

Information processing systems — Computer graphics — Graphical Kernel System (GKS) language bindings —

Part 2: Pascal

The European Standard EN 28651-2:1992 has the status of a
British Standard

UDC 681.3.04:800.92

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW



Document Sharing Hub

Committees responsible for this British Standard

The preparation of this British Standard was entrusted by the Information Systems Technology Standards Committee (IST/-) to Technical Committee IST/31, upon which the following bodies were represented:

- British Computer Society
- British Office Technology Manufacturers' Alliance
- Business Equipment and Information Technology Association
- Computer Graphics Suppliers Association
- Middlesex Polytechnic Faculty of Art and Design
- National Computing Centre Ltd.
- Science and Engineering Research Council (Rutherford Appleton Laboratory)
- University of Leeds
- University of Leicester
- University of London Computer Centre
- University of Manchester

This British Standard, having been prepared under the direction of the Information Systems Technology Standards Committee, was published under the authority of the Board of BSI and comes into effect on 28 February 1989

© BSI 05-2000

The following BSI references relate to the work on this standard:
Committee reference IST/31
Draft for comment DD 114-2

ISBN 0 580 17067 5

Amendments issued since publication

Amd. No.	Date of issue	Comments
7538	December 1992	Indicated by a sideline in the margin

Contents

	Page
Committees responsible	Inside front cover
National foreword	ii
Foreword	2
0 Introduction	3
1 Scope and field of application	3
2 References	3
3 The Pascal language binding of GKS	3
4 Error handling	25
5 Pascal GKS data structures	26
6 GKS functions	36
Annex A Data types in compilation order	91
Annex B Metafile Item Types	95
Annex C Example programs	96
Annex D Function lists	120
Table 1 — Abbreviations ordered alphabetically	4
Table 2 — GKS function names and Pascal names ordered by Pascal name	6
Table 3 — GKS function names and Pascal names ordered by GKS function name	11
Table 4 — GKS function names and Pascal names ordered by level	17
Publications referred to	Inside back cover

National foreword

This Part of BS 7040 has been prepared under the direction of the Information Systems Technology Standards Committee. It is identical with ISO 8651-2:1988 *“Information processing systems — Computer graphics — Graphical Kernel System (GKS) language bindings — Part 2: Pascal”* published by the International Organization for Standardization (ISO). The UK actively participated in the development of ISO 8651-2 through BSI Technical Committee IST/31.

In 1992 the European Committee for Standardization (CEN) accepted ISO 8651-2:1988 as European Standard EN 28651-2:1992. As a consequence of implementing the European Standard this British Standard is renumbered as BS EN 28651-2 and any reference to BS 7040-2 should be read as a reference to BS EN 28651-2.”

This Part of BS 7040 supersedes DD 114-2:1986 which is withdrawn.

Terminology and conventions. The text of the International Standard has been approved as suitable for publication as a British Standard without deviation. Some terminology and certain conventions are not identical with those used in British Standards; attention is drawn especially to the following.

Wherever the words “International Standard” appear, referring to this standard, they should be read as “Part of BS 7040”.

Cross-references

International Standard	Corresponding British Standard
ISO 7185:1983 ^a	BS 6192:1982 <i>Specification for computer programming language Pascal</i> (Identical)
ISO 7942:1985 ^a	BS 6390:1985 <i>Specification for a set of functions for computer graphics programming, the Graphical Kernel System (GKS)</i> (Identical)

^a Undated in the text.

The Technical Committee has reviewed the provisions of ISO 2382-13, to which reference is made in the text, and has decided that they are acceptable for use in conjunction with this British Standard.

Up-to-date information on existing and projected British Standards publications for Computer Graphics is included in the “BASIC LIST” published as part of the BSI Information Technology Services (BITS) initiative.

For details contact:

Marketing Department
BSI
Linford Wood
Milton Keynes
MK 14 6LE

A British Standard does not purport to include all the necessary provisions of a contract. Users of British Standards are responsible for their correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, pages i and ii, the EN title page, pages 2 to 124, an inside back cover and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

UDC 681.3.04:800.92

Descriptors: Data processing, graphic data processing, computer interfaces, graphical kernel system, programming languages, Pascal

English version

Information processing systems — Computer graphics — Graphical Kernel System (GKS) language bindings Part 2: Pascal

(ISO 8651-2:1988)

Systèmes de traitement de l'information —
Infographie — Interfaces langage avec GKS
Partie 2. Pascal
(ISO 8651-2:1988)

Graphische Systeme der
Informationsverarbeitung — Sprachbindungen
für das Graphische Kernsystem (GKS)
Teil 2. Pascal
(ISO 8651-2:1988)

This European Standard was approved by CEN on 1992-08-13. CEN members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CEN member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN member into its own language and notified to the Central Secretariat has the same status as the official versions.

