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

GSI Online Offline System

GOOSY Data Acquisition and Analysis Commands and Macros

H.G.Essel, M.Richter

June, 28 1988

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

Chapter 1

GOOSY Commands

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 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 -

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.

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 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	-

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

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

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 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 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 -

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 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 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 -

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 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

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 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

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.

Chapter 2

DCL Commands

ALIAS

ALIAS command arguments

PURPOSE Handle GOOSY alias command names

ARGUMENTS

command subcommand key:
CREATE
DELETE
SHOW
A quotation mark enters menu.

arguments argument list depending on command.

Description

FUNCTION Alias names can be defined on two levels:
Environment and Global.
Alias names defined for a certain environment
are activated together with the CRENV command and deactivated with the DLENV command. Global alias names are active anytime. An alias is searched first in the environment table and then in the global table. Alias names are deleted during logout.

Version 1.01

Author H.G.Essel

Last Update 14-APR-1987

CREATE

CALLING ALIAS CREATE name string environment /GLOBAL
CRALIAS name string environment /GLOBAL

name Name of the alias. May contain alphanumeric characters including _ $.
A quotation mark enters menu.

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

DELETE

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

SHOW

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

ATENVIR

ATENV*IR environment

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

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

CLINK

CL file/switches /switches

PURPOSE Link programs. Defaults are updated.

ARGUMENTS

file Name of object file (def.=last)

switches Linker switches. Behind SPACE are not updated

DESCRIPTION

FUNCTION Two problems are solved:
1. To keep last input as default
2. To inculde standard options.

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

REMARKS -

File name CLINK.COM

Dataset -

Version 1.01

Author H.Essel

Last Update 20-JAN-1984

COMMENT

COMMENT topic command arguments

PURPOSE Write and read comments for a topic.

ARGUMENTS

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

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

arguments argument list depending on command.

Description

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

Version 1.01

Author H.G.Essel

Last Update 10-MAR-1991

CREATE

CALLING COMMENT topic CREATE

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

READ

CALLING COMMENT topic [READ]keylist begin-date end-date

topic Name of the COMMENT topic. A logical name must be defined for the topic file.

READ If no argument follows, this keyword may be omitted.

keylist List of keys. Only entries marked with specified keys are output. * means all keys and defaults begin-date to 1-MAR-1991. Not specified defaults begin-date to YESTERDAY.

date Date from/to wich file entries are to be displayed. Defaults depend on keylist. Begin-date default is YESTERDAY when keylist is empty, and 1-MAR-1991 when keylist is not empty (but may be *). End-date defaults to TOMORROW, if not specified.

WRITE-EDIT

CALLING COMMENT topic WRITE-or-EDIT keylist

topic Name of the COMMENT topic. A logical name must be defined for the topic file.

WRITE Lines are prompted and are inserted in the file.

EDIT The headline is written to a temporary file and the editor in invoked.

keylist List of keys. Keys mey be specified as filter when reading entries.

KEYS

CALLING COMMENT topic KEYS

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

Format

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

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

Example

Creating a new topic file for topic MYCOM:

```
$ DEFINE CMT$MYCOM SYS$LOGIN:CMT_MYCOM.TXT
$ COMM MYCOM CREATE
> ...
$ ...
```

Insert entry:

```
$ COMM MYCOM W
> ...
$ ...
```

Insert entry with key (use editor):

```
$ COMM MYCOM EDIT key1,key2
> ...
$ ...
```

Read entries with key (no date limit):

```
$ COMM MYCOM READ key1
$ ...
```

Read all entries:

```
$ COMM MYCOM READ *
$ ...
```

Read entries with key since yesterday:

```
$ COMM MYCOM READ key1 yesterday
$ ...
```

Read all entries since yesterday:

```
$ COMM MYCOM READ
$ ...
```

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

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.
```

CRENVIR

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

PURPOSE Creates a GOOSY environment and optional some GOOSY standard components

ARGUMENTS

environment Name of the environment (max. 4 char)

program Optional name of private analysis program. If not specified, MGOOANL is assumed. This private program is started by /$ANL or by /ONLINE or /OFFLINE, if no /DEF is specified.

component Optional component name for analysis program. Default is $ANL.

/ONLINE Creates TMR, DSP, DBM and analysis program specified by program (default=MGOOANL, if /DEF is not specified). If /DEF is specified, a GOOSY standard analysis program is started. If program is specified, this is used regardless of /DEFAULT.

/OFFLINE Creates DSP, DBM and analysis program specified by program (default=MGOOANL, if /DEF is not specified). If /DEF is specified, a GOOSY standard analysis program is started. If program is specified, this is used regardless of /DEFAULT.

/DEFAULT Creates default analysis. Otherwise use specified program, which is the name of your private analysis program (default=MGOOANL). If a program name has been specified, this switch is ignored.

/$TMR Create Transport Manager $TMR

/$ANL Create Analysis program $ANL

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

Description

FUNCTION

A GOOSY environment is created only if it does not already exist. The components specified by qualifiers are created. You may use this command to create components in an existing environment. Specifiy the current name or * for the environment parameter in this case.

A user specific analysis program linked by command LANL is started from current directory by /$ANL.

GOOSY provides a standard analysis program. This analysis is started by /J11 or by /DEFAULT.

By default the analysis programs are started with priority 3 unless /PRIO=p is specified.

NOTE

An environment is deleted by command DLENV.

Version 1.01

Author H.G.Essel

Created 14-JAN-1987

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

Examples

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

CTRL_T

CTRL_T [process][output]

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

Description

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

DEBUG_WINDOW

DEBUG_WINDOW window
DEBWIN window

PURPOSE Setup windows and inits for debug.

ARGUMENTS

window Desired debug window style:

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

Description

FUNCTION -

REMARKS -

EXAMPLE

```
$ DEBWIN MOTIF
```

DLENVIR

DLENV*IR

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

DTENVIR

DTENV*IR

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

ETHDEF

ETHDEF destination ethernet protocol interface

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

Description

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

GOOCONTROL

GOOC*ONTROL [CREATE]or [DISMOUNT]

PURPOSE Defines logical name GOO$CONTROL for control data base. The data base is created and mounted optionally.

ARGUMENTS

CREATE A new data base is created. The name is composed of GOOCTRL_group_node.SEC

DISMOUNT The data base is dismounted. This should be done during LOGOUT.

Description

CALLING GOOC*ONTROL [CREATE]

ARGUMENTS

CREATE A new data base is created

FUNCTION The programs MGOOTMR and MGOOANL keep their control information in a database GOO$CONTROL, if it exists. The command SHOW GOOSY STATUS may be used to display this information on screen. The command GOOCONTROL creates such a data base and defines the logical name GOO$CONTROL. The "physical" data base name is GOOCTRL_<group>_<node>, where node is the one letter node name, group the hex group number. The command should be included in the LOGIN procedure.

REMARKS On each node there must be a control data base. If the section file does not exist, it is created. If the data base is not mounted, this is done.

EXAMPLE GOOCON CRE
creates a data base GOOCTRL_131_E,
if called on EMMA from GOOFY (Group 203).

GUIDE

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

PURPOSE Menu driven guide to use facilities.

ARGUMENTS

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

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

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

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

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

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

Function

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

Next menu

Entering a number the menu of the selected topc is displayed or the command procedure is executed, respectively.

Enter number to select topic : 2 (select topic 2)

Exit a menu

The previous menu is entered by typing "B" or "b" or <CTRL>Z or just <RETURN>. The guide is left by "E" or "e" or <CTRL>Y.

Enter number to select topic : e (exit GUIDE)

Enter number to select topic : B (prev. menu)

Select menu

To enter another menu directly, one can specify a level expression by "=n1.n2....". Then GUIDE enters the menu n1.n2.n3 (counting from the beginning). This is the same as leaving GUIDE and calling again with the level specification.

Enter number to select topic : =1.2.1 (select menu 1.2.1)

Enter number to select topic : = (select top menu)

One can jump over menus by input of a level expression "n1.n2.n3". Then these topics are selected beginning from the current one.

Enter number to select topic : 2.4 (select topic 2.4)

DCL proc.

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

Enter number to select topic : @DIR *.PPL

Enter number to select topic : @@dclproc

HELP

DCL help can be invoced directly by:

Enter number to select topic : H keywords

Examples

EXAMPLE

