOOSY_RESULT.LOG

Name of a file onto which the results should be written. To direct the outputs to that file the /OUTPUT switch is required!

OUTPUT

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch, default = /NOOUTPUT

Write results into the specified file.

/OUTPUT	The output is directed onto the file.
/NOOUTPUT	results are only written to terminal and into the session logfile.

APPEND

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch, default=/APPEND

Append the output to an existing file.

/APPEND	output is appended to an existing file.
/NOAPPEND	A new output file is created.

CALIBR

Routine arg. Input CHAR(*) VAR

Command par. Set default="/CHAN"

If the spectrum in the selected frame is calibrated the input of the coordinates can occur in calibrated or uncalibrated units:

/CHANN	Coordinates of point are given in spectrum units.
/CALIB	Coordinates of point are in calibrated units

If the coordinates are specified via cursor input the units are known and this switch has no effect!

LOOP

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch

Enter cursor loop until break facility is given. If specified an arbitrary number of points in any frame could be marked.

FUNCTION

The spectrum in the specified frame is integrated within the specified limits. Possible inputs are:

1.) nothing specified
=> The cursor appears to define inegration limits in any frame on the screen.
2.) Frame specified
=> the cursor appears to fix the limits in that frame
3.) window specified
=> cursor appears to select frame

The window could be specified in calibrated units or in channels, depending on the given switch (/CHAN or /CALIB). If calibrated units are used but the spectrum in the specified frame is not connected to a calibration an error is signaled. If I_loop is set a cursor loop is entered which could be interrupted by GKS-Break facility.

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 MODULE ANALYSIS

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

PURPOSE Load module from sharable image.

PARAMETERS

image required character
name of sharable image for routines.

module required character
name of routine.

init character
name of initialization entry of the routine.
Arguments for /UNPACK and /PACK:
POINTER,BIN FIXED(31)
Pointer to event data element and length in bytes.
These arguments are zero until set by the commands
SET EVENT INPUT (for unpack)
SET EVENT OUTPUT (for pack)
Arguments for /START /STOP and /ANAL:
no arguments.

/START switch
The module is called at the beginning of
START INPUT ...
without arguments.

/STOP switch
The module is called at the end of
STOP INPUT ...
without arguments.

/UNPACK switch
The module is called for event unpacking
with one argument:
The pointer to GOOSY buffer or event, depending
on the mode SET ANAL /BUFFER or /EVENT

/PACK switch
The module is called for event packing
with one argument:
The pointer to GOOSY output buffer.

/ANAL switch
The module is called after event unpacking
without arguments.

EXAMPLE

```
$ LSHAR X$ANAL,X$START,X$STOP MYSHARE /GROUP
$ GOOSY
STOP INPUT MAIL
LOAD MOD ANAL MYSHARE X$ANAL $XANAL /ANAL
LOAD MOD ANAL MYSHARE X$START /START
LOAD MOD ANAL MYSHARE X$STOP /STOP
INIT ANAL /ANAL/START/STOP
```

NOTE The modules must be linked into a sharable image by DCL command LSHAR. The initialization entries are called by the
INIT ANAL command.

Description

FUNCTION Loads a sharable image and the specified module. The sharable image must be linked by command:
LSHARIM (see help).
The analysis must be stopped. The modules must
be enabled by INIT ANALYSIS to be called. Previous loaded modules are not called any longer.

File name I$ANACM.PPL

Action rout. I$ANACM_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)

LOCATE BASE

LOCATE BASE base
/[NO]KEEP_MAP

PURPOSE Locate a Data Base
For test purpose only.

PARAMETERS

base Data Base name
required common default

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

Caller M$DMCMD

Author M. Richter

File name M$ALODB.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=M$ALODB(CV_BASE, I_KEEP_MAP)

ARGUMENTS

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
FUNCTION	Locate a Data Base
REMARKS	Module is an action routine.
EXAMPLE	-

LOCATE DIRECTORY

LOCATE DIRECTORY dir base
/[NO]KEEP_MAP

PURPOSE Locate a Data Element Directory
For test purpose only.

PARAMETERS

dir Data Element Directory name
required common default

base Data Base name
required common default

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

Caller M$DMCMD

Author M. Richter

File name M$ALODI.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=M$ALODI(CV_DIR,CV_BASE,I_KEEP_MAP)

ARGUMENTS

CV_DIR	I Data Element Directory name CHAR(*) VAR
CV_BASE	I Data Base name CHAR(*) VAR
I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
FUNCTION	Locate a Data Element Directory name
REMARKS	Module is an action routine.
EXAMPLE	-

LOCATE ELEMENT

LOCATE ELEMENT name
/[NO]KEEP_MAP

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

PARAMETERS

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

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

Caller M$DMCMD

Author M. Richter

File name M$ALODE.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=M$ALODE(CV_NAME,I_KEEP_MAP)

ARGUMENTS

CV_NAME I Node::base:[dir]element-name(i)
CHAR(*) VAR

I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
FUNCTION	Locate a Data Element name array
REMARKS	Module is an action routine.
EXAMPLE	-

LOCATE ID

LOCATE ID element dir base
/[NO]KEEP_MAP

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

PARAMETERS

element ID of Data Element
replaced default:'1'

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

base Data Base name
required common default

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

Caller M$DMCMD

Author M. Richter

File name M$ALOID.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=M$ALOID(L_ELEMENT,L_DIR,CV_BASE,I_KEEP_MAP)

ARGUMENTS

L_ELEMENT I ID of Data Element
BIN FIXED(31)

L_DIR I ID of Directory
BIN FIXED(31)

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Locate a Data Element by Directory index

REMARKS Module is an action routine.

EXAMPLE -

LOCATE POOL

LOCATE POOL pool base
/[NO]KEEP_MAP

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

PARAMETERS

pool Name of Pool
required common default

base Data Base name
required common default

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

Caller M$DMCMD

Author M. Richter

File name M$ALOPO.PPL

Dataset -

EXAMPLE LOC POOL ADAM DB
locate the Pool 'ADAM' in the Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ALOPO(CV_POOL,CV_BASE,I_KEEP_MAP)

ARGUMENTS

CV_POOL	I Name of Pool CHAR(*) VAR
CV_BASE	I Data Base name CHAR(*) VAR
I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
FUNCTION	Locate a Pool in a Data Base
REMARKS	Module is an action routine.
EXAMPLE	-

LOCATE QUEUEELEMENT

LOCATE QUEUEELEMENT element
/[NO]KEEP_MAP

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

PARAMETERS

element Node::base:[dir]name(i)->type(i) required

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

Caller M$DMCMD

Author M. Richter

File name M$ALOQE.PPL

Dataset -

EXAMPLE LOC QUE DB:[EVE]ADAM->L(2)
locate the Data Element with the Data Type 'L'
and the name array index '2' queued to the Data Element 'ADAM' of Directory 'EVE' in the Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ALOQE(CV_ELEMENT,I_KEEP_MAP)

ARGUMENTS

CV_ELEMENT I Node::base:[dir]name(i)->type(i)
CHAR(*) VAR

I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
FUNCTION	Locate a queue Data Element name array
REMARKS	Module is an action routine.
EXAMPLE	-

LOCATE TYPE

LOCATE TYPE name base
/[NO]KEEP_MAP

PURPOSE Locate a Type descriptor
For test purpose only.

PARAMETERS

name Name of Type descriptor
required

base Data Base name
required common default

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

Caller M$DMCMD

Author M. Richter

File name M$ALOTY.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=M$ALOTY(CV_NAME,CV_BASE,I_KEEP_MAP)

ARGUMENTS

CV_NAME	I Name of Type descriptor CHAR(*) VAR
CV_BASE	I Data Base name CHAR(*) VAR
I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
FUNCTION	Locate a Type descriptor
REMARKS	Module is an action routine.
EXAMPLE	-

MODIFY DIRECTORY

MODIFY DIRECTORY dir entries base
/[NO]KEEP_MAP

PURPOSE Modify a Data Element Directory in a Data Base

PARAMETERS

dir Directory name
required common default

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

base Data Base name
required common default

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

Caller M$DMCMD

Author M. Richter

File name M$ARNDI.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=M$ARNDI(CV_DIR,L_DED_ENTRIES,CV_BASE, I_KEEP_MAP)

ARGUMENTS

CV_DIR I Directory name
CHAR(*) VAR

L_DED_ENTRIES I Number of entries for the new Data Element Directory
BIN FIXED(31)

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Renew (modify) a Data Element Directory in a Data Base. The size (number of entries) of a Data Element Directory can be changed.

REMARKS Module is an action routine.

EXAMPLE -

MODIFY FRAME SCATTER

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

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

PARAMETERS

picture	Name of picture frame
frame	Frame which should be modified.
xparam	Parameter for x-axis
yparam	Parameter for y-axis
limits	Limits of displayed spectrum or scatter plot.
condition	Main condition for that frame.
object	Parameter-list which should be checked
xletter	X-lettering on scatterplot axis.
yletter	Y-lettering on scatterplot axis.
node	Default node name for all Data Element name specifications.
base	Default Data Base name for all Data Elements
pic_dir	Default Directory for pictures.
par_dir	default Directory for parameter and condition object

cond_dir	default Directory for condition.
X_SCALE	Scaling mode for X-axis
	/XLIN Linear X-axis
	/XLOG Logarithmic X-axis
	/XSQRT Squareroot X-axis
Y_SCALE	Scaling mode for Y-axis
	/YLIN Linear Y-axis
	/YLOG Logarithmic Y-axis
	/YSQRT Squareroot Y-axis
/ROTATE	Rotate displayed spectra ********** not yet implemented *******
/LETTER	Display lettering
	/LETTER Display lettering on axis
	/NOLETTER Display no lettering
/NUMBER	Display numbering
	/NUMBER Display numbers on axis
	/NONUMBER Display no numbers on axis

Caller	MDISP,MGOODISP,D$DSPCM
Author	W. Spreng

Remarks

File name	D$MOSFR.PPL
Created by	GOO$DISP:D$DSPCM.PPL

Description

CALLING

STS=D$MOSFR(CV_picture,L_frame,CV_xparam,cv_yparam,
CV_limits,CV_condition,CV_object,CV_XLETTER,
CV_Yletter,cv_node,cv_db,
CV_pic_dir,CV_par_dir,CV_cond_dir,
CV_Xscale,CV_Yscale,I_rotate,CV_letter,CV_number,
B_mask)

COMMAND

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

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name specification of picture data element in which a scatter frame should be modified.

FRAME

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Number of the frame to be modified in the specified picture.

XPARAM,YPARAM

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

X- and Y-Parameter for scatter plot. If specified both parameter are required.

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=(0,1023,0,1023)

Limits of displayed scatterframe. A two dimensional window has to be specified:
(xmin,xmax,ymin,ymax)

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. String

Condition to filter the scatter data in this frame. The scatter data are send for that frame only if the result condition flag is true.

If an empty string is specified the actual condition is deleted from the picture data element.

OBJECT

Routine arg. Input CHAR(*) VAR

Command par. String

Condition objects for the specified condition. If not specified only the result flag of the specified condition is checked. If the specified the condition will be executed with the object. Therefore earlier checks of that condition are lost!

If an empty string is specified the actual object is deleted from the picture data element.

If an object is specified without any condition, it will be ignored.

XLETTER,YLETTER

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

X- and Y-lettering for scatter plot axis. If nothing is specified the parameter names are used

NODE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=*

Default node name.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=DB

Default Data Base where the picture is assumed.

PIC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=$PICTURE

Default Directory name for the specified picture.

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=DATA

Default Directory name for the specified parameter and condition object.

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=$CONDITION

Default Directory name for the specified condition.

X_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of X-axis. You can select between three modes:

/XLIN	Display the axis linear.

/XLOG	Display the axis in logarithmic mode.
/XSQRT	Display the square root of axis.

Y_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of Y-axis. You can select between three modes:

/YLIN	Display the axis linear.
/YLOG	Display the axis in logarithmic mode.
/YSQRT	Display the square root of axis.

ROTATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

********** Not yet implemented **********

LETTER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the lettering on the axis.

/LETTER	The lettering is displayed.
/NOLETTER	The lettering is not displayed.

NUMBER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the numbering on the axis.

/NUMBER	The numbering is displayed.
/NONUMBER	The numbering is not displayed.

Function

One picture frame is modified with the specified parameter set, the other frame settings are unaffected. Successive modifications are cumulative that means earlier parameter settings are not destroyed.

If one scatterplot event parameter is specified the other one is required! The scatter data can be filtered by a condition. If no condition object is given only the result flag will be checked, if it is true the data are sent to this frame. If an condition object has been found condition will be applied to that object with the concequence that the resulting flag of an earlier condition check is distroyed! Therefore be carefull specifying an object!

A spectrum frame can be changed to a scatter frame if no overlays are defined for the whole pictures!

MODIFY FRAME SPECTRUM

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

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

PARAMETERS

picture Name of picture frame

frame Number of frame or range of frames

spectrum Name of spectrum to be displayed in the frame.

limits Limits of displayed spectrum

scalim Limits of scaling axis

scalefactor Scaling factor to increase scaling axis.

node Default node name.

base	Default Data Base where the picture is assumed.	
pic_dir	Default picture Directory name.	
spec_dir	Default spectrum Directory name.	
scale	Scaling mode for Y- or Z-axis	
	/LIN	Linear scaling axis
	/LOG	Logarithmic scaling axis
	/SQRT	Squareroot scaling axis
X_SCALE	Scaling mode for X-axis	
	/XLIN	Linear X-axis
	/XLOG	Logarithmic X-axis
	/XSQRT	Squareroot X-axis
Y_SCALE	Scaling mode for Y-axis	
	/YLIN	Linear Y-axis
	/YLOG	Logarithmic Y-axis
	/YSQRT	Squareroot Y-axis
Z_SCALE	Scaling mode for Z-axis	
	/ZLIN	Linear Z-axis
	/ZLOG	Logarithmic Z-axis
	/ZSQRT	Squareroot Z-axis
WINDOW	Window switch	
	/NOWINDOW	Display no windows
	/WINDOW	Display all associated windows of spectrum
	********** not yet implemented *******	
LIFE	Life mode switch	
	/NOLIFE	No life mode
	/LIFE	life mode (update event by event)
	********** not yet implemented *******	
ROTATE	Rotate displayed spectra	

	/NOROTATE	No rotate
	/ROTATE	Rotate

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

SMOOTH Binnig mode

/SMOOTH	Smooth binning
/NOSMOOTH	min/max binning

LETTER Display lettering

/LETTER	Display lettering on axis
/NOLETTER	Display no lettering

NUMBER Display numbering

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

CALIB Perform calibration

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

CHANNELS Display channel numbers

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

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

/HISTO	Draw histograms
/VECTOR	Connect spectrum bins with lines
/MARKER	Sign spectrum contents with markers

For two dimensional spectra:

/CONTOUR	Contour lines are displayed
/CLUSTER	Clusters indicating the count rate

	/ISO	Show spectrum as an isometric plot.
	/SCATTER	Simulate scatter plot data.
/[NO] ERROR	Draw statistical error bars for each bin.	
	/NOERROR	No error bars drawn.
	/ERROR	Error bars are drawn at each bin.

Caller MDBM,MGOODDM

Author W. Spreng

Example

1.) MODIFY FRAME SCATTER pic 1 [$spectrum]s1
Spectrum in frame 1 of picture "pic" has been changed to "s1".
2.) MODIFY FRAME SPECTRUM pic 4:9 [$spectrum]s1(1:5) /log/vector
Spectra "s1(1)...s1(5)" are placed into the frames 4,...,9.

Remarks

File name D$MODFR.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=D$MODFR(CV_picture, CV_frame, CV_spectrum, CV_limits, CV_scalim, R_scalefactor, cv_node, cv_base, CV_pic_dir, CV_spec_dir, CV_scale, CV_Xscale, CV_Yscale, CV_Zscale, I_window, I_life, I_rotate, I_smooth, CV_letter, CV_number, CV_calib, CV_channels, CV_style, I_error, B_mask)

COMMAND MODIFY FRAME SPECTRUM picture frame spectrum limits scalim scalefactor node base pic_dir spec_dir
/LIN /LOG /SQRT [=SCALE]
/XLIN /XLOG /XSQRT [=X_SCALE]
/YLIN /YLOG /YSQRT [=Y_SCALE]
/ZLIN /ZLOG /ZSQRT [=Z_SCALE]
/[NO]WINDOW
/[NO]LIFE
/[NO]ROTATE
/[NO]SMOOTH

/[NO]LETTER
/[NO]NUMBER
/NOCAL/CALAX/CALSPEC [=CALIB]
/[NO]CHANNELS
/HISTO/VECTOR/MARKER/CONTOUR-
/POINT/ISO/CLUSTER/SCATTER [=STYLE]
/[NO]ERROR

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable required

Name of the picture which should be modified.

FRAME

Routine arg. Input CHAR(*) VAR

Command par. String replaceable required

Number of the frame to be modified. If a subset of a spectrum array should be placed into subsequent frames, the frames could be defined like:

start:stop

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

Name of spectrum which should be put into the picture frame. If a subset of a spectrum array should be selected specify:

spectrum(start:stop)

In that case a frame range, with the same number of elements is required.

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String

Limits of displayed spectrum or scatterplot
(100,1024) for 1-dimensional spectra
(100,1024,500,4096) for 2-dimensional spectra

SCALIM

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

Fixed limits of scaling axis. With this parameter the absolute range of scaling axis could be specified.

SCALEFACTOR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

Factor to increase scaling axis. With this factor the upper limit of the scaling axis could be modified in relativ units,e.g. if "scafac"=2.0 the upper limits is increased by a factor of "2.0". This is usefull if overlayed spectra with dynamic offset are defined for that frame (see DEFINE OVERLAY command).

NODE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=*

Default node name.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=DB

Default Data Base where the picture is assumed.

PIC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=$PICTURE

Default Directory name for picture Data Elements.

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=$SPECTRUM

Default spectrum Directory

SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Set the display mode on the scaling axis. You can select between three modes:

/LIN	Display the linear count rate.
/LOG	Display the spectrum contents in logarithmic mode.
/SQRT	Display the square root of the spectrum contents.

X_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of X-axis. You can select between three modes:

/XLIN	Display the axis linear.
/XLOG	Display the axis in logarithmic mode.
/XSQRT	Display the square root of axis.

Y_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of Y-axis. You can select between three modes:

/YLIN	Display the axis linear.
/YLOG	Display the axis in logarithmic mode.
/YSQRT	Display the square root of axis.

Z_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of Z-axis. You can select between three modes:

/ZLIN	Display the axis linear.
/ZLOG	Display the axis in logarithmic mode.
/ZSQRT	Display the square root of axis.

WINDOW

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

********** Not yet implemented **********

LIFE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

********** Not yet implemented **********

ROTATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

********** Not yet implemented **********

SMOOTH

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

To optimize the displayed spectrum data, the display reduces the number of displayed bins by an internal display binsize. The spectrum bins can be gathered in to ways:

/SMOOTH	Display the spectra with smooth binning. In that mode the mean values of the channel contents over the display binsize will be shown. The effect is that the spectrum looks smoother, but the displayed spectrum contents could be fractional numbers.
/NOSMOOTH	The minimum and maximum contents of the display bins are shown. In that modes spikes in the spectra are not smoothed out.

LETTER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the lettering on the axis.

/LETTER	The lettering is displayed.
/NOLETTER	The lettering is not displayed.

NUMBER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the numbering on the axis.

/NUMBER	The numbering is displayed.
/NONUMBER	The numbering is not displayed.

CALIB

Routine arg. Input CHAR(*) VAR

Command par. Set

Display the spectrum in calibrated units. To do this the displayed spectra has to be connected to an existing calibration. Different calibration modes are availible:

/CALAX	The axis is drawn in calibrated units and the spectrum in uncalibrated units. Therefore the distance between two subsequent tics varies.
/CALSPEC	The spectrum is drawn in calibrated units. Then the width of the displayed spectrum bins varies.
/NOCAL	No calibration is performed to the displayed spectra.

In any case the axis with uncalibrated units is displayed too. To prevent this specify the /NOCHANNELS switch!

/CHANNELS

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactivate the display of an axis with the original channel units.

/CHANNELS	Display an axis with original units.
/NOCHANNEL	The axis with original units is not not displayed.

STYLE

Routine arg. Input CHAR(*) VAR

Command par. Set

Defines the display style of the spectrum data. For one and two dimensional spectra different styles are implemented.

One dimensional spectra:

/HISTO	Histograms are generated
/VECTOR	The spectrum contents are connected by polylines.
/MARKER	The spectrum contents are indicated by markers.

For two dimensional spectra:

/CONTOUR	Contour lines are displayed.
/CLUSTER	The spectrum count rates are indicated by clusters of a variable size and colour.
/ISO	A 3D isometric plot is generated.
/SCATTER	The countrate in each bin is simulated by a number of randomly distributed points in that bin. The result is a representation similiar to a scatter plot.

ERROR

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

Display statistical errors at each channel of a one dimensional spectrum.

/ERROR	Statistical errors (the square root of the count rate) are displayed.
/NOERROR	No errors are displayed

Function

One picture frame is modified with the specified parameter set, the other frame settings are unaffected. Successive modifications are cumulative that means earlier parameter settings are not destroyed.

Several members of an array of spectra are put into subsequent frames if a subset of spectra and range of frames is specified, like:

MODIFY FRAME SPECTRUM picture 1:8 spectrum(3:10)

It is possible to change the dimension of the spectrum in the specified frame, if no overlayed spectra are defined. Changes from scatter to spectrum frames are allowed too, with the restriction that no overlays are defined for the whole picture! For the scaling axis fixed limits are defined with 'scalim' or you can set a factor 'scafac' to increase the maximum limit of the scaling axis in dependence of the actual size of the spectrum maximum. This is useful if for e.g. if the contents of the spectrum which should be displayed increases, but the ratio between the maximum channel contents and the maximum of the scaling axis should be fixed.

MODIFY TABLE CONDITION

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

PURPOSE modify condition bit table

PARAMETERS

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

entries integer required:
number of needed conditions

directory string replace default: '$ANLTABS'
default directory

pool string replace default: '$COND_POOL'
default pool

base string replace default: 'DB'
default data base

node string replace default: 'E'
default node

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

Caller MDBM,MGOODBM

Author Th. KROLL

EXAMPLE

```
MODIFY TABLE CONDITION ENTRIES=4096
```

REMARKS

File name GOO$DE:E$AMOCT.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$AMOCT(CV_NAME,L_ENTRIES,CV_DIR,CV_POOL,
CV_BASE,CV_NODE,I_KEEP_MAP)

COMMAND MODIFY TABLE CONDITION name entries
directory pool base node
/[NO]KEEP
Argument /parameter description

NAME

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'ANCO'

ENTRIES

Routine arg. Input BIN FIXED(31)

Command par. integer required:

DIRECTORY

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$ANLTABS'

POOL

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$COND_POOL'

BASE

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'DB'

NODE

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'E'

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP
In a procedure call this argument should be always 1 to prevent an unmap of the data base.

FUNCTION

Modify condition bit table to specified size

MODIFY TABLE SPECTRUM

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

PURPOSE modify spectrum bit table

PARAMETERS

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

entries integer required:
number of needed spectras

directory string replace default: '$ANLTABS'
default directory

pool string replace default: '$SPEC_POOL'
default pool

base string replace default: 'DB'
default data base

node string replace default: 'E'
default node

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

Caller MDBM,MGOODBM

Author Th. KROLL

EXAMPLE

MODIFY TABLE SPECTRUM ENTRIES=4096

REMARKS

File name GOO$DE:E$AMOST.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$AMOST(CV_NAME,L_ENTRIES,CV_DIR,CV_POOL,
CV_BASE,CV_NODE,I_KEEP_MAP)

COMMAND MODIFY TABLE SPECTRUM name entries
directory pool base node
/[NO]KEEP
Argument /parameter description

NAME

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'ANSP'

ENTRIES

Routine arg. Input BIN FIXED(31)

Command par. integer required:

DIRECTORY

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$ANLTABS'

POOL

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$SPEC_POOL'

BASE

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'DB'

NODE

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'E'

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP
In a procedure call this argument should be always 1 to prevent an unmap of the data base.

FUNCTION

Modify Spectrum bit table to specified size

MOUNT BASE

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

PURPOSE Mount an existing Data Base (section)

PARAMETERS

base Data Base name
required common default

basefile Data Base file name
required common default

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

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

Caller M$DMCMD

Author M. Richter

File name M$AMODB.PPL

Dataset -

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

Remarks

REMARKS -

Description

CALLING STS=M$AMODB(CV_BASE,CV_BASEFILE,
CV_PERMANENT,CV_GLOBAL)

ARGUMENTS

CV_BASE I Data Base name
CHAR(*) VAR

CV_BASEFILE I Data Base file name
CHAR(*) VAR

CV_PERMANENT I Section permanence
CHAR(*) VAR

CV_GLOBAL I Section scope
CHAR(*) VAR

FUNCTION Mount an existing Data Base (section)

REMARKS Module is an action routine.

EXAMPLE -

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

OVERLAY

```
OVERLAY spectrum xpara ypara binfactor
          trans=(xfactor,xoffset,yfactor,yoffset)
          frame node base spec_dir par_dir
        /ADJUST
        /SAVE
        /[NO]ERROR
        /HISTO/VECTOR/MARKER - =[STYLE]
        /ISO/CLUSTER/CONTOUR/SCATTER
```

PURPOSE Add spectrum to frame

PARAMETERS

spectrum	Name of new spectrum, the spectrum could be one or two dimensional.
xpara	X-parameter for additional scatterplot parameters
ypara	Y-parameter for additional scatterplot parameters
binfactor	Binning factor.
xoffset	Offset X-direction for one dim. spectra
xfactor	Factor X-direction for one dim spectra
yoffset	Offset Y-direction for one dim spectra
yfactor	Factor Y-direction for one dim spectra
frame	Number of frame
node	Default node name
base	Default Data Base name
spec_dir	Default spectrum Directory
par_dir	Default parameter Directory

/ADJUST	New spectrum is adjusted by cursor
/SAVE	Save the defined overlays for the specified frame in the picture Data Element.

Example

1.) OVERLAY spec fr=4
Spectrum "spec" is overlayed in frame 4
2.) OVERLAY spec TRANS=1.5,100,2.0,-500 9 /save
Spectrum "spec" is added to frame 9 with the transformation factors:
x_fac = 1.5 x_shift = 100
y_fac = 2.0 y_shift = -500
and additionally the defined overlays are strored in the Data Base if a picture Data Element is on the screen.
3.) OVERLAY spec fr=2/adjust
The cursor appears to define a linear transformation before "spec" is added to frame 2. 4.) OVERLAY spec fr=2/NOERROR/MARKER
Spectrum is displayed in MARKER mode and without errorbars.
5.) OVERLAY xparam=[$event]x.geli(10) yparam=[$event]y.geli(9) frame=5
The specified scatterplot parameters are additionally displayed in the scatterframe 5.

Caller	MDISP,MGOODISP,D$DSPCM
Author	W. Spreng

Remarks

File name	D$DOVER.PPL
Created by	D$DSPCM.PPL
REMARKS	The dimensions of the original spectrum in the specifeid frame and of the overlayed spectrum have to be the identical.

Description

CALLING

```
STS=D$DOVER(CV_spectrum,CV_xpara,CV_ypara,
      RA_trans,L_frame,
      CV_node,CV_base,CV_spec_dir,CV_par_dir,
      I_adjust,I_save,I_error,CV_style,LA_dim,B_mask)
```

COMMAND OVERLAY spectrum xpara ypara binfactor
trans=(xfactor,xoffset,yfactor,yoffset)
frame node base spec_dir par_dir
/ADJUST
/SAVE
/[NO]ERROR
/HISTO/VECTOR/MARKER - =[STYLE]
/ISO/CLUSTER/CONTOUR/SCATTER

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. String

Name of spectrum, which should be additionally displayed in a frame. The dimension of the original frame spectrum and of the specified spectrum have to be the same.

XPARA,YPARA

Routine arg. Input CHAR(*) VAR

Command par. String

The X and Y-parameter for additional scatterplot correlations, which should be displayed in the specified frame. Both parameter have to be specified.

BINFACTOR

Routine arg. Input BIN FIXED(31)

Command par. Integer, default=0

Binning Factor. If larger than "0" it specifies the number of bins to be summed up in the display. This corresponds to a temporarily modification of the spectrum binsize. E.g. if the binning factor is 2 the count rate in one displayed spectrum bin is given by the sum of two neighbouring bins in the original spectrum!

If =0 an internally calculated display binsize is used to optimize the displayed data.

TRANS

Routine arg. Input (*) BIN FLOAT(24)

Command par. Real array default=(1.0,0.0,1.0,0.0)

Defines a linear transformation for overlayed spectra in the horizontal (x) and vertical (y) axis:
(xfactor,xoffset,yfactor,yoffset)
This transformation is applied to one dimensional spectra.

FRAME

Routine arg. Input (*) BIN FIXED(31)

Command par. Integer replaceable

Number of the frame in which the overlays should be shown. If more than one frame is on the screen the frame has to be specified.

The type of this frame must be consistent with the specified spectrum or scatter-parameter. E.g. it is impossible to define scatter parameter for a spectrum frame.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=*

Default node name

BASE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=DB

Default Data Base name

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=$SPECTRUM

Default spectrum Directory

PAR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=DATA

Default parameter Directory

ADJUST

Routine arg. Input BIN FIXED(15), valid values 0 and 1

Command par. Switch

New spectrum is adjusted by cursor. This switch is only valid for 1 -dim spectra. If it is specified the cursor appears to define two points in the original spectrum. After that the specified frame is deleted and the spectrum which should be overlayed is displayed. Again the cursor appears to define two points in that spectrum. This cursor inputs define a linear transformation applied to the overlayed spectrum.

SAVE

Routine arg. Input BIN FIXED(15), valid values 0 and 1

Command par. Switch

Save the defined overlays for the specified frame in the picture which is actually displayed. To save the defined overlays for all frames give the command:

OVERLAY /SAVE

To save the overlays for only one frame the frame number must be specified.

ERROR

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

Display statistical errors at each channel of a one dimensional spectrum.

/ERROR Statistical errors (the square root of the count rate) are displayed.

/NOERROR No errors are displayed

STYLE

Routine arg. Input CHAR(*) VAR

Command par. Set

Defines the display style of the spectrum data. For one and two dimensional spectra different styles are implemented.

One dimensional spectra:

/HISTO	Histograms are generated
/VECTOR	The spectrum contents are connected by polylines.
/MARKER	The spectrum contents are indicated by markers.

For two dimensional spectra:

/CONTOUR	Contour lines are displayed.
/CLUSTER	The spectrum count rates are indicated by clusters of a variable size and colour.
/ISO	A 3D isometric plot is generated.
/SCATTER	Scatter plot data are simulated. In each the countrate is indicated by a number of points, randomly distributed in the bin.

Function

Additional spectra or scatterparameters are added to one frame of the actually displayed picture.

For overlayed spectra (one and two dimensional) a transformation can be defined which is applied to the spectrum data before displaying them.

The two one-dim. spectra, which should be overlayed could be adjusted automaticly by setting '/ADJUST'. The cursor is then activated to get two points in the orginal spectrum. After that this frame will be deleted and the spectrum to be overlayed is shown. The cursor is activated again to get the corresponding points in that spectrum. With this coordinates a transformation in x-direction is defined which will be applied when overlaying the spectra.

Overlayed spectra are displayed in the modes as defined in the specified frame. The display style of the overlays can be modified, e.g:

OVERLAY spectrum /MARKER

Display the spectrum in marker mode. Futhermore it is possible to activate or deactivate the display of error bars.

To save the defined overlayed spectra or parameters for one frame in a Picture Data Element '/SAVE' has to be specified.

ATTENTION If in scatter frames additional scatter parameters should be displayed, specify first all parameters for all frames, then give the command:

OVERLAY/SAVE

which saves the additional scatter plots in your Data Base. After that the overlayed scatter plots are activated!

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,");

PLOT METAFILE

PLOT METAFILE file type command queue copies font

PURPOSE	Plot a metafile.
PARAMETERS	
file	VMS-file name for meta file. The default file type is ".MET".
type	The device type of the specified output device.
command	Optional print command (queue specification ignored) This argument is replaced. To clear it, specify " ".
queue	Queue-name of device at which the plotfile should be printed.
copies	Number of copies to produced. If a physical address is specified this argument is ignored.
font	Font for text in the picture
Caller	MDISP,MGOODISP
Author	W. Spreng

Example

1.) PLOT METAFILE test.met LN03 " " SYS$LN03_C
The metafile TEST.MET will be plotted on the LN03-PLUS Laser printer queue SYS$LN03_C.
2.) PLOT METAFILE test.met RP02 " "IBM::
Metafile test.met should be sent to IBM and plotted on the RP02 BENSON plotter.
3.) PLOT META test.met POST "PP A POST"
Format metafile to postscript format and print with PP command (DCL).

Remarks

Created by	D$DSPCM.PPL
File name	GOO$DISP:D$PLMET.PPL

Description

CALLING STS=D$PLMET(CV_file,CV_type,CV_command,CV_queue, CV_copies,L_font)

COMMAND PLOT METAFILE file type command queue copies font

FILE

Routine arg. CHAR(*) VAR

Command par. String required

VMS file-name of the metafile, default file type is ".MET". Take care that the file is really a metafile.

TYPE

Routine arg. CHAR(*) VAR

Command par. String required

Device type of plotter. Supported types are:

1.) LN03 laser printer
2.) HP7550A3,HP7550A4 pen plotters
3.) POST postscript
4.) for plots send to IBM the following plotter types are supported:
RP01 for the large BENSON plotter
RP02 for the small BENSON plotter
VP01 for the BENSON vector plotter.

COMMAND

Routine arg. CHAR(*) VAR

Command par. String optional

DCL command to print file. The picture is formatted into a file according type specification. Then this file is printed by command. The queue specification is ignored.
This argument is replaced. To clear it, specify " ".

QUEUE

Routine arg. CHAR(*) VAR

Command par. String required

Queue name or physical address of device. For available queue names at GSI use HELP PRINTER. It is possible to specify a physical address to copy the file to the device. In that case a colon has to be specified at the end, e.g. TXB4:

If a plot should be send to IBM the queue name has to be set to IBM::

COPIES

Routine arg. CHAR(*) VAR

Command par. String default = 1

Number of copies to produced. If a physical address is specified this argument is ignored.

FONT

Routine arg. CHAR(*) VAR

Command par. String default=0

Font number to change text fonts defined in the GKS-bundle tables which are used in case of hardware character requirement. The following GKS-font numbers are supported:

0	Hardware text font
-1...-11	proportional fonts.
-101..-111	proportional italics
-201..-211	mono spaced fonts
-301..-311	mono spaced italics

Function

This procedure plots a metafile on the specified plotter. First the device independent metafile is converted into a device dependent plotfile which is finaly sent to the specified queue or physical address. If in the queuename a colon is found it is assumed that a physical device address is specified and that the device is not spooled. Then plotfile is copied to the device. For the IBM a double colon is required.

PLOT PICTURE

PLOT PICTURE type command queue copies font file
/[NO]FLAG
/[NO]PRINT

PURPOSE	Send the current active picture to a plotter
PARAMETERS	
type	Device type of specified queue.
command	Optional print command (queue specification ignored) This argument is replaced. To clear it, specify " ".
queue	Optional queue-name of device at which the plotfile should be printed. Used, if no command was given.
copies	Number of copies to produced.
font	Font for text in the picture
font	Font for text in the picture
/[NO] FLAG	Print flag page
/[NO] PRINT	Avoid printing. A file should be specified.
Caller	MDISP,MGOODISP
Author	W. Spreng

Example

PLOT PICTURE ln03 queue=sys$ln03_a
sends one copy to LN03_plus plotter at queue
SYS$LN03_A.
PLOT PICTURE post "PS A POST"

Format postscript file and print with PS command
PLOT PICTURE color "PT A POST"
Format color postscript file and print with PT
PLOT PICTURE post "P A POST"
Format postscript file and print on LN03 printer A.
PLOT PICTURE lj250 "PI I 80"
Format inkjet file and print on LJ250 printer I.
PLOT PICTURE ln03 " " sys$ln03_a
The " " is necessary to clear a previous command.
PLOT PICTURE ln03 FILE=XXX.LN3 /NOPRINT
Store formatted picture in file without printing.

Remarks

Created by GOO$DISP:D$DSPCM.PPL

File name GOO$DISP:D$PLPIC.PPL

Description

CALLING STS=D$PLPIC(CV_type,CV_command,CV_queue,CV_copies,
L_font,CV_file,i_flag,i_print)

COMMAND PLOT PICTURE type command queue copies font file
/[NO]FLAG /[NO]PRINT

TYPE

Routine arg. CHAR(*) VAR

Command par. String required

Plotter type. Supported types are:
LN03 - Laser printer
POST - Postscript output format
COLOR - Color postscript output format
LJ250 - Color inkjet output format
SIXEL - Sixel output format
HP7550A3,HP7550A4 - colour pen plotter

COMMAND

Routine arg. CHAR(*) VAR

Command par. String optional

DCL command to print file. The picture is formatted into a file according type specification. Then this file is printed by command. The queue specification is ignored.
This argument is replaced. To clear it, specify " ".

QUEUE

Routine arg. CHAR(*) VAR

Command par. String optional

Used, if no print command was specified. Queue name or physical address of device. If a colon (":") is found it will be assumed that a physical address is specified. Available queue names at GSI: HELP PRINTER

COPIES

Routine arg. CHAR(*) VAR

Command par. String default = 1

Number of copies to produced. If a physical address is specified this argument is ignored.

FILE

Routine arg. CHAR(*) VAR

Command par. String optional

Optional filename to store formatted picture. Normally one uses /NOPRINT to avoid printing.

FONT

Routine arg. BIN FIXED(31)

Command par. Integer default=0

Font number to change text fonts defined in the GKS-bundle tables which are used in case of hardware character requirement.

NOTE That means, hardware font must be enabled, i.e.:
DEFINE FRAME SETUP 10 0 (is default after start)

GTS-GRAL GKS The following GKS-font numbers are supported:

0	Hardware text font
-1...-11	proportional fonts.
-51	thick proportional font
-101..-111	proportional italics
-151	thick proportional italics
-201..-211	mono spaced fonts
-251	thick mono spaced font
-301..-311	mono spaced italics
-351	thick mono spaced italics

Recommended is font -51.

DEC GKS The standard DEC software fonts are:

0	Hardware text font
-1,1	Standard ISO font (Thin, ugly).
-15	Thick Roman (But underscore is arrow)

Recommended is font -15.
With Postscript format you may use one of the following fonts:

-101..-104	Times (Roman, italic, Bold, Bold italic)
-105..-108	Helvet. (Roman,italic,bold,bold italic)
-109..-112	Courier (Roman,italic,bold,bold italic)
-114..-117	Lubalin (Roman,italic,bold,bold italic)
-118..-121	School (Roman,italic,bold,bold italic)
-122..-125	Av.Garde (Roman,italic,bold,bold italic)
-126..-129	Souvenir (Roman,italic,bold,bold italic)

/FLAG

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch replacable default=/noflag

If set, a flag page is printed

/PRINT

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch replacable default=/print

If set, picture is printed. /NOPRINT is used together with the file argument to store a picture in a file without printing.

Function

The current active GOOSY-picture is formatted in a file type 'type'. The file is printed on specified queue or by specified command. If a colon is found in the queuename it is assumed that a physical device is specified.

The data stored in the Workstation Independent Segment Storage (WISS) are associated to the plotter and a plotfile is generated. Finally this plotfile will be printed/copied to the specified device.

This command is not supported in the fast display version!

PLOT PLOTFILE

PLOT PLOTFILE file command queue copies
/[NO]DELETE
/[NO]FLAG

PURPOSE	Plot device specific plotfile
file	VMS-file name of plotfile.
command	Optional print command (queue specification ignored) This argument is replaced. To clear it, specify " ".
queue	Plotter queue name or physical device adress.
copies	number of copies to be made.
/[NO] DELETE	Delete plotfile after printing.
/[NO] FLAG	Print Flag page.
Caller	MDISP,MGOODISP
Author	W. Spreng

Example

PLOT PLOTFILE X.ln03 queue=sys$ln03_a
sends one copy to LN03_plus plotter at queue
SYS$LN03_A.
PLOT PLOT X.PS "PS A POST"
Print X.PS with PS command
PLOT PLOT X.PS "P A POST"
Print file on LN03 printer A.
PLOT PLOT X.ln03 " " sys$ln03_a
The " " is necessary to clear a previous command.

Remarks

Created by D$DSPCM.PPL

File name GOO$DISP:D$PLPFI.PPL

Description

CALLING

```
STS=D$PLPFI(CV_file,CV_command,CV_queue,CV_copies,
            I_delete,i_flag)
```

COMMAND

```
PLOT PLOTFILE file command queue copies
                /[NO]DELETE
                /[NO]FLAG
```

FILE

Routine arg. CHAR(*) VAR

Command par. String required

Plotfile name. Default file type is ".PLT" Wildcards are supported at any position in the file specification.

COMMAND

Routine arg. CHAR(*) VAR

Command par. String optional

DCL command to print file. The picture is formatted into a file according type specification. Then this file is printed by command. The queue specification is ignored. This argument is replaced. To clear it, specify " ". At GSI, there are the commands P, PP, PS and PI. Typing these DCL commands gives more information.

QUEUE

Routine arg. CHAR(*) VAR

Command par. String optional

Used, if no print command was specified. Queue name or physical address of device. If a colon (":") is found it will be assumed that a physical address is specified. Available queue names at GSI: HELP PRINTER

COPIES

Routine arg. CHAR(*) VAR

Command par. String default=1

Number of copies to be produced. If a physical address is specified this argument is ignored.

/DELETE

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch default=/delete

If set, the Plotfile will be deleted

/FLAG

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch replacable default=/noflag

If set, a flag page is printed

Function

If in the GOOSY-display process a plotter has been allocated as a spooled device the generated plotfile "file" could be sent to the device "queue".

If in the queuename a colon is found it is assumed that a physical device address is specified and that the device is not spooled. Then plotfile is copied to the device. Instead of a queue the complete print command can be specified. When the print command is P, PP, PS or PI, the print is done without spawning (GSI queues assumed). Otherwise the print command is spawned.

PRINT

PRINT command printer form file
/DELETE

PURPOSE	Plot device specific plotfile
command	GSI print command (P, PP, PS, PT or PI).
printer	GSI printer letter (A,B,C...)
form	GSI style (like in DCL commands)
file	File to be submitted or printed.
/DELETE	Qualifier, i.e. /DELETE

Description

CALLING	@CALL U$PRINT(CV_command,CV_printer,CV_form ,CV_file,CV_qualifier)
COMMAND	PRINT command printer sform file /DELETE

COMMAND

Routine arg.	CHAR(*) VAR
Command par.	String GSI print command (P, PP, PS, PT or PI). Instead of specifying printer and form separately, they may be specified with command, i.e. "P A 80"

PRINTER

Routine arg.	CHAR(*) VAR

Command par. String

GSI printer letter (A,B,C...) A list of available printers at GSI can be obtained by DCL command HELP PRINTER

FORM

Routine arg. CHAR(*) VAR

Command par. String

GSI style (like in DCL commands) A list of available forms at GSI can be obtained by DCL commands P PP PS or PI, respectively.

FILE

Routine arg. CHAR(*) VAR

Command par. String required

File to be printed.

/DELETE

Routine arg. CHAR(*) VAR

Command par. qualifier

/DELETE delete file after print

Function

FUNCTION Print file.

EXAMPLE

```
@CALL U$PRINT('P','A','POST','TEST.PS','/D')
  @CALL U$PRINT('P','A','80','TEST.TXT',")
  @CALL U$PRINT('P A 80',",",'TEST.TXT',")
  PRINT P A 80 test.txt
```

PROJECT

PROJECT spectrum target window dimension node base spec_dir
/ADD/SUB/CLEAR [=ACTION]
/POLYGON

PURPOSE	Project window in 2-dim spectrum
PARAMETERS	
spectrum	Source spectrum.
target	Target spectrum
window	Limits of the projection window.
dimension	Dimension onto which the projection has to be performed.
node	Default node name
base	Default Data Base name
spec_dir	Default spectrum Directory
ACTION	Add or subtract result of projection.
	/ADD Add projection
	/SUB Substract projection
	/CLEAR Clear source spectrum
/POLYGON	Interprete the window as a polygon name.

Examples

PROJECT two one_x

The two dimensional spectrum "two" is projected onto dimension 1 (default). The cursor appears to specify the projection window at the y-axis. The result is stored in spectrum one_x.

PROJECT two one_x win=* /ADD
Project total range of dimension 2 onto dimension 1 and add result to spectrum "one_x."

PROJECT two one_y win=(100,200) dim=2/SUB
Project specified window (on dimension 1) onto dimension 2 and subtract result from spectrum "one_y".

PROJECT two one_y WIN=polygon dim=2/POLYGON
The projection range is "polygon" and the projection occurs onto the y-axis.

Remarks

REMARKS Up to now only projections of two dimensional spectra are supported.

Created by D$DSPCM.PPL

Description

CALLING STS=D$PROJ(CV_spectrum, CV_target, CV_window, L_dimension, CV_node, CV_base, CV_dir, CV_action, I_polygon)

COMMAND PROJECT spectrum target window dimension node base spec_dir
/ADD/SUB/CLEAR [=ACTION]
/POLYGON

SPECTRUM

Routine par. Input CHAR(*) VAR

Command par. Required, replaceable
Multidimensional spectrum which should be projected onto a one dimensional spectrum.
=> Up to now only 2.-dim spectra are supported!<=

TARGET

Routine par. Input CHAR(*) VAR

Command par. Required, replaceable
Target spectrum. Spectrum for projection. This has to be a one dimensional spectrum. If it does not exist, it will be created with the attributes of "spectrum" in dimension "dimension".

WINDOW

Routine par. Input CHAR(*) VAR

Command par. Optional, replaceable

Limits of the projection window. This is a one dim. window. Specify it like:

(min,max) - minimum and maximum window limit
* - the whole spectrum range

The upper limit is exclusiv; e.g for a binsize 1 spectrum the window (1,2) is one bin; but (1,2.1) is the same as (1,3) which projects two bins (because all values >= 2 and <3 belong to the bin with the value 3)!

You can specify a polygon as a projection region. In that case this parameter is interpreted as a name of the polygon and the "/POLYGON" must be set. All bins for which the middle of the bin is in the polygon are considered during the projection.

DIMENSION

Routine par. Input BIN FIXED(31)

Command par. Integer Default=1

Dimension onto which the projection has to be performed.

NODE

Routine par. Input CHAR(*) VAR

Command par. Global replaceable default=*

Default node name

BASE

Routine par. Input CHAR(*) VAR

Command par. Global replaceable default=DB

Default Data Base name

SPEC_DIR

Routine par. Input CHAR(*) VAR

Command par. Global replaceable default=$SPECTRUM

Default spectrum Directory.

POLY_DIR

Routine par. Input CHAR(*) VAR

Command par. Global replaceable default=$POLYGON

Default polygon Directory.

ACTION

Routine par. Input CHAR(*) VAR

Command par. Set default= /CLEAR

Add or subtract result of projection.

/ADD	The result is added to the contents of the target spectrum
/SUB	The result is subtracted from the contents of the target spectrum.
/CLEAR	The result of the projection is directly stored in the target spectrum. Old spectrum data are lost.

POLYGON

Routine par. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch

If set the specified window is interpreted as the name of a polygon data element. The projection range is the area enclosed by this polygon.

Function

A two dimensional spectrum is projected onto a one dimensional spectrum. The axis onto which the projection should be performed could be specified with 'dimension'. The projection range could be specified in the following way:

(min,max) : minimum, maximum window limits

* : for the whole spectrum range

If no window is specified the cursor appears to select the limits in the specified spectrum.
The source and target spectrum have to be of compatible data types, i.e.:

D->D
R->R
L->L
I->I
I->L

Specifying the /POLYGON switch the window is interpreted as the Data Element name of a polygon. In that case the projection range is the area enclosed by the polygon. All spectrum bins for which the middle of the bin is in the polygon are considered during the projection.

With the projection data several actions could be performed

/ADD	The result is added to the contents of the target spectrum
/SUB	The result is subtracted from the contents of the target spectrum.
/CLEAR	The result of the projection is directly stored in the target spectrum. Old spectrum data are lost.

PROTECT SPECTRUM

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

PURPOSE protect one (or all) spectrum

PARAMETERS

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

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

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

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

/LOG Switch default:
Output names of protected spectra.

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

Caller mdbm

Author H.G.Essel

Example

PROT SP A(3,9) protect one spectrum
PROT SP A(3) protect one spectrum
PROT SP A(3:9) protect seven spectra
PROT SP A(*) protect all members
PROT SP A* protect all members
PROT SP A protect all members

Remarks

File name E$PROSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$PROSP(CV_NAME,CV_SPEC_DIR,CV_BASE,CV_NODE,
I_LOG,I_KEEP_MAP)

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

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of Spectrum (wildcard) to be protected
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.
Arrays without index are protected totally.
For one dimensional arrays a range may be
specified like x(3:5). No wildcard is allowed in
this case. Single members of one and twodimensional
arrays may be protected by specifying the index:
X(7) or Y(4,5).

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Name of Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Name of Node

LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Output protected spectrum names.

KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Inhibit the unmap (detach) of the Data Base

Function

The spectrum will not be cleared by CLEAR command.
To clear the spectrum it must be UNPROTECTed first.

PROTOCOL

PROTOCOL line
/SESSION/COMMAND/GLOBAL

PURPOSE	Write line(s) into log file.
PARAMETERS	
/SESSION	Write text into session logfile SLOG_... (default)
/COMMAND	Write text into command logfile CLOG_...
/GLOBAL	Write text into global logfile GLOG_...
EXAMPLE	PROT "Starting new target xxx" write text line into session log file
Action rout.	U$TCLOG
Author	H.G.Essel

Remarks

File name	U$TCLOG.PPL
REMARKS	-

Description

Writes specified text into log file. If no text was specified, a prompt loop is entered.

READ CAMAC SPECTRUM

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

PURPOSE Read spectrum data from MR2000 into GOOSY spectrum

PARAMETERS

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

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

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

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

/LOG Switch default: ”
Output list of copied spectra

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

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

Caller mdbm

Author H.G.Essel

Example

```
READ CA SP A(1,2)
READ CA SP A(*)
READ CA SP A* /ADD
READ CA SP A (clear first spectrum only)
```

Remarks

File name E$ARCSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$ARCSP(CV_NAME,CV_SPEC_DIR,CV_BASE,CV_NODE,
                I_LOG,I_ADD,I_KEEP_MAP)
```

COMMAND

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

NAME

Routine arg. Input CHAR(*) VAR

Command par.

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

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par.

```
string global replace default: '$SPECTRUM'
  Name of Spectrum directory
```

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Name of Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Name of Node

LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Output list of copied spectra

ADD

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Instead or overwrite, add MR2000 channels to spectrum channels.

KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Inhibit the unmap (detach) of the Data Base

Function

The data part of MR2000 is copied into specified * GOOSY spectrum. The range in the MR2000 is defined in the spectrum. Channel contents may be overwritten (default) or added (/ADD).

REFRESH

REFRESH frame
/[NO]UPDATE
/[NO]WINDOW
/[NO]OVER

PURPOSE	Refresh picture as displayed
PARAMETERS	
frame	Number of frame or spectrum name. To refresh the whole picture specify frame=*.
/[NO] UPDATE	Get new spectrum copy
/[NO] WINDOW	Redraw windows too
/[NO] OVER	Redraw overlayed spectra
Caller	MDISP,MGOODISP
Action rout.	D$REFR
Author	W. Spreng

Example

1.) REFRESH *
Redraw the current picture to get rid of informations not stored in segments. Windows and overlays are lost.
2.) REFRESH
Redraw the whole picture with new spectrum contents.
3.) REFRESH 3/upd/over
Redraw frame 3 with new spectrum contents and redraw the overlayed spectra too. The other frames on the screen are not affected.

Remarks

File name D$REFR.PPL

Created by D$DSPCM.PPL

Description

CALLING STS=D$REFR(CV_frame,I_update,I_window,I_over,B_mask)

COMMAND REFRESH frame
/[NO]UPDATE
/[NO]WINDOW
/[NO]OVER

FRAME

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

Frames which should be refreshed. Valid inputs are:
n - a single frame number as shown on screen
n:m - a range of valid frames
* - for all frames on screen

UPDATE

Routine arg. Input BIN FIXEWD(15) valid values are 0 and 1

Command par. Switch.

If specified the spectra in the refreshed frames are displayed with the actual spectrum contents. Otherwise the spectra are redrawn as actually displayed on the screen.

WINDOW

Routine arg. Input BIN FIXEWD(15) valid values are 0 and 1

Command par. Switch.

The integration windows, the marked points and the fitted functions are redrawn in the refreshed frames.

OVER

Routine arg. Input BIN FIXEWD(15) valid values are 0 and 1

Command par. Switch.

The overlayed spectra are redrawn with their actual spectrum contents.

FUNCTION

Refresh a single frame or several frames of the current picture.

It is possible to perform a refresh getting the actual spectrum contents, then /UPDATE have to be set. If /WINDOW and /OVER is not specified the displayed windows and overlayed spectra are lost after the refresh.

If a scatterframe is refreshed the scatterpoints are deleted. Therefore this command can be used for an effective and fast refresh of the scatter frames. It is much faster than a new DISPLAY PICTURE or DISPLAY SCATTER command, because an initialization of the scatterframes is not necessary.

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);

REPLACE CONDITION WINDOW

REPLACE CONDITION WINDOW condition limits dimension cond_dir base node
/CHANN/CALIBR [=CALIBR]
/XAXIS/YAXIS [=AXIS]

PURPOSE Set or replace window condition by cursor

PARAMETERS

condition Name of window condition

limits Limits for condition window

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

cond_dir Default condition Directory

base Default Data Base name

node Default node name

CALIBR Specifies units of input

/CHAN	limits specified in channels
/CALIB	limits specified in calibrated units

AXIS Axis at which the condition limits should be set with cursor

/XAXIS	x-coordinates of input used
/YAXIS	y-coordinates of input used

Example

SET CONDITION WINDOW c1 100,400 5
Lower limit 100; upper limit 400 is set in dimension 5 of condition C1.
SET CONDITON WINDOW c1 100,400 *
Lower limit 100; upper limit 400 is set in all dimensions of condition C1.
SET CONDITON WINDOW c1 100,400 3:5
Lower limit 100; upper limit 400 is set in dimension 3 to 5 of condition C1.
SET CONDITON WINDOW c_array 100,400 3:5
Lower limit 100; upper limit 400 is set in dimension 3 to 5 of condition C_ARRAY(1).
SET CONDITON WINDOW c_array(2:4) 100,400 3:5 /CALIB
The specified limits are given in calibrated units they are transformed into spectrum units and then they are set in dimension 3 to 5 of condition C_ARRAY(2) to (4)
SET CONDITON WINDOW c_array(*) dim=*
The cursor appears to specify the limits set in all dimensions of all condition array members.

Description

CALLING STS=D$SCWIN(CV_cond,CV_limits,CV_dim,CV_dir,CV_base, cv_node,CV_cal,CV_Axis)

COMMAND REPLACE CONDITION WINDOW condition limits dimension cond_dir base node
/CHANN/CALIBR [=CALIBR]
/XAXIS/YAXIS [=AXIS]

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of condition window. Valid inputs are:
COND - for a single condition window
COND(3) - for a single member in a condition array
COND(1:4) - for a several members of a condition array
COND(*) - for all members of a condition array
Wildcards in the condition and directory name are not supported.

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String

Limits for condition window, only one pair of limits is supported. The specified limits are set for all dimensions.

DIMENSION

Routine arg. Input CHAR(*) VAR

Command par. String default=*

Dimension in which the limits should be set. Possible input values:
n - single number
n:m - range
* - set all dimensions

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$CONDITION

Default condition Directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node for Data Base

CALIBR

Routine arg. Input CHAR(*) VAR

Command par. Set default=/CHAN

Possible inputs are:

/CHAN	Limits are specified in original spectrum coordinates (channels).
/CALIB	Limits are specified in calibrated units.

Limits specified in calibrated units will be recalculated into original spectrum units, before they are set in the condition windows.

AXIS

Routine arg. Input CHAR(*) VAR

Command par. Set default=/XAXIS

Specifies the axis at which the graphical input should occur.

/XAXIS	Limits are taken from X-axis.
/YAXIS	Limits are taken from Y-axis.

Function

Set limits in the specified window condition. For multi-dimensional conditions it is necessary to specify the dimensions which should be set. If the window limits are specified the are set in all specified dimensions. If no window limits are specified they are set via graphical cursor input. Therefore a graphical input device has to be allocated.

REPLACE POLYGON

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

PURPOSE Set points in polygon

PARAMETERS

polygon Name of polygon data element

frame Frame in which the polygon should be edited.

xpoints X-Points of the polygon.

ypoints Y-Points of the polygon.

poly_dir Default condition Directory

base Default Data Base name

node Default node name

poly_pool Default Pool for polygons.

mode Switch from modification to deletion of existing points.

/MODIFY	Existing points are modified.
/DELETE	Existing points are ignored.

Caller MDISP,MGOODISP,D$DSPCM

Author W. Spreng

Remarks

File name D$SPOLY.PPL

Created by D$DSPCM.PPL

Description

CALLING STS=D$SPOLY(CV_polygon, L_frame, RA_xpoints, RA_ypoints, CV_poly_dir, CV_base, cv_node, CV_poly_pool, I_XY, CV_mode LA_array)

COMMAND REPLACE POLYGON polygon frame points poly_dir base node poly_pool
/IXYPOINTS
/DELETE/MODIFY [=MODE]

POLYGON

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name specification for the polygon, which should be set or modified. If the polygon Data Element does not exist, it will be created.

FRAME

Routine arg. Input BIN FIXED(31)

Command par. Integer, default=0

Number of frame in which the polygon should be set or modified. This frame is used to display the polygon points if an existing polygon should be modified (/MODIFY switch set).

XPOINTS

Routine arg. Input (*) BIN FLOAT(24)

Command par. Real array

Array of x-values for the polygon. If this points are specified the corresponding y-points have to be defined, too. If no points are given a graphical polygon editor is envoked to set or modify the polygon points.

YPOINTS

Routine arg. Input (*) BIN FLOAT(24)

Command par. Real array

Array of y-values for the polygon. If this points are specified the corresponding x-points have to be defined, too. If no points are given a graphical polygon editor is envoked to set or modify the polygon points.

POLY_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$POLYGON

Default directory for polygons.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node name for the Data Base.

POLY_POOL

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$PIC_POOL

Default Pool in which the polygon should be created.

XYPOINTS

Routine arg. Input BIN FIXED(15)

Command par. Switch

If this switch has been specified the values in XPOPINTS are interpreted as successive (x,y)-pairs. E.g.: XPOINTS=100,200,300,400 are the points (x,y)1=100,200 and (x,y)2=300,400.

MODE

Routine arg. Input CHAR(*) VAR

Command par. Switch default=/MODIFY

If the specified polygon has been defined earlier, the existing points can be modified or ignored

/MODIFY	The existing points are displayed and can be modified.
/DELETE	The existing points should be deleted and all polygon points have to be set again.

Function

Set or modify points in the specified polygon Data Element. If the polygon does not exist it will be created.

If no points are specified a graphical polygon editor is envoked, which allows to delete, insert or to append new points in the existing polygon. At the moment up to 64 points can be graphicly handled. If you need more please contact the GOOSY group.

The points could be specified directly in "xpoints" and "ypoints", then all existing points in the polygon are ignored. No changes of existing points is possible. If "points" is specified you have to define the whole polygon-points!

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();

SAVE DISPLAY

SAVE DISPLAY file directory

PURPOSE	Save displayed picture in a metafile.
PARAMETERS	
file	File name for the metafile.
directory	VMS-directory for the file
Caller	MDISP,MGOODISP,D$DSPCM
Author	W. Spreng

Example

1. SAVE DISPLAY spectrum
   Creates a metafile with the file specification SYS$LOGIN[.METAFILE]SPECTRUM.MET
   2. SAVE DISPLAY spectrum []
   Saves the picture in the metafile "spectrum.met" in the actually used directory.
   3. SAVE DISPLAY []spectrum
   Is an invalid input!!!

Remarks

File name	D$SAVE.PPL
Created by	GOO$DISP:D$DSPCM.PPL
REMARKS	This command is not supported in the fast display version.

Description

CALLING	STS=D$SAVE(CV_file,CV_directory,B_mask)
COMMAND	SAVE DISPLAY file directory

FILE

Routine arg. CHAR(*) VAR

Command par. String required

VMS-file name of the metafile to be created. The default file extension is ".MET".

DIRECTORY

Routine arg. CHAR(*) VAR

Command par. String replaceable default=[username.METAFILE]

VMS-directory name in which the metafile should be stored. If the directory does not exist, it will be created.

Function

The displayed picture is saved in a GKS-metafile. The directory for the metafile will be created if it does not exist.

A GKS output metafile is allocated and the picture is saved in that file, except scatterdata and spectrum updates all graphical data are stored in the created file.

A metafile is a device independent plotfile and it can be redisplayed by DISPLAY METAFILE command or on a plotter by the PLOT METAFILE command.

This command is not supported in the fast version of the GOOSY display process.

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 /NO-COMPRESS. 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 ANALYSIS

SET ANALYSIS
/[NO]ANALYSIS
/[NO]DYNAMIC
/[NO]SYNCHRON
/[NO]EVENT
/[NO]START
/[NO]STOP
/[NO]FOREIGN
/[NO]TABLES

PURPOSE Set analysis parameters.

PARAMETERS

/[NO] ANALYSIS Switch ON/OFF calling of analysis routine in the event loop

/[NO] DYNAMIC Switch ON/OFF calling of dynamic list executor in the event loop

/[NO] SYNCHRON Switch ON/OFF DECnet output synchronization

/[NO] EVENT Select event or buffer unpacking.

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

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

/[NO] FOREIGN Call foreign unpack routine X$UPFOR. The data buffer may have different structure than GOOSY buffers. Buffers are not checked in FOREIGN mode.

/[NO] TABLES Clear spectrum execution tables

REMARKS -

Description

FUNCTION This command allows to set analysis parameters:

1. Switch ON/OFF calling X$ANAL in event loop.
2. Switch ON/OFF calling dyn.list.exec.
3. Switch ON/OFF DECnet output synchronization.
4. Select buffer or event unpacking.
5. Enable/disable calling of start module
6. Enable/disable calling of stop module
7. Enable/disable calling of X$UPFOR.

File name I$ANACM.PPL

Action rout. I$ANACM_SET

Version 1.01

Author H.G.Essel

Last Update 12-APR-1985

SET CALIBRATION FIXED

SET CALIBRATION FIXED name unit start step input calib uncalib parameters polynom module image cal_dir base node
/[NO]FILE
/FIT/MODULE/PARAMETER/PROMPT/TABLE [=ACTION]

PURPOSE	Set table for a fixed-type calibration.
PARAMETERS	
name	Name of calibration
unit	Description of the calibration units.
start	Uncalibrated value for the first entry in the calibration table.
step	Stepwidth in uncalibrated units between two calibrated table entries.
input	Specifies a file which can be used to read the uncalibrated and calibrated values
parameters	Parameters for linear transformation
calib	Calibrated values
uncalib	Uncalibrated values.
module	User module which should be dynamicly linked out of a sharable image.
image	Sharable image which contains the specified user module.
cal_dir	Directory for calibration data elements.
base	Data Base name
node	Node name for Data Base file
/[NO] FILE	The required inputs should be read from an input file.
ACTION	Action which should be performed.

/FIT	The given calibrated and uncalibrated values are fitted with a polygon.
/MODULE	The calibration table must be filled by a user written procedure.
/PARAMETER	The parameters for a polygon are given and the table is created with that values.
/PROMPT	The given calibrated values are used and the uncalibrated values are specified by cursor inputs. Then a fit is performed.
/TABLE	All specified calibrated values are put directly into the calibration table

Caller MDISP,MGOODISP

Author W. Spreng

Example

1.) SET CALIBRATION FIXED name energy -30 2.0 PARA=(100.0,30.0,0.5,0.001) /PARAMETER

With the specified polynom parameter the calibration table is calculated. The first table entry corresponds to an uncalibrated value of "-30.0" the stepwidth between two subsequent entries is "2.0".

2.) SET CALIBRATION FIXED name energy -30 2.0 cal=(100.0,200,400) uncal=(0,200,300) poly=2/FIT

A polynom of the 2nd power is fitted to the listed calibrated and uncalibrated values. The resulting parameters are used to calculate the calibration table.

3.) SET CALIBRATION FIXED name energy -30 2.0 file /FILE/FIT

As above, but the list of calibrated and uncalibrated values are read from file.

4.) SET CALIBRATION FIXED name energy -30.0 2.0 calib=(0.0,20.0,30.0,40.0,50.0) /TABLE

The specified calibrated values are set directly in the table.

5.) SET CALIBRATION FIXED name energy -30.0 2.0 module=X$calib imange=USERSHR /MODULE

The module X$CALIB is called, it has to be linked into the sharable image USERSHR.

Remarks

Created by D$DSPCM.PPL

Description

CALLING

STS=D$SFICA(CV_NAME,CV_UNIT,R_START,R_STEP,CV_INPUT, RA_CALIB, RA_UNCAL, RA_PARAMETER, L_POLY, CV_MODULE, CV_IMAGE, CV_CAL_DIR, CV_BASE, CV_NODE, I_FILE, CV_ACTION)

COMMAND

SET CALIBRATION FIXED name unit start step input calib uncalib parameters polynom module image cal_dir base node
/[NO]FILE
/FIT/MODULE/PARAMETER/PROMPT/TABLE [=ACTION]

NAME

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of fixed calibration which should be set.

UNIT

Routine arg. Input CHAR(*) VAR

Command par. String required

Description for the calibration units. This title appears later on as the lettering at the calibrated spectrum axis.

START,STEP

Routine arg. Input BIN FLOAT(24)

Command par. Float required

For the calibrations of type FIXED the range of the calibration table is determined by the uncalibrated value corresponding to the first table entry, the stepwidth between two table entries and the total length of the calibration table.

"START" specifies the uncalibrated value for the first entry in the calibration table. "STEP" determines the Stepwidth of uncalibrated values between two subsequent table entries.

INPUT

Routine arg. Input CHAR(*) VAR

Command par. String

Specifies a file from which the calibrated and/or the uncalibrated values are read. Input from the specified file is possible in all cases, for which the calibrated and/or uncalibrated values for a fit are used. For details see the function description.

If a file name is specified the /FILE switch is required.

CALIB

Routine arg. Input (*) BIN FLOAT(24)

Command par. Float array

Input of calibrated values (e.g the energies of a calibration source) which should be used to determine the parameter of a polynom performing a fit with the specified calibrated and uncalibrated values. Therefore for each calibrated value the corresponding uncalibrated value is required.

If calibrated values are given the list of uncalibrated values can be specified directly in "UNCALIB" parameter (this mode is activated by the /FIT switch) or they can be specified via graphical inputs if /PROMPT has been specified!

Futhermore the specified calibrated values are put directly into the calibration table if the /TABLE switch is set.

See the /ACTION parameter for a detail description.

UNCALIB

Routine arg. Input (*) BIN FLOAT(24)

Command par. Float array

Input of uncalibrated values which should be used to determine the parameter of the linear polynom performing a fit with the specified uncalibrated and calibrated values. Therefore for each uncalibrated value the corresponding calibrated value is required.

This list of values is used if the /FIT switch has been specified. See the /ACTION parameter for a detail description.

PARAMETER

Routine arg. Input (*) BIN FLOAT(24)

Command par. Float array

Array containing the parameter for a polynom, used to determine the calibration table. The parameters should be specified like:

R_0,R_1,...,R_n n is arbitrary

The calibration table is calculated with these parameters:

cal = R_0 + R_1*uncal + ... + R_n*(uncal**n)

If the polynom parameter should be used specify the /PARAMETER switch. See the /ACTION parameter for a detail description.

POLYNOM

Routine arg. Input BIN FIXED(31)

Command par. Integer default=0

Largest exponent of the polynom which should be fitted to the calibrated values.

MODULE

Routine arg. Input CHAR(*) VAR

Command par. String

User module which should be linked dynamicly from a user defined sharable image. The user routine must set the calibration table. If a user module should be called the /MODULE switch has to be set and the logical name of the sharable image has to be specified. The user-module is called with the following parameter:

L_status=user(R_start,R_step,L_entries,RA_table)

R_start - Start value of table.
BIN FLOAT(24) INPUT

R_step - Stepwidth
BIN FLOAT(24) INPUT

L_entries - Number of table entries
BIN FIXED(31) INPUT

RA_table- Array which contains the final calibrated values.

(*) BIN FLOAT(24) OUTPUT

A module "user" can be linked into the sharable image "SHARE.EXE" by:

