# Release Notes for HP GKS 7.2

on OpenVMS Integrity servers

September 2008



**Hewlett-Packard Company**
**Palo Alto, California**

# CONTENTS

## TABLES

# Preface

This document is the release notes for HP GKS Version 7.2 on OpenVMS Integrity servers.

This document is supplied in both PostScript and text form. You can find the release notes in the following locations:

**Table 1: HP GKS Release Notes**

| | |
|---|---|
| Text | SYS$HELP:HPI64GKS072.RELEASE_NOTES |
| PostScript | SYS$HELP:HPI64GKS072_RELEASE_NOTES.PS |

### Associated Documents

- *Installing HP GKS for OpenVMS Integrity servers Systems*
- *DEC GKS User's Guide*
- *DEC GKS C Binding Reference Manual*
- *DEC GKS FORTRAN Binding Reference Manual*
- *DEC GKS GKS3D$ Binding Reference Manual*
- *DEC GKS GKS$ Binding Reference Manual*
- *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*

# 1 Product Summary

HP GKS is a library of graphics functions to be included in application programs.

HP GKS Version 7.2 and its predecessors, Digital GKS Version 6.x, comply with the international standard for three-dimensional graphics (GKS–3D International Standard (ISO 8805(E) 1988)), and are upwardly compatible extensions to the ISO GKS standard (ISO 7942-1985).

HP GKS is complementary to the PHIGS graphics system; GKS is suitable for unstructured, fixed images, while PHIGS is used for structured, editable graphics data.

In this document, HP GKS refers to Version 7.2.

## 1.1 Release Description

HP GKS Version 7.2 is a release of HP GKS on OpenVMS Integrity servers running on Itanium™ processor-based Integrity™ servers. It is essentially a port to the Integrity platform of DIGITAL GKS Version 6.5, released in 1997, with the following differences:

- HP GKS is provided as a POLYCENTER Software Installation (PCSI) installation kit;
- VAX float and IEEE floating point arithmetic are supported;
- Several problems reported against Digital GKS Version 6.5 have been fixed;
- PEX and the PEX workstation types (240, 241, 242, 243; 340, 341, 342, 343) are no longer supported.

It contains support for HP LaserJet II printers and compatible HP PCL devices. This document describes completed bug fixes, and highlights changes and corrections to the HP GKS documentation set.

At present, HP GKS Version 7.2 for OpenVMS Integrity servers supports the following Device types.

- Motif
- PostScript
- Color PostScript
- HP-GL
- LaserJet (HP PCL)
- ReGIS
- Tektronix 4014
- Tektronix 4107, 4128, 4129, and 4207
- Color Sixel (LJ250, LA324)
- Black & White Sixel (LN03, LA34, LA50, LA100, DEClaser 2100 & 2200)
- DDIF
- CGM

## 1.2 Release History

For those familiar with previous releases of Digital GKS on other platforms, HP GKS Version 7.2 is the latest in the Digital GKS Version 6.*x* line, which is the successor to the DEC GKS Version 5.*x* line. The DEC GKS Version 5.*x* line was the successor to both the DEC GKS–3D line and the DEC GKS Version 4.*x* line. The following table summarizes important milestones in HP GKS history:

**Table 2: Release History**

| Version | Year | Major changes |
|---|---|---|
| 6.0 | 1994 | NIST Certification release |
| 5.3 | 1994 | Full device support on Alpha; LaserJet support |
| 5.2 | 1993 | First unified release on Alpha and VAX |
| 5.1 | 1993 | First release on Alpha architecture (Alpha-only release) |
| 5.0 | 1992 | Merger of DEC GKS–3D V1.2 and DEC GKS V4.2 |

## 1.3 Operating System Support

HP GKS Version 7.2 for OpenVMS Integrity servers supports OpenVMS Integrity servers Version 8.2-1 and higher.

## 1.4 Motif Version 1.5 Support

This release of HP GKS supports Motif Version 1.5 and higher.

**NOTE**

**When displaying to some servers running Motif V1.2 or higher, you may need to set the Mwm*clientAutoPlace resource to False in the Motif Window Manager resource file on the server. The name of this file is Mwm on UNIX systems, or DECW$MWM.DAT on OpenVMS systems. Doing so will ensure correct window placement.**

If, on successive runs of a GKS application, you see messages like this:

```
HP GKS - WARNING - 'BORDER SIZE' attribute incorrect: use
"DEFINE GKS$DECW_BORDER_SIZE 224" and restart application.
HP GKS - WARNING - 'TITLE SIZE' attribute incorrect: use
"DEFINE GKS$DECW_TITLE_SIZE -196" and restart application.
```

with numbers that vary even if you do not redefine the logicals as suggested, then you should apply the Mwm*clientAutoPlace fix.

# 2 Changes for Alpha AXP and Integrity server Systems

This section describes the changes necessary to use HP GKS . The escape, input initialization, and input inquiry functions were changed in DEC GKS Version 5.1 to expect a structure for the data record argument. The data record format was modified because of the 64-bit addresses and 64-bit long integers on the DIGITAL UNIX (formerly known as DEC OSF/1 AXP; later as Compaq Tru64 UNIX, then HP Tru64 UNIX) system. Old applications may not have to be recoded at this time for the OpenVMS and ULTRIX platforms. However, HP recommends that you recode the affected functions to the new format to gain maximum portability between supported platforms. Applications *must* be recoded for the UNIX platform.

## 2.1 Changes to Escapes

The addition of 64-bit addresses and data types required changes to the escape data record. In previous versions of DEC GKS, pointers were returned in the integer array, as were some data objects (such as XIDs), which were declared long in C. On OpenVMS Alpha, OpenVMS Integrity servers, and OpenVMS VAX, an integer, an address, and a C long are all 32-bit quantities. On DIGITAL UNIX, only the integer is a 32-bit quantity; the address and the C long are both 64-bit quantities. This made it necessary to change the escape data record. Its previous form (in a C description) was as follows:

```
typedef struct {
    Gint     nints;           /* Number of integers */
    Gint     nfloats;         /* Number of floats */
    Gint     nstrings;        /* Number of strings */
    Gint     *int_array;      /* Pointer to integer array */
    Gfloat   *float_array;    /* Pointer to float array */
    Gint     *str_len_array;  /* Pointer to string length array */
    Gchar   **str_ptr_array;  /* Pointer to string pointer array */
} Gescaperecord;
```

The data structure now contains four new items:

- A count of generic pointers
- An array of generic pointers
- A count of longs
- An array of longs

The new data structure is as follows:

```
typedef struct {
    Gint        nints;             /* Number of integers */
    Gint        nfloats;           /* Number of floats */
    Gint        nstrings;          /* Number of strings */
    Gint        *int_array;        /* Pointer to integer array */
    Gfloat      *float_array;      /* Pointer to float array */
    Gint        *str_len_array;    /* Pointer to string length array */
    Gchar       **str_ptr_array;   /* Pointer to string pointer array */
    Gint        npointers;         /* Number of generic pointers */
    Gchar       **ptr_ptr_array;   /* Pointer to generic pointer array */
    Gint        nlongs;            /* Number of longs */
    Glong       *long_array;       /* Pointer to long array */
} Gescaperecord;
```

### 2.1.1 Affected Escapes

DIGITAL UNIX supports only the new escape record format. OpenVMS Alpha, OpenVMS Integrity servers, and OpenVMS VAX still support the old escape record format, but the newer format is preferred. Escapes that previously passed pointers or objects (such as XIDs) declared long in C will require require the most changes. These escapes are as follows:

- Inquire Window Identifiers (–304)
- Inquire Pasteboard Identifier (–307)
- Inquire Menu Bar Identifier (–308)
- Inquire Shell Identifier (–309)
- Inquire GDP Extent (–404)
- Set Background Pixmap (–501)
- Inquire Double Buffer Pixmap (–502)
- Inquire Background Pixmap (–503)

Table 3 describes the new escape data records.

## Table 3: Changed Escapes

| Argument | Required Value |
|---|---|
| **–304 Inquire Window Identifiers** | |
| in_data | Number of integers (1) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (ws_id) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (0) |
| | Address of long array (NULL) |
| out_data | Number of integers (0) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (NULL) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (1) |
| | Address of generic pointer array (X_display_id) |
| | Number of longs (1) |
| | Address of long array (X_window_id) |

**Table 3 (Cont.): Changed Escapes**

| Argument | Required Value |
|---|---|
| **–307 Inquire Pasteboard Identifier** | |
| in_data | Number of integers (1) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (ws_id) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (0) |
| | Address of long array (NULL) |
| out_data | Number of integers (0) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (NULL) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (1) |
| | Address of generic pointer array (Pasteboard_widget_id) |
| | Number of longs (0) |
| | Address of long array (NULL) |

**Table 3 (Cont.): Changed Escapes**

| Argument | Required Value |
|---|---|
| **–308 Inquire Menu Bar Identifier** | |
| in_data | Number of integers (1) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (ws_id) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (0) |
| | Address of long array (NULL) |
| out_data | Number of integers (0) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (NULL) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (1) |
| | Address of generic pointer array (Menu_bar_widget_id) |
| | Number of longs (0) |
| | Address of long array (NULL) |

**Table 3 (Cont.): Changed Escapes**

| Argument | Required Value |
|---|---|
| **–309 Inquire Shell Identifier** | |
| in_data | Number of integers (1) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (ws_id) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (0) |
| | Address of long array (NULL) |
| out_data | Number of integers (0) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (NULL) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (1) |
| | Address of generic pointer array (Shell_widget_id) |
| | Number of longs (0) |
| | Address of long array (NULL) |

**Table 3 (Cont.): Changed Escapes**

| Argument | Required Value |
|---|---|
| **–404 Inquire GDP Extent** | |
| in_data | Number of integers (4) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (ws_id, number_of_points, GDP_id, sizeof(GDP_data_record)) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (3) |
| | Address of generic pointer array (address of array of X values, address of array of Y values, address of GDP data record) |
| | Number of longs (0) |
| | Address of long array (NULL) |
| out_data | Number of integers (1) |
| | Number of reals (4) |
| | Number of strings (0) |
| | Address of integer array (status) |
| | Address of real array (minimum X, maximum X, minimum Y, maximum Y) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (0) |
| | Address of long array (NULL) |

**Table 3 (Cont.): Changed Escapes**

| Argument | Required Value |
|---|---|
| **–501 Set Background Pixmap** | |
| in_data | Number of integers (1) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (ws_id) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (1) |
| | Address of long array (Pixmap_id) |
| out_data | Number of integers (0) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (NULL) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (0) |
| | Address of long array (NULL) |

**Table 3 (Cont.): Changed Escapes**

| Argument | Required Value |
|---|---|
| **–502 Inquire Double Buffer Pixmap** | |
| in_data | Number of integers (1) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (ws_id) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (0) |
| | Address of long array (NULL) |
| out_data | Number of integers (0) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (NULL) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (1) |
| | Address of long array (Pixmap_id) |

**Table 3 (Cont.): Changed Escapes**

| Argument | Required Value |
|---|---|
| **–503 Inquire Background Pixmap** | |
| in_data | Number of integers (1) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (ws_id) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (0) |
| | Address of long array (NULL) |
| out_data | Number of integers (0) |
| | Number of reals (0) |
| | Number of strings (0) |
| | Address of integer array (NULL) |
| | Address of real array (NULL) |
| | Address of string lengths array (NULL) |
| | Address of string pointer array (NULL) |
| | Number of pointers (0) |
| | Address of generic pointer array (NULL) |
| | Number of longs (1) |
| | Address of long array (Pixmap_id) |

### 2.1.2 How to Use the New Format

To use the new data record format, do the following:

1. Convert your arrays of integers to data records. The include files contain language-specific escape data records.
2. Make sure you are passing both an input and an output record. In previous versions of DEC GKS, you could have a zero size output data record when no data was to be returned.
3. Load the appropriate sections of the data record for the escape to be performed. Make sure to enter 0 in all counts that are not used, and make sure unused pointers are a safe invalid value (such as NULL in C). This holds true for both the input and output data records, even if they are not used. If the escape is one of the escapes listed in Section 2.1, you must rearrange some of the data.
4. Make sure the sizes passed for the data records are correct. This is especially true if your sizes were previously hardcoded. Use a compiler or language feature (such as the C sizeof( ) operator) to get the size in bytes, because the size will vary depending on the architecture and operating system.