```
GUIDE ! display facilities
  GUIDE GOOSY ! enter first level menu
  GUIDE GOOSY 1.2 ! enter menu 1.2
  GUIDE/OUTPUT=GOOSY_GUIDE.LIS GOOSY /LIST
                        ! write all levels into file
  GUIDE GOOSY * ! display all levels and
                        ! prompt for level
```

Examples for answering the prompt:

Enter number to select topic : 2 (select topic 2)

Enter number to select topic : 2.4 (select topic 2.4)
starting at current level.

Enter number to select topic : =1.2.1 (select menu 1.2.1)
starting from level 0.

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

Guide-Programming

Function

GUIDE processes two kinds of files:

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

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

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

Menu-Design

The menus ore defined in text files
GU_facility_menu.TXT

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

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

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

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

```
! comment line
XYZ—— menu line (enters next menu)
@ABCDE—— menu line (executes DCL procedure)
        continuation without double bars
```

Here, topic 1 enters the next menu level reading text from file GU_facility_XYZ.TXT

Topic 2 executes a DCL procedure named GU_facility_ABCDE.COM The text files should not contain more than 19 true text lines (without comments).

Command-procedures

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

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

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

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

The answer is in global symbol PROMPT_ANSWER.

If the prompt was broken by <CTRL>Z , a 3 is returned in $STATUS and PROMPT_ANSWER is "". The default specification is optional. There can be optionally specified a HELP keyword as P4. Then a ? as input enters HELP with that keyword. The guide procedures may also be call MDCLLIST to get parameters. The procedures are always called with a ? as P1. Then MDCLLIST enters a parameter menu.

GUIDE sets a global symbol GUIDE_VERB (verbosity).

GUIDE_VERB is TRUE by default.

GUIDE_VERB is FALSE if GUIDE is called with /BRIEF.

Using that symbol one can control the verbosity of output.

GUIDE-initialization

Specific initializations for a guide can be done in a command procedure named

```
GUIDE$facil:GU_facil_INITIALIZATION
```

This procedure is called before any menu is entered.

It is not called for listings (/LIST/FILE/MENU).

GUIDE-finish

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

Guide-guide

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

Qualifiers

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

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

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

LSHARIM

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

PURPOSE Link modules into a sharable image.

ARGUMENTS

module Modules to be linked into sharable image:
1. List of file names (wildcards)
2. @file contains file names
3. @library(module name list)

image Name of generated executable sharable image

/GLOBAL=list Globals which occur in the modules

/SHARE*LOG=n Logical name of the sharable image

/MAP=mapfile Optional map file.

/KEEP Keep all temporary files

/GROUP The logical name for the sharable image is entered in the GROUP table instead of the JOB table.

/DEBUG Link with debugger.

DESCRIPTION

CALLING

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

/MAP=mapfile
/KEEP
/DEBUG

ARGUMENTS

module Modules to be linked into the sharable image. The following inputs are possible:

1.) A VAX/VMS file specification list with any wildcards; e.g. X$*USER*. The default file type is .OBJ. The specified modules have to be compiled.

2.) A file, which contains the list of modules to be linked into the sharable image. The file has to be specified with a leading "@"; e.g.

@list.dat

Per line one module name is expected. The modules have to be compiled.

3.) A library specification; e.g

@opriv.olb(X$*)
@opriv.olb(X$USER_ANAL)

image Name of generated executable sharable image file. Default type is .EXE.

/GLOBAL=list All FORTRAN COMMON blocks or PL/I EXTERNAL variables have to be placed in unshared sections. Therefore all globals occuring in your program have to be known. If only one global global parameter occurs define it directly:

/GLOBAL=name

Futhermore a file containing a list of all used global paramters can be specified,e.g.:

/GLOBAL=@list.dat

In each line one global parameter name is assumed.

/SHARE*LOG=n Logical name assigned to the the sharable image in JOB table. If not specified, the sharable image name is used.

/MAP=mapfile Optional map file.

/GROUP The logical name for the sharable image is entered in the GROUP table instead of the JOB table. You need GRPNAM priviledge for that. Note that the the logical name is valid for all sessions in the group.

/KEEP Keep all temporary files

/DEBUG Link with debugger.

FUNCTION

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

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

Example

LSHARIM x$*.obJ anal.exe /MAP=anal.map/share=usershr

All modules starting with X$ are linked into the generated sharable image "ANAL.EXE". The map file "ANAL.MAP" will be produced and the logical name "USERSHR" is assigned to the sharable image.

LSHARIM x$start,x$stop anal.exe

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

LSHARIM @file.dat anal.exe

All modules listed in "FILE.DAT" are used building the sharable image "ANAL.EXE".

LSHARIM @opriv(X$*) anal.exe

All modules X$* foung in the obejct library opriv are linked to a sharable image.

MANUAL

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

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

DESCRIPTION

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

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

MTAPE

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

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

Description

FUNCTION

The tape is optionally initialized and then mounted. The density specification is used for initialization, the blocksize for mounting.

If /DISMOUNT is specified, the tape presently mounted (if any) is dismounted. Then You must mount the new tape on the device, and enter <RETURN> to continue. Without the /DISMOUNT qualifier it is assumed that the desired tape volume is already mounted on the device.

Version	1.01
Author	H.G.Essel
Last Update	8-OCT-1987

OPSER

OPSER command

PURPOSE Execute priviledged operator commands

ARGUMENTS

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

Description

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

EXAMPLE To enter main menu:
$ OPSER
To enter sub menu for REPLY:
$ OPSER REPLY

PLOTMET

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

PURPOSE Plot a metafile on specified plotter

ARGUMENTS

metafile Name of the metafile which should be plotted on the specified queue or physical device.

type Device type for format. The following types are supported by GOOSY:

1.) LN03 Laser printer (=default)
2.) HP7550A3,HP7550A4 pen plotter
3.) POST postscript

command Optional print command (enclose in ""). If specified, plotter is ignored.

plotter Queue name or physical address of the plotter. If a colon (":") is specified at this position it is assumed that a physical adress has beeen specified. Is ignored when a print command is given.

copies Number of copies which should be printed. If no Printer queue is specified "copies" is ignored. (default=1)

font Font to be used to modify the default text bundle table. This argument could be used to produce pictures with nicer lettering. (default=0)

Description

FUNCTION This procedure plots the specified metafile on a plotter.

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

Example

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

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

SELECT_MBD

SELECT_MBD mbd

PURPOSE Select a valid MBD controller on a VAX

ARGUMENTS

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

Description

CALLING SELECT_MBD mbd

ARGUMENTS

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

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

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

EXAMPLE SELECT_MBD B ! Selects the 2nd MBD

Utility UTIL

Home direct. GOO$EXE

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

SETMESSAGE

SETMESSAGE facility qualifier

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

Description

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

GOOSY

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

Example

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

VMS

CALLING SETMESSAGE VMS /ON/NOPREFIX

/ON Switch ON all output of VMS Messages

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

Example

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

TLOCK

TLOCK

PURPOSE Lock terminal by password

Description

FUNCTION Locks terminal by password. The password is prompted after call. Then it must be reentered to leave the procedure. If you forget the password, the job must be canceled!

Version 1.01

Author H.G.Essel

Last Update 1-DEC-1988

VMESTRUC

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

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

Description

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

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

Syntax

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

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

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

EXAMPLE

S1

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

S2

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

S3

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

