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

G SI O nline O ffline S Y stem

GOOSY Data Management Commands

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

CALCULATE SPECTRUM

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

PURPOSE Perform spectrum arithmetic operations.

PARAMETERS

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

operand_1 First spectrum operand.

operand Aritmetic operation which should be performed.

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

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

spec_dir Default spectrum directory.

base Default data base name.

node Default node.

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

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

Caller MDBM,MGOODBM

Author W. Spreng

Example

1.) CALCULATE SPECTRUM [test]a b + [$spectrum]c
The spectra b and [$spectrum]c are added
and the result is stored in [test]a.

2.) CALCULATE SPECTRUM a b * 2.75 /constant
The contents of spectrum b is increased by the
factor 2.75 and the result is stored in spectrum
a.

3.) CALCULATE SPECTRUM a b + c 2.0
The performed operation for each bin is:
a = b + c*2.0

Remarks

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

File name GOO$DE:E$CACSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument and parameter description

RESULT

Routine arg. Input CHAR(*) VAR

Command par. String

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

OPERAND_1

Routine arg. Input CHAR(*) VAR

Command par. String

First spectrum operand.

ACTION

Routine arg. Input CHAR(*) VAR

Command par. String

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

+ addition
- subtraction
* multiplication
: division

OPERAND_2

Routine arg. Input CHAR(*) VAR

Command par. String

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

FACTOR

Routine arg. Input CHAR(*) VAR

Command par. String

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

SPEC_DIR

Routine arg. Input CHAR(*) VAR

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

Default spectrum directory.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replacable default="DB"

Default data base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replacable default="*"

Default node.

/CONSTANT

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

Command par. Switch

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

/KEEP

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

Command par. Switch

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

Function

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

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

Result spectrum of type R
Operand 1 spectrum type L

Operand 2 spectrum type R

===> is allowed!

Result spectrum of type L
Operand 1 spectrum type L
Operand 2 spectrum type R

===> is not allowed!

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

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

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

CALIBRATE SPECTRUM

CALIBRATE SPECTRUM spectrum calibration spec_dir cal_dir base node

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

Example

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

Description

CALLING

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

COMMAND CALIBRATE SPECTRUM spectrum calibration spec_dir cal_dir base node

SPECTRUM

Routine arg. CHAR(*) VAR

Command arg. String required

Name of spectrum which should be connnected with a calibration.

CALIBRATION

Routine arg. CHAR(*) VAR

Command arg. String required

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

SPEC_DIR

Routine arg. CHAR(*) VAR

Command arg. String global replaceable default=$SPECTRUM

Default Directory name for spectra.

CAL_DIR

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=$CALIB

Default Directory for calibration Data Elements.

BASE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=*

Node name for Data Base section file.

Function

FUNCTION

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

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

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

CLEAR CAMAC SPECTRUM

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

PURPOSE clear one (or all) spectrum

PARAMETERS

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

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

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

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

/LOG Switch default: ”
Output list of cleared spectra

/CAMAC Switch default: ”
Clear spectrum in CAMAC.

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

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

Caller mdbm

Author H.G.Essel

Example

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

Remarks

File name E$ACCSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

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

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

NAME

Routine arg. Input CHAR(*) VAR

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

SPEC_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

LOG

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

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

CAMAC

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

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

SPECTRUM

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

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

KEEP_MAP

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

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

Function

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

CLEAR CONDITION COUNTER

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

PURPOSE clear condition counters specified by name

PARAMETERS

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

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

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

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

/LOG Switch default: none
Output list of cleared conditions

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

Caller MDBM

Author K.Winkelmann

Example

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

Remarks

File name GOO$DE:E$CLCO.PPL

Created by GOO$DE:E$DECMD.PPL

Description

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

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

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name expression of condition. Wildcards are
accepted in the form:
* x* *x *x* x*y
Name array are supported. Index may be wildacrd.
This is assumed, if name is wildcarded. Arrays without index are cleared totally. For one dimensional arrays a range may be specified like x(3:5). No wildcard is allowed in this case. Single members of one and twodimensional arrays may be cleared by specifying the index: X(7) or Y(4,5).

DIR

Routine arg. Input CHAR(*) VAR

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

Type Input CHAR(*) VAR
Name of condition directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'

Type Input CHAR(*) VAR
Name of Data base

NODE

Routine arg. Input CHAR(*) VAR

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

Type Input CHAR(*) VAR
Name of Node

LOG

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

Command par. switch default: ''

Type Input BIN FIXED(15) Output list of cleared conditions

KEEP_MAP

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

Command par. switch default: ''

Type Input BIN FIXED(15) Inhibit unmap of data Base

Function

The condition counters are cleared

Remarks

The Module is an action routine.

Example

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

CLEAR ELEMENT

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

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

PARAMETERS

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

dir string global replace default: ’DATA’
Default directory

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

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

/LOG switch default: ”
Displays cleared Data Element.

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

Caller M$DMCMD

Author Th. Kroll

EXAMPLE

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

Remarks

File name M$ACLDE.PPL

Ceated by GOO$DM:M$DMCMD.PPL

Description

CALLING

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

COMMAND

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

Argument /Parameter description

NAME

Routine arg. Input CHAR(*) VAR

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

DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

LOG

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

Command par. switch default: ”

/LOG Display cleared Data Elements.

KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

FUNCTION

Clear data element. Reset the dataelement to zero.

CLEAR PICTURE

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

PURPOSE Clear spectra defined in a picture data element

PARAMETERS

picture Picture data element name

frame Frame specification.

pic_dir Default directory name

base Default data base name.

node Default node name

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

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

Example

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

Remarks

Created by E$DECMD.PPL

File name D$CLRPI.PPL

Description

CALLING

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

COMMAND

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

PICTURE

Routine arg. CHAR(*) VAR

Command par. String required

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

FRAME

Routine arg. CHAR(*) VAR

Command par. String required default=*

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

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

PIC_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default = $PICTURE

Default Picture Directory.

BASE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default = DB

Default Data Base name

NODE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default = $PICTURE

Default node name

/KEEP_MAP

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

Command par. Switch negatable default=/KEEP_MAP

Switch to prevent the dettachment of the Data Base

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

LOG

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

Command par. Switch negatgable default=/NOLOG

Controls if the specification of all spectra and pictures are listed.

If you use the /LOG switch the following information is displayed:

(1) the name of the picture which will be handled.

(2) all spectra in that picture which are cleared.

Function

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

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

CLEAR SPECTRUM

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

PURPOSE clear one (or all) spectrum

PARAMETERS

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

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

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

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

/LOG Switch default: ”
Output list of cleared spectra

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

Caller mdbm

Author K.Winkelmann

Example

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

Remarks

File name E$CLSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument description

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”

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

SPEC_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

LOG

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

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

KEEP_MAP

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

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

Function

The data part of the spectrum is set to zero

COMPRESS BASE

```
COMPRESS BASE base file
                 /DISMOUNT
```

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

PARAMETERS

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

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

/DISMOUNT switch default:
Dismount data base after copy.

Caller MDBCOPY

Author H.G.Essel

Example

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

Remarks

File name GOO$DM:M$ACMPB.PPL

Created by GOO$DM:M$DMCMD.PPL

Description

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

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

BASE

Routine arg. Input CHAR(*) VAR

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

FILE

Routine arg. Input CHAR(*) VAR

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

/DISMOUNT

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

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

Function

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

CONVERT BASE

CONVERT BASE base file size

PURPOSE Convert data base.

PARAMETERS

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

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

Caller MDBCOPY,MUTIL

Author H.G.Essel, B.Dechant

Example

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

Remarks

File name GOO$DM:M$ACOAL.PPL

Created by GOO$DM:M$COBCM.PPL

Description

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

COMMAND CONVERT BASE base file size Argument /parameter description:

BASE

Routine arg. Input CHAR(*) VAR

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

FILE

Routine arg. Input CHAR(*) VAR

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

FILE

Routine arg. Input BIN FIXED(31)

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

Function

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

COPY BASE

```
COPY BASE base file size area
                  /DISMOUNT
```

PURPOSE Expand and copy data base.

PARAMETERS

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

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

size integer
Optional size of new base in Kbytes.

area integer
Optional new number of entries in area directory.

/DISMOUNT switch default:
Dismount data base after copy.

Caller MDBCOPY,MUTIL

Author H.G.Essel

Example

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

Remarks

File name GOO$DM:M$ACODB.PPL

Created by GOO$DM:M$COBCM.PPL

Description

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

COMMAND COPY BASE base file size area /DISMOUNT

Argument /parameter description:

BASE

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”

Name of the data base to be expanded and copied.

FILE

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”

File name of output file. Default file type is .SEC.

AREA

Routine arg. Input BIN FIXED(31)

Command par. integer

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

SIZE

Routine arg. Input BIN FIXED(31)

Command par. integer

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

/DISMOUNT

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: The source data base is dismounted after the copy.

Function

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

COPY CONDITION

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

PURPOSE	Copy source Condition to destination Condition
PARAMETERS	
NAME	String replace default: ” Source Condition name, may be wildcarded.
DESTNAME	String replace default: ” Destination Condition name, may be wildcarded.
COND_DIR	String global replace default: '$CONDITION' Source Condition directory name, may be wildcarded.
ESTCOND_DIR	String replace default: '$CONDITION' Destination Condition directory name, may be wildcarded.
BASE	String global replace default: 'DB' Source Condition data base.
DESTBASE	String replace default: 'DB' Destination Condition data base.
NODE	String global replace default: 'E' Source Condition node name.
DESTNODE	String global replace default: 'E' Destination Condition node.

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

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

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

[NO] KEEP_MAP Switch default: /KEEP_MAP

Caller MDBM,MGOODBM

Author Th.KROLL

Example

EXAMPLE

```
COPY CONDITION E::DB:[$CONDITION]S1
                 E::NEWDB:[$CONDITION]NEW_S1/NOCONFIRM
```

Creates the condition NEW_S1 on Database NEWDB
and copies the data from condition S1 from
Database DB to condition NEW_S1.

```
COPY CONDITION E::DB:[$CONDITION]S(1)
                 E::NEWDB:[$CONDITION]NEW_S2/NOCONFIRM
```

Creates the Condition NEW_S2 on Database NEWDB
and copies the data from Condition S(1) from
Database DB to Condition NEW_S2.

```
COPY CONDITION E::DB:[$CONDITION]S(1:5)
                 E::NEWDB:[$CONDITION]NEW_S(3:7)/NOCO
```

Creates the conditionarray NEW_S(3:7) on Database
NEWDB and copies the data from conditionarray S
from Database DB to conditionarray NEW_S.

```
COPY CONDITION E::DB:[$CONDITION]S(*)
                 E::NEWDB:[$CONDITION]NEW_S(*)/NOCO
```

Creates the conditionarray NEW_S(*) on Database
NEWDB with the same limits as source Conditions
and copies the data from Conditionarray S from
Database DB to Conditionarray NEW_S.

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

E::NEWDB:[$CONDITION]NEW_S(*)/REPLACE

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

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

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

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

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

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

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

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

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

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

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

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

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

Remarks

File name E$COCO.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument / Parameter description.

NAME

Routine arg. Input CHAR(*) VAR

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

DESTNAME

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

DESTCOND_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

DESTBASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

DESTNODE

Routine arg. Input CHAR(*) VAR

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

/REPLACE

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

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

/CONFIRM

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

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

/LOG

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

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

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

Remarks

Module is an action routine.

Example

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

COPY ELEMENT

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

PURPOSE Copy source Dataelement to destination Dataelement

PARAMETERS

name required string global replace default: ” Source Dataelement name

destname required string global replace default: ” Destination Dataelement name

dir required string global replace default: ’DATA’ Default source directory

destdir required string global replace default: ’DATA’ Default destination directory

destpool required string global replace default: ’DATA’ Default destination pool

base required string global replace default: ’DB’ Default source database

destdb required string global replace default: ’DSTDB’ Default destination database

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

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

I_REPLACE switch default: ” Replace deastination dataelement

I_ALL switch default: ” Replace or copy a complete name array

I_NOCONFIRM switch default: ” Don’t ask to confirm creation of a new Dataelement

I_KEEP_MAP switch default: /KEEP_MAP Inhibit unmap of Data Base

Caller	MDBM,MGOODBM
Author	Th. Kroll

EXAMPLE

Following types of copy are supported :

```
 Dataelement EMIL and ADAM defined as scalar
 Dataelement ILSE and MARIE defined as name arrays
 Dataelement OTTO and EVA defined as
 queue name arrays
 Dataelement KLAUS and WILLI defined as
 queued member
   COPY ELEM DB:[dir]EMIL DB:[dir]ADAM
    The existing DE ADAM will be deleted and a
    new DE will be created and then the data will be
    copied.
   COPY EL DB:[dir]ILSE(n) DB:[dir]ADAM
    The existing DE ADAM will be deleted and a
    new DE will be created and then the data will be
    copied.
   COPY EL DB:[dir]ILSE DB:[dir]MARIE /REPLACE/ALL
     DE MARIE must exist, the data of MARIE will be
     overwritten.
   COPY EL DB:[dir]ILSE(n) DB:[dir]MARIE(m)/REPLACE
     DE MARIE must exist, member (m)
     will be replaced
   COPY EL DB:[dir]EMIL DB:[dir]MARIE(m)/REPLACE
     DE MARIE must exist, member(m) will be replaced
   COPY EL DB:[dir]ILSE(n:m) DB:[dir]MARIE(o:p)/REPLA
     DE MARIE must exist, members (o:p) will
     be replaced. The limits of the array must match.
   COPY EL DB:[dir]OTTO DB:[dir]EVA/REPLACE
     DE EVA must exist, Queue header will be repl.
   COPY EL DB:[dir]OTTO->type(i) DB:[dir]EVA->type(j)
         /REPLACE
     DE EVA must exist, queued name array member(j)
     will be replaced
   COPY EL DB:[dir]OTTO->type DB:[dir]EVA->type
         /ALL/REPLACE
     DE EVA must exist, complete queued DE wil be
     copied.
```

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

File name M$ACODE.PPL

Created by GOO$DM:M$DMCMD.PPL

Description

CALLING

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

COMMAND

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

rguments / Parameter description.

ELEMENT

Routine arg. Input CHAR(*) VAR

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

DEST_ELEMENT

Routine arg. Input CHAR(*) VAR

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

DIR

Routine arg. Input CHAR(*) VAR

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

DESTDIR

Routine arg. Input CHAR(*) VAR

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

DESTPOOL

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

DESTBASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

DESTNODE

Routine arg. Input CHAR(*) VAR

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

/REPLACE

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

/ALL

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

/NOCONFIRM

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

/KEEP_MAP

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

Function

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

COPY MEMBER

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

PURPOSE Copy Data Element member to another Data Element member

PARAMETERS

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

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

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

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

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

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

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

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

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

Caller MDBM,MGOODBM

Author Th. KROLL

Example

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

Remarks

File name M$ACOME.PPL

Created by GOO$DM:M$DMCMD.PPL

Description

CALLING

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

COMMAND

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

Argument / Parameter description.

MEMBER

Routine arg. Input CHAR(*) VAR

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

DESTMEMBER

Routine arg. Input CHAR(*) VAR

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

DIR

Routine arg. Input CHAR(*) VAR

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

DESTDIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

DESTBASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

DESTNODE

Routine arg. Input CHAR(*) VAR

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

/KEEP_MAP

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

Command par. sswitch default: '/KEEP_MAP'

/KEEP_MAP Inhibit unmap of Data Base

/NOKEEP_MAP Unmap of Data Base

Function

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

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

COPY Polygon

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

PURPOSE Copy source Polygon to destination Polygon

PARAMETERS

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

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

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

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

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

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

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

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

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

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

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

[NO] KEEP_MAP Switch default: /KEEP_MAP

Caller MDBM,MGOODBM

Author Th.KROLL

Example

EXAMPLE

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

```
                E::NEWDB:[$CONDITION]NEW_S(*)/REPLACE
  Replaces the conditionarray NEW_S(3:7) on Database
  NEWDB and copies the data from conditionarray S
  from Database DB to Conditionarray NEW_S.
COPY CONDITION E::DB:[$CONDITION]S(2)
                E::NEWDB:[$CONDITION]S_COPY(4)/NOCON
  Creates the Condition S_COPY(4) on Database NEWDB
  and copies the data from Condition S(2) from
  Database DB to Condition S_COPY(4).
COPY CONDITION E::DB:[$CONDITION]S(2)
               E::NEWDB:[$CONDITION]S_COPY(4)/REPLACE
  Copies the Condition S(2) from database DB to the
  existing Condition S_COPY(4) on destination
  Database NEWDB ( a replace is done).
COPY CONDITION E::DB:[$CONDITION]S(1:5)
               E::NEWDB:[$CONDITION]S_COPY(1:5)/REPLA
  Copies the Condition S(1:5) from database DB to the
  existing Condition S_COPY(1:5) on destination
  Database NEWDB ( a replace is done).
COPY CONDITION E::DB:[$CONDITION]S(1)
                 E::NEWDB:[$CONDITION]S_TEST /REPLACE
  Copies the Condition S(2) from database DB to the
  existing Condition S_TEST on destination
  Database NEWDB ( a replace is done).
COPY CONDITION E::DB:[$CONDITION]S_ORIGNAL
                E::NEWDB:[$CONDITION]S_COPY(1)/REPLACE
  Copies the Condition S_ORIGINAL from database DB to
  the existing Condition S_COPY(1) on destination
  Database NEWDB ( a replace is done).
COPY CONDITION E::DB:[$CONDITION]S_ORIGNAL
                E::NEWDB:[$CONDITION]S_COPY/REPLACE
  Copies the Condition S_ORIGINAL from database DB to
  the existing Condition S_COPY on destination
  Database NEWDB ( a replace is done).
```

Remarks

File name E$COPO.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument / Parameter description.

NAME

Routine arg. Input CHAR(*) VAR

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

DESTNAME

Routine arg. Input CHAR(*) VAR

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

POLY_DIR

Routine arg. Input CHAR(*) VAR

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

DESTPOLY_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

DESTBASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

DESTNODE

Routine arg. Input CHAR(*) VAR

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

/REPLACE

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

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

/CONFIRM

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

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

/LOG

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

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

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

Remarks

Module is an action routine.

Example

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

COPY SPECTRUM

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

PURPOSE Copy source Spectrum to destination Spectrum

PARAMETERS

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

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

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

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

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

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

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

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

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

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

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

[NO] KEEP_MAP Switch default: /KEEP_MAP

Caller MDBM,MGOODBM

Author Th.KROLL

Example

EXAMPLE

```
COPY SPECTRUM E::DB:[$SPECTRUM]S1
              E::NEWDB:[$SPECTRUM]NEW_S1/NOCONFIRM
```

Creates the spectrum NEW_S1 on Database NEWDB and copies the data from spectrum S1 from Database DB to Spectrum NEW_S1.

```
COPY SPECTRUM E::DB:[$SPECTRUM]S(1)
              E::NEWDB:[$SPECTRUM]NEW_S2/NOCONFIRM
```

Creates the spectrum NEW_S2 on Database NEWDB and copies the data from spectrum S(1) from Database DB to Spectrum NEW_S2.

```
COPY SPECTRUM E::DB:[$SPECTRUM]S(1:5)
              E::NEWDB:[$SPECTRUM]NEW_S(3:7)/NOCO
```

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

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

Creates the spectrumarray NEW_S(*) on Database NEWDB with the same limits as source Spectrum S and copies the data from Spectrumarray S from Database DB to Spectrumarray NEW_S.

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

```
  Replaces the spectrumarray NEW_S(3:7) on Database
  NEWDB and copies the data from spectrumarray S
  from Database DB to Spectrumarray NEW_S.
COPY SPECTRUM E::DB:[$SPECTRUM]S(2)
                 E::NEWDB:[$SPECTRUM]S_COPY(4)/NOCON
  Creates the spectrum S_COPY(4) on Database NEWDB
  and copies the data from spectrum S(2) from
  Database DB to Spectrum S_COPY(4).
COPY SPECTRUM E::DB:[$SPECTRUM]S(2)
                 E::NEWDB:[$SPECTRUM]S_COPY(4)/REPLACE
  Copies the spectrum S(2) from database DB to the
  existing Spectrum S_COPY(4) on destination
  Database NEWDB ( a replace is done).
COPY SPECTRUM E::DB:[$SPECTRUM]S(1:5)
                 E::NEWDB:[$SPECTRUM]S_COPY(1:5)/REPLA
  Copies the spectrum S(1:5) from database DB to the
  existing Spectrum S_COPY(1:5) on destination
  Database NEWDB ( a replace is done).
COPY SPECTRUM E::DB:[$SPECTRUM]S(1)
                  E::NEWDB:[$SPECTRUM]S_TEST /REPLACE
  Copies the spectrum S(2) from database DB to the
  existing Spectrum S_TEST on destination
  Database NEWDB ( a replace is done).
COPY SPECTRUM E::DB:[$SPECTRUM]S_ORIGNAL
                 E::NEWDB:[$SPECTRUM]S_COPY(1)/REPLACE
  Copies the spectrum S_ORIGINAL from database DB to
  the existing Spectrum S_COPY(1) on destination
  Database NEWDB ( a replace is done).
COPY SPECTRUM E::DB:[$SPECTRUM]S_ORIGNAL
                 E::NEWDB:[$SPECTRUM]S_COPY/REPLACE
  Copies the spectrum S_ORIGINAL from database DB to
  the existing Spectrum S_COPY on destination
  Database NEWDB ( a replace is done).
```

Remarks

File name E$COSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument / Parameter description.

NAME

Routine arg. Input CHAR(*) VAR

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

DESTNAME

Routine arg. Input CHAR(*) VAR

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

SPEC_DIR

Routine arg. Input CHAR(*) VAR

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

DESTSPEC_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

DESTBASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

DESTNODE

Routine arg. Input CHAR(*) VAR

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

/REPLACE

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

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

/CONFIRM

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

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

/LOG

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

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

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

Remarks

Module is an action routine.

Example

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

CREATE AREA

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

PURPOSE Create an Area in a Data Base

PARAMETERS

area Area name
required common default

base Data Base name
required common default

pool Pool name
required common default

areabytes Size in bytes
replace default:'100'

cluster Cluster size in bytes
replace default:'4'

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

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

Caller M$DMCMD

Author M. Richter

File name M$ACRAR.PPL

Dataset -

Remarks

REMARKS -

Description

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

ARGUMENTS

CV_AREA I Area name
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

CV_POOL I Pool name
CHAR(*) VAR

L_AREABYTES I Area size in bytes
BIN FIXED(31)

L_CLUSTER I Cluster size in bytes
BIN FIXED(31)

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

FUNCTION Create an Area in a Data Base

REMARKS Module is an action routine.

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

CREATE BASE

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

PURPOSE Create a new Data Base (section)

PARAMETERS

base Data Base name required common replaced default

basefile Data base file name required common replaced default

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

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

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

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

basepages Size in pages
replace default:'500'

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

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

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

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

Caller M$DMCMD

Author M. Richter

File name M$ACRDB.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING

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

ARGUMENTS

CV_BASE Data Base name
CHAR(*) VAR
Input

CV_BASEFILE Data Base file name
CHAR(*) VAR
Input

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

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

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

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

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

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

CV_PERMANENT Section permanence
CHAR(*) VAR
Input

CV_GLOBAL Section scope
CHAR(*) VAR
Input

FUNCTION Create a new Data Base (section)

REMARKS Module is an action routine.

EXAMPLE -

CREATE CALIBRATION FIXED

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

PURPOSE Create a Data Element for fixed calibration.

PARAMETERS

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

Remarks

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

Description

CALLING

```
STS=E$CRCFI(CV_name,L_entries,CV_cal_dir,
            CV_cal_pool,CV_base,CV_node,I_keep_map)
```

COMMAND

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

NAME

Routine arg. CHAR(*) VAR

Command par. String required

Name of calibration Data Element table which should be created.

ENTRIES

Routine arg. BIN FIXED(31)

Command par. Integer default=0

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

CAL_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$CALIB

Directory for calibration Data Element

CAL_POOL

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$CAL_POOL

Pool for calibration Data Element

BASE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default = DB

Default Data Base name

NODE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default = *

Default node name

KEEP_MAP

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

Command par. Switch negatable default=/KEEP_MAP

Switch to prevent the dettachment of the Data Base

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

Function

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

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

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

CREATE CALIBRATION FLOAT

```
CREATE CALIBRATION FLOAT name entries cal_dir
                         cal_pool base node
                         /KEEP_MAP
```

PURPOSE Create Data Element for float calibration.

PARAMETERS

name Name of calibration Data Element.

entries Number of uncalibrated/calibrated points.

cal_dir Directory for calibration

cal_pool Pool for calibration.

base Data Base name

node node name for Data Base file

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

Caller MDBM,MGOODBM

Author W.Spreng

Remarks

Created by GOO$DE:E$DECMD.PPL

File name GOO$DE:E$CRCFL.PPL

Description

CALLING

```
STS=E$CRCFL(CV_name,L_entries,CV_cal_dir,
            CV_cal_pool,CV_base,CV_node,I_keep_map)
```

COMMAND

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

NAME

Routine arg. CHAR(*) VAR

Command par. String required

Name of calibration Data Element table which should be created.

ENTRIES

Routine arg. BIN FIXED(31)

Command par. Integer default=0

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

CAL_DIR

Routine arg. CHAR(*) VAR

Command par. String global replacable default=$CALIB

Directory for calibration Data Element

CAL_POOL

Routine arg. CHAR(*) VAR

Command par. String global replacable default=$CAL_POOL

Pool for calibration Data Element

BASE

Routine arg. CHAR(*) VAR

Command par. String global replacable default = DB

Default Data Base name

NODE

Routine arg. CHAR(*) VAR

Command par. String global replacable default = *

Default node name

KEEP_MAP

Routine arg.	BIN FIXED(15) valid values 0 and 1
Command par.	Switch negatgable default=/KEEP_MAP Switch to prevent the dettachment of the Data Base

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

Function

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

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

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

CREATE CALIBRATION LINEAR

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

PURPOSE Create Data Element for linear calibration.

PARAMETERS

name Name of calibration Data Element.

cal_dir Directory for calibration

cal_pool Pool for calibration.

base Data Base name

node node name for Data Base file

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

Caller MDBM,MGOODBM

Author W.Spreng

Remarks

Created by GOO$DE:E$DECMD.PPL

File name GOO$DE:E$CRCLI.PPL

Description

CALLING

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

COMMAND

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

NAME

Routine arg. CHAR(*) VAR

Command par. String required

Name of calibration Data Element table which should be created.

CAL_DIR

Routine arg. CHAR(*) VAR

Command par. String global replacable default=$CALIB

Directory for calibration Data Element

CAL_POOL

Routine arg. CHAR(*) VAR

Command par. String global replacable default=$CAL_POOL

Pool for calibration Data Element

BASE

Routine arg. CHAR(*) VAR

Command par. String global replacable default = DB

Default Data Base name

NODE

Routine arg. CHAR(*) VAR

Command par. String global replacable default = *

Default node name

/KEEP_MAP

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

Command par. Switch negatgable default=/KEEP_MAP

Switch to prevent the dettachment of the Data Base

/NOKEEP_MAP Deattach Data Dase.