### 2.1.3 Documentation Changes

The main documentation changes are listed in Section 2.1. A general point is that the current documentation explicitly states sizes in bytes. These are only correct for the older style of record. The stated sizes are *not* valid in the Alpha and Integrity server system environments, due to the addition of data in the escape and input records and the change in the size of some of the data objects.

## 3 Installation Notes

HP GKS Version 7.2 for OpenVMS requires OpenVMS Integrity servers Version 8.2-1 or higher.

If you install HP GKS Version 7.2 for OpenVMS on a workstation, you must also install DECwindows Motif Version 1.5 or higher for the IVP to run successfully. Review all changes to the *Installing DEC GKS* manuals as described in the HP GKS Documentation paragraphs before installing HP GKS.

# 4 Enhancements

This section describes the enhancements included in HP GKS Version 7.2.

## 4.1 Metafile Header Format Change

The format of the metafile header line has been updated to include a four-digit year field and to be uniform across all platforms. The new format is designated Version 3. Old-format (Version 2) metafiles can still be read by HP GKS.

The new format is as follows:

```
0        1         2         3         4         5         6         7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
GKS3   username     /DEC GKS   Version 6.4  1996/07/25 3 4 3 81214 1 1
^    ^author designation (39 bytes)         ^date      ^ ^ ^ ^ ^ ^
^item header                                 format version^ ^ ^ ^ ^
.                                            item header size^ ^ ^ ^
.                                                item type size^ ^ ^
.                                                data length size^ ^
.                                                integer field size^
.                                           floating-point field size^
```

## 4.2 NIST Certification

Digital GKS was certified on Alpha AXP platforms by the U.S. National Institute of Standards and Technology, starting with GKS V6.0 on OpenVMS Alpha and GKS V6.0A on DIGITAL UNIX (formerly known as DEC OSF/1 AXP; later as Compaq Tru64 UNIX, then HP Tru64 UNIX). GKS was certified on and for a restricted subset of all supported workstations:

| WS type | Description |
|---|---|
| 231 | Motif output/input |
| 61 | Black & white PostScript |

Certain changes in the behavior of HP GKS were necessary for certification; all these changes are enabled by defining the GKS$NIST_CERTIFIED logical to be any non-empty value:

- Valid workstations are restricted to the list above; this change affects Open Workstation, Inquire List of Workstation Types, and the WDT inquiry functions.
- The error returned in certain error conditions is different:
  - Normalization transformation number invalid in Initialize Locator or Initialize Stroke;
  - Initial point(s) invalid in Initialize Locator or Initialize Stroke;
  - Number of points less than zero in Initialize Stroke;
  - Initial string larger than buffer in Initialize String;

  - Initial segment name less than or equal to zero, or initial pick identifier less than zero, in Initialize Pick or Initialize Pick 3;
  - GKS not in proper state in any of the Initialize Input routines.
- The Motif workstation is mapped at Open Workstation time.
- The number of marker types is restricted to 18.
- Separate viewport data for both current and requested viewports is maintained for inquiries.
- IRGs are not allowed to be optimized.
- Deferral functionality is changed: previously, output primitives were deferred (i.e. not sent to the workstation) if an IRG was pending. If the GKS$NIST_CERTIFIED logical is in effect, output primitives are **not** deferred.

### 4.3 Operating System Specific documentation updated

The operating system specific information files,

```
SYS$HELP:DECGKS_CBIND_OP_SPEC.PS, TXT
SYS$HELP:DECGKS_FBIND_OP_SPEC.PS, TXT
SYS$HELP:DECGKS_G3DBIND_OP_SPEC.PS, TXT
SYS$HELP:DECGKS_GBIND_OP_SPEC.PS, TXT
```

have been updated and are current as of HP GKS Version 6.5.

### 4.4 New Locator PET: -14, Device Coordinates in Raster Units

Digital GKS Version 6.3 introduces a new locator prompt and echo type, -14. This PET is identical to PET's 6 and -11, except that the position is reported in raster units.

### 4.5 New Escape: Set Fill Simplification Method

Digital GKS Version 6.3 introduces a new escape, *Set Fill Simplification Method*, to provide an application-level method of choosing the fill simplification method. The syntax of the Set Fill Simplification Method escape is:

**–140 Set Fill Simplification Method Arguments:**

| Argument | Required Value |
|---|---|
| in_data | number of integers (2)<br>number of reals (0)<br>number of strings (0)<br>address of integer array (*ws_id, fill_simplification_method*)<br>address of real array (NULL)<br>address of string lengths array (NULL)<br>address of string pointer array (NULL)<br>number of pointers (0)<br>address of generic pointer array (NULL)<br>number of longs (0)<br>address of long array (NULL) |
| out_data | number of integers (0)<br>number of reals (0)<br>number of strings (0)<br>address of integer array (NULL)<br>address of real array (NULL)<br>address of string lengths array (NULL)<br>address of string pointer array (NULL)<br>number of pointers (0)<br>address of generic pointer array (NULL)<br>number of longs (0)<br>address of long array (NULL) |

The input data record contains the following elements:

| Component | Description |
|---|---|
| ws_id | The workstation ID |
| fill_simplification_method | 0 = default<br>1 = MapKernel |

### 4.6 New Escape: Hit Test

Digital GKS Version 6.2 introduces a new escape, *Hit Test*, to provide programs with an alternative asynchronous method (other than Get Pick) for determining whether a segment is picked. For example, you can input a position asynchronously via Get Locator, then use Hit Test to detect a segment hit.

You specify a workstation, a transformation, a location, and an optional aperture as input. The location may be either a two-dimensional or a three-dimensional point in world coordinates. However, if you wish to specify an aperture (in device coordinates, as for Initialize Pick), you must specify a Z-value for the location. For two-dimensional applications, specify 0.0 as the Z-value.

The syntax of the Hit Test escape is:

### –390 Hit Test Arguments:

| Argument | Required Value |
|---|---|
| in_data | number of integers (2)<br>number of reals (2-4)<br>number of strings (0)<br>address of integer array (*ws_id, transform*)<br>address of real array (*x, y, z (optional), aperture(optional)*)<br>address of string lengths array (NULL)<br>address of string pointer array (NULL)<br>number of pointers (0)<br>address of generic pointer array (NULL)<br>number of longs (0)<br>address of long array (NULL) |
| out_data | number of integers (3)<br>number of reals (0)<br>number of strings (0)<br>address of integer array (*status, segment, pick_id*)<br>address of real array (NULL)<br>address of string lengths array (NULL)<br>address of string pointer array (NULL)<br>number of pointers (0)<br>address of generic pointer array (NULL)<br>number of longs (0)<br>address of long array (NULL) |

The input data record contains the following elements:

| Component | Description |
|---|---|
| ws_id | The workstation ID |
| transform | The normalization transformation number |
| x, y, z | The point in world coordinates |
| aperture | The pick aperture |

The output data record contains the following elements:

| Component | Description |
|---|---|
| status | The error status; 0 = no error |
| segment | The segment number |
| pick_id | The pick ID |

## 4.7 X11R5 Scalable Font Support

X11R5 Scalable Fonts are now supported. Support is enabled by setting the GKS$USE_SCALABLE_FONTS logical to the value 1. Any other value results in scalable font support *not* being enabled.

## 4.8 Inquire DBuffer Pixmap & Inquire Background Pixmap supported

The Inquire Double Buffer Pixmap (–502) and Inquire Background Pixmap (–503) escapes are now supported.

## 4.9 New Environmental Controls

### 4.9.1 GKS$FILL_SIMPLIFY_METHOD / GKS$COMPLEX_FILL_METHOD

The GKS$FILL_SIMPLIFY_METHOD / GKS$COMPLEX_FILL_METHOD logicals have been added to control fill simplification. Defining either of these logicals to be one (1) enables the MapKernel fill simplification method. MapKernel is currently supported only for two-dimensional geometry. MapKernel is intended for applications (like mapping) which involve extremely complex fill areas with holes, "bridges," and self-intersections. In general, MapKernel simplification is slower than the default method, but it will generate correct output in cases where the default method will not (see "Complex Fill Areas" in the "Problems Fixed" section).

### 4.9.2 The GKS$HATCH_SIMULATION logical

The GKS$HATCH_SIMULATION logical has been added to control hatch simulation. GKS interior hatch styles may be *simulated* in a device-independent way, or they may be drawn by the output device (*if* the output device supports hatch styles). Setting the GKS$HATCH_SIMULATION logical to any non-null value forces GKS to do the device-independent hatch simulation regardless of whether the output device supports hatch styles.

### 4.9.3 The GKS$VISUAL_CLASS logical

The GKS$VISUAL_CLASS logical has been added to control the class of visual selected for the Motif workstation types (230, 231). Normally, GKS selects the "default" visual provided by the display node's X server. However, sometimes it is desirable to override the X server's choice of visual and use a visual with different characteristics. For example, you may have an application which depends on rapid color changes. This is easily achieved without screen regeneration, via the GKS Set Color Representation call, on PseudoColor visuals. However, on displays with advanced graphics options, e.g. the ZLX(p)-E, ZLX-M, and ZLX(p)-L cards, the default is usually a TrueColor visual. TrueColor visuals do *not* support dynamic color redefinition, so each Set Color Representation call results in an implicit regeneration, dramatically impacting your application's performance. In this case, you can maximize performance by defining GKS$VISUAL_CLASS to have any of the values "P," "Pseudo," or "PseudoColor."

GKS$VISUAL_CLASS takes on string values as shown in the following table. Case is ignored.

| Value | Visual Class Selected |
|---|---|
| TrueColor | TrueColor |
| True | |
| T | |
| PseudoColor | PseudoColor |
| Pseudo | |
| P | |
| DirectColor | DirectColor |
| Direct | |
| D | |
| GrayScale | GrayScale |
| Gray | |
| G | |
| StaticColor | StaticColor |
| SC | |
| StaticGray | StaticGray |
| SG | |

### 4.9.4 The GKS$X_INPUT_MODEL logical

The GKS$X_INPUT_MODEL logical has been added to control the X input model selected for the Motif workstation types (230, 231). The X input models (No Input, Passive, Locally Active, and Globally Active) are described in the Inter-Client Communication Conventions Manual, section 4.1.7. An application's input model determines how the application handles input focus.