```
LSHARE user.obj share.exe /share=usershr
```

"USERSHR" is the logical image name, which can be inserted into the "IMAGE" parameter of this command.

IMAGE

Routine arg. Input CHAR(*) VAR

Command par. String

Sharable image in which the user module can be found.

CAL_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$CALIB

Default directory for calibration Data Elements.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

FILE

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch negatable default=/NOFILE

If /FILE has been specified all input necessary to perform the fit are read from the specified file in the parameter "INPUT".

The list of calibrated and uncalibrated values should be read from file if this switch together with /FIT has been specified.

The list of calibrated values is read from file and the corresponding values are prompted via cursor inputs if this switch has been specified together with /PROMPT.

The values of all table entries are read from file if the /TABLE switch is set.

See the /ACTION parameter for a detail description.

ACTION

Routine arg. Input CHAR(*) VAR

Command par. Set default=/PARAMETER

Specifies which input should be used and which should be performed . Possible inputs are:

/FIT	The given calibrated and uncalibrated values are fitted with a arbitrary polygon, which is then used to fill the calibration table.
/MODULE	The calibration table must be filled by a user written procedure.
/PROMPT	The given calibrated values are used and the uncalibrated values are specified via cursor inputs. Then a polynom fit is performed, to get polynom parameter used to fill the calibration table.
/PARAMETER	The parameters for a polygon are given and the table is created with that values.
/TABLE	All specified calibrated values are put directly into the calibration table.

Function

The calibration table for a FIXED-type calibration is set. In a fixed-calibration table the stepwidth between two uncalibrated values is fixed. The corresponding calibrated values are keept in a table for each step. Therefore the range of the calibration table is determined by the uncalibrated value for the first table entry "start", the stepwidth between two entries "step" and total range of the calibration table.

To set the values in the calibration table different modes are implemented. In the most cases it is sufficient to fit a sequence of calibrated values (e.g. energies from a calibration source) and uncalibrated values (normaly the spectrum channels of the corresponding energies) by a polynom, which is used to calculate the calibration table. But sometimes the calibration function should be anything, in that case the calibration table can be set in a user module or by directly specifying all table entries.

Modes The following modes are implemented:

1. /FIT The calibrated and uncalibrated values are specified and a least square fit should be performed with the data using an arbitrary polynom with the largest exponent given by the parameter "polynom". This polynom is used to calculate the calibration table. In that mode the input from a file is possible.

2. /PROMPT The calibrated values and the maximum polynom power are specified. The corresponding uncalibrated values are prompted by cursor inputs. A fit is performed to get the polynom parameters which are used to create the calibration table. In that mode the calibrated values can be read from file.

3. /TABLE If the calibration table has been calculated earlier it is possible to put the calibrated values directly into the table by the /TABLE switch. In that case the number of specified values and the number of table entries has to be identical. In that mode the calibrated values can be read from file.

4. /MODULE The calibration table could be set with a user written procedure which must be linked into a sharable image. The names of the user module and the sharable image have to be specified in the "MODULE" and "IMAGE" parameter. The user-module is called with the following parameter:

```
L_status=user(R_start,R_step,L_entries,RA_table)
        R_start - Start value of table.
                  BIN FLOAT(24) INPUT
        R_step - Stepwidth
                  BIN FLOAT(24) INPUT
        L_entries - Number of table entries
                  BIN FIXED(31) INPUT
        RA_table- Array which contains the final
                  calibrated values.
                (*) BIN FLOAT(24) OUTPUT
```

If the return status is not success the calibration will not be modified.

To link a module into a shareable image a DCL command is provided:

```
LSHAR user image.exe/share=usershr
```

which linke the module "USER" into the executeable shareable image "IMAGE.EXE" and the image is assigned to the logical name "USERSHR", which can be used for the required "IMAGE" parameter.

5./PARAMETER The parameters of an arbitrary polynom, which should be used to calculate the calibration table, can be specified in "PARAMETER":

R_0,R_1,...,R_n n-arbitrary

The calibration table is calculated with these parameters:

cal = R_0 + R_1*uncal + ... + R_n*(uncal**n)

File input Futhermore there are two input possibilities for the required calibrated and/or uncalibrated values. The first one is to specify them directly in the corresponding parameter. The second one is to read the values from file by specifying the "INPUT" and the /FILE switch.

The input format of this file is very simple:

A "!" in the first column is interpreted as a comment line. The calibrated an uncalibrated values should be order in columns separated by blank or by comma. In each record one or two values are read, depending if only calibrated or additionally uncalibrated values should be read. If two values are required the first value is uncalibrated the second is calibrated.

An example for file input if "/FIT" is specified:

```
! Comment line
    100.0 200.0
    150.0 301.0
! another comment line
    300.0,605.0
    400.0, 790.0
! end of data
```

SET CALIBRATION FLOAT

SET CALIBRATION FLOAT name unit input module image
cal_dir base node
/[NO]FILE

PURPOSE	Set table for a float-type calibrations.
PARAMETERS	
name	Name of calibration
unit	Description of the calibration units. This string is shown at the calibration axis.
input	Specifies a file which can be used to read the uncalibrated and calibrated values
module	User module which should be dynamicly linked out of a sharable image.
image	Sharable image which contains the specifeid user module.
cal_dir	Directory for calibration data elements.
base	Data Base name
node	Node name for Data Dase file
/[NO] FILE	The required inputs should be read from an input file. If this switch is not specified the inputs should be specified in the command.
Caller	MDISP,MGOODISP
Author	W. Spreng

Example

1.) SET CALIBRATION FLOAT name energy file/FILE
The complete table (a list of calibrated and uncalibrated values is read from file.
2.) SET CALIBRATION FLOAT name energy module=X$CALIB image=USERSHR
The module X$CALIB is called, it has to be linked into the sharable image USERSHR.

Remarks

Created by D$DSPCM.PPL

File name GOO$DISP:D$SFLCA.PPL

Description

CALLING STS=D$SFLCA(CV_NAME,CV_UNIT,CV_INPUT,CV_MODULE,
CV_IMAGE,CV_CAL_DIR,CV_BASE,CV_NODE,
I_FILE)

COMMAND SET CALIBRATION FLOAT name unit input module image
cal_dir base node
/[NO]FILE

NAME

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of float calibration which should be set.

UNIT

Routine arg. Input CHAR(*) VAR

Command par. String required

Description for the calibration units. This title appears later on as the lettering at the calibrated spectrum axis.

INPUT

Routine arg. Input CHAR(*) VAR

Command par. String

Specifies a file from which the calibrated and/or the uncalibrated values for the calibration table are read. If a file name is specified the /FILE switch is required.

MODULE

Routine arg. Input CHAR(*) VAR

Command par. String

User module which should be linked dynamicly from a user defined sharable image. The user routine must set the calibration table. The user-module is called with the following parameters:

L_status=user(L_entries,RA_uncal,RA_cal)
L_entries - Number of table entries
BIN FIXED(31) INPUT
RA_uncal - Array which contains a list of uncalibrated values.
(*) BIN FLOAT(24) OUTPUT
RA_cal - Array which contains the final calibrated values.
(*) BIN FLOAT(24) OUTPUT

A module "user" can be linked into the sharable image "SHARE.EXE" by:

LSHARE user.obj share.exe /share=usershr

"USERSHR" is the logical image name, which can be inserted into the "IMAGE" parameter of this command.

IMAGE

Routine arg. Input CHAR(*) VAR

Command par. String

Sharable image in which the user module can be found.

CAL_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$CALIB

Default Directory for calibration Data Elements.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

FILE

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch negatable default=/NOFILE

If this switch is set the values of all table entries are read from file.

Function

The calibration table for a FLOAT-type calibration

Modes The following modes are implemented:

1. /FILE The whole table is read from the specified file. The input format of this file is very simple:

A "!" in the first column is interprted as a comment line. The calibrated an uncalibrated values should be order in columns separated by blank or by comma. In each record two values are read. The first value is uncalibrated the second is calibrated.

2. Module A user written subroutine is called which fills the calibration table. The routine has to be linked into a sharable image. The procedure is called with three the following parameters:

L_status=user(L_entries,RA_uncal,RA_cal)

L_entries - Number of table entries
BIN FIXED(31) INPUT

RA_uncal - Array which contains a list of uncalibrated values.
(*) BIN FLOAT(24) OUTPUT

RA_cal - Array which contains the final calibrated values.
(*) BIN FLOAT(24) OUTPUT

If the return status is not success the calibration will not be modified.

To link a module into a shareable image a DCL command is provided:

```
LSHAR user image.exe/share=usershr
```

which linke the module "USER" into the executeable shareable image "IMAGE.EXE" and the image is assigned to the logical name "USER-SHR", which can be used for the required "IMAGE" parameter.

SET CALIBRATION LINEAR

SET CALIBRATION LINEAR name unit input parameters
calib uncalib cal_dir base node
/FILE
/FIT/PROMPT/PARAMETER [=/ACTION]

PURPOSE Set parameters for a linear calibration.

PARAMETERS

name Name of calibration

unit Description of calibration units.

input Specifies a file which can be used to read the uncalibrated and calibrated values

parameters Parameters for linear transformation

calib Calibrated values

uncalib Uncalibrated values.

cal_dir Directory for calibration Data Elements.

base Data Base name

node Node name for Data Base file

/[NO] FILE The required inputs should be read from an input file.

ACTION Action which should be performed. Possible inputs are:

/FIT	Perform a fit with given calibrated and uncalibrated values.
/PROMPT	The uncalibrated values are picked with cursor inputs.
/PARAMETER	The specified parameters are used.

Caller	MDISP,MGOODISP
Author	W. Spreng
Dataset	D$SLICA.PPL

Example

1.) SET CALIBRATION LINEAR name energy (100.0,0.5) /PARAMETER
The parameter (100.0 offset, factor 0.5) are put into the calibration Data Element. If a spectrum is displayed with these calibration the description at the axis is "energy".
2.) SET CALIBRATION LINEAR name energy calib=(100.0,200,400) uncalib=(0,200,300) /FIT
A fit is performed with the list of calibrated and uncalibrated values. The resulting polynom parameter are stored in the calibration.
3.) SET CALIBRATION LINEAR name energy file /FILE /FIT
The list of calibrated and uncalibrated values are read from file.
4.) SET CALIBRATION LINEAR name energy calib=(100.0,200,400) /PROMPT
The uncalibrated values are prompted via cursor after that the fit is performed.
5.) SET CALIBRATION LINEAR name energy file/FILE /PROMPT
The calibrated values are read from file.

Remarks

Created by	D$DSPCM.PPL
File name	GOO$DISP:D$SLICA.PPL

Description

CALLING	STS=D$SLICA(CV_NAME,CV_UNIT,CV_INPUT, RA_PARAMETER, RA_CALIB, RA_UNCALIB, CV_CAL_DIR, CV_BASE, CV_NODE, I_FILE, CV_ACTION)
COMMAND	SET CALIBRATION LINEAR name unit input parameters calib uncalib cal_dir base node /[NO]FILE /FIT/PROMPT/PARAMETER [=ACTION]

NAME

Routine arg.	Input CHAR(*) VAR

Command par. String required

Name of the calibration which should be set.

UNIT

Routine arg. Input CHAR(*) VAR

Command par. String required

Description for the calibration units. This title appears later on as the lettering at the calibrated spectrum axis.

INPUT

Routine arg. Input CHAR(*) VAR

Command par. String

Specifies a file from which the calibrated and/or the uncalibrated values are read. Input from the specified file is possible in all cases, for which the calibrated and/or uncalibrated values for a fit are used. For details see the function description.

If a file name is specified the /FILE switch is required.

PARAMETERS

Routine arg. Input (*) BIN FLOAT(24)

Command par. Float array default = (0.0,1.0)

Array containing the parameter of the linear polygon used for the calibration. If these parameters should be used specify the /PARAMETER switch. See the /ACTION parameter for a detail description.

CALIB

Routine arg. Input (*) BIN FLOAT(24)

Command par. Float array

Input of calibrated values which should be used to determine the parameter of the linear polynom by performing a fit with the specified calibrated and uncalibrated values. Therefore for each calibrated value the corresponding uncalibrated value is required.

If calibrated values are given the list of uncalibrated values can be specified directly in "UNCALIB" parameter (this mode is activated by the /FIT switch) or they can be specified via graphical inputs if /PROMPT has been specified! See the /ACTION parameter for a detail description.

UNCALIB

Routine arg. Input (*) BIN FLOAT(24)

Command par. Float array

Input of uncalibrated values which should be used to determine the parameter of the linear polynom performing a fit with the specified uncalibrated and calibrated values. Therefore for each uncalibrated value the corresponding calibrated value is required.

This list of values is used if the /FIT switch has been specified. See the /ACTION parameter for a detail description.

CAL_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$CALIB

Default Directory for calibration Data Elements.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

FILE

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch negatable default=/NOFILE

If /FILE has been specified all input necessary to perform the fit are read from the specified file in the parameter "INPUT".

The list of calibrated and uncalibrated values should be read from file if this switch together with /FIT has been specified.

The list of calibrated values is read from file and the corresponding values are prompted via cursor inputs if this switch has been specified together with /PROMPT. See the /ACTION parameter for a detail description.

ACTION

Routine arg. Input CHAR(*) VAR

Command par. Set default=/PARAMETER

Specifies which input should be used and which should be performed . Possible inputs are:

/FIT	The given calibrated and uncalibrated values are fitted with a linear polygon.
/PROMPT	The given calibrated values are used and the uncalibrated values are specified via cursor inputs. Then a fit is performed.
/PARAMETER	The specified parameters are used.

Function

The two parameter of a linear polygon are stored in a calibration Data Element. The polygon parameter can be determined in different modes:

Modes The following modes are implemented:

1./PARAMETER The parameters are specified and should be copied into the Data Element. In that case /PARAMETER must be set.

2. /FIT The calibrated and uncalibrated values are specified and a least square fit should be performed through the data. This mode is activated with /FIT.

3. /PROMPT Only calibrated values are specified and the corresponding values should be specified via cursor input. If you like this specify /PROMPT.

File input There are two possibilities to specify the required calibrated and uncalibrated values. They can be specified directly in the command or they are read from the specified inputfile if the /FILE switch is set! The input from file is convenient if e.g. the energies of the same calibration source should be used to calibrate different spectra. In that case the corresponding peaks can be picked by cursor in each spectrum for each energy value found in the specified file.

The input format of this file is very simple:

A "!" in the first column is interprted as a comment line. The calibrated an uncalibrated values should be order in columns separated by blank or by comma. In each record one or two values are read, depending if only calibrated or additionally uncalibrated values should be read. If two values are required the first value is uncalibrated the second is calibrated.

An example for file input if "/FIT" is specified:

```
! Comment line
! uncalibr.—calibr.
     100.0 200.0
     150.0 301.0
! another comment line
     300.0,605.0
     400.0, 790.0
! end of data
```

SET CONDITION PATTERN

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

PURPOSE set stored pattern of a pattern condition

PARAMETERS

name String default :
Name of condition

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

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

base String replace default : 'DB'
Default Data Base

node String replcae default : '*'
Default node

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

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$SECOP.PPL

Dataset -

NAME

TYPE String default: none
Name of condition which is wanted to be modified

PATTERN

TYPE String default: 0
Pattern specification, may be of the form '010101'B or 010101

INVPAT

TYPE String default: 0
Inversion pattern specification, may be of the same form

INDEX

TYPE Integer default: 1
Internal index

COND_DIR

TYPE String replace default: none
Default condition directory

BASE

TYPE String replace default: none
Default Data Base

NODE

TYPE String replace default: none
Default node

KEEP_MAP

TYPE Switch default: /KEEP_MAP
Valid values are:

[NO] KEEP_MAP Inhibit unmap of Data Base atfer command execution

Function

to modify the pattern of a pattern condition

Example

—

Remarks

REMARKS Module is an action routine

Description

Argument description

NAME

Type Input CHAR(*) VAR
Name of condition which is wanted to be modified

SPEC

Type Input CHAR(*) VAR
Pattern specification, may be of the form '010101'B or 010101

INVPATT

Type Input CHAR(*) VAR
Inversion pattern specification, may be of the same form

IND

Type Input CHAR(*) VAR
Internal index

DIR

Type Input CHAR(*) VAR
Default condition directory

BASE

Type Input CHAR(*) VAR
Default Data Base

NODE

Type Input CHAR(*) VAR
Default node

KEEP_MAP

Type Input BIN FIXED(15)
Valid values are:

0 Unmap the Data Base

1 Inhibit unmap of the Data Base
Inhibit unmap of Data Base atfer command execution

Function

to modify the pattern of a pattern condition

Remarks

Module is an action routine

Example

SET CONDITION WINDOW

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

PURPOSE	Set window condition
PARAMETERS	
condition	Name of window condition
limits	Limits for condition window
dimension	Dimensions for multi-dimensional condition windows which should be set.
cond_dir	Default condition Directory
base	Default Data Base name
node	Default node name
Caller	MDBM,MGOODBM
Author	W. Spreng

Example

1.) SET CONDITION WINDOW c1 100,400 5
Lower limit 100; upper limit 400 is set in dimension 5 of condition C1.
2.) SET CONDITON WINDOW c1 100,400 *
Lower limit 100; upper limit 400 is set in all dimensions of condition C1.
3.) SET CONDITON WINDOW c1 100,400 3:5
Lower limit 100; upper limit 400 is set in dimension 3 to 5 of condition C1.
4.) SET CONDITON WINDOW c_array 100,400 3:5
Lower limit 100; upper limit 400 is set in dimension 3 to 5 of condition C_ARRAY(1).
5.) SET CONDITON WINDOW c_array(2:4) 100,400 3:5

Lower limit 100; upper limit 400 is set in dimension 3 to 5 of condition C_ARRAY(2) to (4)
6.) SET CONDITON WINDOW c_array(*) 100,400 *
Lower limit 100; upper limit 400 is set in all dimensions of all condition array members.

Remarks

File name E$SCWIN.PPL

created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$SCWIN(CV_condition,CV_limits,CV_dimension,
                CV_cond_dir,CV_base,cv_node,I_KEEP_MAP)
```

COMMAND

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

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of condition window. Valid inputs are:
- COND - for a single condition window
- COND(3) - for a single member in a condition array
- COND(1:4) - for a several members of a condition array
- COND(*) - for all members of a condition array

Wildcards in the condition and directory name are not supported.

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String required

Limits for condition window, only one pair of limits is supported. The specified limits are set for all dimensions.

DIMENSION

Routine arg. Input CHAR(*) VAR

Command par. String default=*

Dimension in which the limits should be set. Possible input values:
n - single number
n:m - range
* - set all dimensions

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$CONDITION

Default condition Directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node for Data Base

KEEP_MAP

Routine arg. Input BIN FIXED(31) valid values 0 and 1

Command par. Switch negatable default=/KEEP_MAP

If set the mapping context will be kept. If /NOKEEP_MAP specified the Data Base will be dettached after the command completion.

Function

Set limits in the specified window condition. For multi-dimensional conditions it is necessary to specify the dimensions which should be set. If the window limits are specified the are set in all specified dimensions.

SET DEVICE COLOR

SET DEVICE COLOR name index r g b
/UPDATE

PURPOSE	Allocate a graphical device
PARAMETERS	
name	Logical device name use TT for current terminal.
index	Color index (0-max)
R,G,B	Color values for read, green, blue (0-100)
/UPDATE	update color setup
Caller	MDISP,MGOODISP,D$DSPCM
Author	W. Spreng

Example

Remarks

File name	D$SECOL.PPL
Created by	GOO$DISP:D$DSPCM.PPL

Description

CALLING	STS=D$SECOL(CV_dev,l_index,R_r,R_g,R_b,I_update)
COMMAND	SET DEVICE COLOR name index r g b /UPDATE

NAME

Routine par. Input CHAR(*) VAR

Command arg. String required

Logical or physical device name. For a graphical Terminal it is the VMS-Terminal address; e.g.:

TXA6	for a terminal connected directly to one VAX.
LTA999	for a terminal connected to a LAT server.
TT	for the terminal of the session.

For spooled devices (all supported plotters) you can specify a file name with or without a file type.

For Metafiles a file name or a logical name which refers to a file is required.

For MICRO-VAX II and GPX devices the device name is arbitrary, it is displayed at the generated graphic window.

INDEX

Routine par. Input BIN FIXED(31)

Command arg. Integer, Default=1

Specifies the color index to be changed. Valid values are 0-max, where 0 is the background color.

1	picture lettering
2	spectrum name
3	axes and lettering
4	lettering info
6	spectrum

R

Routine par. Input BIN FLOAT(24)

Command arg. Float

Percentage of red color admixture. Valid values aree 0-100.

G

Routine par. Input BIN FLOAT(24)

Command arg. Float

Percentage of green color admixture. Valid values 0-100.

B

Routine par. Input BIN FLOAT(24)

Command arg. Float

Percentage of blue color admixture. Valid values 0-100.

UPDATE

Routine par. Input BIN Fixed(15)

Command arg. Switch

There are some devices which are not possible to update theire color map dynamicaly. In that case a change of the color map can be make visible by redrawing the whole picture. To improve the performance of the color map changes in that case the update is suppressed until it will be specified.

Function

If you do not agree with the default color map of GOOSY, you have the possibility to define you own color lay out by specifying the color index and the RGB values of that color.

SET DISPLAY MODE

SET DISPLAY MODE
/STANDARD/FAST [=VERSION]

PURPOSE Select display version.

PARAMETERS

VERSION Version which should be used.

- **/STANDARD** Standard display version with full functionality.
- **/FAST** Reduced version which needs less CPU.

Caller MDISP, MGOODISP

Author W. Spreng

Remarks

Created by GOO$DISP:D$DSPCM.PPL

File name GOO$DISP:D$DVERS.PPL

Description

CALLING STS=D$DVERS(CV_VERSION)

COMMAND SET DISPLAY MODE
/STANDARD/FAST [=VERSION]

VERSION

Routine arg. CHAR(*) VAR

Command par. Set default=/STANDARD

Change actual display version. Possible inputs are:

/STANDARD	Standard display version is used. The whole display functionality is supported.
/REDUCED	The reduced display version is activated. The internal graphic storage is not used, this reduces the CPU usage to 60% of the standard version. But not all commands are supported and several are only available with reduced functionality.

FUNCTION

Up to now you can select between two display versions. The STANDARD implementation, which supports all commands and the FAST version which works without any intermediate graphic storage.

The fast version saves CPU-time, it needs about 60% of the standard version CPU-time. It is recommended to use this reduced version when GOOSY should run on small VAX's like the 750 or a MICRO-VAX II and a CPU intensive analysis is active. Futhermore the fast display version do not need any disk storage for the intermediate graphic storage.

This advantages have to be payed with a reduced functionality:

1.) The PLOT PICTURE command is not supported.

2.) The SAVE DISPLAY command is not supported.

3.) The REFRESH command works only like REFRESH * /UPD.

4.) When you allocate one device which could not delete a specified section on the screen, the display surface of all devices will be cleared during the EXPAND command! E.g if an plotter or a metafile is allocated this will happen.

After the FAST version is selected the standard version could be activeted at any time, e.g. to plot a picture. But then first a new DISPLAY command must be given first!

SET DYNAMIC LIST

SET DYNAMIC LIST dyn_list dyn_type key value dyn_dir base node

PURPOSE Modify attached dynamic list

PARAMETERS

dyn_list Dynamic list name specification
If *, all attached lists are modified.
common required default

dyn_type Type of dynamic sublist or *
default:'*'

key Keyword for parameter to be changed.
LIST value=ON/OFF
SUBLIST value=ON/OFF

value Value for parameter

dyn_dir Default directory
common default:'$DYNAMIC'

base Default data base name
common default:'DB'

node Default node name
common default:'E'

EXAMPLE -

Caller M$DLCMD

Author H.G. Essel

File name M$ASTLL.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ASTLL(CV_DYN_LIST,CV_TYPE,CV_PARAM,
CV_VALUE,CV_DYN_DIR,CV_BASE,CV_NODE)

ARGUMENTS

CV_DYN_LIST Dynamic list name specification
CHAR(*) VAR
Input

CV_TYPE Type of sublist or *
CHAR(*) VAR
Input

CV_PARAM Parameter name
CHAR(*) VAR
Input
List of parameters and values:
LIST ON/OFF
SUBLIST ON/OFF

CV_VALUE Value for parameter
CHAR(*) VAR
Input

CV_DYN_DIR Default Directory
CHAR(*) VAR
Input

CV_BASE Default Data Base name
CHAR(*) VAR
Input

CV_NODE Default node name
CHAR(*) VAR
Input

FUNCTION Set parameters of dynamic list

REMARKS Module is an action routine.

EXAMPLE -

SET EVENT INPUT

SET EVENT INPUT name type directory base

PURPOSE Set input event data element.

PARAMETERS

name string replace default=DB:[DATA]EVENT
DE name specification base:[dir]name

type string replace
Type of DE. If specified, this type is verified.
If the data element in the base has a different
type an error message is returned.

directory string replace default=DATA
Default directory, if not specified in name.

base string replace default=DB
Default base, if not specified in name.

FUNCTION All initialization entries of unpack routines (the loaded ones too) are called with the pointer to the data element and the length as arguments.

EXAMPLE SET EV IN DB:[DATA]MYEVENT

Description

FUNCTION This procedure locates the specified data element and calls $XEVENT, $XUPJ11, $XUPEVT and $XUPCMP.

File name I$ANACM.PPL

Action rout. I$ANACM_SET_EVI

Version 1.01

Author H.G.Essel

Last Update 12-APR-1985

SET EVENT OUTPUT

SET EVENT OUTPUT name type directory base

PURPOSE Set output event data element.

PARAMETERS

name string replace default=DB:[DATA]NEWEVENT
DE name specification base:[dir]name

type string replace
Type of DE. If specified, this type is verified.
If the data element in the base has a different
type an error message is returned.

directory string replace default=DATA
Default directory, if not specified in name.

base string replace default=DB
Default base, if not specified in name.

FUNCTION All initialization entries of pack routines (the loaded ones too) are called with the pointer to the data element and the length as arguments.

EXAMPLE SET EV OUT DB:[NEW]EVENT

Description

FUNCTION This procedure locates the specified data element and calls $XPACMP and $XPAEVT. Then the routines X$PACMP or X$PAEVT copy this data element into the output buffers. The routione is selected by START ANAL OUTPUT /COMPRESS or /COPY

File name I$ANACM.PPL

Action rout. I$ANACM_SET_EVO

Version 1.01

Author H.G.Essel

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 LETTERING

SET LETTERING specname dim text spec_dir base node /[NO]KEEP_MAP

PURPOSE set lettering at display axisses

PARAMETERS

specname name of spectrum
Wildcards in the directory, data element name and dataelement index are supported.
required

dim dimension (0 ... 8)
required

text text to be written at the axis

spec_dir default Directory
repalce default:'$SPECTRUM'

base default Data Base
repalce default:'DB'

node default node
repalce default:'E'

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

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$SELET.ppl

Dataset -

Examples

```
SET LETTERING spectrum 0 "This is the x-axis"
SET LETTERING spectrum 1 "This is the y-axis"
```

Description

CALLING STS=E$SELET(CV_SPECNAME,CV_DIM,CV_TEXT
CV_SPEC_DIR,CV_BASE,CV_NODE,I_KEEP_MAP,B_MASK)

COMMAND SET LETTERING specname dim text spec_dir base node
/[NO]KEEP_MAP
Argument and parameter description.

NAME

Routine par. Input CHAR(*) VAR

Command arg. String requested replacable

Name of spectrum. Wildcards in the directory, data element name and dataelement index are supported.

DIM

Routine par. Input CHAR(*) VAR

Command arg. String requested

Dimension to be modified, can be between 0 and 8. Where 0 is the x-axis, 1 the y-axis, 2 the z-axis etc.

TEXT

Routine par. Input CHAR(*) VAR

Command arg. String

Text to be written at the axis. If there are blanks in the text, it has to be enclose in quotes (").

SPEC_DIR

Routine par. Input CHAR(*) VAR

Command arg. String global replacable default=$SPECTRUM
Default spectrum directory name.

BASE

Routine par. Input CHAR(*) VAR

Command arg. String global replacable default=DB
Default data base name.

NODE

Routine par. Input CHAR(*) VAR

Command arg. String global replacable default=*
Default node name.

/KEEP_MAP

Routine par. Input BIN FIXED(15)

Command arg. Switch
Inhibit unmap of Data Base

Function

FUNCTION The text at the axisses are written into the spectrum data element.

SET LOCK OUTPUT

```
SET LOCK OUTPUT
            /[NO]PRCID
            /[NO]LKID
            /[NO]PARID
            /[NO]UIC
            /[NO]PRCNAM
            /[NO]STATE
```

PURPOSE Set lock output spezification (called in MUTIL)

PARAMETERS

/PRCID negatable switch
Output process-id

/LKID negatable switch
Output lock-id

/PARID negatable switch
Output parent lock-id

/UIC negatable switch
Output UIC

/PRCNAM negatable switch
Output process name

/STATE negatable switch
Output lock state

Caller MLOCKS

Author Gertrud Schneider

Example

SET LOCK OUTPUT /UIC /LKID /NOPRCID
In the output UIC and lock-id is now displayed,
process-id is no longer displayed.

Remarks

File name U$LOCKCM.PPL

Created by U$LOCKCM.PPL

Description

CALLING STS=U$SLKOUT(I_PRCID,I_LKID,I_PARID,I_UIC,
I_PRCNAM,I_STATE,B_mask)

COMMAND SET LOCK OUTPUT
/[NO]PRCID
/[NO]LKID
/[NO]PARID
/[NO]UIC
/[NO]PRCNAM
/[NO]STATE
Arguments/Parameters description

/PRCID

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch

/PRCID Outputprocess-id

/LKID

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch

/LKID Output lock-id

/UIC

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch
/UIC	Output UIC

/PARID

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch
/UIC	Output parent ID

/PRCNAM

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch
/PRCNAM	Output process name

/STATE

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch
/STATE	Output lock state

Function

The routine controlls the output of the command SHOW LOCKS . With every switch a bit will be set for the wanted information in the output. A bit, once set, is conserved and can only deleted with the negated switch (/NO....). The presetting, when used first time, is

/NOPRCID /LKID /PARID /NOUIC /PRCNAM /STATE

SET MEMBER

SET MEMBER mem_spec value dir base node
/[NO]KEEP_MAP
/[NO]LOG

PURPOSE Change value of an Data Element member or a full Data Element member array (wildcarded).

PARAMETERS

mem_spec Node::base:[dir]name(i)->type(i)
required common default

value Enter new value for DE member
required

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/[NO] LOG [Do not]write result to terminal.
default: '/LOG'

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

Caller M$DMCMD

Author M. Richter

File name M$ASTME.PPL

Dataset -

EXAMPLE SET MEMB DB:[DATA]ADAM.ALPHA.BETA 4711
set the value '4711' in the member 'BETA' of
structure 'ALPHA' of Data Element 'ADAM' in Directory 'DATA' of
Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ASTME(CV_ELEMENT,CV_VALUE,
CV_DIR,CV_BASE,CV_NODE,I_LOG,I_KEEP_MAP)

ARGUMENTS

CV_ELEMENT I Node::base:[dir]name(i)->type(i)
CHAR(*) VAR

CV_VALUE I New value for DE member
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_LOG Write output.
BIN FIXED (15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Change value of an Data Element member

REMARKS Module is an action routine.

EXAMPLE -

SET MWPC

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

PURPOSE SET MWPC modes.

PARAMETERS

acknow Integer Number of acknowledges to sent back for each buffer.

EXAMPLE

Description

FUNCTION -

File name I$ACVME.PPL

Action rout. I$ACV_SET_MWPC

Version 1.01

Author H.G.Essel

Last Update 08 MAR-1993

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 SCATTER BUFFER

SET SCATTER BUFFER value dyn_list dyn_dir base node

PURPOSE Set scatter buffer size

PARAMETERS

value Number of display points to be collected, before the buffer is sent to display
default:'1000'

dyn_list Dynamic list name specification
common required default
If *, all attached lists are modified.

dyn_dir Default directory
common default:'$DYNAMIC'

base Default data base name
common default:'DB'

node Default node name
common default:'E'

EXAMPLE -

Caller M$DLCMD

Author H.G.Essel

File name M$ASTSB.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ASTSB(CV_VALUE,CV_DYN_LIST
,CV_DYN_DIR,CV_BASE,CV_NODE)

ARGUMENTS

CV_VALUE Number of scatter points to be collected in buffer.

CV_DYN_LIST Dynamic list name specification

CV_DYN_DIR Default directory

CV_BASE Default data base name

CV_NODE Default node name

FUNCTION Set buffer size of scatter buffer. The buffer size is specified as number of scatter points to be stored in the buffer. Note that there is at least one point per scatter frame. If there are bit variables, several points per frame might be sent. The buffer size must not be set below the maximum number of points for a picture.

REMARKS Module is an action routine.

EXAMPLE -

SET SPECTRUM POINT

SET SPECTRUM POINT spectrum xpoint ypoint file
spec_dir base node
/[NO]KEEP_MAP

PURPOSE	Set spectrum channel to specified value.
PARAMETERS	
spectrum	Name of spectrum to be set.
xpoint	List of x-values converted into spectrum channels.
ypoint	List of y-values used as spectrum contents.
file	Name of file, used to read X-Y values.
spec_dir	Default spectrum directory.
base	Default data base name.
node	Default node name.
/KEEP_MAP	Keep Data Base mapping context.
Caller	MDBM,MGOODBM,E$DECMD
Author	W. Spreng

Example

1. SET SPECTRUM POINT a 300,400,500 3.7,5.8,10.5
The spectrum points 300,400,500 are converted into spectrum indices of spectrum "a" and the values 3.7,5.8,10.5 are written into these spectrum bins.
2. SET SPECTRUM POINT a FILE=daten.dat
The X-Y values are read from the specified file.
3. SET SPECTRUM POINT a 100,200 20.6,40.8 FILE=daten.dat
The specified x-y values and the values read from file are put into the spectrum "a".

Remarks

File name E$SSPPO.PPL

Created by GOO$DE:D$DECMD.PPL

Description

CALLING STS=E$SSPPO(CV_spectrum,RA_XPOINT,RA_YPOINT,CV_FILE,
CV_spec_dir,CV_base,CV_node,I_keep_map,LA_ELEMENTS)

COMMAND SET SPECTRUM POINT spectrum xpoint ypoint file
spec_dir base node
/[NO]KEEP_MAP

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. String

Name of the spectrum in which the points should be set. Up to now only one dimensional spectra are supported.

XPOINT

Routine arg. Input (*) BIN FLOAT(24)

Command par. String

List of x-values, which are converted into the corresponding spectrum indices. Several values, separated by commas can be specified.

YPOINT

Routine arg. Input (*) BIN FLOAT(24)

Command par. String

A list of values, which are put into the spectrum channels as given in the parameter "XPOINT". The number of X and Y values have to be the same!

FILE

Routine arg. Input CHAR(*) VAR

Command par. String

Name of the file, used to read the X and Y-values. The format of the file is very simple: A "!" at the begin of each record indicates a comment line. In each record containing a valid input two values are expected. The first value is the x-value and the second separated by blanks or comma is the corresponding y-value.

SPEC_DIR

Routine arg. CHAR(*) VAR

Command arg. String global replaceable default=$SPECTRUM

Default Directory name for spectra.

BASE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=*

Node name for Data Base section file.

KEEP_MAP

Routine arg. Input BIN FIXED(15), valid inputs are 0 and 1

Command par. Switch

If this flag is set the context of all Data bases will be kept:

/KEEP_MAP Keep mapping context.

/NOKEEP_MAP Database will be detached.

FUNCTION

The specified points are set in the spectrum. The x-values are converted into a valid spectrum index and the corresponding y-value is written into that bin. The x and y-pairs can be specified directly and/or read from a file with a simple input format:

A "!" at the begin of each record indicates a comment line. In each record containing a valid input two values are expected. The first value is the x-value and the second, separated by blanks or comma, is assumed to be the corresponding y-value.

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 ALIAS

SHOW ALIAS name environment
/GLOBAL/ACTIVE [=SCOPE]

PURPOSE Show alias name (in MUTIL, GOOSY, MDBM, MDISP).

PARAMETERS

name required string default: ”
Name of alias. Valid characters as for logical names. Asterisk for all names.

environment string replace default: ”
Optional name of environment in which the alias is valid.

/GLOBAL qualifier set
Show global alias names.

/ACTIVE qualifier set
Show active alias names (default).

Caller MUTIL, GOOSY, MDBM, MDISP

Author H.G.Essel

Example

Show global alias names only:
SHOW ALI /GLOB
Show environment alias names only:
SHOW ALI ENV=susi
Show all active alias names:
SHOW ALI
Show one alias name:
SHOW ALI CS

Remarks

File name	U$SHALI.PPL
Created by	GOO$UTIL:U$ALICM

Description

CALLING	STS=U$SHALI(CV_alias,CV_environment,CV_scope)
COMMAND	SHOW ALIAS name environment /GLOBAL/ACTIVE [=SCOPE] Argument / Parameter description.

ALIAS

Routine arg.	Input CHAR(*) VAR
Command par.	required string default: ” Name of alias. Valid characters as for logical names. Asterisk for all alias names.

ENVIRONMENT

Routine arg.	Input CHAR(*) VAR
Command par.	string replace default: ” Specifies the environment in which the alias is valid.

SCOPE

Routine arg.	Input CHAR(*) VAR
Command par.	set default: ’/ACTIVE’
/GLOBAL	Show global alias names.
/ACTIVE	Show active alias names.

Function

Show a logical name in table LNM$ENV_env or LNM$GOOSY.

SHOW ANALYSIS

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

PURPOSE	Show analysis status
timer	integer optional time intervall [sec]for /RATE default is 5 sec.
output	string optional output file
/PRINT	print output
/BRIEF	Brief output
/CLEAR	clear counters
/OUTFILE	Show current output file header
/INFILE	Show current input file header
/[NO] RATE	Show data rate
PURPOSE	Show analysis status
EXAMPLE	SHOW ANA

Description

FUNCTION	This procedure writes some information about the present analysis status to terminal
File name	I$ANACM.PPL

Action rout.	I$ANACM_SHOW
Version	1.01
Author	H.G.Essel
Last Update	12-APR-1985

SHOW AREA

```
SHOW AREA area base output
    /[NO]FULL
    /[NO]DIRECTORY
    /[NO]KEEP_MAP
    /PRINT
```

PURPOSE Show an Area in a Data Base

PARAMETERS

area Area name
required common default

base Data Base name
required common default

output optional output file
optional

/[NO] FULL Show full Area information
default: '/NOFULL' Area information without bit map

/[NO] DIRECTORY Show Area directory
default: '/NODIRECTORY'

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

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHAR.PPL

Dataset -

EXAMPLE SHO AR ALPHA OUT='ALPHA.LIS'
show the Area 'ALPHA' of Data Base 'DB' and write
the output into file 'ALPHA.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING STS=M$ASHAR(CV_AREA,CV_BASE,CV_OUT,I_FULL,
I_DIRECTORY,I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_AREA I Area name
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

CV_OUT I optional file for output
CHAR(*) VAR

I_FULL I Show full information
BIN FIXED(15)

I_DIRECTORY I Show Area Directory
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

I_PRINT I Print output
BIN FIXED(15)

FUNCTION Show an Area in a Data Base

REMARKS Module is an action routine.

EXAMPLE -

SHOW BUFFER DUMP

SHOW BUFFER DUMP

PURPOSE	Show writing to LMD file.
PARAMETERS	
EXAMPLE	SHO BUF DUMP

Description

FUNCTION	Show writing LMD file.
File name	I$ANACM.PPL
Action rout.	I$ANACM_LMD_SHOW
Version	1.01
Author	H.G.Essel
Last Update	12-Mar-1993

SHOW CALIBRATION

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

PURPOSE	Show calibration information
PARAMETERS	
calibration	Name of calibration.
output	Output file
cal_dir	Default calibration directory
base	Default data base name
node	Default node name
/PRINT	Output file is printed
/LINKS	Links to all spectra are listed.
/TABLE	Show total table contents.
Caller	MDBM, MGOODBM
Author	W. Spreng

Examples

1.) SHOW calibration [*]*
All calibrations in all existing directories are listed.
2.) SHOW calibration [*]A* /LINKS
Calibrations starting with "A" are listed, additionally all spectra calibrated with these data elements are listed.

Remarks

Created by E$SHECM.PPL

Module name GOO$DE:E$SHCAL.PPL

Description

CALLING STS=E$SHCAL(CV_calibration,CV_output,CV_cal_dir, CV_base, CV_node, I_print, I_links, I_table)

COMMAND SHOW CALIBRATION calibration output cal_dir base node
/PRINT
/LINKS
/TABLE

CALIBRATION

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of calibration which should be shown. Any wildcardspecification is allowed.

OUTPUT

Routine arg. Input CHAR(*) VAR

Command par. String

File in which the output should performed. The output will be printed with the /PRINT switch.

CAL_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$CALIB

Default Picture Directory.

BASE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

PRINT

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

The generated output is printed on the default print queue.

LINKS

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

All established links to spectra are shown. Therefore all spectra connected to the specified calibration are listed.

TABLE

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

The contents of the calibration table is listed.

Function

Informations about calibrations should be shown. All existing links to spectra and/or the total table contents can be listed.

SHOW CAMAC SPECTRUM

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

PURPOSE show CAMAC spectra

PARAMETERS

name String default: '*'
name of spectrum. Wildcards are supported:
* x* *x *x* x*y Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

spec_dir String replace default: '$SPECTRUM'
Name of Spectrum directory.

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

node String replace default: '*'
Name of Node.

output String default: none
optional output file

width Integer global replace default: 80
Line length for histogram (80 or 132)

/PRINT	Switch default: none Print output
/WHAT	Set replace default: /ATTRIBUTES /ATTRIBUTES :Display attributes of spectra /DATA :Display data /STATUS :calls SHOW TAB/SP /ALL :Display all
/FULL	Switch default: none Display full spectrum information
/MEMBERS	Switch default: none Display information for all spectrum members
/LOG	Switch default: none Display data in logarithmic format
/INTEGRAL	Switch default: none Display channel contents integral
/ZERO	Switch default: none Display channel contents including zero channels
/CAMAC	Switch default: /CAMAC Display data of MR2000 memory (=default)
/[NO] KEEP_MAP	Switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
FUNCTION	The specified spectrum is accessed and various parts of it are displayed on the terminal. The data part of the spectrum is fetched from the MR2000 memory optionally.
Caller	E$SHECM
Author	K.WINKELMANN
File name	E$SHCSP.PPL
Dataset	GOO$DE:E$SHCSP.PPL

Function

The specified spectrum is accessed and various parts of it are displayed on the termina. AS long as the data of the spectrum is not accessed, the command works like SHOW SPECTRUM /CAMAC. The data filed of the spectrum may, however, be fetched from the MR2000 memory rather then from the data base.

Example

SHOW CAM SPEC OTTO /DATA
shows MR2000 data of spectrum otto
SHOW CAM SPEC OTTO /NOCAMAC /DATA
shows data base data of spectrum otto
SHOW CAM SPEC *
shows list of all CAMAC spectra.

Remarks

REMARKS Module is an action routine.

Created by E$SHECM.PPL

Description

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

CALLING STS=E$SHCSP(CV_NAME,CV_DIR,CV_BASE,CV_NODE,CV_OUT,
L_WIDTH,I_PRINT,CV_DETAIL,I_FULL,
I_MEMBERS,I_LOG,I_INTEGRAL,I_ZERO,
I_CAMAC,I_KEEP_MAP,B_MASK)
Argument description

NAME

Type Input CHAR(*) VAR
Name of spectrum. Wildcards are supported:
* x* *x *x* x*y Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

DIR

Type Input CHAR(*) VAR
Default directory

BASE

Type Input CHAR(*) VAR
Default data base

NODE

Type Input CHAR(*) VAR
Default node

OUTPUT

Type Input CHAR(*) VAR
Optional file for output

WITDH

Type Input BIN FIXED(31)
Line size for histogram (80 or 132).

PRINT

Type Input BIN FIXED(15)
valid values are:
No print of output file is done Print output file on line printer

WHAT

Type Input CHAR(*) VAR
valid values are:
Display attributes of the spectrum Display data of the spectrum Display attributes,all and status of the spectrum Display status of spectrum

MEMBERS

Type Input BIN FIXED(15)
valid values are:

Display only first and last member of spectrum. Display information about all spectrum members.

FULL

Type Input BIN FIXED(15)
valid values are:
Display only short info about the spectrum: Spectrumname,spectrumtype,limits,bins,bits
Display full information about the spectrum: Spectrumname,spectrumtype,Database,Dire
Dimensions, Index in analyzetable, Creationdate Lettering of X-Axis,Lettering of Y-Axis, Lower limit, upper limit, Number of bins, Binsize, Factor, Offset, Underflow - and Overflow counts, Underflow and Overflow contents, freeze and executed bit

LOG

Type Input BIN FIXED(15)
Valid values are:
Display decimal data Display logarithmic data

INTEGRAL

Type Input BIN FIXED(15)
Valid values are:
Display channel contents Display integral channel contents

ZERO

Type Input BIN FIXED(15)
Valid values are:
Display channel contents without zero's Display channel contents with zero's

CAMAC

Type Input BIN FIXED(15)
Valid values are:
Display data of spectrum in data base Display data of spectrum in MR2000 memory

KEEP_MAP

Type Input BIN FIXED(15)
Valid values are:
Unmap Data Base Inhibit unmap of the Data Base

B_MASK

Type Output BIT(32) ALIGNED

Function

The specified spectrum is accessed and various
parts of it are displayed on the terminal. The difference to SHOW SPECTRUM is that the contents of CAMAC spectra in the MR2000 memory can be output.

Remarks

Module is an action routine.

Example

SHOW CONDITION

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

PURPOSE show attributes of a condition

PARAMETERS

name String replace default: '*'
Name of condition. Wildcards are supported:
* x* *x *x* x*y
Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.
default=*

cond_dir String replace default: '$CONDITION'
Default Directory

base String replace default: 'DB'
Default Data Base

node String replace default: '*'
Default node

output String replace default: none
Optional output file

/PRINT Switch default: none
Print output

/FULL Switch default: none
Full output

/MEMBERS Switch default: none
Output all members of indexed conditions.

what Set default: none

/ATTRIBUTES	show all attributes of a condition
/COUNTERS	show all counters of a condition
/FLAGS	show all flags
/FULL	show everything
/STATUS	show bit tables and counters

/all show all of them Show bit tables and counters

TYPE Set default: /ANY

/WIND	show window conditions
/MULTI	show multi window conditions
/PATT	show pattern conditions
/COMP	show composed conditions
/POLY	show polygon conditions
/FUNC	show function conditions
/ANY	show all types of conditions

Show type of conditions

KEEP_MAP switch default: /KEEP_MAP

/NOKEEP_MAP	Unmap the Data Base
/KEEP_MAP	inhibit the unmap (detach) of the Base

FUNCTION Condition 'name' is searched and its attributes are listed, esp. type, window limits, pattern bits, related conditions, condition table name

EXAMPLE SHOW CONDITION emil /ATTR
shows all attributes from condition emil

Caller E$SHECM

Author K.Winkelmann

File name E$SHCO.PPL

Dataset -

Description

COMMAND

```
SHOW CONDITION name cond_dir base node output
   /PRINT
   /FULL
   /MEMBERS
   /ATTRIBUTES/COUNTERS/FLAGS/STATUS/ALL
   /POLY/WIND/MULTI/PATT/COMP/FUNC/ANY
   /[NO]KEEP_MAP
```

CALLING

```
STS=E$SHCO(CV_NAME,CV_DIR,CV_BASE,
                CV_NODE,CV_OUT,I_FULL,I_MEMBERS,
                CV_DETAIL,CV_TYPE,,,,,,,,,,,,,,,,,,,,,,
                I_PRINT,I_KEEP_MAP,B_MASK)
```

Argument description

NAME

Type Input CHAR(*) VAR

Name of condition. Wildcards are supported:
* x* *x *x* x*y

Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

DIR

Type Input CHAR(*) VAR

Default directory name

BASE

Type Input CHAR(*) VAR

Default base name

NODE

Type Input CHAR(*) VAR

Default node

OUT

Type Input CHAR(*) VAR

Optional file for output

MEMBERS

Type	Input BIN FIXED(15) Valid values are:
0	Output first and last members of indexed conditions.
1	Output all members of indexed conditions.

FULL

Type	Input BIN FIXED(15) Valid values are:
0	Short output is done.
1	Full output is done. Switch for full output

DETAIL

Type	Input CHAR(*) VAR
/ATTRIBUTES'	
/COUNTERS'	
/FLAGS'	
/STATUS'	
/ALL'	
	Detailness

TYPE

Type	Input CHAR(*) VAR
'/WINDOW'	Window condition
'/PATTERN'	Pattern condition
'/COMPLEX'	Complex condition
'/FUNCTION'	Function condition
'/POLYGON'	Polygon condition

'/ANY'	Any condition type Select single condition types

PRINT

Type	Input BIN FIXED(15) Valid values are:
0	No print is done.
1	Print output file on Line printer. Switch for print output

KEEP_MAP

Type	Input BIN FIXED(15) Valid values are:
0	Deattach of the Data Base.
1	Inhibit unmap of Data Base.

MASK

Type	Input BIT(32) ALIGNED

Function

Condition 'name' is searched and its attributes are listed, esp. type, window limits, pattern bits, related conditions, condition table name

Remarks

Module is an action routine

Example

—

SHOW DEVICES

SHOW DEVICE

PURPOSE Show all allocated user devices

Remarks

File name D$SHALL.PPL

Description

CALLING STS=D$SHALL()

COMMAND SHOW DEVICE

Function

Show all allocated graphical devices. The information on the devices allocated in the current session is fetched from the device description table and displayed. The logical device names, the physical address, the device type and category (input or output) and the GOOSY main device type are listed.

SHOW DIRECTORY

SHOW DIRECTORY dir base output
/[NO]FULL
/[NO]DIRECTORY
/[NO]KEEP_MAP
/PRINT

PURPOSE Show Data Elements of a Data Base Directory

PARAMETERS

dir Directory name
required common default

base Data Base name
required common default

output optional output file

/[NO] FULL Show full Directory information
default: '/NOFULL' brief Directory information

/[NO] DIRECTORY Show Master Directory
default: '/NODIRECTORY'

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

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHDI.PPL

Dataset -

EXAMPLE SHO DIR EVE OUTP='EVE.LIS' /FULL
show the full information about the Directory 'EVE'
of the Data Base 'DB' and write the output into the file 'EVE.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING STS = M$ASHDI(CV_DIR,CV_BASE,CV_OUT,
I_FULL,I_DIRECTORY,I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_DIR I Directory name
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

CV_OUT I Optional file for output
CHAR(*) VAR

I_FULL I Show full Directory information
BIN FIXED(15)

I_DIRECTORY I Show Master Directory
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

I_PRINT I Print output
BIN FIXED(15)

FUNCTION Show Data Elements of a Data Base Directory

REMARKS Module is an action routine.

EXAMPLE -

SHOW DISPLAY GLOBALS

```
SHOW DISPLAY GLOBALS
    /SPECTRUM
    /PICTURE
```

PURPOSE Show global display parameter.

PARAMETERS

/SPECTRUM Global parameter for spectra are listed.

/PICTURE Global parameter for pictures are listed.

Caller MDISP,MGOODISP,D$DSPCM

Author W. Spreng

Remarks

File name D$SDIGL.PPL

Created by GOO$DISP:D$DSPCM.PPL

Description

CALLING STS=D$SDIGL(I_spectrum,I_picture)

COMMAND

```
SHOW DISPLAY GLOBALS
    /SPECTRUM
    /PICTURE
```

SPECTRUM

Routine arg. Input BIN FIXED(15) valid values are 0 and 1.

Command par. Switch

The global parameters for spectra, define by the DEFINE DISPLAY SPECTRUM command are listed.

PICTURE

Routine arg. Input BIN FIXED(15) valid values are 0 and 1.

Command par. Switch

The global parameters for spectra, define by the DEFINE DISPLAY PICTURE command are listed.

FUNCTION

The global display parameters are listed as defined with the DEFINE DISPLAY SPECTRUM and DEFINE DISPLAY PICTURE command.

SHOW DYNAMIC ATTACHED

SHOW DYNAMIC ATTACHED dyn_list dyn_type dyn_dir base node output
/PRINT
/[NO]QUEUE
/FULL

PURPOSE Show attached dynamic list

PARAMETERS

dyn_list Dynamic list name specification
required
common default
If *, all attached lists are shown.

dyn_type Type of dynamic sublist or *
default:'*'

dyn_dir Default directory
default:'$DYNAMIC'
common default

base Default data base name
default:'DB'
common default

node Default node name
default:'E'
common default

output Optional output file

/PRINT Print output

/[NO] QUEUE Display queue content

/FULL Display full queue content

EXAMPLE -

Caller M$DLCMD

Author H.G.Essel

File name M$ASHLL.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ASHLL(CV_DYN_LIST,CV_TYPE,
CV_DYN_DIR,CV_BASE,CV_NODE,CV_OUT,
I_PRINT,I_QUEUE,I_FULL)

ARGUMENTS

CV_DYN_LIST Dynamic list name specification
CHAR(*) VAR
Input

CV_TYPE Type of sublist or *
CHAR(*) VAR
Input

CV_DYN_DIR Default Directory
CHAR(*) VAR
Input

CV_BASE Default Data Base name
CHAR(*) VAR
Input

CV_NODE Default node name
CHAR(*) VAR
Input

CV_OUT Optional output file
CHAR(*) VAR
Input

I_PRINT	If 1, print output
I_QUEUE	If 1, the content of the queues is displayed.
I_FULL	If 1, the content of the queues is fully displayed.
FUNCTION	Show local attached dynamic list If dynamic list name is *, all attached lists are shown.
REMARKS	Module is an action routine.
EXAMPLE	-

SHOW DYNAMIC LIST

SHOW DYNAMIC LIST dyn_list dyn_type dyn_dir base node output
/[NO]KEEP_MAP
/PRINT

PURPOSE Show Dynamic List (elements)

PARAMETERS

dyn_list Dynamic List name specification
required common default

dyn_type Type of dynamic sublist
common default:'*'

dyn_dir Default Directory
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

output optional output file
optional

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

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHDL.PPL

Dataset -

EXAMPLE SHO DYN LIS DYNA1 $DYNAMIC DB OUTP='DYNA1.LIS'
show the full information about the Dynamic List 'DYNA1' in Directory 'DYNAMIC' of the Data Base 'DB' and write the output into the file 'DYNA1.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING STS=M$ASHDL(CV_DYN_LIST,CV_DYN_TYPE,
CV_DYN_DIR,CV_BASE,CV_NODE,CV_OUT,
I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_DYN_LIST Dynamic List name specification
CHAR(*) VAR
Input

CV_DYN_TYPE Type of dynamic sublist
CHAR(*) VAR
Input

CV_DYN_DIR Default Directory
CHAR(*) VAR
Input

CV_BASE Default Data Base name
CHAR(*) VAR
Input

CV_NODE Default node name
CHAR(*) VAR
Input

CV_OUT optional file for output
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

I_PRINT	Print output BIN FIXED(15) Input
FUNCTION	Show Dynamic List (elements)
REMARKS	Module is an action routine.
EXAMPLE	-

SHOW ELEMENT

SHOW ELEMENT name dir base node output
/DECIMAL/HEXADECIMAL/OCTAL/BINARY
/LONGWORD/WORD/BYTE
/[NO]DATA
/[NO]FULL
/[NO]KEEP_MAP
/PRINT

PURPOSE Show a Data Element descriptor and value

PARAMETERS

name Node::base:[dir]name(i)->type(i)
dir, name, type, and index might be given with wildcards '*'
required common default

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

output optional output file
optional
DECIMAL/HEXADECIMAL/OCTAL/BINARY
: Output radix for Data Types
replaced default:'/DECIMAL'

/LONGWORD/WORD/BYTE output format for atomic Data Type Y
replaced default:'/LONGWORD'

/[NO] DATA Shows all the data of a Data Element
default:'/NODATA'

/[NO] FULL Shows all control information of a Data Element
default:'/NOFULL'

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

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHDE.PPL

Dataset -

EXAMPLE SHO ELEM DB:[EVE]ADAM(17) OUTP='ADAM.LIS' /DATA
show the Data Element with the index '17' of the
name array 'ADAM' in Directory 'EVE' of Data Base 'DB' and write the output to the file 'ADAM.LIS' of the default VAX/VMS directory. The data will be shown but only brief control information.
SHO ELEM DB:[EVE]*A*
show all Data Elements of Directory 'EVE'
containing an 'A' in their names. The output will be without data in short form.
SHO ELEM DB:[EVE]BETA->*
show all queued Data Elements of Data Element
'BETA' in Directory 'EVE' of Data Base 'DB'. The output will be without data in short form.