```
            long z_4
        endstr
    endstr
generates PL/1:
    DCL 1 SN$z ,
        2 LA$z_1 BIN FIXED(31),
        2 SN$z_2(10) ,
        3 LN$z_3 BIN FIXED(31),
        3 LN$z_4 BIN FIXED(31);
and C:
    struct s_z
    {
    long l_z_1;
    struct s_z_2
    {
    long l_z_3;
    long l_z_4;
    } z_2;
    } z;
```

WCLOSE

WCLOSE file

PURPOSE Wait for file to be closed.

ARGUMENTS

file File to be checked.

Description

FUNCTION Trys to open specified file. If the file is locked, it waits and retries, if not the file is closed. This command should be used to wait for an analysis writing an output file after closing the output file by STOP ANAL OUT /CLOSE because pending buffers are written before the file is closed. If in a DCL procedure the analysis would be deleted after the STOP command, the last buffer cannot be written into the file.

EXAMPLE

```
$ GOOSY START ANAL OUT X.LMD
  $ GOOSY START INPUT FILE Y.LMD
  $ MGOOWAIT ...
  $ GOOSY STOP ANAL OUT /CLOSE
  $ WCLOSE X.LMD
  $ GOOSY DELETE PROCESS $ANL
  $ ...
```

Version 1.01

Author H.G.Essel

Last Update 15-FEB-1989

Chapter 3

Macros

$ACCU

$ACCU(type,base,dir,name,incr,dim,x1,x2,...)

PURPOSE Accumulate spectrum

ARGUMENTS

type Type of spectrum (S,L,I or R)

base Name of data base (plain text)

dir Directory (plain text)

name Data element name (plain text)

incr Increment (expression)

dim Dimensionality (1 or 2)

x1,x2.. Expressions to calculate the bin number (as many as dim)

FUNCTION Generates code to accumulate spectra. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The spectrum must be located by $LOC(SPEC,...) The macro expansion may be controlled to

expand inline code
expand fast inline code
expand subroutine call

The fast inline code does NOT check freeze bits
does NOT set execute bits
does NOT increment counters

The modes can selected by the COMPILE switches:
/FAST /CALL /MACRO (=default)
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

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

```
$ACCU(L,db,$spectrum,s1,1,1,x);
$ACCU(R,db,$spectrum,s2,1,2,x,y);
$ACCU(I,db,$spectrum,s3,1,2,x,y);
$ACCU(S,db,$spectrum,s3,1,2,x,y);
```

Version 1.01

Author H.G.Essel

Last Update 27-AUG-1985

Include name GOOINC($ACCU)

$ACCU1

$ACCU1(type,base,dir,name,ind,incr,dim,x1,x2,...)

PURPOSE Accumulate 1-dim. indexed spectrum

ARGUMENTS

type	Type of spectrum (S,L,I or R)
base	Name of data base (plain text)
dir	Directory (plain text)
name	Data element name (plain text)
ind	Name index (expression)
incr	Increment (expression)
dim	Dimensionality (1 or 2)
x1,x2..	Expression to calculate bin number (as many as dim)

FUNCTION Generates code to accumulate spectra. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The spectrum must be located by $LOC1(SPEC,...) The macro expansion may be controlled to

expand inline code
expand fast inline code
expand subroutine call

The fast inline code does NOT check freeze bits
does NOT set execute bits
does NOT increment counters

The modes are selected by the COMPILE switches:
/FAST /CALL /MACRO (=default)
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$ACCU1(L,db,$spectrum,tof,5,inc,1,x);
$ACCU1(R,db,$spectrum,ede,7,inc,2,e,de);
$ACCU1(I,db,$spectrum,ede,7,inc,2,e,de);
```

Version 1.01

Author H.G.Essel

Last Update 27-AUG-1985

Include name GOOINC($ACCU1)

$ACCU2

$ACCU2(type,base,dir,name,i1,i2,incr,dim,x1,x2,...)

PURPOSE Accumulate 2-dim. indexed spectrum

ARGUMENTS

type Type of spectrum (S,L,I or R)

base Name of data base (plain text)

dir Directory (plain text)

name Data element name (plain text)

i1,i2 Name indices (expressions)

incr Increment (expression)

dim Dimensionality (1 or 2)

x1,x2.. Expression to calculuate bin number (as many as dim)

FUNCTION Generates code to accumulate spectra Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The spectrum must be located by $LOC2(SPEC,...) The macro expansion may be controlled to

expand inline code
expand fast inline code
expand subroutine call

The fast inline code does NOT check freeze bits
does NOT set execute bits
does NOT increment counters

The modes are selected by the COMPILE switches:
/FAST /CALL /MACRO (=default)
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$ACCU2(L,db,$spectrum,tof,2,1,1,1,t);
$ACCU2(R,db,$spectrum,ede,6,10,inc,2,e,de);
$ACCU2(I,db,$spectrum,ede,6,10,inc,2,e,de);
```

Version 1.01

Author H.G.Essel

Last Update 27-AUG-1985

Include name GOOINC($ACCU2)

$ATTACH

$ATTACH(type,base,access)

PURPOSE Attach data base items

ARGUMENTS

type Type of item:
BASE

base Name of data base (plain text)

access Access mode:
W for write
R for readonly

FUNCTION This macro can be called during the initialization of programs accessing data bases. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS You must include $MACRO in the program.

EXAMPLE
```
@INCLUDE $MACRO($MACRO);
  $ATTACH(BASE,db,W);
  IF ^ STS$success THEN @RET(STS$value);
```

Version 1.01

Author H.G.Essel

Last Update 27-AUG-1985

Include name GOOINC($ATTACH)

$COND

$COND(type,base,dir,name,result,dim,x1,x2,...)

PURPOSE Executes condition and returnes result.

ARGUMENTS

type Type of condition:
MW MWI WC ANY INCL IDENT EXCL POLY

base Name of data base (plain text)

dir Directory of condition (plain text)

name name of condition (plain text)

result Result variable (BIT1 except BF31 for MW)

dim Dimensionality (must be 1 for MW)

x1,x2... Expression to be tested (as many as dim)

FUNCTION Generates code to check condition. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The condition must be located by $LOC(COND,...)
You must include $MACRO in the program.
Condition can be executed several times.
The macro expansion may be controlled to
expand inline code
expand fast inline code
expand subroutine call
The fast inline code does NOT check freeze bits
does NOT set execute bits
does NOT increment counters
The modes are selected by the COMPILE switches:
/FAST /CALL /MACRO (=default)

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$COND(WC,db,$condition,win,B_res,1,X);
$COND(WC,db,$condition,win,B_res,2,X,Y);
$COND(ANY,db,$condition,pat,B_res,1,X);
$COND(MW,db,$condition,multi,L_res,1,X);
$COND(MWI,db,$condition,multi,L_res,1,X);
$COND(POLY,db,$condition,pl,B_res,1,X,Y);
```

Version 1.01

Author H.G.Essel

Last Update 27-NOV-1986

Include name GOOINC($COND)

MW

Multi window, dim=1.
result is BIN FIXED(31)
Object must be BIN FLOAT(24). Result is the number of the LAST matching subwindow. The dimension parameter is ignored. All bits of the subwindows are set if true. If the subwindows overlap, the index of the last matching is returned. The order of subwindows is the order of checking. All subwindows are checked to set the result bits.

MWI

Multi window, dim=1.
result is BIN FIXED(31)
Object must be BIN FLOAT(24). Result is the number of the FIRST matching subwindow. The dimension parameter is ignored. NO bits of the subwindows are set. If the subwindows overlap, the index of the first matching is returned. The order of subwindows is the order of checking.
This type should be used if the subwindows do not
overlap, because checking is terminated after the first true subwindow.
In /FAST mode the condition result (index) cannot
be used in a subsequent dynamic list.

