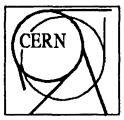
ALEPH 89-12
DATACQ 89-3
30th JANUARY 1989

A.Aimar University of Torino
J.Harvey RAL/CERN-EF
M.Lubich University of Innsbruck
G.Waltermann MPI/Munich



ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
Laboratoire Européen pour la Physique des Particules
European Laboratory for Particle Physics

G P H - Reference Manual

Version 1.1

Abstract

This GPH - Reference Manual gives you a full description of all routines used within the GPH package described in the 'GPH - User Guide'

Contents

Contents

1 Introduction

The GPH - Reference Manual describes all routines used within the GPH package.

The GPH package itself is described in the 'GPH User Guide' which is published as:

- ALEPH 89-11
- DATACQ 89-3
- 30th January 1989

GPH-routines are listed in alphabetical order with desription of input and output. The routines are all declared as functions, as they are returning '0' on any error - otherwise '1'.

2 STRUCTURE OF GPH-ROUTINES

Overall Control functions :

- GPH_INIT
- GPH_CREATE_GLOBAL
- GPH_MAP_GLOBAL
- GPH_INQUIRE_COLOUR_SETUP
- GPH_CLOSE
- GPH_ERROR

Icon handling :

- GPH_OPEN_ICON
- GPH_CLOSE_ICON
- GPH_INSERT_CIRCLE
- GPH_INSERT_FILLAREA
- GPH_INSERT_FREELINE
- GPH_INSERT_POLYLINE
- GPH_INSERT_POLYMARKER
- GPH_INSERT_TEXT
- GPH_GET_ICONID

Objecttype handling :

- GPH_OPEN_OBJTYPE
- GPH_CLOSE_OBJTYPE
- GPH_SET_DETLEVEL
- GPH_GET_OBJTYPEID

Component handling :

- GPH_CREATE_COMPONENT
- GPH_GET_COMPONENTID

Object handling :

- GPH_CREATE_OBJECT
- GPH_GET_OBJECTID

Retrieve information on graphics primitives :

- GPH_GETPRIM_ICON

- GPH_GETPRIM_OTCOMP
- GPH_GETPRIM_OBJTYPE
- GPH_GETPRIM_COMPONENT
- GPH_GETPRIM_OBJECT
- GPH_GETPRIM_DETELEMENT

Drawing :

- GPH_DRAW_ICON
- GPH_DRAW_OBJTYPE
- GPH_DRAW_COMPONENT
- GPH_DRAW_OBJECT

Window managing :

- GPH_OPEN_OUTPUT
- GPH_CLOSE_OUTPUT
- GPH_OPEN_WINDOW
- GPH_SET_WINDOW
- GPH_CLEAR_WORKSTATION

Pick facility :

- GPH_PICK_OBJECT

Graphics Aspects :

- GPH_CHANGE_COLOUR
- GPH_PUT_FILLASP
- GPH_PUT_LINEASP
- GPH_PUT_MARKERASP
- GPH_PUT_TEXTASP
- GPH_RESET
- GPH_SET_BORDER
- GPH_SET_BACKGR
- GPH_SET_COLOUR
- GPH_SET_LINEASP
- GPH_SET_FILLASP
- GPH_UPDATECOLOUR_ICON
- GPH_UPDATECOLOUR_OBJECT
- GPH_UPDATECOLOUR_OBJTYPE
- GPH_UPDATEFASP_ICON

- GPH_UPDATEFASP_OBJECT
- GPH_UPDATELASP_ICON
- GPH_UPDATELASP_OBJECT
- GPH_UPDATEMASP_ICON
- GPH_UPDATEMASP_OBJECT
- GPH_UPDATETASP_ICON
- GPH_UPDATETASP_OBJECT

Metafile creation :

- GPH_OPEN_METAFILE (GKS)
- GPH_CLOSE_METAFILE (GKS)
- GPH_WRITE_METAFILE (UIS)

Some (maybe) useful internal Geometry-routines :

- GPI_GET_CENTER
- GPI_GET_MAX_WINDOW
- GPI_ROTATE_SHAPE
- GPI_SCALE_SHAPE
- GPI_SHIFT_SHAPE

3 REFERENCE MANUAL

GPH_CHANGE_COLOUR

GPH_CHANGE_COLOUR gives a new definition of a colour in terms of red, green and blue intensities.

FORMAT

GPH_CHANGE_COLOUR *colour_id,red_intens, green_intens,blue_intens*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

colour_id

VMS Usage: **longword (unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifier of the colour.

red_intens

VMS Usage: **longword (unsigned)**
type: **floating**
access: **read_only**
mechanism: **by reference**
New red intensity for the colour [0.,1.].

green_intens

VMS Usage: **longword (unsigned)**
type: **floating**
access: **read_only**
mechanism: **by reference**
New green intensity for the colour [0.,1.].

blue_intens

VMS Usage: **longword (unsigned)**
type: **floating**
access: **read_only**
mechanism: **by reference**
New blue intensity for the colour [0.,1.].

DESCRIPTION

GPH_CHANGE_COLOUR updates the current intensity definition of the given colour.

GPH_CLEAR_WORKSTATION

GPH_CLEAR_WORKSTATION clears at the moment of the call the specified workstation window.

FORMAT

GPH_CLEAR_WORKSTATION *workstation_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

workstation_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the workstation window to be cleared.

DESCRIPTION

GPH_CLEAR_WORKSTATION can be called at any moment after the specified workstation window had been opened. Otherwise no action being taken.

GPH_CLOSE

GPH_CLOSE ends the use of GPH routines, started by a call to GPH_INIT

FORMAT GPH_CLOSE

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS *NONE*

DESCRIPTION

GPH_CLOSE can only be called at the end of the use of GPH calls. Before being closed GPH must be opened by GPH_INIT.

GPH_CLOSE_ICON

GPH_CLOSE_ICON closes the icon definition currently open.

FORMAT GPH_CLOSE_ICON

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS *NONE*

DESCRIPTION

GPH_CLOSE_ICON can only be called at the end of a sequence that defines an icon i.e. a sequence started with a call to GPH_OPEN_ICON and containing a set of calls to GPH_INSERT_x.