/KEEP_MAP	Keep the whole Data Base mapping context

Function

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

Linear calibrations contains the two parameter of a linear polynom.

CREATE CONDITION COMPOSED

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

PURPOSE create a composed condition

PARAMETERS

name required string replace default: ”
name of the condition

expression string replace default: ”
Boolean expression of conditions.

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

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

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

node string global replace default: ’E’
default node

/[NO] DYNAMIC switch default: /DYNAMIC’

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

CRE COND COMP CC (A&B)—C

Remarks

File name GOO$DE:E$ACRCC.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”

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

EXPRESSION

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”

Boolean expression of conditions

COND_DIR

Routine arg. Input CHAR(*) VAR

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

COND_POOL

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/DYNAMIC

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

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

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

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

/KEEP_MAP

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

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

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

MASK

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

Function

Create a composed condition.

CREATE CONDITION FUNCTION

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

PURPOSE create a function condition

PARAMETERS

name required string replace default: ”
name of the condition

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

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

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

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

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

CRE COND FUNCTION USER MYSHARE(CHECK)

Remarks

File name GOO$DE:E$ACRFC.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”

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

FUNCTION

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”

Specification of function to be called. The module must be linked in a sharable image. Both are specified by 'image(module)'.

COND_DIR

Routine arg. Input CHAR(*) VAR

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

COND_POOL

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/DYNAMIC

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

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

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

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

/KEEP_MAP

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

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

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

MASK

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

Function

Create a function condition.

CREATE CONDITION MULTIWINDOW

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

PURPOSE create a multiwindow condition

PARAMETERS

name required string replace default: ”
name of the condition

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

dimension integer default: 1
Internal dimension

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

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

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

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE COND MULTI BINS (23,64) 5
```

Remarks

File name GOO$DE:E$ACRMW.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

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

LIMITS

Routine arg. Input CHAR(*) VAR

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

DIMENSION

Routine arg. Input BIN FIXED(31)

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

COND_POOL

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/DYNAMIC

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

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

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

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

/KEEP_MAP

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

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

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

MASK

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

Function

Create a window condition.

CREATE CONDITION PATTERN

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

PURPOSE create a pattern condition

PARAMETERS

name required string replace default: ”
name of the condition

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

dimension integer default: 1
Internal dimension

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

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

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

node string global replace default: 'E'
default node

invert string default: '0'
inversion pattern

/[NO] DYNAMIC switch default: /DYNAMIC'

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

checkmode set default: /IDENT valid values are:

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

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

Remarks

File name GOO$DE:E$ACRPC.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

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

PATTERN

Routine arg. Input CHAR(*) VAR

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

DIMENSION

Routine arg. Input BIN FIXED(31)

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

COND_POOL

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

INVERT

Routine arg. Input CHAR(*) VAR

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

/DYNAMIC

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

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

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

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

CHECKMODE

Routine arg. Input CHAR(*) VAR

Command par. set default: /IDENT

/IDENT pattern and object must be identical

/EXCL all bits set in object must be set in pattern (like ANY exclusive additional bits set in object)

/ANY pattern and object must have at least one common bit set

/INCL all bits set in pattern must be set in object (like IDENT inclusive additional bits set in object).

/KEEP_MAP

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

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

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

MASK

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

Function

Create a pattern condition.

CREATE CONDITION POLYGON

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

PURPOSE create a polygon condition

PARAMETERS

name required string replace default: ”
name of the condition

polygon string replace default: ”
Name of polygon.

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

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

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

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

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

node string global replace default: 'E'
default node

/[NO] DYNAMIC switch default: /DYNAMIC'

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

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

Remarks

File name GOO$DE:E$ACRPL.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

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

POLYGON

Routine arg. Input CHAR(*) VAR

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

DIMENSION

Routine arg. Input BIN FIXED(31)

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

POLY_DIR

Routine arg. Input CHAR(*) VAR

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

COND_POOL

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/DYNAMIC

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

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

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

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

/KEEP_MAP

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

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

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

MASK

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

Function

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

CREATE CONDITION WINDOW

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

PURPOSE create a window condition

PARAMETERS

name required string replace default: ”
name of the condition

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

dimension integer default: 1
Internal dimension

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

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

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

node string global replace default: ’E’
default node

/[NO] DYNAMIC switch default: /DYNAMIC’

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

CRE COND WIND ENER (23,64)

Remarks

File name GOO$DE:E$ACRWC.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

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

COMMAND

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

Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

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

LIMITS

Routine arg. Input CHAR(*) VAR

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

DIMENSION

Routine arg. Input BIN FIXED(31)

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

COND_POOL

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/DYNAMIC

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

Command par. negatable switch default: /DYNAMIC

/DYNAMIC everything is changeable

/NODYNAMIC not everything is changeable

/DOCUMENT

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

Command par. switch default: /NODOCUMENT

/DOCUMENT documentation will be held

/NODOCUMENT documentation will not be held

/KEEP_MAP

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

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

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

MASK

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

Function

Create a window condition.

CREATE DIRECTORY

**CREATE DIRECTORY dir dedentries base
/[NO]KEEP_MAP**

PURPOSE	Create a Data Element Directory in a Data Base
PARAMETERS	
dir	Data Element Directory name required, common default
dedentries	Number of entries for Data Element Directory replace default:'100'
base	Data Base name required, common default
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
EXAMPLE	CREA DIR PARAM 300 DB create the Data Element Directory 'PARAM' in the Data Base 'DB' with space for 300 Data Elements.
Caller	M$DMCMD
Author	M. Richter
File name	M$ACRDI.PPL
Dataset	-

Remarks

REMARKS -

Description

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

ARGUMENTS

CV_DIR I Data Element Directory name
CHAR(*) VAR

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

CV_BASE I Data Base name
CHAR(*) VAR

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

FUNCTION Create a Data Element Directory in a Data Base

REMARKS Module is an action routine.

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

CREATE DYNAMIC ENTRY BITSPECTRUM

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

PURPOSE Create a spectrum dynamic list entry

PARAMETERS

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

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

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

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

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

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

par_dir string global replace default: 'DATA'
Default directory

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

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

Remarks

File name M$ACEBS.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING

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

COMMAND

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

Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

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

SPECTRUM

Routine arg. Input CHAR(*) VAR

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

PARAMETER

Routine arg. Input CHAR(*) VAR

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

INCREMENT

Routine arg. Input CHAR(*) VAR

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

CONDITION

Routine arg. Input CHAR(*) VAR

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

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

PAR_DIR

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

SPEC_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/UPDATE

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

Command par. switch default: "

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

/CHECK

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

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

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

Execution

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

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

Arrays

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

Examples

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

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

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

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC ENTRY COMPOSED

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

PURPOSE Create a composed condition dynamic list entry

PARAMETERS

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

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

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

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

CRE DYN EN COMPOSED dlist [d]all_ok /MASTER
[d]is the directory specification
CRE DYN ENTR COMP l [$CONDITION]CC

Remarks

File name M$ACECC.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING STS=M$ACECC(CV_dyn_list,CV_condition,
CV_dyn_dir,CV_cond_dir,CV_base,CV_node,
I_master,I_update,I_check,I_keep_map)

COMMAND CREATE DYNAMIC ENTRY COMPOSED dyn_list condition
dyn_dir cond_dir base node
/MASTER
/UPDATE
/[NO]CHECK
/[NO]KEEP_MAP
Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

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

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specifiaction of composed condition.
Directory is defaulted from COND_DIR.
No array supported.

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/MASTER

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

Command par. switch default: ”

/MASTER	Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated. Master condition result bits are checked any time, even if the condition was already executed. So, if the same master condition is in two dynamic lists, both lists are skipped, if the condition was false.

/UPDATE

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: ”
/UPDATE	The modification becomes active in an analysis immediately. If not specified, several modifications of the dynamic list can be made without touching a running analysis.

/CHECK

Routine arg.	Input BIN FIXED(15) valid values 0 or 1
Command par.	switch default: '/CHECK'
/CHECK	The dynamic list is checked for validity.
/NOCHECK	No check is done. If several entries are added in a command procedure, only the last command should use the /CHECK qualifier to save CPU time. The others should use the /NOCHECK qualifiers.

/KEEP_MAP

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

Function

Create a dynamic list entry for composed condition. A boolean expression of conditions is executed. The expression is specified in the condition data el.

Execution

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

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

CREATE DYNAMIC ENTRY FUNCTION

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

PURPOSE Create a function condition dynamic list entry

PARAMETERS

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

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

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

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

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

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

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN ENTRY FUNCTION dlist x$cond
                 MOD=privshar(x$cond)
                 PAR=([d]$event.z4.de(5),$event.z5)
  [d]is the directory specification
  X$COND must be in the form:
  ENTRY(BIT(1) ALIGNED,POINTER,POINTER)
  RETURNS(BIN FIXED(31))
  CRE DYN ENTR FUNC l c_cali MYSHR(CALI)
                    PAR=[DATA]EVENT.RAW
  CALI must be in the form:
  ENTRY(BIT(1) ALIGNED,POINTER)
  RETURNS(BIN FIXED(31))
  Call procedure CALI which is linked in sharable
   image MYSHR. Pass pointer to data element as second argument.
```

Remarks

File name M$ACEFC.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING STS=M$ACEFC(CV_dyn_list,CV_condition,CV_module,
CV_parameter,CV_dyn_dir,CV_par_dir,CV_cond_dir,

```
  CV_base,CV_node,
  I_master,I_update,I_check,I_keep_map)
```

COMMAND CREATE DYNAMIC ENTRY FUNCTION dyn_list condition module parameter dyn_dir par_dir cond_dir base node

```
   /MASTER
   /UPDATE
   /[NO]CHECK
   /[NO]KEEP_MAP
```

Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

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

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name of the function condition.

MODULE

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Specification of module in the form image(module), where image is the name of the sharable image in which the module must be linked. The first argument of the procedure must be a BIT(1) ALIGNED variable returning the condition value ('1'B for true, '0'B for false).

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
List of data element member name specifications separated by commas. The pointers to these data element members are passed as arguments to the procedure. Base and directory are defaulted by the values specified by BASE and PAR_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

PAR_DIR

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/MASTER

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

Command par. switch default: ”

/MASTER Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated. Master condition result bits are checked any time, even if the condition was already executed. So, if the same master condition is in two dynamic lists, both lists are skipped, if the condition was false.

/UPDATE

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

Command par. switch default: ”

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

/CHECK

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

Command par. switch default: ’/CHECK’

/CHECK The dynamic list is checked for validity.

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

/KEEP_MAP

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

Command par. switch default: ’/KEEP_MAP’

/NOKEEP_MAP The data base is detached after the command.

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

Function

Create a function condition entry.

Execution

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

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

CREATE DYNAMIC ENTRY INDEXEDSPECTRUM

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

PURPOSE Create an indexed spectrum dynamic list entry

PARAMETERS

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

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

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

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

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

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

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

par_dir string global replace default: ’DATA’
Default directory

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN EN INDEX dlist [d]ener(1:10)
            PAR=[d]$event.e
            IND=[d]$event.i
CRE DYN EN INDEX dlist [d]ede(1:5)
            PAR=([d]$event.e(1:5),$event.de(1:5))
            IND=[d]$event.i
[d]is the directory specification
```

Remarks

File name M$ACEIS.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING

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

COMMAND

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

Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

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

SPECTRUM

Routine arg. Input CHAR(*) VAR

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

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications separated by commas. As may as spectrum dimension. Directory is defaulted by PAR_DIR.

INDEX

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Data element member specification or multiwindow used for index. Directory is defaulted by PAR_DIR. Therefore a multi window must be specified with directory.

INCREMENT

Routine arg. Input CHAR(*) VAR

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

CONDITION

Routine arg. Input CHAR(*) VAR

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

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

PAR_DIR

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

SPEC_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/UPDATE

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

Command par. switch default: ''

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

/CHECK

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

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

Create a dynamic list entry for indexed spectrum accumulation. Supports spectra of type BIN FIXED(31) or BIN FLOAT(24) with up to 2 dimensions. Coordinates can be BIN FIXED(31) or BIN FLOAT(24). Specified spectrum must be an array.

Execution

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

1. **Master procedures (PROCEDURE /MASTER)**
2. **Master pattern conditions (PATTERN /MASTER)**
3. **Master window conditions (WINDOW /MASTER)**
4. **Master function conditions (FUNCTION /MASTER)**
5. **Master polygon conditions (POLYGON /MASTER)**
6. **Master composed conditions (COMPOSED /MASTER)**
7. **Procedures (PROCEDURE)**
8. **Pattern conditions (PATTERN)**
9. **Window conditions (WINDOW)**
10. **Multi Window conditions (MULTI)**
11. **Function conditions (FUNCTION)**

12.Polygon conditions (POLYGON)

13.Composed conditions (COMPOSED)

14.Spectrum accumulation indexed (INDEXED)

15.Spectrum accumulation (SPECTRUM)

16.Bit spectrum accumulation (BITSPECTRUM)

17.Scatter plots (SCATTER)

Arrays

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

Examples

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

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

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

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC ENTRY MULTIWINDOW

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

PURPOSE Create a multiwindow condition dynamic list entry

PARAMETERS

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

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

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

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

par_dir string global replace default: 'DATA'
Default directory

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

CRE DYN EN MULTI dlist [d]e_recoil
PAR=[d]$event.ener
[d]is the directory specification
CRE DYN ENTR MULTI l [$CONDITION]M PAR=[DATA]EVT.X

Remarks

File name M$ACEMW.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING STS=M$ACEMW(CV_dyn_list,CV_condition,
CV_parameter,CV_dyn_dir,CV_par_dir,CV_cond_dir,
CV_base,CV_node,
I_update,I_check,I_keep_map)

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

DYN_LIST

Routine arg. Input CHAR(*) VAR

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

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of the multiwindow condition.
Directory is defaulted from COND_DIR.

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications
separated by commas. The number of elements must match the number of subwindows. The directory is defaulted from PAR_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

PAR_DIR

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/UPDATE

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

Command par. switch default: ”

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

/CHECK

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

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

Create a multiwindow condition dynamic list entry. Check specified member versus all window limits. For each check a result bit is set, which may be used to increment a spectrum array member. In addition, the number of the last matching window may be used as index of a spectrum array member (see INDEXED).

Execution

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

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

Arrays

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

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

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC ENTRY PATTERN

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

PURPOSE Create a pattern condition dynamic list entry

PARAMETERS

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

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

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

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

par_dir string global replace default: 'DATA'
Default directory

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN EN PATTERN dlist [d]main_pat
              PAR=[d]$event.pat
              /MASTER
 [d]is the directory specification
 CRE DYN ENTR PAT l [$CONDITION]P PAR=[DATA]EVENT.B
```

Remarks

File name M$ACEPC.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING

```
STS=M$ACEPC(CV_dyn_list,CV_condition,CV_parameter,
  CV_dyn_dir,CV_par_dir,CV_cond_dir,
  CV_base,CV_node,
  I_master,I_update,I_check,I_keep_map)
```

COMMAND

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

Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

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

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of a pattern condition. The directory name is defaulted from COND_DIR.

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications separated by commas. The number of elements must match the internal dimensionality. The members must be BIT(16) ALIGNED or BIT(32) ALIGNED. The directory is defaulted from PAR_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

PAR_DIR

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/MASTER

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

Command par. switch default: ”

/MASTER Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated. Master condition result bits are checked any time, even if the condition was already executed. So, if the same master condition is in two dynamic lists, both lists are skipped, if the condition was false.

/UPDATE

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

Command par. switch default: ”

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

/CHECK

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

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

Create a pattern condition entry. Check specified members versus patterns. Note that four test modes can be specified with the pattern condition DE (ALL, ANY, MATCH, EXCL). The values of the members can be inverted bitwise. Up to 8 internal dimensions.

Execution

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

1.Master procedures (PROCEDURE /MASTER)

2.Master pattern conditions (PATTERN /MASTER)

3.Master window conditions (WINDOW /MASTER)

4.Master function conditions (FUNCTION /MASTER)

5.Master polygon conditions (POLYGON /MASTER)

6.Master composed conditions (COMPOSED /MASTER)

7.Procedures (PROCEDURE)

8.Pattern conditions (PATTERN)

9.Window conditions (WINDOW)

10.Multi Window conditions (MULTI)

11.Function conditions (FUNCTION)

12.Polygon conditions (POLYGON)

13.Composed conditions (COMPOSED)

14.Spectrum accumulation indexed (INDEXED)

15.Spectrum accumulation (SPECTRUM)

16.Bit spectrum accumulation (BITSPECTRUM)

17.Scatter plots (SCATTER)

Arrays

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

Examples

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

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

```
  CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
```

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

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC ENTRY POLYGON

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

PURPOSE Create a polygon condition dynamic list entry

PARAMETERS

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

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

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

polygon string replace default: ”
Name of a polygon.

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

par_dir string global replace default: 'DATA'
Default directory

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

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN ENTR POLY 1 [$CONDITION]poco
        PAR=([DATA]EVENT.RAW(1),[DATA]EVENT.RAW(2))
```

Remarks

File name M$ACEPL.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING

```
STS=M$ACEPL(CV_dyn_list,CV_condition,
  CV_parameter,CV_polygon,
  CV_dyn_dir,CV_par_dir,CV_cond_dir,CV_poly_dir,
  CV_base,CV_node,
  I_master,I_update,I_check,I_keep_map)
```

COMMAND

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

/[NO]KEEP_MAP
Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

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

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
name specification of polygon condition. Directory is defaulted from COND_DIR.

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
List of two data element member names separated by commas. Directory is defaulted from PAR_DIR.

POLYGON

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”
Optional specification of a polygon. Normally the polygon is specified in the polygon condition. Directory is defaulted from POLY_DIR.

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

PAR_DIR

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

POLY_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/MASTER

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

Command par. switch default: ''

/MASTER Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated. Master condition result bits are checked any time, even if the condition was already executed. So, if the same master condition is in two dynamic lists, both lists are skipped, if the condition was false.

/UPDATE

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

Command par. switch default: ”

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

/CHECK

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

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

Create a dynamic list entry for polygon condition.

Execution

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

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

CREATE DYNAMIC ENTRY PROCEDURE

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

PURPOSE Create a procedure call dynamic list entry

PARAMETERS

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

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

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

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

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

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

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

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

[d]is the directory specification
X$LOOP must be in the form:
ENTRY(POINTER,POINTER) RETURNS(BIN FIXED(31))

```
CRE DYN ENTR PROC l MYSHR(CALI) PAR=[DATA]EVENT.RAW
```

Call procedure CALI which is linked in sharable image MYSHR. Pass pointer to data element as argument.
CALI must be in the form:
ENTRY(POINTER) RETURNS(BIN FIXED(31))

Remarks

File name M$ACEPR.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING

```
STS=M$ACEPR(CV_dyn_list,CV_module,
  CV_parameter,CV_condition,
  CV_dyn_dir,CV_par_dir,CV_cond_dir,
  CV_base,CV_node,
  I_master,I_update,I_check,I_keep_map)
```

COMMAND

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

Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”

Dynamic list name specification. Node, base and directory are defaulted from the according parameters NODE, BASE and DYN_DIR.

MODULE

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”

Specification of module in the form image(module), where image is the name of the sharable image in which the module must be linked.

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. string replace default: ”

List of data element member name specifications separated by commas. The pointers to these data element members are passed as arguments to the procedure. If not specified, base and directory are defaulted from PAR_DIR and BASE.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. string default: ”
If specified, the procedure is called only if the condition was true. The condition must be executed explicitely (either in a dynamic list or in the analysis program. If not specified, base and directory are defaulted from COND_DIR and BASE.

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

PAR_DIR

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/MASTER

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

Command par. switch default: ”

/MASTER Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated.

/UPDATE

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

Command par. switch default: ”

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

/CHECK

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

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

Create a dynamic list entry for procedure call. Calls module from sharable image. Pointers to data elements specified in argument list are passed.

Execution

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

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

CREATE DYNAMIC ENTRY SCATTER

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

PURPOSE Create a scatter plot dynamic list entry

PARAMETERS

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

picture required string replace default: ”
Name specification of picture.

process required string replace default: ”
Name of display process.

condition string default: ”
Name of a condition. If specified the scatter points are sent only, if the condition was true.

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN ENTR SCATTER l [$PICTURE]p GN_SUSI___$DSP
```

Remarks

File name M$ACESC.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING

```
STS=M$ACESC(CV_dyn_list,CV_picture,
  CV_process,CV_condition,
  CV_dyn_dir,CV_base,CV_node,
  I_update,I_check,I_keep_map)
```

COMMAND

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

Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

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

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
Name specification of picture.

PROCESS

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
name of display process.

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. string default: ”
If specified, the scatter points are sent only if
the condition was true. The condition must be executed explicitely (either in a dynamic list or in the analysis program.

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/UPDATE

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

Command par. switch default: ”

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

/CHECK

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

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

Create a dynamic list entry for scatter plot. This entry is created by the DISPL command. Several display processes may create several scatter plot entries.

CREATE DYNAMIC ENTRY SPECTRUM

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

PURPOSE Create a spectrum dynamic list entry

PARAMETERS

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

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

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

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

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

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

par_dir string global replace default: ’DATA’
Default directory

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN EN SPECTRUM dlist [d]ener(1:10)
            PAR=[d]$event.e(1:10)
CRE DYN EN SPECTRUM dlist [d]ede(1:5)
            PAR=([d]$event.e(1:5),$event.de(1:5))
[d]is the directory specification
```

Remarks

File name M$ACESP.PPL

Created by GOO$DM:M$DMCMD

Description

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

COMMAND

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

Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

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

SPECTRUM

Routine arg. Input CHAR(*) VAR

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

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications separated by commas. As may as spectrum dimension. Directory is defaulted by PAR_DIR.

INCREMENT

Routine arg. Input CHAR(*) VAR

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

CONDITION

Routine arg. Input CHAR(*) VAR

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

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

PAR_DIR

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

SPEC_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/UPDATE

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

Command par. switch default: ”

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

/CHECK

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

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

Create a dynamic list entry for spectrum accumulation. Supports spectra of type BIN FIXED(31) or BIN FLOAT(24) with up to 2 dimensions. Coordinates can be BIN FIXED(31) or BIN FLOAT(24). Specified spectrum can be an array.

Execution

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

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

Arrays

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

Examples

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

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

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

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC ENTRY WINDOW

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

PURPOSE Create a window condition dynamic list entry

PARAMETERS

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

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

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

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

par_dir string global replace default: 'DATA'
Default directory

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

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

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

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

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

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

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

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE DYN EN WINDOW dlist [d]e_recoil
              PAR=[d]$event.ener
  [d]is the directory specification
  CRE DYN ENTR WIN l [$CONDITION]W
                              PAR=[DATA]EVENT.RAW
```

Remarks

File name M$ACEWC.PPL

Created by GOO$DM:M$DMCMD

Description

CALLING

```
STS=M$ACEWC(CV_dyn_list,CV_condition,
  CV_parameter,CV_dyn_dir,CV_par_dir,CV_cond_dir,
  CV_base,CV_node,
  I_master,I_update,I_check,I_keep_map)
```

COMMAND

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

Argument / Parameter description.

DYN_LIST

Routine arg. Input CHAR(*) VAR

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

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
name specification of window condition. The
directory is defaulted from COND_DIR. An array must
be specified by [dir]name(ll:ul)

PARAMETER

Routine arg. Input CHAR(*) VAR

Command par. required string replace default: ”
List of data element member name specifications separated by commas. The number of elements must match the number of subwindows.

DYN_DIR

Routine arg. Input CHAR(*) VAR

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

PAR_DIR

Routine arg. Input CHAR(*) VAR

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

COND_DIR

Routine arg. Input CHAR(*) VAR

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

BASE

Routine arg. Input CHAR(*) VAR

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

NODE

Routine arg. Input CHAR(*) VAR

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

/MASTER

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

Command par. switch default: ''

/MASTER Valid for conditions and procedures. Master Functions are executed first of all other entries. Master conditions are executed first of all other conditions. If a master conditions result is false, the dynamic list execution is terminated. Master condition result bits are checked any time, even if the condition was already executed. So, if the same master condition is in two dynamic lists, both lists are skipped, if the condition was false.

/UPDATE

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

Command par. switch default: ''

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

/CHECK

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

Command par. switch default: '/CHECK'

/CHECK The dynamic list is checked for validity.

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

/KEEP_MAP

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

Command par. switch default: '/KEEP_MAP'

/NOKEEP_MAP The data base is detached after the command.

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

Function

Create a dynamic list entry for window condition. Check specified members versus window limits. Up to 8 internal dimensions.

Execution

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

1.**Master procedures (PROCEDURE /MASTER)**

2.**Master pattern conditions (PATTERN /MASTER)**

3.**Master window conditions (WINDOW /MASTER)**

4.**Master function conditions (FUNCTION /MASTER)**

5.**Master polygon conditions (POLYGON /MASTER)**

6.**Master composed conditions (COMPOSED /MASTER)**

7.**Procedures (PROCEDURE)**

8.Pattern conditions (PATTERN)

9.Window conditions (WINDOW)

10.Multi Window conditions (MULTI)

11.Function conditions (FUNCTION)

12.Polygon conditions (POLYGON)

13.Composed conditions (COMPOSED)

14.Spectrum accumulation indexed (INDEXED)

15.Spectrum accumulation (SPECTRUM)

16.Bit spectrum accumulation (BITSPECTRUM)

17.Scatter plots (SCATTER)

Arrays

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

Examples

```
CRE DYN EN WINDOW dlist [d]e_recoil(1:5)
              PARA=[d]$event.ener
 CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
              PARA=[d]$event.e(2:4) CONDI=[d]de_window
 CRE DYN EN SPECTRUM dlist [d]ede(1:4)
              PARA=([d]$event.e,$event.de)
 CRE DYN EN INDEXED dlist [d]ede(1:7)
              PARA=([d]$event.e,$event.de)
              INDEX=[d]a.b(1)
```

The difference between windows and multiwindows is that multiwindows have only one object for all * subwindows, but one result bit for each, whereas windows need one object per subwindow, but has only one result bit (set, if all subwindows are true). Multi windows may be used as filters for spectrum array accumulation. The internal dimension of the window must match the specified index range. It may also be used for indexed spectrum accumulation. Then the index of the matching subwindow is used to select the spectrum member. In the first case, the subwindows may overlapp, in the second case this makes normally no sense.

```
 CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
```

```
            PARA=[d]$event.e(2:4) CONDI=[d]m_window
CRE DYN EN INDEXED dlist [d]ener(2:4)
            PARA=[d]$event.e(1) INDEX=[d]m_window
```

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE DYNAMIC LIST

```
CREATE DYNAMIC LIST dyn_list entries dyn_dir pool
                        buffer base node
    /[NO]KEEP_MAP
```

PURPOSE Create a dynamic list in a Data Base

PARAMETERS

dyn_list Dynamic list name specification
required common replaced

entries Number of entries
replaced default:'100'

dyn_dir Default Directory
replaced common default:'$DYNAMIC'

pool Pool name
replaced default:'$DYNAMIC'

buffer Buffer size in bytes
replaced default:'0' (means 80*entries)

base Default Data Base name
replaced common default:'DB'

node Default node name
replaced common default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author H.G. Essel

File name M$ACRDL.PPL

Dataset -

EXAMPLE CRE DYN LIST L1 200
create dynamic list L1 in Pool $DYNAMIC with 200 entries

Remarks

REMARKS -

Description

CALLING

```
STS=M$ACRDL(CV_DYN_LIST,L_ENTRIES,CV_DYN_DIR
          ,CV_POOL,L_BUFFER,CV_BASE,CV_NODE,I_KEEP_MAP)
```

ARGUMENTS

CV_DYN_LIST I Dynamic list name specification

L_ENTRIES Number of entries

CV_DYN_DIR Default Directory

CV_POOL Pool name

L_BUFFER Buffer size in bytes. 80 bytes/entry should be adequate and is used as default, if L_BUFFER is 0.