WC

Window, dim=1...4.
result is BIT(1) ALIGNED

Objects must be BIN FLOAT(24). Result is TRUE, if
all objects are inside their subwindow limits

INCL

Pattern condition, dim=1...4.
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
(object & pattern) = pattern
all subchecks must be true

ANY

Pattern condition, dim=1...4.
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
(object & pattern) ^ = 0
all subchecks must be true

IDENT

Pattern condition, dim=1...4
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
object = pattern
all subchecks must be true

EXCL

Pattern condition, dim=1...4
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
(object & pattern = object)
all subchecks must be true

POLY

Polygon condition, dim=2

result is BIT(1) ALIGNED
Objects must be BIN FLOAT(24). Result is TRUE, if
point is inside the polygon

$COND1

$COND1(type,base,dir,name,ind,result,dim,x1,x2,...)

PURPOSE Executes 1-dim. indexed condition and returnes result

ARGUMENTS

type Type of condition:
MW MWI WC ANY INCL IDENT EXCL POLY

base Name of data base (plain text)

dir Directory of condition (plain text)

name name of condition (plain text)

ind Name index (expression)

result Result variable (BIT1 except BF31 for MW)

dim Dimensionality (must be 1 for MW)

x1,x2... Expression to be tested (as many as dim)

FUNCTION Generates code to check condition. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The condition must be located by $LOC1(COND,...)
Condition can be executed several times.
The macro expansion may be controlled to
expand inline code
expand fast inline code
expand subroutine call
The fast inline code does NOT check freeze bits
does NOT set execute bits
does NOT increment counters
The modes are selected by the COMPILE switches:
/FAST /CALL /MACRO (=default)
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$COND1(WC,db,$condition,win,3,B_res,1,X);
$COND1(WC,db,$condition,win,2,B_res,2,X,Y);
$COND1(ANY,db,$condition,pat,6,B_res,1,X);
$COND1(MW,db,$condition,multi,I,L_res,1,X);
$COND1(POLY,db,$condition,poly,10,B_res,1,X);
```

Version 1.01

Author H.G.Essel

Last Update 23-AUG-1988

Include name GOOINC($COND1)

MW

Multi window, dim=1.
result is BIN FIXED(31)
Object must be BIN FLOAT(24). Result is the number
of the LAST matching subwindow. The dimension parameter is ignored. All bits of the subwindows are set if true. If the subwindows overlap, the index of the last matching is returned. The order of subwindows is the order of checking. All subwindows are checked to set the result bits.

MWI

Multi window, dim=1.
result is BIN FIXED(31)
Object must be BIN FLOAT(24). Result is the number of the FIRST matching subwindow. The dimension parameter is ignored. NO bits of the subwindows are set. If the subwindows overlap, the index of the first matching is returned. The order of subwindows is the order of checking.
This type should be used if the subwindows do not
overlap, because checking is terminated after the first true subwindow.
In /FAST mode the condition result (index) cannot
be used in a subsequent dynamic list.

WC

Window, dim=1...4.
result is BIT(1) ALIGNED

Objects must be BIN FLOAT(24). Result is TRUE, if
all objects are inside their subwindow limits

INCL

Pattern condition, dim=1...4.
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
(object & pattern) = pattern
all subchecks must be true

ANY

Pattern condition, dim=1...4.
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
(object & pattern) ^ = 0
all subchecks must be true

IDENT

Pattern condition, dim=1...4
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
object = pattern
all subchecks must be true

EXCL

Pattern condition, dim=1...4
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
(object & pattern = object)
all subchecks must be true

POLY

Polygon condition, dim=2

result is BIT(1) ALIGNED
Objects must be BIN FLOAT(24). Result is TRUE, if
point is inside the polygon

$COND2

$COND2(type,base,dir,name,i1,i2,result,dim,x1,x2,..)

PURPOSE Executes 2-dim. indexed condition and returnes result

ARGUMENTS

type Type of condition:
MW MWI WC ANY INCL IDENT EXCL POLY

base Name of data base (plain text)

dir Directory of condition (plain text)

name name of condition (plain text)

i1,i2 Name indices (expression)

result Result variable (BF31 except BF31 for MW)

dim Dimensionality (must be 1 for MW)

x1,x2... Expression to be tested (as many as dim)

FUNCTION Generates code to check condition. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The condition must be located by $LOC2(COND,...)
Condition can be executed several times.
The macro expansion may be controlled to
expand inline code
expand fast inline code
expand subroutine call
The fast inline code does NOT check freeze bits
does NOT set execute bits
does NOT increment counters
The modes are selected by the COMPILE switches:
/FAST /CALL /MACRO (=default)
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$COND2(WC,db,$condition,win,1,2,B_res,1,X);
$COND2(WC,db,$condition,win,7,1,B_res,2,X,Y);
$COND2(ANY,db,$condition,pat,3,2,B_res,1,X);
$COND2(MW,db,$condition,multi,1,1,L_res,1,X);
$COND2(POLY,db,$condition,poly,1,1,b_res,1,X);
```

Version 1.01

Author H.G.Essel

Last Update 23-AUG-1988

Include name GOOINC($COND2)

MW

Multi window, dim=1.
result is BIN FIXED(31)
Object must be BIN FLOAT(24). Result is the number
of the LAST matching subwindow. The dimension parameter is ignored. All bits of the subwindows are set if true. If the subwindows overlap, the index of the last matching is returned. The order of subwindows is the order of checking. All subwindows are checked to set the result bits.

MWI

Multi window, dim=1.
result is BIN FIXED(31)
Object must be BIN FLOAT(24). Result is the number of the FIRST matching subwindow. The dimension parameter is ignored. NO bits of the subwindows are set. If the subwindows overlap, the index of the first matching is returned. The order of subwindows is the order of checking.
This type should be used if the subwindows do not
overlap, because checking is terminated after the first true subwindow.
In /FAST mode the condition result (index) cannot
be used in a subsequent dynamic list.

WC

Window, dim=1...4.
result is BIT(1) ALIGNED

Objects must be BIN FLOAT(24). Result is TRUE, if
all objects are inside their subwindow limits

INCL

Pattern condition, dim=1...4.
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
(object & pattern) = pattern
all subchecks must be true

ANY

Pattern condition, dim=1...4.
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
(object & pattern) ^ = 0
all subchecks must be true

IDENT

Pattern condition, dim=1...4
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
object = pattern
all subchecks must be true

EXCL

Pattern condition, dim=1...4
result is BIT(1) ALIGNED
Objects must be BIT(32) ALIGNED. They are inverted
using the invert patterns stored in the condition
(object & pattern = object)
all subchecks must be true

POLY

Polygon condition, dim=2

result is BIT(1) ALIGNED
Objects must be BIN FLOAT(24). Result is TRUE, if
point is inside the polygon

$DE

$DE(base,dir,name,member)

PURPOSE	Data elements reference
ARGUMENTS	
base	Name of data base (plain text)
dir	Directory (plain text)
name	Name (plain text)
member	Member specification of structure to be accessed.
FUNCTION	From the first three arguments a pointer name is built. This pointer points to the member expresssion The pointer is declared by the $LOC macro. Its name is P$_base_directory_name.
REMARKS	The data base and pool must be attached. The data element must be located by $LOC. You must include $MACRO in the program.
EXAMPLE	@INCLUDE $MACRO($MACRO); $DE(db,data,d1,i_s_array(2,3))=0; X=$DE(db,par,d2,l_sa_struc.x(I))+5.; $DE(db,eva,d3,event.pattern)='0'B;
Version	1.01
Author	H.G.Essel
Last Update	16-Nov-1987
Include name	GOOINC($DE)