The GKS$X_INPUT_MODEL logical takes on string values as shown in the following table. Case is ignored.

| Value | X Input Model Selected |
|---|---|
| None | No Input |
| No_Input | |
| N | |
| Passive | Passive Input |
| P | |

| Value | X Input Model Selected |
|---|---|
| Local | Locally Active |
| Locally_Active | |
| L | |
| Global | Globally Active |
| Globally_Active | |
| G | |

The default input model in GKS is Globally Active. However, in some circumstances it is useful to use simpler input models. For example, some low-end graphics options (e.g. ZLX-E1) do not support multiple simultaneous colormaps. So, if your GKS application creates a new, different colormap, the application may not display in the correct colors unless you change the X input model to Passive or Locally Active *and* set focus to your application's window via clicking or moving the mouse pointer into it.

### 4.10 Metafile Logging

The metafile logging feature present in DEC GKS Version 4.*x* was reintroduced in HP GKS Version with improvements.

To use the metafile logging feature, define the GKS$META_LOG logical to be the filename of the file to which you wish to send metafile logging output, then run your GKS application. GKS calls will be logged to the output file as if it were a normal metafile output workstation (wstype 2) with additional data reflecting changes in deferral state and input-related transactions.

### 4.11 Unification of GKS and GKS/Japanese

HP GKS/Japanese is now integrated into HP GKS and is no longer a separate product. Japanese-specific components are included in an extra saveset in each kit: There are also some Japanese-specific example programs included in the examples saveset.

### 4.12 New Escape: Inquire Version

DEC GKS Version 6.0 introduces a new escape, *Inquire Version*, to enable programs to find out certain information about the version of GKS with which they are running. The syntax of the Inquire Version escape is:

**–399 Inquire Version Arguments:**

| Argument | Required Value |
|---|---|
| in_data | number of integers (0)<br>number of reals (0)<br>number of strings (0)<br>address of integer array (NULL)<br>address of real array (NULL)<br>address of string lengths array (NULL)<br>address of string pointer array (NULL)<br>number of pointers (0)<br>address of generic pointer array (NULL)<br>number of longs (0)<br>address of long array (NULL) |
| out_data | number of integers (5)<br>number of reals (1)<br>number of strings (1)<br>address of integer array (*error_status, IEEE/VAXfloat indicator, version_number:major, version_number:minor, version_number:ECO/MUP level*)<br>address of real array (*always_1.0*)<br>address of string lengths array (*length of version_string*)<br>address of string pointer array (*version_string*)<br>number of pointers (0)<br>address of generic pointer array (NULL)<br>number of longs (0)<br>address of long array (NULL) |

The output data record contains the following elements:

| Component | Description |
|---|---|
| error_status | The error status; 0 = no error |
| IEEE/VAXfloat | 1 = IEEE, 2 = VAXfloat |
| version:major | The number to the left of the decimal point in the version number |
| version:minor | The number to the right of the decimal point in the version number |
| version:ECO/MUP | (If present) trailing letter; 1 = A, 2 = B, etc.; otherwise zero |
| always_1.0 | Used to verify linking consistency on OpenVMS Alpha |
| version_string | "HP GKS V7.2" |

## 4.13 HP LaserJet II Support

The Hewlett-Packard LaserJet II printer, and other devices utilizing HP PCL Printer Language, are now supported. Workstation type 261 indicates a PCL device. Type $ HELP GKS for more information.

## 4.14 IEEE support

IEEE arithmetic is supported on HP GKS for OpenVMS Integrity servers only.

### 4.15 VAX Float support

VAX Float arithmetic is supported on HP GKS for OpenVMS Integrity servers, in addition to the default IEEE floating point format.

Linking can be accomplished as shown in the following examples:

```
IEEE Float
$ LINK program,SYS$LIBRARY:GKSCBND/OPT      !C binding
$ LINK program,SYS$LIBRARY:GKSFORBND/OPT    !Fortran binding
$ LINK program,SYS$LIBRARY:GKS$BND/OPT      !GKS$ binding
$ LINK program,SYS$LIBRARY:GKS3D$BND/OPT    !GKS3D$ binding

VAX Float
$ LINK program,SYS$LIBRARY:GKSCBND_VF/OPT      !C binding
$ LINK program,SYS$LIBRARY:GKSFORBND_VF/OPT    !Fortran binding
$ LINK program,SYS$LIBRARY:GKS$BND_VF/OPT      !GKS$ binding
$ LINK program,SYS$LIBRARY:GKS3D$BND_VF/OPT    !GKS3D$ binding
```

### 4.16 New Entrypoints in GKS$/GKS3D$ Bindings

Several new entrypoints are provided, beginning with DEC GKS Version 5.2, to facilitate the creation of descriptors in C, Fortran, or Ada programs using the GKS$ or GKS3D$ binding.

**Table 4: Descriptor Creation Entrypoints**

| Entrypoint Name | Description |
|---|---|
| gks$init_1d_int_desc | Initialize 1D integer array descriptor |
| gks$init_1d_real_desc | Initialize 1D float array descriptor |
| gks$init_2d_int_desc | Initialize 2D integer array descriptor |
| gks$init_2d_real_desc | Initialize 2D float array descriptor |
| gks$init_fort_2d_int_desc | Initialize Fortran 2D integer array descriptor |
| gks$init_fort_2d_real_desc | Initialize Fortran 2D float array descriptor |
| gks$init_string_desc | Initialize character string descriptor |
| gks3d$init_1d_int_desc | Initialize 1D integer array descriptor |
| gks3d$init_1d_real_desc | Initialize 1D float array descriptor |
| gks3d$init_2d_int_desc | Initialize 2D integer array descriptor |
| gks3d$init_2d_real_desc | Initialize 2D float array descriptor |
| gks3d$init_fort_2d_int_desc | Initialize Fortran 2D integer array descriptor |
| gks3d$init_fort_2d_real_desc | Initialize Fortran 2D float array descriptor |
| gks3d$init_string_desc | Initialize character string descriptor |

The interface is as follows (in C terms); parameters are described in the following table.

```
gks$init_1d_int_desc( iarray, lbound, ubound, desc );
gks$init_1d_real_desc( rarray, lbound, ubound, desc );
gks$init_2d_int_desc( i2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks$init_2d_real_desc( r2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks$init_fort_2d_int_desc( i2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks$init_fort_2d_real_desc( r2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks$init_string_desc( string, length, desc );
gks3d$init_1d_int_desc( iarray, lbound, ubound, desc );
gks3d$init_1d_real_desc( rarray, lbound, ubound, desc );
gks3d$init_2d_int_desc( i2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks3d$init_2d_real_desc( r2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks3d$init_fort_2d_int_desc( i2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks3d$init_fort_2d_real_desc( r2array, lbound1, ubound1, lbound2, ubound2, desc2d);
gks3d$init_string_desc( string, length, desc );
```

**Table 5: Parameters**

| Data Type | Name | In/Out | Description |
|---|---|---|---|
| int | *iarray | In | 1-D Array of integers |
| float | *rarray | In | 1-D Array of reals |
| int | *i2array | In | 2-D Array of integers |
| float | *r2array | In | 2-D Array of reals |
| int | *lbound | In | Lower bound |
| int | *ubound | In | Upper bound |
| int | *lbound1 | In | Row lower bound |
| int | *ubound1 | In | Row upper bound |
| int | *lbound2 | In | Column lower bound |
| int | *ubound2 | In | Column upper bound |
| int | *length | In | Length of character string |
| gks3d_1darray_desc | *desc | Out | 1-D Array Descriptor |
| gks3d_2darray_desc | *desc2d | Out | 2-D Array Descriptor |

**NOTE**

**On alignment-sensitive platforms (Alpha, RISC, Integrity server), these routines check the alignment of the descriptor (output) argument. The user is responsible for providing a suitably aligned (and sized) array or data structure for the descriptor argument. A 72-byte array will suffice, if correctly aligned. Correct alignment in Fortran for all platforms can be achieved as follows:**

```
C     Fragment of Cell Array Example
      REAL*4        CORN1(3)
      REAL*4        CORN2(3)
      REAL*4        CORN3(3)
      INTEGER*4     DESC(18)
      REAL*8        ALIGNME
      EQUIVALENCE  (ALIGNME, DESC(1))
      INTEGER*4     COLORS(100,200)
      ...
      CALL  GKS3D$INIT_FORT_2D_INT_DESC( COLORS, 1, 100, 1, 200, DESC )
      CALL  GKS3D$CELL_ARRAY3( CORN1, CORN2, CORN3,
     *                         1, 1, 100, 200, DESC )
```

### 4.17 Ada Support

Development of Ada programs using an Ada interface to the GKS$ and GKS3D$ bindings is supported on all platforms. See the GKS3D_DEFS.ADA and GKSDEFS.ADA files provided with the HP GKS kit.

**NOTE**

**Passing arguments by descriptor is not supported on all platforms. On DIGITAL UNIX and ULTRIX platforms, use the alternative subprogram declarations containing 'GKS3D_DESCRIPTOR' or 'GKS_DESCRIPTOR' parameters. Use the INIT_*_DESC routines to aid in constructing the GKS3D_DESCRIPTOR or GKS_DESCRIPTOR records.**

**For maximum portability, use the GKS*DESCRIPTOR alternatives on all platforms.**

### 4.18 Pascal Support

The environment files SYS$LIBRARY:GKS3D_DEFS.PEN and SYS$LIBRARY:GKSDEFS.PEN are now provided with the HP GKS kit. The GKS3D_DEFS.PAS and GKSDEFS.PAS source files are still provided as well.

Should it become necessary to rebuild the GKS3D_DEFS.PEN or GKSDEFS.PEN file, some additional steps are necessary:

1. Create a file (usually an empty file, unless your installation has specific requirements):

```
$ OPEN/WRITE EMPTYFILE EMPTY.FILE
$ CLOSE EMPTYFILE
```

2. Define a logical name GKS3D_DEFS_MODULE_INIT or GKSDEFS_MODULE_INIT pointing to the file just created:

```
$ DEFINE GKS3D_DEFS_MODULE_INIT EMPTY.FILE
```

3. Compile the GKS*DEFS.PAS file as usual:

```
$ PASCAL/ENVIRONMENT=SYS$LIBRARY:GKS3D_DEFS.PEN GKS3D_DEFS.PAS
```

The environment files (.PEN) provided with the GKS Development kit are IEEE compatible. For the VAX float compatible .PEN files, complete Step 1 and Step 2, and then specify the /FLOAT qualifier in the PASCAL command (Step 3) as shown:

```
$ PASCAL/ENVIRONMENT=SYS$LIBRARY:GKS3D_DEFS.PEN/FLOAT=<float_type>  GKS3D_DEFS.PAS
```

### 4.19 Encapsulated PostScript

HP GKS Version 7.2 supports encapsulated PostScript. Use either workstation type 65 (black and white) or 66 (color).

### 4.20 ISO-Latin1 Character Support

The PostScript device handler supports ISO-Latin1 characters. To enable this feature on OpenVMS systems, set the GKS$ISO_ENCODE logical to 1. By default, this feature is disabled (0).

Font IDs –101 to –129 are supported for ISO-Latin1. Spacing and positioning errors may occur if you use the other font IDs.

No special setup is required to output ISO-Latin1 characters on OpenVMS printer queues.

## 4.21 Motif Support

This release supports DECwindows Motif. All the functionality provided by the DECwindows workstation types (21*x*) is provided by the Motif workstation types (23*x*).