CV_BASE I Default Data Base name

CV_NODE I Default node name

I_KEEP_MAP I 1= Inhibit unmap of Data Base

REMARKS Module is an action routine.

FUNCTION Create a dynamic list in a Data Base. Each action in the dynamic list takes one entry slot except scatter plot entries. These need one more slot for each condition specified in the picture. Information under following key words:

Related_commands

CREATE DYNAMIC LIST listname

CREATE DYNAMIC ENTRY PROCEDURE listname entry

CREATE DYNAMIC ENTRY PATTERN listname entry

CREATE DYNAMIC ENTRY WINDOW listname entry

CREATE DYNAMIC ENTRY FUNCTION listname entry

CREATE DYNAMIC ENTRY COMPOSED listname entry

CREATE DYNAMIC ENTRY MULTIWINDOW listname entry

CREATE DYNAMIC ENTRY POLYGON listname entry

CREATE DYNAMIC ENTRY SPECTRUM listname entry

CREATE DYNAMIC ENTRY INDEXED listname entry

CREATE DYNAMIC ENTRY BIT listname entry

CREATE DYNAMIC ENTRY SCATTER istname entry

DELETE DYNAMIC ENTRY listname type entry

SHOW DYNAMIC LIST listname type

ATTACH DYNAMIC LIST listname (Analysis only)

DETACH DYNAMIC LIST listname (Analysis only)

SHO DYNAMIC ATTACHED listname type (Analysis only)

SET DYNAMIC LIST Listname type key value (Analysis only)

STOP DYNAMIC LIST Listname type (Analysis only)

START DYNAMIC LIST Listname type (Analysis only)

listname Data element name of dynamic list

type Type of dynamic list entry:
PROCEDURE,FUNCTION,PATTERN,MULTI,WINDOW
POLYGON,COMPOSED,SPECTRUM,INDEXED,BIT,SCATTER

entry Name of dynamic list entry = name of data element

Execution

Note that for conditions, spectra and picture frames freeze bits may be set/cleared by commands. This disables/enables the execution of individual entries without modifications of the dynamic list itself. he order of execution is:

1. **Master procedures (PROCEDURE /MASTER)**
2. **Master pattern conditions (PATTERN /MASTER)**
3. **Master window conditions (WINDOW /MASTER)**
4. **Master function conditions (FUNCTION /MASTER)**
5. **Master polygon conditions (POLYGON /MASTER)**
6. **Master composed conditions (COMPOSED /MASTER)**
7. **Procedures (PROCEDURE)**
8. **Pattern conditions (PATTERN)**
9. **Window conditions (WINDOW)**
10. **Multi Window conditions (MULTI)**
11. **Function conditions (FUNCTION)**
12. **Polygon conditions (POLYGON)**
13. **Composed conditions (COMPOSED)**
14. **Spectrum accumulation indexed (INDEXED)**
15. **Spectrum accumulation (SPECTRUM)**
16. **Bit spectrum accumulation (BITSPECTRUM)**
17. **Scatter plots (SCATTER)**

Arrays

Spectra or conditions may be arrays. In this case an index range must be specified. All additional data elements must be either scalar or indexed by the same range. Ranges are specified by (l:u).

Examples

```
CRE DYN EN WINDOW dlist [d]e_recoil(1:5)
              PARA=[d]$event.ener
  CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
               PARA=[d]$event.e(2:4) CONDI=[d]de_window
  CRE DYN EN SPECTRUM dlist [d]ede(1:4)
               PARA=([d]$event.e,$event.de)
  CRE DYN EN INDEXED dlist [d]ede(1:7)
               PARA=([d]$event.e,$event.de)
               INDEX=[d]a.b(1)
```

The difference between windows and multiwindows is that multiwindows have only one object for all * subwindows, but one result bit for each, whereas windows need one object per subwindow, but has only one result bit (set, if all subwindows are true). Multi windows may be used as filters for spectrum array accumulation. The internal dimension of the window must match the specified index range. It may also be used for indexed spectrum accumulation. Then the index of the matching subwindow is used to select the spectrum member. In the first case, the subwindows may overlapp, in the second case this makes normally no sense.

```
  CRE DYN EN SPECTRUM dlist [d]ener1(2:4)
               PARA=[d]$event.e(2:4) CONDI=[d]m_window
  CRE DYN EN INDEXED dlist [d]ener(2:4)
               PARA=[d]$event.e(1) INDEX=[d]m_window
```

[d]is the directory specification

In both cases 'm_window' must have 3 subwindows.

CREATE ELEMENT

```
CREATE ELEMENT name pool typename refer datalength
                    cluster queuehead dir base node
    /[NO]PROTECT
    /[NO]REPLACE
    /[NO]KEEP_MAP
```

PURPOSE Create a Data Element in a Data Base

PARAMETERS

name	Data Element name (with index) required common replaced default
pool	Pool name required common replaced default
typename	Type name required
refer	List of refer values optional
datalength	Size in bytes replaced default:'0'
cluster	Cluster size in bytes replaced default:'16'
queuehead	Queue header specification optional
dir	Default Directory common replaced default:'DATA'
base	Default Data Base name common replaced default:'DB'

node Default node name
common replaced default:'E'

/[NO] PROTECT Protect deletion of new Data Element
default:'/NOPROTECT'

/[NO] REPLACE Replace old Data Element
default:'/NOREPLACE'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CREA ELE [PARAM]CALIB(1:10) PARA PARTYP 256 CLUST=16
create the Data Element name array CALIB with 10
members in Directory PARAM. The data of 256 bytes each will be in the Pool PARA. The Data type is PARTYP. PARAM, PARA, and PARTYP must exist already. The cluster size is 16 bytes.

Caller M$DMCMD

Author M. Richter

File name M$ACRDE.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ACRDE(CV_NAME,CV_POOL,CV_TYPENAME,
LA_REFER,I_DATALENGTH,L_CLUSTER,CV_QUEUEHEAD,
CV_DIR,CV_BASE,CV_NODE,I_PROTECT,
I_REPLACE,I_KEEP_MAP)

ARGUMENTS

CV_NAME I Data Element name (with index)
CHAR(*) VAR

CV_POOL I Pool name
CHAR(*) VAR

CV_TYPENAME I Type name
CHAR(*) VAR

LA_REFER I Refer values
(*) BIN FIXED(31)

I_DATALENGTH I Size in bytes
BIN FIXED(15)

L_CLUSTER I Cluster size in bytes
BIN FIXED(31)

CV_QUEUEHEAD I Queue header specification
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_PROTECT I Protect deletion of new Data Element
BIN FIXED(15)

I_REPLACE I Replace old Data Element
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Create a Data Element in a Data Base

REMARKS Module is an action routine.

CALLING

```
STS=M$ACRDE(CV_NAME,CV_POOL,CV_TYPENAME,
  LA_REFER,I_DATALENGTH,L_CLUSTER,CV_QUEUEHEAD,
  CV_DIR,CV_BASE,CV_NODE,I_PROTECT,
  I_REPLACE,I_KEEP_MAP)
```

EXAMPLE

```
STS$VALUE=M$ACRDE('[PARAM]CALIB(1:10)','PARA',
'PARTYP',0,256,16,”,”,”,”,0,0,1);
```

create the Data Element name array CALIB with 10 members in Directory PARAM. The data of 256 bytes each will be in the Pool PARA. The Data type is PARTYP. PARAM, PARA, and PARTYP must exist already. The cluster size is 16 bytes.

CREATE LINK

CREATE LINK link_from link_to dir base node
/MULTIPLE
/[NO]KEEP_MAP

PURPOSE Create a link between two Data Elements

PARAMETERS

link_from Source Data Element name specification
required, common default

link_to Target Data Element name specification
required, common default

dir Default Directory
required, common default

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/MULTIPLE Multiple identical links allowed

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CRE LIN [EVE]KAIN [ADAM]ABEL
create a link between the Data Element KAIN from Directory EVE and the Data Element ABEL from Directory ABEL, both in Data Base DB.

Caller M$DMCMD

Author M. Richter

File name M$ACRLI.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ACRLI(CV_LINK_FROM,CV_LINK_TO,
CV_DIR,CV_BASE,CV_NODE,I_MULTIPLE,I_KEEP_MAP)

ARGUMENTS

CV_LINK_FROM I Source Data Element name specification
CHAR(*) VAR

CV_LINK_TO I Target Data Element name specification
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_MULTIPLE I Multiple identical links allowed
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Create a link between two Data Elements

REMARKS Module is an action routine.

EXAMPLE -

CREATE OVERLAY

```
CREATE OVERLAY picture frame spectrum xparam yparam
trans=(xfactor,xoffset,yfactor,yoffset) dynshift node base pic_dir spec_dir
par_dir
   /[NO]KEEP_MAP
```

PURPOSE Add spectrum or scatterparameter to frame of a picture data element

PARAMETERS

picture	Name of the picture data element
frame	Number of picture frame for which an overlay should be created.
spectrum	Name of new spectrum, the spectrum could be one or two dimensional.
xparam	X-parameter for additional scatterplot parameters
yparam	Y-parameter for additional scatterplot parameters
xoffset	Offset X-direction for one dim. spectra
xfactor	Factor X-direction for one dim spectra
yoffset	Offset Y-direction for one dim spectra
yfactor	Factor Y-direction for one dim spectra
dynshift	Dynamic Y-offset
entries	Number of overlays which should be allocated for each frame.
node	Default node for Data Base
base	Default Data Base
pic_dir	Default Directory for pictures
spec_dir	Default Directory for spectra
par_dir	Default Directory for scatter parameter.

/[NO] KEEP_MAP	Inhibit the unmap of the whole Data Base.
Caller	MDBM,MGOODBM,E$DECMD
Author	W. Spreng

Example

1.) CREATE OVERLAY pic 2 spec DYNSHIFT=1.5
In picture "PIC" spectrum "SPEC" should be overlayed in frame 2 with a dynamic shift of 1.5.

2.) CREATE OVERLAY pic 1 xpara=par_x ypara=par_y

The scatterplot parameters par_x,par_y are additionally added to frame 1 in picture "PIC"

Remarks

File name	D$DEFOV.PPL
Created by	E$DECMD.ppl
REMARKS	If a dynamic y-shift is produced a dynamic y-scaling axis could be generated specifying the scaling factor in the MODIFY FRAME SPECTRUM command.

Description

CALLING

```
STS=D$DEFOV(CV_picture,L_frame,CV_spectrum,CV_xparam
            CV_yparam,RA_trans,R_dynshift,L_entries
            CV_node,CV_base,CV_pic_dir,CV_spec_dir,
            CV_par_dir,LA_dim,B_mask)
```

COMMAND

```
CREATE OVERLAY picture frame spectrum xparam yparam
trans=(xfactor,xoffset,yfactor,yoffset) dynshift entries node base
pic_dir spec_dir par_dir
    /[NO]KEEP_MAP
```

PICTURE

Routine arg.	Input CHAR(*) VAR

Command par. String required replaceable

Name specification for the picture for which an overlay should be defined.

FRAME

Routine arg. Input BIN FIXED(31)

Command par. Integer replaceable

Number of the frame for which the overlays should be defined.

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. String

Name of spectrum which should be added to the specified spectrum frame. The spectrum could be one or two dimensional.

XPARAM,YPARAM

Routine arg. Input CHAR(*) VAR

Command par. String

The X and Y-parameter for additional scatterplot correlations, which should be displayed in the specified scatter frame.

TRANS

Routine arg. Input (*) BIN FLOAT(24)

Command par. Real array default=(1.0,0.0,1.0,0.0)

Defines a linear transformation for overlayed spectra in the horizontal (x) and vertical (y) axis:
(xfactor,xoffset,yfactor,yoffset)
This transformation is applied to one and two dimensional spectra.

DYNSHIFT

Routine arg. Input BIN FLOAT(24)

Command par. Real

If several spectra should be overlayed and shifted into Y-direction it is useful to make this shift dynamicly, if the spectrum contents are changed by aquisition. This could be achieved specifing 'dynshift'. The offset is then determined as :
Y-offset = dynshift * (Maximum channel contents)

ENTRIES

Routine arg. Input BIN FIXED(31)

Command par. Integer default = 1

Specifies how many entries in the dataelement for overlays should be created for each frame. One overlay takes one slot.

This parameter can be used to prevent the replacement of the overlay dataelements, if not enough slots are availible to create a new overlay, because with each replacement free fragments are generated in your database which may not be used futheron. Therefore it is recommended to reduce the number of dataelement replacements by increasing this parameter.

The number of overlays is not limited by this parameter, you can generate more overlays as preliminary specified, if necessary the existing dataelement will be increased.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=*

Default node name

BASE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=DB

Default Data Base name

PIC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=$PICTURE

Default Picture Directory

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=$SPECTRUM

Default spectrum Directory

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=DATA

Default parameter Directory

/KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch negatgable default=/KEEP_MAP

Switch to prevent the dettachment of the Data Base

/NOKEEP_MAP	Deattach Data Dase.
/KEEP_MAP	Keep the whole Data Base mapping context

Function

Several overlays of spectra and/or scatterplot parameter can be defined for each frame of an existing picture. The information about the overlays is kept in additional Data Elements, which are different for spectra and scatterplots. If for 'picture' such an overlay Data Element is created for each frame 'entries' slots are reserved for overlayed spectra or scatterplots, depending on type of the specified frame.

It is obvious that the frame type (spectrum or scatter) must match the specified overlay. Futhermore it is imposssible to generate an overlay of a one and a two dimensional spectrum.

For the overlay of spectra a linear transformation in x- and y-direction can be defined with the help of the transformation parameter:

(xfactor,xoffset,yfactor,yoffset)

If the spectrum contents are changed by the acquisition, overlayed one dimensional spectra can be dynamicly shifted in y-direction, using the parameter "dynshift". The offset in Y-direction is then determined by:

Y-offset = R_dynshift * (Maximum channel contents)

To define a dynamic scaling axis specify the scaling factor in the MODIFY FRAME SPECTRUM command.

The created overlays are not displayed by default with the DISPLAY PICTURE command. To have a look at them give the OVERLAY command without specifying any parameter!

CREATE PICTURE

```
CREATE PICTURE name frames condition object
                   pic_dir pic_pool cond_dir par_dir
                   base node
    /[NO]PROMPT
    /[NO]KEEP_MAP
```

PURPOSE	Create a picture Data Element
PARAMETERS	
name	Name specification of the picture Data Element
frames	Number of frames in the picture
condition	Main condition for the picture
object	Parameter-list which should be checked
pic_dir	Default picture Directory
pic_pool	Default picture pool
cond_dir	Default condition Directory
par_dir	Default parameter Directory
base	Default Data Base
node	Default node name

/[NO] PROMPT Enter interactive prompting menu:

/NOPROMPT	no prompting
/PROMPT	prompting is performed (DEFAULT)

/[NO] KEEP_MAP Inhibit the unmap of the whole Data Base.

/NOKEEP_MAP	Data Base will be deattached

/KEEP_MAP Keep Data Base (DEFAULT)

Caller MDISP

Action rout. D$CRPI

Author W. Spreng

Example

1.) CREATE PICTURE a 4/noprompt
Picture "A" is created with 4-Frames and no prompting is performed.
2.) CREATE PICTURE b
Picture "B" with 1-Frame is created and the automatic prompting menue will be invoked.

Remarks

File name GOO$DISP:D$CRPI.PPL

Created by GOO$DISP:D$DSPCM.PPL

Description

CALLING

```
STS=D$CRPI(CV_pic_name,L_frames,CV_condition,
           CV_object CV_pic_dir,CV_pic_pool,
           CV_COND_DIR,cv_par_dir,CV_base,
           CV_node,CV_prompt,I_keep_map,B_mask)
```

COMMAND

```
CREATE PICTURE name frames condition object
               pic_dir pic_pool cond_dir par_dir
               base node
    /[NO]PROMPT
    /[NO]KEEP_MAP
```

NAME

Routine arg. CHAR(*) VAR

Command par. String required

Name of the picture which should be created.

FRAMES

Routine arg. BIN FIXED(31)

Command par. Integer default=1

Number of picture frames. Each frames contains one spectrum or one scatterplot. The maximum number of possible frames is 64.

CONDITION

Routine arg. CHAR(*) VAR

Command par. String

General picture condition for all scatterplots in this picture. The scatter data are show if the result flag of the condition is true.

OBJECT

Routine arg. CHAR(*) VAR

Command par. String

Condition object for the general picture condition. If this parameter is specified the condition is checked against the object. This will destroy the result of an earlier check of the same condition!

PIC_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$PICTURE

Default picture Directory.

PIC_POOL

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$PIC_POOL

Default picture Pool.

COND_DI

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$CONDITION

Default condition Directory.

PAR_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=DATA

Default parameter Directory.

BASE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

PROMPT

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch default = /PROMPT

If set an interactive prompting menue is envoked in which the parameter for each picture frame can be set.

Set /NOPROMPT to prevent this; e.g if pictures are created in command procedures!

KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch default = /KEEP_MAP

If set the mapping context will be kept. If /NOKEEP_MAP specified the Data Base will be dettached after the command completion.

Function

The procedure creates a picture Data Element in the Data Base with the specified number of frames.

The global picture condition is checked for each scatterplot defined in that picture. If the condition flag, which is set in the analysis or in the dynamic list, is true the scatterdata are displayed. If additionally a condition object is specified the global picture condition is checked against this object, but the result flag of an earlier condition check is no longer valid.

If /PROMPT is defined an interactive editing menu is envoked where for each frame all parameters are prompted. It is recommended to use this mode in an interactive session to be shure that all frames are completely specified! The editing menu has the advantage that the default display modes could be displayed.

If /NOPROMPT is specified (e.g. this is necessary for BATCH jobs) the frames has to be specified with the MODIFY FRAME command. In that case take care that all frames will be modified!

CREATE POLYGON

```
CREATE POLYGON polygon points
                    poly_dir poly_pool base node
     /[NO]KEEP_MAP
```

PURPOSE Create a polygon in a Data Base

PARAMETERS

polygon polygon name specification
required common replaced

points Number of points
replaced default:'1'

poly_dir Default Directory
replaced common default:'$POLYGON'

poly_pool Pool name
replaced default:'$PIC_POOL'

base Default Data Base name
replaced common default:'DB'

node Default node name
replaced common default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DECMD

Author H.G. Essel

File name E$ACRPO.PPL

Dataset -

EXAMPLE CRE POLY p1 20
create polygon p1 in Pool $PIC_POOL with 20 points.

Remarks

REMARKS -

Description

CALLING

```
STS=E$ACRPO(CV_POLYGON,L_POINTS,
  CV_POLY_DIR,CV_POOL,CV_BASE,
  CV_NODE,I_KEEP_MAP)
```

ARGUMENTS

CV_POLYGON Polygone name specification

L_POINTS Number of points

CV_POLY_DIR Default Directory

CV_POOL Pool name

CV_BASE Default Data Base name

CV_NODE Default node name

I_KEEP_MAP Inhibit unmap of Data Base

REMARKS Module is an action routine.

FUNCTION Create a polygon in a Data Base. The size of the polygon specified by L_POINTS is dynamically enlarged if necessary.

Related_commands

CREATE POLYGON polygon

SET POLYGON polygon

DELETE POLYGON polygon

SHOW POLYGON polygon

CREATE POOL

CREATE POOL pool areaminsize base
/[NO]KEEP_MAP

PURPOSE Create a Pool in a Data Base

PARAMETERS

pool Pool name
common default:'DATA'

areaminsize Minimum size of Areas in bytes
replace default:'512'

base data
required, common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CRE POOL ALPHA 10240 DB
create the Pool ALPHA in the Data Base DB. The Areas of this Pool will have a size of at least 10 kByte.

Caller M$DMCMD

Author M. Richter

File name M$ACRPO.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ACRPO(CV_POOL,L_AREAMINSIZE,CV_BASE,
I_KEEP_MAP)

ARGUMENTS

CV_POOL I Pool name
CHAR(*) VAR

L_AREAMINSIZE I Minimum size of Areas in bytes
BIN FIXED(31)

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Create a Pool in a Data Base

REMARKS Module is an action routine.

EXAMPLE -

CREATE SPECTRUM

```
CREATE SPECTRUM name type limits binsize
                    spec_dir spec_pool base node maxspec
                    branch crate station offset
  /CAMAC
  /[NO]DOCU
  /[NO]MEANVALUES
  /[NO]SQW
  /[NO]ERRV
  /[NO]ERRH
  /[NO]VARBINS
  /DYNAMIC/STATIC
  /ANALOG/DIGITAL
  /[NO]KEEP_MAP
```

PURPOSE create a spectrum

PARAMETERS

name name of spectrum
required

type data type of spectrum, may be S,H,I,L,R,D or N
replace default:'L'

limits limits of the spectrum
replace default:'(0,1023)'

binsize size of bins
replace default:'1'

spec_dir default Directory
replace default='$SPECTRUM'

spec_pool default pool
replace default='$SPEC_POOL'

base default Data Base
replace default='DB'

node default node
replace default='E'

maxspec maximum number of spectra
default=1024

branch Number of CAMAC branch
replace default=0

crate Number of CAMAC crate on branch
replace default=1

station Number of MR2000 in crate
replace default=2

offset Start address of data in MR2000
replace default=0

/[NO] DOCU if on documentation and history are held
replace default:'/NODOCU'

/[NO] MEANVALUES mean values of the bins are held
replace default:'/NOMEANVALUES'

/[NO] SQW if on sum of square weights per bin are held
replace default:'/NOSQW'

/[NO] ERRV if on vertical errors are held
replace default:'/NOERRV'

/[NO] ERRH if on horizontal errors are held
replace default:'/NOERRH'

/[NO] VARBINS if on spectrum has variable binsize
replace default:'/NOVARBINS'

/DYNAMIC/STATIC possibility of modifications of attributes
replace default:'/DYNAMIC'

/ANALOG/DIGITAL /ANALOG for floating point accumulation, /DIGITAL for accumulation of integers.

The difference is that in analog spectra a bin represents an intervall, but in digital spectra a number (or a range of numbers). Therefore digital

spectra have one more bin for the last number.
For analog spectra the upper limit is excluded from last bin. Digital spectra are displayed shifted by .5 to the left to center the numbers ticks.
replace default:'/ANALOG'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CREATE SPECTRUM emil H (0,2000,1,10) /NODOCU
a two-dimensional spectrum 'emil' is created
containing one-byte data in the limits x:0 to 2000, y:1 to 10 . No documentation will be held

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$CRESP.PPL

Dataset -

Remarks

REMARKS action routine

Description

CALLING

```
STS=E$CRESP(CV_SPEC,CV_TYPE,CV_LIMITS
      ,CV_BINSIZE,CV_DIR,CV_POOL,CV_BASE,CV_NODE
      ,L_TAB,I_BRANCH,I_CRATE,I_STATION,L_OFFSET
      ,I_CAMAC,CV_DOCU,CV_MEANVALUES,CV_SQW,CV_ERRV
      ,CV_ERRH,CV_VARBINS,CV_DYNAMICSTATIC,
      ,CV_ANALOGDIGITAL,I_KEEP_MAP,B_MASK)
```

ARGUMENTS

CV_SPEC name of spectrum
CHAR(*) VAR
Input

CV_TYPE data type of spectrum may be
N - spec contains no data
S - spec contains scatter data (1 bit per channel)
H - spec contains one byte integer data

I - two bytes integer data
L - four byte integer data
R - four byte floating point data
D - eight byte (double precision) float.pt. data
CHAR(*) VAR
Input

CV_LIMITS limits of the spectrum , limits given in pairs of two per dimension e.g. (0,10,1,100) defines a spectrum where x is between 0 and 10, and y betweeen 1 and 100
CHAR(*) VAR
Input

CV_BINSIZE size of the bin, may be a floating point number , default is 1.0
BIN FLOAT(16)
Input

CV_DIR default Directory
CHAR(*) VAR
Input

CV_POOL default pool
CHAR(*) VAR
Input

CV_BASE default Data Base
CHAR(*) VAR
Input

CV_NODE default node
CHAR(*) VAR
Input

L_TAB Maximum number of spectra.
BIN FIXED(31)
Input
Default=1024

I_BRANCH Number of CAMAC branch for CAMAC spectra.
BIN FIXED(15)
Input
Default=0

I_CRATE Number of CAMAC crate for CAMAC spectra.
BIN FIXED(15)

Input
Default=0

I_STATION Station of MR2000 in CAMAC crate for CAMAC spectra.
BIN FIXED(15)
Input
Default=0

L_OFFSET Start address of CAMAC spectrum in MR2000.
BIN FIXED(31)
Input
Default=0

I_CAMAC If 1, spectrum is CAMAC spectrum.
BIN FIXED(15)
Input
Default=0

CV_DOCU '/DOCU' if documentation and history is wanted
CHAR(*) VAR
Input

CV_MEANVALUES '/MEANVALUES' if mean values per bin are wanted
CHAR(*) VAR
Input

CV_SQW '/CV_SQW' if squares of weights are wanted to be hel
CHAR(*) VAR
Input

CV_ERRV '/ERRV' if vertical errors are wanted to be held
CHAR(*) VAR
Input

CV_ERRH '/ERRH' if horizontal errors are wanted to be held
CHAR(*) VAR
Input

CV_VARBINS '/VARBINS' if sjpectrum has variable bins ehich are then held with the spectrum
CHAR(*) VAR
Input

CV_DYNAMICSTATIC '/DYNAMIC' if all attributes of the spectrum ma be modified

'/STATIC' data type and numbers of dimensions can not be modified
CHAR(*) VAR
Input

CV_ANALOGDIGITAL '/ANALOG' if input is float.pt.number,
'/DIGITAL' if input is integer
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

FUNCTION A spectrum is created in the data management including all queued data elements . All structures are initialized according to the attributes given by the command.