CEN members are the national standards bodies of Austria, Belgium, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

CEN

European Committee for Standardization
Comité Européen de Normalisation
Europäisches Komitee für Normung

Central Secretariat: rue de Stassart 36, B-1050 Brussels

© 1992 Copyright reserved to CEN members

Foreword

The Technical Board has decided to submit the International Standard 8651-2:1988 *Information processing systems — Computer graphics — Graphical Kernel System (GKS) language bindings — Part 2: Pascal* for Formal Vote. The standard was accepted.

At present the standard exists in the English and French versions only.

This European Standard shall be given the status of a national standard, either by publication of an identical text or by endorsement, at the latest by February 1993 and conflicting national standards shall be withdrawn at the latest by February 1993.

According to the CEN/CENELEC Common Rules the following countries are bound to implement this standard:

Austria, Belgium, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland, United Kingdom.

0 Introduction

The Graphical Kernel System (GKS), the functional description of which is given in ISO 7942, is specified in a language-independent manner and needs to be embedded in language-dependent layers (language bindings) for use with particular programming languages.

The purpose of this part of ISO 8651 is to define a standard binding for the Pascal computer programming language.

1 Scope and field of application

ISO 7942 specifies a language-independent nucleus of a graphics system. For integration into a programming language, GKS is embedded in a language-dependent layer obeying the particular conventions of that language. This part of ISO 8651 specifies such a language-dependent layer for the Pascal language.

2 References

ISO 7942, *Information processing systems — Computer graphics — Graphical Kernel System (GKS) functional description*.

ISO 7185, *Programming languages — Pascal*.

ISO 2382-13, *Data processing — Vocabulary — Part 13: Computer Graphics*.

3 The Pascal language binding of GKS

3.1 Specification

The GKS language binding interface for ISO Pascal (ISO 7185) shall be as described in clauses 3, 4, 5, and 6.

3.2 Mapping of GKS function names to Pascal procedure names

The function names of GKS are all mapped to Pascal procedures which begin with the letter “G”. Words and phrases used in the GKS function names are often abbreviated in the Pascal representation. There is a set of such abbreviations given in Table 1 and the resulting Pascal procedure names are listed in Table 2, Table 3, and Table 4. For example, the abbreviation for the GKS function DELETE SEGMENT FROM WORKSTATION is GDelSegWs. “Del”, “Seg”, “Ws” are the abbreviations for DELETE, SEGMENT and WORKSTATION. Conjunctives such as “from”, “and”, “of” and “to” are mapped to null strings, as are a number of other words used in the GKS abstract names. For example, INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES is mapped to GInqMaxWsSt. Here LENGTH and TABLES are represented by null strings.

3.3 The many-one nature of the Pascal interface

There is not a strict one-to-one correspondence between GKS abstract functions and Pascal procedures. A method employing variant records is used to represent several logically related GKS abstract functions by one Pascal procedure. The first parameter of such a procedure is always an enumerated type which is the tag field of a variant record which is itself a parameter of the Pascal procedure. This technique is used across two classes of abstract functions — those relating to the setting and inquiring of output primitive representations, and those relating to the setting and inquiring of the input classes. Where this method is used, the rules for deriving the Pascal name of the GKS abstract function are

a) Output Primitive Representations

- 1) The GKS words Polyline, Polymarker, Text, and Fill Area are replaced by “Prim” (which is the abbreviation for PRIMITIVE).
- 2) The first parameter of the function is an enumerated type (GEPrim) which has one of the values GVPolyline, GVPolymarker, GVText, GVFillArea.

b) Input Classes

- 1) The GKS words Locator, Stroke, Valuator, Choice, Pick, and String are replaced by “Input”.
- 2) The first parameter of the function is an enumerated type (GEInputClass) which has one of the values GVLocator, GVStroke, GVValuator, GVChoice, GVPick, GVString.

3.4 The one-one nature of the Pascal interface

The Pascal interface to GKS described in 3.3 reflects the GKS major dimensions of Output Primitive Representations and Input Classes. However, the possibility exists that on small systems such an interface might cause difficulties, especially with respect to implementation of the Input Classes. Therefore, the Pascal Binding also adopts a mandatory representation which uses a one-one mapping for the setting of primitive representations and input classes.

3.5 The one-many nature of the Pascal interface

The GKS abstract functions INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES and INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES are represented by the method described in 3.4. In addition, to allow for the possible frequent use of only some of the information, these functions have also been split into a number of Pascal procedures. Both representations are mandatory.