The Motif workstation types are as follows:

- 230—Output only
- 231—Output and input
- 232—X drawable
- 233—Motif widget
- 330—Japanese Output only
- 331—Japanese Output and input
- 332—Japanese X drawable
- 333—Japanese Motif widget

### 4.21.1 Using Motif Window Manager Resources

The following resources for the Motif window manager are useful for controlling the focus of windows created by HP GKS:

- Mwm*keyboardFocusPolicy (Explicit, Pointer)—Set this resource to Explicit to determine window focus by clicking on the window or border. Set it to Pointer to determine window focus by the location of the pointer.
- Mwm*autoKeyFocus (True, False)—If keyboardFocusPolicy is set to Explicit, set this resource to True to return focus to the window originally in focus. Set it to False to avoid giving focus to any window when the window currently with focus is iconified or removed.
- Mwm*deiconifyKeyFocus (True, False)—If the keyboardFocusPolicy is set to Explicit, set this resource to True to give a window focus when it is restored from an icon. Set it to False to retain the current focus when a window is deiconified. Because GKS iconifies a window when it is created and then deiconifies it, use the deiconifyKeyFocus resource to control the initial focus of the application window.
- Mwm*focusAutoRaise (True, False)—Set this resource to True to raise a window to the top of the stack when it is given focus. Set it to False to retain the current stacking order when a window is given focus.
- Mwm*startupKeyfocus (True, False)—If the keyboardFocusPolicy is set to Explicit, set this resource to True to give input focus to a window when it is mapped. Set it to False to avoid giving focus to a window when it is mapped.
- Mwm*clientAutoPlace (True, False)—You may need to set this resource to False to ensure correct window placement when you are running your application on an OpenVMS client and displaying to an OpenVMS server running Motif Version 1.2.

### 4.21.2 GKS Motif Problems

The following problems exist with the HP GKS Motif device handler:

- On OpenVMS systems, using the "CLOSE" option on the title bar kills the application. There is no method to disable this behavior.
- String input device number 3 (ASCII) with echoing ON does not trigger on each character (as it does in NOECHO mode).
- Choice devices 2, 3, 4, and 5 with echoing ON respond to their respective keyboard and mouse buttons, as well as selection of the menu with the mouse. They should only respond to the keyboard.
- Choice device 9 with NOECHO appears to have no possible trigger.
- Choice device 9 with echo ON triggers with any mouse movement over the window region.
- The RESET button is an item in the menu bar. This is incorrect for Motif style guide conformance, which requires the button to be an item in a pulldown menu with the name of the pulldown menu listed in the menu bar. This will be addressed in a future release.

### 4.21.3 Avoiding Window Problems

If you are using the Motif workstation types , you may see the following messages when running applications:

```
HP GKS - 'MENU BAR SIZE' attribute incorrect: use
"DEFINE GKS$DECW_MENU_BAR_SIZE 30" and restart application.

HP GKS -'BORDER SIZE' attribute incorrect: use
"DEFINE GKS$DECW_BORDER_SIZE 16" and restart application.

HP GKS -'TITLE SIZE' attribute incorrect: use
"DEFINE GKS$DECW_TITLE_SIZE 17" and restart application.
```

If you see these messages, HP GKS is making assumptions about the size of the window manager borders, title bar, and the application menu bar, based on the window manager and the resolution of the screen. HP GKS uses these values to calculate the maximum display space remaining for the application to use. If for some reason these assumptions are not correct, HP GKS displays these error messages. Digital suggests that you follow the error message instructions and restart the application. If you do not follow the instructions, you may have the following problems:

- The default positioning will not be correct.
- The application window may not fill the screen (if requested).
- The RESET button may not properly reset the window to the correct location and size.
- Scroll bars may be displayed when you use the default window size.
- Resizing the window may result in parts of the graphic output being clipped.

Digital does not consider these messages and the user-actions required to be a long term solution. Digital is continuing to work with OSF to develop a better solution.

# 5 Areas of Nonconformance

This section describes areas of nonconformance in HP GKS.

## 5.1 Initial Input Prompt Position for STROKE and LOCATOR

HP GKS implements PETs in a windowing environment using a pointer device (a mouse). It is against the philosophy of a windowing system to position the mouse from the application. In HP GKS, the initial locator and stroke positions are not prompted at the initial position, as required by the GKS standard.

# 6 Restrictions

This section describes the HP GKS Version 7.2 restrictions.

## 6.1 General Restrictions

This section describes the general restrictions to HP GKS.

### 6.1.1 Output Drawing Performance Degradation

When input is active (SAMPLE or EVENT mode) and the active PETs have to be simulated by HP GKS, output drawing performance is slow. Digital recommends that you turn off input before you call a group of GKS output primitives.

### 6.1.2 Line Cap Style

The line cap style attribute is applied only to lines that have the SOLID linestyle attribute.

### 6.1.3 Limitation with STRING Precision Text Strings

If text of precision STRING contains invalid character codes, the text is drawn with the default stroke font number (1).

### 6.1.4 Fill Area Set

The fill area interior style is always simulated if the fill area set includes more than one fill area. This results in reduced performance. Also, the fill area interior style PATTERN is not supported with the function FILL AREA SET if there is more than one fill area defined in the set.

### 6.1.5 Fill Area and Fill Area Set

FILL AREA and FILL AREA SET calls not defining planar fill areas can result in an unexpected failure of HP GKS. HP GKS only provides support for planar fill areas and fill area sets (as required by the standard).

### 6.1.6 Fill Area Interior Style Pattern

The following pattern functions are provided, but have no effect on output:

- SET PATTERN SIZE
- SET PATTERN REFERENCE POINT AND VECTORS
- SET PATTERN REFERENCE POINT

The fill area interior style PATTERN is not supported with the function FILL AREA SET if there is more than one fill area defined in the set.

### 6.1.7 GDP Restricted Text

The Restricted Text GDP function is not supported by HP GKS Version 7.2.

### 6.1.8 Widget Callback Procedure Name and Motif

When using widget callback procedures for the Motif workstation type, avoid using the name "gfx_create_proc". This is already used by the Motif UIL files and a name collision will occur.

The workstation description table inquiry functions for the Motif workstation types require operating states WSOP, WSAC, and SGOP. To call these inquiries while not in these states, use escape –110, Associate Workstation Type Connection ID.

### 6.1.9 Fill Areas—SIZE RESTRICTION REMOVED

**NOTE**

**THIS RESTRICTION HAS BEEN REMOVED beginning with Digital GKS Version 6.3:**

Formerly, fill areas on Motif workstations were required to fit in one Xbuffer. Typically, a fill area of less than 2048 points complied with this requirement.

**NOTE**

**This restriction did not apply to fill areas with interior style HOLLOW.**

### 6.1.10 Copying DDIF Files from DIGITAL UNIX or ULTRIX to OpenVMS

On OpenVMS, files containing DDIF information are expected to have the DDIF semantic attribute set. If they do not have this attribute, they will be ignored by DDIF tools.

After copying a DDIF file from DIGITAL UNIX or ULTRIX to OpenVMS, you should check the semantic attribute with the following command:

```
$ DIR/FULL
```

If the file semantic attribute is not DDIF, you must set it with the following command:

```
$ SET FILE/SEMANTIC=DDIF filename.doc
```

### 6.1.11 Writing To VT330 or VT340 Devices

The most effective use of VT330 and VT340 devices occurs when HP GKS keeps the devices in ReGIS graphics mode while the workstation remains open. However, ASCII text sent to the screen using a FORTRAN WRITE (or similar statements in other languages) is interpreted as a ReGIS command that can leave the terminal in an unpredictable state.

To avoid this problem, never put text on the screen when the workstation is open unless you are using the GKS TEXT or MESSAGE function. When using HP GKS with the VT330 or VT340, Digital recommends that the error file specified in the OPEN GKS function be directed to a file. Digital also recommends that you use the OpenVMS command SET TERMINAL/NOBROADCAST to prevent messages from being sent to the terminal by OpenVMS, which will cause similar problems.

### 6.1.12 Text Path LEFT on Tektronix–4207 Terminals

Due to firmware problems, text path LEFT using font –109 in STRING precision does not work properly on some Tektronix 4207 terminals. If you have this problem, have Tektronix update the firmware in your terminal to Version 12.2 or later.

### 6.1.13 Ada Language Restrictions

Applications using Ada tasks should make sure that all HP GKS calls are performed from the same Ada task, or that proper synchronization is done.

### 6.1.14 Multitasking

Applications consisting of multiple tasks or threads, such as OpenVMS ASTs or DIGITAL UNIX or ULTRIX signals, must make sure that GKS calls are all performed from the same code thread, or that proper synchronization is done.

## 6.2 OpenVMS-Specific Restrictions

The following sections describe the OpenVMS operating system restrictions for HP GKS Version 7.2.

### 6.2.1 Opening Multiple Motif Workstations

Applications that open multiple Motif workstations of type 230, 231, 240, or 241 must use the same value for the workstation type each time. There is a limitation in the device handlers that breaks X event handling if the workstation type value is different. For example, opening two workstations using type %x000a00e7 for each is acceptable, but opening one with %X000000e7 and another with %x000a00e7 is not (workstation type 231 with the default number of colors and with 10 colors). This will be addressed in a future release.

# 7 Fixed Problems

This section describes problems fixed since the last release.

## 7.1 Problems fixed in V7.2

HP GKS Run-Time kit does not recognize the GKS-RT license.

## 7.2 Problems fixed since V6.5

### 7.2.1 Incorrect Size Returned by Get Item Type function

The Get Item Type From Metafile function returned a size that was slightly too large, and inconsistent across Alpha and VAX systems. The excess was one (1) byte on VAX systems, and four (4) bytes on Alpha systems.

### 7.2.2 Extra Page or Output File

For NIST conformance, the behavior of Set Color Representation was changed such that setting the representation of the first color index caused the display to become NOT EMPTY even if nothing had been drawn yet.

This change caused hardcopy devices to emit a blank page (e.g. wstype 62, PostScript) or an extra output file (e.g. wstype 66, Encapsulated PostScript) if Set Color Representation was called for color #0 before anything was drawn.

This behavior has now been placed under the control of the GKS$NIST_CERTIFIED logical. If this logical is NOT in effect, no blank page or extra file will be generated.

### 7.2.3 Choice Menu Item Not Selectable

In some situations, an item on a choice menu that should have been active (selectable) was not selectable.

This behavior occurred in situations like the following:

1. Application brings up a choice menu in Request Mode;
2. User clicks on one of the choices and leaves the mouse in place;
3. Application brings up another choice menu, replacing the first one;
4. The choice item under the mouse pointer is now unselectable; the user must move the mouse away and then back to make the item selectable.

### 7.2.4 Rangecheck Error when Printing or Converting PostScript file

A rangecheck error occurred when a Japanese PostScript file was sent to certain printers or to a PostScript-to-PDF converter like Adobe Acrobat Distiller.

The problem was that newer printers and the conversion tools expect CID fonts, rather than the OCF fonts emitted by GKS.

HP GKS now supports CID fonts.

## 7.3 Problems fixed since V6.4

### 7.3.1 Crash when Clipping Complex Fill Area

A crash sometimes occurred when certain vertices of complex fill areas coincided with one of the clipping planes.

### 7.3.2 String Input from Keypad

During String input in NOECHO mode on workstation type 231, characters input from the numeric keypad were not recognized.

### 7.3.3 Dot Polymarkers on HPGL Devices

Dot polymarkers are actually simulated on HPGL devices: they are drawn as short lines. However, the lines drawn were too long, resulting in markers appearing as slashes.

The length of the line drawn to represent a dot on HPGL devices has now been limited to 3.0 times the device increment in the X direction.