Remarks

REMARKS -

Description

CALLING STS=M$ASHDE(CV_NAME,CV_DIR,CV_BASE,
CV_NODE,CV_OUT,CV_RADIX,CV_OUT_Y,
I_DATA,I_FULL,I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_NAME I Node::base:[dir]name(i)->type(i)
CHAR(*) VAR

CV_DIR	I Default Directory CHAR(*) VAR
CV_BASE	I Default Data Base name CHAR(*) VAR
CV_NODE	I Default node name CHAR(*) VAR
CV_OUT	I optional file for output CHAR(*) VAR
CV_RADIX	I Output radix for Data Types CHAR(*) VAR
CV_OUT_Y	I Output format for atomic Data Type Y CHAR(*) VAR
I_DATA	I Show Data Element with all its data BIN FIXED (15)
I_FULL	I Show all Data Element control information BIN FIXED (15)
I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
I_PRINT	I Print output BIN FIXED(15)
FUNCTION	Show a Data Element descriptor and value
REMARKS	Module is an action routine.
EXAMPLE	-

SHOW GOOSY STATUS

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

PURPOSE	GOOSY Status report
ARGUMENTS	
environment	Optional environment
p1	Process name to be monitored
p2	Process name to be monitored
p3	Process name to be monitored
/$TMR	Monitor $TMR of environment
/$ANL	Monitor $ANL of environment
Information	See HELP MSTATUS, HELP GOOCONTROL
EXAMPLE	GOOSY> SHOW G ST P1=GN_HGE___$TMR $ GSTAT P1=GN_HGE___$TMR $ GSTAT SUSI /$TMR/$ANL
Action rout.	M$GOOST
Author	H.G.Essel

Remarks

The same action can be invoced by the DCL symbols
MSTATUS SHO GOOSY ST proc_1 proc_2 proc_3
or GSTATUS proc_1 proc_2 proc_3

File name	M$GOOST.PPL

Description

CALLING STAT=M$GOOST(CV_environ
,CV_process_1,CV_process_2,CV_process_3
,I_tmr,I_anl)

ARGUMENTS -

CV_environ I: Optional environment name

CV_process_1 I: Process name to be monitored

CV_process_2 I: Process name to be monitored

CV_process_3 I: Process name to be monitored

I_tmr I: if 1, monitor $TMR of environment

I_anl I: if 1, monitor $ANL of environment

FUNCTION Two types of processes can be monitored:
TMR transport manager
ANL analysis programs
Presently a maximum of three processes can be monitored. Only one of them can be a TMR. The processes must run on the same node in the same group. The logical name GOO$CONTROL must be defined as the name of a data base, which must be created before starting the TMR and analysis programs. This should be done using the GOOCONTROL command, which should be included in the LOGIN.COM. The TMR and analysis programs create data elements in this data base and deposit their control information. The names of the data elements are the process names.

REMARKS -

EXAMPLE -

SHOW HOME_BLOCK

SHOW HOME_BLOCK base output
/PRINT
/BITMAP

PURPOSE	Show the Home Block of a Data Base
PARAMETERS	
base	Data Base name required common default
output	optional output file optional
/BITMAP	output bitmap
/PRINT	Print output
Caller	M$SHMCM
Author	M. Richter
File name	M$ASHHB.PPL
Dataset	-
EXAMPLE	SHO HOM DB OUTP='HBDB.LIS' show the Home Block Information of Data Base 'DB' and write it into the file 'HBDB.LIS' of the default VAX/VMS directory.

Remarks

REMARKS	-

Description

CALLING STS=M$ASHHB(CV_BASE,CV_OUT,I_BITMAP,I_PRINT)

ARGUMENTS

CV_BASE I Data Base name
CHAR(*) VAR

CV_OUT I optional file for output
CHAR(*) VAR

I_BITMAP Output bitmap
BIN FIXED(15)

I_PRINT I Print output
BIN FIXED(15)

FUNCTION Show the Home Block of a Data Base

REMARKS Module is an action routine.

EXAMPLE -

SHOW LINK

SHOW LINK link_from dir base node
output
/IN/OUT/ALL
/MATCH/TREE
/[NO]KEEP_MAP
/PRINT

PURPOSE Show Data Element link(s)

PARAMETERS

link_from Source Data Element name specification
required common default

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

output optional output file
optional

/IN/OUT/ALL In, out, all links
replaced default:'/ALL'

/MATCH/TREE Matching or total link tree
default:'/MATCH'

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

/PRINT Print output

Caller M$DMCMD

Author	M. Richter
File name	M$ASHLI.PPL
Dataset	-
EXAMPLE	SHO LINK DB:[EVE]ADAM /OUT OUT='ADAM.LIS' show all outgoing links from Data Element 'ADAM' in Directory 'EVE' of Data Base 'DB' and write the output into the file 'ADAM.LIS' of the VAX/VMS default directory.

Remarks

REMARKS -

Description

CALLING STS=M$ASHLI(CV_LINK_FROM,CV_DIR,
CV_BASE,CV_NODE,CV_OUT,CV_SELECT,CV_ACTION,
I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_LINK_FROM	I Source Data Element name specification CHAR(*) VAR
CV_DIR	I Default Directory CHAR(*) VAR
CV_BASE	I Default Data Base name CHAR(*) VAR
CV_NODE	I Default node name CHAR(*) VAR
CV_OUT	I optional file for output CHAR(*) VAR
CV_SELECT	I In, out, all links CHAR(*) VAR
CV_ACTION	I Matching or total link tree CHAR(*) VAR
I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)

I_PRINT	I Print output BIN FIXED(15)
FUNCTION	Show Data Element link(s)
REMARKS	Module is an action routine.
EXAMPLE	-

SHOW LOCKS

SHOW LOCKS process lock request granted
/PROCESS
/LOCKS
/EXCLUSIVE

PURPOSE Display system locks

PARAMETERS

process string replace default=*
Process name (wildcard)

lock string replace default=*
Lock name (wildcard)

request string replace
Select lock mode request

granted string replace
Select lock mode granted

/PROCESS switch
Sort by process names

/LOCKS switch
Sort by lock names

/EXCLUSIVE switch
Specified locks are to be excluded

Caller MLOCKS

Author Gertrud Schneider

Example

SHO LOCKS GOOFY DB* /LOCKS
Shows all locks of process GOOFY with lockname beginning with DB... sorted by lockname

Remarks

File name U$LOCKCM.PPL

Created by U$LOCKCM.PPL

Description

CALLING STS=U$LOCKS(CV_process,CV_lock,
CV_request,CV_granted,
I_PROCESS,I_LOCKS,I_EXCL)

COMMAND SHOW LOCKS process lock request granted
/PROCESS
/LOCKS
/EXCLUSIVE
Arguments/Parameters description

PROCESS

Routine arg. Input CHAR(*) VAR

Command par. string replace default=*
Process name, wildcard allowed

LOCK

Routine arg. Input CHAR(*) VAR

Command par. string replace default=*
Lock name, wildcard allowed

REQUEST

Routine arg. Input CHAR(*) VAR

Command par. string replace
Select lock mode request : NL,CR,CW,PR,PW,EX

GRANTED

Routine arg. Input CHAR(*) VAR

Command par. string replace
Select lock mode granted : NL,CR,CW,PR,PW,EX

/PROCESS

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch
/PROCESS	Sort by process names

/LOCKS

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch
/LOCKS	Sort by lock names

/EXCLUSIVE

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch
/EXCLUSIVE	All specified locks are to be excluded

Function

The routine gets all locks in the system with SYS$GETLKI in a loop. If the given specification matches the lock, it is stored in a based structure After a sorting pass, if specified, the locks are displayed using the SMG routines. If there is more data than could displayed on one screen you can use several keypad keys and the cursor positioning keys to controll scrolling.

KP4 direction down
KP5 direction up
KP7 move 100 lines
KP8 move 15 lines
KP1 shift right
KP2 shift left
PF1 + KP4 move to bottom
PF1 + KP5 move to top
PREV SCREEN move 15 lines up
NEXT SCREEN move 15 lines down
^ or v move 5 lines up or down
> or < move to the right or left margin

SHOW MAPPING

SHOW MAPPING base area

PURPOSE	Show mapping of a Data Base
PARAMETERS	
base	Data Base name required common default
area	optional area string optional
Caller	M$SMAPCM
Author	H.G.Essel
File name	M$ASHMAP.PPL
Dataset	-
EXAMPLE	SHO MAP DB

DESCRIPTION

CALLING	STS=M$ASHMAP(cv_dbnam,CV_area)
ARGUMENTS	
cv_dbnam	I Name of Data Base. A logical name translation will be performed. CHARACTER (*) VAR
CV_area	I optional area string. All areas containing the string will be listed. CHARACTER (*) VAR
FUNCTION	The routine will show the mapping of all areas. When an area is mapped, the name is displayed. The access (read or write) is displayed. No change in mapping is done.
REMARKS	Names will be converted to upper case characters only.

SHOW MAPPING

SHOW MAPPING base area

PURPOSE	Show mapping of a Data Base
PARAMETERS	
base	Data Base name required common default
area	optional area string optional
Caller	M$SMAPCM
Author	H.G.Essel
File name	M$ASHMAP.PPL
Dataset	-
EXAMPLE	SHO MAP DB

DESCRIPTION

CALLING	STS=M$ASHMAP(cv_dbnam,CV_area)
ARGUMENTS	
cv_dbnam	I Name of Data Base. A logical name translation will be performed. CHARACTER (*) VAR
CV_area	I optional area string. All areas containing the string will be listed. CHARACTER (*) VAR
FUNCTION	The routine will show the mapping of all areas. When an area is mapped, the name is displayed. The access (read or write) is displayed. No change in mapping is done.
REMARKS	Names will be converted to upper case characters only.

SHOW MAPPING

SHOW MAPPING base area

PURPOSE	Show mapping of a Data Base
PARAMETERS	
base	Data Base name required common default
area	optional area string optional
Caller	M$SMAPCM
Author	H.G.Essel
File name	M$ASHMAP.PPL
Dataset	-
EXAMPLE	SHO MAP DB

DESCRIPTION

CALLING	STS=M$ASHMAP(cv_dbnam,CV_area)
ARGUMENTS	
cv_dbnam	I Name of Data Base. A logical name translation will be performed. CHARACTER (*) VAR
CV_area	I optional area string. All areas containing the string will be listed. CHARACTER (*) VAR
FUNCTION	The routine will show the mapping of all areas. When an area is mapped, the name is displayed. The access (read or write) is displayed. No change in mapping is done.
REMARKS	Names will be converted to upper case characters only.

SHOW MEMBER

SHOW MEMBER mem_spec dir base node output
/[NO]KEEP_MAP
/PRINT

PURPOSE Show value of a Data Element member or a full Data Element member array (wildcarded).

PARAMETERS

mem_spec Node::base:[dir]name(i)->type(i)
required

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

cv_out optional file for output
optional

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

/PRINT Print output
BIN FIXED(15)

Caller M$DMCMD

Author M. Richter

File name M$ASHME.PPL

Dataset -

EXAMPLE SHO MEM DB:[EVE]ADAM(12).KAIN OUT='ADAM.LIS'
show the member 'KAIN' of the Data Element with the index '12' of the name array 'ADAM' in the Directory 'EVE' of the Data Base 'DB' and write the output into the file 'ADAM.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING STS=M$ASHME(CV_ELEMENT,CV_DIR,CV_BASE,
CV_NODE,CV_OUT,I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_ELEMENT I Node::base:[dir]name(i)->type(i)
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

CV_OUT I Optional output file
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

I_PRINT I Print output file
BIN FIXED (15)

FUNCTION Show value of a Data Element member

REMARKS Module is an action routine.

EXAMPLE -

SHOW PICTURE

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

PURPOSE	Show picture information
PARAMETERS	
picture	Name of picture.
output	Output file
pic_dir	Default picture Directory
base	Default Data Base name
node	Default node name
/PRINT	Output file is printed
/FULL	Short information about all frames is shown.
/DATA	Detailed information about all frames is shown.
/KEEP_MAP	Inhibit the unmapping of the whole Data Base.
Caller	MDBM, MGOODBM
Author	W. Spreng

Example

1.) SHOW PICTURE pic [*]*
All pictures in all existing Directories are listed.

Remarks

File name D$SHPI.PPL

Created by GOO$DE:E$SHECM.PPL

Description

CALLING

```
STS=D$SHPI(CV_picture,CV_output,CV_pic_dir,
           CV_base,CV_node,I_full,I_data,I_keep)
```

COMMAND

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

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of picture for which informations should be shown. Wildcards in Directory and picture name are supported.

OUTPUT

Routine arg. Input CHAR(*) VAR

Command par. String

File in which the output should performed. The output will be printed with the /PRINT switch.

PIC_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$PICTURE

Default picture Directory.

BASE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

PRINT

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

The generated output is printed on the default print queue.

FULL

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

If set information for each frame the spectrum or scatterplot parameter are listed.

DATA

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

If set the whole frame parameter, e.g specified and last window, display modes of the axis, all defined overlays etc., are listed.

KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch default = /KEEP_MAP

If set the mapping context will be kept. If /NOKEEP_MAP specified the database will be dettached after the command completion.

Function

Information about the picture frames are displayed on the terminal. If '/FULL' is specified the spectra and scatterplot parameters for all frames are listed. If additionally '/DATA' is specified all user specified picture data are shown.

SHOW POLYGON

SHOW POLYGON name poly_dir base node output
/PRINT
/[NO]FULL
/[NO]DATA
/[NO]KEEP_MAP

PURPOSE show attributes of a polygon

PARAMETERS

name name of polygon. Wildcards are supported:
* x* *x *x* x*y

poly_dir default Directory
replace default='$PICTURE'

base default Data Base
replace default='DB'

node default node
replace default='E'

output optional output file

/PRINT print output

/[NO] FULL show full header

/[NO] DATA show polygon points

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

EXAMPLE SHOW POLY emil /DATA
shows points of polygon emil

Caller E$SHECM

Author H.G.Essel

File name E$ASHPO.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=E$ASHPO(CV_NAME,CV_DIR,CV_BASE,
CV_NODE,CV_OUT,I_FULL,I_DATA,
I_PRINT,I_KEEP_MAP,B_MASK)

ARGUMENTS

CV_NAME name of polygon. Wildcards are supported:
* x* *x *x* x*y
CHAR(*) VAR
Input

CV_DIR default directory name
CHAR(*) VAR
Input

CV_BASE default base name
CHAR(*) VAR
Input

CV_NODE default node
CHAR(*) VAR
Input
optional file for output
CHAR(*) VAR
Input

I_FULL switch for full output
BIN FIXED(15)
Input

I_DATA switch to show polygon points
BIN FIXED(15)
Input

I_PRINT	Print output BIN FIXED(15) Input
I_KEEP_MAP	Inhibit unmap of Data Base BIN FIXED (15) Input
B_MASK	mask which params were specified BIT(32) ALIGNED Output
FUNCTION	polygon 'name' is searched and its attributes are listed
REMARKS	-
EXAMPLE	-

SHOW POOL

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

PURPOSE Show Areas of a Data Base Pool
The name, size, filling level, cluster size, and number of fragments are shown for each Area. The Pool Directory can be shown with /DIRECTORY.

PARAMETERS

pool Pool name, wild cards are allowed.
The Pool name will be ignored for /DIRECTORY.
required common default

base Data Base name
required common default

output optional output file

/[NO] FULL Show the full information of all Areas of a Pool.
Together with the /DIRECTORY option all entries in the Pool Directory are listed.
default:'/NOFULL'

/[NO] DIRECTORY Show information of the Pool Directory
Together with the /FULL option all entries in the Pool Directory are listed. With the /DIRECTORY any given Pool name will be ignored.
default:'/NODIRECTORY'

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

/PRINT Print output

Caller	M$DMCMD
Author	M. Richter
File name	M$ASHPO.PPL
Dataset	-
EXAMPLE	SHO POO ADAM OUT='ADAM.LIS' show the Areas of the Pool 'ADAM' of the Data Base 'DB' and write the output into the file 'ADAM.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING STS=M$ASHPO(CV_POOL,CV_BASE,CV_OUT,
I_FULL,I_DIRECTORY,I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_POOL	I Pool name CHAR(*) VAR
CV_BASE	I Data Base name CHAR(*) VAR
CV_OUT	I optional file for output CHAR(*) VAR
I_FULL	Show full Pool and Area information BIN FIXED(15)
I_DIRECTORY	Show Pool Directory information BIN FIXED(15)
I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
I_PRINT	I Print output BIN FIXED(15)

FUNCTION Show Areas of a Data Base Pool. Optional full Area information will be shown. The Pool Directory can be shown by the option /DIRECTORY.

REMARKS Module is an action routine.

EXAMPLE -

SHOW SCATTER BUFFER

SHOW SCATTER BUFFER dyn_list dyn_dir base node

PURPOSE Show scatter buffer size

PARAMETERS

dyn_list Dynamic list name specification
common required default
If *, all attached lists are shown.

dyn_dir Default directory
common default:'$DYNAMIC'

base Default data base name
common default:'DB'

node Default node name
common default:'E'

EXAMPLE -

Caller M$DLCMD

Author H.G.Essel

File name M$ASHSB.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ASHSB(CV_DYN_LIST
,CV_DYN_DIR,CV_BASE,CV_NODE)

ARGUMENTS

CV_DYN_LIST Dynamic list name specification

CV_DYN_DIR Default directory

CV_BASE Default data base name

CV_NODE Default node name

FUNCTION Show buffer size of scatter buffer. The buffer size is specified as number of scatter points to be stored in the buffer. Note that there is at least one point per scatter frame. If there are bit variables, several points per frame might be sent. The buffer size must not be set below the maximum number of points for a picture.

REMARKS Module is an action routine.

EXAMPLE -

SHOW SPECTRUM

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

PURPOSE show spectra

PARAMETERS

name String default: '*'
name of spectrum. Wildcards are supported:
* x* *x *x* x*y Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

spec_dir String replace default: '$SPECTRUM'
Name of Spectrum directory.

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

node String replace default: '*'
Name of Node.

output String default: none
optional output file

width Integer global replace default: 80
Line length for histogram (80 or 132)

/PRINT Switch default: none
Print output

/WHAT	Set replace default: /ATTRIBUTES /ATTRIBUTES :Display attributes of spectra /DATA :Display data /STATUS :calls SHOW TAB/SP /ALL :Display all
/FULL	Switch default: none Display full spectrum information
/MEMBERS	Switch default: none Display information about all spectrum members.
/LOG	Switch default: none Display data in logarithmic format
/INTEGRAL	Switch default: none Display channel contents integral
/ZERO	Switch default: none Display channel contents including zero channels
/CAMAC	Switch default: none Display CAMAC spectra only
/[NO] KEEP_MAP	Switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
FUNCTION	The specified spectrum is accessed and various parts of it are displayed on the terminal
EXAMPLE	SHOW SPEC OTTO /FULL shows otto and all queued data elements
Caller	E$SHECM
Author	K.WINKELMANN
File name	E$SHSP.PPL
Dataset	GOO$DE:E$SHSP.PPL

Function

The specified spectrum is accessed and various
parts of it are displayed on the terminal

Example

SHOW SPEC OTTO

Remarks

REMARKS Module is an action routine.

Created by E$SHECM.PPL

Description

CALLING

STS=E$SHSP(CV_NAME,CV_DIR,CV_BASE,CV_NODE,CV_OUT,
L_WIDTH,I_PRINT,CV_DETAIL,I_FULL,
I_MEMBERS,I_LOG,I_INTEGRAL,I_ZERO,
I_CAMAC,I_KEEP_MAP,B_MASK)

COMMAND

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

NAME

Type Input CHAR(*) VAR
Name of spectrum. Wildcards are supported:
* x* *x *x* x*y Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

DIR

Type Input CHAR(*) VAR
Default directory

BASE

Type Input CHAR(*) VAR
Default data base

NODE

Type Input CHAR(*) VAR
Default node

OUTPUT

Type Input CHAR(*) VAR
Optional file for output

WITDH

Type Input BIN FIXED(31)
Line size for histogram (80 or 132).

PRINT

Type Input BIN FIXED(15)
valid values are:
No print of output file is done Print output file on line printer

WHAT

Type Input CHAR(*) VAR
valid values are:
Display attributes of the spectrum Display data of the spectrum Display attributes,all and status of the spectrum Display status of spectrum

FULL

Type Input BIN FIXED(15)
valid values are:
Display only short info about the spectrum: Spectrumname,spectrumtype,limits,bins,bits
Display full information about the spectrum: Spectrumname,spectrumtype,Database,Dire
Dimensions, Index in analyzetable, Creationdate Lettering of X-Axis,Lettering of Y-Axis, Lower limit, upper limit, Number of bins, Binsize, Factor, Offset, Underflow - and Overflow counts, Underflow and Overflow contents, freeze and executed bit

MEMBERS

Type Input BIN FIXED(15)
valid values are:
Display only first and last indexed spectrum: Display information about all spectrum members:

LOG

Type Input BIN FIXED(15)
Valid values are:
Display decimal data Display logarithmic data

INTEGRAL

Type Input BIN FIXED(15)
Valid values are:
Display channel contents Display integral channel contents

ZERO

Type Input BIN FIXED(15)
Valid values are:
Display channel contents without zero's Display channel contents with zero's

CAMAC

Type Input BIN FIXED(15)
Valid values are:
Display all spectra Display CAMAC spectra only

KEEP_MAP

Type Input BIN FIXED(15)
Valid values are:
Unmap Data Base Inhibit unmap of the Data Base

B_MASK

Type Output BIT(32) ALIGNED

Function

The specified spectrum is accessed and various
parts of it are displayed on the terminal.
Output can be interrupted by ^ Z.

Remarks

Module is an action routine.

Example

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 TABLE

SHOW TABLE name table tab_dir base node output
/CONDITION /SPECTRUM /ALL
/CONTENT /COUNTS
/PRINT
/[NO]KEEP_MAP

PURPOSE show flag tables from the analysis

PARAMETERS

NAME String replace default: '*'
Name of items (conditions or spectra)

TABLE String replace default: '*'
Name of table (may be omitted)

TAB_DIR String replace default: '$ANLTABS'
Name of Directory

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

NODE String replace default: '*'
Name of node

OUTPUT String replace default: ''
Optional output file

TYPE Set replace default: /ALL /CONDITION : /SPECTRUM : /ALL : Kind of table

CONT Set replace default: /COUNTS /CONTENT : /COUNTS : Display spectrum content or counts

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

/PRINT Switch default: none
Print output

FUNCTION The contents of the tables (flags for freeze and executed (spectra) or freeze,executed,result and result preset (conditions) are shown.
In addition, condition counters and spectrum contents and overflows are displayed

EXAMPLE SHO TA /SP

Caller E$SHECM

Author K.Winkelmann

File name GOO$DE:E$SHANT.PPL

Dataset -

NAME

Type String replace default: '*'
Name spec of conditions or spectra (wildcard)

TABLE

Type String replace default: '*'
Name of analysis table, NYI,
in the moment default name ANSP and ANCO from directory $ANLTABS are taken

TAB_DIR

Type String replace default: '$ANLTABS'
Default name of Directory

BASE

Type String replace default: 'DB'
Default name of Base

NODE

Type String replace default: '*'
Default name of Node

OUTPUT

Type String replace default: "
Optional file for output

TYPE

Type Set replace default: /ALL
Valid values are:

'/COND' ANCO is shown

'/SPECTRUM' ANSP is show

'/ALL' indicates that both will be shown.

CONT

Type Set replace default: /COUNTS
Valid values are:

'/CONTENT'

'/COUNTS'
Spectrum contents

KEEP_MAP

Type Switch replace default: /KEEP_MAP
Valid values are:

[NO] KEEP_MAP Inhibit unmap of the Data Base.

PRINT

Type Switch default: none
Valid values:

'/PRINT'1 print output file . Print output file on lineprinter.

MASK

Type Input BIT(32) ALIGNED
Mask which params were specified

Function

The contents of the tables (flags for freeze and executed (spectra) or freeze,executed,result and result preset (conditions) are shown.
In addition, condition counters and spectrum
contents and overflows are displayed

Example

SHO TA /SP

Remarks

REMARKS Module is an action routine.

Created by GOO$DE:E$SHECM.PPl

Description

Argument description

SPEC

Type Input CHAR(*) VAR
Name spec of conditions or spectra (wildcard)

NAME

Type Input CHAR(*) VAR
Name of analysis table, NYI,
in the moment default name ANSP and ANCO from directory $ANLTABS are taken

DIR

Type Input CHAR(*) VAR
Default name of Directory

BASE

Type Input CHAR(*) VAR
Default name of Base

NODE

Type	Input CHAR(*) VAR Default name of Node

OUT

Type	Input CHAR(*) VAR Optional file for output

KIND

Type	Input CHAR(*) VAR
'/COND'	ANCO is shown
'/SPECTR'	ANSP is shown
'/ALL'	indicates that both will be shown.

CONT

Type	Input CHAR(*) VAR Valid values are:
'/CONTENT'	
'/COUNTS'	
	Spectrum contents

KEEP_MAP

Type	Input BIN FIXED(15) Valid values are:
0	Deattach the data base.
1	Inhibit unmap of the Data Base.

PRINT

Type	Input BIN FIXED(15) Valid values:
0	no print of output file is done.
1	print output file . Print output file on lineprinter.

MASK

Type Input BIT(32) ALIGNED
Mask which params were specified

Function

The contents of the tables (flags for freeze and executed (spectra) or freeze,executed,result and result preset (conditions) are shown.
In addition, condition counters and spectrum
contents and overflows are displayed

Remarks

Module is an action routine.

Example

—

SHOW TP0 KEYPAD

SHOW TP0 KEYPAD

PURPOSE	Display of GOOSY auxiliary keypad definition active in GOOSY prompter MGOOTP0 May be called by PF2 key.
EXAMPLE	SHO TP0 KEY
Action rout.	U$TCSHK
Author	H.G.Essel

Remarks

File name	U$TCSHK.PPL

Description

CALLING	STS=U$TCSHK
FUNCTION	Pressing the PF2 key on the auxiliary keypad this procedure is activated and displays the keypad set-up defined by the command procedure INI_TP0.
REMARKS	-
EXAMPLE	-

SHOW TREE

SHOW TREE base output
/[NO]KEEP_MAP
/PRINT

PURPOSE	Show indices of a Data Base Pool Directory For test purpose only.
PARAMETERS	
base	Data Base name required common default
output	optional output file optional
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
/PRINT	Print output
Caller	M$DMCMD
Author	M. Richter
File name	M$ASHTR.PPL
Dataset	-
EXAMPLE	SHO TREE DB show the name tree of the Pool Directory of the Data Base 'DB'.

Remarks

REMARKS	-

Description

CALLING STS=M$ASHTR(CV_BASE,CV_OUT,I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_BASE I Data Base name
CHAR(*) VAR

CV_OUT I optional file for output
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

I_PRINT I Print output
BIN FIXED(15)

FUNCTION Show indices of a Data Base Pool Directory

REMARKS Module is an action routine.

EXAMPLE -

SHOW TYPE

SHOW TYPE type base output

PURPOSE Show a Type descriptor. Wild card are allowed.

PARAMETERS

type required string default: " type descriptor name

base required string global replace default: " Data Base name

output string default: " optional output file

/DATA switch default: /PLI Output of Type descriptor as a PL-I Structure or only the data.

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

/PRINT switch default: Print output on Line Printer.

Caller M$SHMCM

Author M. Richter

Example

SHO TYP $SPECTRUM DB OUT='SPEC.LIS'
show the Data Element Type '$SPECTRUM' of the Data
Base 'DB' and write the output into the file 'SPEC.LIS' of the default VAX/VMS directory.

Remarks

File name M$ASHTY.PPL

Created by GOO$DM:M$SHMCM.PPL

Description

CALLING

STS = M$ASHTY(CV_TYPE,CV_BASE,CV_OUT,CV_DATATYPE,
I_KEEP_MAP,I_PRINT)

COMMAND

SHOW TYPE type base output
/DATA/PLISTRUC
/[NO]KEEP_MAP
/PRINT
Argument / Parameter description

TYPE

Routine arg. Input CHAR(*) VAR

Command par. required string default :"
Type descriptor name. Wild cards are allowed

BASE

Routine arg. Input CHAR(*) VAR

Command par. required string default :"
Data Base name

OUTPUT

Routine arg. Input CHAR(*) VAR

Command par. required string default :"
Optional file for output

/DATA

Routine arg. Input CHAR(*) VAR

Command par. required string default :"
Output of Type descr. like a Data Element or a
PL/I-Structure

/KEEP_MAP

Routine arg. Input CHAR(*) VAR

Command par. required string default :"
Inhibit unmap of Data Base

/PRINT

Routine arg. Input CHAR(*) VAR

Command par. required string default :"
Print output

Function

Show a Type descriptor in directory $TYPE in
Data Base 'Base'.

Remarks

Module is an action routine.

Example

```
STS=M$ASHTY('*SPD*','DB',",'/PLI',1,0)
```

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

SLEEP

SLEEP

PURPOSE	Put the prompter in a HIBERNATE state.
FUNCTION	This procedure puts the command prompter (MGOOTP0) in a HIBERNATE state until a ^ C is entered.
EXAMPLE	SLEEP
Action rout.	U$TCHIB
Author	Walter F.J. Mueller

Remarks

File name	U$TCHIB.PPL

Description

CALLING	@CALL U$TCHIB()
ARGUMENTS	-
FUNCTION	This procedure puts the command prompter (MGOOTP0) in a HIBERNATE state until a ^ C is entered.
REMARKS	-
EXAMPLE	@CALL U$TCHIB();

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 ANALYSIS OUTPUT

START ANALYSIS OUTPUT file size buffersize
device directory
type subtype stream
headerinput headeroutput
/PROMPT
/EDIT
/[NO]OPEN
/[NO]SYNCHRON
/COPY/COMPRESS/INPUT
/MBD/J11
/BYTE/KBYTE/PAGE/BUFFER

PURPOSE

Start data output from analysis. Output is done to DECnet. If a file is specified, output is written to file too.

PARAMETERS

file
string
File specification, disk or tape.
Keep in mind filename comventions for IBM.

size
integer default=35000
maximal file size in Kbytes.

buffersize
integer default=16384
Buffer size in bytes.

device
string replace
Default device

directory
string replace
Default directory

type
integer replace
Type for output buffer (default=4)
Ignored if /MBD or /J11 is specified.

subtype	integer replace Subtype for output buffer (default=1) Ignored if /MBD or /J11 is specified.
stream	integer replace Maximum numbers of buffers containing event fragments
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).
/OPEN	Open new output file (=default)
/NOOPEN	Continue writing to old file. Only possible if files output has been stopped by STOP OUT FILE /NOCLOSE
/SYNCHRON	Set synchron mode. Wait for output to DECnet, if a DECnet channel has been opened by another anaysis.
/MBD	Use buffer type/subtype like MBD buffers.
/J11	Use buffer type/subtype like J11 buffers. In both cases the type subtype parameters described above are ignored.
/COPY	Use X$PAEVT for copying output event into output buffer (=default).
/INPUT	Use X$PAEVT for copying event into output buffer. The difference to /COPY is that the original event from input buffer is copied. No output data element is needed or used.
/COMPRESS	Use X$PACMP for packing Output event into output buffer.
/PAGE	file size in pages
/BYTE	file size in bytes
/KBYTE	file size in Kbyte
/BUFFER	file size in buffers

EXAMPLE

START ANAL OUT DAY$ROOT:[SCRATCH]F1.LMD
copies output data element to output buffer with
type/subtype 4,1.
SET ANAL/EVENT
STA ANA OUT X /MBD/INPUT
copy MBD events directly from input buffer to
output buffer. Event unpacking must be enabled!
STA AN OUT X /COMP
copy compressed output data element to buffer.
In the buffer the event wil have a standard type.
When these events are read by analysis, they are
decompressed and copied into the data element.

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

Description

FUNCTION

This procedure starts data output from analysis. Output is done to DECnet. If a file is specified, output is written to file too. The AMR checks if another analysis program has opened a link. In this case the output buffer is written to this DECnet channel. If /SYNCHRON is set, it waits for the acknowledge. Otherwise buffers are written to the DECnet channel only if the receiver acknowledged the previous buffer.
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$ANACM.PPL

Action rout. I$ANACM_STA_OUT

Version 1.01

Author H.G.Essel

Last Update 12-APR-1985

START ANALYSIS RANDOM

START ANALYSIS RANDOM bufevents events

PURPOSE Start analysis for Monte Carlo.

PARAMETERS

bufevents integer default=100
Number of events per virtual buffer to process.

events integer default=0
Number of events to process (0 = infinite)

EXAMPLE START ANAL RAN 100

Description

FUNCTION This command starts the analysis without input. No event is copied into the data base. The user analysis routine must generate its own raw data. The second argument specifies, how many events (Calls of X$ANAL) are executed before a command can be executed. Note, that X$ANAL may return status XIO_STOPINPUT to stop the analysis.

File name I$ANACM.PPL

Action rout. I$ANACM_STA_RAN

Version 1.01

Author H.G.Essel

Last Update 12-APR-1985

START BUFFER DUMP

START BUFFER DUMP file buffers size
/ANALYSIS
/KBYTE

PURPOSE Open file and write buffers to LMD file.

PARAMETERS

file LMD file name

buffers Number of buffers to write.

size Buffersize in bytes. The size is defaulted from buffer size of last active input channel. (file, mailbox or net). The buffer size of net input can be obtained only by first received buffer.

/ANALYSIS Call I$BUFFER. By default, I$BUFFER is not called as long as buffer dump is active (performance). When /ANALYSIS is specified, each incoming buffer is written to tape and analyzed.

/KBYTE Specify size in Kbytes instead of bytes.

EXAMPLE
```
STA BUF DUMP TEST.LMD 100
  SHOW BUFFER DUMP
  STOP BUFFER DUMP
```

NOTE Note that no other analysis output must be active

Description

FUNCTION Opens output file to write input buffers to file. No analysis output must be active.

File name I$ANACM.PPL

Action rout. I$ANACM_LMD_START

Version 1.01

Author H.G.Essel

Last Update 12-Mar-1993

START DYNAMIC LIST

START DYNAMIC LIST
dyn_list dyn_type dyn_dir base node

PURPOSE Start execution of dynamic list

PARAMETERS

dyn_list Dynamic list name specification
required
common default
If *, all attached lists are modified.

dyn_type Type of dynamic sublist or *
Empty : Start main list
* : Start all sublists
type : Start sublist "type"
default:"

dyn_dir Default directory
default:'$DYNAMIC'
common default

base Default data base name
default:'DB'
common default

node Default node name
default:'E'
common default

EXAMPLE -

Caller M$DLCMD

Author H.G.Essel

File name M$ASTDL.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ASTDL(CV_DYN_LIST,CV_TYPE
,CV_DYN_DIR,CV_BASE,CV_NODE)

ARGUMENTS

CV_DYN_LIST Dynamic list name specification

CV_TYPE Type of sublist
Empty : Start main list
* : Start all sublists
type : Start sublist "type"

CV_DYN_DIR Default directory

CV_BASE Default data base name

CV_NODE Default node name

FUNCTION Start execution of dynamic list

REMARKS Module is an action routine.

EXAMPLE -

START INPUT FILE

START INPUT FILE file buffers events
skip_buffer skip_event
device directory
/CLEAR
/OPEN
/FOREIGN
/[NO]HEADER

PURPOSE Start data analysis from file at current position. Open it if it was not open.

PARAMETERS

file required string replace
File specification, disk or tape

buffers integer default=0
Number of buffers to process (0 = infinite)
When this number of buffers is processed, input stops, but the file remains open. The next START INPUT FILE command continues. Skipped buffers are not counted.

events integer default=0
Number of events to process (0 = infinite)
When this number of events is processed, input stops, but the file remains open. The next START INPUT FILE command continues. Events skipped by command are not counted.

skip_buffer integer default=0
Number of buffers (records) to skip

skip_event integer default=0
Number of events to skip

device string replace
Default device

directory string replace
Default directory

/CLEAR	Clear counters, even if file is continued.
/OPEN	Open new file. Close it first, if it was open.
/FOREIGN	Foreign buffer format. I$buffer calls unpack routine X$UPFOR
/HEADER	File has GOOSY header (=default).
/NOHEADER	File has no GOOSY header. If the file has no header, but /HEADER is specified, the file must be closed after reading the first buffer and opened again. This can be time consuming on tapes.
EXAMPLE	START INP FILE DAY$ROOT:[SCRATCH]F1.LMD START INP FILE M1:F1.LMD 1000
NOTE	It is recommended to skip/process either events or buffers, because events in skipped buffers are not counted. When processing a number of events, there are two buffers pending after stop. The one with the last event, and the next one. These two buffers are processed first with next START INPUT FILE command, except /OPEN was given. Skipping two buffers these two buffers are skipped. Otherwise processing starts with the next event in the first pending buffer.

Description

FUNCTION	This procedure starts data taking from file at current position. If no file is open, it will be opened in any case. If a file is open, it will be closed and reopened, if /OPEN is specified, but will remain open at the same position, if /OPEN is NOT specified. When an optional buffer/event limit is reached, file input stops, but file remains open.
File name	I$ANACM.PPL
Action rout.	I$ANACM_STA_FIL
Version	1.01
Author	H.G.Essel
Last Update	12-APR-1985

START INPUT MAILBOX

START INPUT MAILBOX mbx_name mbx_number
buffers events bufevents
skip_buffers size

PURPOSE Open input stream from mailbox

PARAMETERS

mbx_name string replace
name of mailbox GOOSY_name_# (def.=environment)

mbx_number integer replace default=1
Number of mailbox GOOSY_name_# (1,2,3) (def.=1)

buffers integer default=0
Number of buffers to process (0 = infinite)

events integer default=0
Number of events to process (0 = infinite)

bufevents integer default=0
Number of events per buffer to process (0=infinite)

skip_buffers integer default=0
Number of buffers to be skipped

size integer replace default=8192
Buffersize in bytes. This size must match the size
as specified in INI ACQUIS, i.e. 8192 for MBD (=default) and 8192 for J11 single crate system.

EXAMPLE START INPUT MAILBOX SUSI SIZE=8192

Description

FUNCTION This procedure opens a mailbox to read data from TMR. The name of the mailbox is GOOSY_name_n. The mailbox must exist (is created by TMR).

File name	I$ANACM.PPL
Action rout.	I$ANACM_OP_MBX
Version	1.01
Author	H.G.Essel
Last Update	12-APR-1985

START INPUT NET

START INPUT NET node environment component
buffers events
/TMR/ANL
/MULTI

PURPOSE Open input stream from network

PARAMETERS

node required string replace
Name of node, where partner runs

Environment required string replace
Name of the environment of the partner

Component string default=$TMR
Name of the component of the partner (default=$TMR)

buffers integer default=0
Number of buffers to process (0 = infinite)

events integer default=0
Number of events to process (0 = infinite)

/TMR Net input from transport manager (=default)

/ANL Net input from analysis

/MULTI Allow multiple input links

EXAMPLE
```
START INPUT NET B TEST $TMR
START INPUT NET B TEST $ANL /ANL
```

Description

FUNCTION This procedure opens a link to TMR or ANL processes to get data.

File name I$ANACM.PPL

Action rout. I$ANACM_OP_NET

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

START MR2000

START MR2000 branch crate station
/INITIALIZE

PURPOSE Start/initialize MR2000.

PARAMETERS

branch Integer replace default: 0
Number of CAMAC branch.

crate Integer replace default: 1
Number of crate on branch.

station Integer replace default: 2
Station number of MR2000 in crate.

/INITIALIZE switch default: ”
Initialize MR2000 before start.

Caller mdbm

Author H.G.Essel

Example

```
STA MR2000 0,3,20 /INIT
```

Remarks

File name E$AGOMR.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$AGOMR(I_branch,I_crate,I_station, I_initialize)

COMMAND START MR2000 branch crate station /INITIALIZE
Argument description

BRANCH

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 0
Number of CAMAC branch.

CRATE

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 1
CAMAC Crate number.

STATION

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 2
Station number of MR2000 in crate

INITIALIZE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”
Initialize MR2000 before start.

Function

Module I$CMR2G is called to start the MR2000.
If /INITIALIZE is specified, I$CMR2I is called before.

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 SCATTER

START SCATTER
/SYNCHRONOUS /ASYNCHRONOUS [=/MODE]

PURPOSE Start scatter plots for actual picture

PARAMETERS

MODE Transfer mode of scatter data.

/SYNCHRONOUS The analysis waits for request from the display process.

/ASYNCHRONOUS The analysis does not wait for a request from the display process.

Caller MDISP, MGOODISP

Author W. Spreng

Remarks

Created by D$DSPCM.PPL

File name D$GO.PPL

Description

CALLING STS=D$GO(CV_mode,B_mask)

COMMAND START SCATTER
/SYNCHRONOUS/ASYNCHRONOUS [= MODE]

/MODE

Routine arg. Input CHAR(*) VAR

Command par. Switch default = /SYNCHRONOUS

Determines the transfer mode of the scatter data from the analysis to the display process.

/SYNCHRONOUS The analysis waits for request from display process and continues if the scatter data are accepted by the display. This modes slows down the analysis.

/ASYNCHRONOUS Analysis continues when a buffer has been sent to the display process. The next buffer will be sent after the request from the display process. In that mode the scatter plot does not influence the analysis.

FUNCTION

To start the scatter plots in the picture which is actually displayed on the screen. The freeze bit in the picture header has to be resetted.

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 ANALYSIS OUTPUT

STOP ANALYSIS OUTPUT /[NO]CLOSE

PURPOSE	Stop data output from analysis
PARAMETERS	
/CLOSE	Close output file (=default).
EXAMPLE	STOP ANAL OUT

Description

FUNCTION	This procedure stops data output from analysis.
File name	I$ANACM.PPL
Action rout.	I$ANACM_STO_OUT
Version	1.01
Author	H.G.Essel
Last Update	12-APR-1985

STOP ANALYSIS RANDOM

STOP ANALYSIS RANDOM

PURPOSE	Close input stream from mailbox
PARAMETERS	
EXAMPLE	STOP ANAL RAN

Description

FUNCTION	This procedure stops the analysis.
File name	I$ANACM.PPL
Action rout.	I$ANACM_STO_RAN
Version	1.01
Author	H.G.Essel
Last Update	12-APR-1985

STOP BUFFER DUMP

STOP BUFFER DUMP

PURPOSE	Stop writing to LMD file. Close File
PARAMETERS	
EXAMPLE	STO BUFFER DUMP

Description

FUNCTION	Stop writing LMD file. Close file.
File name	I$ANACM.PPL
Action rout.	I$ANACM_LMD_STOP
Version	1.01
Author	H.G.Essel
Last Update	12-Mar-1993

STOP DYNAMIC LIST

STOP DYNAMIC LIST dyn_list dyn_type dyn_dir base node

PURPOSE Stop execution of dynamic list

PARAMETERS

dyn_list Dynamic list name specification
If *, all attached lists are modified.
common required default

dyn_type Type of dynamic sublist or *
Empty : Start main list
* : Start all sublists
type : Start sublist "type"
default:"

dyn_dir Default directory
common default:'$DYNAMIC'

base Default data base name
common default:'DB'

node Default node name
common default:'E'

EXAMPLE -

Caller M$DLCMD

Author H.G.Essel

File name M$ASPDL.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ASPDL(CV_DYN_LIST,CV_TYPE,CV_DYN_DIR, CV_BASE,CV_NODE)

ARGUMENTS

CV_DYN_LIST Dynamic list name specification

CV_TYPE Type of sublist
Empty : Start main list
* : Start all sublists
type : Start sublist "type"

CV_DYN_DIR Default directory

CV_BASE Default data base name

CV_NODE Default node name

FUNCTION Stop execution of dynamic list

REMARKS Module is an action routine.

EXAMPLE -

STOP INPUT FILE

STOP INPUT FILE
/CLOSE

PURPOSE Stop reading input file, optional close.

PARAMETERS

/CLOSE close file. File will be opened again by start with START INP FILE. Otherwise keep file open. Continue with START INP FIL at the same position.

EXAMPLE STOP INPUT FILE

Description

FUNCTION This procedure stops reading input file. The input stream may be resumed by START INPUT FILE command at the same position, if stopped without /CLOSE

File name I$ANACM.PPL

Action rout. I$ANACM_STO_FIL

Version 1.01

Author H.G.Essel

Last Update 12-APR-1985

STOP INPUT MAILBOX

STOP INPUT MAILBOX mbx_num

PURPOSE	Close input stream from mailbox
PARAMETERS	
mbx_num	integer replace default=1 Number of mailbox (1,2,3). The mailbox' name is GOOSY_name_n, n=1,2,3
EXAMPLE	STOP INPUT MAIL 1

Description

FUNCTION	This procedure closes a mailbox to read data from TMR.
File name	I$ANACM.PPL
Action rout.	I$ANACM_CLO_MBX
Version	1.01
Author	H.G.Essel
Last Update	12-APR-1985

STOP INPUT NET

STOP INPUT NET

PURPOSE	Close input stream from DECnet
EXAMPLE	STOP INPUT NET

Description

FUNCTION	This procedure closes a link to read data from TMR.
File name	I$ANACM.PPL
Action rout.	I$ANACM_CLO_NET
Version	1.01
Author	H.G.Essel
Last Update	12-APR-1985

STOP MR2000

STOP MR2000 branch crate station

PURPOSE Stop MR2000.

PARAMETERS

branch Integer replace default: 0
Number of CAMAC branch.

crate Integer replace default: 1
Number of crate on branch.

station Integer replace default: 2
Station number of MR2000 in crate.

Caller mdbm

Author H.G.Essel

Example

```
STOP MR2000 0,3,20
```

Remarks

File name E$AHAMR.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$AHAMR(I_branch,I_crate,I_station)

COMMAND STOP MR2000 branch crate station Argument description

BRANCH

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 0
Number of CAMAC branch.

CRATE

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 1
CAMAC Crate number.

STATION

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 2
Station number of MR2000 in crate

Function

Module I$CMR2H is called to stop the MR2000.

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 SCATTER

STOP SCATTER

PURPOSE	Stop scatter plots for actual picture
Caller	MDISP, MGOODISP
Author	W. Spreng

Remarks

Created by	D$DSPCM.PPL
Dataset	D$STOP.ppl

Description

CALLING	STS=D$STOP()
COMMAND	STOP SCATTER

Function

To stop the scater plots in the picture which is actually displayed on the screen. The freeze bit in the picture header has to be set.

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');

SUMUP SPECTRUM

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

PURPOSE Integrate specified window

PARAMETERS

spectrum name of spectrum

window Limits of integration window

file File for the output of the results.

CALIBR Specifies the unit in which the coordinates are given.

/CHAN	Original spectrum units.
/CALIB	Calibrated units.

/[NO] OUTPUT Output occus additionally onto the specified file.

/[NO] APPEND Results are appended to an existing file.

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

Caller MDBM,MGOODBM

Author W. Spreng

Example

1.) SUMUP SPECTRUM a 100,400
Spectrum A is integrated from channel 100 to 400
2.) SUMUP SPECTRUM a 100,400/calib

Spectrum A is integrated from the calibrated value 100 to 400

3.) SUMUP SPECTRUM a 100,400,200,500

Window 100,400 and 200,500 are integrated. 3.) SUMUP SPECTRUM a 100,400,200,500 /OUTPUT

Output is directed onto the file GOOSY_RESULTS.

Remarks

File name E$SUMUP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$SUMSP(CV_spectrum,CV_window,CV_file, CV_spec_dir, CV_base, CV_node, CV_calibr, I_output, I_append, I_keep_map)

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

SPECTRUM

Routine arg. CHAR(*) VAR

Command arg. String required

Name of spectrum which should be summed up.

WINDOW

Routine arg. Input CHAR(*) VAR

Command par. String

Window which should be used as integration range. Any valid display window specification is allowed (see the GOOSY Display Manual):
(xmin,xmax) for a one dimensional window
* for the whole spectrum range
It is possible to define up to 30 windows which should be integrated by:
(xmin,xmax,xmin,xmax,...)

If values are omitted they are replaced by the corresponding spectrum minimum or maximum limit!

The upper window limits are exclusive! E.g. for a one dimensional analog spectrum of binsize 1 the window (1,3) yields to an integration of two bins! Although value 3 belongs to the third bin.

FILE

Routine arg. CHAR(*) VAR

Command arg. String replacable default=GOOSY_RESULT.LOG

Name of a file onto which the results should be written. To direct the outputs to that file the /OUTPUT switch is required!

SPEC_DIR

Routine arg. CHAR(*) VAR

Command arg. String global replaceable default=$SPECTRUM

Default Directory name for spectra.

BASE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=*

Node name for Data Base section file.

CALIBR

Routine arg. Input CHAR(*) VAR

Command par. Set default="/CHAN"

If the spectrum in the selected frame is calibrated the input of the coordinates can occur in calibrated or uncalibrated units:

/CHANN	Coordinates of point are given in spectrum units.
/CALIB	Coordinates of point are in calibrated units

If the coordinates are specified via cursor input the units are known and this switch has no effect!

OUTPUT

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch, default = /NOOUTPUT

Write results into the specified file.

/OUTPUT	The output is directed onto the file.
/NOOUTPUT	results are only written to terminal and into the session logfile.

APPEND

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch, default=/APPEND

Append the output to an existing file.

/APPEND	output is appended to an existing file.
/NOAPPEND	A new output file is created.

KEEP_MAP

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch

Inhibit the unmap of the Data Base.

/KEEP_MAP	Data Base will not be dettached after command termination.
/NOKEEP_MAP	Dettach the Data Base in any case.

FUNCTION

The spectrum in the specified frame is integrated within the specified limits. The window could be specified in calibrated units or in channels, depending on the given switch (/CHAN or /CALIB). If calibrated units are used but the spectrum in the specified frame is not connected to a calibration an error is signaled.

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

TYPE FILE

TYPE FILE file skip buffers events id outfile
/HEADER /DATA
/EVENTHEADER
/SAMPLE
/PRINT

PURPOSE Output GOOSY list mode data file (called in MUTIL).

PARAMETERS

file required string replace
List mode data file.

skip integer default=0
Optional number of buffers to skip.

buffers integer default=1
Optional number of buffers to output

events integer default=10
Optional number of events to output

id integer default= 0
Optional number of events to output

output string replace
Optional output file.

/HEADER switch
Output GOOSY file header only.

/DATA switch
Output formatted data.

/SAMPLE switch
Output one event per buffer. Valid for /DATA.

/EVENTHEADER switch
Output event header only. Valid for /DATA.

/PRINT switch
Print output file. Valid for /DATA.

Caller MUTIL

Author H.G.Essel

Example

```
$ MUTIL TYPE FIL X.LMD OUT=X.HEAD /HEAD
file header is written into X.HEAD.
$ MUTIL TYPE FIL X.LMD 10 1 Y.LIS /DATA
Write 11th buffer of X.LMD to Y.LIS in ASCII.
```

Remarks

File name I$FILCM.PPL

Created by I$FILCM.PPL

Description

CALLING
```
STS=I$FIL_T(CV_file,L_skip,L_buffers,L_events,L_id,
            CV_outfile,CV_output,
            I_sample,I_header,I_print)
```

COMMAND
```
TYPE FILE file skip buffers events id outfile
            /HEADER /DATA
            /EVENTHEADER
            /SAMPLE
            /PRINT
   Arguments/Parameters description
```

FILE

Routine arg. Input CHAR(*) VAR

Command par. required string replace
File name for input.

SKIP

Routine arg. Input BIN FIXED(31)

Command par. integer default=0
Optional number of buffers to be skipped.

BUFFERS

Routine arg. Input BIN FIXED(31)

Command par. integer default=1
Optional number of buffers to be output.

EVENTS

Routine arg. Input BIN FIXED(31)

Command par. integer default=10
Optional number of events to be output.

ID

Routine arg. Input BIN FIXED(31)

Command par. integer default=0
Optional number of subevent to be output (FEP id).

OUTFILE

Routine arg. Input CHAR(*) VAR

Command par. string replace
Optional file name for output.

/OUTPUT

Routine arg. Input CHAR(*) VAR

Command par. switch default=/DATA

/HEADER	Output header only.
/DATA	Output formatted data

/SAMPLE

Routine arg. Input BIN FIXED(15) valid values 0,1

Command par. switch

Output one event per buffer

/EVENTHEADER

Routine arg. Input BIN FIXED(15) valid values 0,1

Command par. switch

Output event header only

/PRINT

Routine arg. Input BIN FIXED(15) valid values 0,1

Command par. switch

Print output file

Function

Read specified input file and output GOOSY file header and data. Only standard GOOSY data formats are supported.

UNFREEZE CONDITION

UNFREEZE CONDITION name cond_dir base node
/[NO]KEEP_MAP

PURPOSE unfreeze a condition, enable the execution of a condition

PARAMETERS

name name of condition
required

cond_dir default Directory
replace default:'$COND'

base default Data Base
replace default:'DB'

node default node
replace default:'E'

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

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$UNFCO.PPL

Dataset -

Remarks

REMARKS action routine

Description

CALLING STS=E$UNFCO(CV_NAME,CV_DIR,CV_BASE, CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_NAME default directory
CHAR(*) VAR
Input

CV_DIR default directory
CHAR(*) VAR
Input

CV_BASE default base
CHAR(*) VAR
Input

CV_NODE default node
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

FUNCTION The freeze bit in the analysis flag tables ([$ANLTABS]ANCO is the default) in which the specified condition lies, is reset. This enables any subsequent execution of the condition by the analysis program or any dynamic list. The result bit and the preset bit are set to zero.

REMARKS action routine

EXAMPLE -

UNFREEZE SPECTRUM

UNFREEZE SPECTRUM name spec_dir base node
/[NO]KEEP_MAP

PURPOSE unfreeze a spectrum, enable the accumulation

PARAMETERS

name name of spectrum
required

spec_dir default Directory
replace default:'$COND'

base default Data Base
replace default:'DB'

node default node
replace default:'E'

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

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$UNFSP.PPL

Dataset -

Remarks

REMARKS action routine

Description

CALLING STS=E$UNFSP(CV_NAME,CV_DIR,CV_BASE,
CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_NAME default spectrum
CHAR(*) VAR
Input

CV_DIR default directory
CHAR(*) VAR
Input

CV_BASE default base
CHAR(*) VAR
Input

CV_NODE default node
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

FUNCTION The freeze bit in the analysis flag tables ([$ANLTABS]ANCO is the default) in which the specified spectrum lies, is reset. This enables any subsequent accumulation (execution) of the spectrum by the analysis program or any dynamic list.

REMARKS action routine

EXAMPLE -

UNPROTECT SPECTRUM

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

PURPOSE Unprotect one (or all) spectrum

PARAMETERS

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

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

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

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

/LOG Switch default:
Output names of unprotected spectra.

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

Caller mdbm

Author H.G.Essel

Example

UNPROT SP A(3,9) unprotect one spectrum
UNPROT SP A(3) unprotect one spectrum
UNPROT SP A(3:9) unprotect seven spectra
UNPROT SP A(*) unprotect all members
UNPROT SP A* unprotect all members
UNPROT SP A unprotect all members

Remarks

File name E$UPROSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$UPROSP(CV_NAME,CV_SPEC_DIR,CV_BASE,CV_NODE,
I_LOG,I_KEEP_MAP)

COMMAND UNPROTECT SPECTRUM name spec_dir base node
/LOG
/[NO]KEEP_MAP
Argument description

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of Spectrum (wildcard) to be unprotected
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.
Arrays without index are unprotected totally.
For one dimensional arrays a range may be
specified like x(3:5). No wildcard is allowed in
this case. Single members of one and twodimensional
arrays may be unprotected by specifying the index:
X(7) or Y(4,5).

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Name of Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Name of Node

LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Output unprotected spectrum names.

KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Inhibit the unmap (detach) of the Data Base

Function

The spectrum will be cleared by CLEAR command.

UPDATE BASE

UPDATE BASE base

PURPOSE Update a Data Base, write all pages to Section File

PARAMETERS

BASE Name of the Data Base
common required default

EXAMPLE UPD DA DB

Caller M$DMCMD

Author M. Richter

File name M$UPDB.PPL

Dataset -

Remarks

REMARKS All pages of the Data Base will be written back into its Section File, even if other processes access the same Data Base at the same time.

Description

CALLING STS=M$UPDB(cv_db_name)

ARGUMENTS

cv_db_name I Name of the Data Base to be updated
CHARACTER(*) VAR
Input

FUNCTION The whole Data Base 'cv_db_name' will be mapped and then written back into the Section File. This update will be forced by the procedure even if other processes are using the Data Base at the same time.

REMARKS Module is an action routine.
The Data Base must be mounted before the call.

EXAMPLE -

UPDATE DYNAMIC LIST

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

PURPOSE Update a Dynamic List

PARAMETERS

dyn_list Dynamic List name specification
required common default

dyn_dir Default Directory
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

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

Caller M$DMCMD

Author H.G.Essel

File name M$AUPDL.PPL

Dataset -

EXAMPLE UPDAT DYN LI DYNA_1 $DYNAMIC
update the Dynamic List 'DYNA_1' of the Directory '$DYNAMIC' in the Data Base last used.

Remarks

REMARKS -

Description

CALLING STS=M$AUPDL(CV_dyn_list,CV_dyn_dir,CV_base,CV_node, I_keep_map)

ARGUMENTS

CV_dyn_list I Dynamic List name specification
CHAR (*) VAR

CV_dyn_dir I Default Directory
CHAR (*) VAR

CV_base I Default Data Base name
CHAR (*) VAR

CV_node I Default node name
CHAR (*) VAR

I_keep_map I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Update a Dynamic List

REMARKS Module is an action routine.

EXAMPLE -

UPDATE FRAMES

UPDATE FRAMES frames seconds
/REFRESH

PURPOSE	Update frames in the actual picture on screen
PARAMETERS	
frames	Array of frame numbers, e.g: 1,5,3,8
seconds	time difference betwen two updates
/REFRESH	refresh the spectrum data.
Caller	MDISP, MGOODISP
Author	W. Spreng

Example

UPDATE FRAMES 1,4,3,8,5 10
The contents of the spectra in frame 1,3,4,5,8 are displayed every 10-seconds.

Remarks

File name	D$DUPD.PPL
Created by	GOO$DISP:D$DSPCM.PPL

Description

CALLING	STS=D$DUPD(LA_frames,l_time,I_refresh,La_array, B_mask)
COMMAND	UPDATE FRAMES frames seconds /REFRESH

FRAMES

Routine arg. (*) BIN FIXED(31)

Command par. Integer array required

List of frames for which an update should be performed.

SECONDS

Routine arg. Input BIN FIXED(31)

Command par. Integer required

Time in seconds between two updates.

REFRESH

Routine arg. Input BIN FIXED(15)

Command par. Switch

If set the specified frames are refreshed. That means the old spectrum is deleted and the new data are drawn into the existing frame.

Function

The spectrum contents in the current active picture are updated every 'seconds'. Specified Scatterframes or two dimensional spectra are ignored. The new spectrum contents are overlayed into the existing frames.

WAIT

WAIT seconds

PURPOSE	Wait n seconds.
PARAMETERS	
seconds	integer Number of seconds to wait
EXAMPLE	GOOSY> WAIT 10
Action rout.	U$TCWAIT
Author	H.G.Essel

Remarks

File name	U$TCWAIT.PPL
Defined in	U$TPRCM.PPL
REMARKS	-

Description

Wait some seconds.

WRITE CAMAC SPECTRUM

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

PURPOSE Write spectrum data from GOOSY spectrum into MR2000

PARAMETERS

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

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

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

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

/LOG Switch default: ”
Output list of copied spectra

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

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

Caller mdbm

Author H.G.Essel

Example

```
WRITE CA SP A(1,2)
WRITE CA SP A(*)
WRITE CA SP A* /ADD
WRITE CA SP A (copy first spectrum only)
```

Remarks

File name E$AWCSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$AWCSP(CV_NAME,CV_SPEC_DIR,CV_BASE,CV_NODE,
                I_LOG,I_ADD,I_KEEP_MAP)
```

COMMAND

```
WRITE CAMAC SPECTRUM name spec_dir base node
                /ADD
                /LOG
                /[NO]KEEP_MAP
Argument description
```

NAME

Routine arg. Input CHAR(*) VAR

Command par.

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

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par.