$DE1

$DE1(base,dir,name,index,member)

PURPOSE	Data elements reference
ARGUMENTS	
base	Name of data base (plain text)
dir	Directory (plain text)
name	Name (plain text)
index	Index expression for name array
member	Member specification of structure to be accessed.
FUNCTION	From the first three arguments a pointer name is built. This pointer points to the member expresssion The pointer is declared by the $LOC1 macro. Its name is P1$_base_directory_name(i).
REMARKS	The data base and pool must be attached. The data element must be located by $LOC1. You must include $MACRO in the program.
EXAMPLE	@INCLUDE $MACRO($MACRO); $DE1(db,data,d1,5,i_s_array(2,3))=0; X=$DE1(db,par,d2,2,l_sa_struc.x(I))+5.; $DE1(db,eva,d3,4,event.pattern)='0'B;
Version	1.01
Author	H.G.Essel
Last Update	16-Nov-1987
Include name	GOOINC($DE1)

$DE2

$DE2(base,dir,name,i1,i2,member)

PURPOSE	Data elements reference (2-dim)
ARGUMENTS	
base	Name of data base (plain text)
dir	Directory (plain text)
name	Name (plain text)
i1,i2	Two index expressions for name array
member	Member specification of structure to be accessed.
FUNCTION	From the first three arguments a pointer name is built. This pointer points to the member expresssion The pointer is declared by the $LOC2 macro. Its name is P2$_base_directory_name(i,k).
REMARKS	The data base and pool must be attached. The data element must be located by $LOC2. You must include $MACRO in the program.
EXAMPLE	@INCLUDE $MACRO($MACRO); $DE2(db,data,d1,5,4,i_s_array(2,3))=0; X=$DE2(db,par,d2,K,J,l_sa_struc.x(I))+5.; $DE2(db,eva,d3,K+4,I*(J-1),event.pattern)='0'B;
Version	1.01
Author	H.G.Essel
Last Update	16-Nov-1987
Include name	GOOINC($DE2)

$DETACH

$DETACH(type,base)

PURPOSE Detach data base items

ARGUMENTS

type Type of item:
BASE

base Name of data base (plain text)

FUNCTION This macro can be called at the end of programs accessing data bases. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS You must include $MACRO in the program.

EXAMPLE
```
@INCLUDE $MACRO($MACRO);
  $DETACH(BASE,db);
  IF ^ STS$success THEN @RET(STS$value);
```

Version 1.01

Author H.G.Essel

Last Update 27-AUG-1985

Include name GOOINC($DETACH)

$LOC

$LOC(type,base,dir,name,access,descr)

PURPOSE Locate data elements for analysis

ARGUMENTS

type Type of data element:
SPEC spectrum
COND condition
DE general data element

base Name of data base (plain text)

dir Directory (plain text)

name Name (plain text)

access Access mode:
W for write
R for readonly

descr Data element type. will be checked. (plain text)

SPEC	L,I,R,S
COND	WC,PC,MW,POLY
DE	name of data element type (optional)

FUNCTION This routine must be called during the initialization of the analysis routine. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The data base and pool must be attached.
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$LOC(SPEC,db,$spectrum,tof,W,L);
```

```
$LOC(COND,db,$condition,w1,W,WC);
$LOC(DE,db,eva,event,W,SE$E1_1);
```

Version	1.01
Author	H.G.Essel
Last Update	27-AUG-1985
Include name	GOOINC($LOC)

SPEC

Any spectrum accessed by $ACCU must be located first by this macro:
$LOC(SPEC,base,dir,name,W,t);
$SECDEF must be included.
Four pointers are declared for each spectrum:
P$_base_directory_spectrum_t used by $ACCU
P$_base_directory_spectrum_$H points to SE$SPHE
P$_base_directory_spectrum_$A points to SE$SPDTT
P$_base_directory_spectrum_$D points to SE$SPDti
where t=I,L,S or R and i=1 or 2

COND

Any condition accessed by $COND must be located first by this macro:
$LOC(COND,base,dir,name,W,t);
$SECDEF must be included.
Three pointers are declared for each condition:
P$_base_directory_condition_t used by $COND
P$_base_directory_condition_$H points to SE$COHE
P$_base_directory_condition_$D points to SE$COxxx
where xxx is a key for different condition types.
and t=WC,PC,MW,POLY
Command LIBLIS GOOTYP(SE$CO*) lists these names.

DE

Any data element to be accessed must be located first by this macro:
$LOC(DE,base,dir,name,W,type);
$SECDEF must be included.
After that, the pointer to the data element is:
P$_base_dir_name.

This pointer is declared as STATIC.
The length of the data element is returned in:
    L$_base_dir_name.
This Longword is declared STATIC.

$LOC1

$LOC1(type,base,dir,name,ll,ul,access,descr)

PURPOSE Locate 1-dim. data element arrays for analysis

ARGUMENTS

type Type of data element:
SPEC spectrum
COND condition
DE general data element

base Name of data base (plain text)

dir Directory (plain text)

name Name (plain text)

ll Boundary: lower limit (number)

ul Boundary: upper limit (number)

access Access mode:
W for write
R for readonly

descr Data element type. will be checked. (plain text)

SPEC	L,I,R,S
COND	WC,PC,MW,POLY
DE	name of data element type (optional)

FUNCTION This routine must be called during the initialization of the analysis routine. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The data base and pool must be attached.
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$LOC1(SPEC,db,$spectrum,tof,1,5,W,L);
$LOC1(COND,db,$condition,w1,1,10,W,WC);
$LOC1(DE,db,eva,event,-2,5,W,SE$E1_1);
```

Version 1.01

Author H.G.Essel

Last Update 27-AUG-1985

Include name GOOINC($LOC1)

SPEC

Any spectrum accessed by $ACCU must be located first by this macro:
$LOC1(SPEC,base,dir,name,1,5,W,t);
$SECDEF must be included.
Four pointers are declared for each spectrum:
P1$_base_directory_spectrum_t(k) used by $ACCU1
P1$_base_directory_spectrum_$H(k) to SE$SPHE
P1$_base_directory_spectrum_$A(k) to SE$SPDTT
P1$_base_directory_spectrum_$D(k) to SE$SPDti
where t=I,L,S or R and i=1 or 2

COND

Any condition accessed by $COND must be located first by this macro:
$LOC1(COND,base,dir,name,1,6,W,t);
$SECDEF must be included.
Three pointers are declared for each condition:
P1$_base_directory_condition_t(i) used by $COND1
P1$_base_directory_condition_$H(i) to SE$COHE
P1$_base_directory_condition_$D(i) to SE$COxxx
where xxx is a key for different condition types.
and t=WC,PC,MW,POLY
Command LIBLIS GOOTYP(SE$CO*) lists these names.

DE

Any data element to be accessed must be located first by this macro:
$LOC1(DE,base,dir,name,2,4,W,descr);

$SECDEF must be included.
After that, the pointer to the i-th data element is:
P1$_base_dir_name(i).
This pointer is declared as STATIC.