GPH_CLOSE_METAFILE

GPH_CLOSE_METAFILE closes an open GKS metafile opened by the call to GPH_OPEN_METAFILE.

FORMAT

GPH_CLOSE_METAFILE *metafile_id*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

metafile_id
VMS Usage: **longword(unsigned)**
type: **read_only**
access: **by reference**
mechanism:
Identifies the metafile to close.

DESCRIPTION

This routine is for *GKS* -metafiles only!!! GPH_CLOSE_METAFILE can only be called at the end of a sequence started with a call to GPH_OPEN_METAFILE. Between these two calls, every call to GPH drawing routines is saved in this file.
The file is a standard GKS-metafile. Multiple paging is possible (call to GPH_CLEAR_WORKSTATION!) and the file can be printed by following a (rather complicated) procedure. Please consult a GKS-metafile-expert or the CERN-publictaion CERN/DD/US/111 'Guide to Computer Graphics at CERN'.

GPH_CLOSE_OBJTYPE

GPH_CLOSE_OBJTYPE closes the ObjectType definition currently open.

FORMAT GPH_CLOSE_OBJTYPE

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS *NONE*

DESCRIPTION

GPH_CLOSE_OBJTYPE can only be called at the end of a sequence that defines an ObjectType i.e. a sequence started with a call to GPH_OPEN_OBJTYPE and containing a set of calls to GPH_CREATE_COMPONENT.

GPH_CLOSE_OUTPUT

GPH_CLOSE_OUTPUT closes an open workstation opened by the call to GPH_OPEN_OUTPUT.

FORMAT

GPH_CLOSE_OUTPUT *workstation_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

workstation_id
VMS Usage: **longword(signed)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the workstation to close.

DESCRIPTION

GPH_CLOSE_OUTPUT can only be called at the end of a sequence started with a call to GPH_OPEN_OUTPUT. Between these two calls, every call to GPH drawing routines is executed on this workstation.

GPH_CREATE_COMPONENT

GPH_CREATE_COMPONENT creates a new component of the ObjectType currently opened.

FORMAT

GPH_CREATE_COMPONENT *icon_id,shift,rotation, scaling, component_name, component_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

icon_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the icon that transformed gives the objecttype component.

shift

VMS Usage: **longword(signed)**
type: **real array(3)**
access: **read_only**
mechanism: **by reference**
Shift transformation on the icon to obtain the objecttype component.

rotation

VMS Usage: **longword(signed)**
type: **real array(2)**
access: **read_only**
mechanism: **by reference**
Rotation transformation on the icon to obtain the objecttype component.

scaling

VMS Usage: **longword(signed)**
type: **real array(3)**
access: **read_only**
mechanism: **by reference**
Scaling transformation on the icon to obtein the objecttype component.

component_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**

component_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**

DESCRIPTION

GPH_CREATE_COMPONENT allows the creation of a component of an ObjectType like transformation of a defined icon. This 'transformation' is made by the shift,rotation and scaling of that icon. If one (or more) of these transformations does not have to be executed use the predefined parameters NOSHIFT,NOROT, NOSCAL instead of arrays without meaning. All aspects of the component are those of the icon; to update some aspects or colour is possible to use GPH_UPDATExx_OBJTYPE.

GPH_CREATE_GLOBAL

GPH_CREATE_GLOBAL creates a global section file to hold the graphical DataBase.

FORMAT

GPH_CREATE_GLOBAL *file_name,section_name*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

file_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Name of file to be stored.

section_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Internal section name.

DESCRIPTION

GPH_CREATE_GLOBAL is at the moment on top of a very VAX-specific system of 'Global Sections'. Please contact the related manuals if you want to know more about them.
The 'file_name' should describe completely the name of the file holding the DataBase.
The 'section_name' identifies internally within the application the private copy of the global section in memory. It has no effect within GPH at the creation-moment. Give it just a nice name....

GPH_CREATE_OBJECT

GPH_CREATE_OBJECT creates a new object.

FORMAT

GPH_CREATE_OBJECT *objtype_id,shift, rotation,scaling, object_name,object_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

objtype_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifier of the ObjectType.

shift

VMS Usage: **longword(signed)**
type: **real array(3)**
access: **read_only**
mechanism: **by reference**
Shift transformation on the ObjectType to obtain the object.

rotation

VMS Usage: **longword(signed)**
type: **real array(2)**
access: **read_only**
mechanism: **by reference**
Rotation transformation on the ObjectType to obtain the object.

scale

VMS Usage: **longword(signed)**
type: **real array(3)**
access: **read_only**
mechanism: **by reference**
Scaling transformation on the ObjectType to obtain the object.

object_name

VMS Usage: **character string**
type: **string**
access: **read_only**

mechanism: **by descriptor**
Name of the object created.

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Unique identifier of the object created.

DESCRIPTION

GPH_CREATE_OBJECT creates a new instance of an ObjectType. The object can be the result of a transformation of the chosen ObjectType. Transformation means shift, rotation and scaling.

GPH_DRAW_COMPONENT

GPH_DRAW_COMPONENT draws the specified component of the specified object.

FORMAT

GPH_DRAW_COMPONENT *object_id,comp_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifier of the object.

comp_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifier of the component

DESCRIPTION

GPH_DRAW_COMPONENT draws a component of an object concerning the identifiers received.

GPH_DRAW_ICON

GPH_DRAW_ICON draws an icon.

FORMAT

GPH_DRAW_ICON *icon_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

icon_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifier of the icon.

DESCRIPTION

GPH_DRAW_ICON draws an icon receiving the identifier..

GPH_DRAW_OBJECT

GPH_DRAW_OBJECT draws an object.

FORMAT

GPH_DRAW_OBJECT *object_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

object_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifier of the object.

DESCRIPTION

GPH_DRAW_OBJECT draws an object receiving the identifier..

GPH_DRAW_OBJTYPE

GPH_DRAW_OBJTYPE draws an ObjectType

FORMAT

GPH_DRAW_OBJTYPE *objtype_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

objtype_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifier of the ObjectType.