```
string global replace default: '$SPECTRUM'
  Name of Spectrum directory
```

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Name of Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Name of Node

LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Output list of copied spectra

ADD

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Instead of overwrite, add spectrum channles to MR2000 channels.

KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Inhibit the unmap (detach) of the Data Base

Function

The data part of MR2000 is writte from specified * GOOSY spectrum. The range in the MR2000 is defined in the spectrum. Channel contents may be overwritten (default) or added (/ADD).

ZOOM FRAME

ZOOM FRAME frame picture dyn_scat
pic_dir dyn_dir buffer_size base node

PURPOSE Zoom one frame of a picture.

PARAMETERS

frame Frame number as displayed on screen

picture Name of picture which contains the frame to be zoomed.

dyn_scat Name of dynamic list for scatter plot

pic_dir Default picture Directory

dyn_dir Default Directory name for dynamic list

buffer_size Number of scatter points in scatter buffer

base Default Data Base

node Default node name

Caller MDISP, MGOODISP

Author W .Spreng

Remarks

File name GOO$DISP:D$ZOOM.PPL

Created by GOO$DISP:D$DSPCM.PPL

Description

CALLING STS=D$ZOOM(L_frame,CV_picture,CV_dyn_scat,
CV_pic_dir,CV_dyn_dir,CV_buffer_size
,CV_base,CV_node)

COMMAND ZOOM FRAME frame picture dyn_scat
pic_dir dyn_dir buffer_size base node

FRAME

Routine arg. Input BIN FIXED(31)

Command par. Integer required

Number of the frame which should be zoomed.

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of the picture from which the specified frame should be zoomed. If not specified the actual active picture is used.

DYN_SCAT

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$SCATTER

Name of dynamic list used to create the entries necessary for the scatter plot. The SCATTER entry in this dynamic list is necessary to tell the analysis which scatterplots are requested by your display.

ATTENTION The dynamic list has to be attatched before a scatterplot could be created.

PIC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$PICTURE

Default Directory for pictures.

DYN_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global raplacable default=$DYNAMIC
Default Directory name for dynamic list

BUFFER_SIZE

Routine arg. Input CHAR(*) VAR

Command par. String global replace default=1000
Number of scatter points in scatter buffer.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB
Default Data Dase.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*
Default node for Data Base.

Function

The specified frame in the specified picture is displayed on the screen. It is not necessary that the picture is displayed on the screen. If the picture is omitted the current active picture is assumed.

Chapter 2

DCL Procedures

acctng

```
@accttng p1 p2 p3
```

PURPOSE Takes the ACCOUNTNG file of VMS and do selective inquiries using ACCOUNTING (the VMS utility).

ARGUMENTS P1 Input accountng.dat file. The file must be on
directory SYS$SPECIFIC:[ACC]
(on DONALD = V8600A$ACC, on EMMA = V8600B$ACC).
P2 Start date from which accounting report begins P3 End date until accounting report stops

Description

FUNCTION ACCTNG analyzed the input file for selective inquiries and check P2 and P3 of valid dates.ACCTNG is the main driving COM file. The DCL Procedure ACC_SUB1.COM is called from this Module as PERCENTER.EXE .

File name ACCTNG.COM

Dataset goo$exe

Version 1.01

Author T.KROLL

Last Update 21-FEB-1986

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

BFXT

@GOO$EXE:BFXT inirep action file /ID=MYDATA/MODE=CHAR /FILELN=fileln/MSGLEVEL=msglevel/NOEXECUTE

PURPOSE	send data to the IBM via BFX/NETEX
ARGUMENTS	
inirep	INITIALIZE or REPLY, I or R, the partner has to answer with R or I respectively
action	SEND or RECEIVE, S or R
file	VAX file to be sent
/FILELN=	logical name of the file to be transferred (default: NSC$BFXT)
/MSGLEVEL=	message level, 0 most information
/ID=	id of the transfer
/MODE=	mode (CHAR or BIT)
/NOEXECUTE	generated command procedure will be kept and will not be executed

Description

FUNCTION	A DCL procedure is edited in which a BFX transfer of the specified file is performed. This temporary procedure is held on a dataset BFX_<generic>.TMP. After that it will be executed, unless /NOEXECUTE is specified. The keywords specify (NETEX) parameters of the transfer.
Version	1.01
Remarks	On the IBM has to be a counterpart performing a BFXT(I or R) with RECEIVE or SEND respectively

Author	K.Winkelmann	
Last Update	30-MAR-1987	
	21-OCT-87	Error removed in Stringhandling/P.K.
	12-FEB-88	Error removed in DELETE of tempfile /KW

CADDRESS

CADDRESS infile outfile /PRINTER=p

PURPOSE Create and optional print address files..

ARGUMENTS

infile Input file

outfile Ouput file

/PRINTER=p Print file on laser printer p. p=A,B,C...

DESCRIPTION

CALLING CADDRESS infile outfile /PRINTER=p

PURPOSE Create and optional print address files..

ARGUMENTS

infile Input file

outfile Ouput file

/PRINTER=p Print file on laser printer p. p=A,B,C...

FUNCTION An address list from the file with the logical name INPUT will be formatted and written to the file with the logical name OUTPUT. Each address item must be separated with two leading "??", the line with the ZIP code must be precedet by one "?". If the string 'D R U C K S A C H E' appears in an input line an extra blanck line will be written after that line. The output file can be printed on a DEC laser by commands like PA80F, PB80F, PC80F,... depending on the laser printer to be used. This output can be copied by any copier on copy label sheets, e.g. 'HERMA copy-print DIN A 4'.

The logical names INPUT and OUTPUT must be defined before the call.

Example

EXAMPLE

Usage:

```
$ DEFINE INPUT KP1$ROOT:[USER.TEXT]ADDR.DAT
$ DEFINE OUTPUT KP1$ROOT:[USER.TEXT]ADDR.LIS
$ MADDR
$ PC80F OUTPUT
or
$ DEFINE/USER INPUT KP1$ROOT:[USER.TEXT]ADDR.DAT
$ DEFINE/USER OUTPUT KP1$ROOT:[USER.TEXT]ADDR.LIS
$ MADDR
$ PC80F KP1$ROOT:[USER.TEXT]ADDR.LIS
```

Example of input format:

```
??D R U C K S A C H E
Herrn
H. Kiowski
Hochschul-Rechenzentrum der
Universitt Frankfurt
Postfach 11 19 32
?6000 Frankfurt/M 1
??D R U C K S A C H E
Herrn
Prof. Dr. H. Backe
Physikalisches Institut der
Universitt Mainz
Postfach 3980
?6500 Mainz
```

CADJUST

CADJUST infile outfile /ADJUST/TAB

PURPOSE	Adjust GOOSY documentation headers.
ARGUMENTS	
infile	Input file
outfile	Optional ouput file
/ADJUST	Adjustment also done outside documentation headers
/TAB	Replace tabs by spaces

DESCRIPTION

CALLING	CADJUST infile outfile /ADJUST/TAB
ARGUMENTS	
infile	name of input file (COM,PLI,PPL,FOR,PAS,MOD,MAR,MSG)
outfile	name of output file (def. = input)
/ADJUST	Adjustment is done outside headers too. There is, however still a difference between headers and code: Outside it is not required that a comment line is closed. A warning will be given, but no close delimiter is added. Actually the line is not modified. Inside, an open comment line is closed without notice. Tabs are replaced everywhere.
/TAB	Replaces in headers spaces with tabs.

FUNCTION

In sources of FOR,COM,MSG,MAR comment lines will be stripped from trailing spaces and delimiters. PLI,GIP,PAS,MOD comment lines will be adjusted to 72 characters per line. If adjustment is not possible (line too long) a message is printed to terminal. The last character before the end delimiter is used as fill character. Adjustment is performed only in header blocks except /ADJ is specified. Only lines with a comment delimiter in first column are processed. All TABs are replaced by SPACEs in all lines If /TAB is specified however, all SPACEs are converted back into TABs if this reduced the file size. Number of lines and maximal line length are printed at the end.

CALLEX

CALLEX library filespec GL-switches

PURPOSE	Extract Calling sequence from PLI source.
ARGUMENTS	
library	Destination text library (GOOCALL)
filespec	PLI source file specification (Wildcards)
GL-switches	Switches used for GLPUT

Description

FUNCTION	-
File name	CALLEX.COM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	26-NOV-1985

CALLTREE

CALLTREE input /MODULE= /OUTPUT= /TT /EXCLUDE=

PURPOSE	Makes a calling tree out of a cross reference
ARGUMENTS	
input	Input specification. This can be a Library or a map file.
/MODULE=	Module name. The calling tree is built only for this module.
/OUTPUT=	Output file. If omitted the output gets the name from the input with the extension ".TRE".
/TT	Output on the terminal
/EXCLUDE=	List of Module names, excluded in the output

Description

FUNCTION

Creates a calling tree from a cross reference list. If the input is an object-library LIBRARY is called with the /CROSS_REFERENCE qualifier. Cross reference list or map-file are input to the called Program MCALLTREE. This program builds the calling tree for all modules or for one module specified with /MOD. Output is written to terminal or output file.

EXAMPLE:

CALLTREE GOOLIB /OUTP=GOOTREE.LIS

Generates a cross_reference list from GOOLIB, builds the calling tree and writes output to GOOTREE.LIS

CALLTREE GOOLIBSHR.MAP /MOD=S$MESS /TT

Writes calling tree from S$MESS on the Screen.

File name CALLTREE.COM

Dataset -

Version	1.0
Author	G.Schneider
Last Update	20-FEB-1986

CBACKUP

CBACKUP P1 P2 ... P8

PURPOSE Execute the VMS BACKUP utility and return an exit status based on the analysis of all errors.

ARGUMENTS

P1...P8 Any valid parameters or qualifiers for BACKUP

Description

CALLING CBACKUP P1 P2 ... P8

ARGUMENTS

P1...P8 Any valid parameters or qualifiers for BACKUP

FUNCTION The VMS BACKUP utility is executed with the parameters and qualifiers specified. The BACKUP error messages are than analysed in order to determine a more usefull exit status. The exit status will be:

SUCCESS	All files processed.
WARNING	Not all files could be processed due to the inaccessability of files or directories. Normaly at least one of the following errors had occured: File is read protected File is open for write File is deacess locked (RSX) File is deleted (NOSUCHFILE)
ERROR	Severe system errors like parity errors ect. occured.
FATAL	Any fatal BACKUP error like device full ect. occured.

REMARKS -

EXAMPLE CBACKUP *.PLI TEST.BCK/SAVE

CBATCHDCL

```
CBATCHDCL "dcl_line1" "dcl_line2"
          "dcl_line3" "dcl_line4"
          /LOG_FILE=file /DEFAULT=dir
          /QUEUE=q /AFTER=time /NAME=n
          /CHARACTERISTICS=c/CPUTIME=t/RESTART
          /NODE=n
          /VERIFY/ACCOUNTING
          /WARNING/ERROR/FATAL
```

PURPOSE Executes up to 4 DCL command lines in a BATCH job under the current default directory either on the local or an remote node.

ARGUMENTS

dcl_line1..4 Up to 4 DCL commands, almost any command without prompts or continuation lines is legal.

/LOG_FILE=file Log file name for SYS$OUTPUT of command

/DEFAULT=dir The default device and directory setting.

/QUEUE=q Batch queue name (default= 'BATCH_QUEUE').

/AFTER=time Start of job execution (default = immediate).

/NAME=n Job name to be used.

/CHARACTERISTICS=c Job charactersistics

/CPUTIME=t Job time limit

/RESTART Enables job restart after crash

/NODE=n The job is executed on another node.

/VERIFY Enables verify

/ACCOUNTING Executes an SHOW PROC /ACCOUNTING at the end

/WARNING Abort after the first warning

/ERROR Abort after the first error (default)

/FATAL Abort after the first fatal error

Note The last DCL command and the first qualifier MUST be separated by a blank or tab.

DESCRIPTION

CALLING

```
CBATCHDCL "dcl_line1" "dcl_line2"
          "dcl_line3" "dcl_line4"
          /LOG_FILE=file /DEFAULT=dir
          /QUEUE=q /AFTER=time /NAME=n
          /CHARACTERISTICS=c/CPUTIME=t/RESTART
          /NODE=n
          /VERIFY/ACCOUNTING
          /WARNING/ERROR/FATAL
```

ARGUMENTS

dcl_line1..4 Any DCL command. If any delimiters (blanks,..) are included, the command has to be enclosed in quotes ("). If no line is specified, up to 4 DCL commands are prompted. NOTE: Command lines including a quote are permitted, but in this case all quotes within the command line have to be doubled.
Note further, that the last DCL command and the frist qualifier MUST be separated by at least one blank or tab. This is in contrast to the usual syntax of DCL commands !

/LOG_FILE=file Output file for SYS$OUTPUT log-file.
If this parameter is ommitted, a file with the name
CBATCH_xxx.LOG (xxx=job number)

is used as log file. This file is deleted if the command completes with a success or informative status but kept in all other cases. If a file name is specified, a file with the default type .LOG is created under the default directory. NOTE: Only the output to SYS$OUTPUT produced by the command is written to this file. The protocol of the LOGIN and LOGOUT procedure is omitted.
The job number is determined by examination of the process name. If the process name is changed by your LOGIN procedure to a format other than

<string>_<job-number>

the PID is used instead of the job number.

/DEFAULT=dir The default device and directory setting for the batch job. The default is the current setting of the issueing session.
NOTE: The job must have write access to the default directory because the log file is stored there. Therefore SYS$LOGIN: is taken as default if the /NODE qualifier has been specified with an access control string indicating execution under another account.
A VMSmail is send (even on the local node) if the job can't write access the default directory.

/QUEUE=q Batch queue name. If not specified, the default is taken from the symbol BATCH_QUEUE. If this also is not specified, SYS$BATCH is used.

/AFTER=time Requests, that the batch job is held till the specified time. The defualt is immediate execution For details refer to the documentation of the DCL command
SUBMIT /AFTER

/NAME=n The batch job name to be used. If not specified, the name is build automatically with the format
CBATCH_<verb>

where 'verb' is either the first command token of the DCL command line or the file name of command procedure.

/CHARACTERISTICS=c Job charactersistics, for details look in the description of the DCL SUBMIT command. Ask your system manager about usage of characteristics.

/CPUTIME=t Job time limit. For details look in the description of the DCL SUBMIT command.

/RESTART Enables job restart after crash. For details look in the description of the DCL SUBMIT command.

/NODE=n The batch submit on the specified node. The /LOG qualifier is ignored in this case, the log file is always send back as a MAIL message.
NOTE: If /NODE is specified as a node name with account information and if /DEFAULT is not specified, the default directory will be

SYS$LOGIN of the account specified in the node name. This ensures, that the default directory is writable.

/VERIFY Enables verify. Executes an SET VERIFY as the first DCL command.

/ACCOUNTING Executes an SHOW PROC /ACCOUNTING as the last DCL command.

/WARNING Abort after the first warning.

/ERROR Abort after the first error (default).

/FATAL Abort after the first fatal error.

FUNCTION This procedure submits a standard batch job which executes up to 4 DCL command lines.
The procedure GOO$EXE:CBATCHEXE.COM is submitted, all other information is packed in the submit parameters

REMARKS The batchjob runs in the context defined by your login procedures (SYSTEM login plus user LOGIN.COM). Therefore only logical names and DCL symbols defined in those procedures are known in the batchjob. .SK The only information passed from your session to the batch environment is the default directory, the command lines and the log file name and in case of a remote submit the issueing node and user.

EXAMPLE CBATCHDCL "SH USER" /LOG=SHUSER Will write a list of all users to the file SHUSER.LOG under the current default directory

* CBAT "PUR *.*/LO" Will purge all files on the current default directory. A logfile with the name JOBxxx.LOG will be available is any error occurs during the purge command.

* CBAT "NWDCL * ""SH US""" Example for a command line including quotes.

* CBAT "DIR" /NODE=V8600A Will be submitted on node V8600A, the log file containing the directory listing is returned as a MAIL message.

* CBAT "SHOW PROC" /NODE=V750A"USER password" Will submit a job for USER on node V750A and return the log file as MAIL message. Note, that in this case the default directory is SYS$LOGIN: of USER.

CBINHEX

CBINHEX in_file out_file

PURPOSE This procedure creates a HEX file containing the FDL description and the contents in hexadecimal format of another file. Those HEX files can be converted back with CHEXBIN and are usefull for sending them over stupid communication links.

ARGUMENTS

in_file Any input file.

out_file Output HEX file. If this parameter is omitted, the file name is that of the in_file, the type will be ".HEX".

Description

FUNCTION -

File name CBINHEX.COM

Dataset -

Version 1.01

Author B.Kolb

Last Update 24-OCT-1985

30-oct-85	changed open to create and append to avoid VFC record type./B.Kolb

CDBLQUOTE

CDBLQUOTE input

PURPOSE Replace every quote in a text string by a double quote.

ARGUMENTS

input Name of input DCL symbol

DESCRIPTION

CALLING CDBLQUOTE input output

ARGUMENTS

input Name of input DCL symbol

Example CDBLQUOTE IN

FUNCTION All single quotes (') will be replaced by double quotes (""). The resulting string is written to the global DCL symbol _CDBLQUOTE_OUTPUT.

REMARKS -

CDCLCOM

CDCLCOM inc module

PURPOSE performs a compile of PL/I text contained in 'file' within a dummy frame (GOO$DM:U$DCLDU.PPL)

ARGUMENTS

inc name of file or library containing PL/I text. This text should reasonable within an embracing PL/I procedure.

module name of module, if library specification is used, empty, if file is used

Description

FUNCTION if a file is specified the contents of 'inc' is included into the procedure U$DCLDU and then compiled in a separate process. In case of a library specification, a library extract from 'inc(module)' to a temporay file is performed.

File name goo$exe:CDCLCOM.com

Dataset -

Version 1.01

Author K.Winkelmann

Last Update 16-oct-1985

Updates

26-sep-85	compile with /DEB reveals more warnings /KW
01-oct-85	/switch replaced by /qual for COMPILE /HE
16-oct-85	library specification is allowed /KW

CDCLLIST

CDCLLIST dsc p1 p2 p3 p4 p5 p6 p7

PURPOSE This procedure provides a mechanism which allows to pass parameters and qualifiers to DCL procedures in the same way as to DCL commands.

ARGUMENTS

dsc Argument descriptor string.

p1..p7 Parameters passed to calling procedure

DESCRIPTION

CALLING CDCLLIST dsc p1 p2 p3 p4 p5 p6 p7

ARGUMENTS

dsc Argument descriptor string.
This string contains a description of the allowed parameters and qualifiers for the called procedure.
Format:
"pref para_dsc qual_dsc"

pref Name of the calling procedure, will be used as with an leading underscore as prefix for all symbolnames.

para_dsc Parameter descriptor.
This descriptor consists of a series of parameter definitions separated by one blank.
A parameter definition has the format:
name[=default]

The parameter is treated as a required parameter and will be prompted if the default is a question mark (?).

qual_dsc — Qualifier descriptor.
This descriptor consists of a series of qualifier definitions separated by a slash. A qualifier definition has the format:

/name[=[default]]

The name may be specified with an imbedded asterisk (*) to mark the allowed abriviation. The qualifier may have a value if a default, even a blank one, is specified.

p1..p7 — Parameters passed to calling procedure

FUNCTION

The following actions are preformed:

1.The descriptor is parsed

2.For every parameter a symbol with the name

<pref><name>

is defined and set with either

a specified positional parameter p1..p7

a prompted value if required (default=?)

or the default string (if ommitted blank)

3.For every qualifier a symbol with the name

<pref><name>

is defined. If no default string was specified a simple qualifier is expected and the symbol is set with either

the full qualifier name with a leading slash

or a null string

If a default string, even a null string, was specified a qualifier with value is expected and the symbol is set with either

the qualifier value

or the default string

REMARKS

-

EXAMPLE

Some calls of CDCLLIST with the resulting symols:

```
CDCLLIST "TEST para1 para2 para3" xxx yyy
        _TEST_PARA1 = "XXX"
        _TEST_PARA2 = "YYY"
        _TEST_PARA3 = ""
CDCLLIST "TEST para1=? para2=? para3=murks" abcd
        _TEST_PARA1 = "ABCD"
```

```
        _TEST_PARA2 will be prompted !!
        _TEST_PARA3 = "murks"
    CDCLLIST "TEST para1 /li*st/out*put=" xx /LI
        _TEST_PARA1 = "XX"
        _TEST_LIST = "/LIST"
        _TEST_OUTPUT= ""
    CDCLLIST "TEST file=? /li*st/comm*ent=" -
       test.dat /LIS/COMM="This is a comment"
        _TEST_FILE = "TEST.DAT"
        _TEST_LIST = "/LIST"
        _TEST_COMMENT= "This is a comment"
```

The CDCLLIST procedure was initially used in the module management procedure paket. You will find a lot of examples in the procedures:

```
GOO$EXE:GLSET.COM
GOO$EXE:GLSHOW.COM
ect.
```

CDELSYM

CDELSYM prefix start end

PURPOSE	Delete global symbols created by CNAMELIST or CWILDCARD
ARGUMENTS	
prefix	Prefix of symbols
start	lowest number of symbol (normally 0)
end	highest number of symbols

Description

FUNCTION	CWILDCARD and CNAMELIST generate symbols of the form <prefix>_<number>. <number> runs from 0 to <prefix>_0. These symbols can be deleted by CDELSYM.
File name	CDELSYM.COM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	19-JUL-1985

CDIFFER

CDIFFER file1 file2 /LIST/PARA/DIFFER

PURPOSE	Allows wildcarded DIFFERENCE with suppressed output.
ARGUMENTS	
file1	Wildcarded file specification.
file2	optional file specification.
/LIST	optional output differences. Otherwise differences are not output, only file names.
/PARA	optional output differences in parallel.
/DIFFER	Only names of different files are listed.

Description

FUNCTION	Calls DIFFERENCE. Outputs only a message if files are different or not unless /LIST is specified. Then the differences are output too.
Version	1.01
Author	G.H.Essel
Update	18-APR-1988
Last Update	R.Fritzsche 04-OCT-1989

CDIRO

CDIRO file-spec. /SIZE /DATE /FULL/TEST

PURPOSE	sorts a DIR command output
ARGUMENTS	
file-spec	any valid wild carded file name
/SIZE	list is sorted with descending size
/DATE	list is sorted with increasing age, last modified files appear first (default)
/FULL	every file is listed with the complete directory name
/TEST	for test purposes

Description

FUNCTION	a DIR command is taken and the generated list is sorted according to the specified SIZE or DATE respectively
File name	CDIRO.COM
Dataset	-
Version	1.01
Author	K.Winkelmann
Last Update	11-FEB-1985 03-JUL-1985 SHORT is default, /FULL has to be specified if every file should be listed with the complete directory specification

CDOCUM

CDOC module type formatter library

PURPOSE Generates documentation from source file and inserts text module to library

ARGUMENTS

module name of module (e.g. U$MENU)

type type of source file (e.g. PPL or COM)

format (SCR,RNO,HLP,DOC,TOC,BRIEF)

library name of text library (opt.) or logical name

DESCRIPTION

CALLING CDOC module type formatter library

ARGUMENTS

module name of module (e.g. U$MENU)

type type of source file (e.g. PPL or COM)

format (SCR,RNO,HLP,DOC,TOC,BRIEF)

library name of text library (opt.) or logical name

Example CDOC MGENHEA PPL RNO PROGRAM.TLB CDOC U$ASCII PPL SCR,HLP GOOTMOD

FUNCTION See CEXTRACT, CFORMAT, MEXTHEA, MFORMDO. If a library is specified, a text module is inserted.
Depending on the formatter specification, help and print files (RNO) are generated and/or a script file (SCR).

REMARKS If the module name contains '$' signs, these are replaced by 'X' to built the file name

CEASYMAIL

@GOO$EXE:CEASYMAIL file recipient /SUBJECT=s/DELETE

PURPOSE	Send a file as MAIL to a High Energy Physics DECnet user via EARN via the Mailer at CUNYVM.
ARGUMENTS	
file	File to be sent
recipient	Recipient user name specification
/SUBJECT=s	One line subject description
/DELETE	Delete file after send

Description

CALLING	@GOO$EXE:CEASYMAIL file recipient /SUBJECT=s/DELETE
ARGUMENTS	
file	File to be sent. Must be a printable file in any RMS record format, but the maximal record size is limited to 80 character (Hollerit never dies).
recipient	Recipient user name specification of an EASYnet (DEC) node. The format is in the normal VMSmail notation: [EASY::]<node>::<user>
/SUBJECT=s	One line mail subject description. Note, that the string must be enclosed in quotes if it contains delimiters like a blank.
/DELETE	Signals, that the file is to be deleted after being sent. Note, that there is no guarantee at all, that the mail will reach the addressee. The file is nevertheless deleted.

FUNCTION This procedure sends the given file over EARN and BITNET to the CUNYVM EASY-gateway, which forwards the file to the given EASY net node.

The procedure generates a temporary file, which contains the proper header and trailer for the CUNYVM gateway. Than it submits a batch job which uses the JNET SEND command to send this file to the gateway.

REMARKS -

EXAMPLE @GOO$EXE:CEASYMAIL TEST.TXT EASY::VXALFB::USER

CEDITDOC

CED file key

PURPOSE Calls MGENHEAD for header generation and the VMS editor LSEDIT.

ARGUMENTS

file file name to be created

key PLI,FOR,DCL,TSO,VAX,IBM,MOD,PAS,MAIN,PROC,BRIEF
e.g. PLI,PROC
FOR,MAIN,BR
Default is DCL,BRIEF

DESCRIPTION

CALLING CED file key

ARGUMENTS

file file name to be created

key Specifies type of source
PLI,MAIN[,BRIEF]
PLI,PROC[,BRIEF]
DCL[,BRIEF]
TSO[,BRIEF]
VAX[,BRIEF]
IBM[,BRIEF]
PAS[,BRIEF]
MOD[,BRIEF]

Example CED MGENHEAD.PPL PPL,MAIN

FUNCTION See MGENHEAD. The header is interactively generated. Then the editor is called.

CFILTYPES

CFILT*YPES file switches /D*IRECT

PURPOSE Outputs list of all file types on a directory

ARGUMENTS

file Directory specification, from where the file types are collected. Directories must not be wildcarded. Default= [current dir]*.*.

switches Directory switches for /DIR (opt.) Switches must be enclosed in "", if they contain DCL delimiters as slashes! /EXCLUDE switch is not allowed!

/D*IRECT Output directory listing for each file type (opt.)

Description

FUNCTION A directory is done. Then all different file types are printed. If /DIR is specified, directories for each file type are output using 'switches'. The listing is written to SYS$OUTPUT, unless a file is specified by /OUTPUT=file.

File name CFILTYPES.COM

Dataset -

Version 1.01

Author H.G.Essel

Last Update 26-AUG-1985

CHANAL

CHANAL infile outfile /COPY/FULL

PURPOSE	Check GOOSY analysis programs.
ARGUMENTS	
infile	Input file with analysis routine.
outfile	Output file for new analysis routine (/COPY).
/COPY	Copy input to output file. The type specifications are inserted in the $LOC macro calls if not there.
/FULL	List lines which are modified

Description

FUNCTION	Calls MANALCH to check calling $ACCU and $COND macros. Condition types and spectrum data types are checked in the $LOC macros. The input file is copied to the output file if /COPY is given. Otherwise only a check is done.
Version	1.01
Author	H.G.Essel
Last Update	10-JAN-1989

CHECKDATE

CHECKDATE newfile oldfile

PURPOSE Checks wether newfile is younger then oldfile (status=1) or not (status=11)

ARGUMENTS

newfile Filespecification for file which is assumed to be younger according creation date.

oldfile File which should be older.

Description

FUNCTION The f$file_attributes function is used to determine the creation dates of the both files. Status is 1 if newfile is younger then oldfile and exists, otherwise 11. If oldfile does not exist, status is 1. If newfile does not exist, status is 11.

File name CHECKDATE.COM

Dataset -

Version 1.01

Author H.G.Essel

Last Update 26-JUL-1985

CHEPMAIL

@GOO$EXE:CHEPMAIL file recipient /SUBJECT=s/DELETE

PURPOSE Send a file as MAIL to a High Energy Physics DECnet user via EARN via the Mailer at CERNVAX.

ARGUMENTS

file File to be sent

recipient Recipient user name specification

/SUBJECT=s One line subject description

/DELETE Delete file after send

Description

CALLING @GOO$EXE:CHEPMAIL file recipient /SUBJECT=s/DELETE

ARGUMENTS

file File to be sent. Must be a printable file in any RMS record format, but the maximal record size is limited to 80 character (Hollerit never dies).

recipient Recipient user name specification of an HEP node. The format is in the normal VMSmail notation:
[HEP::]<node>::<user>

/SUBJECT=s One line mail subject description. Note, that the string must be enclosed in quotes if it contains delimiters like a blank.

/DELETE Signals, that the file is to be deleted after being sent. Note, that there is no guarantee at all, that the mail will reach the addressee. The file is nevertheless deleted.

FUNCTION This procedure sends the given file over EARN and BITNET to the CERNVAX HEP-gateway, which forwards the file to the given HEP net node.

The procedure generates a temporary file, which contains the proper header and trailer for the CERNVAX gateway. Than it submits a batch job which uses the JNET SEND command to send this file to the gateway.

REMARKS -

EXAMPLE @GOO$EXE:CHEPMAIL TEST.TXT HEP::VXALFB::USER

CHEXBIN

CHEXBIN in_file out_file

PURPOSE This procedure processes a HEX file send of a stupid communication network and creates the file described in the HEX file.

ARGUMENTS

in_file File in HEX format, the default type is ".HEX". This file must have been created with the CBINHEX procedure and must contain the FDL file specification and the contents of the file in hexadecimal format. The file may contain further headers and some trailing junk. This parts are ignored.

out_file Output file. The default name and type is that of the original file as determined from the FDL header. The file will be created in the current default directory if this parameter is omitted.

Description

FUNCTION This procedure extracts the FDL header, creates a file with CREATE/FDL and calls MHEXBIN to fill in the contents.

File name CHEXBIN.COM

Dataset -

Version 1.02

Author B.Kolb

Last Update 24-OCT-1985

CINCHLP

CINC*HLP format incl.libr.(module) help-libr.

PURPOSE	Generates documentation of include modules.
ARGUMENTS	
format	Formatter list (passes to CFORMAT)
module	Module specification (wildcard).
incl.libr.	Text library containing the modules.
help-libr.	Help library to be updated (opt.).

Description

CALLING	CINC*HLP format incl.libr.(module) help-libr.
ARGUMENTS	
format	Formatter list (passes to CFORMAT)
module	Module specification (wildcard).
incl.libr.	Text library containing the modules.
help-libr.	Help library to be updated (opt.).
FUNCTION	The text modules are extracted one by one from the text library. A small documentation header with module name and library name is added following the rules for GOOSY documentations. The module body itself is written under the subheader "Body". If a help library is specified, the helpfile is added to this library.
REMARKS	PLI include modules only.
EXAMPLE	CINC HLP,SCR GOOINC($*) GOOHINC Extract modules beginning with "$" from GOOINC and format for SCRIPT and HELP. The temporary help file is added to GOOHINC, the SCRIPT source is in GOOINC.SCR.

CINSNUM

CINS input output

PURPOSE Inserts line numbers (9 char.) to source files

ARGUMENTS

input name of input file

output name of output file

DESCRIPTION

File name CINSNUM.COM

Dataset (CINSNUM)

Version 1.01

Author H.G.Essel

Last Update 15-JAN-1984

REMARKS -

CLCOUNT

CLCOUNT file symbol /NOHEAD

PURPOSE	Outputs number of lines of source files
ARGUMENTS	
file	Wildcarded file specification. The lines of these files are summed up.
symbol	optional DCL symbol to be set to number of lines. Note that the symbol is not set to zero at the beginning.
/NOHEAD	Documentation headers are skipped

Description

FUNCTION	Calls MLCOUNT /NOHEAD for each of the specified file and sums up the number of lines found. /NOHEAD counts outside the documentation headers only.
Version	1.01
Author	H.G.Essel
Last Update	3-APR-1986

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

CMAIL

CMAIL file recipient /ED*IT /DEL*ETE /SUB*JECT=s /NOSE*LF /SPOOL=s

PURPOSE Send or spool a MAIL with VMSmail or other mail systems on V8600B only

ARGUMENTS

file File to be sent

recipient Recipient name specification or distribution list. This may be a logical name, i.e.:

EARN/BITNET	EARN::<node>::<user>
DFN	DFN::<node>::<user> use DFN also for Internet or networks not listed here
ARPA net	ARPA::[<net>::]<node>::<user>
Finland net	FI::[<net>::]<node>::<user>
Australia	AU::[<net>::]<node>::<user>
JANET (GB)	JANET::<node>::<user>
HEP DECnet	HEP::<node>::<user>
EASYnet	EASY::<node>::<user>
X25	PSI::<node>::<user>

/EDIT Call the LSEDIT to edit the file first

/DELETE Delete file after send

/SUBJECT=s One line subject description

/NOSELF Disables sending a copy to the sender

/SPOOL=s signals that this a MAIL spool request

Description

CALLING CMAIL file recipient /ED*IT /DEL*ETE /SUB*JECT=s /NOSE*LF /SPOOL=s

ARGUMENTS

file File to be sent. If the /EDIT qualifier was given the file will be created if not existing already and the file can then be edited with LSEDIT.

recipient Recipient name specification. Everything allowed for VMSmail may be entered for this parameter:

- a single user name
- a distribution list reference (@filename)
- a list of users and distribution lists

A distribution list contains lines in the formats listed above, "!" is the comment delimiter. Note, that distribution lists may be nested to any depth allowed by your open file quota. In addition to VMSmail addresses one may specify network addresses in the form:

EARN/BITNET	EARN::<node>::<user>
DFN	DFN::<node>::<user> use DFN also for Internet or networks not listed here
ARPA net	ARPA::[<net>::]<node>::<user>
Finland net	FI::[<net>::]<node>::<user>
Australia	AU::[<net>::]<node>::<user>
JANET (GB)	JANET::<node>::<user>
UK net	UK::[<net>::]<node>::<user>
EDU net	EDU::[<net>::]<node>::<user>
HEP DECnet	HEP::<node>::<user>
EASYnet	EASY::<node>::<user>
X25	PSI::<node>::<user>

/EDIT The LSEDIT is called first to edit the input file or create the input file.

/DELETE Signals, that the file is to be deleted after being send.

/SUBJECT=s One line mail subject description. Note, that the string must be enclosed in quotes if it contains delimiters like a blank.

/NOSELF Disables sending a copy of the mail file to the sender. Note, that the VMSmail default (set with the MAIL command SET COPY) is ignored by this procedure. This is the default, if the sender is in the list of recipients. This avoids the reciept of two copies of the same MAIL.

/SPOOL=s ONLY FOR INTERNAL USE. Signals, that this is a mail spool request, the value specifies the send date of the mail.

FUNCTION This procedure builds the distribution list and calls either MAIL (for DECnet) or CEARNMAIL (for EARN) or CARPAMAIL (for ARPAnet) or others to send the mail. If a send fails, it is retried for a week and then returned with VMSmail to the sender.

REMARKS If executed in a NETWORK process, the procedure submits a BATCH job to do the work because MAIL can't be used in a NETWORK process. The mail spool jobs have the job name CMAIL_SPOOL. If they are deleted with DELETE/ENTRY, dont't forget to delete their temporary files CMAIL_*.TMP. The mail spooling retries every 15 minutes during the first hour, every hour for the first 6 hours, every 6 hours for the first day and every day for a full week. The mail is then returned to the sender with VMSmail and may be reused with the MAIL EXTRACT command. CMAIL runs on V8600B only

EXAMPLE

```
CMAIL TEST.TXT @DIST.DIS
  CMAIL TEST.TXT EARN::DDAGSI1::USER
  CMAIL TEST.TXT DFN::VAX.HMI.DBP.DE::USER
  CMAIL TEST.TXT HEP::VXALFB::USER
```

CMANUAL

CMAN <utility list> <format> <prefix>

PURPOSE Composes GOOSY manuals (Programs, modules, procedures)

ARGUMENTS

u.list List of text libraries for input

format Format list (Qualifier list for GLFORMAT)

prefix Prefix for putput file names on GOO$DOC

DESCRIPTION

CALLING CMAN <utility list> <format>

ARGUMENTS

u.list List of text libraries for input

format Format list (Qualifier list for GLFORMAT)
/TOC generates files with short descriptions of all modules. These files are concatenated with the documentation files. The help entry for the short description is DIRECTORY.

prefix Prefix for putput file names on GOO$DOC. If the BRIEF switch is specified, prefix defaults to 'BR_'.

FUNCTION Each library of the list is processed separately. All text is extracted and formatted as specified. Output is written to files named as the text libraries with proceding prefix The directory GOO$DOC is used to store the results. Temporary files are on GOO$SCRATCH.

EXAMPLE CMAN GOOTUTIL,GOOTMOD /SCR/HELP/TOC NEW_ Do documentation for GOOTUTIL and GOOTMOD. Generate script, help files with short descriptions (TOC).

NEW_UTILITY.SCR, NEW_UTILITY.HLP
NEW_MODULE.SCR, NEW_MODULE.HLP

CMESSAGE

CMESSAGE facility [switch]

PURPOSE Utility to insert a new message in a message file

ARGUMENTS

facility Name of facility

switch optional:

NEW A new facility and facility file is created. In this case the facility directory must be updated. The editor is called for that purpose. This directory can be displayed by
H @MES FACIL

EXAMPLE CMESS IO

DESCRIPTION

File name CMESSAGE.COM

Dataset -

Version 1.01

Author H.G.Essel

Last Update 10-JAN-1984

REMARKS The files must be on GOO$MSG and must have names
X<facility>.MSG
The file is loaded from GOO$MSG to private directory. The editor is called. A test compilation is done. The modified file is copied back to GOO$MSG. If a new file is to be created, a form is copied from GOO$MSG to X<facility>.MSG. Then the procedure continues as described above.

NOTE The message files must have following attributes:

```
GLSET ATT GOO$MSG:x.MSG COM/OLB=GOOLIB
GLSET ATT GOO$MSG:x.MSG EPI -
        /EPI1="CMSG GOO$SCRATCH GOO$MSG:x.MSG"
GLSET ATT GOO$MSG:x.MSG UPD/MARK=COM,EPI
```

CMODMAP

CMODMAP file /COMPILE /OUTPUT=file

PURPOSE	Outputs list of modules called by module in file. (Only first level calls)
ARGUMENTS	
file	File specification of a PLI-file (source or list).
/COMPILE	File is compiled with /show=map/lis into temporary files. The list file is analyzed. This qualifier is default for filetypes PPL and PLI.
/OUTPUT=	Output is written to file.

Description

FUNCTION	The PLI-compiler generated listing (/SHOW=MAP) is passed through SEARCH. The output is passed through MMODMAP. If a PLI-list file already exists, it may be analyzed. It must be generated by /SHOW=MAP/LIS. Otherwise one should specify the PPL-source file with the /COMP qualifier.
File name	CMODMAP.COM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	17-JAN-1986

CMSGLIST

CMSG*LIST [directory][module]

PURPOSE Generates script and help file of all GOOSY messages

ARGUMENTS

directory Optional output directory. Default is current. If started in update job, GOO$SCRATCH should be specified.

module Optional message file. Is specified, the document. is extracted, written to GOOTREC and the message documentation is done.

Description

FUNCTION Links object GOO$MSG:MMESLIST with all message files of GOO$MSG, which must be in GOOLIB. Generates XGOOMSG.TXT which is PUT to GOOTMES($MESSAGE). XGOOMSG.HLP is inserted in GOOHMES. All document headers are extracted and inserted in GOOTREC. Then a helpset is generated from GOOTREC. RECOVER.HLP is written to GOOHREC and deleted.

File name CMSGLIST.COM

Dataset -

Version 1.02

Author H.G.Essel

Last Update 3-SEP-1985

05-SEP-85	A script documentation of RECOVER is no longer generated /HE Each message file generates only one text and help module named GOO<fac>. Help is updated only for this facility.
28-oct-85	Created GOOTMES for Message doc.text.

CMTBACK

CMTBACK output

PURPOSE Perform an image system backup for all disk devices on a magtape without verification.

ARGUMENTS

output Magtape MTA0:, MUA0:, or HSC000$MUA0:

DESCRIPTION

CALLING CMTBACK output

ARGUMENTS

output Magtape MTA0:, MUA0:, or HSC000$MUA0:

REMARKS BACKUP Utility is used without /VERIFY
Files excluded =
"*.FBK,*.PBK,*.IBK,*.BAK,*.BCK,*.IMB"
Use this command procedure under a system account!

Example -

CNAMELIST

```
CNAMELIST <namelist> <prefix>
          /DIR*ECTORY
          /SHORT
          /LIB*RARY=<library>
          /SEL*ECT=<string>
          /EXC*LUDE=<string>
          /SEA*RCH=<string>
          /SIN*CE=<date>
          /BEF*ORE=<date>
          /OUT*PUT=<file>
```

PURPOSE Defines a set of symbols with names got from <namelist>.

ARGUMENTS

namelist List of items separated by commas, each of which can be one of the folowing:
1 A name specification of a file (wildcards)
/DIRECT must be specified
/SHORT returns filename+type
2 A library module (one asterisk in module name)!
/LIBRARY=lib specifies text library
3 A name (default, if no /LIB and no /DIR)
4 @filename containing a list of names
(One per line, SPACE is delimiter).
For each name a global symbol is created, if no output is specified.

prefix The symbol names are built by:
<prefix>_<number>,
where <number> runs from 1 to <prefix>_0.
Default is "namelist"

/DIRECTORY Items of namelist are passed to DIRECT to get the individual names.

/SHORT	Short form of filename (name + type)
/LIBRARY=	Library used to get the individual names.
/SEARCH=	String to be searched in files.
/SELECT=	String used to select names (string must be in file or module text).
/EXCLUDE=	String used to exclude names (string must NOT be in file or module text).
/SINCE=	Date (used in DIRECTORY search and LIB/LIST)
/BEFORE=	Date (used in DIRECTORY search and LIB/LIST)
/OUTPUT=	Instead of assigning symbols, the names are written to specified output file.

Description

FUNCTION

If the first character of a namelist item is a @, then names are read line by line from specified file. Otherwise the item is passed to get a directory list or a library list, depending on specification of /LIB=<library> or /DIR. If neither switch is specified, namelist item is the name itself. The selection switches /SELECT and /EXCLUDE work on the content of the datasets rather than the names. The strings must be enclosed in "" if they contain characters recognized by MDCLLIST. Global symbols are set with the names of the name lists or the names are written to output file, if specified.

The symbol names are built by:

<prefix>_<number>,

where <number> runs from 1 to <prefix>_0.

You can delete the symbols by command

CDELSYM <prefix> 0 <prefix>_0

NOTE

Filenames are fully qualified unless /SHORT is specified.

Examples

CNAME T*.COM,@DIR,X*.PPL ABC /DIR

Generates symbols:

ABC_1 , ABC_2 ... ABC_n, where n = ABC_0

Each symbol is a file name or name in DIR.DIS

CNAME EMMA,DONALD NODE

Generates symbols:

NODE_1 , NODE_2 , where NODE_0 is 2
  CNAME T*,@LIST,X*Y GOO /LIB=GOOLIB
Generates symbols:
GOO_1 , GOO_2 ... GOO_n, where n = GOO_0
Each symbol is a module name of GOOLIB or LIST.DIS
  CNAME * FOO /LIB=GOOLIB /SINCE=12-JUL-1985
Generates symbols:
FOO_1 , FOO_2 ... FOO_n, where n = FOO_0
Each symbol is a module name of GOOLIB inserted
before 12-JUL-1985.
  CNAME GOO$UTIL:*.PPL GOO /DIR /SEL=SYS$
Generates symbols:
GOO_1 , GOO_2 ... GOO_n, where n = GOO_0
Each symbol is the name of a file containing string
SYS$.
may be used as follows:

```
$ CNAME GOO$UTIL:*.PPL GOO /SEL="SYS$"
$ loop_count = 1
$ loop_max = GOO_0
$ G_loop_start:
$ If loop_count .GT. loop_max THEN GOTO G_loop_end
$ name = GOO_'loop_count' ! is name of item
$ loop_count = loop_count+1
$ GOTO G_loop_start
$ G_loop_end:
```

CNOTICE

CNOTICE recipient/NOSELF
"subject" "line 1" ... "line 6"

PURPOSE Send a short notice with VMSmail

ARGUMENTS

recipient Specification of the mail recipient(s)

/NOSELF Inhibits sending of a carbon copy to sender.

subject Mail subject

line n Up to 6 lines with a short notice

Description

CALLING

CNOTICE recipient/NOSELF
"subject" "line 1" ... "line 6"

ARGUMENTS

recipient Specification of the mail recipient(s), either a single user name, a list of user names separated by commas or a distribution list (@filename). For a full discusion look in the description of the CMAIL command.

/NOSELF Inhibits sending of a carbon copy to sender. NOTE, that this qualifier must be entered immediately after the recipient parameter without any intermediate blanks or tabs !

subject Mail subject.

line n Up to 6 lines which make up the short notice. Each line may contain either text or a file reference of the format:
@file_name[/DELETE]

The text found in the file is included in the notice. The file is deleted afterwards if the /DELETE qualifier has been specified. If no line is specified in an INTERACTIVE session, the Editor is invoked to enter your message in a predefined form.

FUNCTION The procedure builds a temporary file from the input lines specified and calls CMAIL to deliver this the the recipients.

REMARKS -

EXAMPLE

```
CNOTICE MUELLER "Seminar schedule" -
    "The new seminar schedule can be found in the" -
    "dataset SEMI.TXT, the preliminary version is:" -
    " " -
    "@SEMI.TXT" -
```

COMPILE

COM*PILE file /PRE*QUAL=(list)/Q*UALIFIER=(list)
/G*IPSY=list/LIBRARY=(list)/DEBUG/KEEP
/OLB=library/SINCE=time/BEFORE=time/NEWPLI
/FAST/MACRO/CALL/BATCH/COMPILE

PURPOSE	General compile procedure for all compilers THE PREVIOUS VERSION CAN BE CALLED BY OCOMPILE !
ARGUMENTS	
file	Default extension from symbol defcompi. Wildcarding supported.
/PREQUAL	List of precompiler qualifiers
/QUALIFIER	List of compiler qualifiers
list	List of items separated by commas.
/GIPSY	List of GIPSY qualifiers
list	List of items separated by commas.
/LIBRARY	List of private Libraries
list	List of items separated by commas.
/COMPILE	Compile in any case
/DEBUG	Compile with debug switch set
/KEEP	Hold temporary source
/OLB=	Private object library
/SINCE=	Compile only those sources dated later than "time"
/BEFORE=	Compile only those sources dated ealrier than "time"
time	Specify an absolute time or a combination of absolute and delta time.

/NEWPLI	Use the new PL/I Compiler
/CALL	Compile option for GOOSY Macros: Expand macros to subroutine calls
/MACRO	Compile option for GOOSY Macros: Expand macros to PL/1 code
/FAST	Compile option for GOOSY Macros: Expand macros to fast PL/1 code
/BATCH	Compile in batch job in queue SYS$COMP.
Example	COM ABC /QUAL=(LIST,SHOW=INCLUDE) COM ABC /QUAL=(LIST,SHOW=(INCLUDE,MAP))
Languages	For the following file types the appropriate compilers are called: PPL, PLI, MOD, PAS , DEF, MSG, DAR, FOR, MAR, MAC, C, MBD, COM, TEX, FONT, PGIP, CGIP, DGIP, CPGIP

DESCRIPTION

CALLING

COM*PILE file /PRE*QUAL=(list)/Q*UALIFIER=(list)
/G*IPSY=list/LIBRARY=(list)/DEBUG/KEEP
/OLB=library/SINCE=time/BEFORE=time/NEWPLI
/FAST/MACRO/CALL/BATCH/COMPILE/DIR=d

ARGUMENTS

file	File name, wildcards included, specifies one or more files to be compiled. The default extension is taken from default compiler (your global symbol "defcompi"). With an "*" you get your last input.
/PREQUAL	List of precompiler qualifiers This list will be passed to the precompiler,if used.
/QUALIFIER	List of compiler qualifiers This list will be passed to the called compiler.
list	List of items separated by commas. An item may be a sublist like /QUAL=(list).
/GIPSY	List of GIPSY qualifiers This list will be passed to GIPSY.
list	List of items separated by commas. An item may be a sublist like /GIPSY=(list).

/LIBRARY	List of private Libraries Each item of this list will be added to the file name with the qualifier "/LIB".
list	List of up to three items separated by commas.
/COMPILE	Compile in any case
/DEBUG	Compile with debug switch set, added to /QUAL PLI: /DEB=ALL/CHECK/SHOW=(INCL,MAP,EXPANS, SOURCE,TERMINAL,TRACE)/NOOPTIMIZE FOR: /DEB/NOOPT MAR: /DEB
/KEEP	hold temporary source (output of precompiler) FORTRAN precompiler generates .FORTEMP, PLI precompiler .PLITEMP files.
/OLB=	Private object library
/DIRECTORY=	Set default directory.
/SINCE=	Compile only those sources dated later than "time"
/BEFORE=	Compile only those sources dated ealrier than "time"
time	Specify an absolute time or a combination of absolute and delta time. See section 2.5 in the VAX/VMS DCL dictionary for details on specifying times or access the HELP topic SPECIFY.
/NEWPLI	Use the new PL/I Compiler
/CALL	Compile option for GOOSY Macros: Expand macros to subroutine calls
/MACRO	Compile option for GOOSY Macros: Expand macros to PL/1 code
/FAST	Compile option for GOOSY Macros: Expand macros to fast PL/1 code (No checks of bits, no counter increments).
/BATCH	Compile in batch job in queue SYS$COMP.
FUNCTION	There are three features in this procedure: 1. Compilation only if the source is younger than the object

Depending on the specified file extension, different compilers are called. If /COM is specified, the file is compiled in any case. The prequalifier, enclosed in brackets, are passed to a precompiler, if used. Compiler qualifiers are not specified as normal!!: all qualifiers have no slashes and are seperated by commas e.g. /QUAL=(NOOPT,LIST,NOOBJECT). 2. Use wild cards in file name to compile more than one file. Use /SINCE and/or /BEFORE switch to compile only sources of a specified time intervall. 3. To keep the last input as default If the first parameter (file name) is only an *, the last specified input is used. All switches directly following the the file name are kept as default for the next call. After a blank switches are temporary used.

Examples

COM MGENHEAD.PPL/PRE=(V) /KEEP/COM calls precomposer MPRECOMP with tagword V and then PLI compiler. /KEEP option holds MGENHEAD.PLITEMP. Compilation is done without respect to the dates of source and object file. COM */DEB /COM Compiles again MGENHEAD.PPL/PRE=(V) /DEB/COM COM ABC /QUAL=(LIST,SHOW=(INCLUDE)) RSX-MACRO compilation with macro expension listing COM XYZ.MAC /QUAL=(LIST,MACRO)

Compiler

FORTRAN

input File type .FOR compiled by FORTRAN

input File type .DAR calls preprocessor and generates FORTRAN file .FORTEMP. Then FORTRAN is called.

PLI

input File type .PLI compiled by CPLI

input File type .PPL calls preprocessor and generates PLI file .PLITEMP. Then CPLI is called.

C

input File type .C compiled by CC

GIPSY

input File type .DGIP compiled by GIPSY to file .COM

input File type .PGIP calls preprocessor MPRECOM and generates GIPSY file .PGIPTEMP. Then GIPSY generates .PLITEMP. Then CPLI is called.

input File type .CGIP compiled by GIPSY to file .CGIPTEMP. Then CC is called.

input File type .CPGIP compiled by GIPSY to file .CPGIPTEMP. Then CP020 is called.

MODULA

input File type .MOD compiled by MODULA

MESSAGE

input File type .MSG is processed by preprocessor MPREMES which generates file .MSGTEMP. Then MESSAGE is called.

MACRO

input File type .MAC compiled by CRSXMAC for RSX

input File type .MAR compiled by VAX MACRO

PASCAL

input File type .PAS compiled by PASCAL

MBD

input File type .MBD compiles by CMBDCOM

LATEX

input File type .TEX compiled by CTEXCOM

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

COPTLIST

COPTLIST list keylist

PURPOSE Parse IBM styled optional parameter list for DCL procedures.

ARGUMENTS

list List of optional parameters as input to calling DCL procedure. The items of the list have the form <key>[(<value>)]. Each key must be preceded by a —. No spaces are allowed.

keylist List of allowed keys. The keys have the form <key>[()], where an asterisk in <key> marks abbreviation. If the brackets are specified, the according input key must have the value enclosed in brackets as well. The keys are separated by blanks.

DESCRIPTION

FUNCTION This procedure is called by DCL command procedures to parse optional parameter lists as for IBM clists. The items of the list as keyed in have the form:

<key>
<key>(<value>)

separated by spaces.
Each valid key of the calling procedure must be assigned a global symbol name. These names must begin with two underscores to avoid reassignments. At the end of the procedure these symbols must be deleted by DEL/SYM/GL <name>.

EXAMPLE The calling procedure may have the following argument list:

pos1 pos2 pos3 XXX YYY(<value>)
Then the first statements should be:

```
$ name1 = p1
$ name2 = p2
$ name3 = p3
$ __XXX:==""
```

```
$ _YYY:=="default"
$ COPTLIST —'P4'—'P5' X*XX Y*YY()
$ .....
$ FINISH:
$ DELETE/SYMBOL/GLOBAL _XXX
$ DELETE/SYMBOL/GLOBAL _YYY
$ EXIT
```

After this call, the symbol __XXX has the value "XXX" if specified or "" if not. Symbol _YYY has the value "<value>" if specified or the value "default" if not. Example:

@XYZ 1 2 3 X results in __XXX="XXX", __YYY="def."

@XYZ 1 2 3 Y(ABC) results in _XXX="", _YYY="ABC"

The value for YYY is prompted if YYY was specified without value.

File name COPTLIST.COM

Dataset -

Version 1.01

Author H.G.Essel

Last Update 30-MAR-1984

CPLICOM

PPL_file/switches /switches %DEB %NEWPLI

PURPOSE	Compile PLI programs including certain text libraries. Defaults are updated.
ARGUMENTS	
file	Name of PLI source file (def.=last)
switches	PLI switches. Behind SPACE are not updated
%DEB	/DEB=ALL/CHECK/SHO=(INC,MAP,EX,SO,TER,TRA)/LIS/NOOPT
%NEWPLI	Selects the new PL/I Compiler

DESCRIPTION

FUNCTION

Two problems are solved:

1. To keep last input as default
2. To inculde text libraries.

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 *.

Example

```
CP ABC.PLI/X /Y == PLI ABC.PLI/X/Y
  CP == PLI ABC.PLI/X
  CP */Z == PLI ABC.PLI/X/Z
```

REMARKS	-
File name	CPLICOM.COM
Dataset	-
Version	1.01
Author	H.Essel

Last Update 20-JAN-1984

Last Update 05-NOV-1985
Added %DEB switch. H.E.
GKS$ROOT:[LIB]GKSINC/LIB included M.R.
GKS$ROOT:[LIB]GKSINC/LIB removed H.E.
GOOTYP/LIB included H.E.
Text libraries now as log. names PLI$LIBRARY_n G.S.
13-mar-1986:
PLI switches behind PLI command /HE
Translate text libraries PLI$LIBRARY_01,02,03 first /HE
Translate text libraries from job table /HE

CPRECOM

CPRE*COM file(switches)

PURPOSE	Call precompiler
ARGUMENTS	
file	File containing preprocessor code (PPL)
(switches)	preprocessor tagwords

Description

FUNCTION	The precompiler switches are specified in parentheses behind the file name. The PLI output is written to a file with type TMP.
File name	CPRECOM.COM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	28-JAN-1985

CPRINT

CPRINT file_list /DESTINATION=d

PURPOSE Print a file on a spooled printer.

ARGUMENTS

file_list List of files to be printed.

/DESTINATION=d Specifies printer to be used.
Currently: VAX, MS or KP

Description

CALLING CPRINT file_list /DESTINATION=d

ARGUMENTS

file_list List of files to be printed. A list of files specifiers, separated by commas, may be specified. Each list item may contain any valid wild cards. The default file name is the preceeding list item, for the first item ".LIS" is taken.

/DESTINATION=d This qualifier defines the printer destination. If not specified, the symbol

VAX_DEST

is evaluated and taken as default. If the symbol is not defined, "VAX" will be the default.
Currently available destinations are:

VAX	Printer in VAX/780 room
KP	Terminal Printer in KP devision
MS	Printer in 'MESSTATION'.

FUNCTION

The destination qualifier is evaluated according to the mentioned default rules. Than a logical name

SPOOL_LP_<destination>

is evaluted. The equivalence string contains the node and physical device name of the spooled device.

All specified files are than copied (one by one) to the spooled device. This ensures that all files are printed with a meaningfull banner page.

NOTE

The files are not QUEUED for printing like with the PRINT command but SPOOLED. This means that a temporary copy of the file is created. Therefore the orgininal files may be deleted immediately after the completion of the CPRINT command.

REMARKS

The logical spool device names may be used in the form

SPOOL_LP_<destination>:[<name>.<type>]

in any context requiring an output file specifier.

EXAMPLE

CPRINT [MAIER.TEST]TEST*.COM,DUMMY.PPL /DEST=VAX
Will print the files:

[MAIER.TEST]TEST*.COM
[MAIER.TEST]DUMMY.PPL

on the line printer "VAX".

CPURGE

CPU*RGE filespec [purge qual.]

PURPOSE	Does a secure purging
ARGUMENTS	
filespec	File specification passed to DIRECT and PURGE
purge qual.	Purge qualifier

Description

FUNCTION	We look for all empty files under filespec. If an empty file is found with lower versions, i.e. the empty file would be the only one after the purge, this empty file is prompted to delete. After these prompts a confirm is necessary to really PURGE filespec. PURGE qualifier may be added as P2. THe empty file search, however, is done always for filespec.
NOTE	If no "dangerous" empty file is found, the PURGE is done without confirm. For dangerous empty files lower versions exist.
File name	CPURGE.COM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	13-NOV-1985

CREDB

CREDB basename filename size[KB]
/DYNLISTS=d /SPECTRA=s /CONDITIONS=c
/PICTURES=p /DIRECTORIES=d /POOLS=p
/POLYGONS=p /NEW /SAVE=file

PURPOSE Create an preformat a new GOOSY data base.

ARGUMENTS

basename Name of data base.

filename File specification for data base file. The name must be equal the data base name, the file type should be .SEC.

size Size of data base in Kilobytes. (1 VMS page is .5KB)

/DYNLISTS=d Maximum number of dynamic lists. (def=10)

/SPECTRA=s Maximum number of spectra (def=100)

/CONDITIONS=c Maximum number of conditions (def=100)

/PICTURE=p Maximum number of pictures and frames (def=100)
Each frame takes one entry!

/DIRECTORIES=d Maximum number of directories (def=20)

/POOLS=p Maximum number of pools (def=20)

/POLYGONS=p Maximum number of polygons (def=20)

/NEW An old existing data base is deleted and a new one is created. If not specified, an error is given.

/SAVE=file Save command procedure to create data base in file.

Description

CALLING CREDB basename filename size[KB]
/DYNLISTS=d /SPECTRA=s /CONDITIONS=c
/PICTURES=p /DIRECTORIES=d /POOLS=p
/POLYGONS=p /NEW /SAVE=file

ARGUMENTS

basename Name of data base.

filename File specification for data base file. The name must be equal the data base name, the file type should be .SEC.

size Size of data base in Kilobytes. (1 VMS page is .5KB)

/DYNLISTS=d Maximum number of dynamic lists. (def=10)

/SPECTRA=s Maximum number of spectra (def=100). The bit table for spectra is created with this number of entries.

/CONDITIONS=c Maximum number of conditions (def=100) The bit table for conditions is created with this number of entries.

/PICTURE=p Maximum number of pictures and frames (def=100)
Each frame takes one entry!

/DIRECTORIES=d Maximum number of directories (def=20)

/POOLS=p Maximum number of pools (def=20)

/POLYGONS=p Maximum number of polygons (def=20)

/NEW An old existing data base is deleted and a new one is created. If not specified, an error is given.

/SAVE=file Save command procedure to create data base in file. If the base already exists, the command procedure is written, but not executed.

FUNCTION A new GOOSY data base is created. The default directories for spectra, conditions, pictures and dynamic lists are created.
A directory and pool for user data elements is created, both named DATA. The directory has 100 slots.

REMARKS The data base file must not exist (except /NEW or /SAVE=file is specified).

EXAMPLE

```
CREDB DB DB.SEC 5000 /SPEC=200
  CREDB DB DB.SEC 5000 /SPEC=200/SAVE=DB
    save command procedure in DB.COM. If DB.SEC exists,
    the command procedure is not executed.
```

CREMNUM

CREM input output col check

PURPOSE Removes line numbers (<col> char.) from source files checking first <check> characters to be digits.

ARGUMENTS

input	name of input file
output	name of output file
col	number of columns tru(def.=8)
check	number of characters to be tested to be digits (def.=8)

DESCRIPTION

File name	CREMNUM.COM
Dataset	(CREMNUM)
Version	1.01
Author	H.G.Essel
Last Update	15-JAN-1984
REMARKS	-

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
```

CREPEAT

CREP*EAT dcl_line

PURPOSE Repeats a dcl command continuously, beginning on screen top.

ARGUMENTS

dcl_line DCL line to be executed.

Description

FUNCTION -

File name CREPEAT.COM

Dataset -

Version 1.01

Author H.G.Essel

Last Update 7-FEB-1985

CREPLACE

CREPL*ACE file old new /U*PPER/P*RINT/F*ORMAT/OUT=file

PURPOSE	Replaces old string by new calling MREPLACE. Documentation headers will be adjusted optionally.
ARGUMENTS	
file	Input file specification (name list with wild cards or @file with file names).
old	String to be replaced.
new	String to be inserted.
/OUT=file	specifies output file (default=input).
/U*PPER	Input lines and oldstring are uppercased before comparison.
/P*RINT	Modified lines and their sources are printed to SYS$OUTPUT.
/F*ORMAT	Try to adjust new lines inside header blocks

Description

FUNCTION	Calls MREPLACE for each file. Replaces first string by second. Tries to adjust header lines. If not possible, output warning message.
File name	CREPLACE.COM
Dataset	-
Version	1.01
Author	H.G.essel
Last Update	17-OCT-1985

CSYMDIR

@GOO$EXE:CSYMDIR

PURPOSE Create symbols for SET DEFAULT from the directory tree

ARGUMENTS

Description

FUNCTION The procedure scans the whole directory tree of the calling user. It creates a symbol for each directory and subdirectory to perform a SET DEFAULT DCL command. Each symbol is composed of a preceding 'Y' and then the name of the deepest subdirectory, e.g.:
for the Directory: US1$ROOT:[USER.TEX.LETTER]
the symbol: YLETTER will be generated.
The symbol YLOGIN sets the default back to the main directory.
This procedure may be called in the user's LOGIN.COM.