REMARKS Action routine

EXAMPLE -

CREATE TABLE CONDITION

CREATE TABLE CONDITION name entries
directory pool base node
/[NO]KEEP

PURPOSE create condition bit table

PARAMETERS

name string replace default: 'ANCO'
name of the table

entries integer replace default: 1024
Number of entries (conditions)

directory string replace default: '$ANLTABS'
default directory (will be created).

pool string replace default: '$COND_POOL'
default pool

base string global replace default: 'DB'
default data base

node string global replace default: 'E'
default node

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE TAB COND ENTRIES=512
```

Remarks

File name GOO$DE:E$ACRCT.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$ACRCT(CV_name,L_entries,
        CV_directory,CV_pool,CV_base,CV_node,
        I_keep_map)
```

COMMAND

```
CREATE TABLE CONDITION name entries
  directory pool base node
  /[NO]KEEP
```

Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'ANCO'
Name specification of the table. This has the standard GOOSY format base:[directory]name. Base and directory are defaulted by the explicit parameters. The values here specified are not replaced as defaults!

ENTRIES

Routine arg. BIN FIXED(31)

Command par. integer replace default: 1024
This defines the maximum number of conditions which can be created. Each condition member of an array occupies one entry.

DIRECTORY

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$ANLTABS'
default directory

POOL

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$COND_POOL'
default pool

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
default base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
default node

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

/NOKEEP_MAP unmap Data Base In a procedure call this argument should be always 1 to prevent an unmap of the data base.

Function

Create a condition table. In this table the bits for freeze, execution, preset and result are allocated.

CREATE TABLE SPECTRUM

CREATE TABLE SPECTRUM name entries
directory pool base node
/[NO]KEEP

PURPOSE create spectrum bit table

PARAMETERS

name string replace default: 'ANSP'
name of the table

entries integer replace default: 1024
Number of entries (spectra)

directory string replace default: '$ANLTABS'
default directory (will be created).

pool string replace default: '$SPEC_POOL'
default pool

base string global replace default: 'DB'
default data base

node string global replace default: 'E'
default node

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller MDBM, MGOODBM

Author H.G.Essel

Example

```
CRE TAB SPEC ENTRIES=512
```

Remarks

File name GOO$DE:E$ACRST.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$ACRST(CV_name,L_entries,
        CV_directory,CV_pool,CV_base,CV_node,
        I_keep_map)
```

COMMAND

```
CREATE TABLE SPECTRUM name entries
  directory pool base node
  /[NO]KEEP
```

Argument /parameter description:

NAME

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'ANSP'
Name specification of the table. This has the standard GOOSY format base:[directory]name. Base and directory are defaulted by the explicit parameters. The values here specified are not replaced as defaults!

ENTRIES

Routine arg. BIN FIXED(31)

Command par. integer replace default: 1024
This defines the maximum number of spectra which can be created. Each spectrum member of an array occupies one entry.

DIRECTORY

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$ANLTABS'
default directory

POOL

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$SPEC_POOL'
default pool

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
default base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
default node

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP

/KEEP_MAP Do not unmap Data Base

/NOKEEP_MAP unmap Data Base In a procedure call this argument should be always 1 to prevent an unmap of the data base.

Function

Create spectrum table. In this table the bits for freeze, execution are allocated.

CREATE TYPE

CREATE TYPE typefilename base
/COMPILE/REFERFILE/COMPREF/NOCOMPNOREF
/[NO]KEEP_MAP

PURPOSE Create a Type descriptor from a PL/I structure.

PARAMETERS

typefilename Type descriptor name. A filename with a leading '@' (default file extension = .TXT) or a library module also with a leading '@', lib(module). This string must be given in double quotes "".
required

base Data Base name
required, common default

/COMPILE /REFERFILE /COMPREF /NOCOMPNOREF Test features:
/COMPILE : performs a test compilation of the Type structure
/REFERFILE : Creates a file with the REFER values of Type structure. The name will be identical to the Type structure name, but the first character will be replaced aby an 'R'.
/COMPREF : performs both compilation and REFER file creation.
/NOCOMPNOREF : performs neither a test compilation nor a creation of the REFER file.
default:'/NOCOMPNOREF'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE CRE TYPE "@GOOTYP(SM$DL)" DB
Create the Type SM$DL in Data Base from the library GOOTYP.
CRE TYPE DB "@PRIVAT"
Create the Type PRIVAT in Data Base from the

callers file PRIVAT.TXT.

Caller M$DMCMD

Author M. Richter

File name M$ACRTY.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ACRTY(CV_TYPEFILENAME,CV_BASE,CV_COMPREF,
I_KEEP_MAP)

ARGUMENTS

CV_TYPEFILENAME I Type descriptor file name (def ext = .TXT)
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

CV_COMPREF I Compile and/or write REFER file
/COMPILE : performs a test compilation of the
Type structure
/REFERFILE : Creates a file with the REFER values
of Type structure. The name will be identical to the Type structure
name, but the first character will be replaced aby an 'R'.
/COMPREF : performs both compilation and REFER file
creation.
/NOCOMPNOREF : performs neither a test compilation
nor a creation of the REFER file. This is the default.
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Create a Type descriptor

REMARKS Module is an action routine.

EXAMPLE -

DECALIBRATE SPECTRUM

DECALIBRATE SPECTRUM spectrum spec_dir base node

PURPOSE	Disconnect a spectrum from its calibration.
PARAMETERS	
spectrum	Name of spectrum
spec_dir	Directory for spectrum Data Element.
base	Data Base name
node	Node name for Data Base file
Author	W. Spreng
Caller	MDBM,MGOODBM

Remarks

Created by	GOO$DE:E$DECMD.PPL
File name	GOO$DE:E$DECAL.PPL

Description

CALLING	STS=E$DECAL(CV_spectrum,CV_spec_dir,CV_base,CV_node)
COMMAND	DECALIBRATE SPECTRUM spectrum spec_dir base node

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. String required

Name specification of the spectrum which should be disconnected from its calibration. No wildcards in the spectrum specification are supported.

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$SPECTRUM

Default spectrum Directory.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

Function

FUNCTION The specified spectrum is disconnected from any calibration. No wildcards in the spectrum specification are supported.

DECOMPRESS BASE

```
DECOMPRESS BASE file base basefile
                    /MOUNT
```

PURPOSE Restores compressed and copied data base. Command is executed by MUTIL.

PARAMETERS

file required string default: ”
Name of input file containing compressed base.

base required string default: ”
name of the data base to be restored.

basefile required string default: ”
name of data base file.

/MOUNT switch default:
Mount restored data base.

Caller MDBCOPY

Author H.G.Essel

Example

```
MDBCOPY DECOMP BASE db.cmp db db.sec
```

Remarks

File name GOO$DM:M$ADECB.PPL

Created by GOO$DM:M$DMCMD.PPL

Description

CALLING STS=M$ADECB(CV_file,CV_base,CV_basefile,I_mount)

COMMAND DECOMPRESS BASE file base basefile
/MOUNT
Argument /parameter description:

FILE

Routine arg. Input CHAR(*) VAR

Command par. required string default: ”
File name of input file containig compressed
data base. This file is generated by command
MDBCOPY COMPRESS BASE base basefile file
Default file type is .CSEC.

BASE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of the data base to be restored.

BASEFILE

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
File name of the data base.

/MOUNT

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default:
The restored data base is mounted.

Function

Decompress and copy a bata base. The output file of the data base must not exist. The data base to be restored must not be mounted. The base will be not mounted untill /MOUNT is specified.

DEFINE PICTURE SETUP

DEFINE PICTURE SETUP picture rows frames pic_dir base node /EQUAL

PURPOSE	Define frame organization on screen for one picture
PARAMETERS	
picture	Picture name specification
rows	Number of frame-rows on screen.
frames	Number of frame in each row.
pic_dir	Picture Directory
base	Data Base name
node	Node name
/EQUAL	The frames should be of equal size
Caller	MDBM, MGOODBM
Author	W. Spreng

Remarks

Created by	GOO$DISP:D$DSPCM.PPL
File name	GOO$DISP:D$PISUP.PPL

Examples

```
DEFINE PICTURE SETUP picture 3 3,6,2
Arrange frames in 3 rows:
                          1.row 3 frames.
                          2.row 6 frames.
```

3.row 2 frames.

The total number of frames in the picture must be 11. If the size of the frames should be equal specify:

DEFINE PICTURE SETUP picture 3 3,6,2 /EQUAL

Description

CALLING STS=D$PISUP(CV_PICTURE,L_ROWS,LA_FRAMES, CV_PIC_DIR,CV_BASE,CV_NODE,I_equal)

COMMAND DEFINE PICTURE SETUP picture rows frames pic_dir base node /EQUAL

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name of the picture for which the frame arrangement should be modified.

ROWS

Routine arg. Input BIN FIXED(31)

Command par. Integer required

Number of the rows in which the frames should be arranged. This number determines the vertical size of the frames. The number of rows can be any number in the range of 1 to the total number of frames in the picture.

FRAMES

Routine arg. Input (*) BIN FIXED(31)

Command par. Integer array required

Specifies the number of frames in each row. Therefore for each row one value has to be specified and the sum over all values has to be equal to the number of total frames in the picture.

E.g. if the frames should be arranged in 4 rows, 4 values are required, e.g. 3,4,4,1:

3 frames in row 1

4 frames in row 2
4 frames in row 3
1 frames in row 4
——————————
12 frames in total picture.

PIC_DIR

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=$PICTURE

Default picture Directory.

BASE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

/EQUAL

Routine arg. Input (*) BIN FIXED(15)

Command par. Switch

Normally the frames in each row are spread out over the total horizontal range of the display. Therfore the frames in different rows have different sizes if the amount of frames in the rows are different.

But it is possible to give all frames the same size. This can be done by /EQUAL switch. The size of the frames is determined by the row containing the most frames.

Function

With this command the default frame organization on the screen could be changed separately for each picture. This is possible every time after the creation of the picture.

The frames in the picture could be arranged in an arbitrary number of rows on the screen. This means the y-extents of all frames are identical. For each row the number of frames in that row has to be specified in the parameter 'frames'. The sum of all elements in 'frames' should be identical to the number of total frames in the picture.

Normally the frames in each row are spread out over the total horizontal range of the display. Therfore the frames in different rows have different sizes if the amount of frames in the rows are different.

But it is possible to give all frames the same size. This can be done specifying the /EQUAL switch. The size of the frames is determined by the row containing the most frames.

The frames are arranged row by row on the screen, from the left to the right.

DELETE CALIBRATION

```
DELETE CALIBRATION calibration node base cal_dir
                   /[NO]LOG
                   /[NO]KEEP_MAP
```

PURPOSE Delete a calibration Data Element.

PARAMETERS

calibration Calibration name specification

node Default node name

base Default Data Base

cal_dir Default Directory

/[NO] LOG Display picture name after deletion

/NOLOG nothing is displayed.

/LOG Names of deleted pictures are shown.

/[NO] KEEP_MAP Keep data base mapping context.

/NOKEEP_MAP Data Base will be dettached

/KEEP_MAP Keep Data Base (DEFAULT)

Caller MDBM,MGOODBM

Author W. Spreng

Example

1.) DELETE CALIBRATION a

Calibration "A" will be deleted.

2.) DELETE CALIBRATION [*]*test* /LOG

All calibrations containing the string "TEST" in the name are deleted and all calibration found are listed.

Remarks

File name GOO$DISP:E$DELCA.PPL

Created by GOO$DE:E$DECMD.PPL

REMARKS The specified calibrations could not be deleted if links to other Data Element exist. This is the case if one or several spectra are connected to this calibration.

Description

CALLING

```
STS = E$DELCA(CV_calib,CV_node,CV_db,CV_cal_dir,
              I_LOG,I_keep_map)
```

COMMAND

```
DELETE CALIBRATION calibration node base cal_dir
                /[NO]LOG
                /[NO]KEEP_MAP
```

CALIBRATION

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name of calibrations which should be deleted. Wildcards in calibration and directory name are allowed.

NODE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

BASE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

CAL_DIR

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=$CALIB

Default calibration Directory.

/LOG

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch default=/NOLOG

List the name of the pictures which has been deleted:

/NOLOG	nothing is listed.
/LOG	Names of deleted pictures are shown.

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch default=/KEEP_MAP

Switch to keep or to dettach the whole Data Base mapping context:

/NOKEEP_MAP	Data Base will be deattached
/KEEP_MAP	Keep Data Base (DEFAULT)

Function

The specified Calibration Data Element is deleted from the specified Data Base in the specified Directory on the section file on the specified Node.

Wildcards in the calibration name and in the directory name are supported.

DELETE CONDITION

DELETE CONDITION name cond_dir base node
/[NO]CONFIRM
/[NO]KEEP_MAP

PURPOSE delete a condition

PARAMETERS

name String replace default: none Name of condition

cond_dir String replace default: '$CONDITION' Name of condition directory

base String replace default: 'DB' Name of Data Base

node String replace default: '*' Name of Node

/[NO] CONFIRM Switch default : '/NOCONFIRM'
Noconfirm to delete condition

/[NO] KEEP_MAP Switch default: /KEEP_MAP Inhibit the unmap (detach) of the whole Data Base

EXAMPLE DEL CONDITION D::DB:[OTTO]S1

Caller MDBM

Author K.Winkelmann

File name GOO$DE:E$DLCO.PPL

Dataset -

Remarks

REMARKS -

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

STS=E$DLCO(CV_NAME,CV_DIR,CV_BASE,
CV_NODE,I_CONFIRM,I_KEEP_MAP)

Argument description

NAME

Type Input CHAR(*) VAR

Name of condition, may be fully qualified (or wildcarded (nyi))

DIR

Type Input CHAR(*) VAR

Default directory, taken if directory is not in the name specification

BASE

Type Input CHAR(*) VAR

Default base, is taken if base name is not in the name specification

NODE

Type Input CHAR(*) VAR

Default node ,...

CONFIRM

Type Input BIN FIXED(15)

0 Confirm to delete condition

1 Noconfirm to delete condition

KEEP_MAP

Type Input BIN FIXED(15)

Valid values are:

0 Unmap of Data Base

1	Inhibit unmap of Data Base

Function

The header element of a condition is searched. Then a queued data elements are deleted. Existing links to the analysis tables are deleted. Finally the header element is deleted. Wildcards for Condition names are supported.

Remarks

Module is an action routine.

Example

```
STS$VALUE=E$DLCO('OTTO','DATA','DB','/NOCONFIRM',
                  '/KEEP_MAP')
```

DELETE DYNAMIC ENTRY

```
DELETE DYNAMIC ENTRY dyn_type dyn_list
              namelist aux dyn_dir base node
    /UPDATE
    /[NO]KEEP_MAP
```

PURPOSE Delete a Dynamic List entry

PARAMETERS

dyn_type Entry type (* = all types)
common default:'SPECTRUM'

dyn_list Dynamic List name specification
required common default

namelist Name list: base:[dir]name,... or * for all entries
default:'*'

aux Used by scatter plot
default:'*'

dyn_dir Default directory for Dynamic List
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/UPDATE Update

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author H.G. Essel

File name M$ADLLE.PPL

Dataset -

EXAMPLE DEL DYN ENT SPEC DYNA1 * /UPD
delete all entries in the Dynaic List DYNA1 with the type SPECTRUM and update the list.

Remarks

REMARKS -

Description

CALLING

```
STS=M$ADLLE(CV_DYN_TYPE,CV_DYN_LIST,
  CV_NAMELIST,CV_AUX,CV_DYN_DIR,CV_BASE,
  CV_NODE,I_UPDATE,I_KEEP_MAP)
```

ARGUMENTS

CV_DYN_TYPE I Entry type
CHAR(*) VAR

CV_DYN_LIST I Dynamic List name specification
CHAR(*) VAR

CV_NAMELIST I Name list: base:[dir]name,...
CHAR(*) VAR

CV_AUX I Auxiliary string (used by scatter plot entries)
CHAR(*) VAR

CV_DYN_DIR I Default directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_UPDATE I Update
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION	Delete a Dynamic List entry
REMARKS	Module is an action routine.
EXAMPLE	-

DELETE DYNAMIC LIST

**DELETE DYNAMIC LIST dyn_list dyn_dir base node
/[NO]KEEP_MAP**

PURPOSE Delete a Dynamic List

PARAMETERS

dyn_list Dynamic List name specification
required common default

dyn_dir Default Directory for Dynamic List
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE DEL DYN LIST DYNA1
delete the Dynamic List 'DYNA1' in the last
Directory requested.

Caller M$DMCMD

Author H.G. Essel

File name M$ADLDL.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ADLDL(CV_DYN_LIST,CV_DYN_DIR,CV_BASE,
CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_DYN_LIST I Dynamic List name specification
CHARACTER(*) VAR

CV_DYN_DIR I Default Directory
CHARACTER(*) VAR

CV_BASE I Default Data Base name
CHARACTER(*) VAR

CV_NODE I Default node name
CHARACTER(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Delete a Dynamic List entry

REMARKS Module is an action routine.

EXAMPLE -

DELETE ELEMENT

DELETE ELEMENT name dir base node
/UNPROTECT
/[NO]KEEP_MAP

PURPOSE Delete a Data Element

PARAMETERS

name Node::base:[dir]name(i)->type(i)
The name array index might be a wild card with the * as all members. If name is a name array but i was not given then all members of the name array will be deleted (like (*)).
required common default

dir Default Directory
common default: 'DATA'

base Default Data Base name
common default: 'DB'

node Default node name
common default: 'E'

/UNPROTECT Overide deletion protection of Data Element

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ADLDE.PPL

Dataset -

EXAMPLE DEL DB:[DATA]ADAM
delete the Data Element 'ADAM' in the Directory 'DATA' of the Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ADLDE(CV_NAME,CV_DIR,CV_BASE,C_NODE,
I_UNPROTECT,I_KEEP_MAP)

ARGUMENTS

CV_NAME I Node::base:[dir]name(i)->type(i)
The name array index might be a wild card with the
* as all members. If name is a name array but i was not given then all members of the name array will be deleted (like (*)).
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_UNPROTECT I Override deletion protection of Data Element
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Delete a Data Element or a Data Element name array.

REMARKS Module is an action routine.
Since Data Element name arrays could only be
deleted completely, the name array index must be given as a wild card '*' or without any index.

EXAMPLE -

DELETE LINK

```
DELETE LINK link_from link_to dir base node
    /ALL
    /MATCH/IN/OUT
    /[NO]KEEP_MAP
```

PURPOSE Delete Data Element link(s)

PARAMETERS

link_from Target Data Element name specification
required common default

link_to Target Data Element name specification
required common default

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/ALL Delete all links selected by /SELECT

/MATCH/IN/OUT Matching, incomming, outgoing links
replaced default:'/MATCH'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ADLLI.PPL

Dataset -

EXAMPLE DEL LINK DB:[EVE]KAIN DB:[ADAM]ABEL
delete link between the Data Element 'KAIN' of the Directory 'EVE' and the Data Element 'ABEL' of the Directory 'ADAM', both in Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ADLLI(CV_LINK_FROM,CV_LINK_TO,
CV_DIR,CV_BASE,CV_NODE,I_ALL,CV_SELECT,I_KEEP_MAP)

ARGUMENTS

CV_LINK_FROM I Source Data Element name specification
CHAR(*) VAR

CV_LINK_TO I Target Data Element name specification
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_ALL I Delete all links selected by /SELECT
BIN FIXED(15)

CV_SELECT I Matching, incomming, outgoing links
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Delete Data Element link(s)

REMARKS Module is an action routine.

EXAMPLE -

DELETE OVERLAY

DELETE OVERLAY picture frame node base pic_dir
/[NO]KEEP_MAP

PURPOSE	Delete the defined overlayed spectra or scatterplot parameters for the specified frames.
PARAMETERS	
picture	Name of picture.
frame	Number or range of frames.
node	default node name.
base	Default Data Base name.
pic_dir	Default Picture Directory.
/[NO] KEEP_MAP	Inhibit the unmap of the whole Data Base.

/NOKEEP_MAP	Data Base will be deattached
/KEEP_MAP	Keep Data Base (DEFAULT)

Caller	MDBM,MGOODBM,E$DECMD
Author	W. Spreng

Remarks

File name	-
Created by	E$DECMD.PPL

Example

1.) DELETE OVERLAY a 1
The overlayed spectra/scatterplots for frame number 1 are deleted from picture 'A'.
2.) DELETE OVERLAY A 3:8
The overlays for frame 3 to 8 are deleted from picture 'A'.
3.) DELETE OVERLAY A *
The overlays for each frame are deleted from picture 'A'.

Description

CALLING STS=D$DDEOV(CV_picture,CV_frame,CV_node,CV_base,
CV_pic_dir)

COMMAND DELETE OVERLAY picture frame
node base pic_dir
/[NO]KEEP_MAP

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of picture for which the overlays should be deleted.

FRAME

Routine arg. Input CHAR(*) VAR

Command par. String required

Number of frame for which the overlayed spectra or additional scatterplot parameters should be deleted. Possible inputs are:
n - for a single frame
n:m - for a range of frames
* - for all frames
If the overlays for all frames should be deleted all Data Elements keeping information about overlays will be deleted from the Data Base. If a single frame or a range offrames has been specified the corresponding Data Elements for the overlays are marked unused, but the Data Element itself will not be changed or deleted!

NODE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=*

Default node name

BASE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=DB

Default Data Base name

PIC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=$PICTURE

Default picture directory

KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch default = /KEEP_MAP

If set the mapping context will be kept. If /NOKEEP_MAP specified the Data Base will be dettached after the command completion.

FUNCTION

The overlays for the specified picture will be deleted or marked unused depending on the specified frames.

If all frames in the picture are specified, the Data Elements keeping information about the overlays will be deleted. If a single frame or a range of frames is specified the corresponding overlays are marked unused, but the Data Element itself remains unchanged. Therefore no free areas are generated in that case!

DELETE PICTURE

```
DELETE PICTURE picture node base pic_dir
                  /[NO]LOG
                  /[NO]KEEP_MAP
```

PURPOSE Delete a Picture Data Element.

PARAMETERS

picture	Picture name specification
node	Default node name
base	Default Data Base
pic_dir	Default Directory
/[NO] LOG	Display picture name after deletion

/NOLOG	nothing is displayed.
/LOG	Names of deleted pictures are shown.

/[NO] KEEP_MAP Keep data base mapping context.

/NOKEEP_MAP	Data Base will be dettached
/KEEP_MAP	Keep Data Base (DEFAULT)

Caller MDISP,MGOODISP

Author W. Spreng

Example

1.) DELETE PICTURE a

Picture "A" will be deleted.

2.) DELETE PICTURE [*]*test* /LOG

All pictures containing the string "TEST" in their name are deleted and all pictures found are listed.

3.) DELETE PICTURE * /LOG

Delete all pictures in the default Directory and print a list of them.

Remarks

File name GOO$DISP:D$DEPIC.PPL

Created by GOO$DE:E$DECMD.PPL

REMARKS The specified picture could not be deleted if links to other Data Element exist. This could happen if scatter-plots are defined in the picture and if this picture has an entry in the dynamic list!

Description

CALLING

```
STS = D$DEPIC(CV_picture,CV_node,CV_db,CV_pic_dir,
              I_LOG,I_keep_map,B_mask)
```

COMMAND

```
DELETE PICTURE picture node base pic_dir
               /[NO]LOG
               /[NO]KEEP_MAP
```

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name of pictures which should be deleted. Wildcards in picture and directory name are allowed

NODE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

BASE

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

PIC_DIR

Routine arg. Input (*) CHAR(*) VAR

Command par. String global replaceable default=$PICTURE

Default picture Directory.

/LOG

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch default=/NOLOG

List the name of the pictures which has been deleted:

/NOLOG	nothing is listed.
/LOG	Names of deleted pictures are shown.

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 and 1

Command par. Switch default=/KEEP_MAP

Switch to keep or to dettach the whole Data Base mapping context:

/NOKEEP_MAP	Data Base will be deattached
/KEEP_MAP	Keep Data Base (DEFAULT)

Function

The specified Picture Data Element is deleted from the specified Data Base in the specified Directory on the section file on the specified Node.

Wildcards in the picture name and in the Directory are supported.

DELETE POLYGON

```
DELETE POLYGON name poly_dir base node
    /[NO]CONFIRM
    /[NO]KEEP_MAP
```

PURPOSE delete a polygon

PARAMETERS

name String replace default :
name of polygon, may be fully qualified
or wildcarded

poly_dir String replace default : '$POLYGON'
default directory, taken if directory is not
in the name specification

base String replace default : 'DB'
default base, is taken if base name is not in the
name specification

node String replace default : '*'
default node ,...

/[NO] CONFIRM Switch default : '/NOCONFIRM'
Noconfirm to delete polygon

/[NO] KEEP_MAP Switch default : '/KEEP_MAP'
Inhibit the unmap (detach) of the whole Data Base

EXAMPLE DEL POLYGON D::DB:[$polygon]POLY_1

Caller MDE

Author H.G.Essel

File name GOO$DE:E$DLPO.PPL

Dataset -

Function

The header element of a polygon is searched. Then a queued data elements are deleted. Finally the header element is deleted.

Example

```
STS$VALUE=E$DLPO('OTTO','$POLYGON','DB','E',
                  0,1,1)
```

Remarks

REMARKS Module is an action routine.

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$DLPO(CV_NAME,CV_DIR,CV_BASE,
           CV_NODE,I_CONFIRM,I_KEEP_MAP)
```

Argument description:

NAME

Type Input CHAR(*) VAR
name of polygon, may be fully qualified (or wildcarded (nyi))

POLY_DIR

Type Input CHAR(*) VAR
default directory, taken if directory is not in the name specification

BASE

Type Input CHAR(*) VAR
default base, is taken if base name is not in the name specification

NODE

Type	Input CHAR(*) VAR default node ,...

CONFIRM

Type	Input BIN FIXED(15)
0	Confirm to delete polygon
1	Noconfirm to delete polygon

KEEP_MAP

Type	Input BIN FIXED(15)
0	Inhibit unmap of Data Base
1	Unmap of Data Base

Function

The header element of a polygon is searched. Then a queued data elements are deleted. Finally the header element is deleted.
Wildcards for polygon names are supported.

Remarks

Module is an action routine.

Example

```
STS$VALUE=E$DLPO('S1_POLY','$POLYGON','DB',0,1)
```

DELETE POOL

```
DELETE POOL pool base
     /[NO]KEEP_MAP
```

PURPOSE Delete a Data Base Pool

PARAMETERS

pool Pool name
required common default

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE DEL POOL ADAM DB

Caller M$DMCMD

Author M. Richter

File name M$ADLPO.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ADLPO(CV_POOL,CV_BASE,I_KEEP_MAP)

ARGUMENTS

CV_POOL I Pool name
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Delete a Data Base Pool

REMARKS Module is an action routine.