3.6 Implementation of the interfaces

Since any of the methods referred to in 3.3, 3.4 and 3.5 can be implemented easily in terms of another, the additional interfaces do not present a great burden for the implementor, nor does it cause an additional burden for application programs. Implementors are encouraged to use one method in the core of their implementation. In any event all sets of procedures shall be provided.

Table 1 — Abbreviations ordered alphabetically

GKS word	Abbreviation
ACCUMULATE	Accum
ALIGNMENT	Align
ALL	NULL
AND	NULL
ASPECT SOURCE FLAGS	ASF
ASSOCIATE	Assoc
ATTRIBUTE	Attr
ATTRIBUTES	Attr
AVAILABLE	NULL
CHARACTER	Char
CLASSIFICATION	Class
CLIPPING	Clip
COLOUR	Colr
CONNECTION	Conn
CURRENT	Cur
DEFAULT	Def
DEFERRAL	Defer
DELETE	Del
DETECTABILITY	Det
DIMENSIONS	Dim
DYNAMIC	Dyn
EVALUATE	Eval
EXPANSION	Expan
FACILITIES	Facil
FACTOR	NULL
FILL AREA	Fill
FROM	NULL
GENERALIZED DRAWING PRIMITIVE	GDP
GRAPHICAL KERNEL SYSTEM	GKS
GKSM	NULL
HIGHLIGHTING	Highlight
IDENTIFIER	Id
IN	NULL
INDEX	Ind
INDICATOR	NULL

Table 1 — Abbreviations ordered alphabetically

GKS word	Abbreviation
INDICES	Ind
INDIVIDUAL	Indiv
INITIALISE	Init
INPUT	NULL
INQUIRE	Inq
INTERIOR	Int
LENGTH	NULL
LIST	NULL
LOGICAL	NULL
MATRIX	NULL
MAXIMUM	Max
MODIFICATION	Mod
NAME	NULL
NORMALIZATION	Norm
NUMBER	Num
NUMBERS	Num
OF	NULL
ON	NULL
OPERATING	Op
POLYLINE	Line
POLYMARKER	Marker
PRECISION	Prec
PREDEFINED	Pred
PRIMITIVE	Prim
QUEUE	NULL
REFERENCE	Ref
REPRESENTATION	Rep
REQUEST	Req
SEGMENT	Seg
SEGMENTS	Seg
SET	NULL
SIMULTANEOUS	NULL
SPACE	NULL
STATE	St
SUPPORTED	NULL
TABLES	NULL
TO	NULL
TRANSFORMATION	Tran
UPDATE	Upd
USE	NULL
VALUE	NULL
VALUES	NULL
VISIBILITY	Vis
WITH	NULL
WORKSTATION	Ws
NOTE NULL represents the null string	

Table 2 — GKS function names and Pascal names ordered by Pascal name

GKS Function Name	Level	Pascal Name
ACCUMULATE TRANSFORMATION MATRIX	L1a	GAccumTran
ACTIVATE WORKSTATION	L0a	GActivateWs
ASSOCIATE SEGMENT WITH WORKSTATION	L2a	GAssocSegWs
AWAIT EVENT	L0c	GAwaitEvent
CELL ARRAY	L0a	GCellArray
CLEAR WORKSTATION	L0a	GClearWs
CLOSE GKS	L0a	GClosegks
CLOSE SEGMENT	L1a	GCloseSeg
CLOSE WORKSTATION	L0a	GCloseWs
COPY SEGMENT TO WORKSTATION	L2a	GCopySegWs
CREATE SEGMENT	L1a	GCreateSeg
DEACTIVATE WORKSTATION	L0a	GDeactivateWs
DELETE SEGMENT	L1a	GDelSeg
DELETE SEGMENT FROM WORKSTATION	L1a	GDelSegWs
EMERGENCY CLOSE GKS	L0a	GEmergencyClosegks
ERROR HANDLING	L0a	GErrorHandling
ERROR LOGGING	L0a	GErrorLogging
ESCAPE	L0a	GEscape
ESCAPE	L0a	GEscapeGeneralized
EVALUATE TRANSFORMATION MATRIX	L1a	GEvalTran
FILL AREA	L0a	GFill
FLUSH DEVICE EVENTS	L0c	GFlushDeviceEvents
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDP
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDPGeneralized
GET CHOICE	L0c	GGetChoice
GET CHOICE	L0c	GGetInput(choice
GET LOCATOR	L0c	GGetInput(locator
GET PICK	L1c	GGetInput(pick
GET STRING	L0c	GGetInput(string
GET STROKE	L0c	GGetInput(stroke
GET VALUATOR	L0c	GGetInput(valuator
GET ITEM TYPE FROM GKSM	L0a	GGetItemType
GET LOCATOR	L0c	GGetLocator
GET PICK	L1c	GGetPick
GET STRING	L0c	GGetString
GET STROKE	L0c	GGetStroke
GET VALUATOR	L0c	GGetValuator
INITIALISE CHOICE	L0b	GInitChoice
INITIALISE CHOICE	L0b	GInitInput(choice
INITIALISE LOCATOR	L0b	GInitInput(locator
INITIALISE PICK	L1b	GInitInput(pick
INITIALISE STRING	L0b	GInitInput(string
INITIALISE STROKE	L0b	GInitInput(stroke
INITIALISE VALUATOR	L0b	GInitInput(valuator
INITIALISE LOCATOR	L0b	GInitLocator
INITIALISE PICK	L1b	GInitPick
INITIALISE STRING	L0b	GInitString
INITIALISE STROKE	L0b	GInitStroke
INITIALISE VALUATOR	L0b	GInitValuator
INQUIRE ASPECT SOURCE FLAGS	L0a	GInqasf
INQUIRE SET OF ACTIVE WORKSTATIONS	L1a	GInqActiveWs
INQUIRE SET OF ASSOCIATED WORKSTATIONS	L1a	GInqAssocWs