### 7.3.4 Metafile Read Item with Maximum Record Length = Zero

The GKS Standard specifies that a maximum record length of zero may be passed to the metafile Read Item function in order to skip an item. HP GKS was inappropriately returning error 166 (Maximum item data record length is invalid).

### 7.3.5 Picking DOT Polymarkers

If a dot-type polymarker primitive in a segment contained multiple markers, only the first dot could be detected by pick input or hit test.

### 7.3.6 String and Choice Devices simultaneously active in Event Mode

If a string device and a choice device 3 (keyboard choice input) were both active in event mode at the same time, the string device sometimes returned garbage.

### 7.3.7 Choice Device 3 and Widget Workstations (233, 243)

Choice device 3 (keyboard input) on widget workstation types (233, 243) caused a crash.

### 7.3.8 GQHRF (Inquire HLHSR Facilities)

The Fortran binding Inquire HLHSR Facilities function GQHRF did not return correct data or a correct error code.

### 7.3.9 Set Segment Detectability on Small Segment

A Set Segment Detectability operation on a segment with very small extents (e.g. a segment consisting of a single dot polymarker) could cause a crash.

### 7.3.10 Small Text

A cluster of problems involving small text have been fixed. These problems resulted in characters being drawn as horizontal lines or not being drawn at all. The problems could occur in any of the following situations:

- Output to workstation from WISS via Associate Segment with Workstation or Copy Segment to Workstation;
- Metafile output;
- Metafile interpretation.

### 7.3.11 Fill Area Size Restriction Lifted on Motif Workstations

The restriction that a fill area was required to fit in one X buffer on Motif workstations has been lifted. Formerly, fill areas on these workstations were limited to about 1800 points on OpenVMS systems, and about 8000 points on UNIX / OSF/1 or ULTRIX systems. Any fill area larger than the limit was simplified by removing points while attempting to minimize the impact on the appearance of the fill area.

In case you desire the old fill area reduction behavior for performance (or other) reasons, Digital GKS Version 6.3 (and higher) recognizes a new logical: GKS$MAX_FILL_POINTS. Set it to the maximum number of points you wish to allow before reduction takes place. (However, if you set it below the pre-Version 6.3 limit, the effect will be as if you had set it to that limit.)

### 7.3.12 Hit Test with Stroke Precision Text or Polylines

The hit test escape had problems with stroke precision text or polyline primitives.

## 7.4 Problems fixed since V6.1

### 7.4.1 Complex Fill Areas

Extremely complex fill areas (with holes, "bridges," and self-intersections) were sometimes not filled correctly when clipped. This problem is fixed, in the case of two-dimensional geometry only, by the introduction of the MapKernel fill simplification method. MapKernel fill simplification is enabled by defining the logical GKS$FILL_SIMPLIFY_METHOD or GKS$COMPLEX_FILL_METHOD to be one (1).

### 7.4.2 String Input on Tektronix Terminals

On Tektronix 41xx and 42xx terminals, string input characters were not echoed until ENTER was pressed.

### 7.4.3 Degenerate thick lines

Degenerate thick lines (i.e. only one point specified) caused a crash.

### 7.4.4 Status from Get Pick in Dynamic Environment

In a dynamic environment in which the display is constantly changing (e.g. segments being repeatedly created, deleted, and re-created), a Get Pick operation might be attempted at a time at which the display is locked because the picture is incomplete. Get Pick will now return a status of NO PICK (GKS(3D)$ bindings: GKS(3D)$K_STATUS_NOPICK, C binding: GP_NOPICK, Fortran binding: GNPICK) when this situation occurs.

### 7.4.5 String Precision Text on Regis Terminals

String-precision text drawn with multiple Text calls was sometimes drawn with incorrect width.

### 7.4.6 Old-format Escapes

Old-format escapes are now fully supported.

### 7.4.7 Crash with Small Text

A crash occurred when extremely small text was drawn on some wstypes, e.g. Tektronix 41xx/42xx (80, 82-88).

### 7.4.8 Choice/Keyboard Input Sometimes Not Recognized

Choice/keyboard input was sometimes not recognized on wstype 231; for example, choice device 3 with echoing off only responded to keyboard input when the pointer was over a visible portion of the GKS window.

### 7.4.9 Deleting Segments on Regis Terminals

Deleting a segment on a Regis terminal did not erase the screen.

### 7.4.10 Fortran Binding / Open Workstation

When the Fortran Binding Open Workstation function (gopwk) was called with a non-default connection identifier, some values of the connection identifier were not interpreted correctly.

## 7.5 Problems fixed since V5.3

### 7.5.1 Metafile Changes

The following changes/fixes pertaining to metafile input/output were made:

- Attribute settings in the GKS state list cannot be deferred when a metafile output workstation is active.
- The clip flag and NDC viewport must be handled differently when writing a GKSM metafile.
- Ensured that Inquire Character Up Vector returns the user-specified character up vector except when reading a metafile, in which case only the direction of the vector is meaningful.
- Fixed Set Pattern Size in GKSM metafile.
- Text primitives coming from metafile input must have their text geometry attributes mapped from World Coordinates to Normalized Device Coordinates.

- The Message function must exist for the Metafile Input workstation, and must return ERROR_33 (Specified WS is of category MI).

### 7.5.2 Read/Interpret Metafile with Huge Records

A crash occurred on OpenVMS systems when attempting to read/interpret a metafile with a huge (>32,767 bytes) polyline record.

### 7.5.3 Color PostScript Cell Array

Cell arrays output via the PostScript image primitive sometimes resulted in output not intelligible to PostScript interpreters.

### 7.5.4 Text Extents with PostScript Courier Fonts

Extents were incorrect for text written in Courier fonts –109, –110, –111, and –112 (ISO encoding) and –111 & –112 (non-ISO encoding).

### 7.5.5 CGM Polylines/markers Broken up into Multiple "Line"/"Marker" Commands

Polylines and polymarkers consisting of more than 255 points were broken up into multiple "Line"/"Marker" commands.

### 7.5.6 CGM and DDIF Text Always Stroked

Text output to CGM and DDIF workstations was always stroked in GKS V5.*x*, regardless of precision. String precision text is now emitted as CGM or DDIF text, respectively, as in GKS V4.*x*.

### 7.5.7 Out of Memory When Drawing GDPs in High-Precision CGM

Drawing GDPs to the high-precision CGM workstation resulted in an Out-of-Memory error because of the large number of points computed. The number of points computed for GDPs in high-precision CGM is now limited to the same number of points as for low-precision CGM. A new logical, GKS$MAX_GDP_POINTS, is provided for the purpose of overriding this limit.

## 7.6 Problems fixed since V5.2

### 7.6.1 Three-Point Arc GDPs Sometimes Not Drawn Correctly

GDPs –107 and –339 (arc and filled arc, respectively, defined by three points on the circumference) were not drawn correctly if the points were specified in clockwise order.

### 7.6.2 Graphics Not Redrawn on Terminals After Choice Menu Erased

Graphics obscured by choice menus on terminals were not getting redrawn once selection was made and the choice menu was erased.

### 7.6.3 144 DPI Now Works on Sixel Devices

Attempting to specify 144 Dots Per Inch for Sixel devices resulted instead in graphics being printed at 90 DPI.

### 7.6.4 WISS Device Lost ASFs

Aspect Source Flags were lost when a segment was associated.

### 7.6.5 Line Style / Line Join Problem on LJ250

Line join problems occurred with non-solid lines drawn on the LJ250 with width other than the default line width.

### 7.6.6 View Input Priority and Locator 3/String 3 Input

The Set View Transformation Input Priority function effected no change in the view transformation input priority. Thus, the view with highest input priority by default, view 0, was always used to perform the inverse view mapping of locator 3 and stroke 3 points.

### 7.6.7 PostScript and ISO Encoding

#### 7.6.7.1 Crash with Font –101 and Characters with Eighth Bit Set

With GKS$ISO_ENCODE set to 1, text in Font –101 containing characters with the eighth bit set (e.g. characters with diacritical marks used in European languages) caused GKS to crash.

#### 7.6.7.2 Bad Text Extents with Font –103

With GKS$ISO_ENCODE set to 1, Inquire Text Extents returned wrong values for text in Font –103.

### 7.6.8 Crash with Soft Clipping, Stroke Precision Text followed by Polyline

A crash occurred when soft clipping was activated, stroke precision text was drawn, and then a polyline was drawn.

### 7.6.9 Inquire Shell/Pasteboard/Menu Bar ID Escapes

The Inquire Shell ID, Inquire Pasteboard ID, and Inquire Menu Bar ID escapes did not function properly.

### 7.6.10 Font Memory Not Freed

Stroke Precision font memory was not deallocated by Close Workstation.

### 7.6.11 Reading and Writing 2-D Metafiles

Some items read from a GKSM (2-D) metafile could not be written to another GKSM metafile.

### 7.6.12 Text Extents on workstation 23x

Text extents were computed incorrectly for some sizes of some string or character precision fonts.

### 7.6.13 Coordinate Display with Locator PET -11

Locator prompt and echo type -11 displayed the current position in normalized device coordinates. The current position is now displayed correctly in world coordinates.

### 7.6.14 Workstation Size and Position on workstation 23x

When an application opened two or more workstations on the same screen, the position and/or size of one or more of the windows was incorrect in some instances.

### 7.6.15 Workstation Viewport Positioning on workstation 23x

Under certain circumstances, the workstation viewport was moved when no user action or application Set Workstation Viewport occurred.

### 7.6.16 Clipping on workstation 23x

Clipping was done incorrectly in some cases, causing some or all graphics not to appear.

### 7.6.17 Incorrect Error Return from Set Pattern Representation

When Set Pattern Representation was called from Fortran, an incorrect error return occurred.

### 7.6.18 Metafile Output of Segments under Non-Identity Transform

The Associate Segment, Copy Segment, and Insert Segment operations on a metafile output workstation resulted in incorrect coordinate output for any segments with a non-identity transform.

### 7.6.19 Floating Overflow in Select Transform

A Floating Point Overflow occurred in Select Transform when window coordinate ranges were large (e.g. Y = [-1e20, +1e20]).

### 7.6.20 Infinite Loop after Calls to Set Color Representation

Certain sequences of calls to Set Color Representation caused GKS to loop infinitely.

### 7.6.21 Metafile Output of Text in Segments under Non-Identity Transform

If text in a segment was subjected to a non-identity transform, then written to a metafile (workstation type 2), bad direction vector information was written, resulting in the text not being visible on subsequent read/display operations.

### 7.6.22 PostScript Cell Array File Size

Because the CELL ARRAY primitive was simulated by fill areas, PostScript output of large cell arrays took inordinate amounts of time and disk space.

### 7.6.23 Inquire GDP Extent Escape

The Inquire GDP Extent escape returned excessively generous bounds in certain cases.

### 7.6.24 Center-Point-Angle Arc GDPs

The Center-Point-Angle Arc GDPs worked incorrectly.

### 7.6.25 OpenVMS-Specific Problems Fixed

#### 7.6.25.1 AST Queue Overflow

Some applications could cause AST queue overflow, resulting in a fatal I/O error in the X server causing the application(s) to crash. If your application exhibited this behavior, define the GKS$FAST_PICK logical:

```
$ DEFINE GKS$FAST_PICK 1
```

### 7.6.26 Font Set Incorrectly in Encapsulated PostScript with ISO Encoding

Encapsulated PostScript file header setup was done after ISO-Latin1 setup, causing text to be output using the default font rather than the user-specified font.