Version 1.01

Author J. Hoffmann

Last Update 30-SEP-1987

CSYMHLP

CSYM input output

PURPOSE Calls MSYMHLP to generate help files from command procedures.

ARGUMENTS

input file name of input

output file name of output

DESCRIPTION

File name CSYMHLP.COM

Dataset -

Version 1.02

Author H.G.Essel

Last Update 10-JAN-1984

REMARKS See MSYMHLP.

CTEXCOM

CTEXCOM file /NOLIST/NOLATEX/PASSES=n/DVI=drv/DELETE=l

PURPOSE	'Compile' TEX source using PLAIN TeX ot LaTeX
ARGUMENTS	
file	File name.
/NOLIST	Delete Listing
/NOLATEX	Use PLAIN TeX
/PASSES=n	Call TeX n times.
/DVI=drv	Call DVI driver for device type <drv>
/DELETE=l	Delete list of files <l> created during processing

Description

FUNCTION	'Compile' TeX source using eithaer PLAIN TeX or LaTeX
Version	1.01
Author	H. Grein
Last Update	24-APR-1987

CTEXMANUAL

CTEXMANUAL manual /REF*ORMAT
/REL*EASE=r/VER*SION=v/BR*IEF

PURPOSE Create GOOSY manual

ARGUMENTS

manual list of requested manuals, separated by commas:
OVERV*IEW (Programs, Modules, Commands, Macros)
VMS*PRIM
COM*MAND
PPL*MAC
PRO*GRAM
UTI*LITY
REC*OVER

/REFORMAT call GLFORMAT before to create the TeX source

/RELEASE=r specify current GOOSY release number

/VERSION=v specify current GOOSY version number

/BRIEF Use /BRIEF qualifier for GLFORMAT. Updates, Internals and Implementation blocks are always excluded.

Description

FUNCTION Create GOOSY manuals formatted by the TeX typesetting system.
Output is written to GOOSY_manual.DVI and may
be printed by the DVIPRINT command.

Version 1.01

Author H. Grein

Last Update 9-JAN-1987

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

CVTISOL

CVTISOL <input file>,<output file>[/SHORT]

PURPOSE	Convert ISOLDE spectra into SATAN readable format.
ARGUMENTS	
input file	File name of ISOLDE spectrum
output file	File name of SATAN compatible spectrum
/SHORT	outputs compressed data

Description

FUNCTION	Reads ISOLDE formatted spectrum from input file and outputs an ASCII file (80 bytes per record). First record contains header information, each following 8 longwords in ASCII in noncompressed mode, or as many as possible in compress mode
Version	1.01
Author	H.G.Essel
Last Update	24-SEP-1986

CWHAT

CWHAT options arguments

PURPOSE Activates the WHAT Utility to display details about the VMS system status.

FORMAT
CWHAT ACCOUNT
CWHAT DEVICE
CWHAT LINKS
CWHAT LOCKS
CWHAT MAILBOX
CWHAT OBJECTS
CWHAT PROCESS
CWHAT TERMINAL

REMARKS -

ACCOUNT

CALLING CWHAT ACCOUNT /IO_QUOTA /PROCESS_QUOTA /PRIVILEGES

PURPOSE Show parameters of accounts.

ARGUMENTS

/IO_QUOTA Show IO_QUOTA parameters

/PROCESS_QUOTA Show process quotas

/PRIVILEGES Show accounts privileges

FUNCTION Displays account parameters.

REMARKS Activates the SHOW ACCOUNT option of WHAT.

EXAMPLE CWHAT ACC /IO

DEVICE

CALLING CWHAT DEVICE

PURPOSE Show parameters of devices.

ARGUMENTS -

FUNCTION Displays device parameters.

REMARKS Activates the SHOW DEVICE option of WHAT.

EXAMPLE CWHAT DEV

LINKS

CALLING CWHAT LINKS

PURPOSE Show parameters of active DECnet links.

ARGUMENTS -

FUNCTION Displays network link parameters.

REMARKS Activates the SHOW LINKS option of WHAT.

EXAMPLE CWHAT LIN

LOCKS

CALLING CWHAT LOCKS

PURPOSE Show user locks

ARGUMENTS -

FUNCTION Displays user lock parameters

REMARKS Activates the SHOW LOCKS option of WHAT.

EXAMPLE CWHAT LO

MAILBOX

CALLING CWHAT MAILBOX

PURPOSE Show parameters of mailboxes.

ARGUMENTS -

FUNCTION Displays mailbox parameters.

REMARKS Activates the SHOW MAILBOX option of WHAT.

EXAMPLE CWHAT MAI

OBJECT

CALLING CWHAT OBJECT

PURPOSE Show parameters of DECnet objects.

ARGUMENTS -

FUNCTION Displays network object parameters.

REMARKS Activates the SHOW OBJECT option of WHAT.

EXAMPLE CWHAT OBJ

PROCESS

CALLING CWHAT PROCESS /IO /PAGES

PURPOSE Show parameters of active processes.

ARGUMENTS

/IO Show IO parameters

/PAGES Show workingset and paging parameters

FUNCTION Displays process parameters.

REMARKS Activates the SHOW PROCESS option of WHAT.

EXAMPLE CWHAT PRO /IO

TERMINAL

CALLING	CWHAT TERMINAL
PURPOSE	Show parameters of active terminals.
ARGUMENTS	-
FUNCTION	Displays terminal parameters.
REMARKS	Activates the SHOW TERMINAL option of WHAT.
EXAMPLE	CWHAT TER

CWV

CWV

PURPOSE inquires a diary, 'Wiedervorlage'

ARGUMENTS -

Description

FUNCTION from the current directory a file WV.TXT is read, in which the user has previously edited his appointments. Starting in column 1 a date is given, if that date matches the current date, the text beginning at column 8 will be written on the terminal. Three types of dates are possible: :OL. :LI. a certain date, as yymmdd , the text will be printed on all following days. :LI. a certain day of the week, a number between 1 and 7 (Sunday=1, ..., Saturday = 7). The text will be printed on that day only. :LI. a certain day of the month, e.g. 13. , a number followed by a dot. The text will be printed on the specified date and on the three following days. :EOL.

An example is given in [WINKELMAN]WVEXAMPLE.TXT .
The file containing the apointments can be interpreted also on the IBM (command WV (like 'Wiedervorlage')in TSO, dataset name has to be WV.TEXT) .

File name GOO$EXE:CWV.COM

Dataset -

Version 1.01

Author K.Winkelmann

Last Update 3-JUL-1985

D0_BACK

@D0_BACK output ALL

PURPOSE Perform an image backup for disk device DUA0: on a magtape without verification.

ARGUMENTS

output Magtape MUA0:, MUA1:, MUA2:, MUA3:

ALL If given as "ALL" NOBACKUP will be ignored

DESCRIPTION

CALLING @D0_BACK output ALL

ARGUMENTS

output Magtape MUA0:, MUA1:, MUA2:, MUA3:

ALL If given as "ALL" NOBACKUP will be ignored

REMARKS BACKUP Utility is used without /VERIFY
Files excluded (if not ALL) =
"*.FBK,*.PBK,*.IBK,*.BAK,*.BCK,*.IMB"
Use this command procedure under a system account!

Example -

DCFIBM

DCFIBM vaxfile "parm list"

PURPOSE Sends VAXfile to IBM DCFTEMP.TEXT and starts batch job calling DCF on IBM

ARGUMENTS

vaxfile VAXFILE name, default type : ".scr" Wild cards are not allowed

parm list DCF parameter list like LASER,DEST(RZ) etc. The parm list has to be enclosed in quotes (").

DESCRIPTION

FUNCTION This command procedure sends a script input file to the IBM dataset DCFTEMP.TEXT and formats it with DCF. Note, that to print it on the IBM LASER PRINTER 3820, the keyword LASER must be given in the parm list. If the keyword LASER isn't specified the destination parameter dest(d) is ignored. If the keywordparameter 'dest(d)' is not specified in the parm list, the value specified by the symbol 'SCR_DEST' will be taken. To define the symbol, set

```
SCR_DEST :== destination
```

valid destinations are:

```
SCR_DEST :== UL
SCR_DEST :== RZ
SCR_DEST :== KP
SCR_DEST :== TP
```

For the IBM the default account number 'IBM_ACC' will be taken. If this is blank, the account number will be prompted. To define a default IBM-account number, set (in your login.com) :

```
IBM_ACC :== account
```

e.g. IBM_ACC :== PR99

The IBM password will be prompted unless it is defined

already by a global symbol, namely IBM_PW. If your document is not printed, look at the Messages in your mail or in file 'DCFTEMP.LIST' on the IBM for errors.

EXAMPLE

DCFIBM HALLO "LASER"

The VAX-file HALLO.SCR will be sent to the IBM sequential dataset DCFTEMP.TEXT and will then be printed on the laser printer "UL" with DCF. For the destination the default 'SCR_DEST' was taken. The messages are returned to the mail on the VAX. DCFIBM VORTRAG.TEXT "LASER DEST(RZ) SIMPLEX"

The VAX-file VORTRAG.TEXT will be sent to the IBM sequential dataset DCFTEMP.TEXT and will then be printed on the laser printer "RZ" only on one side of the paper. Default: duplex (printing on both sides). For more Information about DCFIBM contact P.Kohn,RZ Tel.: 557

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

DOCIBM

DOC vaxfile [dest]PROP TWOPASS

PURPOSE Sends a file to IBM DOCTEMP.TEXT(DUMMY) and starts batch job calling DOC

ARGUMENTS

vaxfile VAX-file containing script source (Default type is .SCR)

dest Laser printer destination, default: "UL"

PROP Proportional character font

TWOPASS TWOPASS DOC option enable

DESCRIPTION

CALLING DOC vaxfile [dest]PROP TWOPASS

ARGUMENTS

vaxfile VAX-file containing script source
name.typ;version (n.t;v)
Wild cards are not allowed in name nor type. The default type is ".SCR".

dest Laser printer destination, default: "UL". Other printers: "RZ", "KP".

(optional)

PROP Proportional character font. Else default font = Prestige Elite

TWOPASS TWOPASS DOC option enable. Default = disabled

FUNCTION This command procedure "DOCIBM.COM" sends a doc input file from the VAX into the IBM PDS DOCTEMP.TEXT(DUMMY), formats it with DOC and prints it on a laser printer, default: "UL"

IBM for errors. For the IBM the default account number 'IBM_ACC' will be taken. If this is blank, the account number will be prompted. To define a default IBM-account number, set (in your LOGIN.COM):

IBM_ACC :== account

e.g. IBM_ACC :== PR99

The password for the IBM will be prompted unless it is already defined by the global symbol IBM_PW.

REMARKS

If your Document ist not printed, look in file 'SCRITEMP.LIST' on the IBM for errors or see in the listing sent to the VAX (new software).

The VAX and the IBM information are parameters for the internal SUB GOO$EXE:BDOCIBM. This batch procedure will send the file(s) from VAX to IBM.

The command BDOCIBM should never be given directly! The BDOCIBM must be executed in the batch queue SYS$IBM, which serializes the access of the link hardware.

If the new software is applicable however, BDOCIBM will be executed as DCL procedure. There will be a LOG-file SYS$LOGIN:BDOCIBM.LOG . These LOG-files will be purged by DOCIBM.

EXAMPLE

DOC HALLO

The VAX-file HALLO.SCR will be sent to the IBM partioned dataset DOCTEMP.TEXT(DUMMY) and will then be printed on the laser printer "UL" with DOC.

DOC ADAM "" PROP

The VAX-file ADAM.SCR will be printed with proportional character font (160, 163).

Document

Desciption of the module documentation tools

Commands The following commands are provided to generate documentation for VMS help utility and printer output (RNO and SCRIPT).

CEDITDOC	Opens a new file and prompts for all information needed to generate a documentation header. The formatted header is written to the file. The editor is called. Supported are main programs and procedures (PLI, FORTRAN, PASCAL, MACRO, DCL, TSO).
CEXTRACT	The documentation header is extracted from a source file (or files) and written to a text file and optional to a text library. Input from several source files may be output to one text file.
CFORMAT	Reads text files or modules generated by CEXTRACT and formats it for VMS help libraries (using DSR), for printer output (DSR) or for SCRIPT. Parts of the input may be selected.
CDOC	Extracts and formats documentation of one module.

Help More information about these commands can be obtained by HELP @UTIL <command>.

DESCRIPTION

The general ideas for the documentation system are:

Source The module documentation is implemented as comment in the modules source file. It is preformatted to provide comfortable reading.

Formatter Because the documentation should not depend on either the VAX or the IBM, the RUNFOFF or the SCRIPT formatter may be used. On both machines, VAX and IBM, the documentation can be done.

Modularity The documentation block is devided into named subblocks. These names are used as headlines in the printer output and as subkeywords for VMS-help. A specified subset of subblocks may be selected or excluded for the printer output. This allows to produce manuals for users or programmers with different verbosity.

Flexibility Because the input for the formatters (DSR and SCRIPT) is generated by a program (MFORMDO), other formatters can be implemented. Similar, SCRIPT statements in the documentation text can be emulated by DSR statements.

SYNTAX

The syntax of the documentation blocks in the module source file must follow these rules (we recommend fix length 72 character lines):

Lines Any line has the form
<delim.(2 char.)><text(68 char.)><delim.(2 char.)>
The delimiters depend on the type of the source code
(e.g. for PLI /* and */).

Main block The named main block starts with
<delim.>1+ <type><***><delim.>
where type specifies the module type (e.g. PLI Procedure).
<*> is used as delimiter and filler character. The next line must be the module specification:
<delim.>+ Module<6 sp>: <name><sp><delim.>
where <name> is the name of the block (the module name).
A main block is terminated by a line
<delim.>1- <type><***><delim.>

Subblocks Subblocks are opened by a line
<delim.>2+ <name><***><delim.>
and closed by another subblock or by end of block.
Similar, sub-subblocks are opened by
<delim.>3+ <name><***><delim.>

Lists Definition lists are opened by a line
<delim.>+ <item (>11 char.)>: <text><delim.>

Following text lines must begin at column 19. If they start with <space>'s,
they will not be formatted. Definition sublists are opened by a line
<delim.>-<15 sp><item (>10 char.)>: <text><delim.>
Following text lines must begin at column 31. If they start with <space>'s,
they will not be formatted. Lists and sublists are closed by a subblock (may be with empty name). A sublist is closed by a line
<delim.>+ :<sp><delim.>

Includes Subblocks from another file may be included by a line
<delim.>0+ <filename><***><delim.>

TEXTFILE

It is possible to write the textfile directly using any editors. This is useful for documentation modules which do not concern single program modules. In this case, it makes no sense to generate a module header in the above described format. The syntax of the text files is different from the header syntax, because it must be more compact and free formatted. Especially the lines may have any length and no delimiters are needed. NOTE: All text lines except the following start at column 2:

Main block The named main block is opened by (starting at column 1)
1+ <type>
where type specifies the module type (e.g. PLI Procedure).
The next line must be the module specification:
+ Module<6 sp>: <name>
where <name> is the name of the block (the module name).
A main block is terminated by a line
1- <type>

Subblocks Subblocks are opened by a line
2+<name>
and closed by another subblock or by end of block.
Similar, sub-subblocks are opened by
3+<name>

Lists Definition lists are opened by a line
+ <item (>11 char.)>:
Following text lines must begin at column 2. If they start with <space>'s,
they will not be formatted.
NOTE: At least one text line must follow.

Definition sublists are opened by a line
  - <item (>10 char.)>:
  Following text lines must begin at column 2. If they start with <space>'s,
they will not be formatted.
  NOTE: At least one text line must follow.
Lists and sublists are closed by a subblock (may be with empty name).
A sublist is closed by a line
  + :

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

DVIIBM

DVIIBM dvifile switches

PURPOSE Sends DVI-file to IBM. via batch job

ARGUMENTS

vaxfile VAX file name, default type : ".DVI"

switches /DEVICE=ibm-grapg-device /TONAME=ibm-filename /MAXPAGES=no /FROMPAGE=no /ACCOUNT=ibm-acc /DELETE (see HELP @UTIL DVIIBM switches)

DESCRIPTION

CALLING DVIIBM vaxfile switches

ARGUMENTS

vaxfile VAX-File-Specification :
name.typ;version (n.t;v)
Wild cards are not allowed

switches /DEVICE=ibm-grapg-device /TONAME=ibm-filename /MAXPAGES=no /FROMPAGE=no /ACCOUNT=ibm-acc /DELETE (see HELP @UTIL DVIIBM switches)

FUNCTION This command procedure sends one TeX-DVI-file to an IBM DVI-file and optionally to an IBM raster plotter. For the IBM the default account number 'IBM_ACC' will be taken. If this is blank, the account number will be prompted. To define a default IBM-account number, set (in your LOGIN.COM):

IBM_ACC :== account

e.g. IBM_ACC :== PR01

The password for the IBM will be prompted unless it is already defined by the global symbol IBM_PW. If no filename with the /TOFILE switch is entered, the default name uid.VAX.DVI is used.

REMARKS - There will be a LOG-file SYS$LOGIN:BDVIIBM.LOG . These LOG-files will be purged by DVIIBM.

EXAMPLE

Switches

/DEVICE=d The TeX output is sent to the specified device d. Valid values are RP01 or RP02 (raster plotter), LASER1 (Computer center) or LASER2 (Theory section) (Laser Printer IBM 3820). If omitted, the value of the global symbol TEX_DEST is used, e. g., if you include in your SYS$LOGIN:LOGIN.COM file:

```
$ TEX_DEST :== LASER2
```

/TONAME=n The VAX-DVI-file is copied to the dataset uid.n.DVI. (Default n=VAX) For devices LASER1 or LASER2, DVI3820 is forced.

/MAXPAGES=m Maximum number of pages (Default=1). Ignored, if no gaphic output was selected by the /DEVICE switch

/FROMPAGE=p First page no to be drawn. p=number selects a page page number. p=* selects the first page of the text.

/ACCOUNT=a Overrides the ibm-account in the symbol IBM_ACC

/DELETE The VAX-DVI file is deleted after sendig to the IBM.

DVIPRI

DVIPRI
file(s) /DEVICE=dev /STARTPAGE=spage
/PAGES=npage /DELETE /SEP*ARATE/[NO]PR*INT

PURPOSE	Print TeX's device independant (DVI) file
ARGUMENTS	
file(s)	file specification (opt. wildcarded)
/DEVICE=dev	specification of output-device
/STARTPAGE=spage	page number to start printout
/PAGES=npage	number of pages to be printed
/DELETE	if specified, DVI files and any temp. files will be deleted
/SEPARATE	print odd and even pages separately
/[NO] PRINT	For /SEPARATE /NOPRINT is default, otherwise /PRINT

Description

CALLING	DVIPRI file(s) /DEVICE=dev /STARTPAGE=spage /PAGES=npage /DELETE /SEP*ARATE/[NO]PR*INT
ARGUMENTS	
file(s)	file specification, optional wildcarded Default: TEXPUT.DVI
/DEVICE=dev	output device specification. Default: value of global symbol TEX_OUTPUT. Valid devices are: LN03_A laserprinter LN03 in the EE departement LN03_B laserprinter LN03 in the KP3 departement LN03_C laserprinter LN03 in the Messtation

LN03_D laserprinter LN03 in the AP departement
LN03_E laserprinter LN03 in the KP2 departement
LN03_F laserprinter LN03 in the accelerator departement
LN03_G laserprinter LN03 in SHIP hut

/STARTPAGE=spage page number, where printout will start

/PAGES=npage number of pages to be printed, default: 199.

/DELETE if specified, the input file(s) and all temporary files will be deleted

/SEPARATE print odd and even pages separately. This is useful, if you want to copy the formatted output on an automatic copy machine.

/[NO] PRINT For /SEPARATE /NOPRINT is default, otherwise /PRINT. This option can be used to generate two LN3 files which can be printed separately on a printer using the PATEX command. The file types are LN3_EVEN and LN3_ODD. The printed even pages can be feed to the printer printing the odd pages on the back side. Think about page orders! This switch is not used for IBM output.

FUNCTION The specified files will be processed by the device specific driver procedure and send to the specified device. If the global symbol TEX_OUTPUT is not defined, DVIPRI prompts for the device name

REMARKS If an IBM device is specified, the file is sent to the IBM unsing the command DVIIBM. Try HELP DVIIBM to get more inforamtion.

EXAMPLE DVIPRI PHD* /DEV=LN03_A/DELETE

ECLINE

ECL*INE dcl-line /LIS*T /CONF*IRM /NOASS*IGN
/$$1=list /$$2=list
/1LIB*RARY=lib /2LIB*RARY=lib
/1SIN*CE=date /2SIN*CE=date
/1SH*ORT /2SH*ORT
/1DIR*ECT /2DIR*ECT
/1SEA*RCH=list /2SEA*RCH=list

PURPOSE Execute dcl line with dummies replaced from list

ARGUMENTS

dcl_line DCL line. Place holders are $$1 and $$2. These place holders are replaced by items from list 1 and list 2, respectively. The DCL commands are then executed. A $$1 must be present before $$2, if specified! Quotes inside dcl_line must be doubled. Dcl_line must be qouted. Two $$1 may be replaced. Both must occur before a $$2, if any.

/LIST The DCL lines generated are NOT executed.

/NOLIST The DCL lines generated are not listed.

/CONFIRM Each DCL line generated is confirmed before execution.

/NOASSIGN Some DCL procedures do not run, if a
ASS/USER SYS$COMMAND SYS$INPUT
is given. E.g. if the CREATE command is used.

/$$1= List specification passed to CNAMELIST. Can be list of file names (wildcards), list of library module names (wildcards) or container files (@filename) or just a namelist.

/1LIBRARY= specifies library, if list is module name list

/1DIRECT specifies that list is file list

/1SHORT	For file lists file names are in the form: <name>.<type> otherwise full filespec.
/1SINCE=	Date specification for list 1
/1SEARCH=	Search list. /$$1 specifies files. Only files matching containing the search list are selected.
/2....	same switches for list 2

Description

FUNCTION

The lists specified by /$$1= and /$$2= are passed to CNAMELIST together with library and date switches. Any combination of items from list 1 and list 2 are inserted in the DCL command line and the line is executed.

NOTE

If not otherwise specified, SYS$INPUT is define to SYS$COMMAND.

EXAMPLES

```
ECL "glput $$1 goo$exe" /$$1="*.com" /1DIR/1SHORT
   executes:
$  glput ABC.COM goo$exe
            ...
ECL "glset attr goo$exe:$$1 $$2" -
/$$1="*.com" /1DIR/1SHORT -
/$$2="DOC/mhlb=goohinc/mtlb=gootinc,UPD/mark=DOC"
   executes:
$  glset attr goo$exe:ABC.COM doc/mhlb=goohinc -
   /mtlb=gootinc
$  glset attr goo$exe:ABC.COM upd/mark=doc
            ...
ECL "glput $$1.TXT GOOINC($$1)" /$$1=@inc.lis
   executes (ABC and XYZ are in inc.lis):
$  glput ABC.TXT GOOINC(ABC)
$  glput XYZ.TXT GOOINC(XYZ)
            ...
   Note that we need a container file here to get
   the file names without type !
```

File name ECLINE.COM

Dataset -

Version 1.01

Author H.G.Essel

Last Update 1-AUG-1985
2-dec-1985 Double quoting in WRITE and INQUIRE /HE.
11-dec-1986 Second $$1 allowed /HE 14-feb-1989 /NOLOG qual added /HE 10-jun-1991 /SEARCH qualifier added /HE

EDDT

EDDT file /READ/PROFILE=prof

PURPOSE Call full screen edit with a profile depending on the file type.

ARGUMENTS

file File to be edited.

/READ Signals that the file is to be read only.

/PROFILE=p Profile dataset to be used.

DESCRIPTION

CALLING EDDT file

ARGUMENTS

file File to be edited, default type is .PPL. The file name is stored and used as default for a subsequent call.

/READ Signals that the file is to be read only. The given file must exist, a pending edit recovery will be ignored.

/PROFILE=p Name of the profile dataset used for initialization of EDIT/EDT. The default profile depends on the file type, for details of the default profile search refer to the the function description. For detail about the profiles look in section PROFILE.

FUNCTION First is determined by searching a JOURNAL file whether a previous EDIT session terminated abnormaly. In this case is prompted if a recovery (EDIT/RECOVER) is wanted.
If no profile is specified, the edit profile is selected depending on the file type. A file with the name

GOOEDTtyp.EDT

is searched on

the current default directory
SYS$LOGIN:
GOO$LIB:

If this fails, a file with the name
GOOEDT.EDT
which is searched in the same directories mentioned above.
The profile file is used as an initial edit command file and specified in the /COMMAND= qualifier in the call of EDIT.

Example ED TEST

PROFILES

Aim The profile mechanism implemented in the EDDT command allows to modify the default EDIT characteristics depending on the file type edited.

Keys The default profiles redefine two in general unused keys of the standart keypad:

<7> The key 7 with the standart definition PAGE is redefined for PPL and PLI files to
SENTENCE

with a sentence delimiter of ";". This allows to handle PL/I statements.

<3> The key 3 with the standart definition CHAR is redefined for all files to
CUT + PASTE

This allows to bring a select range into the PASTE buffer without deleting it in the current edit buffer.

Error_Handling

Error and Message Handling, User's Guide Vers. 1.05

HELP This documentation is available on-line, say 'HELP ERROR_HANDLING'.

General

Any error is associated to an error code (a BIN FIXED(31) number) having a name (error symbol, error id) which is related to a message text. Error codes with their messages have to be defined before they can be used. The error codes are unique systemwide. An error is called reported, if the routine in which the error occurs, returns the error code as a function value. It is called signaled, if only ON-blocks can handle the error. The signalling can be done by the operating system or by the user. The error and message handling subsystem fulfills following requirements: :ul. :li. define errors via an error code :li. specify message text including variable parameters :li. report errors to the calling routine by returning the error code :li. catch unforeseen or signaled errors by general ON-units :li. provide an interface to specific clean-up routines :li. build up a message stack, do proper parameter replacements in message texts :li. handle the message stack (e.g. output, clear) :eul. The message handling routines are AST-reentrant. For performance reasons relate the error code to a message (use of the routines S$M... and the macros ..._MSG) only if it is likely that the message will be printed.

Usage

:ul. :li. All errors have to be declared in the procedure by the macro @DCL_MSG. The names must be unique for one facility, all start with the letter 'X'. :li. Mention all error codes in the procedure comment header. :li. Before using them, define the errors by the command CMES <facility>. CMES calls the editor for the facility message file. You can add your error codes and message texts. For more details say HELP CMES. :li. Use the macro @PROCEDURE to define the main entry point of your PL/I program. This will ensure that the name of the main entry name will occur in the error messages. :li. In case of an unsuccessful execution, report the error immediately by adding the corresponding message to the message stack (macro @ADD_MSG), or by combining this report with a return of the error code to the calling procedure (macro @RET_ADD_MSG). :li. The calling routine analyzes the returned error code (if necessary use the S$MS... routines). One may add explanatory messages (macro @ADD_MSG) or erase the previous messages, because a more specific one can be given now (macro @RET_SET_MSG). :li.

Use the standard ON-block @ON_ANY_W(c_u_p) (@ON_ANY_E(c_u_p)) in procedures if you want to make actions undone (free storage allocations, close files ...) in case of a failure. These ON-blocks will catch all signaled error. The argument of the macro (c_u_p) is the name of a (preferably internal) clean-up procedure without arguments which will be called before the error will be resignaled. However signaled errors of severity WARNING (ERROR in case of @ON_ANY_E) will be converted to reported errors. :li. The main program should contain an ON ANYCONDITION unit which resignals the error to the operating system, e.g.:

```
ON ANYCONDITION BEGIN;
  @DMP_CLR_MSG;
  CALL RESIGNAL;
END;
```

:li. If you write an ON-block to catch specific expectable signaled errors in a procedure (e.g. FIXEDOVERFLOW, ENDFILE, UNDERFLOW), signaled errors of the same kind from lower level subroutines will not be handled (they will be resignaled) if the macro @LOCAL_ERROR is contained at the top of your ON-unit. :li. Note: before any prompt or output is done, make sure that the message stack will be printed (use the macros @PUT_CLR_MSG or @DMP_CLR_MSG). This will keep the synchronicity of the output. :li. You can set the message profile using the routine S$MSPRO. It allows to suppress the additional message identifier, suppress messages for severities up to a given threshold, or to specify a set of facilities for which always detailed messages will be given. These restrictons will be obeyed if @PUT_CLR_MSG is used, the macro @DMP_CLR_MSG however disregards the profile and will always output the entire stack. :eul. NOTE: Currently still following restrictions have to be obeyed: :ul. :li. If the amount of messages exceeds the stack size (currently 32), they are ignored. :li. Following INCLUDEs are needed for the macros : S$MESS, U$PRTCL, $STSDEF, $CHFDEF. :li. Use capital letters for the macro names and the input of S$MSPRO. :li. Do not use the '@' sign elsewhere, even not in comments. :eul.

Severities

There are five different severities which can be assigned to the errors.

S(uccess) to report successful actions
(e.g. ' -S- spectrum !AS created')

I(nformation) to call attention to some aspects during execution, that might help the user
(e.g. ' -I- condition !AS defined but not referenced').

W(arning) calls attention to a possible error, which can be fully recovered. The command might not completely carried out
(e.g.' -W- spectrum !AS is empty ' after a display command).
Errors of this kind will be converted to reported errors in ON-units of the type @ON_ANY_W.

E(rror) the output or the module result is incorrect. If commands exist of several parts, program tries to continue
(e.g. ' -E- limits of dimension !AS of spectrum !AS not modified, spectrum is !AS dimensional',
' -S- number of associated conditions of spectrum !AS modified' , after a SET command which should modify limits and number of conditions).
Errors of this kind will be converted to reported errors in ON-units of the type @ON_ANY_E.

F(atal) system cannot continue execution, execution is totally aborted, command is discontinued, this error will finally be always signaled
(e.g. '-F- Not priviledged to access the data base !AS') .

Commands

CMES define and implement errors and messages

Macros

@ADD_MSG (ec,arg1,arg2,arg3) : add message on stack, ec is the error code, arg1 to arg3 are parameters which will be passed to the message handler routines to complete the messages. Those arguments for all error macros (arg1...arg3) may be of any type.

@DCL_MSG (ec) : declare an error code

@DMP_CLR_MSG : output the entire message stack without any restrictions to SYS$OUTPUT and the session log file and clear the stack. Use it for test purposes. (This macro calls S$MSSHO).

@LOCAL_ERROR : when put in a local ON-block it will prevent it from catching errors from lower level routines

@ON_ANY_W (clean-up-procedure) : catches all signaled errors, calls the specified clean-up procedure, errors of severity WARNING and lower will be reported.

@ON_ANY_E (clean-up-procedure) : catches all signaled errors, calls the specified clean-up procedure, errors of severity ERROR and lower will be reported.

@PROCEDURE : main entry of procedure, name will appear in error messages from that procedure. Don't use it for internal procedures.

@PUT_CLR_MSG [(b_outdest)] output the messages from the message stack according to the valid profile and clear the stack. 'b_outdest' is a bit string indicating the destination of the output and is passed to the routine U$PRTCL. If not specified U$M_PRTT is used. For more details say HELP U$PRTCL. (This macro calls S$MSPUT).

@RET(ec) : return with the specified error code 'ec'.

@RET_ADD_MSG (ec,arg1,arg2,arg3) : add message and return

@RET_SET_MSG (ec,arg1,arg2,arg3) : clear stack, add message and return

Subroutines

S$MSSET	compose error message from error code, clear stack and store it on stack
S$MSADD	compose error message from error code and add it on stack
S$MSKEY	set name of entry for following entry (useful together with S$MSSYS)
S$MSSYS	add complete message as composed by SYS$PUTMSG on message stack
S$MSREM	remove error message from stack
S$MSGET	return specified message
S$MSPUT	call specified entry to pass specified message to it
S$MSCLR	clean message stack
S$MSNUM	returns number of messages on stack
S$MSSRC	returns number on the stack of the specified message
S$MSSHO	output all messages from stack
S$MSPRO	set message profile

Example

```
ME01: @PROCEDURE OPTIONS(MAIN);
DCL ERR01 ENTRY() RETURNS(BIN FIXED(31)) ;
/* catches all errors, outputs the stack and resignals to VMS */
ON ANYCONDITION BEGIN;
  @DMP_CLR_MSG;
  CALL RESIGNAL;
```

```
END;
/* allows to access different parts of the error code */
%INCLUDE $STSDEF;
/* includes declarations for the message handling routines */
%INCLUDE S$MESS;
STS$VALUE = ERR01();
IF ^ STS$SUCCESS THEN @DMP_CLR_MSG;
/* if not successful (severity is W,E or F), */
/* show all messages which have been collected up to now */
/* and clear message stack */
STOP ;
END;
```

```
ERR01: @PROCEDURE RETURNS(BIN FIXED(31)) ;
/**/
DCL U$ABC ENTRY EXTERNAL RETURNS(BIN FIXED(31)) ;
@DCL_MSG(XTEST_IEQNULL);
%INCLUDE DCL_PROC; /* standard INCLUDE */
%INCLUDE S$MESS ; /* declaration of message access routines */
/**/
 ...
/* the following ON-unit will call U_CLEANUP, */
/* will add a message on the stack and */
/* will convert any error of severity W in a reported error */
@ON_ANY_W(U_CLEANUP);
 ...
ALLOCATE A ;
 ...
STS$VALUE=U$ABC();
 ...
/*return a previously defined error code */
IF I=0 THEN @RET_ADD(XTEST_IEQNULL);
ELSE @RET(1); /* successful return */
/**/
/* internal clean-up procedure */
U_CLEANUP: PROCEDURE ;
  @ON_CLEANUP_ERROR;
  IF ALLOCATION(A)^ =0 THEN FREE A;
  ...
  RETURN;
END U_CLEANUP ;
END ERR01 ;
```

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

EXTRCALL

EXTRCALL libfile

PURPOSE Calls MCALLSYS or MCALLRTL to extract procedure call statments from a listing LIB /EXTRACT from a library

ARGUMENTS

libfile Input file

Description

FUNCTION The command procedure calls program MCALLSYS for a listing from PLISTARLET, containing DECLARE statements of the System Services or calls program MCALLRTL for a listing of GOORTL, containing DECLARE statements of Runtime Library routines. This listings must be done with LIB /EXTRACT. For each DECLARE-statement a CALL statement is extracted and written to module-name.INC which can be later put to GOOCALL.

Version 1.01

Author G. Schneider

Last Update 24-JUN-1986

GIPSY

GIPSY input output /LIST/DELETE

PURPOSE Call GIPSY processor.

ARGUMENTS

input Input file. Known types are PGIP CGIP DGIP and CPGIP

output Output file, optional.

/TAGS List of active tags. Default is /TAGS=V (on VAX)

/LA*NGUAGE Select 'foreign language' to be processed. This selection concerns the substitution of @-expressions within strings and comments of the selected language. Default is /LANGUAGE=PLI.

/LI*ST List output file

/DELETE Delete output file

/NOA*NNOUNCE If specified, no additional comment is written to the terminal

/TR*ACE If this keyword is given, GIPSY starts with
%SPECIFY TRACE=on
Otherwise it starts with
%SPECIFY TRACE=off

/B*REAK If given, keyboard interrupt is enabled. Default: keyboard interrupt disabled.

/ST*EP Initial value for GIPSY's STEP parameter. STEP is useful for debugging and can be changed at any place in your GIPSY program using
%SPECIFY STEP=1
The default is /STEP=10000

/TE*ST Initial SYSTEMTEST level (developers only). Defaults to /TEST=0

Description

FUNCTION	Process GIPSY files.
Version	1.02
Author	H.G.Essel + U.Post
Last Update	18-JAN-1990

Output

If no output file has been specified, a default output file name is constructed. The first part of its name is taken from the input file's name. The type of the output file depends on the type of the input file.

input type	output type
CGIP	C
CPGIP	CP
DGIP	COM
PGIP	PLI
otherwise	OUT

LANGUAGE

The selected language concerns the replacement of @-expressions within strings and comments of the 'foreign language'. Currently supported are the following languages:

NULL	no comments or strings known
ADA	ADA
ASM	/370 Assembler
C	C
C1	C (only nesting level 1 for comments)
CLIST	TSO CLIST
DCL	DCL

FORT	FORTRAN
MOD	Modula-2
OCCAM	OCCAM-2
PASC	Pascal
PASC1	Pascal (only nesting level 1 for comments)
PLI	PLI
REXX	REXX
SPIT	SPITBOL
SQL	SQL

GLCNVPROJECT

@GOO$EXE:GLCNVPROJECT file project

PURPOSE Convert project name within a history file

ARGUMENTS

file Name of the history file to be converted.

project Name of the new project for the history file.

Description

CALLING @GOO$EXE:GLCNVPROJECT file project

ARGUMENTS

file Name of the history file to be converted.

project Name of the new project for the history file. Only length 1 to 6 characters allowed.

FUNCTION This procdure converts the project name of a history file from the current to the new one, 'project'. The following modification is done:
The project field in the title record is updated.

REMARKS

EXAMPLE @GOO$EXE:GLCNVPROJECT GOO$MAN:GLHGOOTST.IDX GOOSY
Convert history file of test version to normal version

GLCNVV31

@GOO$EXE:GLCNVV31 file

PURPOSE Convert history file format from version 3.1 to 4.0

ARGUMENTS

file Name of the history file to be converted.

Description

CALLING @GOO$EXE:GLCNVV31 file

ARGUMENTS

file Name of the history file to be converted.

FUNCTION This procdure converts the format of a history file from V3.1 to V4.0 . The following modifications are done:

1. The version field in the title record is updated.

2. The author list field is inserted in the module directory records.

3. The node and home_node field is deleted in the history records. The comment field is shifted left. The procedure creates a backup copy of the history file under the current default directory, the given file name but the extension .OLD.

REMARKS Note, that the lock and directory records, which are modified also in for version V3.1 are NOT changed by this procedure.

EXAMPLE @GOO$EXE:GLCNVV31 GOO$MAN:GLHGOOSY.IDX

GLCOMPILE

GLCOMPILE file

PURPOSE Compile a file in the local test environment

ARGUMENTS

file File or list of files to be compiled.

Description

CALLING GLCOMPILE file

ARGUMENTS

file Files to be compiled. A list of files, separated by commas may be specified, each list item may contain wildcards. The default extension is .PPL.

FUNCTION This procedure calls for each file specified the COMPILE procedure with the option /DEB. In case of PLI, PPL and MAR files it is checked, whether an include or macro library with the name GLTEST exsists. In this case the logical name PLI$LIBRARY or MAR$LIBRARY in the process logical name table will be defined as GLTEST. If the compilation succeeds, the librarian is called to replace the object module in the library GLTEST in the current default directory. If this object will be created if not already exsisting.

REMARKS Because the libraries are always located in the current default directory, one may have several independant test environments in different subdirectories.

EXAMPLE

```
GLCOM TEST.PPL
  GLCOM NX*.PPL,DX*.PPL
```

GLCREATE

GLCREATE option arguments

PURPOSE Creates datasets or entries in the history file directories.

FORMAT GLCREATE DATASET

REMARKS -

DATASET

CALLING GLCREATE DATASET dataset

PURPOSE Create a new dataset, specify attributes and a module header.

ARGUMENTS

dataset Name of the dataset to be created. Currently only file specifications allowed, creation of text library modules is not yet supported.

FUNCTION First is checked whether the dataset name is already in use. If not, the name is reserved and a history record containing the purpose of the module is written.
Then the attributes are evaluated depending on the file type specified and the appropriate attribute records are written.
Finaly the header generator is started and will prompt all neccessary documentation header information.

REMARKS Currently all parameters are prompted !
Only file types PPL and COM supported ! For technical details look in the description of the procedures GLSETATTR and GLWRTATTR.

EXAMPLE GLCREATE DATASET GOO$NET:TEST.PPL

GLDELETE

GLDELETE option arguments

PURPOSE Deletes either datasets or information in the history file.

FORMAT
GLDELETE ABORT
GLDELETE ATTRIBUTE
GLDELETE CORRELATION
GLDELETE DATASET
GLDELETE HISTORY
GLDELETE MARKS
GLDELETE MODULE

REMARKS All command key words have to be specified unabriviated. This is done to protect against unintentional deletion of unrecoverable information.

ABORT

CALLING GLDELETE ABORT dataset

PURPOSE Delete all abort records for a dataset

ARGUMENTS

dataset Dataset, whose abort records are to be deleted.

FUNCTION This command deletes all abort records for the given dataset.
NOTE: The abort records indicate, that an update operation has not been completed and the project is thus in a potentially inconsistent state. The abort records are normally deleted with the successfull completion of the update operation.

===> This command should only be used to delete an abort record for an operation, which will or cannot be retried.

REMARKS For technical details look in the description of the procedure GLDELABOR.

EXAMPLE GLDELETE ABORT GOO$NET:N$PALCM.PPL

ATTRIBUTE

CALLING GLDELETE ATTRIBUTE dataset operation
/PRO1 /PRO2 /PRO3 /PRO4
/MHLB /MTLB
/CHLB /CTLB
/OLB /QUAL /OBT
/OPT /LIB /SHARE /SLB
/EPI1 /EPI2 /EPI3 /EPI4
/MARK

PURPOSE Delete one or all attribute of a dataset.

ARGUMENTS

dataset Name of the dataset whose attributes are to be deleted.

operation Operation for which the attribute has to be deleted. A full wildcard may be specified and will result in the deletion of the attributes of all operations.

/.... If one of the qualifiers is given, only the corresponding attribute will be deleted.

FUNCTION This procedure deletes either a single attribute, all attributes for a single operation or all attributes of a dataset. If the operation is specified explicitely together with one of the qualifiers, this single attribute will be deleted. If the operation is specified but no qualifier, all attributes of the given operation will be deleted. If the operation is specified as a wildcard, all attributes of the dataset will be deleted.

REMARKS For technical details look in the description of the procedure GLDELATTR.

EXAMPLE GLDELETE ATTRIBUTE GOO$CMD:CXCRECM.PPL COM

CORRELATION

CALLING GLDELETE CORRELATION scr_dsn oper dst_dsn

PURPOSE Delete a correlation between datasets.

ARGUMENTS

scr_dsn Correlation source dataset name.

oper Asociated operation of correlation

dst_dsn Correlation destionation dataset name. The arguments may be specified as full wildcards, but scr_dsn and dst_dsn must not be a wildcard simultaneously.

FUNCTION This command deletes a correlation and is complementary to the GLSET CORRELATION command.

REMARKS For technical details look in the description of the procedure GLDELCORR.

EXAMPLE GLDELETE CORRELATION GOOLIB(TEST) LIN GOOLIB(MAIN)

DATASET

CALLING GLDELETE DATASET dataset /COMMENT=c

PURPOSE Delete a dataset in a project directory or library.

ARGUMENTS

dataset Name of the dataset to be deleted.

/COMMENT=c Comment to be written in the history file.

FUNCTION This procedure deletes a dataset. The file in a directory or the module in a library is deleted and will be uncoverably lost. A history record is written indicating the delete status of the dataset.
Furthermore correlations are deleted in order to avoid inconsistencies in update jobs.
Finaly the dataset name is deleted in the dataset and module directories and added in the deleted directory. A summary of all deleted datasets is available with

```
GLSHOW DIRECTORY /DELETED
```

REMARKS For technical details look in the description of the procedure GLDELFILE.

EXAMPLE

```
GLDELETE DATASET GOO$EXE:GLATTRIB.COM
GLDELETE DATASET GOOHUTIL(GLATTRIB) -
        /COMM="Not longer used"
```

HISTORY

CALLING GLDELETE HISTORY dataset

PURPOSE Delete all information about a dataset in the history file.

ARGUMENTS

dataset Name of the dataset to be deleted. A full wildcard may be specified and will result in deleting the history of all datasets in status DELETE.

FUNCTION This procedure deletes all remaining information in the history file for deleted datasets. In order erase all information associated with a dataset two commands are needed:

```
GLDELETE DATASET xxx:yyy.zzz
                 ! delete dataset itself
GLDELETE HISTORY xxx:yyy.zzz
                 ! delete dataset history ect.
```

REMARKS For technical details look in the description of the procedure GLDELHIST.

EXAMPLE GLDELETE HISTORY GOO$EXE:GLATTRIB.COM

MARKS

CALLING GLDELETE MARKS /NOINTERLOCK

PURPOSE Delete all pending marks and the update lock.

ARGUMENTS

/NOINTERLOCK If specified, all marks and the update lock are deleted even if they have been set by another author.

FUNCTION This procedure deletes all pending marks and the update lock. Without the /NOINTERLOCK qualifier only marks created by the issuing author can be deleted. If marks of other authors are to be deleted of if an update lock is active, the /NOINTERLOCK qualifier must be specified.

REMARKS The GLDELETE MARKS /NOINTERLOCK function allows to recover from the deadlock that occurs if an UPDATE job has been STOPed or is crashed.
For technical details look in the description of the procedure GLDELMARK.

EXAMPLE GLDELETE MARKS

MODULE

CALLING GLDELETE MODULE module

PURPOSE Delete all datasets with the same module name.

ARGUMENTS

module Module name, no wildcards allowed.

FUNCTION This command deletes all datasets with the specified module name. This means, all files with a file name matching the given name and all library entries are deleted.
The command scans first the module directory and prompts for each matching dataset. Than the GLDELETE DATASET command is used to perform the actual delete.

REMARKS For technical details look in the description of the procedure GLDELMODU.

EXAMPLE GLDELETE MODULE U$TEST This may delete:
GOO$UTIL:U$TEST.PPL
GOOINC(U$TEST)
GOOLIB(U$TEST)
GOOHMOD(U$TEST)
GOOTMOD(U$TEST)

GLDOCUMENT

GLDOC*UMENT filespec /PUT extr.qual form.qual

PURPOSE	Generate documentation with GLEXTRACT and GLFORMAT.
ARGUMENTS	
/PUT	Use GLPUT rather than COPY for library operations. This implies /NOSUB and a GLUPDATE at the end.
filespec	Source file specification as passed to GLEXTRACT
extr.qual.	Qualifier passed to GLEXTRACT: /MTLB=/CTLB=/TYPE=/SORT
form.qual.	Qualifier passed to GLFORMAT: /SEL=/EXCL=/BRIEF/COM /TEX/SCRI/HELP/PRINT/RNO/HL2/LEV2/HLB=
EXAMPLES	GLDOC TEST.PPL /TEX GLDOC TEST.PPL /HELP GLDOC TEST.PPL /SCR

Description

CALLING	GLDOC*UMENT filespec /PUT extr.qual form.qual
ARGUMENTS	
filespec	Source file specification as passed to GLEXTRACT
/PUT	Use GLPUT rather than COPY for library operations. This implies /NOSUB and a GLUPDATE at the end. This switch is allowed only for GOOSY programmers!
extr.qual.	Qualifier passed to GLEXTRACT: /MTLB=/CTLB=/TYPE=/SORT

form.qual.	Qualifier passed to GLFORMAT: /SEL=/EXCL=/BRIEF/COM /SCRI/HELP/PRINT/HL2/LEV2/HLB=
FUNCTION	Calls GLEXTRACT and GLFORMAT.
REMARKS	-
EXAMPLE	-

GLEDIT

GLEDIT **dataset**
/COMMENT=c/NOMARK/NOSUBMIT/NOCONFIRM

PURPOSE Edit a dataset.

ARGUMENTS

dataset Dataset to be edited.

/COMMENT=c Comment to be used in final GLPUT operation.

/NOMARK Use NOMARK qualifier in final GLPUT.

/NOSUBMIT Use NOSUBMIT qualifier in final GLPUT.

/NOCONFIRM Use NOCONFIRM qualifier in final GLPUT.

DESCRIPTION

CALLING GLEDIT dataset /COMMENT=c/NOMARK/NOSUBMIT

ARGUMENTS

dataset Dataset to be edited.

/COMMENT=c Comment to be used in final GLPUT operation. The comment will be prompted if ommitted.

/NOMARK Use NOMARK qualifier in final GLPUT. This will inhibit the update of any datasets, for details look in the description of GLPUT.

/NOSUBMIT Use NOSUBMIT qualifier in final GLPUT. This will inhibit the submission of the update job. For details look in the description of GLPUT.

/NOCONFIRM Use NOCONFIRM qualifier in final GLPUT. This will delete the processed file on the current directory after the GLPUT. For details look in the description of GLPUT.

FUNCTION GLEDIT is a procedure to edit directly a file or text module. The specified file is copied with GLGET to the current directory, than the editor is called and finally the modified file is written back with GLPUT.

REMARKS Use GLEDIT only for small and 'cosmetic' changes. The modified version will be immediately available for all users of the project!!!

EXAMPLE GLEDIT GOO$EXE:GOOLOG.COM /COMM="GOOXUTIL added"

GLEXTRACT

```
GLEXTRACT file_list /MLIB=ml /CLIB=cl
          /MTLB=mt /MHLB=mh /CTLB=ct /CHLB=ch
          /TYPE=t/NOSUBMIT /SORT /DIRECT=file
          /CALL*LIB=tl
```

PURPOSE Extract documentation from source files or libraries and store in libraries or file.

ARGUMENTS

file_list	List of source files from which the documentation is to be extracted or list of library references library(module).
/MLIB=ml	Text library to store module documentation
/CLIB=cl	Text library to store command documentation
/MTLB=mt	Text library to GLPUT module documentation
/MHLB=mh	Help library to store module documentation using GLPUT and an update job
/CTLB=ct	Text library to GLPUT command documentation
/CHLB=ch	Help library to store command documentation using GLPUT and an update job
/TYPE=t	Dataset type to be used for library extracts
/NOSUBMIT	Use /NOSUBMIT with all GLPUT command issued.
/SORT	A sorting pass sorts all 2+ blocks.
/DIRECT=f	Specifies a file for names of created text modules.
/CALL*LIB=tl	Specifies a text library to insert calling statements for all entries in the file.

Description

CALLING

GLEXTRACT file_list /MLIB=ml /CLIB=cl
/MTLB=mt /MHLB=mh /CTLB=ct /CHLB=ch
/TYPE=t/NOSUBMIT /SORT /DIRECT=file
/CALL*LIB=tl

ARGUMENTS

file_list

List of sources from which the documentation is to be extracted. Each list element may be either a file specification with any wildcard allowed by VMS or a library reference in the format
<library>(<module>)

For libraries wildcards are only allowed in the module field but not for the library name itself.

/MLIB=ml

Text library to store module documentation.

/CLIB=cl

Text library to store command documentation.
Either use file output or library output: either both previous libs are given or none of them.

/MTLB=mt

Text library to GLPUT module documentation.
This switch is allowed only for GOOSY programmers!

/MHLB=mh

Help library to store module documentation using GLPUT and an update job.
This switch is allowed only for GOOSY programmers!

/CTLB=ct

Text library to GLPUT command documentation
This switch is allowed only for GOOSY programmers!

/CHLB=ch

Help library to store command documentation using GLPUT and an update job.
This switch is allowed only for GOOSY programmers!

/TYPE=t

Dataset type to be used for sources extracted from a library. Note, that this type determines the comment delimiters used during documentation extraction. Default is GIP.

/NOSUBMIT

Use /NOSUBMIT with all GLPUT command issued.

/SORT

A sorting pass sorts all 2+ blocks. Should be only one 1+ block in the file. Used for message files.

/DIRECT=f Specifies a file for names of created text modules. This file may be input to GLFORMAT.

/CALL*LIB=tl Specifies a text library to insert calling statements for all entries in the file.
This switch is allowed only for GOOSY programmers!

/VOICE Outputs text for DECtalk.

/SEPARATE Generates one module per 2+ block for DECtalk.

/VLIB=vl Replaces module in library vl for DECtalk.

/VTLB=vl Puts module in library vl for DECtalk.
The text modules have the name of the module. If /SEP is specified, the names are module_header, where header is a 2+ header.

FUNCTION This procedure extracts the documentation blocks found in a source and stores them either in files or in text libraries. The library update can be done either directly or with the Module Management GLPUT command.
The procedure first scans thru the given list of source datasets. Files are processed directly whereas library modules are first extracted on a temporary dataset with a file type specified by the /TYPE qualifier. Each file (or extracted file) is then processed with the MEXTHEA program which locates documentation blocks of the structure:

<cd>1+ <class>
<cd>
<cd>+ Module : <module name>
<cd>
<cd> ... any text in the appropriate format
<cd>
<cd>1- <class>

where <cd> is the comment delimiter of the language being processed, <class> denotes this language. Note, that a single file may have several documentation blocks (e.g. if several procedures or entries are coded in the same file). The extracted text is than processed in one of three ways depending on the given qualifiers: If none of the qualifiers /MLIB /CLIB /MTLB /MHLB /CTLB and /CHLB has been specified, the whole extracted documentation is written in a file with the same name as the source file or library but the file type .TXT. Command documentation blocks are discarted in this case. If /MLIB and/or /CLIB has

been specified, the module and command documentation blocks are inserted in the specified libraries with a module name determined from the documentation block. If /MTLB and/or /CTLB has been specified, the module and command documentation blocks are written written with the GLPUT command to the specified libraries with module names as mentioned above. If /MHLB and/or /CHLB has been specified in addition to /MTLB or /CTLB, an appropriate format mark will be issued by the GLPUT command which will trigger the update of the specified help libraries.

REMARKS -

EXAMPLE GLEXTRACT GOO$EXE:GLEXTRACT.COM

GLFORMAT

```
GLFORMAT input output /SELECT=sl /EXCLUDE=el
                     /TEX/SCRIPT/HELP/PRINT/RNO
                     /BRIEF/COMMAND/TOC/HL2/LEV2
                     /LIB=l /HLB=ml /NOSUBMIT
                     /LOG/DIAGNOSIS
```

PURPOSE Format extracted documentation headers.

ARGUMENTS

input	Input specification.
output	Optional output file name.
/SELECT=s	List of documentation blocks to be selected.
/EXCLUDE=e	List of documentation blocks to be excluded.
/SCRIPT	Generate a file suitable for processing with SCRIPT.
/TEX	Generate a file suitable for processing with LaTeX.
/HELP	Genarate a HELP text module.
/PRINT	Generate a file for printer output.
/RNO	Generate a file for DEC's Runoff. If /TOC is specified, an additional file .RNOTOC is created.
/BRIEF	Process only first documentation block.
/COMMAND	Format a command description.
/TOC	Generate table of contents.
/HL2	Top heading level for SCRIPT is H2 instaed of H1.
/LEV2	All 2+,3+ blocks generate SCRIPT h2-headlines. If /HL2 is specified h3-headlines are generated.

/LIB=l	Help library to store help text module
/HLB=hl	Help library to store help module using GLPUT.
/NOSUBMIT	Use NOSUBMIT qualifier for all GLPUT commands.
/LOG	Log composition of input text file.
/DIAGNOSIS	Temporary file are not deleted.

Description

CALLING

```
GLFORMAT input output /SELECT=sl /EXCLUDE=el
                     /TEX/SCRIPT/HELP/PRINT
                     /BRIEF/COMMAND/TOC/HL2/LEV2
                     /LIB=l /HLB=ml /NOSUBMIT
                     /LOG/DIAGNOSIS
```

ARGUMENTS

input

Input specification of documentation headers to be processed. This may either be a set of text library modules or a list of files. The legal formats are:

```
library(module[,module[,...])
file[,file[,...])
```

where the module or file names may contain any wildcard allowed by LIBRARY or COPY respectively. Input may be a help module or a help library. (File types .HLP or .HLB). In this case only /TEX or /SCRI are used.

output

Optional file name for output files. If not specified, the (first) input file name or module name or library name is used. The file types depend on the formatter:

```
.SCR .TEX .HLP .DOC
```

/SELECT=s

List of documentation blocks to be selected.

/EXCLUDE=e

List of documentation blocks to be excluded. A %I means exclude IMPLEMENTATION,UPDATES,INTERNALS, i.e. a user version is generated.

/SCRIPT

Generate a file suitable for processing with SCRIPT. A file with the file type .SCR will be created.

/TEX	Generate a file suitable for processing with LaTeX. A file with the file type .TEX will be created. Use the TDOCUMENT command to process output.
/HELP	Genarate a HELP text module. A text file with the file type .HLP will be created unless either /LIB or /HLB has been specified.
/PRINT	Generate a file for printer output. A file with the file type .DOC will be created. This is the default if none of the qualifiers /SCRIPT/HELP and /PRINT is specified explicitely.
/BRIEF	Process only first documentation block.
/COMMAND	Signals, that the input refers to command description modules.
/TOC	Generate table of contents. File types .STC .TTC .HTC .DTC.
/HL2	Top heading level for SCRIPT is H2 instaed of H1. This means, that no new page is used for each module. It is default for /BRIEF.
/LEV2	All 2+,3+ blocks generate SCRIPT h2-headlines. If /HL2 is specified h3-headlines are generated.
/LIB=l	Help library to store help text module
/HLB=hl	Help library to store help module using GLPUT. This switch is allowed only for GOOSY programmers!
/NOSUBMIT	Use NOSUBMIT qualifier for all GLPUT commands.
/LOG	Log composition of input text file.
/DIAGNOSIS	Temporary file are not deleted.
FUNCTION	This procedure creates a file of concatinated documentation headers either by extraction from a text library or by simple file concatination. This file is than processed by the program MFORMDO to generate a text file suitable either for RUNOFF, TeX or SCRIPT. In case of HELP or PRINTable documentation the RUNOFF formating is done.
REMARKS	-
EXAMPLE	GLFORMAT GOOTMOD(N$*) /SCRIPT will create SCRIPT documentation of all modules starting with N$ in the file N$.SCR.

GLGET

GLGET dataset file /COMMENT="comm"

PURPOSE Copy a dataset from a project directory or library to the current working directory.

ARGUMENTS

dataset Name of the dataset to be copied.

file File name in the working directory, may be ommitted

/COMMENT=c Comment to be added in history file

DESCRIPTION

CALLING GLGET dataset file /COMMENT="comm"

ARGUMENTS

dataset Name of the dataset to be copied.

file File name in the working directory. If omitted, the output file name is defaulted from dataset name.

/COMMENT=c Comment to be added in history file

FUNCTION -

REMARKS -

Example GLGET GOO$IO:TEST.PLI

GLINFO

GLINFO key

PURPOSE Call GOOSY information system

ARGUMENTS

key may be abbreviated:
ADD
DEL*ETE
MOV*E
SHO*W

NOTE Add new titles in lowercase for more readability. Enclose them in quotes ("").

ADD

CALLING GLINFO ADD topic title

ARGUMENTS

topic One of the following (can be abbreviated):
KN*OWN_ERRORS
FI*XED_ERRORS
PR*OPOSED
SC*HEDULED
IM*PLEMENTED
CO*MMENTS
MO*DIFICATIONS

title Word to identify the problem / proposal. Date and username are added by GLINFO. Characterset like module names!!!

FUNCTION Adds a new title to selected topic. Editor is entered to write text.

REMARKS -

EXAMPLE GLINF ADD KN error_in_xx

DELETE

CALLING GLINFO DELETE topic title

ARGUMENTS

topic One of the following (can be abbreviated):
KN*OWN_ERRORS
FI*XED_ERRORS
PR*OPOSED
SC*HEDULED
IM*PLEMENTED
CO*MMENTS
MO*DIFICATIONS

title Word to identify the problem / proposal Characterset like module names!!!

FUNCTION Deletes title from selected topic.

REMARKS -

EXAMPLE GLIN DEL KN error_in_xxx

MOVE

CALLING GLINFO MOVE topic title topic

ARGUMENTS

topic One of the following (can be abbreviated):
KN*OWN_ERRORS
FI*XED_ERRORS
PR*OPOSED
SC*HEDULED
IM*PLEMENTED
CO*MMENTS
MO*DIFICATIONS

title Word to identify the problem / proposal Date and username are added by GLINFO. Characterset like module names!!!

FUNCTION Moves a title from one topic to another. Editor is entered to modify text.

REMARKS -

EXAMPLE GLIN MOVE KNOWN error_in_zzz FIXED

SHOW

CALLING GLINFO SHOW [topic][title]

ARGUMENTS

topic One of the following (can be abbreviated):
KN*OWN_ERRORS
FI*XED_ERRORS
PR*OPOSED
SC*HEDULED
IM*PLEMENTED
CO*MMENTS
MO*DIFICATIONS

title Word to identify the problem / proposal Characterset like module names!!!

FUNCTION Shows title of selected topic or all topics.

REMARKS -

EXAMPLE GLIN SHOW IMPL

GLMAIL

GLMAIL "subject"
/AUTHOR/USER/PROJECT/KEY=key/CONF=conf

PURPOSE Send MAIL to current projects author group or user group and add this message in the appropriate news file.

ARGUMENTS

subject short subject of your message. Also used as title for notebook, if /KEY is specified.

/AUTHOR send to author group, this is default.

/PROJECT send to project group.

/USER send to user group.

/KEY=key Optional key for notebook.

/CONF=conf optional conference for notebook

DESCRIPTION

CALLING GLMAIL "subject" /AUTHOR/USER/PROJECT/KEY=key/CONF=conf

ARGUMENTS

subject Short subject of your message. The subject string has to be enclosed in quotes (") if it contains any delimiters. Also used as title for notebook, if /KEY is specified.

/AUTHOR Send to author group, this is default. Usernames are defined in file _GL_SYSDIR:GLSproject.DIS

/USER Send to user group. Usernames are defined in file _GL_SYSDIR:GLUproject.DIS

/PROJECT Send to project group. Usernames are defined in file _GL_SYSDIR:GLPproject.DIS

/KEY=key Key for notebook (Required if conference is specified). Should be BUG-FIX, NEW, CHANGE, COMMENT, HINT

/CONF=conf optional conference for notebook. Available notebooks at GSI are GOOSY, VME-PROJ and VME-EXP.