Table 2 — GKS function names and Pascal names ordered by Pascal name

GKS Function Name	Level	Pascal Name
INQUIRE CHARACTER BASE VECTOR	L0a	GInqCharBaseVector
INQUIRE CHARACTER EXPANSION FACTOR	L0a	GInqCharExpan
INQUIRE CHARACTER HEIGHT	L0a	GInqCharHeight
INQUIRE CHARACTER SPACING	L0a	GInqCharSpacing
INQUIRE CHARACTER UP VECTOR	L0a	GInqCharUpVector
INQUIRE CHARACTER WIDTH	L0a	GInqCharWidth
INQUIRE CHOICE DEVICE STATE	L0b	GInqChoiceDeviceSt
INQUIRE CLIPPING	L0a	GInqClip
INQUIRE COLOUR FACILITIES	L0a	GInqColrFacil
INQUIRE COLOUR REPRESENTATION	L0a	GInqColrRep
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	L0a	GInqCurIndivAttr
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	L0a	GInqCurNormTranNum
INQUIRE CURRENT PICK IDENTIFIER	L1b	GInqCurPickId
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	L0a	GInqCurPrimAttr
INQUIRE DEFAULT CHOICE DEVICE DATA	L0b	GInqDefChoiceDeviceData
INQUIRE DEFAULT DEFERRAL STATE VALUES	L1a	GInqDefDeferSt
INQUIRE DEFAULT CHOICE DEVICE DATA	L0b	GInqDefInpuDeviceData(choice
INQUIRE DEFAULT LOCATOR DEVICE DATA	L1b	GInqDefInpuDeviceData(locator
INQUIRE DEFAULT PICK DEVICE DATA	L0b	GInqDefInpuDeviceData(pick
INQUIRE DEFAULT STRING DEVICE DATA	L0b	GInqDefInpuDeviceData(string
INQUIRE DEFAULT STROKE DEVICE DATA	L0b	GInqDefInpuDeviceData(stroke
INQUIRE DEFAULT VALUATOR DEVICE DATA	L0b	GInqDefInpuDeviceData(valuator
INQUIRE DEFAULT LOCATOR DEVICE DATA	L0b	GInqDefLocatorDeviceData
INQUIRE DEFAULT PICK DEVICE DATA	L1b	GInqDefPickDeviceData
INQUIRE DEFAULT STRING DEVICE DATA	L0b	GInqDefStringDeviceData
INQUIRE DEFAULT STROKE DEVICE DATA	L0b	GInqDefStrokeDeviceData
INQUIRE DEFAULT VALUATOR DEVICE DATA	L0b	GInqDefValuatorDeviceData
INQUIRE DISPLAY SPACE SIZE	L0a	GInqDisplaySize
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES	L1a	GInqDynModSegAttr
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	L1a	GInqDynModWsAttr
INQUIRE FILL AREA COLOUR INDEX	L0a	GInqFillColrInd
INQUIRE FILL AREA FACILITIES	L0a	GInqFillFacil
INQUIRE FILL AREA INDEX	L0a	GInqFillInd
INQUIRE FILL AREA INTERIOR STYLE	L0a	GInqFillIntStyle
INQUIRE FILL AREA REPRESENTATION	L1a	GInqFillRep
INQUIRE FILL AREA STYLE INDEX	L0a	GInqFillStyleInd
INQUIRE GENERALIZED DRAWING PRIMITIVE	L0a	GInqgdp
INQUIRE CHOICE DEVICE STATE	L0b	GInqInputDeviceSt(choice
INQUIRE LOCATOR DEVICE STATE	L0b	GInqInputDeviceSt(locator
INQUIRE PICK DEVICE STATE	L1b	GInqInputDeviceSt(pick
INQUIRE STRING DEVICE STATE	L0b	GInqInputDeviceSt(string
INQUIRE STROKE DEVICE STATE	L0b	GInqInputDeviceSt(stroke
INQUIRE VALUATOR DEVICE STATE	L0b	GInqInputDeviceSt(valuator
INQUIRE INPUT QUEUE OVERFLOW	L0c	GInqInputOverflow
INQUIRE LEVEL OF GKS	L0a	GInqLevelgks
INQUIRE POLYLINE COLOUR INDEX	L0a	GInqLineColrInd
INQUIRE POLYLINE INDEX	L0a	GInqLineInd
INQUIRE LINETYPE	L0a	GInqLineType