$LOC2

$LOC2(type,base,dir,name,l1,u1,l2,u2
,access,descr)

PURPOSE Locate 2-dim. data element arrays for analysis

ARGUMENTS

type Type of data element:
SPEC spectrum
COND condition
DE general data element

base Name of data base (plain text)

dir Directory (plain text)

name Name (plain text)

li,ui lower and upper boundaries of i-th dimension, i=1,2. Numbers are required here, no variables are allowed.

access Access mode:
W for write
R for readonly

descr Data element type. will be checked. (plain text)

SPEC	L,I,R,S
COND	WC,PC,MW,POLY
DE	name of data element type (optional)

FUNCTION This routine must be called during the initialization of the analysis routine. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The data base and pool must be attached.
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$LOC2(SPEC,db,$spectrum,tof,1,5,2,4,W,L);
$LOC2(COND,db,$condition,w1,1,10,1,5,W,WC);
$LOC2(DE,db,eva,event,1,2,1,2,W,SE$E1_1);
```

Version 1.01

Author H.G.Essel

Last Update 27-AUG-1985

Include name GOOINC($LOC2)

SPEC

Any spectrum accessed by $ACCU must be located first by this macro:
$LOC2(SPEC,base,dir,name,1,5,1,3,W,t);
$SECDEF must be included.
Four pointers are declared for each spectrum:
P2$_base_directory_spectrum_t(l,k) used by $ACCU
P2$_base_directory_spectrum_$H(l,k) to SE$SPHE
P2$_base_directory_spectrum_$A(l,k) to SE$SPDTT
P2$_base_directory_spectrum_$D(l,k) to SE$SPDti
where t=I,L,S or R and i=1 or 2

COND

Any condition accessed by $COND must be located first by this macro:
$LOC2(COND,base,dir,name,2,4,1,6,W,t);
$SECDEF must be included.
Three pointers are declared for each condition:
P2$_base_directory_condition_t(i,k) used by $COND2
P2$_base_directory_condition_$H(i,k) to SE$COHE
P2$_base_directory_condition_$D(i,k) to SE$COxxx
where xxx is a key for different condition types.
and t=WC,PC,MW,POLY
Command LIBLIS GOOTYP(SE$CO*) lists these names.

DE

Any data element to be accessed must be located first by this macro:
$LOC2(DE,base,dir,name,-3,5,2,5,W,descr);

$SECDEF must be included.
The pointer to the i,k-th data element is:
P2$_base_dir_name(i,k).
This pointer is declared as STATIC.

$MACRO

@INCLUDE $MACRO($MACRO)

PURPOSE Initialize analysis macros

FUNCTION Must be included if any analysis macro is called like $LOCx, $CONDx or $ACCUx.

$SPEC

$SPEC(type,base,dir,name,value,dim,x1,x2,...)

PURPOSE Set spectrum channel

ARGUMENTS

type Type of spectrum (L,I or R)

base Name of data base (plain text)

dir Directory (plain text)

name Data element name (plain text)

value Channel value (expression)

dim Dimensionality (1 or 2)

x1,x2.. Expressions to calculate the bin number (as many as dim)

FUNCTION Generates code to set spectrum channel. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The spectrum must be located by $LOC(SPEC,...) The macro expansion may be controlled to

expand inline code
expand fast inline code
expand subroutine call

The fast inline code does NOT check freeze bits
does NOT set execute bits
does NOT increment counters

The modes can selected by the COMPILE switches:
/FAST /CALL /MACRO (=default)
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$SPEC(L,db,$spectrum,s1,1,1,x);
$SPEC(R,db,$spectrum,s2,1,2,x,y);
$SPEC(I,db,$spectrum,s3,1,2,x,y);
```

Version 1.01

Author H.G.Essel

Last Update 27-AUG-1985

Include name GOOINC($SPEC)

$SPEC1

$SPEC1(type,base,dir,name,ind,value,dim,x1,x2,...)

PURPOSE Set channel in 1-dim. indexed spectrum

ARGUMENTS

type	Type of spectrum (L,I or R)
base	Name of data base (plain text)
dir	Directory (plain text)
name	Data element name (plain text)
ind	Name index (expression)
value	Channel value (expression)
dim	Dimensionality (1 or 2)
x1,x2..	Expression to calculate bin number (as many as dim)

FUNCTION Generates code to set spectrum channel. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The spectrum must be located by $LOC1(SPEC,...) The macro expansion may be controlled to

expand inline code
expand fast inline code
expand subroutine call

The fast inline code does NOT check freeze bits
does NOT set execute bits
does NOT increment counters

The modes are selected by the COMPILE switches:
/FAST /CALL /MACRO (=default)
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$SPEC1(L,db,$spectrum,tof,5,inc,1,x);
$SPEC1(R,db,$spectrum,ede,7,inc,2,e,de);
$SPEC1(I,db,$spectrum,ede,7,inc,2,e,de);
```

Version 1.01

Author H.G.Essel

Last Update 27-AUG-1985

Include name GOOINC($SPEC1)

$SPEC2

$SPEC2(type,base,dir,name,i1,i2,value,dim,x1,x2,...)

PURPOSE Set channel in 2-dim. indexed spectrum

ARGUMENTS

type Type of spectrum (L,I or R)

base Name of data base (plain text)

dir Directory (plain text)

name Data element name (plain text)

i1,i2 Name indices (expressions)

value Channel value (expression)

dim Dimensionality (1 or 2)

x1,x2.. Expression to calculuate bin number (as many as dim)

FUNCTION Generates code to set spectrum channel. Plain text means that these arguments must not be enclosed in quotes and must not be PL/1 variables.

REMARKS The spectrum must be located by $LOC2(SPEC,...) The macro expansion may be controlled to

expand inline code
expand fast inline code
expand subroutine call

The fast inline code does NOT check freeze bits
does NOT set execute bits
does NOT increment counters

The modes are selected by the COMPILE switches:
/FAST /CALL /MACRO (=default)
You must include $MACRO in the program.

EXAMPLE see example routine GOO$TEST:X$ANAL.PPL

```
@INCLUDE $MACRO($MACRO);
$SPEC2(L,db,$spectrum,tof,2,1,1,1,t);
$SPEC2(R,db,$spectrum,ede,6,10,inc,2,e,de);
$SPEC2(I,db,$spectrum,ede,6,10,inc,2,e,de);
```

Version 1.01

Author H.G.Essel

Last Update 27-AUG-1985

Include name GOOINC($SPEC2)

ADD_MSG

@ADD_MSG(errorcode,arg1,arg2,arg3)

PURPOSE	accomplish the error message belonging to the errorcode by the specified arguments and write the message on the internal error- message stack.
ARGUMENTS	
errorcode	name of error
argi	parameters for the message text
Include name	-

Description

CALLING	@ADD_MSG(errorcode,arg1,arg2,arg3)
ARGUMENTS	
errorcode	name of error for which the message should be written
argi	parameters for the message text. Subsequent !AS in the message text as it is defined , will be replaced by specified arguments. Arguments can be omitted from the right.
FUNCTION	The message belonging to the specified error code will be retrieved and accomplished by the given additional arguments if necessary. THe complete message will then be written to the internal message stack, where it is held for further processing (see @DMP_CLR_MSG, @PUT_CLR_MSG).
REMARKS	the arguments argi are of type CHAR VAR, however on the VAX they can be of any type.
EXAMPLE	@ADD_MSG(XUTIL_NOOUTPUT,'U_OUT','CV_LONG'); will generate the following message and write it to the message stack: 'U_OUT tried to output CV_LONG, but did not find a valid control pattern'

BYTE

@BYTE(integer)

PURPOSE returns the ASCII(EBCDIC) character whose code is equivalent to the given integer.

ARGUMENTS

integer integer number

Include name -

Description

CALLING @BYTE(integer)

ARGUMENTS

integer integer number, may be BIN FIXED(15) or BIN FIXED(31

FUNCTION the binary representation of the argument <integer> is interpreted as character in ASCII (VAX) or EBCDIC (IBM) format.

REMARKS If the given number exceeds the valid range of 0 to 255, an error will be signaled.

EXAMPLE