DESCRIPTION

GPH_DELETE_OBJTYPE draws an ObjectType receiving the identifier.

GPH_ERROR

GPH_ERROR sends an error-message using the system call LIB$SIGNAL under VAX VMS.

FORMAT

GPH_ERROR *error_id,type_param,as_param,ul_param*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

error_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifier of the error to print

type_param

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Type of parameter following (used for LIB$SIGNAL call)

as_param

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by reference**
Character-string-parameter (when 'type_param.eq.ASCII' otherwise dummy)

ul_param

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Unsigned_longword_parameter (when 'type_param.eq.UNSLW' otherwise dummy)

DESCRIPTION

GPH_ERROR is called within most of the other GPH-routines to print a message for any error detected.
WARNING: Since it calls LIB$SIGNAL it is a real VAX-VMS-routine!

GPH_GETPRIM_COMPONENT

GPH_GETPRIM_COMPONENT retrieves the information to describe an COMPONENT in terms of graphics primitives.

FORMAT

GPH_GETPRIM_COMPONENT *object_id, comp_id, nprims, npoints, x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
OBJECT-identifier

comp_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
COMPONENT-identifier within specified OBJECT

nprims

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Number of primitives necessary to describe the component.

npoints

VMS Usage: **longword(unsigned)**
type: **integer array**
access: **write_only**
mechanism: **by reference**
Number of points for each primitive

x

VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**

mechanism: **by reference**
Array of all X-coordinates.

y
VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all Y-coordinates.

z
VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all Z-coordinates.

DESCRIPTION

GPH_GETPRIM_COMPONENT can be called at any moment an OBJECT with its specific OBJECT_ID is buildt of several COMPONENTS with their COMPONENT_IDs. If it does not exist an error message is printed. The array 'npoints' has to be dimensioned to the maximum number of primitives ('nprims'). The arrays x,y,z to the maximum number of points for all primitives.

GPH_GETPRIM_DETELEMENT

GPH_GETPRIM_DETELEMENT retrieves the information (in terms of graphics primitives) to describe an element of the ALEPH-detector defined in a character-string 'element' which follows the ALEPH-naming-convention (see below).

FORMAT

GPH_GETPRIM_DETELEMENT *element, nprims, npoints, x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

element

VMS Usage: **character**
type: **string**
access: **read_only**
mechanism: **by reference**

Definitions of the Detector-element following the ALEPH-naming-convention (see below)

nprims

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**

Number of primitives necessary to describe the detector-element

npoints

VMS Usage: **longword(unsigned)**
type: **integer array**
access: **write_only**
mechanism: **by reference**

Number of points for each primitive

x

VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**

Array of all X-coordinates.

y
VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all Y-coordinates.

z
VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all Z-coordinates.

DESCRIPTION

GPH_GETPRIM_DETELEMENT can be called at any moment when the ALEPH-detector (or parts of) is defined within the GPH-database - with the graphical objects being named following the ALEPH-naming-conventions (please see the ALEPH-publication 'ALEPH Resource Naming Conventions' by A.Belk). If an element does not exist an error message is printed.
The array 'npoints' has to be dimensioned to the maximum number of primitives ('nprims'). The arrays x,y,z to the maximum number of points for all primitives.

GPH_GETPRIM_ICON

GPH_GETPRIM_ICON retrieves the information to describe an ICON in terms of graphics primitives.

FORMAT

GPH_GETPRIM_ICON *icon_id,nprims,npoints,x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

icon_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
ICON-identifier

nprims

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Number of primitives necessary to describe the ICON

npoints

VMS Usage: **longword(unsigned)**
type: **integer array**
access: **write_only**
mechanism: **by reference**
Number of points for each primitive

x

VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all X-coordinates.

y

VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**

Array of all Y-coordinates.

z

VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all Z-coordinates.

DESCRIPTION

GPH_GETPRIM_ICON can be called at any moment an ICON with its specific ICON_ID is defined.
The array 'npoints' has to be dimensioned to the maximum number of primitives ('nprims'). The arrays x,y,z to the maximum number of points for all primitives.

GPH_GETPRIM_OBJECT

GPH_GETPRIM_OBJECT retrieves the information to describe an OBJECT in terms of graphics primitives.

FORMAT

GPH_GETPRIM_OBJECT *object_id, nprims, npoints, x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
OBJECT-identifier

nprims

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Number of primitives necessary to describe the object.

npoints

VMS Usage: **longword(unsigned)**
type: **integer array**
access: **write_only**
mechanism: **by reference**
Number of points for each primitive

x

VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all X-coordinates.

y

VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**

mechanism: **by reference**
Array of all Y-coordinates.

z
VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all Z-coordinates.

DESCRIPTION

GPH_GETPRIM_OBJECT can be called at any moment an OBJECT with its specific OBJECT_ID is defined. If it does not exist an error message is printed.
The array 'npoints' has to be dimensioned to the maximum number of primitives ('nprims'). The arrays x,y,z to the maximum number of points for all primitives.

GPH_GETPRIM_OBJTYPE

GPH_GETPRIM_OBJTYPE retrieves the information to describe an ObjectType (OBJTYPE) in terms of graphics primitives.

FORMAT

GPH_GETPRIM_OBJTYPE *objtype_id, nprims, npoints, x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

objtype_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
OBJECT-TYPE-identifier

nprims

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Number of primitives necessary to describe the object-type.

npoints

VMS Usage: **longword(unsigned)**
type: **integer array**
access: **write_only**
mechanism: **by reference**
Number of points for each primitive

x

VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all X-coordinates.

y

VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**

mechanism: **by reference**
Array of all Y-coordinates.

z
VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all Z-coordinates.

DESCRIPTION

GPH_GETPRIM_OBJTYPE can be called at any moment an ObjectType with its specific OBJTYPE_ID is defined. If it does not exist an error message is printed.
The array 'npoints' has to be dimensioned to the maximum number of primitives ('nprims'). The arrays x,y,z to the maximum number of points for all primitives.