FUNCTION The Editor is invoked to enter the message text in a predefined template form. Than a batch job is submitted which first invokes the CMAIL command to distribute the given message to all members of the specified group by using the distribution list:

'_GL_SYSDIR':GL<x>'_GL_PROJECT'.DIS

x = S in case of /AUTHOR

x = U in case of /USER

x = P in case of /PROJECT

Finally the news file will be updated. This sequential file is rewritten, now containing the entered message as the first one. The new file is created if not already existing and has the name:

'_GL_SYSDIR':GL<x>'_GL_PROJECT'.TXT

x = S in case of /AUTHOR

x = U in case of /USER

x = P in case of /PROJECT

Example GLMAIL "New commands implemented"

GLMANAGER

GLMANAGER option arguments

PURPOSE Performs all project manager functions.

FORMAT

GLMANAGER ACCESS
GLMANAGER ANALYSE
GLMANAGER CHEAT
GLMANAGER CREATE
GLMANAGER DISTRIBUTE
GLMANAGER INITIALIZE
GLMANAGER INSTALL
GLMANAGER INVENTORY
GLMANAGER KITBUILD
GLMANAGER OVERVIEW
GLMANAGER PURGE
GLMANAGER RECLAIM
GLMANAGER REMIND
GLMANAGER STATISTIC
GLMANAGER VERIFY
GLMANAGER ZENSUS

REMARKS Most of the subfunctions are pure project management actions and should be executed only by the project manager ! Some GLMANAGER subfunctions allow in contrast to all other module management functions to specify explicitely a project name. In those cases this project is made the current project for the execution of the command and the old project is restored afterwards.

ACCESS

CALLING GLMANAGER ACCESS project /AUTHOR/PASSWORD

PURPOSE Create or modify the project access program or the project author list

ARGUMENTS

project Project name. Must be current project.

/AUTHOR GLGET, edit and GLPUT project author list file.

/PASSWORD Compile and link project access program and GLPUT it to _GL_SYSDIR. The program is invoked by GLACCESS.

EXAMPLE
```
GLMAN ACCESS GOOSY /AUTHOR
    GLA GL-commandline
```

ANALYSE

CALLING GLMANAGER ANALYSE

PURPOSE Invoke ANALYSE/RMS utility to analyse the history file.

ARGUMENTS

FUNCTION Invokes the ANALYSE/RMS utility to analyse the history file. The current file attributes and a file usage statistic is displayed. The whole history file is scanned and structure errors are reported.

REMARKS The file usage statistic should be examined from time to time (every some weeks). If the 'overall space efficiency' is less then 50%, the file should be compressed with the GLMANAGER RECLAIM command.

NOTE Under normal circumstances the 'overall space efficiency' is around 70% and does't vary in time. Only if a large amount of records have been recently deleted (e.g. with GLDELETE HISTORY *), the efficiency may decrease for some time.

EXAMPLE
```
GLMAN ANALYSE
```

CHEAT

CALLING GLMANAGER CHEAT file /WRITE /REWRITE /DELETE

PURPOSE Procedure to write, rewrite or delete a set of records in the history file.

ARGUMENTS

file File containing a list of records to be written or deleted. Default file type is ".XDI".

/WRITE Records are added to history file.
If a new record has a key already existing in the history file an error message will be written and input record is ignored and NOT replaced.

/REWRITE Records are rewritten in history file.
For each record in the input file is checked whether a record with the same key already exists in the history file. In this case the old record is deleted and than replaced by the new one. If no old record is found an error message will be written and the input record is ignored and NOT simply added.

/DELETE Records are deleted in history file.
For each record in the input file is checked whether a record with the same key already exists in the history file. In this case the old record is deleted. If no old record is found an error message will be written.

FUNCTION This function allows in a simple but powerfull way to manipulate records in the history file. Typically a set of records is extracted from the history file with the SEARCH Utility, manipulated with the editor and than either rewritten or deleted.

REMARKS Be aware the absolutely NO consistency checks are done by this procedure. There is no protection against destroying the structure of the history file. Therefore this procedure should only be used if the structure of the history file is really well understood. For technical details look in the description of the procedure GLMANCHEA.

EXAMPLE GLMAN CHEAT test /WRITE

CREATE

CALLING GLMANAGER CREATE project

PURPOSE Define and initialize a new project.

ARGUMENTS

project Name of the new project. This name must have 1 to 6 characters and must not contain any special characters.

FUNCTION This command performs the following actions:

All information needed to build a project definition file is prompted. Especially the logical names of the manager, library, executable and scratch directory are inquired.

The project definition file is created under the project manager directory and a GLPROJECT command executed to make the new project the current project.

The GLMANAGER INITIALIZE command is executed to create and initialize the projects history file.

REMARKS

Be aware, that the execution of this command is the LAST step in setting up a new project!
Before you should work through the following check list:

1. Select the home node for the project.
2. Select or create the project manager account.
3. Login on home node under project manager account
4. Create the 4 default sub-directories for
   manager files
   libraries
   executable images and command procedures
   scratch (especially for UPDATE Jobs)
5. Create any number of subdirectories for the files and modules to be handled.
6. Define logical names for all used subdirectories in a command procedure, which is executed in all LOGIN procedures of project developers.

For technical details look in the description of the procedure GLMAN-CREA.

EXAMPLE

GLMAN CREATE TEST

DISTRIBUTE

CALLING

GLMANAGER DISTRIBUTE project node
/FULL /INC*REMENTAL /SIN*CE=s
/ALL*_VERSIONS
/PAS*SWORD=password
/CLU*STER=c /KEE*P=k /LOG
/MIN*IMUM

PURPOSE

Distribute the current version of a project to another node via DECNET.

ARGUMENTS

project

Name of the project to be distributed. The project must be known in the session (must have been defined with a GLPROJECT command).

node

Node which is to be updated. In case of a cluster update this specifies the node thru which the DECNET accesses to the common file system are done.

/FULL Signals full distribution. All files of the project are updated on the destination node.

/INC*REMENTAL Signals incremental distribution. Only files which have been modified since the last distribution to the destination node are updated.

/SIN*CE=s Only files which have been modified since the specified date are updated on the destination node.
This option should be used carefully and only to recover from crashed distributions. First check with GLSHOW DIRECTORY /NODE the date of the last successfull distribution.

/ALL*_VERSIONS If specified, all file versions matching the modified date selection are updated. The default is only to update the last version.
This qualifier may be used together with /FULL to create a complete copy of a project.

/PAS*SWORD=p Allows to specify project manager password for the destination node. The default is the project password specified in the GLPROJECT command.

/CLU*STER=c Specifies the node name to be used when the node directory in the history file is accessed. If this qualifier is omitted, the physical node name will be used. This qualifier should be used when distributing to a cluster node.

/KEEP=k If specified, all project directories on the destination node will be purged to the given version limit after a successfull distribution.

/LOG Log all file accesses. If specified, all file reads and restores are logged to SYS$OUTPUT.

/MIN*IMUM If specified, only those files necessary to run GOOSY, i.e. a minimum configuration, are updated. The default is all files.
This qualifier may be used together with /FULL to create a complete copy of a project.

FUNCTION This procedure perform the following actions:
The history file is opened with GLENQHIST EXCL.
This allows shared read acces to the project but ensures that there is no Update job outstanding or active.
In case of an incremental distribute the date of
the last successfull distribute is read from the node directory. A full distribution is done if the node is unknown up to now. If the /SINCE qualifier has been specified, this date will be used.

Then the directory is examined. For every directory entry found in the history the NWUPDATE procedure is called to copy either all files of this directory (/FULL) or the files modified after the last successfull update (/INCR).

If all directories have been processed successfully the distribution date is modified in node directory entry. This may be examined with the

GLSHOW DIRECTORY /NODE

command.

Finally a GLMANAGER INSTALL command is executed on the destination node to replace eventually updated versions of installed images.

REMARKS For technical details look in the description of the procedure GLMANDIST.

EXAMPLE

```
GLMAN DISTRIBUTE goosy V750A::
GLMAN DISTRIBUTE goosy MVIIC:: /MIN
```

INITIALIZE

CALLING GLMANAGER INITIALIZE

PURPOSE Create and initialize project history file.

ARGUMENTS

FUNCTION The history file for the current project is created and initialized. A history record for the projects definition file is written, placing this file in status PUT. This also ensures, that the directories are never empty.

REMARKS This command can only be executed, if no history file already exists. For technical details look in the description of the procedure GLMANINIT.

EXAMPLE

```
GLMAN INITIALIZE
```

INSTALL

CALLING GLMANAGER INSTALL project node /DELETE/REPLACE/LIST

PURPOSE Install, replace or list project images.

ARGUMENTS

project Project, for which the images are to be processed. Default is the current project.

node Node, on which the operation is to be executed. Default is the current node, not the home node. An * denotes all nodes in the cluster.

/DELETE Signals that images are to be deleted.

/REPLACE Signals that images are to be replaced.

/LIST Request a listing of the image status.

FUNCTION This procedure first checks whether the project image list exists. This file has the generic name

GOO$MAN:GLI<project>.COM

and contains a list of all installed images of a project (for details look in GL_PRIMER DATABASE).
If found, a CSYSINST command processing the image list is executed. If no qualifiers are specified, the images will be installed with the appropriate attributes. If one of the qualifiers has been specified, it is passed to CSYSINST and will result in a delete, replace or list operation.
If a node name is specified, the command is executed via NWDCL on this node.

REMARKS For technical details look in the description of the procedure GLMANINST.

EXAMPLE GLMAN INSTALL

INVENTORY

CALLING GLMANAGER INVENTORY directory /CLEANUP=f

PURPOSE Take an inventory of one or all directories or libraries of a project.

ARGUMENTS

directory The name of the directory or library for which the inventory has to be taken. If the argument has been omitted or is specified as a full wildcard, an inventory of all directories and libraries is taken.

/CLEANUP=f Specifies the name of a cleanup command procedure. This is a DCL procedure generated by the inventory which contains

DELETE
LIBRARY/DELETE

GLDELETE DATASET
commands for all datasets not found in the history file or in the directories/libraries. This procedure is NOT automatically executed but left for further inspection and editing before an eventual execution to clean up inconsistencies.

FUNCTION This procedure checks whether all datasets found in the dataset directory exist physically and whether all datasets found physically are recorded in the dataset directory.
If the /CLEANUP qualifier is specified with a valid file name, a command procedure is generated suitable to clean up inconsistencies.

REMARKS For technical details look in the description of the procedure GLMANINVE.

EXAMPLE GLMAN INVE GOO$CMD

KITBUILD

CALLING GLMANAGER KITBUILD project device
/DENSITY=den /REWIND/MOUNT
/ALL_VERSIONS/LOG/NOSOURCE

PURPOSE Generate a distribution kit for a project.

ARGUMENTS

project Name of the project to be distributed. The project must be known in the session (must have been defined with a GLPROJECT command).

device Name of the device which will receive the savesets of the distribution kit. This may be either
a magtape device name (mounted FOREIGN !)
or a disk device and directory name.

If omitted, the current default device/directory will be used.

/DENSITY=d Is used to specify the tape write density in the DCL MOUNT command. Will be ignored if not specified together with a /MOUNT qualifier.

/MOUNT If specified, the output tape is automatically mounted and dismounted.

/REWIND If specified, the output tape is rewound before writing the first save set. A volume label is written using the project name as the label.

/ALL_VERSIONS Signals full distribution kit. All file versions are written to the saveset. If ommitted, only the last version of a file is stored.

/LOG Log all file accesses. If specified, all file reads and restores are logged to SYS$OUTPUT.

/NOSOURCE Exclude sources PPL,FOR,MAR,INC

FUNCTION This procedure builds a distribution kit of a project using the VMS BACKUP utility. Several save sets are created either on a disk or tape volume. The first save set has the name

<project>_1ST.BCK

and contains the two files

<project>_READ1ST.TXT
<project>_RESTORE.COM

The first file contains a description of the distribution kit and some hits how to restore it on another node. This file is left on the current default directory and should be printed and mailed together with the distribution medium. The second file contains a sample command procedure for reading the save sets from a tape. For every directory known in the history file a save set is written with the name

<logical_directory_name>.BCK

containing either the last or all versions of all files.

REMARKS For technical details look in the description of the procedure GLMANKITB.
NOTE: A warning is printed, if this command is issued on a node other than the projects home cluster.

EXAMPLE GLMAN KITBUILD goosy MTA0:

PURGE

CALLING GLMANAGER PURGE dsn /KEEP=k /BEFORE=b

PURPOSE Purge history records.

ARGUMENTS

dsn Name of a dataset or directory/library. The history of all datasets in a directory or library will be purged if not a full dataset name is specified.

/KEEP=k Specifies, how many history records are kept. The default and minimal value is 3. Note, that the first history record and the purge record are always kept and are not counted.

/BEFORE=b Specifies, how old a history record must be to be deleted. The default is "-180-" which results in deleting only records older than half a year. The date may be specified either as absolute time in the format

DD-MMM-YYYY

or as a relative time (in units of days) like

"-DDD-"

Note, that the quotes are part of the syntax !!
The specified time must be more than 14 days in the past.

FUNCTION The procedure scans for all datasets matching the 'dataset' specification the history records and deletes a record if it is older than the time given with the /BEFORE qualifier AND if there are more records left then given with the /KEEP qualifer. The first (oldest) history record is never purged and marks always the creation date of a dataset. The procedure writes or updates a PUR history record which indicates the time of the last purge operation and the total number of purges records.

REMARKS For technical details look in the description of the procedure GLMAN-PURG.

EXAMPLE GLMAN PURGE GOOLIB

OVERVIEW

CALLING GLMANAGER OVERVIEW

PURPOSE Compile a full project managers overview.

ARGUMENTS

FUNCTION This procedure uses some Module Management display commands to compile a project managers overview. The output is directed to files with the name:

<project>_<year>_<month>_<day>_<topic>.LIS

e.g.: GOOSY_1985_09_20_SUMMARY.LIS Currently the following topics are created:

OVERVIEW from GLSHOW OVERVIEW

SUMMARY	from GLSHOW DIRECTORY /SUMMARY
MODULES	from GLSHOW DIRECTORY /MODULES
ATTRIBUTES	from GLSHOW DIRECTORY /ATTRIBUTES
STATISTIC	from GLMANAGER STATISTIC
INVENTORY	from GLMANAGER INVENTORY
ZENSUS	from GLMANAGER ZENSUS

REMARKS For technical details look in the description of the procedure GLMANOVER.

EXAMPLE GLMAN OVERVIEW

RECLAIM

CALLING GLMANAGER RECLAIM /COMPRESS

PURPOSE Invoke the CONVERT utility to reclaim unused space in the history file.

ARGUMENTS

/COMPRESS The history file will be compress if specified. The CONVERT utility copies all records to a new file version.
If omitted, the CONVERT/RECLAIM utility is used to reclaim unused buckets.

FUNCTION This procedure invokes the RMS CONVERT utility to either reclaim unused buckets or to recreate and compress the history file.

REMARKS The 'overall space efficiency' (see GLMANAGER ANALYSE) is in general after a compression around 100% or even higher (mainly due to key and data compression). After some usage of the history file this number will decrease to about 70%. Therefore a compression is only reasonable if the space efficiency has dropped to 50 % or lower.

EXAMPLE GLMAN RECLAIM

REMIND

CALLING GLMANAGER REMIND /LOCK_BEFORE=t /ABORT_BEFORE=t

/MAIL

PURPOSE Scan the locked datasets and aborted update actions and list or remind outstanding actions.

ARGUMENTS

/LOCK_BEFORE=t Only datasets locked before the given date will be listed or reminded. Default is "-14-" indicating 2 weaks.

/ABORT_BEFORE=t Only update actions pending longer than the given date will be listed or reminded. Default is "-2-".

/MAIL If specified, a MAIL will be send to the author to remind for an outstanding action. Default is only to list oustanding actions.

FUNCTION This procedure scans all locked datasets and aborted update operations. They are listed in a table indicating author, type of action, time and dataset if they are older than the specified times. If the /MAIL qualifier has been specified, all listed authors will receive a short MAIL indicating that there is still something to fix up.

REMARKS -

EXAMPLE GLMAN REMIND

STATISTIC

CALLING GLMANAGER STATISTIC

PURPOSE List a statistic of record types in the history file

ARGUMENTS

FUNCTION This function reads the entire history file and compiles a statistic abount the different record types found.

REMARKS This function takes quite some time because it is implemented as a DCL procedure. Therefore it should be actived in a batch job at night. For technical details look in the description of the procedure GLMANSTAT.

EXAMPLE GLMAN STATISTIC

VERIFY

CALLING GLMANAGER VERIFY file /TYPES/DIRECTORY/POINTER /TIME/DATA/CORRELATION

PURPOSE Verify structure of the history file.

ARGUMENTS

file History file to be analysed. The history file of the current project will be used if omitted.

/TYPES Read thru the history file and check for illegal record types.

/DIRECTORY Checks the consistency of the module, dataset and directory and deleted directory.

/POINTER Checks the consistency of all pointer records except the directory and time records.

/TIME Checks the consistency of time records.

/DATA Checks the consistency of data records.

/CORRELATION Checks the consistency of correlation records.

FUNCTION This procedure tries to verify the internal consistency of the history file. The qualifiers select different test types whose logic is described in seperate paragraphs below. If any inconsistency is found, a message indicating the reason, the record in truble and possible corrective actions is written. The records to be corrected are written furthermore in 3 files with the names

<project>_INS.XDI with records to be inserted
<project>_DEL.XDI with records to be deleted
<project>_REW.XDI with records to be rewritten

Note, that the VERIFY procedure doesn't make any corrections in the history file. Instead, the project manager should examine the above mentioned XDI files, check again whether the corrections look reasonable and than use the GLMANAGER CHEAT command to modify the history file.

REMARKS Note, that this procedure checks only the logical consistency of the record contents. It does not check, whether the physical file structure is o.k. To check this, use the ANALYSE/RMS DCL command. For technical details look in the description of the procedure GLMANVERI.

EXAMPLE GLMAN VERIFY

/TIME

PURPOSE Check the consistency of time records.

FUNCTION This procedure reads first thru the time records and checks the existence of the corresponding history records. Than the dataset directory is

scanned and for each dataset is checked, whether the time records for all history records exist.

ZENSUS

CALLING GLMANAGER ZENSUS directory

PURPOSE Count lines in dataset.

ARGUMENTS

directory Directory, in which all datasets are to be counted. All directories will be counted if omitted or specified as a full wildcard.

FUNCTION -

REMARKS For technical details look in the description of the procedure GLMANZENS.

EXAMPLE GLMAN ZENSUS

GLPUT

```
GLPUT file dataset
        /COMMENT="comm"
        /SOURCE/GENERATED/FOREIGN
        /NOMARK /NOCORRELATE /NOSUBMIT
        /DOCUMENT /COMPILE /LINK
        /PROLOGUE /EPILOGUE /UPDATE
        /NEW/NOCONFIRM/NODELETE
        /MFORMAT=lib/CFORMAT=lib
```

PURPOSE Store a file from the working directory to a project directory or library.

ARGUMENTS

file	File to be stored
dataset	Logical name of a directory or library
/COMMENT=c	Comment added in history record
/SOURCE	A source dataset is written back after a modification
/GENERATED	A generated dataset is updated
/FOREIGN	A new release of a foreign dataset is imported
/NOMARK	No automatic marking at all
/NOCORRELATE	No generation of correlation marks
/NOSUBMIT	No submission of an update job
/DOCUMENT	Mark for documentaion
/COMPILE	Mark for compilation
/LINK	Mark for linking
/PROLOGUE	Mark for prologue processing

/EPILOGUE	Mark for epilogue processing
/UPDATE	Mark for update processing
/NEW	Don't ask to confirm creation of a new dataset
/NOCONFIRM	Don't ask to confirm delete
/NODELETE	Don't delete input file
/MFORMAT=l	Mark to format module documentation
/CFORMAT=l	Mark to format command documentation

DESCRIPTION

CALLING

GLPUT file dataset
/COMMENT="comm"
/NOMARK /NOCORRELATE /NOSUBMIT
/DOCUMENT /COMPILE /LINK
/PROLOGUE /EPILOGUE /UPDATE
/NEW/NOCONFIRM/NODELETE
/MFORMAT=lib/CFORMAT=lib

ARGUMENTS

file File to be stored

dataset Logical name of a directory or library If a file is to be stored in a directory specify:
directory[:[file_name]]

If a module is to stored in a library specify:
library[(module)]

/SOURCE A source dataset is written back after a modification. This is the default in an interactive session and assumes, that the dataset has been get with the GLGET command and been modified in the authors private directory.

/GENERATED A generated dataset is updated. This is the default in the context of an update job and assumes, that the dataset has been generated from other sources and will replace the old version.

/FOREIGN A new release of a foreign dataset is imported. This assumes, that a new release of a foreign dataset is to be imported.

/COMMENT=c Comment added in history record. If the string contains any delimiters, it must be enclosed in quotes.

/NOMARK No automatic marking at all. Only the marks specified by one of the qualifiers /DOCUMENT, /COMPILE, /LINK, /PROLOGUE or /EPILOGUE are made. The default marks as well as the correlation marks are not issued.

/NOCORRELATE No automatic generation of correlation marks. The outbound correlations of the dataset are ignored.

/NOSUBMIT No submission of an update job. This allows to issue multiple GLPUT commands with the whole update done in one job.
NOTE: The project is locked after a GLPUT /NOSUBMIT for all other users! The allows the update of several datasets in a consistent way (see examples). But don't forget to start the update job with either a GLPUT without /NOSUBMIT or a GLUPDATE command.

/DOCUMENT Mark for documentation.

/COMPILE Mark for compilation.

/LINK Mark for linking.

/PROLOGUE Mark for prologue processing.

/EPILOGUE Mark for epilogue processing.

/UPDATE Mark for update processing. The default marks are ignored, if at least one of the above six qualifiers has been specified.

/NEW Don't ask to confirm creation of a new dataset. If not specified and in INTERACTIVE mode, the creation of a new dataset has to be confirmed.

/NOCONFIRM Don't ask to confirm delete.

/NODELETE Don't delete input file.

/MFORMAT=l Mark to format module documentation. The mark contains the specified library as parameter which will be used in the format step of the update job to define the module help library.

/CFORMAT=l Mark to format command documentation. The mark contains the specified library as parameter which will be used in the format step of the update job to define the command help library.

NOTE: The mark will always refer to all commands with the same first verb. This is done by using a wildcarded module name. The command
GLPUT test.txt tlib(SHOW_TIME) /CFORMAT=hlib
.BR. will result in a mark for TLIB(SHOW_*).

FUNCTION

This is the central procedure to store a dataset in a directory or library controlled by the module management. The following actions are performed:
First is checked, whether the intended operation is allowed.
If allowed, the dataset is copied to the specified directory or replaced(inserted) in library.
The the destination dataset is a known file on the current node, the CSYSINST command is invoked to reinstall the image on all nodes in the cluster containing the home node.
If all COPY or LIBRARY operations are completed successfully, all versions of file are deleted in the working directory. In an INTERACTIVE session, every file deletion has to be confirmed unless the /NOCONFIRM qualifier has been specified.

REMARKS

-

Example

GLPUT TEST.PLI GOO$IO The dataset TEST.PLI is transfered to the directory GOO$IO, the default and correlation marks are issued and an update job is started if any marks are pending.

```
GLPUT TEST1.PLI GOO$IO /NOSUB
GLPUT TEST2.PLI GOO$IO /NOSUB
GLPUT TEST3.PLI GOO$IO
```

The datasets TEST1.PLI,TEST2.PLI,TEST3.PLI are transfered and only one update job is submitted.

GLRELEASE

GLRELEASE dataset /COMMENT=c /NOINTERLOCK

PURPOSE Release a locked dataset without storing a dataset.

ARGUMENTS

dataset dataset to be released

/COMMENT=c Comment to be stored in history file

/NOINTERLOCK Allows project manager to release any dataset.

DESCRIPTION

CALLING GLRELEASE dataset /COMMENT=c /NOINTERLOCK

ARGUMENTS

dataset dataset to be released. Note, that the dataset must be in status GET !

/COMMENT=c Comment to be stored in history file

/NOINTERLOCK This qualifier allows the project manager to release datasets locked by other authors. You should notify the author who had locked the dataset !!!!
NOTE: This qualifier cannot be abriviated and is only valid under the project manager's account on the projects home node. Use it with great care, otherwise the confusion will be even greater !

FUNCTION This procedure releases a dataset which has been locked with the GLGET command. A history file entry is written indicating the release.

REMARKS -

EXAMPLE GLREL GOO$IO:TEST.PLI

GLSATTR

GLSAT*TR type module /UPD*ATE

PURPOSE Set attributes for certain types of modules

ARGUMENTS

type Module type:
INC*LUDE include modules
MES*SAGE message modules
TYP*E GOOSY data type descriptor module

module name of module
INC : Name of include module
MES : Name of message file (only name is used)
TYP : Name of include module

/UPDATE Glupdate is called after set attribute.

Description

FUNCTION Sets attributes to include or message modules.
INC: EPI /EPI1="CUPDINC module GOOINC GOOHINC"
UPD /MARK=EPI
MES: EPI /EPI1="CMSG GOO$SCRATCH GOO$MSG::module.MSG"
COM /OLB=GOOLIB
UPD /MARK=COM,EPI
TYP: EPI /EPI1="CUPDINC module GOOTYP GOOHTYP"
UPD /MARK=EPI

File name GLSATTR.COM

Dataset -

Version 1.01

Author H.G.Essel

Last Update 21-AUG-1985 28-JAN-1986 TYP implemented /KW

GLSET

GLSET option arguments

PURPOSE Defines or changes attributes and characteristics of datasets in the history file or modifies execution characteristics of module management operations.

FORMAT
GLSET ATTRIBUTES
GLSET CORRELATION
GLSET MODE

REMARKS -

ATTRIBUTES

CALLING
GLSET ATTRIBUTES dataset operation
/LIKE=dataset
/PRO1=dcl /PRO2=dcl /PRO3=dcl /PRO4=dcl
/MHLB=lib /MTLB=lib
/CHLB=lib /CTLB=lib
/OLB=lib /QUAL=q /OBT=t
/OPT=file /LIB=file /SHARE=name /SLB=lib
/EPI1=dcl /EPI2=dcl /EPI3=dcl /EPI4=dcl
/MARK=list

PURPOSE Defines or modifies attributes of a dataset.

ARGUMENTS

dataset Dataset for which the attributes are specified.

operation Module management operation, for which the attributes are specified, valid are:
PRO DOC COM LIN EPI UPD

If this parameter is ommitted and no qualifier has been specified an interactive mode is entered which will prompt all relevant attributes.

	If further qualifiers (except /LIKE) have been specified and this parameter is ommitted it will be prompted.
/LIKE=dsn	The attributes are copied from the specified dataset. If an operation has been specified only the attributes of this operation are copies, otherwise all attributes are copied. NOTE: All other qualifiers conflict with /LIKE.
/PRO1=dcl	First prolog command line, any valid DCL command.
/PRO2=dcl	Second prolog command line, any valid DCL command.
/PRO3=dcl	Third prolog command line, any valid DCL command.
/PRO4=dcl	Fourth prolog command line, any valid DCL command.
/MHLB=lib	Help library for module documentation output.
/MTLB=lib	Text library for module documentation output.
/CHLB=lib	Help library for command documentation output.
/CTLB=lib	Text library for command documentation output.
/OLB=lib	Object library for compiler output.
/QUAL=q	Additional compile Qualifiers
/OBT=t	Type of created object file, if not .OBJ
/OPT=file	List of LINKer options files.
/LIB=file	List of libraries to be searched during LINK.
/SHARE=n	Signal linking of a sharable image and specifies the file name of the image.
/SLB=lib	Name of sharable image symbol library
/EPI1=dcl	First epilog command line, any valid DCL command.
/EPI2=dcl	Second epilog command line, any valid DCL command.
/EPI3=dcl	Third epilog command line, any valid DCL command.
/EPI4=dcl	Fourth epilog command line, any valid DCL command.
/MARK=list	List of operations to be marked for execution in an UPDATE Job by GLPUT or GLUPDATE commands.

FUNCTION This command can be executed in two basic modes:

The operation parameter is specified:

Then the selected attribute is either created or, if already existing, edited. The editing changes only attributes explicitely specified, all others remain unchanged.
If an attribute is specified as one blank, e.g. /LIB=" ", then the attribute will be deleted.

The operation parameter is not specified:

Then all atttributes for all relevant operations are prompted. The file extension is used to select the relevant operations.

REMARKS For a detailed description of the different attributes look in the GL_PRIMER in chapter ATTRIBUTES.
For technical details look in the description of the procedures GLSETATTR and GLWRTATTR.

EXAMPLE GLSET ATTR GOO$CM:CXDSPMN.PPL COM /OLB=GOOLIB

CORRELATION

CALLING GLSET CORRELATION src_dsn operation dst_dsn

PURPOSE Define the corralation of two datasets.

ARGUMENTS

src_dsn Dataset name of the correlation source.

operation Operation to be executed for the destination datset.

dst_dsn Dataset name of the correlation destination.

FUNCTION This procedure defines a single correlation of two datasets. This will result in the automatic execution of the specified operation for the destination dataset if the source dataset is updated in the future.

REMARKS For the automatic generation of correlations for INCLUDEs ect. look in the description of the procedure GLTOOL CORRELATE. For technical details look in the description of the procedure GLSETCORR.

EXAMPLE GLSET CORR GOOINC(C$CRECM) COM GOO$UTIL:GOOSV0.PPL
This will result in the automatic compilation of the file GOO$UTIL:GOOSV0.PPL if the module GOOINC(C$CRECM) has been updated.

MODE

CALLING GLSET MODE /LOG/DEBUG/NOLOG/NODEBUG

PURPOSE Set some execution characteristics of module management procedures.

ARGUMENTS

/LOG Enable logging of all inserted and deleted history file records.

/DEBUG Enable logging of all read history file records. /DEBUG implies /LOG.

/NOLOG Disable all logging. /NOLOG implies /NODEBUG.

/NODEBUG Disable logging of all read history file records.

FUNCTION -

REMARKS -

EXAMPLE GLSET MODE /LOG

GLSHOW

GLSHOW option arguments

PURPOSE Displays information about the current status of the project and it's modules.

FORMAT
GLSHOW ABORTS
GLSHOW ATTRIBUTES
GLSHOW CHANGES
GLSHOW CORRELATIONS
GLSHOW DIRECTORY
GLSHOW HISTORY
GLSHOW LOCKS
GLSHOW MARKS
GLSHOW NODES
GLSHOW OVERVIEW
GLSHOW PROJECT
GLSHOW STATUS

REMARKS -

ABORTS

CALLING GLSHOW ABORTS

PURPOSE Display all modules of the current project, which had been marked for an operation in an UPDATE job, but where is operation failed.

ARGUMENTS -

FUNCTION This procedure lists all marks whose execution has been aborted in an UPDATE job and which have not been retried successfully up to now. For each aborted mark is listed:
dataset name
operation to be done
creation date of mark

issuing author
Note, that GLSHOW OVERVIEW executes this procedure.

REMARKS For technical details look in the description of the procedure GLSHWMARK.

EXAMPLE GLS MA

ATTRIBUTES

CALLING GLSHOW ATTRIBUTES dataset

PURPOSE Display all attributes of a dataset.

ARGUMENTS

dataset Name of the dataset, for which the attributes have to be listed. No wildcards are permitted.

FUNCTION Lists all attributes for a given dataset. The format of the output is similar to a GLSET ATTRIBUTE command. One line is printed for every attribute containing the name of the operation, the name of the attribute and the attribute value.

REMARKS For technical details see the description of the procedure GLSHWATTR.COM.

EXAMPLE GLS ATTR GOO$EXE:GLSHOW.COM

CHANGES

CALLING GLSHOW CHANGES /SINCE=s/BEFORE=b/NOCOMMENT
/ALL/LAST/FIRST/UPDATE/NOUPDATE

PURPOSE Display a time ordered list of all PUT and DELETE dataset operations for a given time interval.

ARGUMENTS

/SINCE=s Date of first entry to be shown. The date should be specified in the format:
dd-mmm-yyyy

or a YESTERDAY or TODAY. Note, that a time specification will be ignored. The default is YESTERDAY if this qualifier is omitted.

/BEFORE=b Date of last entry to be shown in the format given for the /SINCE qualifier. The default is TODAY.

/NOCOMMENT Suppress display of comment field.

/LAST Display only entries, which refer to the last change of a dataset. In other words, show only entries which reflect the current status. This is the default if none of qualifiers /ALL/FIRST /UPDATE/NOUPDATE has been specified.

/FIRST Display only entries, which refer to the first operation with a dataset. This may be usefull to pick new entries made in libraries in a certain time interval.

/UPDATE Display only entries, which refer to operations of an update job.

/NOUPDATE Suppress display, which refer to operations of an update job. In other words, display only operations issued directly by an author. This is the default if none of qualifiers /ALL/FIRST /LAST/UPDATE has been specified.

/ALL Display all entries, cancels the effect of /FIRST /LAST/NOUPDATE/UPDATE.

FUNCTION This procedure displays the 'relevant' history of all datasets in a given time interval. The history entries of PUT and DELETE operations are listed ordered by time. For each entry is displayed:

dataset name
transaction time
operation done (equivalent to new status)
number of later changes
author

The changes done yesterday and today will be shown if no time interval has been specified with the /SINCE and /BEFORE qualifiers. The default is to show only the last entry for a dataset and to skip all entries issued by update job activities. This yields the most condensed list of relevant changes. The /FIRST qualifier is very usefull to obtain a summary of newly created datasets. This may help when a transfer vector or other central things have to be updates manualy at some time intervalls.

REMARKS For technical details look in the description of the procedure GLSH-WCHAN.

EXAMPLE GLS CHA /SINCE=11-JUN-1984

CORRELATIONS

CALLING GLSHOW CORRELATIONS dataset /INBOUND/OUTBOUND

PURPOSE Display all correlations for a dataset.

ARGUMENTS

dataset Name of the dataset, for which the correlations have to be listed. No wildcards are permitted.

/INBOUND Signals, that inbound correlations are to be listed.

/OUTBOUND Signals, that outbound correlations are to be listed. This is the default.

FUNCTION This procedure lists either all inbound or all outbound correlations for a dataset. For each correlation is listed:
the associated dataset name
the operation to be performed

REMARKS For technical details see the descriptoin of the procedure GLSHW-CORR.COM.

EXAMPLE GLS CORR GOO$CMD:CXCRECM.PPL /INB

DIRECTORY

CALLING GLSHOW DIRECTORY name
/DATASET /MODULE /DELETED
/SUMMARY /ATTRIBUTES
/DIRECTORY

PURPOSE List a directory.

ARGUMENTS

name Name of the module(s), for which the directory has to be listed. Trailing wildcards are permitted.

/DATASET List dataset directory, sorted with directory name.

/MODULE List dataset directory, sorted with module name. This is the default if no other qualifier has been specified.

/DELETED List deleted dataset directory.

/ATTRIBUTE List summary of dataset attributes.

/DIRECTORY List directory directory.

/SUMMARY List dataset summary.

FUNCTION This command lists depending on the specified qualifiers one of the project directories.

REMARKS -

EXAMPLE GLS DI GL*

/DATASET

CALLING GLSHOW DIRECTORY /DATASET directory

PURPOSE List the dataset directory.

ARGUMENTS

directory Name of the directory or library, whose datasets are to be listed. A full wildcard is assumed if omitted.

FUNCTION List the datasets ordered alphabetically by full dataset name.

REMARKS For technical details look in the description of the procedure GLSHWDIRD.

EXAMPLE GLS DIR /DIR GOO$CMD

/MODULE

CALLING GLSHOW DIRECTORY /MODULE module

PURPOSE List the dataset directory.

ARGUMENTS

module Module name of the datasets to be listed. A trailing wildcard may be specified. A full wildcard is assumed if omitted.

FUNCTION List the datasets ordered alphabetically by module name. For each selected dataset is printed:
module name (for first dataset only)
type (if file, blank for library modules)
author and coauthors

REMARKS For technical details look in the description of the procedure GLSHWDIRM.

EXAMPLE GLS DIR /MOD GOO$CMD

/DELETED

CALLING GLSHOW DIRECTORY /DELETED module

PURPOSE List the deleted dataset directory.

ARGUMENTS

module Module name of the datasets to be listed. A trailing wildcard may be specified. A full wildcard is assumed if omitted.

FUNCTION List the deleted datasets ordered alphabetically by dataset name.

REMARKS For technical details look in the description of the procedure GLSHWDIRX.

EXAMPLE GLS DIR /DEL GL*

/ATTRIBUTES

CALLING GLSHOW DIRECTORY /ATTRIBUTES directory

PURPOSE Compile a summary of the attributes set for all datasets of a directory or library.

ARGUMENTS

directory Directory to be processed. All datasets will be processed if omitted or specified as a full wildcard

FUNCTION Lists a summary of the attributes set for the datasets matching the directory specification. For each dataset is displayed:
dataset name
default marks (UPD /MARK attribute)
operations, for which attributes are set
number of outbound and inbound correlations

REMARKS For technical details look in the description of the procedure GLSHWDIRA.

EXAMPLE GLS DIR /ATTR GOO$EXE

/DIRECTORY

CALLING GLSHOW DIRECTORY /DIRECTORY

PURPOSE List all directories and libraries.

ARGUMENTS -

FUNCTION Lists all directories and libraries. For each item is displayed:
name
processor type

REMARKS For technical details look in the description of the procedure GLSHWDIRL.

EXAMPLE GLS DIR /DIR

/SUMMARY

CALLING GLSHOW DIRECTORY /SUMMARY module

PURPOSE List all directories summary.

ARGUMENTS

module Module name of the datasets to be listed. A trailing wildcard may be specified. A full wildcard is assumed if omitted.

FUNCTION Lists a directory summary indicating all datasets known for a module name. For each module is displayed:
module name (on first line for a module)
file type (if a file dataset is known)
directory (if a file dataset is known)
object libraries (if containing a module)
help libraries (if containing a module)
other libraries
author (of the first dataset in group)
The help library field is prefixed with a "#", if the module is also found in a text library with the same name.

REMARKS For technical details look in the description of the procedure GLSHWDIRS.

EXAMPLE GLS DIR /SUM GL*

HISTORY

CALLING GLSHOW HISTORY name
/MAXIMUM=m/NOFIRST/NOCOMMENT
/DELETED

PURPOSE Display all history entries of selected modules.

ARGUMENTS

name Name of the dataset or modules to be listed. This parameter may be specified in the formats:
<directory>:<name>.<type>
<library>(<name>)
<module_name>

The module name may be specified with a trailing wildcard. If this parameter is omitted, a full wildcard is assumed and results in the histtory display of all datasets.

/MAXIMUM=m Maximum number of history entries to be displayed.

/NOFIRST The additional display of the first is suppressed.

/NOCOMMENT The display of the comment field is suppressed.

/DELETED List the history of physically deleted datasets.

FUNCTION This procedure displays the history of the selected datasets.
For each dataset is listed:
dataset name (for oldest entry only)
transaction time
operation done (equivalent to new status)
sequence number
author

REMARKS For technical details look in the description of the proecdure GLSHWHIST.

EXAMPLE GLS HI GL*

LOCKS

CALLING GLSHOW LOCKS module /AUTHOR=a/BEFORE=d

PURPOSE Display locked datasets of the current project.

ARGUMENTS

module	Name of a module, Trailing wildcard allowed. A full wildcard is assumed if this parameter has been omitted, resulting in a list of all datasets.
/AUTHOR=a	Name of an author. If specified, only datasets locked by this author are listed.
/BEFORE=d	Date. If specified, only datasets locked before this date are listed.

FUNCTION This procedure lists datasets in the states GET and RES. For each dataset is displayed:

- dataset name
- state
- time since locked
- author who originated the lock
- node on which the locking operation was executed

Note, that GLSHOW OVERVIEW executes this procedure.

REMARKS For technical details look in the description of the procedure GLSHWLOCK.

EXAMPLE GLS LO

MARKS

CALLING GLSHOW MARKS

PURPOSE Display all modules of the current project, which are marked for some action in an UPDATE job. Furthermore is indicated, whether an UPDATE job has been submitted.

ARGUMENTS -

FUNCTION This procedure lists all marks currently active. For each mark is listed:

- dataset name
- operation to be done
- creation date of mark
- issuing author

Note, that GLSHOW OVERVIEW executes this procedure.

REMARKS For technical details look in the description of the procedure GLSHWMARK.

EXAMPLE GLS MA

NODES

CALLING GLSHOW NODES

PURPOSE Display all known nodes.

ARGUMENTS -

FUNCTION This procedure lists all known nodes. For each node is listed:
node name
last update date

REMARKS For technical details look in the description of the procedure GLSHWNODE.

EXAMPLE GLS NO

OVERVIEW

CALLING GLSHOW OVERVIEW project

PURPOSE Displays the most briefly the status of the current or all projects.

ARGUMENTS

project Name of a known project or * for all.

FUNCTION This procedure executes for one or all known projects the commands:
GLSHOW LOCKS
GLSHOW MARKS

REMARKS For technical details look in the description of the procedure GLSHWOVER.

EXAMPLE GLS OV

PROJECT

CALLING GLSHOW PROJECT /FULL

PURPOSE Display all projects known in the current session and the definition of the current project.

ARGUMENTS

/FULL List relevant symbols

FUNCTION This procedure lists first all known projects. For aech project is displayed:
project name
name of definition file
an indication, whether a password was defined

The currently active project is marked by an "*". If /FULL is given, the definition of the current project is listed. For a symbols defined in the project definition file is listed:
symbol name, _GL_....
current value
equivalence string if value is a logical name

REMARKS For technical details look in the description of the procedure GLSHW-PROJ.

EXAMPLE GLS PROJ

STATUS

CALLING GLSHOW STATUS name /NOCOMMENT/DELETED

PURPOSE Display the last history entry of the selected datasets.

ARGUMENTS

name Name of the dataset or modules to be listed.

/NOCOMMENT The display of the comment field is suppressed.

/DELETED List the history of physically deleted datasets.

FUNCTION This procedure is just an abriviation for
GLSHOW HISTORY name /NOFIRST/MAXIMUM=1

REMARKS For technical details look in the description of the procedure GLSH-WHIST.

EXAMPLE GLS ST GL*

GLTEST

GLTEST option arguments

PURPOSE Collects all operations of the local test environment.

FORMAT
GLTEST COMPILE
GLTEST LINK

REMARKS -

COMPILE

CALLING
GLTEST COMPILE files /SEARCH=s/AND/OR
/LIBRARY=l
/NODEBUG/OPTIMIZE/MACHINE_CODE
/LINK/RUN

PURPOSE Compiles a single file or a set of files.

ARGUMENTS

file Files to be compiled. A list of file specifications, separated by commas may be specified, each list item may contain wildcards. The default file type is .PPL.
If a list item is given in the format
@<file_name>

it specifies a file containing a list of file specifications.
If the file specification is omitted, the last file edited with the EDDT command will be processed.

/SEARCH=sl All source file selected by the first parameter are searched for the list of words specified with this qualifier. They are only compiled if a match is found.
NOTE: The '@' format for the file specification is not supported if /SEARCH is used.

/AND All words given in a seach list must occur in a single record in order to select a source for compilation.

/OR Any word given in a search list must occur in any record in order to select a source for compilation. This is the default.

/LIBRARY=l List of additional compile libraries. Note, that the text or macro library GLTEST is always used if existing.

/NODEBUG Compiles programms without any debug options. The default is to compile with /DEBUG/CHECK/NOOPTIMIZE

/OPTIMIZE Compiles programms with debug but without check and with compiler optimisation. This results in programs with code nearly identical to those with normal compilations.
NOTE: Be aware, that there may be inconsistencies in the access of variables in DEBUG !!

/MACHINE_CODE The compiler will generate a listing including the generated machine code. This may help in finding compiler bugs.

/LINK Specifies, that the first object module is to linked as a main program.

/RUN Specifies, that the linked program is executed immediately.

FUNCTION This procedure calls for each file specified the COMPILE procedure with the following options:
no qualifiers == > /LIS/COM/DEB
/NODEBUG == > /LIS/COM
/OPIMIZE == > /SWI=DEBUG /LIS/COM
/MACHINE_CODE == > /SWI=MACHINE_CODE plus others...
For each file is checked, whether a file of same name and type exists in the current default directory. In this case a message is displayed and the local file is used.
In case of PLI, PPL and MAR files it is checked, whether an include or macro library with the name GLTEST exists. In this case a /LIBRARY qualifier is generated automatically for the COMPILE command. If the compilation succeeds, the librarian is called to replace the object module in the library GLTEST in the current default directory. This object library will be created if not already exsisting.

REMARKS Because the libraries are always located in the current default directory, one may have several independant test environments in different subdirectories. For technical details look in the description of the procedures GLTSTCOMP.

EXAMPLE

GLTEST COMP TEST.PPL
Compiles the file TEST.PPL in the default directory.
GLTEST COMP NX*,DX*

Compiles all files matching NX*.PPL and DX*.PPL in the default directory.
GLTEST COMP GOO$CMD:*.PPL

Compiles all PPL files in the directory GOO$CMD. If a file from GOO$CMD has a name like version in in the current default directory, this file will be compiled instead.
GLTEST COMP @DIR.LIS

Compiles all files, whose file names are found in the file DIR.LIS.
GLTEST COMP GOO$CMD:*.PPL /SEAR=SYS$

Compiles all PPL files in the directory GOO$CMD which contain the word SYS$. This compiles effectively all programms using system services.

LINK

CALLING

GLTEST LINK module /RUN/LIB=list/NODEBUG/NOUSER

PURPOSE

Links an executable image.

ARGUMENTS

module

Module name of the main program. This parameter can be specified in the following formats:

module_name
file_name
library_name(module_name)

If only a module name is specified, the library GLTEST.OLB in the current default directory will be used.

/RUN

Specifies, that the linked program is executed immediately.

/LIB=list

List of libraries to be searched explicitely.

/NODEBUG

Links image without /DEBUG option. Note, that the default is to link with /DEBUG.

/NOUSER Links image with /NOUSERLIBRARY option.

FUNCTION This procedure links an executable image using the library GLTEST.OLB in the current default directory.
Unresolved externals are first searched in GLTEST, than in the libraries specified with the /LIB qualifier and finaly in the default link search path.
The image name is determined by the module/file name of the main program.
If the image is a known image, the CSYSINST command will be used to replace the installed version by the just build version.

NOTE If GLTEST or another library specified with /LIB contains any modules, which are also linked in a shareable image found in the default link path, all libraries containing modules linked to the sharable image have to be included in the library list.

REMARKS Because the libraries are always located in the current default directory, one may have several independant test environments in different subdirectories. For technical details look in the description of the procedures GLTSTLINK.

EXAMPLE GLTEST LINK TEST
Links TEST.EXE from GLTEST(TEST).
GLTEST LINK GOOLIB(MCMD)

Links MCMD.EXE from GOOLIB(MCMD).

GLTOOL

GLTOOL option arguments

PURPOSE This command is a collection is software development tools available under the module managemant system.

FORMAT
GLTOOL CORRELATE
GLTOOL VECTOR

REMARKS -

CORRELATE

CALLING GLTOOL CORRELATE dst_dsn src_dsn
/COMPILE /DOCUMENT /LINK

PURPOSE Create automatically correlation records for different types of includes and for sharable image transfer vectors.

ARGUMENTS

dst_dsn Name of the correlation destination.

src_dsn Name of the correlation source.

/COMPILE Look for compile correlations. This is the default if no other qualifier has been specified.

/DOCUMENT Look for documentation correlations.

/LINK Look for link correlations.

FUNCTION This procedure analyses the datasets specified as correlation destination, checks for references of other datasets (which are located in the libraries or directories specified as correlation source) and creates the appropiate correlations.

REMARKS For technical details look in the description of the procedure GLTOOCORR.

EXAMPLE GLTO CORR GOO$CMD:*.PPL GOOINC,GOORTL /COMP

/COMPILE

CALLING GLTOOL CORRELATE /COMPILE dst_dsn src_dsn

PURPOSE Scan source datasets for INCLUDE statements and update the approriate compile correlations.

ARGUMENTS

dst_dsn Specification of the source datasets to be scanned for INCLUDE's. This parameter may be entered in one of the following formats:

<directory>:<name>.<type>
<directory>:*.<type>
<name>.<type>
*.<type>

If the directory is not specfied, a wildcard over all known directories is done. Note, that only a a full wildcard is supported for the name field.

src_dsn List of libraries or directories, in which the INCLUDEed datasets are located. The list may consist of or contain a full wildcard which will be replaced by the directory of the current destination file.

FUNCTION The procedure first checks, which datasets in the dataset directory match the destination specification. Those datasets are scanned for INCLUDE type statements, language specific notes see below. For each INCLUDE statements is checked, whether the included dataset is known in one of the libraries or directories specified in the source specification. If known, an appropriate COMPILE correlation is added to the history file. NOTE: A warning message is printed to include datasets not found in one of the specified libraries or directories. All include datasets with a name starting with "SYS$" are assumed to be found in a system supplied library and will not produce such a message.

Language notes The scan for the INCLUDE type statements is not done with a full syntactical analysis of the source language. This results in the following restrictions:

The INCLUDE statement must be in one source line.
Comment delimiters are recognised only in the record containing an INCLUDE statement.

PLI Looks for %INCLUDE statements of the format:

%INCLUDE module;
%INCLUDE 'file_name';

PPL Looks for %INCLUDE statements as described for PLI and for @INCLUDE statements of the format:
@INCLUDE $MACRO(module);

FOR Looks for INCLUDE statements of the format:
INCLUDE 'file_name'
INCLUDE 'lib_name(module)'
INCLUDE '(module)'

FTN Looks for INCLUDE statements as described for FOR

REMARKS For technical details look in the description of the procedure GLTOOCORR.

EXAMPLE GLTO CORR GOO$CMD:*.PPL GOOINC,GOORTL /COMP All PPL source programms in the directory GOO$CMD will be analysed and all compile correlations for INCLUDEs refering to text modules found in the libraries GOOINC or GOORTL will be updated.

/DOCUMENT

CALLING GLTOOL CORRELATE /DOCUMENT dst_dsn src_dsn

PURPOSE Scan source datasets for documentation includes and update the approriate document correlations.

ARGUMENTS

dst_dsn Specification of the source datasets to be scanned for includes. This parameter may be entered in one of the following formats:
<directory>:<name>.<type>
<directory>:*.<type>
<name>.<type>
*.<type>

If the directory is not specfied, a wildcard over all known directories is done. Note, that only a a full wildcard is supported for the name field.

src_dsn List of libraries or directories, in which the included datasets are located. The list may consist of or contain a full wildcard which will be replaced by the directory of the current destination file.

FUNCTION The procedure first checks, which datasets in the dataset directory match the destination specification. Those datasets are scanned for documentation includes of the format:
<comment_delimiter>0+<dataset_name>*...

REMARKS For technical details look in the description of the procedure GLTOOCORR.

EXAMPLE GLTO CORR *.PPL * /DOC
Scans all PPL programs for documentation includes. Only included files which are in the same directory as the including file are recognized.

/LINK

CALLING GLTOOL CORRELATE /LINK dst_dsn src_dsn

PURPOSE Scan the load map for an excutable or sharable image for the object module and sharable image references und update the appropiated link correlations.

ARGUMENTS

dst_dsn Specification of the image file(s) whose inbound correlations are to be investigated. This parameter may be entered in one of the following formats:

<directory>:<name>.EXE
<directory>:*.EXE
<name>.EXE
*.EXE

If the directory is not specfied, a wildcard over all known directories is done. Note, that only a a full wildcard is supported for the name field.

src_dsn List of libraries or directories, in which the object modules and images are located.

FUNCTION The procedure first checks, which datasets in the dataset directory match the destination specification. The 'Object Module Synopsis' parts of the map files of those images are then analysed. A correlation is set up for each object module and sharable image reference which can be located in the libraries and directories specified with <src_dsn>.

REMARKS For technical details look in the description of the procedure GLTOOCORR.

EXAMPLE GLTO CORR *.EXE GOOSHRLIB,GOOLIB,GOO$LIB /LINK

VECTOR

CALLING GLTOOL VECTOR library vector /SELECT=s/REJECT=r

PURPOSE Create automatically a transfer vector which matches all or a subset of the entries of an object library.

ARGUMENTS

library Name of the library to be scanned. Only object libraries can be specified, the default file type is .OLB.

vector Name of the transfer vector to be generated. The default file type is .MAR.

/SELECT=s List of entries to be selected. Only trailing wildcards are allowed. All entries not rejected are used if ommitted.

/REJECT=r List of entries to be rejected. Only trailing wildcards are allowed. All selected entries are used if ommitted. If an entry matched both the select and the reject list, it is rejected.

FUNCTION This procedure analyses the library and creates a transfer vector matching all entry points which match the /SELECT qualifier. The transfer vector has the standard GOOSY format:

```
.TITLE <name>
.IDENT /1.01A/
.LIBRARY /GOOMAC/
.MCALL TRV_BEGIN,TRV_CALLG,TRV_END
TRV_BEGIN NAME=<name>,LENGTH=<length>
......
TRV_CALLG entry_name ; module_name
TRV_END
.END
```

The <name> is the filename of the transfer vector. The <length> is calculated as

```
length = ((n + 128) / 64) * 64
```

and results in a page aligned vector with at least one spare page for extensions. NOTE: Check the generated transfer vector before the actual use. It must NOT contain references to globals which are not an entry point (e.g. message symbols, global constants ect.).

REMARKS For technical details look in the description of the procedure GLTOOLIBV.

EXAMPLE GLTO VECT GOOSHRLIB GOODMVEC /SELECT=M$*

GLUPDATE

GLUPDATE dataset
/NOMARK /NOCORRELATE /NOSUBMIT
/DOCUMENT /COMPILE /LINK
/PROLOGUE /EPILOGUE
/UPDATE

PURPOSE Mark a file for update actions.

ARGUMENTS

dataset	Dataset to be updated
/NOMARK	No automatic marking at all
/NOCORRELATE	No generation of correlation marks
/NOSUBMIT	No submission of an update job
/DOCUMENT	Mark for documentaion
/COMPILE	Mark for compilation
/LINK	Mark for linking
/PROLOGUE	Mark for prologue processing
/EPILOGUE	Mark for epilogue processing
/UPDATE	Mark for update processing

DESCRIPTION

CALLING GLUPDATE dataset
/NOMARK /NOCORRELATE /NOSUBMIT
/DOCUMENT /COMPILE /LINK
/PROLOGUE /EPILOGUE
/UPDATE

ARGUMENTS

dataset Dataset to be updated. Any dataset already existing in the project nay be specified.

/..... For details look in the description of the procedure GLPUT. All qualifiers work in exactly the same way.

FUNCTION

Actions are equivalent to GLPUT, except that no file is tranfered. If no file is specified, simply an update job is submitted if there are pending marks.

REMARKS

For technical details look in the description of the procedures GLEXEMARK and GLEXESUBM.

Example

GLUPDATE GOO$IO:TEST.PLI /COMP This command initiates the only the compilation of the dataset GOO$IO:TEST.PLI and submits an update job.

GLUPDATE GOO$IO:TEST.PLI

The MARK attribute is used to determine the default marks to be done after a modification. In addition all correlations are evaluated and the appropriate marks issued. Finaly an update job is submitted.

GLUPDATE

This command submits an update job if there are pending marks, otherwise no actions are performed. This form of the GLUPDATE command may be used to start the update job after several GLPUT /NOSUBMIT commands.

GNEWS

GNEWS outfile /SYSTEM/FILE

PURPOSE	Output all GOOSY mails.
ARGUMENTS	
outfile	Optional filename for output.
/SYSTEM	Output system mails. (Must type a dummy for
/FILE	Output to specified file.

Description

FUNCTION	Outputs the GOOSY summary mail file to terminal or file.
Version	1.01
Author	H.G.Essel
Last Update	16-JUN-1988

GNOTES

GNOTES command

PURPOSE Read or write notes in one of the GOOSY notebooks.

ARGUMENTS

command READ DIRECTORY or WRITE

EXAMPLES

```
$ GNOTES DIRECTORY /GOOSY ! all notes
$ GNOTES DIRECTORY /GOOSY/ALL ! notes + replies
$ GNOTES DIRECTORY BUGFIX /GOOSY ! all BUGFIX notes
$ GNOTES READ 2 /GOOSY ! read second note
$ GNOTES READ 2.4 /GOOSY ! read 4th reply
$ GNOTES WRITE "titel" /USER/KEY=HINT
$ GNOTES WRITE "titel" /USER/REPLY=2/FILE=filename
The default is /USER/KEY=COMMENT, "titel" is required
and should be between 30 and 50 characters.
If no file was specified, the editor is called to
edit the text.
```

Description

FUNCTION There are two VAX notes conferences for GOOSY:

GOOSY and GOOSY-USER

and a VME conference:

VME-EXPERIMENTS

The GOOSY conference is used by the GOOSY group to report updates. All GOOSY mails are stored here. The USER conference is for the users to report errors, hints, requests and comments. GOOSY notes are qualified by keywords BUGFIX, HINT, CHANGE, ERROR, COMMENT or NEW. USER notes are qualified by keywords REQUEST, HINT, COMMENT, ERROR. To make it easier to write and read notes, there is a DCL command for that. This command is not very fast, but should be used at least for writing notes. If one is

familiar with VAX notes, reading notes directly maybe more convenient. The command GNOTES may be invoked in different ways:

```
$ GNOTES
GNOTES> DIRECTORY /GOOSY/ALL ! notes + replies
GNOTES> READ 2 /GOOSY ! read second note
GNOTES> WRITE "titel" /USER/KEY=HINT
GNOTES> ^ Z
$ GNOTES ? enters menus
        Leave menus by PF4
$ GNOTES WRITE ? enters WRITE menu
        Leave menus by PF4
```

Version 1.01

Author R.Thomitzek

Last Update 30-NOV-1988

DIRECTORY

CALLING GNOTES DIRECTORY key /USER/GOOSY/VME/ALL /CONFERENCE=conf

PURPOSE Get list of notes of a conference.

key optional one of the following (default=COMMENT): HINT, ERROR, REQUEST, COMMENT, BUGFIX, NEW, CHANGE

/GOOSY List notes of conference GOOSY

/USER List notes of conference GOOSY-USER

/VME List notes of conference VME-EXPERIMENT

/CONFERENCE=c Specify other conference. If no conference is specified, /USER is default.

EXAMPLES

```
$ GNOTES DIRECTORY /GOOSY ! all notes
$ GNOTES DIRECTORY /GOOSY/ALL ! notes + replies
$ GNOTES DIRECTORY BUGFIX /GOOSY ! all BUGFIX notes
```

READ

CALLING GNOTES READ note /USER/GOOSY/VME
/CONFERENCE=conf

PURPOSE Read a note or reply of a conference.

note Number of note to read or note.reply to read a reply to a note. (List can be obtained by GNOTES DIR).

/GOOSY Read note of conference GOOSY

/USER Read note of conference GOOSY-USER

/VME Read note of conference VME-EXPERIMENT

/CONFERENCE=c Specify other conference. If no conference is specified, /USER is default.

EXAMPLES

```
$ GNOTES READ 2 /GOOSY ! read second note
$ GNOTES READ 2.4 /GOOSY ! read 4th reply
```

WRITE

CALLING GNOTES WRITE "title" /KEY=key/FILE=file/REPLY=r
/USER/GOOSY/VME
/CONFERENCE=conf

PURPOSE Write a note to a conference.

"title" Title line of note enclosed in apostrophies
(30 to 50 char.)

/KEY=key optional one of the following (default=COMMENT):
HINT, ERROR, REQUEST, COMMENT, for /USER
additional BUGFIX, NEW, CHANGE for /GOOSY

/FILE=file Optional text file. If not specified, the editor is called to write the text.

/REPLY=r Optional specify a note number here to write a reply for the note. The key argument is ignored in that case.

/GOOSY Write notes to conference GOOSY.
ONLY authorized people can write here!

/USER Write notes to conference GOOSY-USER.

/VME Write notes to conference VME-EXPERIMENT
ONLY authorized people can write here!

/CONFERENCE=c Specify other conference. If no conference is specified, /USER is default.

EXAMPLES

```
$ GNOTES WRITE "titel" /USER/KEY=HINT
$ GNOTES WRITE "titel" /USER/REPLY=2/FILE=filename
```

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.

HLPSCR

HLPSCR help-library module

PURPOSE Extracts modules from a Help library and generates a new output file for SCRIPT

ARGUMENTS

h-library Help library. If no type is specified, .HLB is assumed

module Module name. If no module is specified, all modules are extracted

Description

FUNCTION The specified module is extracted from the help library Program MHLPSCR is called to generate a new output file. Help level numbers and escape sequenzes are changed to SCRIPT commands. Output file has the name 'help-library'.SCR.

File name HLPSCR.COM

Dataset -

Version 1.00

Author G.Schneider

Last Update 29-OCT-1985

IBMSUBMIT

IBMSUBMIT vaxfilejcl
/MAILADDR= /NOANSWER/MSGLEVEL=

PURPOSE submit a JCL on the ibm and send optionally back the resulting output to the VAX

ARGUMENTS

vaxfilejcl VAX-file containing JCL-statements

/MAILADDR= the user can specify a VAX-node address where the mail (containing the output of the submitted job) shall be sent to. Default is the calling node with the calling user.

/NOANSWER the user can suppress the output (if he is sure, that his JCL will correctly run on the IBM) by specifying NOANSWER.

/MSGL=msgl NETEX message level can be set, 0=all, 12=severe only

Description

FUNCTION

1) The input is parsed by MDCLLIST.
2) The nodes are investigated :
   a) calling node
   b) node from where the vaxfilejcl is taken from
   c) node to where the mail shall be sent to
4) If the user has specified "NOANSWER", the program is continued at # 5)m)
5) The vaxfilejcl and a temporary file named IBMSUBMIT.TMP are opened. The vaxfilejcl is read record-by- record, its contents are parsed or changed and then written to the temporary file.
   a) Is first line of vaxfilejcl beginning with "//" ?