```
C=@BYTE(32);
  a blank will be returned on the VAX,
C=@BYTE(64);
  a blank will be returned on the IBM, a '@' on
the VAX.
```

CALL

@CALL procedure

PURPOSE	performs a function call
ARGUMENTS	
procedure	procedure name with arguments
Include name	-

Description

CALLING	@CALL procedure
ARGUMENTS	
procedure	name of procedure followed by the arguments in brackets
FUNCTION	'@CALL' will be replaced by the string 'STS$VALUE=' Care will be taken of the existance of the declaration for the errors (an include of $STSDEF)
REMARKS	implemented as a string variable.
EXAMPLE	@CALL U$PRTCL(CV,U$MPRTTERM);

CLR_MSG

@CLR_MSG

PURPOSE clear the internal message stack on

ARGUMENTS

Include name -

Description

CALLING @CLR_MSG

ARGUMENTS

FUNCTION The internal message stack will be cleared.

REMARKS -

EXAMPLE
```
@CALL MYSUB(CV_NAME,I_NUMBER);
  IF STS$SUCCESS THEN @CLR_MSG;
```

DCL_MSG

@DCL_MSG(errorname);

PURPOSE	declaration of error
ARGUMENTS	
error name	name of error to be declared
Include name	-

Description

CALLING	@DCL_MSG(<error name>);
ARGUMENTS	
error name	name of error, looks like X<fac>_<name> where <fac> is a facility key word and <name> is the name of the specific error.
FUNCTION	the error is declared as GLOBAL REF VALUE
REMARKS	do not declare several errors in one declaration @DCL_MSG
EXAMPLE	@DCL_MSG(XTEST_ER);

DMP_CLR_MSG

@DMP_CLR_MSG

PURPOSE write the internal message stack on the screen

ARGUMENTS

Include name -

Description

CALLING @DMP_CLR_MSG

ARGUMENTS

FUNCTION The internal message stack will be written to the terminal. The stack will be cleared. All messages will be written, regardless of the message profile set (good for test purposes, for message profile dependent output see @PUT_CLR_MSG).

REMARKS -

EXAMPLE

```
@CALL MYSUB(CV_NAME,I_NUMBER);
  IF ^ STS$SUCCESS THEN @DMP_CLR_MSG;
```

ENTRY

label: @ENTRY

PURPOSE	remember name of entry
ARGUMENTS	
label	name of the entry
Include name	-

Description

CALLING	label: @ENTRY
ARGUMENTS	
label	name of the entry
FUNCTION	The name of the entry (label) will be memorized until a redefinition (via @ENTRY or @PROCEDURE) takes place. This name will be used as a prefix for all error messages.
REMARKS	The syntax will be changed into @ENTRYPLI(label) in the near future. However the old form will be understood.
EXAMPLE	S$EXAE:@ENTRY(I_1) RETURNS(BIN FIXED(31)) ;

INCLUDE

@INCLUDE lib(member)

PURPOSE	include PPL code
ARGUMENTS	
lib	library
member	module in library (member of PDS)
Include name	-

Description

CALLING	@INCLUDE lib(member)
ARGUMENTS	
library	DEC library from which a module should be included (not yet implemented), or DD-name which is related by an ALLOC-statement (IBM-MVS). If omitted, <member> is interpreted as file specification (VAX only).
member	module in library(n.y.i.) or member of PDS or VAX file specification.
FUNCTION	The specified source code is included.
REMARKS	The VAX library handling is not yet implemented.
EXAMPLE	@INCLUDE $MACRO(U$PRTCL); the declaration of the routine U$PRTCL is included.

LOCAL_ERROR

@LOCAL_ERROR()

PURPOSE resignals errors from lower procedure levels

ARGUMENTS

Include name -

Description

CALLING @LOCAL_ERROR()

ARGUMENTS

FUNCTION An On-unit, written to catch errors which happen in the local routine ,catches also errors from lower level procedures. @LOCAL_ERROR() will catch in that on-unit the error from lower routines, and will resignal them to higher levels.

REMARKS -

EXAMPLE

```
ON FIXEDOVERFLOW BEGIN;
   @LOCAL_ERROR();
   @CALL U$PRTCL('fixed overflow in my routine',
                 U$M_PRTT);
END /* of ON FIXEDOVERFLOW */;
```

ON_ANY_E

@ON_ANY_E(u_cleanup)

PURPOSE catches all signaled errors, calls <u_cleanup> before resignaling the error

ARGUMENTS

u_cleanup routine to be called after detecting the error and before resignaling

Include name -

Description

CALLING @ON_ANY_E(u_cleanup)

ARGUMENTS

u_cleanup Routine which will be called when handling the signaled error. Arguments may be passed .

FUNCTION All signaled error will be detected by @ON_ANY_E. Following actions will take place:
If specified the routine u_cleanup will be called.
This routine serves to make things undone which has previously been performed in the current routine, e.g. free allocated space, close opened files a.s.o.
Any error during the clean-up will be caught by an internal On-unit and will be resignaled as an unrecoverable fatal error which passes all higher on-units.
The message related to the occurred error will be written on the internal error stack. The text will contain the name of current routine if the macros @PROCEDURE or @ENTRY are used.
Depending on the severity, the error will be resignaled or converted to a reported error (RETURN(errorcode).
Here: error of severity E (error) and less will not be resignaled, but a non local GOTO will be performed and the current routine will return the error code to the calling routine.

REMARKS Due to the lack of several severities in the system commands on the IBM, conversions to reported errors will not take place on these computers. All errors will then be resignaled.

EXAMPLE @ON_ANY_E(U_CLUP(I_COUNT));
the above described actions will take place. An internal subroutine U_CLUP is called from within the standard On-unit, the argument I_COUNT is passed.

ON_ANY_F

@ON_ANY_F(u_cleanup)

PURPOSE	catches all signaled errors, calls <u_cleanup> before resignaling the error
ARGUMENTS	
u_cleanup	routine to be called after detecting the error and before resignaling
Include name	-

Description

CALLING	@ON_ANY_F(u_cleanup)
ARGUMENTS	
u_cleanup	Routine which will be called when handling the signaled error. Arguments may be passed .
FUNCTION	All signaled error will be detected by @ON_ANY_F. If specified the routine u_cleanup will be called. This routine serves to make things undone which has previously been performed in the current routine, e.g. free allocated space, close opened files ,... Any error during the clean-up will be caught by an internal On-unit and will be resignaled as an unrecoverable fatal error The message related to the occurred error will be written on the internal error stack. The text will contain the name of current routine if the macros @PROCEDURE or @ENTRY are used. Depending on the severity, the error will be resignaled or converted to a reported error (RETURN(errorcode). Here: all errors will not be resignaled, but a non local GOTO will be performed and the current routine will return the error code to the calling routine.

REMARKS Due to the lack of several severities in the system commands on the IBM, conversions to reported errors will not take place on these computers. All errors will then be resignaled.

EXAMPLE @ON_ANY_F(U_CLUP(I_COUNT));
the above described actions will take place. An internal subroutine U_CLUP is called from within the standard On-unit, the argument I_COUNT is passed.

ON_ANY_W

@ON_ANY_W(u_cleanup)

PURPOSE	catches all signaled errors, calls <u_cleanup> before resignaling the error
ARGUMENTS	
u_cleanup	routine to be called after detecting the error and before resignaling
Include name	-

Description

CALLING	@ON_ANY_W(u_cleanup)
ARGUMENTS	
u_cleanup	Routine which will be called when handling the signaled error. Arguments may be passed .
FUNCTION	All signaled error will be detected by @ON_ANY_W. Following actions will take place: If specified the routine u_cleanup will be called. This routine serves to make things undone which has previously been performed in the current routine, e.g. free allocated space, close opened files ,... Any error during the clean-up will be caught by an internal On-unit and will be resignaled as an unrecoverable fatal error which passes all higher on-units. The message related to the occurred error will be written on the internal error stack. The text will contain the name of current routine if the macros @PROCEDURE or @ENTRY are used. Depending on the severity, the error will be resignaled or converted to a reported error (RETURN(errorcode). Here: error of severity W (warning) and less will not be resignaled, but a non local GOTO will be performed and the current routine will return the error code to the calling routine

REMARKS Due to the lack of several severities in the system commands on the IBM, conversions to reported errors will not take place on these computers. All errors will then be resignaled.

EXAMPLE @ON_ANY_W(U_CLUP(I_COUNT));
the above described actions will take place. An internal subroutine U_CLUP is called from within the standard On-unit, the argument I_COUNT is passed.

PROCEDURE

<label>:@PROCEDURE

PURPOSE remembers the name of the current module

ARGUMENTS

Include name -

Description

CALLING <label>:@PROCEDURE

ARGUMENTS

FUNCTION The name of the current procedure is taken from the name of <label> and will later be inserted in all error messages uttered in this module.

REMARKS The syntax of this macro will later be changed into @PROCPLI(<label>)

EXAMPLE