GPH_GETPRIM_OTCOMP

GPH_GETPRIM_OTCOMP retrieves the information (in terms of graphics primitives) to describe an ObjectType component 'OTCOMP' of an existing ObjectType 'OBJTYPE'.

FORMAT

GPH_GETPRIM_OTCOMP *objtype_id, otcomp_id, nprims, npoints, x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

objtype_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
OBJECT-TYPE-identifier

otcomp_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
OBJECT-TYPE-COMPONENT-identifier

nprims

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Number of primitives necessary to describe the objtype-component.

npoints

VMS Usage: **longword(unsigned)**
type: **integer array**
access: **write_only**
mechanism: **by reference**
Number of points for each primitive

x
VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all X-coordinates.

y
VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all Y-coordinates.

z
VMS Usage: **longword(signed)**
type: **real array**
access: **write_only**
mechanism: **by reference**
Array of all Z-coordinates.

DESCRIPTION

GPH_GETPRIM_OTCOMP can be called at any moment an object-type with its specific OBJTYPE_ID is built of ObjectType components with their OTCOMP-IDs. If it does not exist an error message is printed. The array 'npoints' has to be dimensioned to the maximum number of primitives ('nprims'). The arrays x,y,z to the maximum number of points for all primitives.

GPH_GET_COMPONENTID

GPH_GET_COMPONENTID retrieves the identifier of a component of an object type.

FORMAT

GPH_GET_COMPONENTID *objtype_id, component_name, component_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

objtype_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifier of the ObjectType.

component_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Name of the Component

component_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Component-ID returned

DESCRIPTION

GPH_GET_COMPONENTID allows to retreive the identifier of a component specified by name and ObjectType which belongs.

GPH_GET_ICONID

GPH_GET_ICONID retrieves the identifier of an icon.

FORMAT

GPH_GET_ICONID *icon_name,icon_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

icon_name,

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Name of the icon.

icon_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Identifier of the icon.

DESCRIPTION

GPH_GET_ICONID allows to retreive the identifier of an icon specified by name.

GPH_GET_OBJECTID

GPH_GET_OBJECTID retrieves the identifier of an object.

FORMAT

GPH_GET_OBJECTID *object_name,object_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

object_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Name of the object.

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Identifier of the object,

DESCRIPTION

GPH_GET_OBJECTID allows to retreive the identifier of an object specified by name.

GPH_GET_OBJTYPEID

GPH_GET_OBJTYPEID retrieves the identifier of an ObjectType.

FORMAT

GPH_GET_OBJTYPEID *objtype_name,object_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

objtype_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Name of the object-type.

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Identifier of the objecttype.

DESCRIPTION

GPH_GET_OBJECTID allows to retreive the identifier of an object-type specified by name.

GPH_INIT

GPH_INIT starts the use of GPH routines.

FORMAT GPH_INIT

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS *NONE*

DESCRIPTION

GPH_INIT must be the first GPH routine called; if that is not done a fatal error is produced. After this call GPH is usable and must be closed by GPH_CLOSE.

GPH_INQUIRE_COLOUR_SETUP

GPH_INQUIRE_COLOUR_SETUP finds out the colour-setup of the workstation in use.

FORMAT

GPH_INQUIRE_COLOUR_SETUP *workstation_type, colours*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

workstation_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Specifies the workstation-type being used (for the GKS-version only!)

colours

VMS Usage: **byte**
type: **logical**
access: **write_only**
mechanism: **by reference**
Logical variable set to .TRUE. for COLOUR-station and .FALSE. for MONOCHROME-station.

DESCRIPTION

GPH_INQUIRE_COLOUR_SETUP can be called at any moment of GPH independent of any outputs or windows opened or closed. But logically it should be called sometime at the beginning of the application to define all sorts of graphical representations different concerning the setup of the station.

GPH_INSERT_CIRCLE

GPH_INSERT_CIRCLE inserts a circle in the definition of the icon currently open.

FORMAT

GPH_INSERT_CIRCLE *x_c,y_c,z_c, radius, theta, phi*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

x_c

VMS Usage: **longword(signed)**
type: **floating**
access: **read_only**
mechanism: **by reference**
X-coordinate of the circle center

y_c

VMS Usage: **longword(signed)**
type: **floating**
access: **read_only**
mechanism: **by reference**
Y-coordinate of the circle center

z_c

VMS Usage: **longword(signed)**
type: **floating**
access: **read_only**
mechanism: **by reference**
Z-coordinate of the circle center

radius

VMS Usage: **longword(signed)**
type: **floating**
access: **read_only**
mechanism: **by reference**
Radius of the circle.

theta

VMS Usage: **longword(signed)**
type: **floating**
access: **read_only**
mechanism: **by reference**
Angle (in radians) around the X axe.

phi

VMS Usage: **longword(signed)**
type: **floating**
access: **read_only**
mechanism: **by reference**
Angle (in radians) around the Z axe.

DESCRIPTION

GPH_INSERT_CIRCLE allows the insertion of a circle the definition of the icon currently open. The circle is inserted only if an open icon exists. The aspects of the circle are like those of polylines of the same icon.

GPH_INSERT_FILLAREA

GPH_INSERT_FILLAREA inserts a fillarea in the definition of the icon currently open.

FORMAT

GPH_INSERT_FILLAREA *number_points,x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

number_points

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Number of points of the fillarea.

x

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
X-coordinates of the fillarea.

y

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
Y-coordinates of the fillarea.

z

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
Z-coordinates of the fillarea.

DESCRIPTION

GPH_INSERT_FILLAREA allows the insertion of a fillarea to the definition of the icon currently open. The fillarea is inserted only if an open icon exists.

GPH_INSERT_FREELINE

GPH_INSERT_FREELINE inserts a freeline primitive in the definition of the icon currently open.

FORMAT

GPH_INSERT_FREELINE *number_points,x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

number_points

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Number of points of the freeline.

x

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
X-coordinates of the freeline.

y

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
Y-coordinates of the freeline.

z

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
Z-coordinates of the freeline.