EXAMPLE -

DELETE SECTION

DELETE SECTION base

PURPOSE Delete Global Section attributes

PARAMETERS

base Global Section name
required common default

EXAMPLE -

Caller M$DMCMD

Author M. Richter

File name M$ADLGS.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING STS=M$ADLGS(CV_BASE)

ARGUMENTS

CV_BASE I Global Section name
CHAR(*) VAR

FUNCTION Delete Global Section attributes

REMARKS Module is an action routine.

EXAMPLE -

DELETE SPECTRUM

DELETE SPECTRUM name spec_dir base node
/CAMAC
/[NO]CONFIRM
/[NO]KEEP_MAP

PURPOSE delete a spectrum

PARAMETERS

name String replace default :
name of spectrum, may be fully qualified
or wildcarded

spec_dir String replace default : '$SPECTRUM'
default directory, taken if directory is not
in the name specification

base String replace default : 'DB'
default base, is taken if base name is not in the
name specification

node String replace default : '*'
default node ,...

/CAMAC Switch default : ”
Delete CAMAC spectra only

/[NO] CONFIRM Switch default : '/NOCONFIRM'
Noconfirm to delete spectra

/[NO] KEEP_MAP Switch default : '/KEEP_MAP'
Inhibit the unmap (detach) of the whole Data Base

EXAMPLE DEL SPECTRUM D::DB:[OTTO]S1

Caller MDE

Author K.Winkelmann

File name GOO$DE:E$DLSP.PPL

Dataset -

Function

The header element of a spectrum is searched. Then
a queued data elements are deleted. Existing links
to the analysis tables are deleted. Finally the
header element is deleted.

Example

```
STS$VALUE=E$DLSP('OTTO','$SPECTRUM','DB','E',
                 0,1,1)
```

Remarks

REMARKS Module is an action routine.

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$DLSP(CV_NAME,CV_DIR,CV_BASE,
           CV_NODE,I_CAMAC,I_CONFIRM,I_KEEP_MAP)
```

Argument description:

NAME

Type Input CHAR(*) VAR
name of spectrum, may be fully qualified
(or wildcarded (nyi))

SPEC_DIR

Type Input CHAR(*) VAR
default directory, taken if directory is not
in the name specification

BASE

Type	Input CHAR(*) VAR default base, is taken if base name is not in the name specification

NODE

Type	Input CHAR(*) VAR default node ,...

CAMAC

Type	Input BIN FIXED(15)
0	Delete all specified spectra
1	Delete CAMAC spectra only

CONFIRM

Type	Input BIN FIXED(15)
0	Confirm to delete spectra
1	Noconfirm to delete spectra

KEEP_MAP

Type	Input BIN FIXED(15)
0	Inhibit unmap of Data Base
1	Unmap of Data Base

Function

The header element of a spectrum is searched. Then a queued data elements are deleted. Existing links to the analysis tables are deleted. Finally the header element is deleted.
Wildcards for spectrum names are supported.

Remarks

Module is an action routine.

Example

```
STS$VALUE=E$DLSP('OTTO','DATA','DB',0,1)
```

DISMOUNT BASE

DISMOUNT BASE base

PURPOSE	Delete Global Section attribute of Data Base
PARAMETERS	
base	name of Data Base common replaced default
EXAMPLE	DISMO DATA DB
Author	M.Richter
Caller	M$DMCMD
File name	M$DMDB.PPL

Description

CALLING STS = M$DMDB(cv_db_name)

ARGUMENTS

cv_db_name I Name of the Data Base to be dismounted. A logical name translation will be performed. This name is the name of the (System) Global Section and not of the Section file.
BIN FIXED (31)

FUNCTION The (System) Global Section with the name 'cv_db_name' (a logical name translation will be performed) will be marked for deletion. The actual deletion of the Global Section takes place when all processes that have mapped the Global Section have unmapped all pages. The database is detached.

REMARKS A conversion to upper case characters and a logical name translation will be performed. The name must match with the name known by the VMS system.

EXAMPLE -

DUMP SPECTRUM

DUMP SPECTRUM name spec_dir base node output /[NO]KEEP_MAP

PURPOSE dump spectra to file 'outfile' in ASCII fornmat for transfer to SATAN VSAM library on the IBM

PARAMETERS

name name of spectrum (wildcard)
default = '*'

spec_dir default directory
required common default = '$SPECTRUM'

database default base
required common default = 'DB'

node default node
required common default = 'E'

output output file
required common default = 'OUTFILE'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller E$DMPCM

Author D. Schall

File name E$DMPSP.PPL

Dataset -

EXAMPLE DUMP SPEC ALPHA OUTP=ALPHA1.DATA
dump the spectrum 'ALPHA' to the file 'ALPHA1.DATA'
in ASCII format for transfer to SATAN VSAM library on the IBM

Remarks

REMARKS -

Description

CALLING

```
STS=E$DMPSP(spec_name,spec_dir,database,node,
            outfile,i_keep_map)
```

ARGUMENTS

spec_name name of spectrum. Wildcards are supported:
* x* *x *x* x*y
Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

spec_dir default Directory
replace default='$SPECTRUM'

database default Data Base
replace default='DB'

node default node
replace default='E'

outfile output file for spectra

/[NO] KEEP_MAP inhibits unmapping (detach) of database default /KEEP_MAP

FUNCTION the specified spectra are accessed and written to a file 'outfile' in SATAN AREAD input format with IBM Jobcontrol DD statements as separators.The first part of the file contains a list of the DD names of the dumped spectra. The spectra can be written to a SATAN VSAM analyzer library with the DCL command SIBMSPEC .

REMARKS -

EXAMPLE -

FREEZE CONDITION

```
FREEZE CONDITION name cond_dir base node
  /ON/OFF
  /[NO]KEEP_MAP
```

PURPOSE freeze a condition

PARAMETERS

name name of condition
required

cond_dir default Directory
replace default:'$CONDITION'

base default Data Base
replace default:'DB'

node default node
replace default:'E'

/ON condition result bit is set ON

/OFF condition result bit is set OFF

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$FRECO.PPL

Dataset -

Remarks

REMARKS action routine

Description

CALLING STS=E$FRECO(CV_NAME,CV_DIR,CV_BASE,CV_NODE,CV_ON,
CV_OFF,I_KEEP_MAP)

ARGUMENTS

CV_NAME default directory
CHAR(*) VAR
Input

CV_DIR default directory
CHAR(*) VAR
Input

CV_BASE default base
CHAR(*) VAR
Input

CV_NODE default node
CHAR(*) VAR
Input

CV_ON if on , result bit is set on
CHAR(*) VAR
Input

CV_OFF if on , result bit is set off, if on is on too, result bit will not be modified
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

FUNCTION The freeze bit in the analysis flag tables ([$ANLTABS]ANCO is the default) in which the specified condition lies, is set. This disables any subsequent execution of the condition by the analysis program or any dynamic list. The result bits are left untouched, except /ON or /OFF are specified. Then the preset values in the analysis tables are set accordingly, as well as the result bits.

REMARKS action routine

EXAMPLE -

FREEZE SPECTRUM

FREEZE SPECTRUM name spec_dir base node
/[NO]KEEP_MAP

PURPOSE freeze a spectrum, inhibit accumulation

PARAMETERS

name name of spectrum
required

spec_dir default Directory
replace default:'$SPEC'

base default Data Base
replace default:'DB'

node default node
replace default:'E'
replace default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$FRESP

Dataset -

Remarks

REMARKS -

Description

CALLING STS=E$FRESP(CV_NAME,CV_DIR,CV_BASE,
CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_NAME name of spectrum
CHAR(*) VAR
Input

CV_DIR name of Directory
CHAR(*) VAR
Input

CV_BASE name of Data Base
CHAR(*) VAR
Input

CV_NODE name of node
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

FUNCTION The freeze bit for the corresponding spectrum in analysis tables is set. This bit is interrrogated each time when accumulation is attempted. If it is on , no accumulation will be done. You can reset this bit by using the command UNFREEZE

REMARKS Module is an action routine.

EXAMPLE -

LOCATE BASE

LOCATE BASE base
/[NO]KEEP_MAP

PURPOSE Locate a Data Base
For test purpose only.

PARAMETERS

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALODB.PPL

Dataset -

EXAMPLE LOC DATABASE DB
locate the Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ALODB(CV_BASE, L_KEEP_MAP)

ARGUMENTS

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Locate a Data Base

REMARKS Module is an action routine.

EXAMPLE -

LOCATE DIRECTORY

LOCATE DIRECTORY dir base
/[NO]KEEP_MAP

PURPOSE	Locate a Data Element Directory For test purpose only.
PARAMETERS	
dir	Data Element Directory name required common default
base	Data Base name required common default
/[NO] KEEP_MAP	Inhibit the unmap (detach) of the whole Data Base default:'/KEEP_MAP'
Caller	M$DMCMD
Author	M. Richter
File name	M$ALODI.PPL
Dataset	-
EXAMPLE	LOC DIR EVE DB locate the Directory 'EVE' in the Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ALODI(CV_DIR,CV_BASE,I_KEEP_MAP)

ARGUMENTS

CV_DIR I Data Element Directory name
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Locate a Data Element Directory name

REMARKS Module is an action routine.

EXAMPLE -

LOCATE ELEMENT

LOCATE ELEMENT name
/[NO]KEEP_MAP

PURPOSE Locate a Data Element name array
For test purpose only.

PARAMETERS

name node::base:[dir]element-name(i)
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALODE.PPL

Dataset -

EXAMPLE LOC ELEM DB:[EVE]ADAM
locate the Data Element 'ADAM' of Directory 'EVE'
in Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ALODE(CV_NAME,I_KEEP_MAP)

ARGUMENTS

CV_NAME I Node::base:[dir]element-name(i)
CHAR(*) VAR

I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
FUNCTION	Locate a Data Element name array
REMARKS	Module is an action routine.
EXAMPLE	-

LOCATE ID

LOCATE ID element dir base
/[NO]KEEP_MAP

PURPOSE Locate a Data Element by Directory index
For test purpose only.

PARAMETERS

element ID of Data Element
replaced default:'1'

dir ID of Directory
replaced default:'1'
common default

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALOID.PPL

Dataset -

EXAMPLE LOC ID 12 3 DB
locate Data Element with the index '12' in the Directory with the index '3' of Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ALOID(L_ELEMENT,L_DIR,CV_BASE,I_KEEP_MAP)

ARGUMENTS

L_ELEMENT I ID of Data Element
BIN FIXED(31)

L_DIR I ID of Directory
BIN FIXED(31)

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Locate a Data Element by Directory index

REMARKS Module is an action routine.

EXAMPLE -

LOCATE POOL

LOCATE POOL pool base
/[NO]KEEP_MAP

PURPOSE Locate a Pool in a Data Base
For test purpose only.

PARAMETERS

pool Name of Pool
required common default

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALOPO.PPL

Dataset -

EXAMPLE LOC POOL ADAM DB
locate the Pool 'ADAM' in the Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ALOPO(CV_POOL,CV_BASE,L_KEEP_MAP)

ARGUMENTS

CV_POOL I Name of Pool
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Locate a Pool in a Data Base

REMARKS Module is an action routine.

EXAMPLE -

LOCATE QUEUEELEMENT

LOCATE QUEUEELEMENT element
/[NO]KEEP_MAP

PURPOSE Locate a queue Data Element name array
For test purpose only.

PARAMETERS

element Node::base:[dir]name(i)->type(i) required

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALOQE.PPL

Dataset -

EXAMPLE LOC QUE DB:[EVE]ADAM->L(2)
locate the Data Element with the Data Type 'L'
and the name array index '2' queued to the Data Element 'ADAM' of Directory 'EVE' in the Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ALOQE(CV_ELEMENT,I_KEEP_MAP)

ARGUMENTS

CV_ELEMENT I Node::base:[dir]name(i)->type(i)
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Locate a queue Data Element name array

REMARKS Module is an action routine.

EXAMPLE -

LOCATE TYPE

LOCATE TYPE name base
/[NO]KEEP_MAP

PURPOSE Locate a Type descriptor
For test purpose only.

PARAMETERS

name Name of Type descriptor
required

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ALOTY.PPL

Dataset -

EXAMPLE LOC TYP $SPECTR DB
locate the Type '$SPECTR' in the Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ALOTY(CV_NAME,CV_BASE,I_KEEP_MAP)

ARGUMENTS

CV_NAME I Name of Type descriptor
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Locate a Type descriptor

REMARKS Module is an action routine.

EXAMPLE -

MODIFY DIRECTORY

```
MODIFY DIRECTORY dir entries base
     /[NO]KEEP_MAP
```

PURPOSE Modify a Data Element Directory in a Data Base

PARAMETERS

dir Directory name
required common default

entries Number of entries for Data Element Directory
replaced default:'100'

base Data Base name
required common default

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ARNDI.PPL

Dataset -

EXAMPLE MODI DIR $SPECTRUM 2000
modify the Data Element Directory '$SPECTRUM' of Data Base 'DB' to allow up to 2000 entries in it.

Remarks

REMARKS -

Description

CALLING STS=M$ARNDI(CV_DIR,L_DED_ENTRIES,CV_BASE,
I_KEEP_MAP)

ARGUMENTS

CV_DIR I Directory name
CHAR(*) VAR

L_DED_ENTRIES I Number of entries for the new Data Element Directory
BIN FIXED(31)

CV_BASE I Data Base name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Renew (modify) a Data Element Directory in a Data Base. The size (number of entries) of a Data Element Directory can be changed.

REMARKS Module is an action routine.

EXAMPLE -

MODIFY FRAME SCATTER

```
MODIFY FRAME SCATTER picture frame xparam yparam limits
condition xletter yletter object node base pic_dir par_dir cond_dir
    /XLIN /XLOG /XSQRT [=X_SCALE]
    /YLIN /YLOG /YSQRT [=Y_SCALE]
    /[NO]ROTATE
    /[NO]LETTER
    /[NO]NUMBER
```

PURPOSE Modify a single frame in a picture data element.
Specify only the parameters which should be changed

PARAMETERS

picture	Name of picture frame
frame	Frame which should be modified.
xparam	Parameter for x-axis
yparam	Parameter for y-axis
limits	Limits of displayed spectrum or scatter plot.
condition	Main condition for that frame.
object	Parameter-list which should be checked
xletter	X-lettering on scatterplot axis.
yletter	Y-lettering on scatterplot axis.
node	Default node name for all Data Element name specifications.
base	Default Data Base name for all Data Elements
pic_dir	Default Directory for pictures.
par_dir	default Directory for parameter and condition object

cond_dir default Directory for condition.

X_SCALE Scaling mode for X-axis

/XLIN	Linear X-axis
/XLOG	Logarithmic X-axis
/XSQRT	Squareroot X-axis

Y_SCALE Scaling mode for Y-axis

/YLIN	Linear Y-axis
/YLOG	Logarithmic Y-axis
/YSQRT	Squareroot Y-axis

/ROTATE Rotate displayed spectra
***************** not yet implemented *******

/LETTER Display lettering

/LETTER	Display lettering on axis
/NOLETTER	Display no lettering

/NUMBER Display numbering

/NUMBER	Display numbers on axis
/NONUMBER	Display no numbers on axis

Caller MDISP,MGOODISP,D$DSPCM

Author W. Spreng

Remarks

File name D$MOSFR.PPL

Created by GOO$DISP:D$DSPCM.PPL

Description

CALLING

```
STS=D$MOSFR(CV_picture,L_frame,CV_xparam,cv_yparam,
    CV_limits,CV_condition,CV_object,CV_XLETTER,
    CV_Yletter,cv_node,cv_db,
    CV_pic_dir,CV_par_dir,CV_cond_dir,
    CV_Xscale,CV_Yscale,I_rotate,CV_letter,CV_number,
    B_mask)
```

COMMAND

```
MODIFY FRAME SCATTER picture frame xparam yparam limits
condition object xletter yletter node base pic_dir par_dir cond_dir
    /XLIN /XLOG /XSQRT [=X_SCALE]
    /YLIN /YLOG /YSQRT [=Y_SCALE]
    /[NO]ROTATE
    /[NO]LETTER
    /[NO]NUMBER
```

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Name specification of picture data element in which a scatter frame should be modified.

FRAME

Routine arg. Input CHAR(*) VAR

Command par. String required replaceable

Number of the frame to be modified in the specified picture.

XPARAM,YPARAM

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

X- and Y-Parameter for scatter plot. If specified both parameter are required.

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String replaceable default=(0,1023,0,1023)

Limits of displayed scatterframe. A two dimensional window has to be specified:
(xmin,xmax,ymin,ymax)

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. String

Condition to filter the scatter data in this frame. The scatter data are send for that frame only if the result condition flag is true.

If an empty string is specified the actual condition is deleted from the picture data element.

OBJECT

Routine arg. Input CHAR(*) VAR

Command par. String

Condition objects for the specified condition. If not specified only the result flag of the specified condition is checked. If the specified the condition will be executed with the object. Therefore earlier checks of that condition are lost!

If an empty string is specified the actual object is deleted from the picture data element.

If an object is specified without any condition, it will be ignored.

XLETTER,YLETTER

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

X- and Y-lettering for scatter plot axis. If nothing is specified the parameter names are used

NODE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=*

Default node name.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=DB

Default Data Base where the picture is assumed.

PIC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=$PICTURE

Default Directory name for the specified picture.

PAR_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=DATA

Default Directory name for the specified parameter and condition object.

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=$CONDITION

Default Directory name for the specified condition.

X_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of X-axis. You can select between three modes:

/XLIN Display the axis linear.

/XLOG	Display the axis in logarithmic mode.
/XSQRT	Display the square root of axis.

Y_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of Y-axis. You can select between three modes:

/YLIN	Display the axis linear.
/YLOG	Display the axis in logarithmic mode.
/YSQRT	Display the square root of axis.

ROTATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

*************** Not yet implemented **********

LETTER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the lettering on the axis.

/LETTER	The lettering is displayed.
/NOLETTER	The lettering is not displayed.

NUMBER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the numbering on the axis.

/NUMBER	The numbering is displayed.
/NONUMBER	The numbering is not displayed.

Function

One picture frame is modified with the specified parameter set, the other frame settings are unaffected. Successive modifications are cumulative that means earlier parameter settings are not destroyed.

If one scatterplot event parameter is specified the other one is required! The scatter data can be filtered by a condition. If no condition object is given only the result flag will be checked, if it is true the data are sent to this frame. If an condition object has been found condition will be applied to that object with the concequence that the resulting flag of an earlier condition check is distroyed! Therefore be carefull specifying an object!

A spectrum frame can be changed to a scatter frame if no overlays are defined for the whole pictures!

MODIFY FRAME SPECTRUM

```
MODIFY FRAME SPECTRUM picture frame spectrum limits scallim
scalefactor node base pic_dir spec_dir
            /LIN /LOG /SQRT [=SCALE]
            /XLIN /XLOG /XSQRT [=X_SCALE]
            /YLIN /YLOG /YSQRT [=Y_SCALE]
            /ZLIN /ZLOG /ZSQRT [=Z_SCALE]
            /[NO]WINDOW
            /[NO]LIFE
            /[NO]ROTATE
            /[NO]SMOOTH
            /[NO]LETTER
            /[NO]NUMBER
            /NOCAL/CALAX/CALSPEC [=CALIB]
            /[NO]CHANNELS
            /HISTO/VECTOR/MARKER/CONTOUR-
            /POINT/ISO/CLUSTER/SCATTER [=STYLE]
            /[NO]ERROR
```

PURPOSE Modify a single frame in a picture Data Element. Specify only the parameters which should be changed.

PARAMETERS

picture Name of picture frame

frame Number of frame or range of frames

spectrum Name of spectrum to be displayed in the frame.

limits Limits of displayed spectrum

scalim Limits of scaling axis

scalefactor Scaling factor to increase scaling axis.

node Default node name.

base Default Data Base where the picture is assumed.

pic_dir Default picture Directory name.

spec_dir Default spectrum Directory name.

scale Scaling mode for Y- or Z-axis

/LIN	Linear scaling axis
/LOG	Logarithmic scaling axis
/SQRT	Squareroot scaling axis

X_SCALE Scaling mode for X-axis

/XLIN	Linear X-axis
/XLOG	Logarithmic X-axis
/XSQRT	Squareroot X-axis

Y_SCALE Scaling mode for Y-axis

/YLIN	Linear Y-axis
/YLOG	Logarithmic Y-axis
/YSQRT	Squareroot Y-axis

Z_SCALE Scaling mode for Z-axis

/ZLIN	Linear Z-axis
/ZLOG	Logarithmic Z-axis
/ZSQRT	Squareroot Z-axis

WINDOW Window switch

/NOWINDOW	Display no windows
/WINDOW	Display all associated windows of spectrum

***************** not yet implemented *******

LIFE Life mode switch

/NOLIFE	No life mode
/LIFE	life mode (update event by event)

***************** not yet implemented *******

ROTATE Rotate displayed spectra

/NOROTATE	No rotate
/ROTATE	Rotate

**************** not yet implemented *******

SMOOTH Binnig mode

/SMOOTH	Smooth binning
/NOSMOOTH	min/max binning

LETTER Display lettering

/LETTER	Display lettering on axis
/NOLETTER	Display no lettering

NUMBER Display numbering

/NUMBER	Display numbers on axis
/NONUMBER	Display no numbers on axis

CALIB Perform calibration

/NOCAL	no calibration performed
/CALAX	Calibrate axis
/CALSPEC	Calibrate spectrum

CHANNELS Display channel numbers

/CHANNELS	Display spectrum channels.
/NOCHANNELS	Display no spectrum channels.

STYLE Define style of displayed spectrum.
For one dimensional spectra:

/HISTO	Draw histograms
/VECTOR	Connect spectrum bins with lines
/MARKER	Sign spectrum contents with markers

For two dimensional spectra:

/CONTOUR	Contour lines are displayed
/CLUSTER	Clusters indicating the count rate

/ISO Show spectrum as an isometric plot.

/SCATTER Simulate scatter plot data.

/[NO] ERROR Draw statistical error bars for each bin.

/NOERROR No error bars drawn.

/ERROR Error bars are drawn at each bin.

Caller MDBM,MGOODDM

Author W. Spreng

Example

1.) MODIFY FRAME SCATTER pic 1 [$spectrum]s1
Spectrum in frame 1 of picture "pic" has been changed to "s1".
2.) MODIFY FRAME SPECTRUM pic 4:9 [$spectrum]s1(1:5) /log/vector
Spectra "s1(1)...s1(5)" are placed into the frames 4,...,9.

Remarks

File name D$MODFR.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=D$MODFR(CV_picture, CV_frame, CV_spectrum, CV_limits, CV_scalim, R_scalefactor, cv_node, cv_base, CV_pic_dir, CV_spec_dir, CV_scale, CV_Xscale, CV_Yscale, CV_Zscale, I_window, I_life, I_rotate, I_smooth, CV_letter, CV_number, CV_calib, CV_channels, CV_style, I_error, B_mask)

COMMAND MODIFY FRAME SPECTRUM picture frame spectrum limits scalim scalefactor node base pic_dir spec_dir

/LIN /LOG /SQRT [=SCALE]
/XLIN /XLOG /XSQRT [=X_SCALE]
/YLIN /YLOG /YSQRT [=Y_SCALE]
/ZLIN /ZLOG /ZSQRT [=Z_SCALE]
/[NO]WINDOW
/[NO]LIFE
/[NO]ROTATE
/[NO]SMOOTH

/[NO]LETTER
/[NO]NUMBER
/NOCAL/CALAX/CALSPEC [=CALIB]
/[NO]CHANNELS
/HISTO/VECTOR/MARKER/CONTOUR-
/POINT/ISO/CLUSTER/SCATTER [=STYLE]
/[NO]ERROR

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable required

Name of the picture which should be modified.

FRAME

Routine arg. Input CHAR(*) VAR

Command par. String replaceable required

Number of the frame to be modified. If a subset of a spectrum array should be placed into subsequent frames, the frames could be defined like:

start:stop

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

Name of spectrum which should be put into the picture frame. If a subset of a spectrum array should be selected specify:

spectrum(start:stop)

In that case a frame range, with the same number of elements is required.

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String

Limits of displayed spectrum or scatterplot
(100,1024) for 1-dimensional spectra
(100,1024,500,4096) for 2-dimensional spectra

SCALIM

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

Fixed limits of scaling axis. With this parameter the absolute range of scaling axis could be specified.

SCALEFACTOR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable

Factor to increase scaling axis. With this factor the upper limit of the scaling axis could be modified in relativ units,e.g. if "scafac"=2.0 the upper limits is increased by a factor of "2.0". This is usefull if overlayed spectra with dynamic offset are defined for that frame (see DEFINE OVERLAY command).

NODE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=*

Default node name.

BASE

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=DB

Default Data Base where the picture is assumed.

PIC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=$PICTURE

Default Directory name for picture Data Elements.

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. String replaceable global default=$SPECTRUM

Default spectrum Directory

SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Set the display mode on the scaling axis. You can select between three modes:

/LIN	Display the linear count rate.
/LOG	Display the spectrum contents in logarithmic mode.
/SQRT	Display the square root of the spectrum contents.

X_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of X-axis. You can select between three modes:

/XLIN	Display the axis linear.
/XLOG	Display the axis in logarithmic mode.
/XSQRT	Display the square root of axis.

Y_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of Y-axis. You can select between three modes:

/YLIN	Display the axis linear.
/YLOG	Display the axis in logarithmic mode.
/YSQRT	Display the square root of axis.

Z_SCALE

Routine arg. Input CHAR(*) VAR

Command par. Set

Display mode of Z-axis. You can select between three modes:

/ZLIN	Display the axis linear.
/ZLOG	Display the axis in logarithmic mode.
/ZSQRT	Display the square root of axis.

WINDOW

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

*************** Not yet implemented **********

LIFE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

*************** Not yet implemented **********

ROTATE

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

*************** Not yet implemented **********

SMOOTH

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

To optimize the displayed spectrum data, the display reduces the number of displayed bins by an internal display binsize. The spectrum bins can be gathered in to ways:

/SMOOTH Display the spectra with smooth binning. In that mode the mean values of the channel contents over the display binsize will be shown. The effect is that the spectrum looks smoother, but the displayed spectrum contents could be fractional numbers.

/NOSMOOTH The minimum and maximum contents of the display bins are shown. In that modes spikes in the spectra are not smoothed out.

LETTER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the lettering on the axis.

/LETTER The lettering is displayed.

/NOLETTER The lettering is not displayed.

NUMBER

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactiveate the numbering on the axis.

/NUMBER The numbering is displayed.

/NONUMBER The numbering is not displayed.

CALIB

Routine arg. Input CHAR(*) VAR

Command par. Set

Display the spectrum in calibrated units. To do this the displayed spectra has to be connected to an existing calibration. Different calibration modes are availible:

/CALAX	The axis is drawn in calibrated units and the spectrum in uncalibrated units. Therefore the distance between two subsequent tics varies.
/CALSPEC	The spectrum is drawn in calibrated units. Then the width of the displayed spectrum bins varies.
/NOCAL	No calibration is performed to the displayed spectra.

In any case the axis with uncalibrated units is displayed too. To prevent this specify the /NOCHANNELS switch!

/CHANNELS

Routine arg. Input CHAR(*) VAR

Command par. Set

Activate or deactivate the display of an axis with the original channel units.

/CHANNELS	Display an axis with original units.
/NOCHANNEL	The axis with original units is not not displayed.

STYLE

Routine arg. Input CHAR(*) VAR

Command par. Set

Defines the display style of the spectrum data. For one and two dimensional spectra different styles are implemented.

One dimensional spectra:

/HISTO	Histograms are generated
/VECTOR	The spectrum contents are connected by polylines.
/MARKER	The spectrum contents are indicated by markers.

For two dimensional spectra:

/CONTOUR	Contour lines are displayed.
/CLUSTER	The spectrum count rates are indicated by clusters of a variable size and colour.
/ISO	A 3D isometric plot is generated.
/SCATTER	The countrate in each bin is simulated by a number of randomly distributed points in that bin. The result is a representation similiar to a scatter plot.

ERROR

Routine arg. Input BIN FIXED(15) valid values 0 or 1.

Command par. Switch negatable

Display statistical errors at each channel of a one dimensional spectrum.

/ERROR	Statistical errors (the square root of the count rate) are displayed.
/NOERROR	No errors are displayed

Function

One picture frame is modified with the specified parameter set, the other frame settings are unaffected. Successive modifications are cumulative that means earlier parameter settings are not destroyed.

Several members of an array of spectra are put into subsequent frames if a subset of spectra and range of frames is specified, like:

MODIFY FRAME SPECTRUM picture 1:8 spectrum(3:10)

It is possible to change the dimension of the spectrum in the specified frame, if no overlayed spectra are defined. Changes from scatter to spectrum frames are allowed too, with the restriction that no overlays are defined for the whole picture! For the scaling axis fixed limits are defined with 'scalim' or you can set a factor 'scafac' to increase the maximum limit of the scaling axis in dependence of the actual size of the spectrum maximum. This is useful if for e.g. if the contents of the spectrum which should be displayed increases, but the ratio between the maximum channel contents and the maximum of the scaling axis should be fixed.

MODIFY TABLE CONDITION

```
MODIFY TABLE CONDITION name entries
      directory pool base node
      /[NO]KEEP
```

PURPOSE modify condition bit table

PARAMETERS

name string replace default: 'ANCO'
name for condition table to be replaced

entries integer required:
number of needed conditions

directory string replace default: '$ANLTABS'
default directory

pool string replace default: '$COND_POOL'
default pool

base string replace default: 'DB'
default data base

node string replace default: 'E'
default node

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole data base

Caller MDBM,MGOODBM

Author Th. KROLL

EXAMPLE

```
MODIFY TABLE CONDITION ENTRIES=4096
```

REMARKS

File name GOO$DE:E$AMOCT.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$AMOCT(CV_NAME,L_ENTRIES,CV_DIR,CV_POOL,
            CV_BASE,CV_NODE,I_KEEP_MAP)
```

COMMAND

```
MODIFY TABLE CONDITION name entries
   directory pool base node
   /[NO]KEEP
```

Argument /parameter description

NAME

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'ANCO'

ENTRIES

Routine arg. Input BIN FIXED(31)

Command par. integer required:

DIRECTORY

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$ANLTABS'

POOL

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$COND_POOL'

BASE

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'DB'

NODE

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'E'

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP
In a procedure call this argument should be
always 1 to prevent an unmap of the data base.