### 7.6.27 Workstation Failed to Deiconify

Motif (23x) workstations would sometimes fail to deiconify when displaying to another machine, especially when the server machine was faster than the client.

### 7.6.28 Crash in Inquire Default Choice Data

The INQUIRE DEFAULT CHOICE DATA function sometimes caused a crash.

### 7.6.29 Crash after Metafile Interpretation

A SET WINDOW operation following metafile interpretation caused a crash.

### 7.6.30 Fill Area Set GDP Did Not Work Correctly

The Fill Area Set GDP did not work correctly: when two fill areas were drawn, one inside the other, the inside fill area was not seen.

### 7.6.31 Performance Problem when Reading Large Metafile Records

Large metafile input records caused a substantial (8x) performance degradation from GKS V4.2 to GKS V5.0. Performance in this context is now comparable to GKS V4.2.

### 7.6.32 Inquire Default Locator Data PET list

The PET returned by the INQUIRE DEFAULT LOCATOR DATA functions was not complete because it did not include PET –13 (locator cursor segment).

### 7.6.33 Performance Problem with Locator Cursor Segment

The performance problem seen with the locator cursor segment when double buffering is enabled has been improved.

### 7.6.34 Output Drawing Performance Degradation while Input Active

The output drawing performance degradation while input is active (in SAMPLE or EVENT mode) has been improved. The performance degradation is now restricted to the simulated PETs.

### 7.6.35 DEC GKS V4.*x* FORTRAN Binding Compatibility

The function GQEWK was corrected to allow applications linked with DEC GKS Version 4.*x* to run on HP GKS Version 7.2

### 7.6.36 DEC GKS V4.*x* GKS$ Binding Compatibility

The array and string descriptor data structure check was changed to allow for DEC GKS Version 4.*x* type NULL descriptor, defined by either a 0 length or NULL pointer element.

### 7.6.37 Cell Array Primitive in CGM files

The color attributes of the cell array primitive were not being entered into the CGM file.

### 7.6.38 Text Attribute Lost on Insert Segment

The character up vector attribute was lost for text primitives that were inserted using the INSERT SEGMENT function.

### 7.6.39 Corrections to Segment Highlighting and Pick Echo

Graphics problems with segment highlighting and pick echo have been corrected for Motif devices.

Resize did not occur at all widget levels for the widget workstation type.

### 7.6.40 Blank Pages in DDIF file

DDIF files used to start off with 2 blank pages.

### 7.6.41 Memory Allocation Error

The CLOSE WORKSTATION function occasionally reported error –3502. This error occurred if the application explicitly deleted all remaining segments before the workstation was closed.

### 7.6.42 Escape Function and the FORTRAN Binding

A number of problems involving the FORTRAN binding escape function were fixed. The errors occurred when HP GKS overwrote the output data record with invalid data.

### 7.6.43 Segment Highlighting Using the Color Method

Using the COLOR method of segment highlighting could result in GKS using an invalid clip region for display list redraws. It appeared as though some segments were not being redrawn.

### 7.6.44 STRING Precision Text on PostScript Device

Some text characters were not displayed when using STRING precision text on PostScript devices.

### 7.6.45 Patterns Unsupported on PostScript Workstation 62

Patterns for PostScript workstation type 62 (color PostScript) are not supported.

### 7.6.46 PostScript Cell Array Output

Color output of the CELL ARRAY function now works correctly.

### 7.6.47 CGM Clear Text Encoding

Negative numbers occurring in the CGM element "RealPrec" had invalid CGM syntax (a space between the minus sign and the numerals is invalid CGM syntax).

### 7.6.48 Metafile Input Workstation and FORTRAN Binding

The FORTRAN binding call GCLWK (CLOSE WORKSTATION) used to open the metafile with APPEND mode. This caused an “Attempted File Access” security alarm, if the file had read-only access. The problem was fixed by using read-only mode to open the file.

### 7.6.49 Text Clipping on DECwindows Device

There was an incompatibility between DEC GKS–3D and HP GKS in the way string precision text was clipped. This incompatibility has been fixed.

### 7.6.50 FORTRAN Data Records

There was a problem with the way GKS manipulated FORTRAN data records.

### 7.6.51 FORTRAN Binding and GQEGDP

The GQEGDP function only returned the first element in the list of GDPs, regardless of the list element requested by the application.

### 7.6.52 Segment Highlighting

Two problems with segment highlighting methods were fixed. One concerned an incorrect change of color when an image was iconified and deiconified. The other problem concerned the size of the extent box which was drawn.

### 7.6.53 Fill Area Primitive on PostScript Device

A problem was fixed concerning fill areas made up of a lot of points (over 500) on the PostScript device.

### 7.6.54 Input Device Prompt and Echo Graphics

There was a problem drawing PETs when an input device was in SAMPLE or EVENT mode. The graphics associated with output primitives corrupted the display of the prompt and echo graphics.

### 7.6.55 Text Primitives and the Normalization Transformation

An error could result due to precision problems when the text primitive was used. If the relative scales of the world window and the normalized device coordinate viewport were vastly different, it was possible for the associated text vectors to have length zero. In the reported test case, the world window was of size 256,000 units and the NDC viewport was 1 unit.

### 7.6.56 Leading Blank Pages on PostScript Device

It was possible to have 2 leading blank pages on the PostScript device.

### 7.6.57 Inquire List of Escapes

The escape list returned by DEC GKS–3D differed from the list returned by HP GKS.

### 7.6.58 Multiple Redraws When HLHSR is ON

Multiple redraws of the display list would occur when HLHSR mode was ON, resulting in intermediate frames being generated. This was especially a problem for hardcopy devices such as the PostScript device.

### 7.6.59 Text Primitive on PostScript Device

The line width scale factor attribute was erroneously being applied to text primitives as well as to line primitives.

### 7.6.60 String Input Focus for Motif Fixed

Focus is automatic for string input devices. Clicking in the window before input is no longer necessary.

### 7.6.61 PostScript Fixed Problems

This section lists corrected PostScript problems.

#### 7.6.61.1 Color PostScript and Portrait Mode

Color PostScript in portrait mode used to produce unprintable output.

#### 7.6.61.2 Color PostScript Color Table

The color PostScript workstation type had an incorrect default color table.

#### 7.6.61.3 Color PostScript Background Color

The color PostScript workstation type did not set the background color correctly.

#### 7.6.61.4 Hatched Interior Style and Portrait Mode

The hatched interior style produced incorrect hatch styles when portrait mode was used.

# 8 Known Problems

This section describes known technical problems in this release.

## 8.1 General Known Problems

This section describes the known general technical problems with HP GKS Version 7.2.

### 8.1.1 DDIF and Pattern Interior Style

The same pattern appears visually different on OpenVMS and DIGITAL UNIX or ULTRIX systems.

### 8.1.2 DDIF Files and DECwrite

When a DDIF file is imported into the DECwrite product, the image is redrawn 3 times.

There is a known problem with the hardware clipping of GDPs against the workstation viewport on Motif devices. The workaround is to use software clipping. Software clipping is the default for HP GKS Version 7.2.

### 8.1.3 Spikes Appear for Some Characters

It is possible for spikes to appear when drawing STROKE and CHARACTER precision text strings. The characters A, M, and W have this problem. The problem occurs on OpenVMS systems containing the SPX graphics processor, and on all DIGITAL UNIX and ULTRIX systems. The output window must be partially obscured for the problem to occur. The problem lies in the Xcfb server. The server has problems when drawing zero-width capped lines.

### 8.1.4 Clipping of Borders

Depending on the current workstation transformation or whether double buffering is on, the border pixels may be clipped. If clipping occurs, a line drawn along the workstation window border may not appear on the output device.

### 8.1.5 Inquiry Workstation Functions

If you call inquiry workstation functions before workstation types 232 or 233 are open, the program ends unexpectedly.

### 8.1.6 PostScript Workstations 61 and 62 Leading Blank Pages

The PostScript workstations 61 and 62 generate leading blank pages if the workstation transformation is changed prior to output.

# 9 Compatibility

This section describes compatibility issues.

## 9.1 DEC GKS Version 4.*x* to HP GKS Version 7.2 Compatibility

### 9.1.1 Value of Constants in Include Files

The values of the line join style constants were incorrect. See the appropriate include file for the new values of the line join constants.

The DEC GKS Version 4.2 DECwindows connection ID notation (where you add a character to the server connection to differentiate multiple connections to the same server) is not required and not supported by HP GKS Version 7.2. HP GKS Version 7.2 allows multiple workstations to use the same workstation type and connection identifier pairs for Motif workstations. For example, the character *a* in the server connection string *nodename::0.a* is no longer supported.

### 9.1.2 GDPs

All the DEC GKS Version 4.2 GDP functions are supported by HP GKS Version 7.2, except the Restricted Text GDP.

### 9.1.3 Metafile Output Format

HP GKS Version 7.2 generates three-dimensional (DEC GKS–3D) metafiles by default. The item headers of three-dimensional metafiles contain the string "GKS3". The geometrical information is stored as *x*, *y*, and *z* coordinates and the following items have a different item number compared to the two-dimensional (HP GKS) metafile: ASF, CELL_ARRAY, COLOUR_REP, GDP, PATTERN_REFPT, and PICK_IDENTIFIER.

HP GKS Version 7.2 can also generate two-dimensional (DEC GKS Version 4.*x*) metafiles. The item headers of two-dimensional metafiles contain the string "GKSM", and the geometrical information is stored as *x* and *y* coordinates. Any attempt to make a three-dimensional call to this metafile output workstation generates an error. The two-dimensional metafile output can be enabled by defining the GKS$METAFILE_TYPE logical to "GKSM", and using the metafile output workstation type 2. See the *DEC GKS Developer's Guide* for more information on metafiles.

### 9.1.4 READ ITEM FROM GKSM

DEC GKS Version 4.2 reversed the two arguments MILDR and MLDR of the function READ ITEM FROM GKSM. This was corrected in HP GKS (then known as DEC GKS) Version 5.0.

### 9.1.5 FORTRAN Binding

The DEC GKS Version 4.2 FORTRAN binding is upwardly compatible with HP GKS Version 7.2. The Version 4.2 FORTRAN include file *gksdefs.for* is provided with Version 7.2 for compatibility. However, Version 7.2 also provides *gks.f* which defines the enumerated constants as specified by the ISO FORTRAN Binding to GKS-3D. Digital recommends that application developers use *gks.f* when developing FORTRAN programs which call the FORTRAN binding. Existing Version 4.2 FORTRAN applications are encouraged, but not required, to substitute the corresponding constant names from *gks.f* for the GKS$*xxxx* constants from

*gksdefs.for*, and to include *gks.f* instead of *gksdefs.for*. Note that the values of the constants for the line join types and the no-choice status are different in the two include files.

#### 9.1.5.1 Linking with the Fortran Binding on OpenVMS

The link command for HP GKS Version 7.2 is as follows:

$ LINK program, SYS$LIBRARY:GKSFORBND/LIB

For more information on the compatibility of the DEC GKS Version 4.*x* C binding, see the *DEC GKS C Binding Reference Manual*.

### 9.1.6 C Binding

The HP GKS Version 7.2 and DEC GKS Version 4.*x* C bindings are source-code compatible, with the exception of the function return codes and a bug fix in the function ginqwsconntype. HP GKS provides run-time compatibility with DEC GKS Version 4.*x* C binding images. Digital recommends recompiling and relinking the application with HP GKS Version 7.2. The source-code compatibility exceptions are as follows:

- HP GKS Version 7.2 function return codes are integers based on the ISO standard error numbers and Digital (negative) error numbers.
- The HP GKS Version 7.2 ginqwsconntype function requires that the Gwsct structure element *type* be a pointer to an object of type Gwstype. The function ginqwsconntype returns the workstation type into this object. This is a change from earlier versions of GKS that returned the element *type* erroneously in the Gwsct structure element *type*, instead of to the location pointed to by this element.