DESCRIPTION

GPH_INSERT_FREELINE allows the insertion of a freeline to the definition of the icon currently open. The freeline is inserted only if an open icon exists. The aspects of the freeline are like those of polylines of the same icon.

GPH_INSERT_POLYLINE

GPH_INSERT_POLYLINE inserts a polyline primitive in the definition of the icon currently open.

FORMAT

GPH_INSERT_POLYLINE *number_points,x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

number_points

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Number of points of the polyline.

x

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
X-coordinates of the polyline.

y

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
Y-coordinates of the polyline.

z

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
Z-coordinates of the polyline.

DESCRIPTION

GPH_INSERT_POLYLINE allows the insertion of a polyline to the definition of the icon currently open. The polyline is inserted only if an open icon exists.

GPH_INSERT_POLYMARKER

GPH_INSERT_POLYMARKER inserts a polymarker primitive in the definition of the icon currently open.

FORMAT

GPH_INSERT_POLYMARKER *number_points,x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

number_points

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Number of polymarkers.

x

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
X-coordinates of the polymarkers.

y

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
Y-coordinates of the polymarkers.

z

VMS Usage: **longword(signed)**
type: **real array(*)**
access: **read_only**
mechanism: **by reference**
Z-coordinates of the polymarkers.

DESCRIPTION

GPH_INSERT_POLYMARKER allows the insertion of a polymarker. The polymarker is inserted only if an open icon exists.

GPH_INSERT_TEXT

GPH_INSERT_TEXT inserts a text string in the definition of the icon currently open.

FORMAT

GPH_INSERT_TEXT *text_string,x,y,z*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

text_string

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Text string.

x

VMS Usage: **longword(signed)**
type: **floating**
access: **read_only**
mechanism: **by reference**
X-coordinate of the first point of the text

y

VMS Usage: **longword(signed)**
type: **floating**
access: **read_only**
mechanism: **by reference**
Y-coordinate of the first point of the text

z

VMS Usage: **longword(signed)**
type: **floating**
access: **read_only**
mechanism: **by reference**
Z-coordinate of the first point of the text

DESCRIPTION

GPH_INSERT_TEXT allows the insertion of a text string in the definition of the icon currently open. The text is inserted only if an open icon exists.

GPH_MAP_GLOBAL

GPH_MAP_GLOBAL maps to a global section file holding the graphical DataBase.

FORMAT

GPH_MAP_GLOBAL *file_name,section_name*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

file_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Name of global section file holding the DataBase.

section_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Internal section name.

DESCRIPTION

GPH_CREATE_GLOBAL is at the moment on top of a very VAX-specific system of 'Global Sections'. Please contact the related manuals if you want to know more about them.
The 'file_name' should describe completely the name of the file holding the DataBase.
The 'section_name' identifies internally within the application the private copy of the global section in memory. It is just useful for multi user mapping of the same global section, but not used by GPH at that level.

GPH_OPEN_ICON

GPH_OPEN_ICON open an icon definition.

FORMAT

GPH_OPEN_ICON *icon_name,icon_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

icon_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Name of the icon.

icon_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Unique identifier for the icon.

DESCRIPTION

GPH_OPEN_ICON starts the definition of an icon; this definition will end with the call GPH_CLOSE_ICON. Note the current aspects and colour become the aspects of this icon.
To change an aspect after the opening one has to call one of the routines GPH_UPDATExxx_ICON; to set an icon aspect before the opening it is enough to change the current aspect using GPH_SET_yyy.
The open icon is alway referable by its unique identifier.

GPH_OPEN_METAFILE

GPH_OPEN_METAFILE opens a GKS metafile that can be closed by the call to GPH_CLOSE_METAFILE.

FORMAT

GPH_OPEN_METAFILE *metafile_name,metafile_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

metafile_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Name of the metafile

metafile_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the metafile.

DESCRIPTION

This routine is for *GKS* -metafiles only!!! GPH_OPEN_METAFILE should only be called at the start of a sequence which will end with a call to GPH_CLOSE_METAFILE. Between these two calls, every call to GPH drawing routines is saved in this file.
For UIS-metafiles please see routine GPH_WRITE_METAFILE.

GPH_OPEN_OBJTYPE

GPH_OPEN_OBJTYPE opens a new ObjectType definition.

FORMAT

GPH_OPEN_OBJTYPE *objtype_name, objtype_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

objtype_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Name of the ObjectType.

objtype_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Unique identifier for the ObjectType.

DESCRIPTION

GPH_OPEN_OBJTYPE starts the definition of an object type; this definition will end with the call GPH_CLOSE_OBJTYPE. After this call every call to GPH_CREATE_COMPONENT will insert in the ObjectType definition the transformation of an icon.

GPH_OPEN_OUTPUT

GPH_OPEN_OUTPUT open a new workstation to receive graphic output.

FORMAT

GPH_OPEN_OUTPUT *workstation_type, workstation_name, connection_id, wkid*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

workstation_type

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifier of the workstation type (only used by GKS - please see the GKS manual).

workstation_name

VMS Usage: **character string**
type: **string**
access: **read_only**
mechanism: **by descriptor**
Name of the workstation.

connection_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Identifier of the connection.

wkid

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Identifier of the output (workstation) open.

DESCRIPTION

GPH_OPEN_OUTPUT causes a window to appear on the screen.
The returned value WKID is the internal GPH identifier for this workstation window and should be used as the workstation window-identifier for all appropriate GPH-routines.
The call to GPH_CLOSE_OUTPUT will erase completely this window from the screen.

GPH_OPEN_WINDOW

GPH_OPEN_WINDOW defines the WORLD-COORDINATE-system for the current workstation window.