FUNCTION

Modify condition bit table to specified size

MODIFY TABLE SPECTRUM

MODIFY TABLE SPECTRUM name entries
directory pool base node
/[NO]KEEP

PURPOSE modify spectrum bit table

PARAMETERS

name string replace default: 'ANSP'
name for spectra table to be replaced

entries integer required:
number of needed spectras

directory string replace default: '$ANLTABS'
default directory

pool string replace default: '$SPEC_POOL'
default pool

base string replace default: 'DB'
default data base

node string replace default: 'E'
default node

/[NO] KEEP_MAP switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole data base

Caller MDBM,MGOODBM

Author Th. KROLL

EXAMPLE

```
MODIFY TABLE SPECTRUM ENTRIES=4096
```

REMARKS

File name GOO$DE:E$AMOST.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$AMOST(CV_NAME,L_ENTRIES,CV_DIR,CV_POOL, CV_BASE,CV_NODE,I_KEEP_MAP)

COMMAND MODIFY TABLE SPECTRUM name entries
directory pool base node
/[NO]KEEP
Argument /parameter description

NAME

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'ANSP'

ENTRIES

Routine arg. Input BIN FIXED(31)

Command par. integer required:

DIRECTORY

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$ANLTABS'

POOL

Routine arg. Input CHAR(*) VAR

Command par. string replace default: '$SPEC_POOL'

BASE

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'DB'

NODE

Routine arg. Input CHAR(*) VAR

Command par. string replace default: 'E'

/KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. negatable switch default: /KEEP_MAP
In a procedure call this argument should be
always 1 to prevent an unmap of the data base.

FUNCTION

Modify Spectrum bit table to specified size

MOUNT BASE

```
MOUNT BASE base basefile
    /PERMANENT/TEMPORARY
    /GLOBAL_SEC/SYSTEM_GLOBALSEC
```

PURPOSE Mount an existing Data Base (section)

PARAMETERS

base Data Base name
required common default

basefile Data Base file name
required common default

/PERMANENT/TEMPORARY Section permanence
default:'/PERMANENT'

/GLOBAL_SEC/SYSTEM_GLOBALSE Section scope
default:'/GLOBAL_SEC'

Caller M$DMCMD

Author M. Richter

File name M$AMODB.PPL

Dataset -

EXAMPLE MOU DAT DB DB
mount the Data Base 'DB' using the section file 'DB.SEC' from the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING STS=M$AMODB(CV_BASE,CV_BASEFILE,
CV_PERMANENT,CV_GLOBAL)

ARGUMENTS

CV_BASE I Data Base name
CHAR(*) VAR

CV_BASEFILE I Data Base file name
CHAR(*) VAR

CV_PERMANENT I Section permanence
CHAR(*) VAR

CV_GLOBAL I Section scope
CHAR(*) VAR

FUNCTION Mount an existing Data Base (section)

REMARKS Module is an action routine.

EXAMPLE -

PROTECT SPECTRUM

```
PROTECT SPECTRUM name spec_dir base node
    /LOG
    /[NO]KEEP_MAP
```

PURPOSE protect one (or all) spectrum

PARAMETERS

NAME required string global replace default: ”
Name of Spectrum (wildcard) to be protected
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.

SPEC_DIR String global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE String global replace default: 'DB'
Name of Data Base

NODE String global replace default: 'E'
Name of node

/LOG Switch default:
Output names of protected spectra.

[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller mdbm

Author H.G.Essel

Example

PROT SP A(3,9) protect one spectrum
PROT SP A(3) protect one spectrum
PROT SP A(3:9) protect seven spectra
PROT SP A(*) protect all members
PROT SP A* protect all members
PROT SP A protect all members

Remarks

File name E$PROSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$PROSP(CV_NAME,CV_SPEC_DIR,CV_BASE,CV_NODE,
I_LOG,I_KEEP_MAP)

COMMAND PROTECT SPECTRUM name spec_dir base node
/LOG
/[NO]KEEP_MAP
Argument description

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of Spectrum (wildcard) to be protected
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.
Arrays without index are protected totally.
For one dimensional arrays a range may be
specified like x(3:5). No wildcard is allowed in
this case. Single members of one and twodimensional
arrays may be protected by specifying the index:
X(7) or Y(4,5).

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Name of Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Name of Node

LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: "
Output protected spectrum names.

KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: "
Inhibit the unmap (detach) of the Data Base

Function

The spectrum will not be cleared by CLEAR command.
To clear the spectrum it must be UNPROTECTed first.

READ CAMAC SPECTRUM

```
READ CAMAC SPECTRUM name spec_dir base node
    /ADD
    /LOG
    /[NO]KEEP_MAP
```

PURPOSE Read spectrum data from MR2000 into GOOSY spectrum

PARAMETERS

NAME required string global replace default: ”
Name of Spectrum (wildcard) to be copied
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.

SPEC_DIR String global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE String global replace default: 'DB'
Name of Data Base

NODE String global replace default: 'E'
Name of node

/LOG Switch default: ”
Output list of copied spectra

/ADD Switch default: ”
Add spectrum channel contents rather than overwrite.

[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller mdbm

Author H.G.Essel

Example

```
READ CA SP A(1,2)
READ CA SP A(*)
READ CA SP A* /ADD
READ CA SP A (clear first spectrum only)
```

Remarks

File name E$ARCSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$ARCSP(CV_NAME,CV_SPEC_DIR,CV_BASE,CV_NODE,
            I_LOG,I_ADD,I_KEEP_MAP)
```

COMMAND

```
READ CAMAC SPECTRUM name spec_dir base node
          /ADD
          /LOG
          /[NO]KEEP_MAP
```

Argument description

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of Spectrum (wildcard) to be copied
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Name of Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Name of Node

LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Output list of copied spectra

ADD

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Instead or overwrite, add MR2000 channels to spectrum channels.

KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Inhibit the unmap (detach) of the Data Base

Function

The data part of MR2000 is copied into specified * GOOSY spectrum. The range in the MR2000 is defined in the spectrum. Channel contents may be overwritten (default) or added (/ADD).

SET CONDITION PATTERN

SET CONDITION PATTERN name pattern invpat index cond_dir base node
/[NO]KEEP_MAP

PURPOSE set stored pattern of a pattern condition

PARAMETERS

name String default :
Name of condition

pattern String default : 0
Specification of pattern
String default : 0
Specification of inverse pattern
Integer default : 1
Internal index

cond_dir String replace default: '$CONDITION'
Default condition Directory

base String replace default : 'DB'
Default Data Base

node String replcae default : '*'
Default node

/[NO] KEEP_MAP Set default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$SECOP.PPL

Dataset -

NAME

TYPE String default: none
Name of condition which is wanted to be modified

PATTERN

TYPE String default: 0
Pattern specification, may be of the form '010101'B or 010101

INVPAT

TYPE String default: 0
Inversion pattern specification, may be of the same form

INDEX

TYPE Integer default: 1
Internal index

COND_DIR

TYPE String replace default: none
Default condition directory

BASE

TYPE String replace default: none
Default Data Base

NODE

TYPE String replace default: none
Default node

KEEP_MAP

TYPE Switch default: /KEEP_MAP
Valid values are:

[NO] KEEP_MAP	Inhibit unmap of Data Base atfer command execution

Function

to modify the pattern of a pattern condition

Example

Remarks

REMARKS	Module is an action routine

Description

Argument description

NAME

Type	Input CHAR(*) VAR Name of condition which is wanted to be modified

SPEC

Type	Input CHAR(*) VAR Pattern specification, may be of the form '010101'B or 010101

INVPATT

Type	Input CHAR(*) VAR Inversion pattern specification, may be of the same form

IND

Type	Input CHAR(*) VAR Internal index

DIR

Type Input CHAR(*) VAR
Default condition directory

BASE

Type Input CHAR(*) VAR
Default Data Base

NODE

Type Input CHAR(*) VAR
Default node

KEEP_MAP

Type Input BIN FIXED(15)
Valid values are:

0 Unmap the Data Base

1 Inhibit unmap of the Data Base
Inhibit unmap of Data Base atfer command execution

Function

to modify the pattern of a pattern condition

Remarks

Module is an action routine

Example

SET CONDITION WINDOW

SET CONDITION WINDOW condition limits dimension cond_dir base node
/[NO]KEEP_MAP

PURPOSE Set window condition

PARAMETERS

condition Name of window condition

limits Limits for condition window

dimension Dimensions for multi-dimensional condition windows which should be set.

cond_dir Default condition Directory

base Default Data Base name

node Default node name

Caller MDBM,MGOODBM

Author W. Spreng

Example

1.) SET CONDITION WINDOW c1 100,400 5
Lower limit 100; upper limit 400 is set in dimension 5 of condition C1.
2.) SET CONDITON WINDOW c1 100,400 *
Lower limit 100; upper limit 400 is set in all dimensions of condition C1.
3.) SET CONDITON WINDOW c1 100,400 3:5
Lower limit 100; upper limit 400 is set in dimension 3 to 5 of condition C1.
4.) SET CONDITON WINDOW c_array 100,400 3:5
Lower limit 100; upper limit 400 is set in dimension 3 to 5 of condition C_ARRAY(1).
5.) SET CONDITON WINDOW c_array(2:4) 100,400 3:5

Lower limit 100; upper limit 400 is set in dimension 3 to 5 of condition C_ARRAY(2) to (4)
6.) SET CONDITON WINDOW c_array(*) 100,400 *
Lower limit 100; upper limit 400 is set in all dimensions of all condition array members.

Remarks

File name E$SCWIN.PPL

created by GOO$DE:E$DECMD.PPL

Description

CALLING

```
STS=E$SCWIN(CV_condition,CV_limits,CV_dimension,
            CV_cond_dir,CV_base,cv_node,I_KEEP_MAP)
```

COMMAND

```
SET CONDITION WINDOW condition limits dimension cond_dir
base node
        /[NO]KEEP_MAP
```

CONDITION

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of condition window. Valid inputs are:
COND - for a single condition window
COND(3) - for a single member in a condition array
COND(1:4) - for a several members of a condition array
COND(*) - for all members of a condition array
Wildcards in the condition and directory name are not supported.

LIMITS

Routine arg. Input CHAR(*) VAR

Command par. String required

Limits for condition window, only one pair of limits is supported. The specified limits are set for all dimensions.

DIMENSION

Routine arg. Input CHAR(*) VAR

Command par. String default=*

Dimension in which the limits should be set. Possible input values:
n - single number
n:m - range
* - set all dimensions

COND_DIR

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=$CONDITION

Default condition Directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. String global replaceable default=*

Default node for Data Base

KEEP_MAP

Routine arg. Input BIN FIXED(31) valid values 0 and 1

Command par. Switch negatable default=/KEEP_MAP

If set the mapping context will be kept. If /NOKEEP_MAP specified the Data Base will be dettached after the command completion.

Function

Set limits in the specified window condition. For multi-dimensional conditions it is necessary to specify the dimensions which should be set. If the window limits are specified the are set in all specified dimensions.

SET LETTERING

SET LETTERING specname dim text spec_dir base node /[NO]KEEP_MAP

PURPOSE set lettering at display axisses

PARAMETERS

specname name of spectrum
Wildcards in the directory, data element name and dataelement index are supported.
required

dim dimension (0 ... 8)
required

text text to be written at the axis

spec_dir default Directory
repalce default:'$SPECTRUM'

base default Data Base
repalce default:'DB'

node default node
repalce default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$SELET.ppl

Dataset -

Examples

SET LETTERING spectrum 0 ”This is the x-axis”
SET LETTERING spectrum 1 ”This is the y-axis”

Description

CALLING STS=E$SELET(CV_SPECNAME,CV_DIM,CV_TEXT
CV_SPEC_DIR,CV_BASE,CV_NODE,I_KEEP_MAP,B_MASK)

COMMAND SET LETTERING specname dim text spec_dir base node
/[NO]KEEP_MAP

Argument and parameter description.

NAME

Routine par. Input CHAR(*) VAR

Command arg. String requested replacable

Name of spectrum. Wildcards in the directory, data element name and dataelement index are supported.

DIM

Routine par. Input CHAR(*) VAR

Command arg. String requested

Dimension to be modified, can be between 0 and 8. Where 0 is the x-axis, 1 the y-axis, 2 the z-axis etc.

TEXT

Routine par. Input CHAR(*) VAR

Command arg. String

Text to be written at the axis. If there are blanks in the text, it has to be enclose in quotes (”).

SPEC_DIR

Routine par. Input CHAR(*) VAR

Command arg. String global replacable default=$SPECTRUM

Default spectrum directory name.

BASE

Routine par. Input CHAR(*) VAR

Command arg. String global replacable default=DB

Default data base name.

NODE

Routine par. Input CHAR(*) VAR

Command arg. String global replacable default=*

Default node name.

/KEEP_MAP

Routine par. Input BIN FIXED(15)

Command arg. Switch

Inhibit unmap of Data Base

Function

FUNCTION The text at the axisses are written into the spectrum data element.

SET MEMBER

SET MEMBER mem_spec value dir base node
/[NO]KEEP_MAP

PURPOSE Change value of an Data Element member or a full Data Element member array (wildcarded).

PARAMETERS

mem_spec Node::base:[dir]name(i)->type(i)
required common default

value Enter new value for DE member
required

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author M. Richter

File name M$ASTME.PPL

Dataset -

EXAMPLE SET MEMB DB:[DATA]ADAM.ALPHA.BETA 4711
set the value '4711' in the member 'BETA' of
structure 'ALPHA' of Data Element 'ADAM' in Directory 'DATA' of Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ASTME(CV_ELEMENT,CV_VALUE,
CV_DIR,CV_BASE,CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_ELEMENT I Node::base:[dir]name(i)->type(i)
CHAR(*) VAR

CV_VALUE I New value for DE member
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Change value of an Data Element member

REMARKS Module is an action routine.

EXAMPLE -

SET SPECTRUM POINT

SET SPECTRUM POINT spectrum xpoint ypoint file
spec_dir base node
/[NO]KEEP_MAP

PURPOSE	Set spectrum channel to specified value.
PARAMETERS	
spectrum	Name of spectrum to be set.
xpoint	List of x-values converted into spectrum channels.
ypoint	List of y-values used as spectrum contents.
file	Name of file, used to read X-Y values.
spec_dir	Default spectrum directory.
base	Default data base name.
node	Default node name.
/KEEP_MAP	Keep Data Base mapping context.
Caller	MDBM,MGOODBM,E$DECMD
Author	W. Spreng

Example

1. SET SPECTRUM POINT a 300,400,500 3.7,5.8,10.5
The spectrum points 300,400,500 are converted into spectrum indices of spectrum ”a” and the values 3.7,5.8,10.5 are written into these spectrum bins.
2. SET SPECTRUM POINT a FILE=daten.dat
The X-Y values are read from the specified file.
3. SET SPECTRUM POINT a 100,200 20.6,40.8 FILE=daten.dat
The specified x-y values and the values read from file are put into the spectrum ”a”.

Remarks

File name E$SSPPO.PPL

Created by GOO$DE:D$DECMD.PPL

Description

CALLING STS=E$SSPPO(CV_spectrum,RA_XPOINT,RA_YPOINT,CV_FILE, CV_spec_dir,CV_base,CV_node,I_keep_map,LA_ELEMENTS)

COMMAND

```
SET SPECTRUM POINT spectrum xpoint ypoint file
                   spec_dir base node
                   /[NO]KEEP_MAP
```

SPECTRUM

Routine arg. Input CHAR(*) VAR

Command par. String

Name of the spectrum in which the points should be set. Up to now only one dimensional spectra are supported.

XPOINT

Routine arg. Input (*) BIN FLOAT(24)

Command par. String

List of x-values, which are converted into the corresponding spectrum indices. Several values, separated by commas can be specified.

YPOINT

Routine arg. Input (*) BIN FLOAT(24)

Command par. String

A list of values, which are put into the spectrum channels as given in the parameter "XPOINT". The number of X and Y values have to be the same!

FILE

Routine arg. Input CHAR(*) VAR

Command par. String

Name of the file, used to read the X and Y-values. The format of the file is very simple: A "!" at the begin of each record indicates a comment line. In each record containing a valid input two values are expected. The first value is the x-value and the second separated by blanks or comma is the corresponding y-value.

SPEC_DIR

Routine arg. CHAR(*) VAR

Command arg. String global replaceable default=$SPECTRUM

Default Directory name for spectra.

BASE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=*

Node name for Data Base section file.

KEEP_MAP

Routine arg. Input BIN FIXED(15), valid inputs are 0 and 1

Command par. Switch

If this flag is set the context of all Data bases will be kept:

/KEEP_MAP Keep mapping context.

/NOKEEP_MAP	Database will be detached.

FUNCTION

The specified points are set in the spectrum. The x-values are converted into a valid spectrum index and the corresponding y-value is written into that bin. The x and y-pairs can be specified directly and/or read from a file with a simple input format:

A "!" at the begin of each record indicates a comment line. In each record containing a valid input two values are expected. The first value is the x-value and the second, separated by blanks or comma, is assumed to be the corresponding y-value.

SHOW AREA

```
SHOW AREA area base output
    /[NO]FULL
    /[NO]DIRECTORY
    /[NO]KEEP_MAP
    /PRINT
```

PURPOSE Show an Area in a Data Base

PARAMETERS

area Area name
required common default

base Data Base name
required common default

output optional output file
optional

/[NO] FULL Show full Area information
default: '/NOFULL' Area information without bit map

/[NO] DIRECTORY Show Area directory
default: '/NODIRECTORY'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHAR.PPL

Dataset -

EXAMPLE SHO AR ALPHA OUT='ALPHA.LIS'
show the Area 'ALPHA' of Data Base 'DB' and write the output into file 'ALPHA.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING

```
STS=M$ASHAR(CV_AREA,CV_BASE,CV_OUT,I_FULL,
              I_DIRECTORY,I_KEEP_MAP,I_PRINT)
```

ARGUMENTS

CV_AREA I Area name
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

CV_OUT I optional file for output
CHAR(*) VAR

I_FULL I Show full information
BIN FIXED(15)

I_DIRECTORY I Show Area Directory
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

I_PRINT I Print output
BIN FIXED(15)

FUNCTION Show an Area in a Data Base

REMARKS Module is an action routine.

EXAMPLE -

SHOW CALIBRATION

```
SHOW CALIBRATION calibration output cal_dir base node
      /PRINT
      /LINKS
      /TABLE
```

PURPOSE	Show calibration information
PARAMETERS	
calibration	Name of calibration.
output	Output file
cal_dir	Default calibration directory
base	Default data base name
node	Default node name
/PRINT	Output file is printed
/LINKS	Links to all spectra are listed.
/TABLE	Show total table contents.
Caller	MDBM, MGOODBM
Author	W. Spreng

Examples

1.) SHOW calibration [*]*
All calibrations in all existing directories are listed.
2.) SHOW calibration [*]A* /LINKS
Calibrations starting with ”A” are listed, additionally all spectra calibrated with these data elements are listed.

Remarks

Created by E$SHECM.PPL

Module name GOO$DE:E$SHCAL.PPL

Description

CALLING STS=E$SHCAL(CV_calibration,CV_output,CV_cal_dir, CV_base, CV_node, I_print, I_links, I_table)

COMMAND

```
SHOW CALIBRATION calibration output cal_dir base node
     /PRINT
     /LINKS
     /TABLE
```

CALIBRATION

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of calibration which should be shown. Any wildcardspecification is allowed.

OUTPUT

Routine arg. Input CHAR(*) VAR

Command par. String

File in which the output should performed. The output will be printed with the /PRINT switch.

CAL_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$CALIB

Default Picture Directory.

BASE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

PRINT

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

The generated output is printed on the default print queue.

LINKS

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

All established links to spectra are shown. Therefore all spectra connected to the specified calibration are listed.

TABLE

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

The contents of the calibration table is listed.

Function

Informations about calibrations should be shown. All existing links to spectra and/or the total table contents can be listed.

SHOW CAMAC SPECTRUM

```
SHOW CAMAC SPECTRUM name spec_dir
                    base node output width
  /PRINT
  /ATTRIBUTES/DATA/ALL/STATUS
  /FULL
  /MEMBERS
  /LOG
  /INTEGRAL
  /ZERO
  /CAMAC
  /[NO]KEEP_MAP
```

PURPOSE show CAMAC spectra

PARAMETERS

name String default: '*'
name of spectrum. Wildcards are supported:
* x* *x *x* x*y Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

spec_dir String replace default: '$SPECTRUM'
Name of Spectrum directory.

base String replace default: 'DB'
Name of Data Base.

node String replace default: '*'
Name of Node.

output String default: none
optional output file

width Integer global replace default: 80
Line length for histogram (80 or 132)

/PRINT Switch default: none
Print output

/WHAT Set replace default: /ATTRIBUTES
/ATTRIBUTES :Display attributes of spectra
/DATA :Display data
/STATUS :calls SHOW TAB/SP
/ALL :Display all

/FULL Switch default: none
Display full spectrum information

/MEMBERS Switch default: none
Display information for all spectrum members

/LOG Switch default: none
Display data in logarithmic format

/INTEGRAL Switch default: none
Display channel contents integral

/ZERO Switch default: none
Display channel contents including zero channels

/CAMAC Switch default: /CAMAC
Display data of MR2000 memory (=default)

/[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

FUNCTION The specified spectrum is accessed and various parts of it are displayed on the terminal. The data part of the spectrum is fetched from the MR2000 memory optionally.

Caller E$SHECM

Author K.WINKELMANN

File name E$SHCSP.PPL

Dataset GOO$DE:E$SHCSP.PPL

Function

The specified spectrum is accessed and various parts of it are displayed on the termina. AS long as the data of the spectrum is not accessed, the command works like SHOW SPECTRUM /CAMAC. The data filed of the spectrum may, however, be fetched from the MR2000 memory rather then from the data base.

Example

SHOW CAM SPEC OTTO /DATA
shows MR2000 data of spectrum otto
SHOW CAM SPEC OTTO /NOCAMAC /DATA
shows data base data of spectrum otto
SHOW CAM SPEC *
shows list of all CAMAC spectra.

Remarks

REMARKS Module is an action routine.

Created by E$SHECM.PPL

Description

COMMAND

```
SHOW CAMAC SPECTRUM name spec_dir
                        base node output width
  /PRINT
  /ATTRIBUTES/DATA/ALL/STATUS
  /FULL
  /MEMBERS
  /LOG
  /INTEGRAL
  /ZERO
  /CAMAC
  /[NO]KEEP_MAP
```

CALLING