```
U$PRTCL: @PROCEDURE
                (CV_OUT,B32) RETURNS(BIN FIXED(31));
```

PUT_CLR_MSG

@PUT_CLR_MSG

PURPOSE write the internal mesage stack on the screen

ARGUMENTS

Include name -

Description

CALLING @PUT_CLR_MSG

ARGUMENTS

FUNCTION The internal message stack will be written to the The message profile, as set by a call to S$MSPRO will be considered.

REMARKS -

EXAMPLE
```
@CALL MYSUB(CV_NAME,I_NUMBER);
  IF ^ STS$SUCCESS THEN @PUT_CLR_MSG;
```

RANK

@RANK(char)

PURPOSE returns a BIN FIXED (15) number which corresponds to the input character <char>.

ARGUMENTS

char character whose binary representation will be inter- preted

Include name -

Description

CALLING @RANK(char)

ARGUMENTS

char single character, input

FUNCTION the binary representation of the input character will be interpreted as an integer number, put into the low order byte of a BIN FIXED(15) number, which will then be returned.

REMARKS has the same functionality as the VAX builtin function RANK.

EXAMPLE

```
IF @RANK(SUBSTR(CV_NAME1,1,1))>
    @RANK(SUBSTR(CV_NAME2,1,1) THEN DO;
```

REPEAT

@REPEAT(cv,i_repeat)

PURPOSE	return the string cv concatenated to cv i_repeat times
ARGUMENTS	
cv	character string
i_repeat	number of concatenations

Description

FUNCTION	The string <cv> will be concatenated to itself I_repeat times, the result will be returned. Note: the resulting string contains one time CV more than the result of the VAX-PLI builtin function COPY.
Example	CV=@REPEAT('ei',3) ; CV gets the value 'eieieei'.
File name	GOOMINC(REPEAT)
Dataset	-
Version	1.01
Author	K.Winkelmann
Last Update	24-JUN-1985

RET

@RET(errorcode)

PURPOSE returns the error code to the calling procedure,

ARGUMENTS

errorcode name of error or number

Include name -

Description

CALLING @RET(errorcode)

ARGUMENTS

errorcode name of error

FUNCTION @RET(errorcode) will be substituted by RETURN(errorcode)

REMARKS -

EXAMPLE @RET(1)
successful completion

RET_ADD_MSG

@RET_ADD_MSG(errorcode,arg1,arg2,arg3)

PURPOSE	return and write specified message onto error stack
ARGUMENTS	
errorcode	name of error
argi	parameters for the related message
Include name	-

Description

CALLING	@RET_ADD_MSG(errorcode,arg1,arg2,arg3)
ARGUMENTS	
errorcode	name of error or number to be returned to calling procedure
argi	arguments which will be substituted in the message text
FUNCTION	the message text related to the given error code will be retrieved, parameters will be substituted and the accomplished text is written to the internal error stack. Then the procedure returns the error number to the calling procedure.
REMARKS	The syntax will be changed later like @RET_ADD_MSG errorcode arg1 arg2 arg3
EXAMPLE	IF ^ STS$SUCCESS THEN @RET_ADD_MSG(STS$VALUE);

RET_SET_MSG

@RET_SET_MSG(errorcode,arg1,arg2,arg3)

PURPOSE	return and write specified message onto error stack
ARGUMENTS	
errorcode	name of error
argi	parameters for the related message
Include name	-

Description

CALLING	@RET_SET_MSG(errorcode,arg1,arg2,arg3)
ARGUMENTS	
errorcode	name of error or number to be returned to calling procedure
argi	arguments which will be substituted in the message text
FUNCTION	the message text related to the given error code will be retrieved, parameters will be substituted and the accomplished text is written to the internal error stack after it has been cleared. Then the procedure returns with the error number to the calling procedure. This macro is useful if a new error message makes previous ones obsolete.
REMARKS	The syntax will be changed later like @RET_SET_MSG errorcode arg1 arg2 arg3
EXAMPLE	IF ^ STS$SUCCESS THEN @RET_SET_MSG(STS$VALUE);

SIZE

@SIZE(reference)

PURPOSE	returns number of bytes allocated to the referenced variable
ARGUMENTS	
reference	name of variable whose size is wanted

Description

FUNCTION	The number of bytes allocated for the referenced variable <reference> is returned (VAX-PLI builtin function SIZE).
File name	GOOMINC(SIZE)
Dataset	-
Version	-
Author	K.Winkelmann
Last Update	24-JUN-1985

STORAGE

@STORAGE(reference)

PURPOSE	returns number of bytes allocated to the referenced variable
ARGUMENTS	
reference	name of variable whose size of storage is wanted

Description

FUNCTION	The number of bytes allocated for the referenced variable <reference> is returned (IBM-PLI builtin function STORAGE).
File name	GOOMINC(STORAGE)
Dataset	-
Version	-
Author	K.Winkelmann
Last Update	24-JUN-1985

TRACE_MSG

@TRACE_MSG(errorcode,arg1,arg2,arg3)

PURPOSE	Write e trace message to the internal error stack. The errorcode is normally returned by another routine signaling an error.
ARGUMENTS	
errorcode	name of error
argi	parameters for the message text. Normally not used.
Include name	-

Description

CALLING	@TRACE_MSG(errorcode,arg1,arg2,arg3)
ARGUMENTS	
errorcode	name of error for which the trace message should be written
argi	parameters for the message text. Not used, if the errorcode was returned by a GOOSY routine call.
FUNCTION	If the message belonging to the specified error code is already on the error stack, a trace message is added.
REMARKS	the arguments argi are of type CHAR VAR, however on the VAX they can be of any type.
EXAMPLE	@CALL U$xxx(); IF ^ STS$SUCCESS THEN DO; @TRACE_MSG(STS$VALUE); GOTO ERROR; END; U$xxx returned an error. Write trace message and handle error.

TRIM

@TRIM(cv_string,cv_lead,cv_trail)

PURPOSE	remove leading and/or trailing characters from a string
ARGUMENTS	
cv_string	string to be trimmed
cv_lead	set of characters to be removed from left
cv_trail	set of characters to be removed from right
Include name	-

Description

CALLING	@TRIM(cv_string,cv_lead,cv_trail)
ARGUMENTS	
cv_string	input string to be trimmed
cv_lead	set of characters, each of them will be removed from the beginning
cv_trail	set of characters, each of them will be removed from right. If cv_lead or cv_trail will be ommitted, they are assumed to be ' '(blank).
FUNCTION	The TRIM function returns a character string that consists of the input string with specified characters removed from the left and right.TRIM takes either one or three arguments. If you supply second and third arguments, TRIM removes characters specified by those arguments from the left and right or the string, respectively.
REMARKS	corresponds to the VAX builtin function TRIM.
EXAMPLE	CV=@TRIM(' go to hell !!!!!!!!!!!',' ',' !'); after execution, CV will have the value CV='go to hell' . CV=@TRIM(' the red rose '); leads to CV='the red rose' .

Contents