FORMAT

GPH_OPEN_WINDOW *Transformation_id, x_min,x_max, y_min,y_max*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

transformation_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
GKS-transformation-identifier (not used for UIS-version)

x_min

VMS Usage: **longword(signed)**
type: **real**
access: **read_only**
mechanism: **by reference**
X-coordinate in World Coordinates

x_max

VMS Usage: **longword(signed)**
type: **real**
access: **read_only**
mechanism: **by reference**
X-coordinate in World Coordinates

y_min

VMS Usage: **longword(signed)**
type: **real**
access: **read_only**
mechanism: **by reference**
Y-coordinate in World Coordinates

y_max

VMS Usage: **longword(signed)**
type: **real**
access: **read_only**

mechanism: **by reference**
Y-coordinate in World Coordinates

DESCRIPTION

GPH_OPEN_WINDOW can be called at any time after the call to GPH_OPEN_OUTPUT (which creates the workstation window) to change the WC-system for the particular workstation window.
WARNING: with multiple windowing you should take care that the workstation window where you want to change the WC-system is activated!! (call GPH_SET_WINDOW !)

GPH_PICK_OBJECT

GPH_PICK_OBJECT enables you to 'pick' an object (or one of its components) using the locator input ('mouse'). It returns the OBJECT_ID and the COMPONENT_ID of the picked object and object component.

FORMAT

GPH_PICK_OBJECT *wkid,pick_stat,object_id,comp_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

wkid

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Workstation_identifier of the window in which you want to pick.

pick_stat

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Return-flag: set to ZERO at PICK-fault, otherwiese ONE

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Object identifier of the picked object.

comp_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Component identifier of the picked component.

GPH_PICK_OBJECT

DESCRIPTION

GPH_PICK_OBJECT is the most simple way of reading the locator input (the 'mouse') in request form. With the internal GPH Object_Identifier and Component_Identifier you can easily change (redraw in different colour, highlight etc.) the picked object using GPH-routines.

GPH_PUT_FILLASP

GPH_PUT_FILLASP defines a fillarea aspect to become referable by an identifier.

FORMAT

GPH_PUT_FILLASP *interion_style, style_index, fillarea_aspect_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

interior_style

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Fillarea interior style.

style_index

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Fillarea style index.

fillarea_aspect_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Unique identifier for the fillarea aspect.

DESCRIPTION

GPH_PUT_FILLASP associates to a fillarea aspect an unique identifier to refere it. This identifier can be used to call GPH_UPDATExx_yy (the updating routines) or to set the aspect as 'current aspect'.
Possibles values are workstation dependent.
To choose the interior style followind parameters are defined:
0 - FHOLLOW
1 - FSOLID
2 - FPATTERN
3 - FHATCH

GPH_PUT_LINEASP

GPH_PUT_LINEASP defines a line aspect to become referable by an identifier.

FORMAT

GPH_PUT_LINEASP *line_type, line_width, line_aspect_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

line_type

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Line type

line_width

VMS Usage: **longword(signed)**
type: **real**
access: **read_only**
mechanism: **by reference**
Line width

line_aspect_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Unique identifier for the line aspect

DESCRIPTION

GPH_PUT_LINEASP associates to a line aspect an unique identifier to refere it. This identifier can be used to call GPH_UPDATExx_yy (the updating routines) or to set the aspect as 'current aspect'.
To choose the line type following parameters are defined:
1 - LSOLID
2 - LDASHED
3 - LDOTTED
4 - LDASH_DOTTED

GPH_PUT_MARKERASP

GPH_PUT_MARKERASP defines a marker aspect to become referable by an identifier.

FORMAT

GPH_PUT_MARKERASP *marker_type, marker_width, marker_aspect_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

marker_type

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Marker type

marker_width

VMS Usage: **longword(signed)**
type: **real**
access: **read_only**
mechanism: **by reference**
Marker width

marker_aspect_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Unique identifier for the marker aspect

DESCRIPTION

GPH_PUT_MARKERASP associates to a marker aspect an unique identifier to refere it. This identifier can be used to call GPH_UPDATExx_yy (the updating routines) or to set the aspect as 'current aspect'.
To choose the marker type following parameters are defined:
1 - MPOINT
2 - MPLUS
3 - MSTAR
4 - MOMARK
5 - MXMARK

GPH_PUT_TEXTASP

GPH_PUT_TEXTASP defines a text aspect to become referable by an identifier.

FORMAT

GPH_PUT_TEXTASP *text_font, text_precision, text_aspect_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

text_font

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Text font

text_precision

VMS Usage: **longword(signed)**
type: **real**
access: **read_only**
mechanism: **by reference**
Text precision.

text_aspect_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **write_only**
mechanism: **by reference**
Unique identifier for the text aspect.

DESCRIPTION

GPH_PUT_TEXTASP associates to a text aspect an unique identifier to refere it. This identifier can be used to call GPH_UPDATExx_yy (the updating routines) or to set the aspect as 'current aspect'.
To choose the text precision following parameters are defined:
0 - TSTRING
1 - TCHAR
2 - TSTROKE

GPH_RESET

GPH_RESET resets all pointers and flags for GPH to ZERO (master clear!)

FORMAT GPH_RESET

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS *none*

DESCRIPTION

GPH_RESET sets all pointers, all aspect-flags and existing definitions to ZERO.

GPH_SET_BACKGR

GPH_SET_BACKGR gives a new colour to the backgruond of all the open wokstations.

FORMAT

GPH_SET_BACKGR *red_intens,green_intens, blue_intens*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

red_intens

VMS Usage: **longword(unsigned)**
type: **floating**
access: **read_only**
mechanism: **by reference**
New red intensity for the background [0.,1.].

green_intens

VMS Usage: **longword(unsigned)**
type: **floating**
access: **read_only**
mechanism: **by reference**
New green intensity for the background [0.,1.].

blue_intens

VMS Usage: **longword(unsigned)**
type: **floating**
access: **read_only**
mechanism: **by reference**
New blue intensity for the background [0.,1.].

DESCRIPTION

GPH_SET_BACKGR set the background colour of all the workstations opened by the call to GPH_OPEN_OUTPUT.

GPH_SET_BORDER

GPH_SET_BORDER defines that a border around a fillarea has to be drawn or not and if yes in what colour.

FORMAT

GPH_SET_BORDER *flag,colour_index*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

flag

VMS Usage: **byte**
type: **logical**
access: **read_only**
mechanism: **by reference**
Logical value .TRUE. or .FALSE. if border wanted

colour_index

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Colour_index of colour for border wanted.