#### 9.1.6.1 Linking with the C Binding on OpenVMS

The link command for HP GKS Version 7.2 differs from DEC GKS Version 4.*x*. The new link command includes the reference to the C binding object library, as follows:

$ LINK program, SYS$LIBRARY:GKSCBND/LIB, SYS$LIBRARY:VAXCRTL/LIB

For more information on the compatibility of the DEC GKS Version 4.*x* C binding, see the *DEC GKS C Binding Reference Manual*.

### 9.1.7 GKS$ Binding

The DEC GKS Version 4.2 GKS$ binding is fully upwards compatible with HP GKS Version 7.2.

The GKS$ interface supports C, Fortran, Ada, and Pascal programs.

#### 9.1.7.1 Linking with the GKS$ Binding on OpenVMS

The link command for HP GKS Version 7.2 differs from DEC GKS Version 4.*x*. The new link command includes the reference to the GKS$ binding object library, as follows:

$ LINK program

For more information on compatibility with the DEC GKS Version 4.*x* GKS$ binding, see the *DEC GKS GKS$ Binding Reference Manual*.

## 9.2 DEC GKS–3D Version 1.*x* to HP GKS Version 7.2 Compatibility

### 9.2.1 General Compatibility

The HP GKS Version 7.2 development environment differs from the one provided with DEC GKS–3D Version 1.2 as follows:

- The HP GKS Version 7.2 link commands are different due to a different file naming and installation schema. See the binding manuals for information on compiling and linking with HP GKS Version 7.2.
- The HP GKS Version 7.2 environment options are prefixed with GKS$ (OpenVMS) and GKS (DIGITAL UNIX or ULTRIX) rather than GKS3D$ and GKS3D, respectively.

### 9.2.2 FORTRAN Binding

The HP GKS Version 7.2 FORTRAN binding is backwardly compatible with DEC GKS–3D Version 1.2, except for the name of the include file. For backwards compatibility, the DEC GKS–3D Version 1.2 include file is also provided.

To port a DEC GKS–3D FORTRAN binding program to HP GKS, modify your source code to include *gks.f*. Then recompile and link your program with HP GKS Version 7.2.

### 9.2.3 C Binding

The HP GKS Version 7.2 C binding is not source-code compatible with the DEC GKS–3D Version 1.2 C binding. However, HP GKS Version 7.2 includes a compatibility mode that provides source code compatibility.

Although run-time compatibility is provided, Digital recommends recompiling and relinking your application with HP GKS Version 7.2.

For more information on the compatibility of the DEC GKS–3D Version V1.*x* C binding, see the *DEC GKS C Binding Reference Manual*.

### 9.2.4 GKS3D$ Binding

The HP GKS Version 7.2 GKS3D$ binding is fully backwards compatible with DEC GKS–3D Version 1.2.

The GKS3D$ interface supports C, Fortran, Ada, and Pascal programs.

#### 9.2.4.1 Linking with the GKS3D$ Binding on OpenVMS

The link command is simply:

$ LINK program

## 9.3 Workstation Support

In addition, HP GKS Version 7.2 supports the DEC GKS Version 4.2 GKSM (two-dimensional) metafile input workstation.

### 9.3.1 Implementation-Specific Features

DEC GKS Version 4.2 implementation-specific behavior may be provided differently with HP GKS Version 7.2. Applications relying on the implementation-specific behavior may have to be updated to produce the intended output with HP GKS.

# 10 HP GKS Documentation

The HP GKS Version 7.2 documentation set contains the following manuals:

- *Installing HP GKS for OpenVMS Integrity servers Systems*
- *DEC GKS User's Guide*
- *DEC GKS C Binding Reference Manual*
- *DEC GKS FORTRAN Binding Reference Manual*,
- *DEC GKS GKS3D$ Binding Reference Manual*
- *DEC GKS GKS$ Binding Reference Manual*
- *Device Specifics Reference Manual for DEC GKS and DEC PHIGS*
- *Building a Device Handler System for DEC GKS and DEC PHIGS*

The following section describes changes in the documentation that occurred after the manuals were printed. The information is arranged by manual title.

## 10.1 General Problems

This section lists the changes that affect more than one manual.

### 10.1.1 Number of Escapes

The bindings manuals incorrectly state that there are 61 GKS escapes. There are 65 escapes.

### 10.1.2 Initial Cursor Position for the String Device

The string class input data record contains a component for the initial cursor position. Although most workstations ignore this value, the Motif workstation positions the cursor at the position indicated by this component.

### 10.1.3 INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER

The maximum number of normalization transformations is 256, numbered 0 to 255.

### 10.1.4 Locator PET –13

In the GKS$, C, and FORTRAN bindings, the section describing the locator PETs incorrectly describes PET 13. The PET number should be –13.

### 10.1.5 Changing the Title String

All the choice, string, and valuator PETs, and locator PET 6 allow you to change the title string.

The default title is the name of the device class. To change the string, increase the data record size by twice the size of a pointer and add the following components to the end of the data record:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| Nexttolast | Title string address | Address | U |
| Last | Title string length | Integer | U |

If you want to have a blank title for a particular device, use a title string of one blank (' ') and a title string length of 1. If you do not specify the title string and title string length, HP GKS uses the default title. If you set the title string address to NULL and the title string length to zero, HP GKS uses the default title.

Be sure the data record length accurately reflects whether you want to use the title components.

## 10.2 DEC GKS User's Guide

The example programs in the *DEC GKS User's Guide* are included on the HP GKS kit. The program names are as follows:

```
File name                  Corresponding manual example

user_manual_1_3.c          Example 2-1
user_manual_3_1.c          Example 6-1
user_manual_4_1.c          Example 3-1
user_manual_5_1.c          Example 4-1
user_manual_6_1.c          Example 5-1
user_manual_7_1.c          Example 5-2
```

## 10.3 DEC GKS GKS$ Binding Reference Manual

The gks3d$sizeof/gks$sizeof utility function, used to determine the size of an input data record, is available with this release of HP GKS. The language-dependent data structures that define the input data record structures for each of the languages supported by the GKS$ binding are also available with this release. See the GKS3D$DEFS.*/GKS$DEFS.* include files to see what data structures and size macros are available for a particular language.

The following sections list all the input data records for the GKS$ binding. The tables within these sections include the order of the component within the input data record, a description of the component, the data type, and whether the workstation uses (U) or ignores (I) the input data record component. When you pass input data records in the INITIALIZE functions, the information order *must* match the order listed in the tables.

**NOTE**

**Be sure to observe alignment constraints on 64-bit systems. Note that the size of an input data record on a 64-bit system may *not* be equal to the sum of the sizes of its components, because of added padding to ensure alignment.**

### 10.3.1 Choice Class

This section lists the input data record information required for choice-class prompt and echo types (PETs).

#### Choice-Class PET 2

The input data record size (INITIALIZE function argument *rec_size*) is 8 on 32-bit systems (e.g. OpenVMS, ULTRIX), 16 on 64-bit systems (e.g. DIGITAL UNIX). The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Number of choice alternatives | Integer | U |
| 2 | Address of the array of prompts turned off (GKS$K_CHOICE_PROMPT_OFF) or on (GKS$K_CHOICE_PROMPT_ON) | Address | U |

### Choice-Class PETs –1, 1, and 3

The input data record size (INITIALIZE function argument *rec_size*) is 12 on 32-bit systems (e.g. OpenVMS, ULTRIX), 24 on 64-bit systems (e.g. DIGITAL UNIX). The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Number of choice alternatives | Integer | U |
| 2 | Address of the array of choice string lengths | Address | U |
| 3 | Address of the array of choice string addresses | Address | U |

## 10.3.2 Locator Class

This section lists the input data record information required for locator-class PETs.

### Locator-Class PET –1

The input data record size (INITIALIZE function argument *rec_size*) is 8. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | X dimension of the box in world coordinate points | Real | U |
| 2 | Y dimension of the box in world coordinate points | Real | U |

### Locator-Class PETs –11, 1, 2, 3, and 6

The input data record size (INITIALIZE function argument *rec_size*) is 0. The input data record (INITIALIZE function argument *data_rec*) is a dummy record.

### Locator-Class PETs –12, –10, –9, –5, –4, and 4

These prompt and echo types require you to use one of two input data records. Which input data record you use depends on the value of the attribute control flag. If the attribute control flag is GKS$K_ACF_CURRENT, the prompt and echo displays the current set of output attributes. If the attribute control flag is GKS$K_ACF_SPECIFIED, the prompt and echo displays the output attributes specified in the input data record.

To specify current output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 4. The input data record (INITIALIZE function argument *data_rec*) contains the component listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Attribute control flag (GKS$K_ACF_CURRENT) | Integer | U |

To specify new output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 32. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Attribute control flag (GKS$K_ACF_SPECIFIED) | Integer | U |
| 2 | Line type ASF (GKS$K_ASF_BUNDLED or GKS$K_ ASF_INDIVIDUAL) | Integer | I |
| 3 | Line width scale factor ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL) | Integer | I |
| 4 | Polyline color index (GKS$K_ASF_BUNDLED or GKS$K_ ASF_INDIVIDUAL) | Integer | I |
| 5 | Polyline index | Integer | I |
| 6 | Line type index | Integer | U |
| 7 | Line width scale factor | Real | U |
| 8 | Polyline color index | Integer | I |

### Locator-Class PETs –2 and 5

Although four input data records are theoretically possible for these PETs, HP GKS supports only two. The input data records depend on the values of the polyline-fill-area control flag, and on the attribute control flag. If the polyline-fill-area control flag is GKS$K_ACF_ POLYLINE, HP GKS uses a polyline to draw the prompt and echo rectangle. If the polyline-fill-area control flag is GKS$K_ACF_FILL_AREA, HP GKS uses a fill area to draw the prompt and echo. HP GKS currently supports only the polyline rectangle.

If the attribute control flag is GKS$K_ACF_CURRENT, the prompt and echo displays the current set of output attributes. If the attribute control flag is GKS$K_ACF_SPECIFIED, the prompt and echo displays the output attributes specified in the input data record.

To specify current output attribute values and a polyline rectangle, the input data record size (INITIALIZE function argument *rec_size*) is 8. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Polyline-fill-area control flag (GKS$K_ACF_POLYLINE) | Integer | I |
| 2 | Attribute control flag (GKS$K_ACF_CURRENT) | Integer | U |

To specify new output attribute values and a polyline rectangle, the input data record size (INITIALIZE function argument *rec_size*) is 36. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Polyline-fill-area control flag (GKS$K_ACF_POLYLINE) | Integer | I |
| 2 | Attribute control flag (GKS$K_ACF_SPECIFIED) | Integer | U |
| 3 | Line type ASF (GKS$K_ASF_BUNDLED or GKS$K_ ASF_INDIVIDUAL) | Integer | I |
| 4 | Line width scale factor ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL) | Integer | I |
| 5 | Polyline color index ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL) | Integer | I |
| 6 | Polyline index | Integer | I |
| 7 | Line type index | Integer | U |
| 8 | Line width scale factor | Real | U |
| 9 | Polyline color index | Integer | I |

### Locator-Class PETs –8, –7, –6, and –3

These PETs require you to use one of two input data records. Which input record you use depends on the value of the attribute control flag. If the attribute control flag is GKS$K_ ACF_CURRENT, the prompt and echo displays the current set of output attributes. If the attribute control flag is GKS$K_ACF_SPECIFIED, the prompt and echo displays the output attributes specified in the input data record.