If not program is aborted with error-message
b) The IBM-jobid is read from the first record
c) Is there a jobcard in line #1 of vaxfilejcl ?
If not program is aborted with error-message
d) IBM-userid, IBM-password and MSGCLASS are read
e) If MSGCLASS is not T, it is changed to T
with a warning message
f) A netcard with NETID = JOBID is inserted
g) a dynamical allocation for the output is inserted
h) The vaxfilejcl is read and the temporary file
is written until
aa) a new jobcard is found
or bb) end_of_file of vaxfilejcl is detected
i) a second job in JCL is appended to
aa) create the IBM-output and
bb) a BFXTR-call to send the jcloutput as mail
(compressed by calling a trimming utility) via HYPERchannel to
DONALD
j) If end_of_file of vaxfilejcl is not yet
detected, the program will be continued at # 5)b)
k) The temporary file is closed.
l) If the calling node is not EMMA/DONALD, then
IBMSUBMIT is called again via NWDCL running DONALD.
m) The temporary file (in case of NOANSWER was
specified, the vaxfilejcl) is transferred to the IBM via BFXTI and will
there be submitted.

REMARKS

EXAMPLE

IBMSUBMIT CONVERT.JCL
the VAX-file CONVERT.JCL is sent to the IBM and will
there be submitted. IBMSUBMIT OUTLIST.JCL /MAILNODE=D::
the VAX-file OUTLIST.JCL is sent to the IBM and will
there be submitted. The output will be sent to the mail of the user on
node DONALD IBMSUBMIT TEST.JCL /NOANSWER
the VAX-file TEST.JCL is sent to the IBM and will
there be submitted. The user will get no mail.

LANL

LA*NL obj_list /OLB=objlib/OPT=optfile/CMD=cmdfile
/EXE=exefile /MAP=mapfile
/DEBUG /SHARE/NOSHARE

PURPOSE Link user specific analysis program

ARGUMENTS

obj_list List of user modules to be linked. Default unpack routines are provided for J11 data and compressed data (analysis output). For these input data the default analysis program J11 can be used, if no user analysis routine is needed.

/OLB=objlib optional user object library

/OPT=optfile optional link options file

/CMD=cmdfile File containing one line of link qualifiers. This name is appended to the link command by @cmdfile which means that the line is inserted in the command line before the command is excuted.
NOT VALID with /DEB switch.

/DEBUG The debugger is linked.
NOT VALID with /CMD switch.

/SHARE Valid together with /DEB switch. If not specified, image is linked from object libraries. If specified, image is linked from sharable images as it is without /DEBUG switch.

/NOSHARE Link from object libraries.
NOT VALID with /CMD switch.

/EXE=exefile Optional name of the exe file. Default is MGOOANL

/MAP=mapfile Optional name of map file.

EXAMPLES

```
LANL X$EVENT,X$ANAL /DEB
  LANL X$EVENT,X$ANAL /DEB/SHARE
  LANL X$E1,X$A1 /EXE=MYANL1
  LANL * /EXE=MYANL1
  LANL X$J11_ANAL /EXE=MYANL1
```

Description

FUNCTION

The user analysis program is linked. The file name is MGOOANL.EXE if not otherwise specified. The first argument is a list of files to be linked with MGOOANL. If /DEB is specified, but not /SHARE, MGOOANL is linked to object libraries instead of sharable images. This is the reason for longer linking time.

Version 1.01

Author H.G.Essel

Last Update 15-JAN-1987

LIBCOPY

LIBCOPY source_lib source_mod target_lib target_mod
/EDIT /GL /LOG

PURPOSE	Copies text modules from one library to another.
ARGUMENTS	
source_lib	Library to extract the module
source_mod	Name of module to be copied. May be wildcarded. Copy is done one by one.
target_lib	Library to insert the module
target_mod	Optional new name for text modules. Default is * which means to use name of source module.
/EDIT	The LSE editor is called for text modules.
/GL	The GOOSY code management calls are used to insert the module. The extract and insert operations are output.

Description

FUNCTION	The specified module(s) is(are) extracted from the source library and inserted in the target library. A new name may be specified (Text libraries).
Version	1.01
Author	H.G.Essel
Last Update	24-MAR-1986

LIBDEL

LIBD*EL library module

PURPOSE	Handle text library modules
ARGUMENTS	
library	Name of text library (file or logical name)
module	Module specification. May contain wildcards. Default is * to delete all modules. With @filename the module names are read from file.

Description

FUNCTION	The specified modules are deleted from library.
File name	LIBDEL.COM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	22-NOV-1985

LIBDIFF

LIBDIFF source_lib source_mod target_lib target_mod
/DIFFER

PURPOSE	Compare text modules from two libraries.
ARGUMENTS	
source_lib	Library one
source_mod	Name of module.
target_lib	Library two
target_mod	Optional name of second module. Default is * which means to use name of source module.
/DIFFER	Only names of different modules are output.

Description

FUNCTION	The specified modules are extracted from the libraries and compared.
Version	1.01
Author	H.G.Essel
Last Update	24-MAR-1986

LIBEXTR

LIBEXTR library module output

PURPOSE	Extract modules from text library.
ARGUMENTS	
library	Name of text library.
module	Module specification. A file containing the module names can be specified by @file
output	Output file name. Default is library.TXT
FUNCTION	The specified modules are extracted from the library.
Version	1.01
Author	H.G.Essel
Last Update	7-JUN-1988

LIBLIS

LIBLIS library module /FULL/SIN*CE=time
/BEF*ORE=time/EXT*RACT/OUT*PUT=file-spec

PURPOSE Lists or extracts specified modules of a library

ARGUMENTS

library Library name

module Module specification of the modules to be listed or extracted. One asterisk is permitted.

/FULL Requests a full description of each module in the module list.

/SINCE=t To specify that only those modules dated later than a particular time are used. You can specify an absolute time or a combination of absolute and delta time.

/BEFORE=t To specify that only those modules dated earlier than a particular time are used.You can specify an absolute time or a combination of absolute and delta time.

/EXTRACT Copies the specified modules from a library into a file. If you specify the /OUTPUT qualifier in conjunction with /EXTRACT output is written into the specified output file. If you omit the /OUTPUT qualifier output is written into a file that has the same file name as the library and a file typ depending on the typ of library (OBJ,MAR,EXE,HLP, TXT).

/OUTPUT=f File name for the output file. In conjunction with the /EXTRACT qualifier the output file contains the modules extracted from a library. Without /EXTRACT qualifier the output file contains the module list. If both /EXTRACT and /OUTPUT are omitted output is displayed on the screen.

Description

CALLING LIBLIS lib-name(module-spec) /FULL/SINCE=time /BEFORE=time/EXTRACT/OUTPUT=file-spec /MODULE=module-spec

ARGUMENTS

lib-name Library name

module-spec Module specification (overwrites /MODULE=m).

/MODULE=m Module specification of the modules to be listed or extracted. One asterisk is permitted. If the /MODULE qualifier is omitted, all modules are assumed.

/FULL Requests a full description of each module in the module list.

/SINCE=t To specify that only those modules dated later than a particular time are used. You can specify an absolute time or a combination of absolute and delta time. See section 2.5 in the VAX/VMS DCL dictionary for details on specifying times or access the HELP topic SPECIFY.

/BEFORE=t To specify that only those modules dated earlier than a particular time are used.You can specify an absolute time or a combination of absolute and delta time. See section 2.5 in the VAX/VMS DCL dictionary for details on specifying times or access the HELP topic SPECIFY.

/EXTRACT Copies the specified modules from a library into a file. If you specify the /OUTPUT qualifier in conjunction with /EXTRACT output is written into the specified output file. If you omit the /OUTPUT qualifier output is written into a file that has the same file name as the library and a file typ depending on the typ of library (OBJ,MAR,EXE,HLP, TXT).

/OUTPUT=f File name for the output file. In conjunction with the /EXTRACT qualifier the output file contains the modules extracted from a library. Without /EXTRACT qualifier the output file contains the module list. If both /EXTRACT and /OUTPUT are omitted output is displayed on the screen.

FUNCTION Creates a specified Library list or extracts specified modules.

REMARKS -

EXAMPLE LIBLIS GOOLIB /MOD=I$*/FULL Writes a full description of each module from the library GOOLIB beginning with I$ on your screen.
LIBLIS GOOINC /EXTRACT/SINCE=01-JUL-1985

/OUTP=UPD.TXT

Copies all Modules of the library GOOINC inserted later than 1-JUL-1985 in the file UPD.TXT.

LIBSEARCH

LIBS*EARCH library module list="search args"
/SINCE=time/BEFORE=time/FILE

PURPOSE	Calls SEARCH on modules temporarily extracted from libraries (text only).
ARGUMENTS	
library	Text library
module	Module name (wildcard). Default = *.
list	argument list passed to SEARCH command. Must be enclosed in "" if delimiters are used (Double quotes inside quoted strings!)! DO NOT USE THE /OUTPUT QUALIFIER ! USE /FILE
/FILE	Search output is written to files named: SEARCH_library_module.OUTPUT
/SINCE=time	Paramaters passed to LIBLIS to select modules
/BEFORE=time	Paramaters passed to LIBLIS to select modules

Description

FUNCTION	The specified modules are extracted to temporary files. Then the search is done.
NOTES	1. For each module one search is done. 2. The temporary files names are build from: SEARCH_library_module These names appear in the SEARCH output.
EXAMPLE	LIBS text c* xxx !search string xxx in modules c* LIBS text(c*) list=xxx !equivalent
Version	1.01

Author H.G.Essel

Last Update 24-MAR-1986

LIBTYPE

LIBT*YPE library module /PRINT /EDIT

PURPOSE	Handle text library modules
ARGUMENTS	
library	Name of text library (file or logical name)
module	Module specification. May contain wildcards, if /EDIT is not specified. Default is *.
/P*RINT	Module(s) is (are) printed.
/E*DIT	No wildcard for module! Module is fetched by GLGET, LSEDIT is called and module is put by GLPUT. If no switch is specified, modules are typed.

Description

FUNCTION	Text library modules are extracted from the library into a temporary file library_module.LIS. If no switch is specified, the content is output to terminal. With /PRINT, the temporary file is printed. With /EDIT the module is copied from the library, edited and written back using the GL commands.
File name	LIBTYPE.COM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	22-NOV-1985

LINKJ11

LINKJ11 objfile /COMPILE

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

LOADKEYS

LOADKEYS

PURPOSE Load F-keys of VT200/VT300.

FUNCTION The function keys are loaded. The keys must be defined as global symbols as:

```
Fnn=="string"
```

One may use CR and LF in the string, e.g.

```
F20=="SET DEF [.X]" + CR + LF + "SHO DEF" + CR +LF
```

Calling LOADKEYS will delete the global symbols Fnn.
The keys are invoked by <SHIFT><key>.

Version 1.01

Author H.G.Essel/Dick Fulton

Last Update 26-MAY-1988

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.

MESDEF

MESDEF facility /CLIB=/[NO]NEW/GLPUT/DELETE/LIST

PURPOSE	Generate message definition file for C programs.
ARGUMENTS	
Facility	Name of facility, i.e. GOOVME.
/CLIB=	Optional directory to copy the definition file.
/[NO] NEW	New version is linked. This is the default.
/GLPUT	Use GLPUT instead of copy, if library is specified.
/DELETE	Delete definition file (makes sense only if library is specified).
/LIST	A list file facility.LIS is written

Description

FUNCTION

Program MMESDEF is linked and called and generates a file <facility>_temp.PLI. This program is compiled, linked and executed and generates the file X<facility>. which contains define statements for the messages. This file can be included in C programs. The reason for this complicated way is that MMESDEF cannot recognise the severity of the messages, because it loops over message numbers. To add new messages, modify the message file on GOO$MSG.

EXAMPLE

```
$ GLGET GOO$MSG:XVME.MSG
  $ LSE XVME.MSG
  $ GLPUT XVME.MSG GOO$MSG
  $ MESDEF GOOVME /CLIB=VME$INC/GLPUT/DEL
```

MOVE

MOVE filespec dest /CONF /LOG

PURPOSE	Copies files and deletes them on source directory.
ARGUMENTS	
filespec	files to be copied (wildcard, list or @file for container file)
dest	Destination file. Must be specified. May be wildcard.
/CONFIRM	Confirm each transaction
/LOG	Display each transaction

Description

FUNCTION	COPY/LOG file dest DELETE/LOG/CONF file;
EXAMPLE	move x.* priv$save:* /conf /log move x1.dat,x2.dat [abc]x.dat /log
NOTE	The target files have the same names as the sources, if an asterisk was soecified as target file. Copy is done file by file.
File name	MOVE.COM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	3-JUL-1985

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

NEWMOD

NEWMOD * /SINCE=<date> /HELP

PURPOSE	Outputs a list of all new help modules
ARGUMENTS	Necessary dummy, if switch is specified
/SINCE=	date, e.g. yesterday (=default)
/HELP	A short help for each key found is displayed (default = /NOHELP)

Description

FUNCTION	outputs list of modules in all GOOSY help libraries.
File name	NEWMOD.COM
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	31-JUL-1985

NWCOPY

NWCOPY node source dest

PURPOSE Copy one or more files to one or more nodes

ARGUMENTS

node Node name, distribution list or wildcard

source File name or list of file names

dest Destination specification

DESCRIPTION

CALLING NWCOPY node source dest

ARGUMENTS

node Specification of nodes where the files have to be copied:

Node name A logical node name in the format
NODE[::]

List A distribution list reference in the format
@FILENAME

The distribution list must contain a list of logical node names in the above mentioned formant, one node per line.

Wildcard A node wildcard in the format
*

This is a synonym for the distribution list @NWDISTOTH which contains a list of all nodes known except the own node.

source	File name or list of file names, all wildcards legal for the COPY command are allowed.
dest	Destination specification, if ommited *.* is assumed.
REMARKS	NOTE: If a file already exsists on the target node, a new version of this file will be created with the next higher version number. If a file of a certain name and type is created for the first time, the version will be 1.
Example	NWCOPY NODE_A TEST.PLI Copies TEST.PLI to NODE_A

NWDCL

NWDCL node dcl_line output

PURPOSE Execute a single DCL command line on one or more nodes.

ARGUMENTS

node Node name, distribution list or wildcard

dcl_line Any DCL command, enclosed in quotes (")

output Output file, if ommitted SYS$OUTPUT is used.

DESCRIPTION

CALLING NWDCL node dcl_line output

ARGUMENTS

node Specification of nodes where the command has to be executed:

Node name A logical node name in the format
NODE[::]

List A distribution list reference in the format
@FILENAME

The distribution list must contain a list of logical node names in the above mentioned formant, one node per line.

Wildcard A node wildcard in the format
*

This is a synonym for the distribution list @NWDISTALL which contains a list of all nodes known.

dcl_line Any DCL command. If any delimiters are included, the command has to be enclosed in quotes ("). The command itself cannot contain quotes!

output Output file. If ommitted SYS$OUTPUT is used. Default extenstion is .LOG.
All output which is written by the command to SYS$OUTPUT will be directed to this file on the sending node.

REMARKS The command is executed in the context of a network process. Therefore only logical names and DCL symbols defined by your login procedures (SYSTEM login plus user LOGIN.COM) are known during execution. The only information passed from your session is the default directory and the command line !!

Example NWDCL * "SH USER"
Shows all users on all nodes

* NWDCL V8600A "CBAT ""DIR"" /LOG=DIR"
Executes on the node V8600A the procedure CBATCH which submits a batch job producing a directory listing on the file DIR.LOG.

NWDEFINE

NWDEFINE user paszwort

PURPOSE Define logical names for DECNET functions

ARGUMENTS

user Your user name

paszwort Your paszwort

DESCRIPTION

CALLING NWDEFINE user paszwort

ARGUMENTS

user Your user name

paszwort Your paszwort

REMARKS This procedure creates the file
LOGINNET.COM

in the login default directory and device. This command procedure contains the definitions of logical node names for DECNET accesses. The currently defined node names are:

NODE_A for V780A or ALICE
NODE_B for MVIII or BALDUIN
NODE_C for CLARA
NODE_D for V8600A or DONALD
NODE_E for V8600B or EMMA
NODE_K for MVIIA or KLAUS
NODE_N for MVIIB or NORA
NODE_O for MVIIC or OTTO
NODE_P for MVIID or PAULA
NODE_Q for MVIIE or QUARK

NODE_R for MVIIF or RITA
NODE_AA for VSAA or WSAA
NODE_AB for VSAB or WSAB
NODE_AC for VSAC or WSAC
NODE_AD for VSAD or WSAD
NODE_AE for VSAE or WSAE
NODE_AF for VSAF or WSAF
NODE_AG for VSAG or WSAG
NODE_AH for VSAH or WSAH
NODE_AI for VSAI or WSAI
NODE_AJ for VSAJ or WSAJ
NODE_AK for VSAK or WSAK
NODE_AL for VSAL or WSAL
NODE_AM for VSAM or WSAM
NODE_AN for VSAN or WSAN
NODE_PIG for PIGGY

The LOGINNET procedure is automatically executed by the standard login procedure GOOLOG.COM.

Example NWDEFINE myname secrect

NWDIFDIR

NWDIFDIR node file_spec

PURPOSE Compare a directory on a remote node with the local directory and create a list of differences.

ARGUMENTS

node Logical node name

file_spec Any wildcarded file name.

Description

CALLING NWDIFDIR node file_spec

ARGUMENTS

node Logical node name

file_spec Any wildcarded file name.

FUNCTION For all files matching file_spec on the remote node is checked, whether a namelike file (version may differ) exists on the local node. If not, the file name is listed in the format of a delete command.

REMARKS -

EXAMPLE -

NWDIRECT

NWDIRECT node dir_spec dir_qual

PURPOSE Compares directories on different nodes

ARGUMENTS

node Node name, distribution list or wildcard

dir_spec Any valid file specification for the DIRECTORY command

dir_qual Any valid selection qualifier for the DIRECTORY command

DESCRIPTION

CALLING NWDIRECT node dir_spec dir_qual

ARGUMENTS

node Specification of nodes whose directories have to be compared with the directory of the sending node:

Node name A logical node name in the format
NODE[::]

List A distribution list reference in the format
@FILENAME

The distribution list must contain a list of logical node names in the above mentioned formant, one node per line.

Wildcard A node wildcard in the format
*

This is a synonym for the distribution list @NWDISTOTH which contains a list of all nodes known except the own node.

dir_spec Any valid file specification for the DIRECTORY command, all wildcard allowed.

dir_qual Any valid selection qualifier for the DIRECTORY command, especially:

/SINCE=date
/BEFORE=date
/CREATE
/MODIFIED
/EXPIRED
/EXCLUDE=file_list

REMARKS This procedure produces a two column listing:
In the left column are all files listed which exsist on the host but not on the remote node.
In the right column are all files listed which exsist on the remote node but not on the host.

Example NWDIRECT * TEST.*

NWLIBRARY

NWLIBRARY node library file qual

PURPOSE Perform a library operation on one or more nodes

ARGUMENTS

node Node name, distribution list or wildcard

library Name of the library to be manipulated

file File containing modules to be replaced or inserted

qual Any LIBRARY qualifiers

DESCRIPTION

CALLING NWLIBRARY node library file qual

ARGUMENTS

node Specification of nodes on which the libraries have to be manipulated:

Node name A logical node name in the format
NODE[::]

List A distribution list reference in the format
@FILENAME

The distribution list must contain a list of logical node names in the above mentioned formant, one node per line.

Wildcard A node wildcard in the format
*

This is a synonym for the distribution list @NWDISTALL which contains a list of all nodes known.

library Name of the library to be manipulated

file	File containing modules to be replaced or inserted, all wildcards allowed.
qual	Any LIBRARY qualifiers
REMARKS	-
Example	NWLIBRARY NODE_A LIB TEST Will replace the object modules contained in the file TEST.OBJ in the library LIB.OLB on node NODE_A NWLIBRARY * LIB.MLB /LIST Will produce a listing of contents of the library LIB.MLB on all known nodes

NWUPDATE

```
NWUPDATE node file_list
         /DESTINATION=d
         /EXCLUDE=l
         /LOG/JOURNAL
         /SINCE=t/BEFORE=t
         /MODIFIED/CREATED/EXPIRED/BACKUP
         /REPLACE/OVERLAY/NEW_VERSION
         /GENERIC
```

PURPOSE Transfer a set of files to one or more nodes using the BACKUP utility.

ARGUMENTS

node	Node name, distribution list or wildcard.
file_list	List of file specifiers with any wildcards.
/DESTINATION=d	Output device on target node.
/EXCLUDE=l	List of file specifiers to be excluded.
/LOG	All file accesses are logged to SYS$OUTPUT.
/JOURNAL	The journal file SYS$LOGIN:NWUPDATE.BJL is updated.
/SINCE=t	Files manipulated after this date are selected.
/BEFORE=t	Files manipulated before this date are selected.
/MODIFIED	Files modified before or after the given date are selected.
/CREATED	Files created before or after the given date are selected.
/EXPIRED	Files expiring before or after the given date are selected.
/BACKUP	Files with a recorded backup date before or after the given date are selected.
/REPLACE	Existing files on the target node are replaced.

/OVERLAY Existing files on the target node are overlayed.

/NEW_VERSION A new version is created if a file already exists on the target node.

/GENERIC Select /REPLACE or /OVERLAY mode depending on file types.

DESCRIPTION

CALLING

```
NWUPDATE node file_list
        /DESTINATION=d
        /EXCLUDE=l
        /LOG/JOURNAL
        /SINCE=t/BEFORE=t
        /MODIFIED/CREATED/EXPIRED/BACKUP
        /REPLACE/OVERLAY/NEW_VERSION
        /GENERIC
```

ARGUMENTS

node Specification of nodes where the files have to be updated:

Node name A logical node name in the format
NODE[::]

List A distribution list reference in the format
@FILENAME

The distribution list must contain a list of logical node names in the above mentioned formant, one node per line.

Wildcard A node wildcard in the format
*

This is a synonym for the distribution list @NWDISTOTH which contains a list of all nodes known except the own node.

file_list List of file specifiers, all wildcards allowed. The list may contain file specifications with different devices but be aware that the output is always directed to a single device.

/DESTINATION=d Output device on target node. The default is SYS$LOGIN:[*...]

but note that SYS$LOGIN is translated on the target node.

For details of the processing of output directories refer to the BACKUP description.

/EXCLUDE=l List of file specifiers to be excluded. All types of wildcards are allowed but the file specifications must not contain a device name.

/LOG All file accesses are logged to SYS$OUTPUT.

/JOURNAL The journal file SYS$LOGIN:NWUPDATE.BJL is updated.

Input File Selection The SINCE and/or the BEFORE qualifier can be used with one of the qualifiers MODIFIED, CREATED, EXPIRED or BACKUP to select input files. If no qualifier is specified all files matching the file_list are selected, if only SINCE and/or BEFORE is specified the files are selected by modification date.

/SINCE=t Files manipulated after this date are selected.

/BEFORE=t Files manipulated before this date are selected.

/MODIFIED Files modified before or after the given date are selected.

/CREATED Files created before or after the given date are selected.

/EXPIRED Files expiring before or after the given date are selected.

/BACKUP Files with a recorded backup date before or after the given date are selected.

Output File Handling One of the qualifiers REPLACE, OVERLAY, NEW_VERSION or GENERIC can be used to determine the mode of output file restoration. If none of these qualifiers is specified, GENERIC is used as default.

/REPLACE Existing files on the target node are first deleted and than a new file is allocated.

/OVERLAY Existing files on the target node are overlayed. In this case the existing file is overwritten and the old file allocation is preserved. This may be advanterous in case of very large files.

/NEW_VERSION A new version is created if a file already exists on the target node.

/GENERIC Select /REPLACE or /OVERLAY mode depending on file types:
/OVERLAY for *.IDX, *.SEC
/REPLACE for *.*LB
none for all other file types

FUNCTION

On the sending node the BACKUP utility is used to create a saveset containing all selected files. This saveset is send to all specified target nodes, where the BACKUP utility is used to restore the files. Finaly the savesets are deleted.

REMARKS

NOTE: With the default restore options, a simple sequential file is not updated if it already exsists on the target node. In all cases except if /NEW_VERSION is used is the version of an updated file equal on sending and target node.

Example

NWUPDATE NODE_A TEST.PLI
Updates file TEST.PLI on node NODE_A

```
NWUPDATE NODE_A *.PLI,*.PPL
```

Updates all files with the extensions PLI and PPL

```
NWUPDATE * *.* /MOD/SINCE=YESTERDAY *.LOG
```

Updates all files modified since yesterday except all LOG-files on all other nodes.

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

PFKEY

PFKEY

PURPOSE Define terminal auxiliary keypad keys.

ARGUMENTS

Description

FUNCTION The procedure associates DCL commands to the auxiliary keypad

File name PFKEY.COM

Dataset -

Version 1.01

Author Th. Kroll, W. Spreng

Last Update 19-NOV-1985

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

PRIL

PRIL vaxfile switches

PURPOSE Sends VAX-file to IBM PRILTMP1.LIST and starts batch job calling PRIL on IBM (VAX-780 only)

ARGUMENTS

vaxfile VAX file name, default type : ".LIS"

switches /ROTATE (D) — /NOROTATE, /DUPLEX(D) — /NODUPLEX, /FLAG_PAGE(D) — /NOFLAG_PAGE, /ANSI — /NOANSI(D) /DELETE — /NODELETE

DESCRIPTION

CALLING PRIL vaxfile switches

ARGUMENTS

vaxfile VAX-File-Specification :
name.typ;version (n.t;v)
Wild cards are allowed in name,type or version

switches /ROTATE (D) — /NOROTATE, /DUPLEX(D) — /NODUPLEX, /FLAG_PAGE(D) — /NOFLAG_PAGE, /ANSI — /NOANSI(D) /DELETE — /NODELETE
(see HELP @UTIL PRIL switches)

FUNCTION This command procedure sends one ore more files to the IBM laser printer.
The VAX and the IBM information are parameters for the internal SUB BPRIL. This batch procedure will send the file(s) from VAX to IBM.
In case of the new software however BPRIL is executed as DCL procedure. For the IBM the default account number 'IBM_ACC' will be taken. If this is blank, the account number will be prompted. To define a default IBM-account number, set (in your LOGIN.COM):

IBM_ACC :== account
e.g. IBM_ACC :== PR99
The password for the IBM will be prompted unless it is already defined by the global symbol IBM_PW. For the destination to the laser printer the default 'IBM_LASER' will be taken. If this is blank, the destination will be prompted. To define the default destination, set
IBM_LASER:== destination
valid destinations are:
IBM_LASER:== UL
or IBM_LASER:== KP

REMARKS

Old software:
The command BPRIL should never be given directly!
The BPRIL must be executed in the batch queue SYS$IBM, which serializes the access of the link hardware. There will be a LOG-file SYS$LOGIN:BPRIL.LOG . These LOG-files will be purged by PRIL.

EXAMPLE

Switches

/ROTATE (Default) The text is printed on the rotated paper

/NOROTATE The text is printed normal

/DUPLEX (Default) The text is printed on both sides of paper

/NODUPLEX For each page a new sheet of paper is used

/DELETE The file is deleted after printing

/NODELETE (Default) The file is not deleted

/ANSI ANSI printer control characters of the file are used

/NOANSI (default) ANSI control characters are not used

PRILS

PRILS file /PROP

PURPOSE	prints VAX sources on the laser printer
ARGUMENTS	
file	script file
PROP	if specified text is printed in proportional font

Description

FUNCTION	To a prefix file, containing SCRIPT commands (GOO$DOC:PRILS.SCR), the specified file is appended and then processed with the SCRIBM command and a font profile.
REMARKS	All characters of the VT220 keyboard are transposed correctly except ~ and ‘.
File name	GOO$EXE:PRILS.COM
Dataset	-
Version	1.01
Author	K.Winkelmann
Last Update	22-NOV-1985 08-sep-1986 delete of temporary file SCR.TMP /KW 14-APR-1987 converted to usage of HYPERchannel /KW

PROMPT

PROMPT prompt-string default /REQUIRED help

PURPOSE	Prompt input from SYS$COMMAND
ARGUMENTS	
prompt-string	Prompt string enclosed in "".
default	Optional default value. If not specified, a null string is default. Answer value is in global symbol PROMPT_ANSWER.
/REQUIRED	Optional: A non empty value is required. A default must be specified, even if as empty string.
help	When specified, a ? as input calls HELP <help>

Description

FUNCTION The prompt-string and the default are used to compose the prompt string for a READ/PROMPT command. A colon is added. The answer is assigned to global symbol PROMPT_ANSWER. $STATUS is returned as 1. A <CTRL> Y or a <CTRL> Z set PROMPT_ANSWER == "", regardless of the default. A 3 is returned in $STATUS in that case.

Examples

EXAMPLE

```
$ PROMPT "Input value" "123"
  Input value (default=123) : <RET>
  ! now global symbol PROMPT_ANSWER has value 123.
```

EXAMPLE

```
$ PROMPT "Input value" "123"
  Input value (default=123) : <CTRL>Z
  ! now global symbol PROMPT_ANSWER has value "" and
  ! $STATUS is 3.
```

EXAMPLE

```
$ PROMPT "Input value"
  Input value : 15
  ! now global symbol PROMPT_ANSWER has value 15.
```

EXAMPLE

```
$ PROMPT "Input value"
  Input value : <RET>
  ! now global symbol PROMPT_ANSWER has value "".
```

EXAMPLE

```
$ PROMPT "Input value" "" /REQ
  Input value : 15
  ! now global symbol PROMPT_ANSWER has value 15.
```

EXAMPLE

```
$ PROMPT "Input value" "" /REQ
  Input value : <RET>
  Parameter is required. Enter value or break with <CTRL>Z:
  Input value : 1
  ! now global symbol PROMPT_ANSWER has value "1".
```

Version 1.01

Author H.G.Essel

Last Update 25-JAN-1988

RIBM

RIBM ibmds vaxfile
/COL=/CHECK=/MEMLIST=/$ALL/INTERACTIVE/NOANSWER

PURPOSE Send IBM dataset to VAX file.

ARGUMENTS

ibmds	Name of partitioned or sequential IBM dataset
vaxfile	Filename and extension of the file on the VAX
/MODE=	Transfer mode (CHAR or BIT) /MODE=BIT : transparent transfer /MODE=CHAR: EBCDIC - ASCII conversion (default)
/COL=	number of columns to be truncated (/COL=0: no truncation) Default: /COL=8
/CHECK=	number of characters to be tested to be digits Default: /CHECK=8
/$ALL	if the parameter is given, the member $ALL of the partitioned IBM dataset is used. It MUST exist and is assumed to contain the names of all members to be transfered.
/INTERACTIVE	interactive execution, default is batch
/NOANSWER	no answer from the IBM is sent back

DESCRIPTION

CALLING RIBM ibmds vaxfile

/COL=/CHECK=/$ALL/INTERACTIVE/NOANSWER

ARGUMENTS

ibmds IBM-Dataset-Specification :

name.type or name.type(member)

(e.g. PR99.ABC.TEXT(BLABLA) or XYZ.DATA)
Wild card characters are allowed in vaxfile.

vaxfile VAX-File-Specification :
name.typ or node"user pass"::device:[directory]name.typ
If no name is specified, the name of ibmds is used or in case of a partitioned ibmds the menber name is used. If no type is specified, the type of ibmds is used.

/MODE= Transfer mode (CHAR or BIT)
/MODE=BIT : transparent transfer
/MODE=CHAR: EBCDIC - ASCII conversion (default)

/COL= Number of columns to be truncated (/COL=0: no truncation)
Default: /COL=8.

/CHECK= Number of characters to be tested to be digits Default: /CHECK=8.

/$ALL If given, the member $ALL of the specified partitioned IBM dataset is used. It MUST exist and is assumed to contain the names of all members to be transfered. The VAX file name is then obsolete, the names of the members will be used as VAX file names.

/INTERACTIVE All transfers are interactive.
Default: batch execution

/NOANSWER No control output from the IBM will be sent back. Default: control output will be sent as mail to the invoking user.

FUNCTION If no directory is specified for the VAX-file, the default directory is assumed.
Sending several members of an partitioned IBM dataset to the VAX, the individual members will be put into VAX-files named [dir]membername.typ

On the VAX side the dataset is scanned for record numbers which will be removed (if not suppressed by /COL=0).

REMARKS IBM dataset and member MUST be available on the IBM. The IBM dataset MUST not be allocated by any IBM job!

It is assumed, that the IBM dataset name has always a filename and a type specification, separated by a '.'. If no account number is given,

the default account number stored in 'IBM_ACC' will be taken. If this is blank, the account number will be prompted.

To define a default IBM account number, set (in your LOGIN.COM):

```
IBM_ACC :== account
```

(e.g. IBM_ACC :== PR99).

If one or more additional '.' occur in the IBM dataset name, the account number must be specified as part of the dataset name.

The password for the IBM will be prompted unless it is already defined by the global symbol IBM_PW.

There will be a LOG-file RIBM.LOG on SYS$LOGIN. These LOG-files will not be purged by RIBM. The LOG-file contains information about the execution of the transfer on the VAX-side. Informations about the execution on the IBM-side are sent as mail from user NSC (if not suppressed by /NOANSWER).

Caution: Raw data (record format U on the IBM) cannot be transfered with this command. Use instead the DCL-Command RIRAW for this purpose.

Part of the functionality is put into BRIBM, which is submitted as batch job on the VAX (unless /INTER was given). The command BRIBM should never be given directly!

EXAMPLE

RIBM PROG.FORT(HALLO) HALLO.FOR

The member HALLO of the IBM-PDS PROG.FORT will be sent to the VAX file HALLO.FOR.

RIBM HALLO.FORT NODE::DEVICE:[DIR.SUBDIR]HALLO.FOR

The SEQUENTIAL IBM dataset HALLO.FORT will be sent to the VAX file HALLO.FOR in the specified subdirectory.

RIBM PROG.FORT [DIR.SUBDIR]*.*

All members of the partitioned IBM dataset PROG.FORT will be sent to the VAX. The individual PDS members will be put into VAX-files named [DIR.SUBDIR]membername.FORT

RIBM PROG.FORT [DIR.SUBDIR]HALLO.FOR /$ALL

The members listed in PROG.FORT($ALL) on the PARTITIONED IBM dataset will be sent to the VAX. The individual PDS members will be put into VAX-files named [DIR.SUBDIR]membername.FOR

RRUN

RR file/switches /switches

PURPOSE Runs programs. Defaults are updated.

ARGUMENTS

file Name of image file(def.=last)

switches Run 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 RRUN.COM

Dataset -

Version 1.01

Author H.Essel

Last Update 20-JAN-1984

SCRIBM

SCRI vaxfile [profile][edit]

PURPOSE Sends VAX-file to IBM SCRITEMP.TEXT and starts batch job calling SCRIPT on IBM

DESCRIPTION

REMARKS Due to the change of 6670 laser printers to IBM 3820/3812 laser printers this command is no longer supported! Use DCFIBM for DCF processing and output on IBM 38XX laser printers. For further help enter HELP DCFIBM.

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
```

SETSYM

SETSYM symbol value

PURPOSE Checks if symbol already exist and outputs message in this case. Sets symbol to value.

ARGUMENTS

symbol Name of symbol

value Name of another already defined symbol or string enclosed in triple " (sorry, DCL)

Description

FUNCTION Uses the F$TYPE function to check weather the symbol already exists. If yes, a message is printed. The symbol assignment is done.

NOTE Unfortunately the checking, weather the symbol exists takes a long time (90 ms). Therefore SETSYM is 6 times slower than assignment.

EXAMPLE

```
$ SETSYM X """ABC""" ! sets X == "ABC"
  $ SETSYM X Y ! sets X == Y
    Symbol X = ABC reset to Y ! output message
```

File name SETSYM.COM

Dataset -

Version 1.01

Author H.G.Essel

Last Update 31-JUL-1985

SIBM

SIBM vaxfile ibmds /MODE=
/MAILADDR=/NOANSWER/INTERACTIVE

PURPOSE Sends VAX-file(s) to IBM-dataset(s) and overwrites existing datasets or members

ARGUMENTS

vaxfile Filename and extension of the file(s) on the VAX

ibmds IBM pds(member) or sds name

/MODE= Transfer Mode (CHAR or BIT)

/MAILADDR= mailaddress for ibm batch output can be given

/NOANSWER no batch output from ibm is wanted

/INTERACTIVE interactive execution on VAX, default is batch

DESCRIPTION

CALLING SIBM vaxfile ibmds

ARGUMENTS

vaxfile VAX-File-Specification :
name.typ;version
Wild cards allowed in name and type, but only as many as in the IBM-Dataset-Specification.
Full VAX-File-Specification is possible like node"user pass"::device:[directory]name.typ;version

ibmds IBM-Dataset-Specification :
name.type or name.type(member)
or account.name.type or account.name.type(member)

e.g. PR99.ABC.TEXT(BLABLA) or XYZ.DATA Wild cards are allowed in name, type and member, but only as many as in the VAX-File-Specification.

/NOANSWER No answer from the submitted batch job on the IBM will be sent to the VAX

/MODE= Transfer Mode (CHAR or BIT), default is /MODE=CHAR

FUNCTION The existence of the VAX-file(s) is checked. The VAX and the IBM information are parameters for the internal BSIBM. This procedure will send the file(s) from VAX to IBM via the HYPERchannel. This command can be given on all VAXes at GSI. Existing sequential datasets or members of partioned datasets will be overwritten !!

>>>>>> Note:

If the Vaxfilename begins with leading numbers
e.g. 123TEST.PLI, the IBM filename will be changed to $123TEST.PLI and if necessary truncated to eight characters. If the parameter /MODE=BIT is used, the destination dataset must exist on the IBM and the Record Length must be compatible with that of the VAX.

REMARKS valid combinations with wild cards are: type and name (and both) can be wildcarded, further file name can be wildcarded and can become member name.

It is assumed, that the IBM-Dataset
name has always a dataset name and a type specification, separated by a '.'. If an additional '.' occurs, the first name part will be interpreted as the account number. If no account number is given, the default account number 'IBM_ACC' will be taken. If this is blank, the account number will be prompted. To define a default IBM-account number, set (in your LOGIN.COM):

IBM_ACC :== account

e.g. IBM_ACC :== PR99

The password for the IBM will be prompted unless it is
already defined by the global symbol IBM_PW.

The output from the IBM batch job will be sent
back to the calling VAX into the user's mail.

EXAMPLE SIBM *.TXT ADAM.TEXT(*)

All VAX-files with the type .TXT will be send to
the IBM PDS dataset ADAM.TEXT. The members will have the same name as the VAX files. SIBM HALLO.FOR HALLO.FORT

The FORTRAN VAX-file HALLO.FOR will be send to the

sequential dataset HALLO.FORT on the IBM. SIBM XY$ROOT:[EVE]ADAM.PLI XY01.ADAM.PLI

The PL/I VAX-file ADAM.PLI of the VAX directory XY$ROOT:[EVE]will be send to the sequential dataset ADAM.PLI under the account XY01 on the IBM.

SIBMGKS

SIBMGKS metafile /PLOTTER=p

PURPOSE	Output GKS-Metafile on Plotter RP01,RP02
ARGUMENTS	
metafile	name of GKS metafile to be plotted on an plotter at the IBM.
/PLOTTER=p	p is name of IBM-Plotter, maybe RP01 = LARGE BENSON RASTERPLOTTER RP02 = SMALL BENSON RASTERPLOTTER

Description

FUNCTION	1) Check GKS-Metafile a) SYNTAX b) EXISTENCE 2) Find IBM-userid and Password 3) Create JCL for IBM 4) Setup IBMSUBMIT
Version	1.01
Author	Petra Kohn
Last Update	6-FEB-1987

SIBMSPEC

SIBMSPEC datafile VSAMlib /RUNID=runid /TIME=
/NOL*IST

PURPOSE Dump GOOSY spectra to SATAN VSAM library

ARGUMENTS

datafile Name of datafile with spectrum data to be transfered. This file must be generated by GOOSY command DUMP SPECTRUM. If the datafile-name is not fully qualified the file is searched on the default directory first and next in the login directory.

VSAMlib SATAN VSAM library name for dump to SATAN analyzer library (default).
>>>> The IBM dataset must exist !! <<<<

/RUN*ID=run Run identification for a dump to a SATAN analyzer library.

/TIME=jobtime Time parameter for IBM batch job in seconds

/NOL*IST suppress listing of spectra characteristics
contained in the datafile

DESCRIPTION

FUNCTION This command procedure will dump GOOSY spectra to a SATAN VSAM library . The IBM datasets must exist. The SYS$IBM batch queue is used, to serialize the access to the link hardware. The transfer will be carried out per FTP.

REMARKS The default IBM account name is 'IBM_ACC'. If this is blank or not set, the account name will be prompted. To define a default IBM account name, set (preferably in your LOGIN.COM) IBM_ACC :== account. Correspondingly, the password will be prompted unless it is alreay defined by the global symbol IBM_PW. /NOLIST speeds up the execution of SIBMSPEC on the VAX significantly!. Temporary files are created on the default directory.

EXAMPLE	SIBMSPEC SCALER.DATA SPECTRA The spectra contained in the file 'SCALER.DATA' are transfered to the SATAN VSAM library 'SPECTRA' which must exist. SIBMSPEC SCALER.DATA SPECTRA /TIME=90 The spectra contained in the file 'SCALER.DATA' are transfered to the SATAN VSAM library 'SPECTRA' which must exist. The maximum jobtime for the IBM batch job is 90 seconds.
File name	GOO$EXE:SIBMSPEC.COM
Dataset	-
Version	2.01
Author	D.Schall

SSYMBOL

SSYM [<name>][/SEARCH=<string>]

PURPOSE Displays symbol translation

ARGUMENTS

name Name of symbol (optional). If not specified, the complete symbol list is displayed. Name may contain wildcard.

/SEARCH Displays all symbol names and their equivalence strings containing <string>.
NOTE: name must be * (DCL syntax!)

DESCRIPTION

FUNCTION Calls SHOW SYMBOL <name> or SHOW SYMBOL/GL/ALL, in case of wildcard PLI - Procedure MSHOSYM will be called.

File name SSYMBOL.COM

Dataset -

Version 1.01

Author H.G.Essel,T.Kroll

Last Update 1-JUN-1984
22-FEB-1985 wildcard supported
27-FEB-1985 spawn in dcl-procedure umgewandelt
24-jun-85 Added SORT pass /H.E.
01-jul-85 Added SEARCH fac. /H.E.
08-jan-87 Renamed CSHOWSYM to SSYMBOL /H.E.
20-Jul-88 MSHOSYM no longer needed /HE

SWSIZE

SWSIZE proc.-name /W*SMIN=min/S*TATUS=status/R*EPEAT=n

PURPOSE	Show processes with their workin set sizes.
ARGUMENTS	
proc.-name	optional name of processes. All process names containing this string, are displayed.
/WSMIN=	Optional minimum working set size
/STATUS=	Optional process status to be selected. i.e. NET
/REPEAT=	Optional number of repetitions

Description

FUNCTION	Caller must have WORLD or SETPRV
Version	1.01
Author	H.G.Essel
Last Update	9-MAR-1988

TDOCUMENT

```
TDOC*UMENT file /TOC /INDEX /GOOSY /REP*ORT
        /REL*EASE=r /VER*SION=v
        /TITLE=tt /AU*THORS=a
        /LABEL=lt /LOGO=ll
```

PURPOSE	Format TeX source, e.g. produced by GLFORMAT of GLDOC, in the GOOSY document style
ARGUMENTS	
file	file containing TeX source. Type default is .TEX
/TOC	produce table of contents (Execute LATEX two times)
/INDEX	produce index table (Execute LATEX three times)
/GOOSY	Produce GOOSY documentation
/REPORT	Use 'report' style instead of 'article'
/RELEASE=r	release number (def.=1)
/VERSION=v	version number (def.=0)
/TITLE=t	produce titlepage. Specify authors too.
/AUTHORS=a	list of authors to appear on title. Required if title is specified.
/LABEL=l	produce a folderlabel
/LOGO=ll	text of folderlogo

Description

CALLING

```
TDOC*UMENT file /TOC /INDEX /GOOSY
        /REL*EASE=r /VER*SION=v
        /TITLE=tt /AU*THORS=a
        /LABEL=lt /LOGO=ll
```

ARGUMENTS

file	file containing TeX source. Default type .TEX.
/TOC	produce table of contents (Execute LATEX two times)
/INDEX	produce index table (Execute LATEX three times)
/GOOSY	Produce GOOSY documentation
/RELEASE=r	release number (def.=1) written on running footnote
/VERSION=v	version number (def.=0) written on running footnote
/TITLE=t	produce titlepage. Specify authors too.
/AUTHORS=a	list of authors to appear on title. Required if title is specified.
/LABEL=l	produce a folderlabel, which will fit on a plastic folder type LEITZ 1015.
/LOGO=ll	text of folderlogo
FUNCTION	Format a TeX text body using the GOOSY document style. The TeX source may not contain declarations like documentstyle, pagestyle, end begin/end{document}.
REMARKS	-
EXAMPLE	TDOC abc /TITLE=title/AUTH=(A,B,C) produce abc.DVI with titlepage

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;
```

VMS_Primer

Short introduction for VAX users at GSI.

Related documents "Common DEC-IBM PLI Standard" by W.F.J.Mueller and H.G.Essel
"GOOSY Conventions (Standards)" by H.G.Essel

Most terminals are connected through a LAT (Local area transport) and DECnet to any node available. The LAT itself echos a <RET> with the prompt 'Local>'. If it prompts with 'Enter username>' , enter your name. Then 'Local>' appears. Now the LAT accepts several commands:

SHOW SERVICE List of available services (nodes)

CONNECT <service> Create session at specified service (node). The node prompts with Username: and Password:

SHOW SESSIONS List of sessions presently controlled by that terminal.

RESUME # Connect terminal to a session specified by number.

HELP List of LAT commands

One can return to the LAT pressing <VA> (or <BREAK>) key and create several sessions (max=4). With <CTRL><backslash> one can switch the terminal from one session to another (equivalent to 'Local> FORWARD' command). The following chapters give you more information:

DCL Some very brief description of DEC's command language.

Information How to get any kind of information.

VMS First intro to devices, directories, logical names, and libraries.

Utilities Useful commands for daily work, programming hints etc.

Symbols List of command symbols defined by standard login procedure.

Programming Some hint for program development

Network Some hints for networking

DCL

In DCL (DEC Command Language) all keywords may be abbreviated as long as they are unique. The interface (CLI) interprets only the first four characters anyway. Programs usually can be terminated by <CTRL>Z, if that doesnt work, by <CTRL>Y . Parameter lists for DCL commands or procedures are converted to uppercase. If lower case parameters are required, they must be enclosed in quotes (").

A command line may be continued in the next line appending a space and a hyphen at the end of the previous line.

DCL command procedures (equ. to IBM CLIST) are executed by
@<filename>.

DCL procedures can be executed as batch jobs without modifications by
SUBM <filename>.

SYNTAX DCL commands may have several keywords, e.g. SHOW DAYTIME. They can have positional parameters delimited by spaces and qualifiers. Qualifiers may have a value. They are preceded by a slash, e.g. DIRECT/SIZE/DATE/SINCE=YESTERDAY. Some commands distinguish between command qualifiers and parameter qualifiers. Qualifier values can be a list of values enclosed in apostrophe's delimited by commas, e.g. PLI X.PLI/SHOW=(INCL,TERM)

SYMBOLS For convenience, DCL commands or parts can be assigned to a symbol by

```
symbol:==command string
DIRP:==DIRECT/SIZE/DATE/OWN/PROT
```

DIRP *.PPL list now all PPL sources with date, size, owner and protection. A set of symbols are defined by the standard login procedures (see SYMBOLS). Others may be added in the user login procedure LOGIN.COM.

Information

This chapter helps you to get information about anything on our VAX. It tells you which manuals are suited for a first jump in. The interactive methods to get information online are described.

Usually all information can be accessed by the HELP command which is implemented in most VMS utilities.

Manuals

We think it makes sense to look through these manuals first to get a rough idea about VMS.

'Guide to Using Command Procedures' :
Get an overview over the lexical functions (functions callable in DCL; HELP LEXICAL)

'Command Language User's Guide' :
(PART I, App. A)

'System Services Reference Manual' :
(PART I) Describes several useful VMS features and routines to use them
(HELP SYSTEM)

'Runtime Library Reference Manual' :
(Introduction) Summary of many useful routines (HELP RTL)

'Utilities Reference Manual' :
The following Chapters may be of special interest:
LIBRARIAN (10)
MESSAGE (11)
MONITOR (12)
MAIL (13)
PHONE (14)

'Record Management Services' :
(Introduction) General description of VMS file system.

Help

The HELP command provides access to all information kept in help libraries. A table of content of these libraries can be obtained by

HELP @<library>

All libraries of the list are searched for a keyword by

HELP <keyword>

To search a certain key in a certain library, type

HELP @<library> <key>

These libraries can be created by users and put into default lists. Presently we have the following libraries besides the system libraries:

MODULE GOOSY subroutines

PROGRAM	GOOSY main programs
COMMAND	GOOSY commands
UTILITY	DCL command procedures
DMTYPES	GOOSY data management data types (type descriptors)
MESSAGE	GOOSY error messages
INCLUDE	Text of all PLI include files
RTLINC	Text of all RTL include files
RECOVER	Error recovery for GOOSY errors
GIPSY	GIPSY interpreter
PPLMAC	GOOSY preprocessor macros
VPW	Fermilab desktop utility
CERN	CERN library
DMODLIB	Subset of DARSY modules.
GSIHELP	Some common utilities (e.g. login procedures, symbols)
XUTILITY	Internal DCL procedures (system programmers only)
XMODULE	Internal PLI procedures (system programmers only)
MANAGER	GOOSY system manager information

User

The following commands may help you to get information about resources.

NEWS	Displays actual information. Add NEW/LOG in your LOGIN.COM to get a short notice about new information. (This is done in GSILOG.COM which is the default GSI login procedure) (Type HELP NEWS for a full description).
SPRO	SHOW PROCESS Displays process information
SPROC	SHOW PROCESS/CONT Displays process information continuously.

DIRS	DIRECTORY/SIZE=ALLOCATE/DATE
DIRP	DIRECTORY/SIZE/DATE/OWNER/PROTECTION
SDEF	SHOW DEFAULT Displays your current directory default.
SDEV	SHOW DEVICE Displays a list of all devices on the VAX. SDEV/FULL <name> gives detailed information.
SLOG	SHOW LOGICAL Displays translation of a logical name (if specified) or the content of all logical name tables (PROCESS, JOB, GROUP, SYSTEM). SLOG/PROCESS SLOG/SYSTEM SLOG/GROUP SLOG/JOB
SSYM	Displays translation of a symbol (if specified) or the content of the process symbol table. (type HELP CSHOW for full description)
EARN	To get EARN node addresses (HELP EARN)
SPSI	To get X25 addresses
H @GSI	
H @UT GSI	
H @U SYMDEF	
H @U LOGIN	These help commands give you information about more command symbols defined for a lot of purposes.
H @U VMS	This primer

System

The following commands are special utilities to get information about system activities.

SUSE	SHOW USER Displays list of interactive users.
SBAT	SHOW QUEUE/BATCH/ALL
SPRI	SHOW QUEUE/DEVICE/ALL
SIN	Displays any system data by WHAT.
CWHAT	Calls WHAT with several selectable items (HELP CWHAT)

SMEM Displays memory resources.

WATCH (HELP @DMODLIB WATCH)
Displays total information about a process.

MONITOR See 'utilities reference manual' and HELP MONITOR.

MCPU	MONITOR PROCESS/TOPCPU Display top CPU users.
MBIO	MONITOR PROCESS/TOPBUF Display top buffered I/O users.
MDIO	MONITOR PROCESS/TOPDIR Display top direct I/O users.
MFAULT	MONITOR PROCESS/TOPFAULT Display paging activity.
SCLU1	display cluster data (exit with ^ Y)
SCLU2	display cluster data (exit with ^ Y)

Login

If a user logs into the system, several command procedures are executed.

GSI$MANAGER :GSISYLOGIN.COM
(HELP @GSI GSISYLOG)

GSI$MANAGER :GSILOG.COM
(HELP @GSI GSILOG)

GOO$EXE :GOOGSILOG.COM
(HELP @UTIL GSILOG)

LOGIN.COM Userwritten login procedure. GOOSY programmers must call GOO$EXE:GOOLOG.COM in their LOGIN.COM.

GOO$EXE :GOOLOG.COM
(HELP @UTIL LOGIN)
This procedure calls:

GOO$EXE :GOOSYMDEF.COM
(HELP @UTIL SYMDEF)

GOO$EXE :GOOLIBDEF.COM
(HELP @UTIL LIBDEF)

VMS

This part gives on overview over the organisation of our VAX.

Nodes

Presently we have the following VAXes in our DECnet-ethernet. The nodes are:

ALICE (V780A) VAX 780 in the room besides the UNILAC control room.

BALDUIN (V750A) VAX 750 in the electronic departement.

CLARA (V730A) VAX 730 of the SIS group

DONALD (V8600A) VAX 8600 in the "Mess-Station" (Cluster "GOOSY")

EMMA (V8600B) VAX 8600 in the "Mess-Station" (Cluster "GOOSY")

Devices

Presently the following disks and tapes are installed. The names beginning with "_" are physical names, others are logical names.

ALICE

D0	_DRA0: Type RM05, 256 MB, Pack.
D1	_DRA1: Type RM05, 256 MB, Pack.
D2	_DUA0: Type RA81, 470 MB, winchester
D3	_DUA1: Type RA81, 470 MB, winchester
M0	_MTA0: TU78 tape, 1600/6250 BPI

BALDUIN

D0	_DUA0: Type RA81, 470 MB, winchester
D1	_DUA1: Type RA81, 470 MB, winchester
M0	_MUA0: Streamer tape 1600/6250 BPI

DONALD Disks are shared with EMMA (HSC)

D0	HSC000$DUA0: Type RA81, 470 MB, winchester
D1	HSC000$DUA1: Type RA81, 470 MB, winchester

D2	HSC000$DUA2: Type RA81, 470 MB, winchester
D3	HSC000$DUA3: Type RA81, 470 MB, winchester
D4	HSC000$DUA4: Type RA81, 470 MB, winchester
M0	HSC000$MUA0: TU78 tape, 1600/6250 BPI
M1	HSC000$MUA1: TU78 tape, 1600/6250 BPI
M2	HSC000$MUA2: TU78 tape, 1600/6250 BPI
M3	HSC000$MUA3: TU78 tape, 1600/6250 BPI
EMMA	Disks are shared with DONALD (HSC)
D0	HSC000$DUA0: Type RA81, 470 MB, winchester
D1	HSC000$DUA1: Type RA81, 470 MB, winchester
D2	HSC000$DUA2: Type RA81, 470 MB, winchester
D3	HSC000$DUA3: Type RA81, 470 MB, winchester
D4	HSC000$DUA4: Type RA81, 470 MB, winchester
M0	HSC000$MUA0: TU78 tape, 1600/6250 BPI
M1	HSC000$MUA1: TU78 tape, 1600/6250 BPI
M2	HSC000$MUA2: TU78 tape, 1600/6250 BPI
M3	HSC000$MUA3: TU78 tape, 1600/6250 BPI

Pseudo-devices

These are logical names for directories, which may be used like disk names. An actual list can be optained by SLOG *$ROOT

GOOSY cluster

"ACC$ROOT" = "D5:[ACC.]"
"BACK$ROOT" = "D5:[GOOBACKUP.]"
"CAD$ROOT" = "D4:[CAD.]"
"CERN$ROOT" = "D1:[CERN.]"
"DAY$ROOT" = "D4:[DAY.]"
"DECUS$ROOT" ="D1:[DECUS.]"
"EE$ROOT" = "D2:[EE.]"
"GKS$ROOT" = "D1:[GKS.]"

```
"GOO$ROOT" = "D1:[GOOSY.]"
"KC2$ROOT" = "D4:[KC2.]"
"KP2$ROOT" = "D2:[KP2.]"
"KP3$ROOT" = "D2:[KP3.]"
"US0$ROOT" = "D0:[US0.]"
"US1$ROOT" = "D1:[US1.]"
"US2$ROOT" = "D2:[US2.]"
"US3$ROOT" = "D3:[US3.]"
"US4$ROOT" = "D4:[US4.]"
"US5$ROOT" = "D5:[US5.]"
"UTIL$ROOT" = "D0:[UTIL.]"
"WA80$ROOT" = "D2:[WA80.]"
"WA80D$ROOT" ="D3:[WA80D.]"
```

ALICE

```
"BACK$ROOT" = "D3:[GOOBACKUP.]"
"CAD$ROOT" = "D0:[CAD.]"
"CERN$ROOT" = "D3:[CERN.]"
"DECUS$ROOT" ="D3:[DECUS.]"
"EE$ROOT" = "D3:[EE.]"
"GKS$ROOT" = "D0:[GKS.]"
"GOO$ROOT" = "D3:[GOOSY.]"
"NODEC$ROOT" ="D2:[NODEC.]"
"US0$ROOT" = "D0:[US0.]"
"US1$ROOT" = "D1:[US1.]"
"US2$ROOT" = "D3:[US2.]"
"UTIL$ROOT" = "D3:[UTIL.]"
"WA80$ROOT" = "D3:[WA80.]"
"WA80D$ROOT" ="D3:[WA80D.]"
```

BALDUIN

```
"CERN$ROOT" = "D1:[CERN.]"
"DAT1$ROOT" = "D0:[DAT1.]"
"DECUS$ROOT" ="D1:[DECUS.]"
"EE$ROOT" = "D1:[EE.]"
"GKS$ROOT" = "D0:[GKS.]"
"GOO$ROOT" = "D1:[GOOSY.]"
"KP3$ROOT" = "D1:[KP3.]"
"US0$ROOT" = "D0:[US0.]"
"US1$ROOT" = "D1:[US1.]"
"US2$ROOT" = "D1:[US2.]"
"UTIL$ROOT" = "D0:[UTIL.]"
```

Directories

There are several logical names for directories.

SYS$MANAGER	Site specific command procedures (system).
SYS$SYSTEM	Programs and images
SYS$HELP	Help libraries
SYS$LIBRARY	Libraries
SYS$LOGIN	Users default directory.
DAR$LIBRARY	DARSY library.

The following directories are specific to GOOSY sources. The names correspond to the utility names.

GOO$CMD	GOOSY command interface
GOO$DM	GOOSY data management
GOO$IO	GOOSY I/O routines
GOO$MSG	GOOSY message files
GOO$SYS	GOOSY system routines
GOO$UTIL	GOOSY general utilities
GOO$EXE	Command procedures and executable programs.
GOO$LIB	GOOSY libraries

Logical Names

Logical names are entries in logical name tables. There are four default tables (LNM$SYSTEM, LNM$GROUP, LNMJOB, LNMPROCESS), and additional tables may be created (see CREATE/NAME_TABLE command). Users should add entries only to their JOB -, PROCESS- or private tables (normally they are restricted by privileges), because GROUP table is used by all processes of the group (same UIC) and SYSTEM by all processes.

Related commands	**DEFINE**	Defines a new logical name in different tables. DEF/PROC <new name> <string> DEF/GROUP <new name> <string> DEF/SYSTEM <new name> <string>

DEASSIGN Deletes a logical name
DEAS <name>

Usage Logical names are used for devices, directories, files, nodes and libraries.

A full list of logical names defined can be obtained by SLOG. Examples of logical names predefined by VMS:

TT	Terminal.
SYS$INPUT	interactive: TT, batch: DCL-file, DCL procedure: DCL-file
SYS$COMMAND	interactive: TT, batch: DCL-file, DCL procedure: TT
SYS$OUTPUT	interactive: TT, batch: Logfile, DCL procedure: TT
SYS$ERROR	interactive: TT, batch: Logfile, DCL procedure: TT
SYS$LOGIN	Users default directory.
SYS$DISK	Users deafult disk

Libraries

There are libraries of type TEXT, HELP, OBJECT and MACRO. All these libraries are handled by the command LIBRARIAN (HELP LIB, 'Utilities Reference Manual'). HELP-, INCLUDE-, and OBJECT-libraries can be put into library lists for the HELP utility, PLI or the linker, respectively. This is done by defining the following logical names (SYSTEM, GROUP or PROCESS):

```
DEFINE LNK$LIBRARY <library>
DEFINE LNK$LIBRARY_1 <library>
DEFINE LNK$LIBRARY_n <library>
DEFINE HLP$LIBRARY <library>
DEFINE HLP$LIBRARY_1 <library>
DEFINE HLP$LIBRARY_n <library>
DEFINE PLI$LIBRARY <library>
DEFINE PLI$LIBRARY_1 <library>
DEFINE PLI$LIBRARY_n <library>
```

The searching order is PROCESS, GROUP, SYSTEM logical name table.

Help Presently we have in the SYSTEM table several libraries (defined in SYS$MANAGER:GSIHELPS.COM). A list of these libraries can be obtained by the HELP command or by typing SLOG HLP$*.
The HELP command searches through all these libraries.

Object	Similar the object libraries are defined. Presently we have some libraries in the GROUP table (defined in GOO$EXE:GOOLIBDEF.COM). A list of the link libaries may be obtained by typing SLOG LNK$* The linker searches through all these libraries.
Text	The CPLI command (used in COMPILE) adds PLI-text libraries as defined by PLI$LIBRARYxx to the PLI command. A list of the PLI text libaries may be obtained by typing SLOG PLI$* The CPLI command searches through all these libraries.

Batch

There are several batch queues on our VAX. Command symbols to start, cancel and list jobs are provided (HELP @GSI BATCH).

Utilities

Utilities

A full description of any of the following command symbols may be obtained by HELP @UTIL <command> or just HELP <command>. Further hints may be obtained by NEW/ALL/BR and NEW <number>.