DESCRIPTION

GPH_SET_BORDER tells GPH when a borderline in a specific colour should be drawn around a fillarea.

GPH_SET_COLOUR

GPH_SET_COLOUR sets the current colour.

FORMAT

GPH_SET_COLOUR *colour_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

colour_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**

Identifies a colour

DESCRIPTION

GPH_SET_COLOUR set the colour specified like 'current colour'. The intensity of a colour can be changed by the call to GPH_CHANGE_ COLOUR

GPH_SET_DETLEVEL

GPH_SET_DETLEVEL sets the current detail level.

FORMAT

GPH_SET_DETLEVEL *det_level*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

det_level
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Value of the detail level.

DESCRIPTION

GPH_SET_DETLEVEL is the call to set the 'current detail level'.

GPH_SET_FILLASP

GPH_SET_FILLASP sets the current fillarea aspect.

FORMAT

GPH_SET_FILLASP *fillarea_aspect_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

fillarea_aspect_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies a fillarea aspect.

DESCRIPTION

GPH_SET_FILLASP set the fillarea aspect specified like 'current fillarea aspect'. This aspect will be used for all fillareas of icons open after the call to this routine.

GPH_SET_LINEASP

GPH_SET_LINEASP sets the current line aspect.

FORMAT

GPH_SET_LINEASP *polyline_aspect_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

polyline_aspect_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies a line aspect.

DESCRIPTION

GPH_SET_LINEASP set the line aspect specified like 'current line aspect'. This aspect will be used for all lines of icons open after the call to this routine.

GPH_SET_WINDOW

GPH_SET_WINDOW 'activates' the specified workstation window on the screen.

FORMAT

GPH_SET_WINDOW *wkid*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

wkid

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Workstation window-identifier

DESCRIPTION

GPH_SET_WINDOW 'activates' the specified workstation window (in GKS-terms it really means GACWK(WKID)!) and sets WKID to be the 'current workstation window'.

GPH_UPDATECOLOUR_ICON

GPH_UPDATECOLOUR_ICON changes colour to an icon.

FORMAT

GPH_UPDATECOLOUR_ICON *icon_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

icon_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the icon.

DESCRIPTION

GPH_UPDATECOLOUR_ICON sets the colour of the specified icon to the 'current colour'. The 'current colour' is chosen by the call to GPH_SET_COLOUR.

GPH_UPDATECOLOUR_OBJECT

GPH_UPDATECOLOUR_OBJECT changes colour to one or more components of an object.

FORMAT

GPH_UPDATECOLOUR_OBJECT *object_id, component_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the object.

component_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the component of the object; the parameter EVERY is allowed.

DESCRIPTION

GPH_UPDATECOLOUR_OBJECT sets the colour of the specified object's component to the 'current colour'.
The component_id 'EVERY' (-1) means all components of the object are updated.
The 'current colour' is chosen by the call to GPH_SET_COLOUR.

GPH_UPDATECOLOUR_OBJTYPE

GPH_UPDATECOLOUR_OBJTYPE changes colour to one or more components of an ObjectType.

FORMAT

GPH_UPDATECOLOUR_OBJTYPE *objtype_id, otcomp_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

objtype_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the ObjectType.

otcomp_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the component of the ObjectType; the parameter EVERY is allowed.

DESCRIPTION

GPH_UPDATECOLOUR_OBJTYPE set the colour of the specified object's component to the 'current colour'.
The component_id 'EVERY' (-1) means all components of the object are updated.
The 'current colour' is chosen by the call to GPH_SET_COLOUR.

GPH_UPDATEFASP_ICON

GPH_UPDATEFASP_ICON changes fillarea aspect to an icon.

FORMAT

GPH_UPDATEFASP_ICON *icon_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

icon_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the icon.

DESCRIPTION

GPH_UPDATEFASP_ICON sets the fillarea aspect of the specified icon to the 'current fillarea aspect'. The 'current fillarea aspect' is chosen by the call to GPH_SET_FILLASP.

GPH_UPDATEFASP_OBJECT

GPH_UPDATEFASP_OBJECT changes fillarea aspect to one or more components of an object.

FORMAT

GPH_UPDATEFASP_OBJECT *object_id, component_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the object.

component_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the component of the object; the parameter EVERY is allowed.

DESCRIPTION

GPH_UPDATEFASP_OBJECT sets the fillarea aspect of the specified object's component to the 'current fillarea aspect'.
The component_id 'EVERY' (-1) means all components of the object are updated.

GPH_UPDATELASP_ICON

GPH_UPDATELASP_ICON changes line aspect to an icon.

FORMAT

GPH_UPDATELASP_ICON *icon_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

icon_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the icon.

DESCRIPTION

GPH_UPDATELASP_ICON sets the line aspect of the specified icon to the 'current line aspect'. The 'current line aspect' chosen by the call to GPH_SET_LINEASP.

GPH_UPDATELASP_OBJECT

GPH_UPDATELASP_OBJECT changes the line aspect to one or more components of an object.

FORMAT

GPH_UPDATELASP_OBJECT *object_id,component_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the object.

component_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the component of the object; the parameter EVERY is allowed.

DESCRIPTION

GPH_UPDATELASP_OBJECT sets the line aspect of the specified object's component to the 'current line aspect'.
The component_id 'EVERY' (-1) means all components of the object are updated.

GPH_UPDATEMASP_ICON

GPH_UPDATEMASP_ICON changes the marker aspect to an icon.

FORMAT

GPH_UPDATEMASP_ICON *icon_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

icon_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the icon.

DESCRIPTION

GPH_UPDATEMASP_ICON sets the marker aspect of the specified icon to the 'current marker aspect'. The 'current marker aspect' is chosen by the call to GPH_SET_MARKERASP.

GPH_UPDATEMASP_OBJECT

GPH_UPDATEMASP_OBJECT changes the marker aspect to one or more components of an object.

FORMAT

GPH_UPDATEMASP_OBJECT *object_id, component_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the object

component_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the component of the object; the parameter EVERY is allowed.