To specify the current output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 20. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Attribute control flag (GKS$K_ACF_CURRENT) | Integer | U |
| 2 | X component of the first world coordinate point | Real | U |
| 3 | Y component of the first world coordinate point | Real | U |
| 4 | X component of the second world coordinate point | Real | U |
| 5 | Y component of the second world coordinate point | Real | U |

To specify new output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 48. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Attribute control flag (GKS$K_ACF_SPECIFIED) | Integer | U |
| 2 | Line type ASF (GKS$K_ASF_BUNDLED or GKS$K_ ASF_INDIVIDUAL) | Integer | I |
| 3 | Line width scale factor ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL) | Integer | I |
| 4 | Polyline color ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL) | Integer | I |

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 5 | Polyline index | Integer | I |
| 6 | Line type index | Integer | U |
| 7 | Line width scale factor | Real | U |
| 8 | Polyline color index | Integer | I |
| 9 | X component of the first world coordinate point | Real | U |
| 10 | Y component of the first world coordinate point | Real | U |
| 11 | X component of the second world coordinate point | Real | U |
| 12 | Y component of the second world coordinate point | Real | U |

#### Locator-Class PET –13

The input data record size (INITIALIZE function argument *rec_size*) is 4. The input data record (INITIALIZE function argument *data_rec*) contains the component listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Segment identifier of the segment to be used for the cursor segment | Integer | U |

### 10.3.3 Pick Class

This section lists the input data record information required for pick-class PETs.

#### Pick-Class PETs 1, 2, and 3

The input data record size (INITIALIZE function argument *rec_size*) is 4. The input data record (INITIALIZE function argument *data_rec*) contains the component listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Size of the pick aperture (prompt) in device coordinates | Real | U |

### 10.3.4 String Class

This section lists the input data record information required for string-class PETs.

#### String-Class PET 1

The input data record size (INITIALIZE function argument *rec_size*) is 8. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Number of characters in the input buffer | Integer | U |

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 2 | Initial cursor position within the string, $1 \leq$ position $\leq$ string_length | Integer | I |

### 10.3.5 Stroke Class

This section lists the input data record information required for stroke-class PETs.

#### Stroke-Class PET 1

The input data record size (INITIALIZE function argument *rec_size*) is 20. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Number of stroke points in the input buffer | Integer | U |
| 2 | Editing position expressed as a stroke point | Integer | I |
| 3 | X component of the world coordinate change vector | Real | U |
| 4 | Y component of the world coordinate change vector | Real | U |
| 5 | Time interval, in seconds | Real | I |

#### Stroke-Class PET 3

his PET requires you to use one of two input data records. Which input data records you use depends on the value of the attribute control flag. If the attribute control flag is GKS$K_ACF_CURRENT, the prompt and echo displays the current set of output attributes. If the attribute control flag is GKS$K_ACF_SPECIFIED, the prompt and echo displays the output attributes specified in the input data record.

To specify the current output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 24. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Number of stroke points in the input buffer | Integer | U |
| 2 | Editing position expressed as a stroke point | Integer | I |
| 3 | X component of the world coordinate change vector | Real | U |
| 4 | Y component of the world coordinate change vector | Real | U |
| 5 | Time interval, in seconds | Real | I |
| 6 | Attribute control flag (GKS$K_ACF_CURRENT) | Integer | U |

To specify new output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 52. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Number of stroke points in the input buffer | Integer | U |
| 2 | Editing position expressed as a stroke point | Integer | I |
| 3 | X component of the world coordinate change vector | Real | U |
| 4 | Y component of the world coordinate change vector | Real | U |
| 5 | Time interval, in seconds | Real | I |
| 6 | Attribute control flag (GKS$K_ACF_CURRENT) | Integer | U |
| 7 | Polymarker type ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL) | Integer | I |
| 8 | Polymarker scale factor ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL) | Integer | I |
| 9 | Polymarker color ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL) | Integer | I |
| 10 | Polymarker index | Integer | I |
| 11 | Polymarker type index | Integer | U |
| 12 | Polymarker scale factor | Real | U |
| 13 | Polymarker color index | Integer | I |

### Stroke-Class PET 4

This PET requires you to use one of two input data records. Which input record you use depends on the value of the attribute control flag. If the attribute control flag is GKS$K_ACF_CURRENT, the prompt and echo displays the current set of output attributes. If the attribute control flag is GKS$K_ACF_SPECIFIED, the prompt and echo displays the output attributes specified in the input data record.

To specify the current output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 24. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Number of stroke points in the input buffer | Integer | U |
| 2 | Editing position expressed as a stroke point | Integer | I |
| 3 | X component of the world coordinate change vector | Real | U |
| 4 | Y component of the world coordinate change vector | Real | U |
| 5 | Time interval, in seconds | Real | I |
| 6 | Attribute control flag (GKS$K_ACF_CURRENT) | Integer | U |

To specify new output attribute values, the input data record size (INITIALIZE function argument *rec_size*) is 52. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Number of stroke points in the input buffer | Integer | U |
| 2 | Editing position expressed as a stroke point | Integer | I |

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 3 | X component of the world coordinate change vector | Real | U |
| 4 | Y component of the world coordinate change vector | Real | U |
| 5 | Time interval, in seconds | Real | I |
| 6 | Attribute control flag (GKS$K_ACF_CURRENT) | Integer | U |
| 7 | Line type ASF (GKS$K_ASF_BUNDLED or GKS$K_ ASF_INDIVIDUAL) | Integer | I |
| 8 | Line width scale factor ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL) | Integer | I |
| 9 | Polyline color ASF (GKS$K_ASF_BUNDLED or GKS$K_ASF_INDIVIDUAL) | Integer | I |
| 10 | Polyline index | Integer | I |
| 11 | Line type index | Integer | U |
| 12 | Line width scale factor | Real | U |
| 13 | Polyline color index | Integer | I |

### 10.3.6 Valuator Class

This section lists the input data record information required for valuator-class PETs.

#### Valuator-Class PETs –3, –2, –1, 1, 2, and 3

The input data record size (INITIALIZE function argument *rec_size*) is 8. The input data record (INITIALIZE function argument *data_rec*) contains the components listed in the following table:

| Component | Description | Data Type | Used or Ignored |
|---|---|---|---|
| 1 | Low value of the numeric range | Real | U |
| 2 | High value of the numeric range | Real | U |

## 10.4 Device Specifics Reference Manual for DEC GKS and DEC PHIGS

This section lists the changes in the *Device Specifics Reference Manual for DEC GKS and DEC PHIGS* manual that occurred after the manual was printed in 1994.

### 10.4.1 GKS_PREDEF Program Example

The GKS_PREDEF program example on page 1-7 is incorrect. The output should be as follows:

```
    Color Information for Digital LCP01 or LCG01 printer (15) devices
```

```
Default color model: RGB (1)
Number of color models: 4
Color models:
       1 - RGB
       2 - CIE
       3 - HSV
       4 - HLS
Number of colors: 8
Color availability: Color
Number of predefined colors: 8
Predefined color table:
================================================
Index              Red       Green        Blue
================================================
  0              1.000       1.000       1.000
  1              0.000       1.000       0.000
  2              1.000       0.000       0.000
  3              0.000       0.000       1.000
  4              0.000       1.000       1.000
  5              1.000       0.000       1.000
  6              1.000       1.000       0.000
  7              0.000       0.000       0.000
================================================
```

## 10.5 DEC GKS GKS3D$ Binding Reference Manual

The input data record information described in Section 10.3 is also true for the GKS3D$ binding, with the following exceptions:

- The GKS3D$ include files are GKS3D$DEFS.*.
- All the constants begin with GKS3D$ instead of GKS$ (for example, GKS3D$K_ACF_ CURRENT).
- The INITIALIZE function parameter *rec_size* is called *data_len* in the GKS3D$ binding.

### 10.5.1 INQUIRE STROKE DEVICE STATE

The syntax lists *stroke_x* and *stroke_y* as separate arguments. In fact, these values are included in a single argument, *wc_stroke2*.

### 10.5.2 INQUIRE STROKE DEVICE STATE 3

There are a few errors in the description of the INQUIRE STROKE DEVICE STATE 3 function. In the syntax section, the argument should be *ret_pts*, not *ret_points*.

The syntax also lists *stroke_x*, *stroke_y*, and *stroke_z* as separate arguments. In fact, these values are included in a single argument, *wc_stroke3*. Further, in the description of this argument, it should read "The points are ordered . . . , for each of the *ret_pts*.

# 11 Tuning OpenVMS DECwindows Systems for HP GKS

The following sections describe system tuning for OpenVMS DECwindows systems for HP GKS Version 7.2.

## 11.1 Picking (AST Limits)

If picking fails, enlarge the ASTlm parameter in the user account. If this parameter is not large enough, the system can hang. A value of 300 should be sufficient for this parameter. Your system manager can change the value by using the AUTHORIZE utility to modify SYS$SYSTEM:SYSUAF.DAT. To use this parameter, you must log out and log back in.

## 11.2 System Process Parameters for the GKS (client) Program

System process parameters need to be large enough to hold GKS, the application, and any GKS segments that the application generates. These parameters should also be set properly (usually large, but setting them arbitrarily large can lead to system thrashing) to minimize paging. This is described thoroughly in the *Guide to OpenVMS Performance Management*. In particular, the process parameters WSQUO, WSDEF, WSEXTENT, and PGFLQUO affect the working set size and maximum size to which the process can grow. These parameters are set on a per-user basis using the AUTHORIZE utility. For more information on the parameters and how to set them, check the *VMS Authorize Utility Manual*.

## 11.3 Process Parameters for the DECwindows Server

The process parameters for the DECwindows server are not set through the AUTHORIZE utility. They are set through system logicals that are checked when the server is started. Four of these parameters are as follows:

- DECW$SERVER_PAGE_FILE —Sets the pagefile quota
- DECW$SERVER_WSDEF—Sets the working set default
- DECW$SERVER_WSEXTENT—Sets the working set extent
- DECW$SERVER_WSQUOTA—Sets the working set quota

These parameters work the same as the system parameters of similar names, and have analogous effects. These parameters need to be defined in your system startup file, SYS$MANAGER:SYSTARTUP_VMS.COM, as system logicals for the DECwindows server to use them. You may need to increase DECW$SERVER_PAGE_FILE if you find you are failing with insufficient virtual memory. Symptoms of insufficient virtual memory for the DECwindows server process include error messages in the server error file DECW$SERVER_0_ERROR.LOG, and the server may abort.

Digital recommends that the following DECwindows server logicals be set to the following minimum values:

- DECW$SERVER_WSDEF 4096
- DECW$SERVER_WSEXTENT 16384
- DECW$SERVER_WSQUOTA 8192

## 11.4 System Parameters

The WSMAX sysgen parameter limits the size that any working set can reach. VIRTUALPAGECNT affects how many virtual pages any process can address. No matter how large you set the working set and page file quotas, their total can never exceed VIRTUALPAGECNT. Finally, you need to be sure that enough page file exists to satisfy the page file quotas of both the GKS client process and the server, as well as other processes that are running.

## 12 Copyright



Printed in the US