```
STS=E$SHCSP(CV_NAME,CV_DIR,CV_BASE,CV_NODE,CV_OUT,

              L_WIDTH,I_PRINT,CV_DETAIL,I_FULL,
              I_MEMBERS,I_LOG,I_INTEGRAL,I_ZERO,
              I_CAMAC,I_KEEP_MAP,B_MASK)
```

Argument description

NAME

Type Input CHAR(*) VAR

Name of spectrum. Wildcards are supported:
* x* *x *x* x*y Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

DIR

Type Input CHAR(*) VAR
Default directory

BASE

Type Input CHAR(*) VAR
Default data base

NODE

Type Input CHAR(*) VAR
Default node

OUTPUT

Type Input CHAR(*) VAR
Optional file for output

WITDH

Type Input BIN FIXED(31)
Line size for histogram (80 or 132).

PRINT

Type Input BIN FIXED(15)
valid values are:
No print of output file is done Print output file on line printer

WHAT

Type Input CHAR(*) VAR
valid values are:
Display attributes of the spectrum Display data of the spectrum Display attributes,all and status of the spectrum Display status of spectrum

MEMBERS

Type Input BIN FIXED(15)
valid values are:

Display only first and last member of spectrum. Display information about all spectrum members.

FULL

Type Input BIN FIXED(15)
valid values are:
Display only short info about the spectrum: Spectrumname,spectrumtype,limits,bins,bits
Display full information about the spectrum: Spectrumname,spectrumtype,Database,Dire
Dimensions, Index in analyzetable, Creationdate Lettering of X-Axis,Lettering of Y-Axis, Lower limit, upper limit, Number of bins, Binsize, Factor, Offset, Underflow - and Overflow counts, Underflow and Overflow contents, freeze and executed bit

LOG

Type Input BIN FIXED(15)
Valid values are:
Display decimal data Display logarithmic data

INTEGRAL

Type Input BIN FIXED(15)
Valid values are:
Display channel contents Display integral channel contents

ZERO

Type Input BIN FIXED(15)
Valid values are:
Display channel contents without zero's Display channel contents with zero's

CAMAC

Type Input BIN FIXED(15)
Valid values are:
Display data of spectrum in data base Display data of spectrum in MR2000 memory

KEEP_MAP

Type	Input BIN FIXED(15) Valid values are: Unmap Data Base Inhibit unmap of the Data Base

B_MASK

Type	Output BIT(32) ALIGNED

Function

The specified spectrum is accessed and various parts of it are displayed on the terminal. The difference to SHOW SPECTRUM is that the contents of CAMAC spectra in the MR2000 memory can be output.

Remarks

Module is an action routine.

Example

SHOW CONDITION

```
SHOW CONDITION name cond_dir base node output
  /PRINT
  /FULL
  /MEMBERS
  /ATTRIBUTES/COUNTERS/FLAGS/STATUS/ALL
  /POLY/WIND/MULTI/PATT/COMP/FUNC/ANY
  /[NO]KEEP_MAP
```

PURPOSE show attributes of a condition

PARAMETERS

name String replace default: '*'
Name of condition. Wildcards are supported:
* x* *x *x* x*y
Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.
default=*

cond_dir String replace default: '$CONDITION'
Default Directory

base String replace default: 'DB'
Default Data Base

node String replace default: '*'
Default node

output String replace default: none
Optional output file

/PRINT Switch default: none
Print output

/FULL Switch default: none
Full output

/MEMBERS Switch default: none
Output all members of indexed conditions.

what Set default: none

/ATTRIBUTES	show all attributes of a condition
/COUNTERS	show all counters of a condition
/FLAGS	show all flags
/FULL	show everything
/STATUS	show bit tables and counters

/all show all of them Show bit tables and counters

TYPE Set default: /ANY

/WIND	show window conditions
/MULTI	show multi window conditions
/PATT	show pattern conditions
/COMP	show composed conditions
/POLY	show polygon conditions
/FUNC	show function conditions
/ANY	show all types of conditions

Show type of conditions

KEEP_MAP switch default: /KEEP_MAP

/NOKEEP_MAP	Unmap the Data Base
/KEEP_MAP	inhibit the unmap (detach) of the Base

FUNCTION Condition 'name' is searched and its attributes are listed, esp. type, window limits, pattern bits, related conditions, condition table name

EXAMPLE SHOW CONDITION emil /ATTR
shows all attributes from condition emil

Caller E$SHECM

Author K.Winkelmann

File name E$SHCO.PPL

Dataset -

Description

COMMAND

```
SHOW CONDITION name cond_dir base node output
   /PRINT
   /FULL
   /MEMBERS
   /ATTRIBUTES/COUNTERS/FLAGS/STATUS/ALL
   /POLY/WIND/MULTI/PATT/COMP/FUNC/ANY
   /[NO]KEEP_MAP
```

CALLING

```
STS=E$SHCO(CV_NAME,CV_DIR,CV_BASE,
              CV_NODE,CV_OUT,I_FULL,I_MEMBERS,
              CV_DETAIL,CV_TYPE,,,,,,,,,,,,,,,,,,,,,
              I_PRINT,I_KEEP_MAP,B_MASK)
```

Argument description

NAME

Type Input CHAR(*) VAR

Name of condition. Wildcards are supported:

* x* *x *x* x*y

Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

DIR

Type Input CHAR(*) VAR

Default directory name

BASE

Type Input CHAR(*) VAR

Default base name

NODE

Type Input CHAR(*) VAR

Default node

OUT

Type Input CHAR(*) VAR

Optional file for output

MEMBERS

Type Input BIN FIXED(15)
Valid values are:

0 Output first and last members of indexed conditions.

1 Output all members of indexed conditions.

FULL

Type Input BIN FIXED(15)
Valid values are:

0 Short output is done.

1 Full output is done.
Switch for full output

DETAIL

Type Input CHAR(*) VAR

/ATTRIBUTES’

/COUNTERS’

/FLAGS’

/STATUS’

/ALL’

Detailness

TYPE

Type Input CHAR(*) VAR

’/WINDOW’ Window condition

’/PATTERN’ Pattern condition

’/COMPLEX’ Complex condition

’/FUNCTION’ Function condition

’/POLYGON’ Polygon condition

'/ANY' Any condition type

Select single condition types

PRINT

Type Input BIN FIXED(15)

Valid values are:

0 No print is done.

1 Print output file on Line printer.

Switch for print output

KEEP_MAP

Type Input BIN FIXED(15)

Valid values are:

0 Deattach of the Data Base.

1 Inhibit unmap of Data Base.

MASK

Type Input BIT(32) ALIGNED

Function

Condition 'name' is searched and its attributes are listed, esp. type, window limits, pattern bits, related conditions, condition table name

Remarks

Module is an action routine

Example

SHOW DIRECTORY

```
SHOW DIRECTORY dir base output
     /[NO]FULL
     /[NO]DIRECTORY
     /[NO]KEEP_MAP
     /PRINT
```

PURPOSE Show Data Elements of a Data Base Directory

PARAMETERS

dir Directory name
required common default

base Data Base name
required common default

output optional output file

/[NO] FULL Show full Directory information
default: '/NOFULL' brief Directory information

/[NO] DIRECTORY Show Master Directory
default: '/NODIRECTORY'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHDI.PPL

Dataset -

EXAMPLE SHO DIR EVE OUTP='EVE.LIS' /FULL
show the full information about the Directory 'EVE'
of the Data Base 'DB' and write the output into the file 'EVE.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING

```
STS = M$ASHDI(CV_DIR,CV_BASE,CV_OUT,
              I_FULL,I_DIRECTORY,I_KEEP_MAP,I_PRINT)
```

ARGUMENTS

CV_DIR I Directory name
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

CV_OUT I Optional file for output
CHAR(*) VAR

I_FULL I Show full Directory information
BIN FIXED(15)

I_DIRECTORY I Show Master Directory
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

I_PRINT I Print output
BIN FIXED(15)

FUNCTION Show Data Elements of a Data Base Directory

REMARKS Module is an action routine.

EXAMPLE -

SHOW DYNAMIC LIST

```
SHOW DYNAMIC LIST dyn_list dyn_type dyn_dir base node output
     /[NO]KEEP_MAP
     /PRINT
```

PURPOSE Show Dynamic List (elements)

PARAMETERS

dyn_list Dynamic List name specification
required common default

dyn_type Type of dynamic sublist
common default:'*'

dyn_dir Default Directory
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

output optional output file
optional

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHDL.PPL

Dataset -

EXAMPLE SHO DYN LIS DYNA1 $DYNAMIC DB OUTP='DYNA1.LIS'
show the full information about the Dynamic List 'DYNA1' in Directory 'DYNAMIC' of the Data Base 'DB' and write the output into the file 'DYNA1.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING

```
STS=M$ASHDL(CV_DYN_LIST,CV_DYN_TYPE,
  CV_DYN_DIR,CV_BASE,CV_NODE,CV_OUT,
  I_KEEP_MAP,I_PRINT)
```

ARGUMENTS

CV_DYN_LIST Dynamic List name specification
CHAR(*) VAR
Input

CV_DYN_TYPE Type of dynamic sublist
CHAR(*) VAR
Input

CV_DYN_DIR Default Directory
CHAR(*) VAR
Input

CV_BASE Default Data Base name
CHAR(*) VAR
Input

CV_NODE Default node name
CHAR(*) VAR
Input

CV_OUT optional file for output
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

I_PRINT Print output
BIN FIXED(15)
Input

FUNCTION Show Dynamic List (elements)

REMARKS Module is an action routine.

EXAMPLE -

SHOW ELEMENT

```
SHOW ELEMENT name dir base node output
    /DECIMAL/HEXADECIMAL/OCTAL/BINARY
    /LONGWORD/WORD/BYTE
    /[NO]DATA
    /[NO]FULL
    /[NO]KEEP_MAP
    /PRINT
```

PURPOSE Show a Data Element descriptor and value

PARAMETERS

name Node::base:[dir]name(i)->type(i)
dir, name, type, and index might be given with wildcards '*'
required common default

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

output optional output file
optional
DECIMAL/HEXADECIMAL/OCTAL/BINARY
: Output radix for Data Types
replaced default:'/DECIMAL'

/LONGWORD/WORD/BYTE output format for atomic Data Type Y
replaced default:'/LONGWORD'

/[NO] DATA Shows all the data of a Data Element
default:'/NODATA'

/[NO] FULL Shows all control information of a Data Element
default:'/NOFULL'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHDE.PPL

Dataset -

EXAMPLE SHO ELEM DB:[EVE]ADAM(17) OUTP='ADAM.LIS' /DATA
show the Data Element with the index '17' of the name array 'ADAM' in Directory 'EVE' of Data Base 'DB' and write the output to the file 'ADAM.LIS' of the default VAX/VMS directory. The data will be shown but only brief control information.
SHO ELEM DB:[EVE]*A*
show all Data Elements of Directory 'EVE' containing an 'A' in their names. The output will be without data in short form.
SHO ELEM DB:[EVE]BETA->*
show all queued Data Elements of Data Element 'BETA' in Directory 'EVE' of Data Base 'DB'. The output will be without data in short form.

Remarks

REMARKS -

Description

CALLING STS=M$ASHDE(CV_NAME,CV_DIR,CV_BASE,
CV_NODE,CV_OUT,CV_RADIX,CV_OUT_Y,
I_DATA,I_FULL,I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_NAME I Node::base:[dir]name(i)->type(i)
CHAR(*) VAR

CV_DIR	I Default Directory CHAR(*) VAR
CV_BASE	I Default Data Base name CHAR(*) VAR
CV_NODE	I Default node name CHAR(*) VAR
CV_OUT	I optional file for output CHAR(*) VAR
CV_RADIX	I Output radix for Data Types CHAR(*) VAR
CV_OUT_Y	I Output format for atomic Data Type Y CHAR(*) VAR
I_DATA	I Show Data Element with all its data BIN FIXED (15)
I_FULL	I Show all Data Element control information BIN FIXED (15)
I_KEEP_MAP	I Inhibit unmap of Data Base BIN FIXED (15)
I_PRINT	I Print output BIN FIXED(15)
FUNCTION	Show a Data Element descriptor and value
REMARKS	Module is an action routine.
EXAMPLE	-

SHOW GOOSY STATUS

```
SHOW GOOSY STATUS environment p1 p2 p3
                /$TMR
                /$ANL
```

PURPOSE	GOOSY Status report
ARGUMENTS	
environment	Optional environment
p1	Process name to be monitored
p2	Process name to be monitored
p3	Process name to be monitored
/$TMR	Monitor $TMR of environment
/$ANL	Monitor $ANL of environment
Information	See HELP MSTATUS, HELP GOOCONTROL
EXAMPLE	GOOSY> SHOW G ST P1=GN_HGE____$TMR $ GSTAT P1=GN_HGE____$TMR $ GSTAT SUSI /$TMR/$ANL
Action rout.	M$GOOST
Author	H.G.Essel

Remarks

The same action can be invoced by the DCL symbols

```
   MSTATUS SHO GOOSY ST proc_1 proc_2 proc_3
      or GSTATUS proc_1 proc_2 proc_3
```

File name	M$GOOST.PPL

Description

CALLING

```
STAT=M$GOOST(CV_environ
            ,CV_process_1,CV_process_2,CV_process_3
            ,I_tmr,I_anl)
```

ARGUMENTS -

CV_environ I: Optional environment name

CV_process_1 I: Process name to be monitored

CV_process_2 I: Process name to be monitored

CV_process_3 I: Process name to be monitored

I_tmr I: if 1, monitor $TMR of environment

I_anl I: if 1, monitor $ANL of environment

FUNCTION Two types of processes can be monitored:

TMR transport manager
ANL analysis programs

Presently a maximum of three processes can be monitored. Only one of them can be a TMR. The processes must run on the same node in the same group. The logical name GOO$CONTROL must be defined as the name of a data base, which must be created before starting the TMR and analysis programs. This should be done using the GOOCONTROL command, which should be included in the LOGIN.COM. The TMR and analysis programs create data elements in this data base and deposit their control information. The names of the data elements are the process names.

REMARKS -

EXAMPLE -

SHOW HOME_BLOCK

```
SHOW HOME_BLOCK base output
   /PRINT
   /BITMAP
```

PURPOSE Show the Home Block of a Data Base

PARAMETERS

base Data Base name
  required common default

output optional output file
  optional

/BITMAP output bitmap

/PRINT Print output

Caller M$SHMCM

Author M. Richter

File name M$ASHHB.PPL

Dataset -

EXAMPLE SHO HOM DB OUTP='HBDB.LIS'
  show the Home Block Information of Data Base 'DB'
and write it into the file 'HBDB.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING STS=M$ASHHB(CV_BASE,CV_OUT,I_BITMAP,I_PRINT)

ARGUMENTS

CV_BASE I Data Base name
CHAR(*) VAR

CV_OUT I optional file for output
CHAR(*) VAR

I_BITMAP Output bitmap
BIN FIXED(15)

I_PRINT I Print output
BIN FIXED(15)

FUNCTION Show the Home Block of a Data Base

REMARKS Module is an action routine.

EXAMPLE -

SHOW LINK

```
SHOW LINK link_from dir base node
     output
     /IN/OUT/ALL
     /MATCH/TREE
     /[NO]KEEP_MAP
     /PRINT
```

PURPOSE Show Data Element link(s)

PARAMETERS

link_from Source Data Element name specification
required common default

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

output optional output file
optional

/IN/OUT/ALL In, out, all links
replaced default:'/ALL'

/MATCH/TREE Matching or total link tree
default:'/MATCH'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHLI.PPL

Dataset -

EXAMPLE SHO LINK DB:[EVE]ADAM /OUT OUT='ADAM.LIS'
show all outgoing links from Data Element 'ADAM' in Directory 'EVE' of Data Base 'DB' and write the output into the file 'ADAM.LIS' of the VAX/VMS default directory.

Remarks

REMARKS -

Description

CALLING STS=M$ASHLI(CV_LINK_FROM,CV_DIR,
CV_BASE,CV_NODE,CV_OUT,CV_SELECT,CV_ACTION,
I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_LINK_FROM I Source Data Element name specification
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

CV_OUT I optional file for output
CHAR(*) VAR

CV_SELECT I In, out, all links
CHAR(*) VAR

CV_ACTION I Matching or total link tree
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

I_PRINT I Print output
BIN FIXED(15)

FUNCTION Show Data Element link(s)

REMARKS Module is an action routine.

EXAMPLE -

SHOW MEMBER

```
SHOW MEMBER mem_spec dir base node output
    /[NO]KEEP_MAP
    /PRINT
```

PURPOSE Show value of a Data Element member or a full Data Element member array (wildcarded).

PARAMETERS

mem_spec Node::base:[dir]name(i)->type(i)
required

dir Default Directory
common default:'DATA'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

cv_out optional file for output
optional

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output
BIN FIXED(15)

Caller M$DMCMD

Author M. Richter

File name M$ASHME.PPL

Dataset -

EXAMPLE SHO MEM DB:[EVE]ADAM(12).KAIN OUT='ADAM.LIS'
show the member 'KAIN' of the Data Element with the index '12' of the name array 'ADAM' in the Directory 'EVE' of the Data Base 'DB' and write the output into the file 'ADAM.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING STS=M$ASHME(CV_ELEMENT,CV_DIR,CV_BASE,
CV_NODE,CV_OUT,I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_ELEMENT I Node::base:[dir]name(i)->type(i)
CHAR(*) VAR

CV_DIR I Default Directory
CHAR(*) VAR

CV_BASE I Default Data Base name
CHAR(*) VAR

CV_NODE I Default node name
CHAR(*) VAR

CV_OUT I Optional output file
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

I_PRINT I Print output file
BIN FIXED (15)

FUNCTION Show value of a Data Element member

REMARKS Module is an action routine.

EXAMPLE -

SHOW PICTURE

```
SHOW PICTURE picture output pic_dir base node
             /PRINT
             /FULL
             /DATA
             /[NO]KEEP_MAP
```

PURPOSE Show picture information

PARAMETERS

picture Name of picture.

output Output file

pic_dir Default picture Directory

base Default Data Base name

node Default node name

/PRINT Output file is printed

/FULL Short information about all frames is shown.

/DATA Detailed information about all frames is shown.

/KEEP_MAP Inhibit the unmapping of the whole Data Base.

Caller MDBM, MGOODBM

Author W. Spreng

Example

1.) SHOW PICTURE pic [*]*
All pictures in all existing Directories are listed.

Remarks

File name D$SHPI.PPL

Created by GOO$DE:E$SHECM.PPL

Description

CALLING

```
STS=D$SHPI(CV_picture,CV_output,CV_pic_dir,
            CV_base,CV_node,I_full,I_data,I_keep)
```

COMMAND

```
SHOW PICTURE picture output pic_dir base node
               /PRINT
               /FULL
               /DATA
               /[NO]KEEP_MAP
```

PICTURE

Routine arg. Input CHAR(*) VAR

Command par. String required

Name of picture for which informations should be shown. Wildcards in Directory and picture name are supported.

OUTPUT

Routine arg. Input CHAR(*) VAR

Command par. String

File in which the output should performed. The output will be printed with the /PRINT switch.

PIC_DIR

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=$PICTURE

Default picture Directory.

BASE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command par. String global replaceable default=*

Default node name.

PRINT

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

The generated output is printed on the default print queue.

FULL

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

If set information for each frame the spectrum or scatterplot parameter are listed.

DATA

Routine arg. BIN FIXED(15) valid values are 0 and 1

Command par. Switch

If set the whole frame parameter, e.g specified and last window, display modes of the axis, all defined overlays etc., are listed.

KEEP_MAP

Routine arg. BIN FIXED(15) valid values 0 and 1

Command par. Switch default = /KEEP_MAP

If set the mapping context will be kept. If /NOKEEP_MAP specified the database will be dettached after the command completion.

Function

Information about the picture frames are displayed on the terminal. If '/FULL' is specified the spectra and scatterplot parameters for all frames are listed. If additionally '/DATA' is specified all user specified picture data are shown.

SHOW POLYGON

```
SHOW POLYGON name poly_dir base node output
  /PRINT
  /[NO]FULL
  /[NO]DATA
  /[NO]KEEP_MAP
```

PURPOSE show attributes of a polygon

PARAMETERS

name name of polygon. Wildcards are supported:
* x* *x *x* x*y

poly_dir default Directory
replace default='$PICTURE'

base default Data Base
replace default='DB'

node default node
replace default='E'

output optional output file

/PRINT print output

/[NO] FULL show full header

/[NO] DATA show polygon points

/[NO] KEEP_MAP inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE SHOW POLY emil /DATA
shows points of polygon emil

Caller E$SHECM

Author H.G.Essel

File name E$ASHPO.PPL

Dataset -

Remarks

REMARKS -

Description

CALLING

```
STS=E$ASHPO(CV_NAME,CV_DIR,CV_BASE,
    CV_NODE,CV_OUT,I_FULL,I_DATA,
    I_PRINT,I_KEEP_MAP,B_MASK)
```

ARGUMENTS

CV_NAME name of polygon. Wildcards are supported:
* x* *x *x* x*y
CHAR(*) VAR
Input

CV_DIR default directory name
CHAR(*) VAR
Input

CV_BASE default base name
CHAR(*) VAR
Input

CV_NODE default node
CHAR(*) VAR
Input
optional file for output
CHAR(*) VAR
Input

I_FULL switch for full output
BIN FIXED(15)
Input

I_DATA switch to show polygon points
BIN FIXED(15)
Input

I_PRINT
Print output
BIN FIXED(15)
Input

I_KEEP_MAP
Inhibit unmap of Data Base
BIN FIXED (15)
Input

B_MASK
mask which params were specified
BIT(32) ALIGNED
Output

FUNCTION polygon 'name' is searched and its attributes are listed

REMARKS -

EXAMPLE -

SHOW POOL

```
SHOW POOL pool base output
    /[NO]FULL
    /[NO]DIRECTORY
    /[NO]KEEP_MAP
    /PRINT
```

PURPOSE Show Areas of a Data Base Pool

The name, size, filling level, cluster size, and number of fragments are shown for each Area. The Pool Directory can be shown with /DIRECTORY.

PARAMETERS

pool Pool name, wild cards are allowed.

The Pool name will be ignored for /DIRECTORY.

required common default

base Data Base name

required common default

output optional output file

/[NO] FULL Show the full information of all Areas of a Pool.

Together with the /DIRECTORY option all entries in the Pool Directory are listed.

default:'/NOFULL'

/[NO] DIRECTORY Show information of the Pool Directory

Together with the /FULL option all entries in the Pool Directory are listed. With the /DIRECTORY any given Pool name will be ignored.

default:'/NODIRECTORY'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base

default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHPO.PPL

Dataset -

EXAMPLE SHO POO ADAM OUT='ADAM.LIS'
show the Areas of the Pool 'ADAM' of the Data Base 'DB' and write the output into the file 'ADAM.LIS' of the default VAX/VMS directory.

Remarks

REMARKS -

Description

CALLING

```
STS=M$ASHPO(CV_POOL,CV_BASE,CV_OUT,
               I_FULL,I_DIRECTORY,I_KEEP_MAP,I_PRINT)
```

ARGUMENTS

CV_POOL I Pool name
CHAR(*) VAR

CV_BASE I Data Base name
CHAR(*) VAR

CV_OUT I optional file for output
CHAR(*) VAR

I_FULL Show full Pool and Area information
BIN FIXED(15)

I_DIRECTORY Show Pool Directory information
BIN FIXED(15)

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

I_PRINT I Print output
BIN FIXED(15)

FUNCTION Show Areas of a Data Base Pool. Optional full Area information will be shown. The Pool Directory can be shown by the option /DIRECTORY.

REMARKS Module is an action routine.

EXAMPLE -

SHOW SPECTRUM

```
SHOW SPECTRUM name spec_dir base node output width
  /PRINT
  /ATTRIBUTES/DATA/ALL/STATUS
  /FULL
  /MEMBERS
  /LOG
  /INTEGRAL
  /ZERO
  /CAMAC
  /[NO]KEEP_MAP
```

PURPOSE show spectra

PARAMETERS

name String default: '*'
name of spectrum. Wildcards are supported:
* x* *x *x* x*y Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

spec_dir String replace default: '$SPECTRUM'
Name of Spectrum directory.

base String replace default: 'DB'
Name of Data Base.

node String replace default: '*'
Name of Node.

output String default: none
optional output file

width Integer global replace default: 80
Line length for histogram (80 or 132)

/PRINT Switch default: none
Print output

/WHAT Set replace default: /ATTRIBUTES
/ATTRIBUTES :Display attributes of spectra
/DATA :Display data
/STATUS :calls SHOW TAB/SP
/ALL :Display all

/FULL Switch default: none
Display full spectrum information

/MEMBERS Switch default: none
Display information about all spectrum members.

/LOG Switch default: none
Display data in logarithmic format

/INTEGRAL Switch default: none
Display channel contents integral

/ZERO Switch default: none
Display channel contents including zero channels

/CAMAC Switch default: none
Display CAMAC spectra only

/[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

FUNCTION The specified spectrum is accessed and various
parts of it are displayed on the terminal

EXAMPLE SHOW SPEC OTTO /FULL
shows otto and all queued data elements

Caller E$SHECM

Author K.WINKELMANN

File name E$SHSP.PPL

Dataset GOO$DE:E$SHSP.PPL

Function

The specified spectrum is accessed and various
parts of it are displayed on the terminal

Example

SHOW SPEC OTTO

Remarks

REMARKS Module is an action routine.

Created by E$SHECM.PPL

Description

CALLING

```
STS=E$SHSP(CV_NAME,CV_DIR,CV_BASE,CV_NODE,CV_OUT,
           L_WIDTH,I_PRINT,CV_DETAIL,I_FULL,
           I_MEMBERS,I_LOG,I_INTEGRAL,I_ZERO,
           I_CAMAC,I_KEEP_MAP,B_MASK)
```

COMMAND

```
SHOW SPECTRUM name spec_dir base node output width
  /PRINT
  /ATTRIBUTES/DATA/ALL/STATUS
  /FULL
  /MEMBERS
  /LOG
  /INTEGRAL
  /ZERO
  /CAMAC
  /[NO]KEEP_MAP
```

Argument description

NAME

Type Input CHAR(*) VAR
Name of spectrum. Wildcards are supported:
* x* *x *x* x*y Name arrays are supported. Index may be (*). This is assumed, if name is wildcarded.

DIR

Type Input CHAR(*) VAR
Default directory

BASE

Type Input CHAR(*) VAR

Default data base

NODE

Type Input CHAR(*) VAR

Default node

OUTPUT

Type Input CHAR(*) VAR

Optional file for output

WITDH

Type Input BIN FIXED(31)

Line size for histogram (80 or 132).

PRINT

Type Input BIN FIXED(15)

valid values are:

No print of output file is done Print output file on line printer

WHAT

Type Input CHAR(*) VAR

valid values are:

Display attributes of the spectrum Display data of the spectrum Display attributes,all and status of the spectrum Display status of spectrum

FULL

Type Input BIN FIXED(15)

valid values are:

Display only short info about the spectrum: Spectrumname,spectrumtype,limits,bins,bits
Display full information about the spectrum: Spectrumname,spectrumtype,Database,Dire
Dimensions, Index in analyzetable, Creationdate Lettering of X-Axis,Lettering of Y-Axis, Lower limit, upper limit, Number of bins, Binsize, Factor, Offset, Underflow - and Overflow counts, Underflow and Overflow contents, freeze and executed bit

MEMBERS

Type Input BIN FIXED(15)
valid values are:
Display only first and last indexed spectrum: Display information about all spectrum members:

LOG

Type Input BIN FIXED(15)
Valid values are:
Display decimal data Display logarithmic data

INTEGRAL

Type Input BIN FIXED(15)
Valid values are:
Display channel contents Display integral channel contents

ZERO

Type Input BIN FIXED(15)
Valid values are:
Display channel contents without zero's Display channel contents with zero's

CAMAC

Type Input BIN FIXED(15)
Valid values are:
Display all spectra Display CAMAC spectra only

KEEP_MAP

Type Input BIN FIXED(15)
Valid values are:
Unmap Data Base Inhibit unmap of the Data Base

B_MASK

Type Output BIT(32) ALIGNED

Function

The specified spectrum is accessed and various parts of it are displayed on the terminal. Output can be interrupted by ^ Z.

Remarks

Module is an action routine.

Example

SHOW TABLE