DESCRIPTION

GPH_UPDATEMASP_OBJECT sets the marker aspect of the specified object's component to the 'current marker aspect'.
The component_id 'EVERY' (-1) means all components of the object are updated.

GPH_UPDATETASP_ICON

GPH_UPDATETASP_ICON changes the text aspect to an icon.

FORMAT

GPH_UPDATETASP_ICON *icon_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

icon_id
VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the icon.

DESCRIPTION

GPH_UPDATETASP_ICON sets the line aspect of the specified icon to the 'current text aspect'. The 'current text aspect' is chosen by the call to GPH_SET_TEXTASP.

GPH_UPDATETASP_OBJECT

GPH_UPDATETASP_OBJECT changes text aspect to one or more components of an object.

FORMAT

GPH_UPDATETASP_OBJECT *object_id, component_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

object_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the object

component_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Identifies the component of the object; the parameter EVERY is allowed.

DESCRIPTION

GPH_UPDATETASP_OBJECT sets the text aspect of the specified object's component to the 'current text aspect'.
The component_id 'EVERY' (-1) means all components of the object are updated.

GPH_WRITE_METAFILE

GPH_WRITE_METAFILE creates a UIS-metafile.

FORMAT

GPH_WRITE_METAFILE *meta_name,meta_id*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

meta_name

VMS Usage: **character**
type: **string**
access: **read_only**
mechanism: **by reference**
Name of the file being created.

meta_id

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Window-identifier of the window to copy into metafile.

DESCRIPTION

GPH_WRITE_METAFILE creates at the moment of the call a file with the name 'meta-name' which is a UIS-metafile of the workstation window specified under 'meta_id'. It is essentially a copy of the display-list for that window. The file corresponds to one page of output and should be converted into a printable file using the VAX command 'RENDER'. Please ask a VAX-workstation specialist for help or consult the VAX-manual 'VMS Workstation Software Guide to Printing Graphics'.

GPI_GET_CENTER

GPI_GET_CENTER finds you the 2-dimensional center-point of a polyline with 'npoints'.

FORMAT

GPI_GET_CENTER *npoints,x,y, x_center, y_center*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

npoints,

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Number of points for polyline.

x,y

VMS Usage: **longword(signed)**
type: **real arrays(*)**
access: **read_only**
mechanism: **by reference**
X,Y-coordinates of polyline.

x_center

VMS Usage: **longword(signed)**
type: **real**
access: **write_only**
mechanism: **by reference**
X-coordinate of the center.

y_center

VMS Usage: **longword(signed)**
type: **real**
access: **write_only**
mechanism: **by reference**
Y-coordinate of the center.

GPI_GET_MAX_WINDOW

GPI_GET_MAX_WINDOW finds you the world-coordinate-window-size around a polyline with 'npoints' (e.g. needed for window-transformation after ZOOM etc.)

FORMAT

GPI_GET_MAX_WINDOW *npoints,x,y, wx1,wx2, wy1,wy2, square, bord*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

npoints,

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Number of points for polyline.

x,y

VMS Usage: **longword(signed)**
type: **real arrays(*)**
access: **read_only**
mechanism: **by reference**
X,Y-coordinates of polyline.

wx1,wx2,wy1,wy2

VMS Usage: **longword(signed)**
type: **real**
access: **write_only**
mechanism: **by reference**
Coordinates of lower/left and upper/right corner of the window.

square

VMS Usage: **byte**
type: **logical**
access: **read_only**
mechanism: **by reference**
Logical flag
set to .TRUE. - the square over the outline of the polyline is calculated (to avoid distortion of the view after transformation..)
set to .FALSE. - no action on the rectangle-outline of the polyline.

bord

VMS Usage: **byte**
type: **logical**
access: **read_only**
mechanism: **by reference**

Logical flag
set to .TRUE. - a small border (5% of size) is added to the final window
set to .FALSE. - no border is added.

GPI_ROTATE_SHAPE

GPI_ROTATE_SHAPE rotates n-points of an x,y,z-array by two angles (theta and phi) in space.

FORMAT

GPI_ROTATE_SHAPE *npoints,x,y,z,angles*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

npoints,

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Number of points to be rotated.

x,y,z

VMS Usage: **longword(signed)**
type: **real arrays(*)**
access: **read/write**
mechanism: **by reference**
X,Y,Z-coordinates of points.

angles

VMS Usage: **longword(signed)**
type: **real array(2)**
access: **read_only**
mechanism: **by reference**
angles(1) = theta
angles(2) = phi

GPI_SCALE_SHAPE

GPI_SCALE_SHAPE scales n-points of an x,y,z-array by a scale-vektor in space.

FORMAT

GPI_SCALE_SHAPE *npoints,x,y,z,scale*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

npoints,

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**

Number of points to be scaled.

x,y,z

VMS Usage: **longword(signed)**
type: **real arrays(*)**
access: **read/write**
mechanism: **by reference**

X,Y,Z-coordinates of points.

scale

VMS Usage: **longword(signed)**
type: **real array(3)**
access: **read_only**
mechanism: **by reference**

Vektor containing scaling in x,y,z.

GPI_SHIFT_SHAPE

GPI_SCALE_SHAPE shifts n-points of an x,y,z-array by a shift-vektor in space.

FORMAT

GPI_SHIFT_SHAPE *npoints,x,y,z,shift*

RETURNS

VMS Usage: **cond_value**
type: **longword(unsigned)**
access: **write only**
mechanism: **by value**

Longword condition value. Return '0' on any error - otherwise '1'.

ARGUMENTS

npoints,

VMS Usage: **longword(unsigned)**
type: **integer**
access: **read_only**
mechanism: **by reference**
Number of points to be shifted.

x,y,z

VMS Usage: **longword(signed)**
type: **real arrays(*)**
access: **read/write**
mechanism: **by reference**
X,Y,Z-coordinates of points.

scale

VMS Usage: **longword(signed)**
type: **real array(3)**
access: **read_only**
mechanism: **by reference**
Vektor containing shift in x,y,z.

That's it.
Have fun !