CPLI*COM	Compile PLI modules with text libraries GOORTL and GOOINC. (default = last)
COM*PILE	Generic compile command. (default = last)
CL*INK	Link program (def. = last)
RR*UN	Run program (def. = last)
CED*ITDOC	Generate interactively documentation header
GLxxx	Program management commands
GTOOL	Interactive menue driven module management.
SIBM/RIBM	Send/receive files to/from IBM (GOOSY cluster and ALICE only)
CREM*NUM	Remove line numbers from IBM-sources
IBM	connect VT100 to IBM (ALICE only)
SCR*IBM	Send SCRIPT file to IBM and print via SCRIL

DOC*IBM	Send SCRIPT file to IBM and print via DOC
MDCLA*NAL	Debug or list DCL procedures (including all levels).
LSE	Language sensitive editor.
ECL	Execute a DCL line in a loop with different arguments
LIBT*YPE	Type or print a text library module
LIBLIS	Type or print library directory
CBATCH	Execute single DCL command in batch.
SEARCH	searches through specified files for a given string.
PARA*LLEL	outputs two files in two columns.
DTC	Desk top calendar
REM*INDER	Reminder for appointments.
CFILT*YPE	Output list of all file types.
CMAIL	Sending mails (ARPA, EARN, PSI)
DVIIBM	Send TEX-DVI file to IBM for processing
PRIL	Print file on laser printer
CREP*LACE	Replace one string by another in a set of files.
System service	Bunch of VMS system routines (HELP SYSTEM)
RT_libr.	Bunch of Run time library routines (HELP RTL)

DCL procedures

We recommend to follow these rules writing DCL procedures:

1. Assign or define local symbols for P1...P8
2. Prompt for requested parameters (provide defaults).
3. Emulate IBM parameter list syntax using CDCLLIST (HELP @U CDCL), if wanted. Otherwise emulate DCL parameter list syntax using MDCLLIST (HELP @P MDCLL).
4. Provide actions for <CTRL>Y and errors.

5. Provide one label for all exits

6. Use unique names for temporary files (and clean up).

7. Provide default file types for file parameters.

8. Support logical names as input for file names.

9. Support wildcards and container files (@<file spec.> instead of files name) as input for file names.

10. Define a symbol for calling the procedure including full directory specification. .BR E.g. TEST:==@GOO$EXE:TEST.COM .BR These symbols must be defined in the file GOO$EXE:GOOSYMDEF.COM (GOOSY special) or GOO$EXE:GOOGSILOG.COM (general). This is done by the command CGOODEF (HELP @U CGOODEF).

Programs

We recommend to follow these rules writing main programs:

1 Implement program execution by a symbol. .BR E.g. TEST:==GOOEXE:TEST.EXE .BR These symbols must be defined in the file GOO$EXE:GOOSYMDEF.COM (GOOSY special) or GOO$EXE:GOOGSILOG (general). This is done by command CGOODEF.

2. Use logical names for I/O devices and files.

3. Read parameters by LIB$GET_FOREIGN. .BR (HELP RTL LIB$ GET_FOREIGN) .BR If the program is called by a DCL symbol defined as above (TEST:==GOOEXE:TE the LIB$GET_FOREIGN returns the string <string>, if you run the program by TEST <string>.

4. Write DCL procedures to do necessary assignments and call program.

5. Follow programming and documentation conventions defined for GOOSY software.

Programming

You may get some usefull hints about programming standards in the paper "GOOSY Conventions (Standards)" by H.G.Essel.

Editor

We recommend the language sensitive editor LSEDIT invoced by LSE <filename>. Besides the screen editing facilities there are the following keys of interest:

PF2 Help key
DO enter command mode (type HELP)
PF1 <=> enter/leave split screen mode
F20 switch window in split screen mode
F17 test compile actual file
F9 read another file for editing
F10 select another file already read for editing
F7 Enter VMS HELP
F11/F12 shift text left/right
PF1 F7 sho key functions
F18 Execute DCL command in a spawned process

To leave the editor, press DO-key and type EXIT<RET> to exit, QUIT<RET> to quit (no modifications).

Compiler

All compilers are invoked by COMP <filename>, depending on the file type. Standard file types are:

.PLI = PLI sources
.PPL = GOOSY PLI preprocessor sources
.FOR = Fortran sources
.MAR = VAX Assembler sources
.PAS = Pascal sources
.PLITEMP = temporary PLI-source generated by GOOSY preprocessor.

The individual default type for COMPILE can be set by DEFCOMPI:==<type> in the login procedure.

The COMPILE command is available on IBM for GOOSY preprocessor code.

Include modules for PLI programs should be inserted in text libraries. Logical names PLI$LIBRARY_n must be assigned to these libraries. Then the PLI-statement %INCLUDE <module> searches the module in all libraries defined by PLI$LIBRARY_n.

If a file has been compiled already and the source has not changed, it is NOT compiled until the /COM qualifier is specified. Thus one can compile a set of files in a command procedure. Only modified files are compiled saving time.

Linker

The linker searches all object libraries defined by LNK$LIBRARY_n.

Debugger

The VMS debugger is a very powerful tool to find errors. It allows to set break points on source lines, step line by line, inspect and set variables etc. Modules to be debugged must be compiled with the /DEB option. The main program must be linked with the /DEB option. Then the debug mode is entered after program start. Type HELP to get more information.

PLI-preprocessor

If PLI-modules are written to run on both, IBM and VAX, it is strongly recommended to use the PLI-preprocessor. It is implemented on both systems and allows to handle the differences between VAX-PLI and IBM-PLI. The standard file/dataset type is .PPL. The COMPILE command calls the preprocessor and then the PLI-compiler. In case of errors the PLI-code is preserved in files named <name>.PLITEMP. The simplest way to maintain two PLI-versions in one file is to mark single lines or blocks of lines with tagwords. Tagwords are specified with the /PRE=(tagword list) qualifier. Then only lines marked with these tagwords are processed. By this method certain lines may be compiled only on IBM, others only on VAX. The standard tagwords are V for VAX code and I for IBM code. V is default on VAX, I on IBM. HELP MPRECOM for more details. For a more detailed discussion about PLI differences see paper "Common DEC-IBM PLI Standard" by W.F.J.Mueller and H.G.Essel.

Networking

On node EMMA there are links to EARN (DDAGSI5) and X25 (GSI_VAX).
More information with HELP EARN, HELP PSI, HELP @GSI X25, HELP CMAIL
On GOOSY cluster (EMMA or DONALD) and ALICE files may be exchanged with IBM.
More information with HELP RIBM and HELP SIBM. On ALICE one can LOGON on the IBM (Command IBM)

EARN

PSI (X25)

IBM-VAX File transfer

The RJE-entry is used to transfer files between VAX and IBM. Transfers are initiated from VAX by commands RIBM and SIBM. Transfers of complete PDS's to VAX must be initiated from IBM. See HELP SIBM or HELP RIBM.

Symbols

Symbols to keep information
  $MODE_BATCH = 0
  $MODE_INTER = 1
  $MODE_NET = 0
  $MODE_OTHER = 0
  $RESTART = "FALSE"
  $SEVERITY = "1"
  $STATUS = "%X00030001"
  COMTMP = "M$LNKDL"
  DEFCOMPI = "PPL"
  EDTTMP = ""
  IBM_ACC = "PR01"
  LNKTMP = "FORLIS"
  PLITMP = "M$LNKDL.TMP"
  PORT_MANAGER = "F"
  SCR_DEST = "UL"
  TERMINAL_FLAVOR = "VT220"
  TERMINAL_TYPE = "VT100"
  TERMINAL_UNIT = "LTA29:"
DCL Commands
  BYE = "LO"
  COP*Y = "COPY/LOG"
  DEL = "DELETE/LOG"
  DTMP = "DELETE *.TMP;*"
  H*ELP = "HELP/PAGE/NOINSTRUCTIONS"
  HOSTV*AX = "SET HOST /X29 /VMS_MODE"
  MBIO = "MONITOR PROCESS/TOPB"
  MCPU = "MONITOR PROCESS/TOPC"
  MDIO = "MONITOR PROCESS/TOPD"
  MFAU*LTS = "MONITOR PROCESS/TOPF"
  MFCP = "MONITOR FCP"
  MIO = "MONITOR IO"
  MLOC*K = "MONITOR LOCK"
  MMOD*E = "MONITOR MODE"
  MNET = "MONITOR DECNET"
  MPAG*E = "MONITOR PAGE"
  MPOO*L = "MONITOR POOL"
  MSTA*TE = "MONITOR STATE"
  NOV*ERIFY = "SET NOVERIFY"

```
    PUR*GE = "PURGE/LOG"
    SPROC = "SHOW PROCESS/CONT"
    SPSI = "SHOW LOGICAL/TABLE=PSI$DTE_TABLE"
    TOD*AY = "DIR/SIZE=ALL/DATE/SINCE=TODAY"
    V*ERIFY = "SET VERIFY"
  Utility commands
    CADJ*UST = "@GOO$EXE:CADJUST.COM"
    CALC = "$CERN$ROOT:[CERNEXE]CALC"
    CBAT*CHDCL = "@GOO$EXE:CBATCHDCL.COM"
    CBIN*HEX = "@GOO$EXE:CBINHEX.COM"
    CHEX*BIN = "@GOO$EXE:CHEXBIN.COM"
    CFILT*YPES = "@GOO$EXE:CFILTYPES.COM"
    CL*INK = "@GOO$EXE:CLINK.COM"
    CMAI*L = "@GOO$EXE:CMAIL.COM"
    CNAMELIST = "@GOO$EXE:CNAMELIST.COM"
    COM*PILE = "@GOO$EXE:COMPILE.COM"
    CPU*RGE = "@GOO$EXE:CPURGE.COM"
    CREPE*AT = "@GOO$EXE:CREPEAT.COM"
    CREPL*ACE = "@GOO$EXE:CREPLACE.COM"
    CWH*AT = "@GOO$EXE:CWHAT.COM"
    DIFRAG = "$UTIL$ROOT:[DECUS.TOOLS]FRAG"
    DIRMAP = "@UTIL$ROOT:[DECUS.TOOLS]DIRMAP"
    DOC*IBM = "@GOO$EXE:DOCIBM.COM"
    DTC = "$SYS$SYSVPWFILES:DTCVAX"
    DTREE = "@UTIL$ROOT:[DECUS.TOOLS]DIRTREE"
    DVIIBM = "@GOO$EXE:DVIIBM.COM"
    EARN = "EDIT/READ/EDT/COMM=GSI$MANAGER:GSIEDT.EDT
           JAN_SYS:DDAGSI5.NETINIT"
    ECL*INE = "@GOO$EXE:ECLINE.COM"
    ED*DT = "@GOO$EXE:EDDT.COM"
    ENC*IPHER = "@UTIL$ROOT:[DECUS.TOOLS]ENCIPHER"
    HLP*SCR = "@GOO$EXE:HLPSCR.COM"
    HO*ST = "@GSI$MANAGER:GSIHOST.COM"
    HOSTI*BM = "@GSI$MANAGER:HOSTIBM.COM"
    LIBLIS = "@GOO$EXE:LIBLIS.COM"
    LO*G = "@GSI$MANAGER:GSILOGOFF.COM"
    LSE*DIT = "LSEDIT"
    MDCL*ANAL = "$GOO$EXE:MDCLANAL"
    MDCLLIST = "$GOO$EXE:MDCLLIST.EXE"
    MLCOUNT = "$GOO$EXE:MLCOUNT.EXE"
    MOV*E = "@GOO$EXE:MOVE.COM"
    MPF*KEY = "$GOO$EXE:MPFKEY.EXE"
```

```
   MTRIM = "$GOO$EXE:MTRIM.EXE"
   NWDCL = "@GOO$EXE:NWDCL.COM"
   PFKEY = "@GOO$EXE:PFKEY.COM"
   PLAY*GAME = "@UTIL$ROOT:[GAMES.COM]PLAYGAME"
   PRIL = "@GOO$EXE:PRIL.COM"
   PRIMER = "HELP @UTILITY PRIMER"
   REM*INDER = "$SYS$SYSVPWFILES:REMINDER"
   RIBM = "@GOO$EXE:RIBM.COM"
   RR*UN = "@GOO$EXE:RRUN.COM"
   SCLU1 = "@GSI$MANAGER:SHOW_CLUSTER
           GSI$MANAGER:SHOW_CL_INI_1.COM"
   SCLU2 = "@GSI$MANAGER:SHOW_CLUSTER
           GSI$MANAGER:SHOW_CL_INI_2.COM"
   SDEVS = "@GSI$MANAGER:SDEVS.COM"
   SIBM = "@GOO$EXE:SIBM.COM"
   SIN*FORM = "$SYS$SYSTEM:WHAT.EXE"
   SMEM = "@GSI$MANAGER:SMEM.COM"
   SSYM*BOL = "@GOO$EXE:CSHOWSYM.COM"
   VPW*MENU = "@SYS$SYSVPWFILES:VPWV4"
   WA*TCH = "$DAR$LIBRARY:MONITOR.EXE"
   WHE*RE = "SHOW LOGICAL SYS$USERNODE"
 Batch symbols
   DBAT*CH = "DELETE SYS$BATCH/ENTRY="
   DFBAT*CH = "DELETE SYS$FAST/ENTRY="
   DIBAT*CH = "DELETE SYS$IBM/ENTRY="
   DPLOT = "DELETE SYS$PLOT/ENTRY="
   DPRI = "DELETE SYS$PRINT/ENTRY="
   DTBAT*CH = "DELETE SYS$TERM/ENTRY="
   FSUB*MIT = "SUB/QUEUE=SYS$FAST"
   ISUB*MIT = "SUB/NOTIF/QUEUE=SYS$IBM"
   SUB*MIT = "SUB/QUEUE=SYS$BATCH"
   TSUB*MIT = "SUB/QUEUE=SYS$TERM"
   SBAT*CH = "SHOW QUEUE/BATCH/ALL"
   SFBAT*CH = "SHOW QUEUE/ALL SYS$FAST"
   SIBAT*CH = "SHOW QUEUE/ALL SYS$IBM"
   SPRI = "SHOW QUEUE/DEV/ALL"
   STBAT*CH = "SHOW QUEUE/ALL SYS$TERM"
 TEX symbols
   DVITYPE = "$TEX$:DVITYPE"
   INITEX = "$TEX$:INITEX.EXE"
   PLTOTF = "$TEX$:PLTOTF"
   POOLTY*PE = "$TEX$:POOLTYPE"
```

```
TANGLE = "$TEX$:TANGLE"
TEX = "$TEX$:TEX.EXE"
TFTOPL = "$TEX$:TFTOPL"
WEAVE = "$TEX$:WEAVE"
```

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

MADDR

MADDR

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

ARGUMENTS

DESCRIPTION

CALLING MADDR

ARGUMENTS

FUNCTION An address list from the file with the logical name INPUT will be formatted and written to the file with the logical name OUTPUT. Each address item must be separated with two leading "??", the line with the ZIP code must be precedet by one "?". If the string 'D R U C K S A C H E' appears in an input line an extra blanck line will be written after that line. The output file can be printed on a DEC laser by commands like PA80F, PB80F, PC80F,... depending on the laser printer to be used. This output can be copied by any copier on copy label sheets, e.g. 'HERMA copy-print DIN A 4'.

The logical names INPUT and OUTPUT must be defined before the call.

EXAMPLE Usage:

```
$ DEFINE INPUT KP1$ROOT:[USER.TEXT]ADDR.DAT
$ DEFINE OUTPUT KP1$ROOT:[USER.TEXT]ADDR.LIS
$ MADDR
$ PC80F OUTPUT
or
$ DEFINE/USER INPUT KP1$ROOT:[USER.TEXT]ADDR.DAT
$ DEFINE/USER OUTPUT KP1$ROOT:[USER.TEXT]ADDR.LIS
$ MADDR
$ PC80F KP1$ROOT:[USER.TEXT]ADDR.LIS
```

Example of input format:

```
??D R U C K S A C H E
Herrn
H. Kiowski
Hochschul-Rechenzentrum der
Universitt Frankfurt
Postfach 11 19 32
?6000 Frankfurt/M 1
??D R U C K S A C H E
Herrn
Prof. Dr. H. Backe
Physikalisches Institut der
Universitt Mainz
Postfach 3980
?6500 Mainz
```

MADJUST

MADJUST input[,output]/ADJUST/TAB

PURPOSE	Adjusts right margins of documentation headers
ARGUMENTS	
input	name of input file (COM,PLI,PPL,FOR,PAS,MOD,MAR,MSG)
output	name of output file (def. = input)
/ADJUST	Adjustment is done outside headers too.
/TAB	Replaces in headers spaces with tabs.

DESCRIPTION

CALLING	MADJUST input[,output]/ADJUST/TAB
ARGUMENTS	
input	name of input file (COM,PLI,PPL,FOR,PAS,MOD,MAR,MSG)
output	name of output file (def. = input)
/ADJUST	Adjustment is done outside headers too. There is, however still a difference between headers and code: Outside it is not required that a comment line is closed. A warning will be given, but no close delimiter is added. Actually the line is not modified. Inside, an open comment line is closed without notice. Tabs are replaced everywhere.
/TAB	Replaces in headers spaces with tabs.

FUNCTION

In sources of FOR,COM,MSG,MAR comment lines will be stripped from trailing spaces and delimiters. PLI,GIP,PAS,MOD comment lines will be adjusted to 72 characters per line. If adjustment is not possible (line too long) a message is printed to terminal. The last character before the end delimiter is used as fill character. Adjustment is performed only in header blocks except /ADJ is specified. Only lines with a comment delimiter in first column are processed. All TABs are replaced by SPACEs in all lines If /TAB is specified however, all SPACEs are converted back into TABs if this reduced the file size. Number of lines and maximal line length are printed at the end.

MANALCH switches

MANALCH

PURPOSE Check GOOSY analysis routines.

ARGUMENTS

input Input from logical name INPUT.

output Output to logical name OUTPUT.

FUNCTION Copies input file to output file. Checks all $ACCUi $SPECi and $CONDi macro calls. Compares spectrum data types and conditions types with the types specified in $LOC macros. When the types are not specified in $LOC macros, they are inserted. Checks, if referenced spectra and conditions are located, and if located spectra and conditions are referenced. Checks if name dimensions in all macro calls are consistent.

NOTE The argument lists of the macros are not checked.

MANL

RUN GOO$EXE:MANL

PURPOSE Attach and execute dynamic lists

ARGUMENTS

Comm. proc. -

DESCRIPTION

CALLING RUN GOO$EXE:MANL

ARGUMENTS

FUNCTION Executes commands to attach and execute dynamic lists

REMARKS -

EXAMPLE R GOO$EXE:MANL

MBASE

MBASE

PURPOSE Mount/dism GOOSY data base.

ARGUMENTS

DESCRIPTION

CALLING MBASE

ARGUMENTS

FUNCTION The program defines MOUNT/DISMOUNT BASE command. If there was any error, the local DCL symbol

GOOSY_STATUS

is set to the last error code. $STATUS is always 1 to prevent DCL to print error messages a second time

REMARKS -

EXAMPLE R GOO$EXE:MBASE

MBINHEX

MBINHEX binfile

PURPOSE	Convert binary files to ASCII HEX format.
ARGUMENTS	
binfile	Binary source file.

Description

FUNCTION	The converted output is written to BINHEX_OUT
REMARKS	Program is called by CBINHEX.COM
File name	MBINHEX.FOR
Dataset	-
Version	1.01
Author	B.Kolb
Last Update	24-OCT-1985

MCALLEX

MCALLEX /PPL/PLI

PURPOSE Extract calling sequence.

ARGUMENTS

/PPL/PLI Module language (PPL is default).

Description

CALLING MCALLEX /PPL /PLI

ARGUMENTS

/PPL/PLI Module language (PPL is default).

FUNCTION Extracts calling sequences of modules.

MCALLRTL

MCALLRTL

PURPOSE Extracts call-statements out of a GOORTL listing

ARGUMENTS

Comm. proc. EXTRCALL

Description

FUNCTION The input is a listing done with LIB /EXTRACT from Library GOORTL, assigned to log. name INPUT. For each DECLARE-statement a CALL statement is extracted and written to module-name.INC

```
STS$VALUE=module-name(
    parameter-1,
    parameter-2,...
    parameter-n);
IF ^ STS$SUCCESS THEN @RET(STS$VALUE);
```

This can be later put to GOOCALL.

Version 1.01

Author G. Schneider

Last Update 18-SEP-1986

MCALLSYS

MCALLSYS

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

ARGUMENTS

Comm. proc. EXTRCALL

Description

FUNCTION The input is a listing done with LIB /EXTRACT from Library PLISTARLET, assigned to log. name INPUT. For each DECLARE-statement a CALL statement is extracted and written to module-name.INC

```
STS$VALUE=module-name(
    parameter-1,
    parameter-2,...
    parameter-n);
IF ^ STS$SUCCESS THEN @RET(STS$VALUE);
```

This can be later put to GOOCALL.

Version 1.01

Author G. Schneider

Last Update 24-JUN-1986

MCALLTREE

MCALLTREE [module-name]

PURPOSE Makes a calling tree out of a cross reference list

ARGUMENTS Builds the calling tree only for this module

Description

FUNCTION The program searchs through the input file, assigned to logical name 'INPUT' for the headline of the cross reference part. All following lines are read and stored internally. A recursive routine then builds the calling tree. If a module name is specified, only for this module the calling tree is built. Output is written to a file assigned to log. name 'OUTPUT'. Modules called in recursive loop, marked with "***", are additional listed at the end.

File name MCALLTREE.PPL

Dataset -

Version 1.0

Author G.Schneider

Last Update 04-MAR-1986

MCMD

R GOO$EXE:MCMD

PURPOSE Test and demonstration program for the command dispatcher. This programm is used to validate command dispatcher functions after changes and is part of the command dispatcher tuturial

ARGUMENTS

Comm. proc. -

DESCRIPTION

CALLING R GOO$EXE:MCMD

ARGUMENTS

FUNCTION The C$CRECM procedure is used to define some simple test commands:

SIMPLE A simple command without arguments.
It may be entered as:

```
SIMPLE
        ! full command verb
SIM
        ! abreviated command verb
SIM ! comment after command
        ! with comment
```

POSITIONAL A command with 3 postional parameters.
It may be entered as:

```
POSTIONAL
        ! all 3 parameters will be
        ! defaulted
POS 1
        ! first parameter specified
        ! others will be defaulted
POS 1 2 XXX
```

```
                                    ! all 3 parameter specified
                POS 1.E3 2.E5 "test string"
                                    ! integers may be specified
                                    ! like floating numbers.
                                    ! strings with delimiters
                                    ! have to be enclosed in
                                    ! quotes.
```

REQUIRED Same as command POSITIONAL, but all 3 parameters are required, prompt strings are defined for the first 2.
It may be entered as:

```
REQUIRED
                    ! all 3 parameters will be
                    ! prompted
REQ 1
                    ! first parameter specified
                    ! others will be prompted
REQ 1 2 XXX
                    ! all 3 parameter specified
REQ FL=2 IN=1 ST=XXX
                    ! all parameters may be
                    ! specified by name.
```

COMPLEX A command with one required parameter, one optional positional parameter and one qualifier.
It may be entered as:

```
COMPLEX
                    ! first parameters will be
                    ! prompted, second defaulted
COM 1/LIS
COM 1 2
```

REMARKS The main procedure is declared with the option

```
RETURNS(BIN FIXED(31))
```

which allows to pass the last status code back to the DCL command dispatcher and allows to check at least the success/nonsuccess status in a calling command procedure.

EXAMPLE -

MCOMHLP

MCOMHLP input output [/HELP]

PURPOSE	Reformats a text module generated by MEXTHEA for command description for MFORMDO.
ARGUMENTS	
input	Text module for command description:
output	temporary output file (+= input for MFORMDO
/HELP	Help formatting reqested
Comm. proc.	CFORMAT.COM CDOCUM.COM

DESCRIPTION

The description of GOOSY commands is in the headers of the action routines. These routines may be in different files. Therefore any command including all sub keys gets its own entry in the text library. Extracting the main keyword followed by *, the whole description is extracted. However, the format must be slightly adjusted for MFORMDO. This is done by this program.

The adjustments are necessary only for HELP text.

MCTRL

STS=MCTRL

PURPOSE	Control inactive users
ARGUMENTS	
I_SECONDS	Cycle time between two subsequent user checks.
Return type	BIN FIXED(31)
Status codes	-
Initialize	-

Description

CALLING	STS = MCTRL

Function

This routine checks all users and sends a message to inactive processes (only interactive processes). The cycle time between two subsequent user checks is fixed to 15 min. If the user remains inactive after four cycles he will be canceled!

The routine needs OPER and WORLD privileges.

MDBCOPY

MDBCOPY

PURPOSE Compress and decompress GOOSY data bases.

ARGUMENTS

Comm. proc. -

DESCRIPTION

CALLING RUN GOO$EXE:MDBCOPY

ARGUMENTS

FUNCTION The program defines several commands to compress and decompress data bases.

REMARKS -

EXAMPLE R GOO$EXE:MDBCOPY

MDBM

MDBM

PURPOSE Activate different Data Base Management Activities.

ARGUMENTS

Comm. proc. -

DESCRIPTION

CALLING RUN GOO$EXE:MDBM

ARGUMENTS

FUNCTION The program defines several Data Base functions as commands. This functions are composed with Data Management procedure calls. To leave this program hit CTRL Z. If there was any error, the local DCL symbol

GOOSY_STATUS

is set to the last error code. $STATUS is always 1 to prevent DCL to print error messages a second time

REMARKS -

EXAMPLE R GOO$EXE:MDBM

MDCLANAL

MDCL*ANAL <command line>

PURPOSE Commands to analyze DCL procedures (Call tree) and generate 'debug' versions.

ARGUMENTS

Command line
DEBUG <file> [OUTPUT=<file>][/INTO]
LIST <file> [OUTPUT=<file>][/FULL]
[/NOCOMM][WIDTH=<columns>]
[/DEPTH=<levels>]

Comm. proc. -

DEBUG

DEBUG <file> [OUTPUT=<file>][/INTO]
[/NOCOMM][WIDTH=<columns>]

Function Reads DCL procedure from specified file. Copies the DCL lines to output file (def.=TTY). Behind each symbol assignment a 'SHOW SYMBOL' statement is added. VERIFY is set ON. Unless /INTO is specified, verification is switched off during execution of other DCL procedures. If /NOCOMM is specified, no comments are written to output file. The begugging procedure leaves VERIFY ON !

Remarks For some restrictions see Description

LIST

LIST <file> [OUTPUT=<file>][/FULL]
[/NOCOMM][WIDTH=<columns>]
[/DEPTH=<levels>]

Function Generates a listing of a DCL procedure. Output to specified file or terminal. All called procedures are included. Unless /FULL is specified, only procedure calls are output. If /NOCOMM is specified, comments are subpressed. Symbol substitution is done, but only global symbols known at command time can be translated. I.e. a procedure call via a symbol which is assigned in the DCL procedure itself cannot be resolved. Calling levels are expanded up to 16. /DEPTH may reduce this number./DEPTH=0 just analyzes the input file, 7 is default.

Remarks For some restrictions see Description

Description

CALLING MDCL*ANAL <command line>

ARGUMENTS

ommand line

DEBUG <file> [OUTPUT=<file>][/INTO]
LIST <file> [OUTPUT=<file>][/FULL]
[/NOCOMM][WIDTH=<columns>]
[/DEPTH=<levels>]

FUNCTION Both commands read an DCL file and output it state- ment by statement after some modifications. Lines without a dollar sign at the beginning are not analyzed except they are continuation lines. Symbol translation of first token in a line or behind the 'THEN' of an IF-THEN statement is performed to find hidden procedure calls.

REMARKS DCL allows the following code:

```
$ THEN = 5
$ IF THEN .eq. 5 THEN CALL_PROC
```

The symbol CALL_PROC will not be found !
Even nicer:

```
$ THEN = "@HELL"
$ IF COMMAND .eqs. THEN THEN THEN
```

The symbol THEN will not be found !

```
$ CALL_PROC = "@PROC"
$ CALL_PROC
```

Symbol known at runtime only ! PROC.COM will not be processed. Recoursive procedures are handled correctly.

OUTPUT Because all input lines of a statement are concatenated they must be reformatted for output. So statements with many continuation lines may look different at output.

Comments at the end of statement lines are output before the statement.

EXAMPLE

MDCL LIST GOO$EXE:GOOLOG /NOCOM /FUL

Full listing of GOOLOG.COM including all called procedures.

MDCL LIST SYS$MANAGER:SYSTARTUP OUT=SYSTARTUP.LIS

Short listing (only procedure calls).

MDCL DEB GOO$EXE:GLPUT.COM OUT=X.COM

X is executable and shows all symbols during execution. Verification is switched off over other procedure calls.

MDCL DEB GOO$EXE:GLPUT.COM OUT=X.COM /INTO

Verification is on through all DCL levels.

MDCLDEB

MDCLD <key list>

PURPOSE Generates debug versions of command procedures.

ARGUMENTS

key list List of keys to select options:

VERIFY	Insert verify. Verify is switched off during any procedure call.
SYMBOL	Any symbol is printed after assignment.

Both are defaulted.

Comm. proc. CDDEBUG

DESCRIPTION

CALLING MDCLD <key list>

ARGUMENTS

key list List of keys to select options:

VERIFY	Insert verify. Verify is switched off during any procedure call.
SYMBOL	Any symbol is printed after assignment.

Both are defaulted.

FUNCTION The input file (a DCL procedure) is read and copied to the output file. If VER is specified, a SET VERIFY is inserted at the beginning. Verification is interrupted during procedure calls (recognized by '$ @'). Single lines can be excluded from verification by a preceeding line '$!@'. If SYMBOL is specified, any line starting with $ <name>

= ... is analyzed and a line '$ SHOW SYMBOL <name>' is inserted behind. Similar, a line like $ ... THEN <name> = ... is analyzed and a line '$ SHOW SYMBOL <name>' is inserted behind. Additional debug statements may be included by
$!DEBUG> <statement>

REMARKS Documentation headers are excluded.

EXAMPLE -

MDCLLIST

MDCLLIST "dsc" p2 p3 .. p8

PURPOSE This procedure provides a mechanism which allows to pass parameters and qualifiers to DCL procedures in the same way as to DCL commands.

ARGUMENTS

dsc Argument descriptor string.

p2..p8 Optional arguments passed tp MDCLLIST

Menu option If a DCL procedure using MDCLLIST is called with an ? as argument a menu is entered.

Description

CALLING MDCLLIST "dsc" p2 p3 .. p8

ARGUMENTS

dsc Argument descriptor string.
This string contains a description of the allowed parameters and qualifiers for the called procedure.
Format:
"pref para_dsc qual_dsc"

pref Name of the calling procedure, will be used as prefix for all symbolnames.

para_dsc Parameter descriptor.
This descriptor consists of a series of parameter definitions separated by one blank.
A parameter definition has the format:
name[=default]
The parameter is treated as a required parameter and will be prompted if the default is a question mark (?).

qual_dsc Qualifier descriptor.
This descriptor consists of a series of qualifier definitions separated by a slash. A qualifier definition has the format:
/name[=[default]]
The name may be specified with an imbedded asterisk (*) to mark the allowed abriviation. The qualifier may have a value if a default, even a blank one, is specified.

p2..p8 Optional arguments passed to MDCLLIST. If at least one is specified, they will define instead of the DCL symbols P1..P8 the arguments and qualifiers. For details on the usage of those arguments refer to the function description.

FUNCTION The following actions are preformed:

1. The command line is parsed in the descriptor part and the optional arguments. If no optional arguments have been specified, the local DCL symbols P1 ... P8 are taken as arguments.
2. The descriptor is parsed.
3. For every parameter a symbol with the name
<pref>_<name>
is defined and set with either
a positional parameter found in the arguments
a prompted value if required (default=?)
or the default string (if ommitted blank)
4. For every qualifier a symbol with the name
<pref>_<name>
is defined. If no default string was specified a simple qualifier is expected and the symbol is set with either
the full qualifier name with a leading slash
or a null string
If a default string, even a null string, was specified a qualifier with value is expected and the symbol is set with either
the qualifier value
or the default string

Menu option If a "?" is entered for a parameter, a menu is called to get the parameters. The prefix is used for help, the parameter and qualifier names as prompt strings. A headline can be specified by local symbol MDCLLIST_help.

REMARKS -

EXAMPLE The MDCLLIST procedure was initially used in the module management procedure paket. You will find a lot of examples in the procedures:

GOO$EXE:GLPUT.COM
GOO$EXE:GLSHOW.COM
ect.

MDISP

MDISP or by display commands

PURPOSE GOOSY display program

ARGUMENTS

DESCRIPTION

CALLING MDISP or by display commands

ARGUMENTS

MDVICNVV

RUN GOO$EXE:MDVICNV

PURPOSE	Convert TeX DVI-file to ASCII Hex-code
ARGUMENTS	
Comm. proc.	GOO$EXE:DVICNV

Description

CALLING	RUN GOO$EXE:MDVICNV
ARGUMENTS	
FUNCTION	-
REMARKS	-
EXAMPLE	-

MEXTHEA

MEXT

PURPOSE Extracts documentation blocks generated by MGENHEAD and generates a text module for MFORMDOC

Comm. proc. CEXTRACT.COM

DESCRIPTION

FUNCTION See also MGENHEAD. Documentation blocks are marked by

xxl+name

xxl-name

where xx are the 'comment on' delimiters depending of the source (e.g. PLI is /*, command procedures is $!). l is the block level. A level of 1 marks a main documentation block must contain the line

Module : <name>

The module name is used as main level keyword for the VAX help library.

Subblocks may be marked within a block by a line

xxn+name

where n may be 2,3. All names must be terminated by '*'. Other input files may be included by a line

xx0+<file name>*

MFIC_CTRL

MFIC_CTRL

PURPOSE Activate different Activities using the command dispatcher.

ARGUMENTS

Comm. proc. -

DESCRIPTION

CALLING RUN GOO$EXE:MFIC_CTRL

ARGUMENTS

FUNCTION The program defines several functions as commands. If there was any error, the local DCL symbol

GOOSY_STATUS

is set to the last error code. $STATUS is always 1 to prevent DCL to print error messages a second time

REMARKS -

EXAMPLE R GOO$EXE:MFIC_CTRL

MFORMDO

MFORM list

PURPOSE Generates SCRIPT and RUNOFF files from textmodules created by MEXTHEA for help and documentation

ARGUMENTS

list List of format key words:
HLP output RUNOFF syntax for help
DOC output RUNOFF syntax for print
SCR output SCRIPT syntax
TEX output LaTeX syntax
TOC generate table of content
BRIEF output main part only
HL1 Use script level HL1
HL2 Use script level HL2
LEV2 2+blocks generate H2 or H3 instead of higthlightenig.
/SELECT=(block list)
/EXCLUDE=(block list)

Comm. proc. CFORMAT.COM
CDOCUM.COM
CDOCALL.COM

DESCRIPTION

A text module as generated by MEXTHEAD is converted to SCRIPT,LaTeX or RUNOFF code. The output may be suited for either adding to help libraries or adding to documentation

MGENCIM

MGENCIM

PURPOSE Generate card image file to send to IBM

ARGUMENTS -

Comm. proc. PRIL

DESCRIPTION

CALLING MGENCIM

ARGUMENTS -

FUNCTION Generates a card image file from any ASCII file. First character is either SPACE, '+' or '1'. SPACE means NEW LINE '+' means continuation line '1'.. means information Second character is either '1', SPACE, '0' or '+' (see ANSI printer control characters)

REMARKS -

EXAMPLE -

MGENHEA

MGEN language type mode

PURPOSE Generates interactively documentation headers for programs, procedures and command procedures.

ARGUMENTS

language PLI,VAX,PDP,IBM,FOR,PAS,MOD,DCL,TSO (def=PLI) GIPSY

type PROC,MAIN,MACRO (def=PROC)

mode BRIEF,FULL (def=FULL)

Comm. proc. CEDITDOC.COM

DESCRIPTION

CALLING MGEN language type mode

ARGUMENTS

language PLI,VAX,PDP,IBM,FOR,PAS,MOD,DCL,TSO (def=PLI)
Selects language. VAX,PDP and IBM generate assembler headers and imply type=PROC. DCL and TSO generate command list headers and ignore type.

type PROC,MAIN (def=PROC)
different types of documentation headers

mode BRIEF,FULL
for a brief or verbose version of header

REMARKS Depending on the specified key words different dialogues are choosen. The documentation is splitted in several blocks. Main blocks are:

1+ GIPSY Proced.
1+ GIPSY Macro
1+ PLI Procedure

1+ PLI Main
1+ PLI Macro
1+ FOR Subroutine
1+ FOR Main
1+ PAS Routine
1+ PAS Main
1+ VAX Module
1+ PDP Module
1+ IBM Module
1+ DCL Command
1+ TSO Command

Sub blocks are:

2+ DESCRIPTION
2+ COMMAND
2+ IMPLEMENTATION
2+ UPDATES
2+ INTERNALS
(others may be added following the syntax)

The names of these blocks are used as second level key words for the help facility. In the printed documentation they are used as head lines.

You may add your own documentation blocks on 2+ or 3+ level. Similar, you may add your own unleveled key words following the syntax as generated by MGENHEA (+ <keyword> : text).

MEXTHEA extracts the program headers and writes textfiles for libraries. MFORMDO generates help and documentation files for SCRIPT and RUNOFF.

NOTE:

The precomposer must run first, because the composer also extracts %INCLUDE statements from the source code. A full description of the documentation system may be obtained by HELP @UTILITY DOCUMENT

MGNS_ESONE

RUN GOO$EXE:MGNS_ESONE

PURPOSE	Network object to perform remote CAMAC accesses.
ARGUMENTS	-
Comm. proc.	GOO$EXE:GESONE.COM Note, that a logical name GESONE which translates to the above file name must be defined.

Description

CALLING	RUN GOO$EXE:MGNS_ESONE
ARGUMENTS	
FUNCTION	This is a GOONET server process which implements remote CAMAC accesses for the GOOSY CAMAC interface. This image should be executed as a VMS NETWORK process. It should be activated with the command procedure GOO$EXE:GESONE. The main procedure declares itself a GOOnet (and DECnet) object with the name GESONE, calls I$ICESO to declare the GNA component GN_CP_ESONE and wait for incomming connections.
REMARKS	-
EXAMPLE	-

MGOOWAIT

MGOOWAIT process

PURPOSE Wait for analysis completion

ARGUMENTS

process Name of analysis process e.g.:
GN_TEST_TEST_$ANL

Description

FUNCTION The analysis program governs two locks:
node_process_STOP and
node_process_RUN
MGOOWAIT waits for the RUN lock end exits.

REMARKS The node prefix is translated from GOO$NODE which is defined in GOOLIBDEF as the short node name.

Directory GOO$UTIL

Version 1.01

Author H.G.Essel

Last Update 24-OCT-1986

MGTOOL

GTOOL [LIBRARY—DOCUMENT—MESSAGE]

PURPOSE	Calls GOOSY tools like code management or documentation tools by menu.
ARGUMENTS	
LIBRARY	Calls code management procedures (GL...)
CDOCUMENT	Calls documentation procedures
MESSAGE	Calls message generation procedure
Comm. proc.	-

Description

CALLING	GTOOL [LIBRARY—DOCUMENT—MESSAGE]
ARGUMENTS	
LIBRARY	Calls code management procedures (GL...)
CDOCUMENT	Calls documentation procedures
MESSAGE	Calls message generation procedure
FUNCTION	All functions are selected by menu.
REMARKS	-
EXAMPLE	-

MGUIDE_DISP

RUN MGUIDE_DISP

PURPOSE Called in guide.com to display one menu

ARGUMENTS The programm mguide_disp need some global symbols. guide_source: contains the rootdirectory. guide_first: contains the first string of every

menu or commandprocedure who is called.
guide_l1 : contains the name of the main menu guide_1 : contains the name of the main programm guide_num : contains the number who is at the end

of guide_ and guide_l.
example: guide_1 –> guide_num = 1

Return type BIN FIXED(31)

Status codes -

Initialize -

Include name GOOINC(mguide_disp)

Description

CALLING RUN mGUIDE_DISP

ARGUMENTS The programm mguide_disp need some global symbols.
GUIDE_1 == "GU_GOOSY.TXT"
GUIDE_L1 == " GOOSY top menu "
GUIDE_FIRST == "GU_GOOSY"
GUIDE_NUM == "1"
GUIDE_SOURCE== "EE$ROOT:[ROGER.GUIDE]"
guide_source: contains the rootdirectory. guide_first: contains the first string of every

menu or commandprocedure who is called.

guide_l1 : contains the name of the main menu guide_1 : contains the name of the main programm guide_num : contains the number who is at the end
of guide_ and guide_l.
example: guide_1 –> guide_num = 1

FUNCTION -

REMARKS -

EXAMPLE -

MHEXBIN

MHEXBIN binfile

PURPOSE	Convert ASCII HEX format file created by BINHEX to binary files.
ARGUMENTS	
binfile	binary output file.

Description

FUNCTION	The input is read from HEXBIN_IN
REMARKS	Program is called by CHEXBIN.COM
MODULES	MHEXBIN_REM_LBL MHEXBIN_LENGTH
File name	MHEXBIN.FOR
Dataset	-
Version	1.01
Author	B.Kolb
Last Update	24-OCT-1985

MHLPSCR

from DCL Procedure HLPSCR

PURPOSE Generates a file for SCRIPT output from a Help file

ARGUMENTS

Description

FUNCTION Reads input from a HELP file, assigned to logical name "INPUT". A SCRIPT header is placed at the beginning of the text. A title is taken from a logical name trans- lation of "OUTPUT". HELP level numbers are changed into SCRIPT commands. Escape sequenzes are changed as follows :

<esc>#3, <esc>#4 -> text
<esc>[Ps;...;Psm
Ps=1,4,5,7 ->
<esc>[0m or <esc>[m ->
All others
<esc><RI>..<RI> -> $.....
Output is written to a file assigned to logical name "OUTPUT".

File name MHLPSCR.PPL

Dataset -

Version 1.00

Author G.Schneider

Last Update 29-OCT-1985

MHLPTEX

from DCL Procedure HLPTEX

PURPOSE Generates a file for TEX output from a Help file

ARGUMENTS

Description

FUNCTION Reads input from a HELP file, assigned to logical name "INPUT". A TEX header is placed at the beginning of the text. HELP level numbers are changed into TEX commands.... Escape sequenzes are changed as follows :

<esc>#3, <esc>#4 -> boldface
<esc>[Ps;...;Psm
Ps=1,4,5,7 -> boldface
<esc>[0m or <esc>[m -> end boldface
All others
<esc><RI>..<RI> -> $.....
Output is written to a file assigned to logical name "OUTPUT".

File name MHLPTEX.PPL

Dataset -

Version 1.00

Author G.Schneider

Last Update 29-OCT-1985

MINSNUM

MINS

PURPOSE Inserts line numbers to PLI source files

Comm. proc. CINSNUM.COM

DESCRIPTION

CALLING MINS

REMARKS -

MJCLTRIM

MJCLTRIM file trimmedfile

PURPOSE trim JCL output files from the IBM

ARGUMENTS

Description

FUNCTION JCL output files from the IBM will be processed. Trailing blanks will be kicked away. Further (nyi) the last page of the JCL-output will additional be set at the front of the trimmed file.

Version 1.01

Author K.Winkelmann

Last Update 14-MAY-1987

MLCOUNT

MLCOUNT input /LIST /NOHEAD
/PPL/PLI/FOR/MAR/COM

PURPOSE Looks for longest record and assignes value to DCL symbols MLCOUNT_MAX_LENGTH and MLCOUNT_LINES.

ARGUMENTS

input Source file.

/LIST Lists lines longer than 80 columns.

/NOHEAD Ignore lines between HEAD documentation blocks

DESCRIPTION

CALLING MLCOUNT input /LIST

ARGUMENTS

input Source file.

/LIST Lists lines longer than 80 columns.

FUNCTION Number of lines and maximal line length are printed at the end.

REMARKS

MLIBWILD

MLIBWILD <module spec>[/FULL]

PURPOSE	Reads input as generated by LIB/LIST and outputs list of matching module names.
ARGUMENTS	
module spec.	Module specification. One asterisk is permitted.
/FULL	Outputs the whole input line. If omitted only the module names are listed.

Description

FUNCTION	Looks through a temporary file ,created by the librarian, which is assigned to logical name "INPUT", for matching names, writing them to a file, assigned to logical name "Output".
File name	MLIBWILD.PPL
Dataset	-
Version	1.02
Author	G.Schneider
Last Update	26-JUL-1985 6-AUG-1985 match-check is done in the separate routine u$match. (G.Schneider) 18-AUG-1985 switch "/FULL" added. (G.Schneider)

MLOCKS

MLOCKS

PURPOSE Show VMS locks.

ARGUMENTS

Comm. proc. -

DESCRIPTION

CALLING RUN GOO$EXE:MLOCKS

ARGUMENTS

FUNCTION The program defines several Data Base functions as commands. This functions are composed with Data Management procedure calls. To leave this program hit CTRL Z.

REMARKS -

EXAMPLE R GOO$EXE:MLOCKS

MMESDEF

MMESDEF <facility>

PURPOSE Generate PL1 program to generate file for messages linked.

ARGUMENTS

Facility Name of facility to be selected.

Comm. proc. GOO$EXE:MESDEF

DESCRIPTION

CALLING MMESGEN <facility>

ARGUMENTS

Facility The name of the facility. It is used to generate the name of a PL1 file <fac.>_temp.PLI.

FUNCTION The object module GOO$UTIL:MMESDEF.OBJ must be linked with the message file(s) of the desired facility(ies). The generated PL1 file must be compiled and linked and executed to generate the definition file X<facil>. All these actions are done in command procedure MESDEF.

REMARKS Link with the /NOUSER option

EXAMPLE MMESDEF GOOVME

MMESLIST

MMES <facility>

PURPOSE List all messages linked.

ARGUMENTS

Facility Name of facility to be selected.

Comm. proc. GOO$EXE:CMESSAGE

DESCRIPTION

CALLING MMES <facility>

ARGUMENTS

Facility The name of the facility. It is used to generate the name of a textfile X<fac.>.TXT. If this parameter is omitted or is *, then all facilities are searched and output. The filename is then XGOOMSG.TXT and XGOOMSG.HLP.

FUNCTION The object module GOO$UTIL:MMESLIST.OBJ must be linked with the message file(s) of the desired facility(ies). The textfile may be processed as standard. It should be sorted by MTXTSORT and formatted by GLFORMAT .../LEV2

REMARKS Link with the /NOUSER option

EXAMPLE MMES XU

MMODMAP

MMODMAP switches

PURPOSE Formats output from "SEARCH module.lis ENTRY"

ARGUMENTS

DESCRIPTION

CALLING MMODMAP

ARGUMENTS

FUNCTION Called in DCL-procedure CMODMAP which compiles a PPL module with SHOW=MAP, calls SEARCH to get all lines with "ENTRY". MMODMAP selects lines behind "External Entry Points and Variables " Then it looks for "ENTRY" in column of "Attributes" and outputs line, if found. A list of referenced modules is output. File is read from INPUT and written to OUTPUT.

MOPER

MOPER

PURPOSE Execute privileged operator commands.

Description

Command

Command keys REPLY TAPEREQUEST

REPLY TAPEREQUEST number

PURPOSE Answer a tape mount request.

PARAMETERS

number number of request. This number is displayed on the operators console.

EXAMPLE MOPER REPL TAPE 2287

MPFKEY

MPFKEY or <PF2>

PURPOSE Display of DCL auxiliary keypad definition

Description

CALLING MPFKEY or <PF2>

FUNCTION Pressing the PF2 key on the auxiliary keypad this procedure is activated and displays the keypad set-up defined by the command procedure PFKEY.

REMARKS -

EXAMPLE -

MPLOTMET

GOOEXE:MPLOTMET metafile,type,command,
queue,copies,font

PURPOSE	Send a metafile to a plotter.
ARGUMENTS	
metafile	Metafile name.
type	Plotter format
command	Optional print command (DCL)
queue	Optional queue name for plotter (if no command).
copies	Number of copies which should produced.
font	Font to be used to modify text bundle table
Comm. proc.	PLOTMET

Description

CALLING	GOOEXE:MPLOTMET metafile,queue,type,copies,font
ARGUMENTS	
metafile	Name of the metafile which should be plotted on the specified queue or physical device.
command	DCL print command. When specifeid, the queue name is ignored.
type	Device type of the specified plotter. The following types are supported by GOOSY: 1.) LN03 Laser printer 2.) HP7550A3,HP7550A4 pen plotter 3.) POST postscript 4.) SIXEL sixel

queue	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. Ignored when command given.
copies	Number of copies which should be printed. If no Printer queue is specified "copies" is ignored.
font	Font to be used to modify the default text bundle table. This argument could be used to produce pictures with nicer lettering.
FUNCTION	This routine interpretes the device independent metafile and creates a device specific plotfile, which is send to the specified plotter, or printed by specified command.
REMARKS	-
EXAMPLE	-

MPOSTRIBM

RUN GOO$EXE:MPOSTRIBM file columns

PURPOSE removes leading numbers, performs necessary character conversions and chops into several files if PDS format

DESCRIPTION

FUNCTION reads record after record look for +PAT to recognize members convert CTRL Z into — translate @ into $ in membername if 'columns' is not equal 0 then the specified numbers of digits are deleted in the beginning of record.

MPRECOM

MPREC file TAGS(tag list) OUT(out file)

PURPOSE Precomposer for PLI programs. Extracts marked lines from master source and outputs PLI source

ARGUMENTS

file file name (no default extension !)

taglist tag words of lines to be selected

out file output file (for PLI text)

Comm. proc. COMPILE

DESCRIPTION

ARGUMENTS

file file name (NO default extension !)

taglist tag words of lines to be selected
There are some pseudo tags which may be specified in the list to select precomposer actions. These tagwords begin with a dollar sign ($):

$NUM numbered source (IBM def.)
$NON no numbered source (VAX def.)
$COM output comments (def.)
$NOC skip comments (not implemented)
$DEB output to terminal

Some tags should be used by convention:

I IBM code
V VAX code
D debug code
HEAD documentation headers

If no taglist is specified at the VAX, V is
default. If an empty taglist () is specified, no tag default is used, but the Macro expansion is still made for VAX.
At the IBM, the tag default I is inserted in the
clist. To specify "no tags", one has to specify a dummy tag.

out file output file (for PLI text)

TAGS The input file is scanned for tagwords. These tagwords are specified by:

```
%%tag:<PLI line>
```

Lines with no tagword or a matching tag word
are written to OUTPUT. Tag blocks may be specified by

```
%%+tag:<PLI line>
   ...
%%-tag:<PLI line>
```

All lines between these expressions are taged.
Tag blocks may overlap.

MACROS Several intrinsic macros are expanded for IBM-VAX compatibility. After tag checking the following expressions are translated (A * means that these expressions are translated without tags I or V specified):

```
  @DSCR(...)
  @CHAR(...)
  @SIZE(...)
  @STORAGE(...)
  @REPEAT(c,f)
* @CALL
  @DCL_MSG(x)
  @INCLUDE x(y)
  @LSTATIC (x)
  @LEXT (x)
  @LFILE (x)
  @LENTRY (x)
  @LFENTRY (x)
  @ON_ANY_W(entry)
  @LOCAL_ERROR
  @RET(mes)
* @RET_SET_MSG(mes,arg1,arg2,arg3)
* @RET_ADD_MSG(mes,arg1,arg2,arg3)
* @ADD_MSG(mes,arg1,arg2,arg3)
* @ENTRY
* @PUT_CLR_MSG(b_outp)
```

* @DMP_CLR_MSG
* @TRACE_MSG(mes,arg1,arg2...)

ARGUMENTS The arguments for the error macros arg1...arg3 may be of any type on the vax. For compatibility with IBM , however, non character arguments have to pass via the inline macro @CHAR(arg). At the IBM the argument will then be converted to a string.

Another way (valid for VAX and IBM) is the usage of the conversion routines U$CHARx(arg) where x stands for L, I, H, R, D, B.

EXAMPLES

```
@RET_ADD_MSG(XUTIL_E,@CHAR(3.14),'ABC',CV_ARG)
@RET_SET_MSG(XUTIL_E,U$CHARR(3.14),U$CHARR(R_ARG))
@RET_SET_MSG(XUTIL_E,U$CHARL(L_I,'X'),@CHAR(L_X))
MPRE ABC.PPL TAGS(V) OUT(ABC.PLI)
compiles ABC.PPL to ABC.PLI with tagword V
MPRE ABC.PPL OUT(ABC.PLI)
compiles ABC.PPL to ABC.PLI with tagword V
MPRE ABC.PPL
compiles ABC.PPL to PLI.TMP with tagword V
```

MPREMES switches

MPREMES

PURPOSE Concatenate continuation lines

ARGUMENTS

DESCRIPTION

CALLING MPREMES

ARGUMENTS

FUNCTION Continuation lines are marked by an hyphen at the end of previous line.

MREMNUM

MREM trunc,check

PURPOSE Removes line numbers from PLI source files (<trunc> char.) (<check> char. are checked to be digits)

Comm. proc. CREMNUM.COM

DESCRIPTION

CALLING MREM trunc,check

REMARKS -

EXAMPLE

```
ASS/USER A.PPL INPUT
  ASS/USER B.PPL OUTPUT
  MREM 9,8
  check first eight characters of first non zero line
to be digits. Truncate first 9 characters of any input line longer than 8.
```

MREPLACE

MREPLACE input[,output]/PRINT/FORMAT/UPPER

PURPOSE	Replaces old string by new string. Adjusts Documentation headers by option.
ARGUMENTS	
input	Source file.
output	Output file (default=input)
oldstring	Must be assigned to DCL symbol CREPLACE_OLD
newstring	Must be assigned to DCL symbol CREPLACE_NEW
/PRINT	Print source and output lines if modified
/FORMAT	Try to format new line in documentation blocks
/UPPER	Input and old string are uppercased for comparison.

DESCRIPTION

CALLING	MREPLACE input[,output]/PRINT/FORMAT/UPPER
ARGUMENTS	
input	Source file.
output	Output file (default=input)
oldstring	Must be assigned to DCL symbol CREPLACE_OLD
newstring	Must be assigned to DCL symbol CREPLACE_NEW
/PRINT	Print source and output lines if modified
/FORMAT	Try to format new line in documentation blocks
/UPPER	Input and old string are uppercased for comparison.

FUNCTION The string assigned to CREPLACE_OLD is searched in each input line. If found it is replaced by the string assigned to CREPLACE_NEW. Replacement is done for each occurrence of the old string. Replacement is done inside strings and comments. If /UPPER is specified, input and old string are uppercased for comparison. The output, however, is not uppercased. The /PRINT option outputs modified lines to SYS$OUTPUT. The /FORMAT option tries to adjust documentation headers. Tabs are replaced by spaces outside strings marked by '. Number of lines and maximal line length are printed at the end.

REMARKS MREPLACE is called by CREPLACE

MSECTION

MSECTION basename [/version]

PURPOSE	Check if a data base is mounted.
ARGUMENTS	
basename	Name of database
/version	Optional version number. If specified but different than VERSION found in directory $CONTROL an error is returned.

Description

FUNCTION	Tries to attach specified data base. If it fails, returns error to DCL.
Version	1.01
Author	H.G.Essel
Last Update	29-AUG-1986

MSHOSYM

from DCL Procedure CSHOWSYM.COM

PURPOSE Display the current value of a global symbol (wildcard)

ARGUMENTS

Description

FUNCTION Looks through a temp. file which is created by the DCL Procedure CSHOWSYM.COM for matching symbols

File name MSHOSYM.PPL

Dataset -

Version 1.01

Author T.Kroll

Last Update 22-FEB-1985 27-AUG-1985 Matching check now in U$MATCH.(G.Schn.)

MSHOW

GOOEXE:MSHOW <command>

PURPOSE	Show commands for use in spawned processes
ARGUMENTS	
command	SHOW command to be excuted. No prompt!
FUNCTION	The program defines several SHOW commands.
REMARKS	-
EXAMPLE	MSHOW SHOW COND *

MSTATUS

MSTATUS SHOW GOOSY STATUS proc1 [proc2 [proc3]]
GSTATUS proc1 [proc2 [proc3]]

PURPOSE Activate GOOSY status program. Equivalent to SHOW GOOSY STATUS command.

ARGUMENTS

proc1 Name of process to be monitored

Comm. proc. -

DESCRIPTION

CALLING MSTATUS SHOW GOOSY STATUS proc1 [proc2 [proc3]]GSTATUS proc1 [proc2 [proc3]]

ARGUMENTS

proc1 Name of process to be monitored

FUNCTION -

REMARKS -

EXAMPLE GSTATUS GN_HGE___$TMR

MSYMHLP

MSYM

PURPOSE Reads a command procedure and generates help file.

Comm. proc. CSYMHLP

DESCRIPTION

CALLING MSYM

REMARKS The command procedure may contain help level key words marked by $!n key. n may be +,-,1,2,3. A level - switches extraction off, level + on.

The purpose is to get help entries for symbol definitions.

MTMR

RUN GOO$EXE:MTMR

PURPOSE DCL process with a subset of the transport manager functions.

ARGUMENTS -

Comm. proc. -

Description

CALLING RUN GOO$EXE:MTMR

ARGUMENTS

FUNCTION This is a DCL program with a subset of the GOOSY transport manager functions.It declares the command sets:

I$CAMCM for general CAMAC handling
I$CUTCM for CAMAC utilities
I$TMRCM for CAMAC management commands

This process should be used to initialize and load the CAMAC environment in STARTUP and LOGIN procedures.

REMARKS -

EXAMPLE -

MTRIM

MTRIM switches

PURPOSE Remove trailing spaces from source INPUT and writes result to destination OUTPUT

ARGUMENTS

switches /TRAILING : remove trailing spaces
/LEADING : remove leading spaces

DESCRIPTION

CALLING MTRIM

ARGUMENTS

switches /TRAILING : remove trailing spaces
/LEADING : remove leading spaces

FUNCTION Trailing and/or leading spaces are removed from input source. The logical names INPUT and OUTPUT must be defined before the call.

MTXTSORT

MTXTS*ORT input file,[output file]

PURPOSE	Sorts 2+ blocks in text file alphabetically.
ARGUMENTS	
input file	name of input file
output file	name of output file (Def.=input)

Description

FUNCTION

Copies lines from input to output until a 2+block is found. Then 2+ block names and lines are stored in memory, sorted and written to output file.

Input must has the form:

```
2+blockname
...
2+blockname
...
1-
2+blockname
...
1-
...
```

File name	MTXTSORT.PPL
Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	5-SEP-1985

MUAMODI

MUA <action string> <match string>

PURPOSE Runs AUTHORIZE and modifies all user accounts (Called in CUAMODI.COM)

ARGUMENTS

Action This string is appended to a MODIFY <username>.

Match If specified, only matching accounts are modified.

DESCRIPTION

FUNCTION The program reads the output as generated by AUTHORIZE (/LIST/BRIEF). It generates a command procedure to modify all user accounts contained in the list. The optional match string may be used to select certain groups of users. The match is checked for all parameters as listed in the /LIST/BRIEF command:

- Owner
- Username
- UIC
- Privileges
- Default disk
- Default directory

Input is read from SYS$INPUT
Output is written to MODAUTHOR.TMP

EXAMPLE

```
$ RU AUTHORIZE
  LIST/BRIEF
  ^Z
$ ASS/USER SYSUAF.LIS INPUT
$ MUA /PASSWORD=GOSSIP [200,
$ @MODAUTHOR.TMP
$ DEL MODAUTHOR.TMP;*
```

This example sets all passwords of group 200

File name MUAMODI.PPL

Dataset	-
Version	1.01
Author	H.G.Essel
Last Update	17-FEB-1984

MUDIRO

GOOEXE:MUDIRO.EXE

PURPOSE sorts a given directory list

ARGUMENTS

Description

FUNCTION It fetches the parameters from the COM commandlist and orders the records on the output file from the directory command according to the specified parameters (time or size)

File name MUDIRO

Dataset -

Version 1.01

Author K.Winkelmann

Last Update 11-FEB-1985 18-FEB-1985 BROUGHT ON VMS 4.0 03-jul-1985 errors removed 30-oct-1985 bugs removed /KW

MUTIL

MUTIL

PURPOSE Activate different Activities using the command dispatcher.

ARGUMENTS

Comm. proc. -

DESCRIPTION

CALLING RUN GOO$EXE:MUTIL

ARGUMENTS

FUNCTION The program defines several functions as commands. If there was any error, the local DCL symbol

GOOSY_STATUS

is set to the last error code. $STATUS is always 1 to prevent DCL to print error messages a second time

REMARKS -

EXAMPLE R GOO$EXE:MUTIL

MVMECMD

MVMECMD

PURPOSE VME command executor.

Description

CALLING MVMECMD

FUNCTION This program allows to execute commands in the VME system. It cannot be used to get data. The INIT ACQ command for VME is executed during startup. The ethernet must be already setup by DCL command ETHDEF.

MVOICEX

MVOICE

PURPOSE Extracts documentation blocks generated by MGENHEAD and generates a text module for U$TALK.

Comm. proc. CEXTRACT.COM

DESCRIPTION

FUNCTION See also MGENHEAD. Documentation blocks are marked by

xxl+name

xxl-name

where xx are the 'comment on' delimiters depending of the source (e.g. PLI is /*, command procedures is $!). l is the block level. A level of 1 marks a main documentation block must contain the line

Module : <name>

The module name is used as main level keyword for the VAX help library.

Subblocks may be marked within a block by a line

xxn+name

where n may be 2,3. All names must be terminated by '*'. Other input files may be included by a line

xx0+<file name>*

MWV

MWV

PURPOSE inquires a diary, 'Wiedervorlage'

ARGUMENTS

Description

FUNCTION from the main directory a file WV.TXT is read, in which the user has previously edited his appointments. Starting in column 1 a date is given, if that date matches the current date, the text beginning at column 8 will be written on the terminal. Three types of dates are possible: :OL. :LI.a certain date, as yymmdd , the text will be printed on all following days. :LI.a certain day of the week, a number between 1 and 7 (Sunday=1, ..., Saturday = 7). The text will be printed on that day only. :LI.a certain day of the month, e.g. 13. , a number followed by a dot. The text will be printed on the specified date and on the three following days. :EOL. An example is given in [WINKELMAN]WVEXAMPLE.TXT . The file containing the apointments can be interpreted also on the IBM (command WV (like 'Wiedervorlage')in TSO, dataset name has to be WV.TEXT) .

File name GOO$UTIL:MWV.PPL

Dataset -

Version 1.01

Author K.Winkelmann and H.J.Lustig

Last Update 3-JUL-1985

Contents