```
SHOW TABLE name table tab_dir base node output
  /CONDITION /SPECTRUM /ALL
  /CONTENT /COUNTS
  /PRINT
  /[NO]KEEP_MAP
```

PURPOSE show flag tables from the analysis

PARAMETERS

NAME String replace default: '*'
Name of items (conditions or spectra)

TABLE String replace default: '*'
Name of table (may be omitted)

TAB_DIR String replace default: '$ANLTABS'
Name of Directory

BASE String replace default: 'DB'
Name of Data Base

NODE String replace default: '*'
Name of node

OUTPUT String replace default: ”
Optional output file

TYPE Set replace default: /ALL /CONDITION : /SPECTRUM : /ALL : Kind of table

CONT Set replace default: /COUNTS /CONTENT : /COUNTS : Display spectrum content or counts

[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

/PRINT Switch default: none
Print output

FUNCTION The contents of the tables (flags for freeze and executed (spectra) or freeze,executed,result and result preset (conditions) are shown.
In addition, condition counters and spectrum
contents and overflows are displayed

EXAMPLE SHO TA /SP

Caller E$SHECM

Author K.Winkelmann

File name GOO$DE:E$SHANT.PPL

Dataset -

NAME

Type String replace default: '*'
Name spec of conditions or spectra (wildcard)

TABLE

Type String replace default: '*'
Name of analysis table, NYI,
in the moment default name ANSP and ANCO from directory $ANLTABS are taken

TAB_DIR

Type String replace default: '$ANLTABS'
Default name of Directory

BASE

Type String replace default: 'DB'
Default name of Base

NODE

Type String replace default: '*'
Default name of Node

OUTPUT

Type String replace default: ”
Optional file for output

TYPE

Type Set replace default: /ALL
Valid values are:

'/COND' ANCO is shown

'/SPECTRUM' ANSP is show

'/ALL' indicates that both will be shown.

CONT

Type Set replace default: /COUNTS
Valid values are:

'/CONTENT'

'/COUNTS'

Spectrum contents

KEEP_MAP

Type Switch replace default: /KEEP_MAP
Valid values are:

[**NO] KEEP_MAP** Inhibit unmap of the Data Base.

PRINT

Type Switch default: none
Valid values:

'/PRINT'1 print output file . Print output file on lineprinter.

MASK

Type Input BIT(32) ALIGNED
Mask which params were specified

Function

The contents of the tables (flags for freeze and executed (spectra) or freeze,executed,result and result preset (conditions) are shown.
In addition, condition counters and spectrum
contents and overflows are displayed

Example

```
SHO TA /SP
```

Remarks

REMARKS Module is an action routine.

Created by GOO$DE:E$SHECM.PPl

Description

Argument description

SPEC

Type Input CHAR(*) VAR
Name spec of conditions or spectra (wildcard)

NAME

Type Input CHAR(*) VAR
Name of analysis table, NYI,
in the moment default name ANSP and ANCO from directory $ANLTABS are taken

DIR

Type Input CHAR(*) VAR
Default name of Directory

BASE

Type Input CHAR(*) VAR
Default name of Base

NODE

Type Input CHAR(*) VAR
Default name of Node

OUT

Type Input CHAR(*) VAR
Optional file for output

KIND

Type Input CHAR(*) VAR

'/COND' ANCO is shown

'/SPECTR' ANSP is shown

'/ALL' indicates that both will be shown.

CONT

Type Input CHAR(*) VAR
Valid values are:

'/CONTENT'

'/COUNTS'

Spectrum contents

KEEP_MAP

Type Input BIN FIXED(15)
Valid values are:

0 Deattach the data base.

1 Inhibit unmap of the Data Base.

PRINT

Type Input BIN FIXED(15)
Valid values:

0 no print of output file is done.

1 print output file . Print output file on lineprinter.

MASK

Type	Input BIT(32) ALIGNED Mask which params were specified

Function

The contents of the tables (flags for freeze and executed (spectra) or freeze,executed,result and result preset (conditions) are shown.

In addition, condition counters and spectrum contents and overflows are displayed

Remarks

Module is an action routine.

Example

SHOW TREE

```
SHOW TREE base output
    /[NO]KEEP_MAP
    /PRINT
```

PURPOSE Show indices of a Data Base Pool Directory
For test purpose only.

PARAMETERS

base Data Base name required common default

output optional output file
optional

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

/PRINT Print output

Caller M$DMCMD

Author M. Richter

File name M$ASHTR.PPL

Dataset -

EXAMPLE SHO TREE DB
show the name tree of the Pool Directory of the
Data Base 'DB'.

Remarks

REMARKS -

Description

CALLING STS=M$ASHTR(CV_BASE,CV_OUT,I_KEEP_MAP,I_PRINT)

ARGUMENTS

CV_BASE I Data Base name
CHAR(*) VAR

CV_OUT I optional file for output
CHAR(*) VAR

I_KEEP_MAP I Inhibit unmap of Data Base
BIN FIXED (15)

I_PRINT I Print output
BIN FIXED(15)

FUNCTION Show indices of a Data Base Pool Directory

REMARKS Module is an action routine.

EXAMPLE -

SHOW TYPE

SHOW TYPE type base output

PURPOSE Show a Type descriptor. Wild card are allowed.

PARAMETERS

type required string default: ” type descriptor name

base required string global replace default: ” Data Base name

output string default: ” optional output file

/DATA switch default: /PLI Output of Type descriptor as a PL-I Structure or only the data.

/[NO] KEEP_MAP switch default: /KEEPMAP Inhibit the unmap (detach) of the whole Data Base

/PRINT switch default: Print output on Line Printer.

Caller M$SHMCM

Author M. Richter

Example

SHO TYP $SPECTRUM DB OUT='SPEC.LIS'
show the Data Element Type '$SPECTRUM' of the Data
Base 'DB' and write the output into the file 'SPEC.LIS' of the default VAX/VMS directory.

Remarks

File name M$ASHTY.PPL

Created by GOO$DM:M$SHMCM.PPL

Description

CALLING STS = M$ASHTY(CV_TYPE,CV_BASE,CV_OUT,CV_DATATYPE,
I_KEEP_MAP,I_PRINT)

COMMAND SHOW TYPE type base output
/DATA/PLISTRUC
/[NO]KEEP_MAP
/PRINT
Argument / Parameter description

TYPE

Routine arg. Input CHAR(*) VAR

Command par. required string default :"
Type descriptor name. Wild cards are allowed

BASE

Routine arg. Input CHAR(*) VAR

Command par. required string default :"
Data Base name

OUTPUT

Routine arg. Input CHAR(*) VAR

Command par. required string default :"
Optional file for output

/DATA

Routine arg. Input CHAR(*) VAR

Command par. required string default :"
Output of Type descr. like a Data Element or a PL/I-Structure

/KEEP_MAP

Routine arg. Input CHAR(*) VAR

Command par. required string default :”
Inhibit unmap of Data Base

/PRINT

Routine arg. Input CHAR(*) VAR

Command par. required string default :”
Print output

Function

Show a Type descriptor in directory $TYPE in Data Base ’Base’.

Remarks

Module is an action routine.

Example

```
STS=M$ASHTY(’*SPD*’,’DB’,”,’/PLI’,1,0)
```

START MR2000

START MR2000 branch crate station
/INITIALIZE

PURPOSE	Start/initialize MR2000.
PARAMETERS	
branch	Integer replace default: 0 Number of CAMAC branch.
crate	Integer replace default: 1 Number of crate on branch.
station	Integer replace default: 2 Station number of MR2000 in crate.
/INITIALIZE	switch default: ” Initialize MR2000 before start.
Caller	mdbm
Author	H.G.Essel

Example

```
STA MR2000 0,3,20 /INIT
```

Remarks

File name	E$AGOMR.PPL
Created by	GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$AGOMR(I_branch,I_crate,I_station,
I_initialize)

COMMAND START MR2000 branch crate station
/INITIALIZE
Argument description

BRANCH

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 0
Number of CAMAC branch.

CRATE

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 1
CAMAC Crate number.

STATION

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 2
Station number of MR2000 in crate

INITIALIZE

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ”
Initialize MR2000 before start.

Function

Module I$CMR2G is called to start the MR2000.
If /INITIALIZE is specified, I$CMR2I is called
before.

STOP MR2000

STOP MR2000 branch crate station

PURPOSE Stop MR2000.

PARAMETERS

branch Integer replace default: 0
Number of CAMAC branch.

crate Integer replace default: 1
Number of crate on branch.

station Integer replace default: 2
Station number of MR2000 in crate.

Caller mdbm

Author H.G.Essel

Example

```
STOP MR2000 0,3,20
```

Remarks

File name E$AHAMR.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$AHAMR(I_branch,I_crate,I_station)

COMMAND STOP MR2000 branch crate station Argument description

BRANCH

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 0
Number of CAMAC branch.

CRATE

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 1
CAMAC Crate number.

STATION

Routine arg. Input BIN FIXED(15)

Command par. integer replace default: 2
Station number of MR2000 in crate

Function

Module I$CMR2H is called to stop the MR2000.

SUMUP SPECTRUM

```
SUMUP SPECTRUM spectrum window file spec_dir base node
    /CHAN/CALIB [=CALIBR]
    /[NO]OUTPUT
    /[NO]APPEND
    /[NO]KEEP_MAP
```

PURPOSE Integrate specified window

PARAMETERS

spectrum name of spectrum

window Limits of integration window

file File for the output of the results.

CALIBR Specifies the unit in which the coordinates are given.

/CHAN	Original spectrum units.
/CALIB	Calibrated units.

/[NO] OUTPUT Output occus additionally onto the specified file.

/[NO] APPEND Results are appended to an existing file.

/[NO] KEEP_MAP Inhibit the unmap of the Data Base.

Caller MDBM,MGOODBM

Author W. Spreng

Example

1.) SUMUP SPECTRUM a 100,400
Spectrum A is integrated from channel 100 to 400
2.) SUMUP SPECTRUM a 100,400/calib

Spectrum A is integrated from the calibrated value 100 to 400

3.) SUMUP SPECTRUM a 100,400,200,500

Window 100,400 and 200,500 are integrated. 3.) SUMUP SPECTRUM a 100,400,200,500 /OUTPUT

Output is directed onto the file GOOSY_RESULTS.

Remarks

File name E$SUMUP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$SUMSP(CV_spectrum,CV_window,CV_file, CV_spec_dir, CV_base, CV_node, CV_calibr, I_output, I_append, I_keep_map)

COMMAND

```
SUMUP SPECTRUM spectrum window file spec_dir base node
    /CHAN/CALIB [=CALIBR]
    /[NO]OUTPUT
    /[NO]APPEND
    /[NO]KEEP_MAP
```

SPECTRUM

Routine arg. CHAR(*) VAR

Command arg. String required

Name of spectrum which should be summed up.

WINDOW

Routine arg. Input CHAR(*) VAR

Command par. String

Window which should be used as integration range. Any valid display window specification is allowed (see the GOOSY Display Manual):

```
  (xmin,xmax) for a one dimensional window
        * for the whole spectrum range
```

It is possible to define up to 30 windows which should be integrated by:

```
   (xmin,xmax,xmin,xmax,...)
```

If values are omitted they are replaced by the corresponding spectrum minimum or maximum limit!

The upper window limits are exclusive! E.g. for a one dimensional analog spectrum of binsize 1 the window (1,3) yields to an integration of two bins! Although value 3 belongs to the third bin.

FILE

Routine arg. CHAR(*) VAR

Command arg. String replacable default=GOOSY_RESULT.LOG

Name of a file onto which the results should be written. To direct the outputs to that file the /OUTPUT switch is required!

SPEC_DIR

Routine arg. CHAR(*) VAR

Command arg. String global replaceable default=$SPECTRUM

Default Directory name for spectra.

BASE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=DB

Default Data Base name.

NODE

Routine arg. CHAR(*) VAR

Command arg. String global replacable default=*

Node name for Data Base section file.

CALIBR

Routine arg. Input CHAR(*) VAR

Command par. Set default="/CHAN"

If the spectrum in the selected frame is calibrated the input of the coordinates can occur in calibrated or uncalibrated units:

/CHANN	Coordinates of point are given in spectrum units.
/CALIB	Coordinates of point are in calibrated units

If the coordinates are specified via cursor input the units are known and this switch has no effect!

OUTPUT

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch, default = /NOOUTPUT

Write results into the specified file.

/OUTPUT	The output is directed onto the file.
/NOOUTPUT	results are only written to terminal and into the session logfile.

APPEND

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch, default=/APPEND

Append the output to an existing file.

/APPEND	output is appended to an existing file.
/NOAPPEND	A new output file is created.

KEEP_MAP

Routine arg. Input BIN FIXED(15); valid input are 0 and 1

Command par. Switch

Inhibit the unmap of the Data Base.

/KEEP_MAP	Data Base will not be dettached after command termination.
/NOKEEP_MAP	Dettach the Data Base in any case.

FUNCTION

The spectrum in the specified frame is integrated within the specified limits. The window could be specified in calibrated units or in channels, depending on the given switch (/CHAN or /CALIB). If calibrated units are used but the spectrum in the specified frame is not connected to a calibration an error is signaled.

UNFREEZE CONDITION

**UNFREEZE CONDITION name cond_dir base node
/[NO]KEEP_MAP**

PURPOSE unfreeze a condition, enable the execution of a condition

PARAMETERS

name name of condition
required

cond_dir default Directory
replace default:'$COND'

base default Data Base
replace default:'DB'

node default node
replace default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$UNFCO.PPL

Dataset -

Remarks

REMARKS action routine

Description

CALLING STS=E$UNFCO(CV_NAME,CV_DIR,CV_BASE,
CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_NAME default directory
CHAR(*) VAR
Input

CV_DIR default directory
CHAR(*) VAR
Input

CV_BASE default base
CHAR(*) VAR
Input

CV_NODE default node
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

FUNCTION The freeze bit in the analysis flag tables ([$ANLTABS]ANCO is the default) in which the specified condition lies, is reset. This enables any subsequent execution of the condition by the analysis program or any dynamic list. The result bit and the preset bit are set to zero.

REMARKS action routine

EXAMPLE -

UNFREEZE SPECTRUM

```
UNFREEZE SPECTRUM name spec_dir base node
     /[NO]KEEP_MAP
```

PURPOSE unfreeze a spectrum, enable the accumulation

PARAMETERS

name name of spectrum
required

spec_dir default Directory
replace default:'$COND'

base default Data Base
replace default:'DB'

node default node
replace default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

EXAMPLE -

Caller E$DECMD

Author K.Winkelmann

File name GOO$DE:E$UNFSP.PPL

Dataset -

Remarks

REMARKS action routine

Description

CALLING STS=E$UNFSP(CV_NAME,CV_DIR,CV_BASE, CV_NODE,I_KEEP_MAP)

ARGUMENTS

CV_NAME default spectrum
CHAR(*) VAR
Input

CV_DIR default directory
CHAR(*) VAR
Input

CV_BASE default base
CHAR(*) VAR
Input

CV_NODE default node
CHAR(*) VAR
Input

I_KEEP_MAP Inhibit unmap of Data Base
BIN FIXED (15)
Input

FUNCTION The freeze bit in the analysis flag tables ([$ANLTABS]ANCO is the default) in which the specified spectrum lies, is reset. This enables any subsequent accumulation (execution) of the spectrum by the analysis program or any dynamic list.

REMARKS action routine

EXAMPLE -

UNPROTECT SPECTRUM

```
UNPROTECT SPECTRUM name spec_dir base node
    /LOG
    /[NO]KEEP_MAP
```

PURPOSE Unprotect one (or all) spectrum

PARAMETERS

NAME required string global replace default: ”
Name of Spectrum (wildcard) to be unprotected
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.

SPEC_DIR String global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE String global replace default: 'DB'
Name of Data Base

NODE String global replace default: 'E'
Name of node

/LOG Switch default:
Output names of unprotected spectra.

[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller mdbm

Author H.G.Essel

Example

UNPROT SP A(3,9) unprotect one spectrum
UNPROT SP A(3) unprotect one spectrum
UNPROT SP A(3:9) unprotect seven spectra
UNPROT SP A(*) unprotect all members
UNPROT SP A* unprotect all members
UNPROT SP A unprotect all members

Remarks

File name E$UPROSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$UPROSP(CV_NAME,CV_SPEC_DIR,CV_BASE,CV_NODE,
I_LOG,I_KEEP_MAP)

COMMAND UNPROTECT SPECTRUM name spec_dir base node
/LOG
/[NO]KEEP_MAP
Argument description

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ''
Name of Spectrum (wildcard) to be unprotected
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.
Arrays without index are unprotected totally.
For one dimensional arrays a range may be
specified like x(3:5). No wildcard is allowed in
this case. Single members of one and twodimensional
arrays may be unprotected by specifying the index:
X(7) or Y(4,5).

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Name of Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Name of Node

LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Output unprotected spectrum names.

KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: ''
Inhibit the unmap (detach) of the Data Base

Function

The spectrum will be cleared by CLEAR command.

UPDATE BASE

UPDATE BASE base

PURPOSE Update a Data Base, write all pages to Section File

PARAMETERS

BASE Name of the Data Base
common required default

EXAMPLE UPD DA DB

Caller M$DMCMD

Author M. Richter

File name M$UPDB.PPL

Dataset -

Remarks

REMARKS All pages of the Data Base will be written back into its Section File, even if other processes access the same Data Base at the same time.

Description

CALLING STS=M$UPDB(cv_db_name)

ARGUMENTS

cv_db_name I Name of the Data Base to be updated
CHARACTER(*) VAR
Input

FUNCTION The whole Data Base 'cv_db_name' will be mapped and then written back into the Section File. This update will be forced by the procedure even if other processes are using the Data Base at the same time.

REMARKS Module is an action routine.
The Data Base must be mounted before the call.

EXAMPLE -

UPDATE DYNAMIC LIST

UPDATE DYNAMIC LIST dyn_list dyn_dir base node
/[NO]KEEP_MAP

PURPOSE Update a Dynamic List

PARAMETERS

dyn_list Dynamic List name specification
required common default

dyn_dir Default Directory
common default:'$DYNAMIC'

base Default Data Base name
common default:'DB'

node Default node name
common default:'E'

/[NO] KEEP_MAP Inhibit the unmap (detach) of the whole Data Base
default:'/KEEP_MAP'

Caller M$DMCMD

Author H.G.Essel

File name M$AUPDL.PPL

Dataset -

EXAMPLE UPDAT DYN LI DYNA_1 $DYNAMIC
update the Dynamic List 'DYNA_1' of the Directory '$DYNAMIC' in the Data Base last used.

Remarks

REMARKS -

Description

CALLING STS=M$AUPDL(CV_dyn_list,CV_dyn_dir,CV_base,CV_node,
I_keep_map)

ARGUMENTS

CV_dyn_list I Dynamic List name specification
CHAR (*) VAR

CV_dyn_dir I Default Directory
CHAR (*) VAR

CV_base I Default Data Base name
CHAR (*) VAR

CV_node I Default node name
CHAR (*) VAR

I_keep_map I Inhibit unmap of Data Base
BIN FIXED (15)

FUNCTION Update a Dynamic List

REMARKS Module is an action routine.

EXAMPLE -

WRITE CAMAC SPECTRUM

WRITE CAMAC SPECTRUM name spec_dir base node
/ADD
/LOG
/[NO]KEEP_MAP

PURPOSE Write spectrum data from GOOSY spectrum into MR2000

PARAMETERS

NAME required string global replace default: ”
Name of Spectrum (wildcard) to be copied
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index.

SPEC_DIR String global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE String global replace default: 'DB'
Name of Data Base

NODE String global replace default: 'E'
Name of node

/LOG Switch default: ”
Output list of copied spectra

/ADD Switch default: ”
Add spectrum channel contents rather than overwrite.

[NO] KEEP_MAP Switch default: /KEEP_MAP
Inhibit the unmap (detach) of the whole Data Base

Caller mdbm

Author H.G.Essel

Example

WRITE CA SP A(1,2)
WRITE CA SP A(*)
WRITE CA SP A* /ADD
WRITE CA SP A (copy first spectrum only)

Remarks

File name E$AWCSP.PPL

Created by GOO$DE:E$DECMD.PPL

Description

CALLING STS=E$AWCSP(CV_NAME,CV_SPEC_DIR,CV_BASE,CV_NODE,
I_LOG,I_ADD,I_KEEP_MAP)

COMMAND WRITE CAMAC SPECTRUM name spec_dir base node
/ADD
/LOG
/[NO]KEEP_MAP
Argument description

NAME

Routine arg. Input CHAR(*) VAR

Command par. required string global replace default: ”
Name of Spectrum (wildcard) to be copied
Wildcards are supported in name as:
* x* *x *x* x*y
One asterisk is supported for index expression:
a(*)
A Wildcard in name defaults to a wildcard in index

SPEC_DIR

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: '$SPECTRUM'
Name of Spectrum directory

BASE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'DB'
Name of Data Base

NODE

Routine arg. Input CHAR(*) VAR

Command par. string global replace default: 'E'
Name of Node

LOG

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: "
Output list of copied spectra

ADD

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: "
Instead of overwrite, add spectrum channles to MR2000 channels.

KEEP_MAP

Routine arg. Input BIN FIXED(15) valid values 0 or 1

Command par. switch default: "
Inhibit the unmap (detach) of the Data Base

Function

The data part of MR2000 is writte from specified * GOOSY spectrum. The range in the MR2000 is defined in the spectrum. Channel contents may be overwritten (default) or added (/ADD).

Contents

1 **GOOSY Commands** 1
CALCULATE SPECTRUM . . . 2
CALIBRATE SPECTRUM . . . 7
CLEAR CAMAC SPECTRUM . . . 10
CLEAR CONDITION COUNTER . . . 14
CLEAR ELEMENT . . . 18
CLEAR PICTURE . . . 21
CLEAR SPECTRUM . . . 25
COMPRESS BASE . . . 28
CONVERT BASE . . . 30
COPY BASE . . . 32
COPY CONDITION . . . 35
COPY ELEMENT . . . 41
COPY MEMBER . . . 46
COPY Polygon . . . 50
COPY SPECTRUM . . . 56
CREATE AREA . . . 63
CREATE BASE . . . 65
CREATE CALIBRATION FIXED . . . 68
CREATE CALIBRATION FLOAT . . . 71
CREATE CALIBRATION LINEAR . . . 74
CREATE CONDITION COMPOSED . . . 77
CREATE CONDITION FUNCTION . . . 81
CREATE CONDITION MULTIWINDOW . . . 85
CREATE CONDITION PATTERN . . . 89
CREATE CONDITION POLYGON . . . 94
CREATE CONDITION WINDOW . . . 99
CREATE DIRECTORY . . . 103
CREATE DYNAMIC ENTRY BITSPECTRUM . . . 105
CREATE DYNAMIC ENTRY COMPOSED . . . 112
CREATE DYNAMIC ENTRY FUNCTION . . . 117

CREATE DYNAMIC ENTRY INDEXEDSPECTRUM 123
CREATE DYNAMIC ENTRY MULTIWINDOW 130
CREATE DYNAMIC ENTRY PATTERN 136
CREATE DYNAMIC ENTRY POLYGON 143
CREATE DYNAMIC ENTRY PROCEDURE 149
CREATE DYNAMIC ENTRY SCATTER 155
CREATE DYNAMIC ENTRY SPECTRUM 159
CREATE DYNAMIC ENTRY WINDOW 166
CREATE DYNAMIC LIST 173
CREATE ELEMENT 178
CREATE LINK 181
CREATE OVERLAY 183
CREATE PICTURE 189
CREATE POLYGON 194
CREATE POOL 196
CREATE SPECTRUM 198
CREATE TABLE CONDITION 204
CREATE TABLE SPECTRUM 207
CREATE TYPE 210
DECALIBRATE SPECTRUM 212
DECOMPRESS BASE 214
DEFINE PICTURE SETUP 216
DELETE CALIBRATION 220
DELETE CONDITION 223
DELETE DYNAMIC ENTRY 226
DELETE DYNAMIC LIST 229
DELETE ELEMENT 231
DELETE LINK 233
DELETE OVERLAY 235
DELETE PICTURE 238
DELETE POLYGON 241
DELETE POOL 244
DELETE SECTION 246
DELETE SPECTRUM 247
DISMOUNT BASE 251
DUMP SPECTRUM 252
FREEZE CONDITION 254
FREEZE SPECTRUM 257
LOCATE BASE 259
LOCATE DIRECTORY 261
LOCATE ELEMENT 263
LOCATE ID 265

LOCATE POOL . . . 267
LOCATE QUEUEELEMENT . . . 269
LOCATE TYPE . . . 271
MODIFY DIRECTORY . . . 273
MODIFY FRAME SCATTER . . . 275
MODIFY FRAME SPECTRUM . . . 282
MODIFY TABLE CONDITION . . . 294
MODIFY TABLE SPECTRUM . . . 297
MOUNT BASE . . . 300
PROTECT SPECTRUM . . . 302
READ CAMAC SPECTRUM . . . 305
SET CONDITION PATTERN . . . 308
SET CONDITION WINDOW . . . 312
SET LETTERING . . . 316
SET MEMBER . . . 319
SET SPECTRUM POINT . . . 321
SHOW AREA . . . 325
SHOW CALIBRATION . . . 327
SHOW CAMAC SPECTRUM . . . 330
SHOW CONDITION . . . 336
SHOW DIRECTORY . . . 341
SHOW DYNAMIC LIST . . . 343
SHOW ELEMENT . . . 346
SHOW GOOSY STATUS . . . 349
SHOW HOME_BLOCK . . . 351
SHOW LINK . . . 353
SHOW MEMBER . . . 356
SHOW PICTURE . . . 358
SHOW POLYGON . . . 362
SHOW POOL . . . 365
SHOW SPECTRUM . . . 368
SHOW TABLE . . . 374
SHOW TREE . . . 380
SHOW TYPE . . . 382
START MR2000 . . . 385
STOP MR2000 . . . 387
SUMUP SPECTRUM . . . 389
UNFREEZE CONDITION . . . 394
UNFREEZE SPECTRUM . . . 396
UNPROTECT SPECTRUM . . . 398
UPDATE BASE . . . 401
UPDATE DYNAMIC LIST . . . 403

WRITE CAMAC SPECTRUM 405