Table 2 — GKS function names and Pascal names ordered by Pascal name

GKS Function Name	Level	Pascal Name
INQUIRE LINEWIDTH SCALE FACTOR	L0a	GInqLineWidthScale
INQUIRE LIST OF COLOUR INDICES	L0a	GInqListColrInd
INQUIRE LIST OF FILL AREA INDICES	L1a	GInqListFillInd
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	L0a	GInqListGDP
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS	L0a	GInqListNormTranNum
INQUIRE LIST OF PATTERN INDICES	L1a	GInqListPatternInd
INQUIRE LIST OF POLYLINE INDICES	L1a	GInqListPolylineInd
INQUIRE LIST OF POLYMARKER INDICES	L1a	GInqListPolymarkerInd
INQUIRE LIST OF FILL AREA INDICES	L1a	GInqListPrimInd(FillArea
INQUIRE LIST OF POLYLINE INDICES	L1a	GInqListPrimInd(Polyline
INQUIRE LIST OF POLYMARKER INDICES	L1a	GInqListPrimInd(Polymarker
INQUIRE LIST OF TEXT INDICES	L1a	GInqListPrimInd(Text
INQUIRE LIST OF TEXT INDICES	L1a	GInqListTextInd
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	L0a	GInqListWsTypes
INQUIRE LOCATOR DEVICE STATE	L0b	GInqLocatorDeviceSt
INQUIRE POLYMARKER COLOUR INDEX	L0a	GInqMarkerColrInd
INQUIRE POLYMARKER INDEX	L0a	GInqMarkerInd
INQUIRE POLYMARKER SIZE SCALE FACTOR	L0a	GInqMarkerSizeScale
INQUIRE POLYMARKER TYPE	L0a	GInqMarkerType
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	L0a	GInqMaxNormTranNum
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	L0a	GInqMaxWsSt
INQUIRE MORE SIMULTANEOUS EVENTS	L0c	GInqMoreEvents
INQUIRE NORMALIZATION TRANSFORMATION	L0a	GInqNormTran
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	L0b	GInqNumInputDevices
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED	L1a	GInqNumSegPriorities
INQUIRE OPERATING STATE VALUE	L0a	GInqOpSt
INQUIRE NAME OF OPEN SEGMENT	L1a	GInqOpenSeg
INQUIRE SET OF OPEN WORKSTATIONS	L0a	GInqOpenWs
INQUIRE PATTERN FACILITIES	L0a	GInqPatternFacil
INQUIRE PATTERN REFERENCE POINT	L0a	GInqPatternRefPoint
INQUIRE PATTERN REPRESENTATION	L1a	GInqPatternRep
INQUIRE PATTERN SIZE	L0a	GInqPatternSize
INQUIRE PICK DEVICE STATE	L1b	GInqPickDeviceSt
INQUIRE PIXEL	L0a	GInqPixel
INQUIRE PIXEL ARRAY	L0a	GInqPixelArray
INQUIRE PIXEL ARRAY DIMENSIONS	L0a	GInqPixelArrayDim
INQUIRE POLYLINE FACILITIES	L0a	GInqPolylineFacil
INQUIRE POLYLINE REPRESENTATION	L1a	GInqPolylineRep
INQUIRE POLYMARKER FACILITIES	L0a	GInqPolymarkerFacil
INQUIRE POLYMARKER REPRESENTATION	L1a	GInqPolymarkerRep
INQUIRE PREDEFINED COLOUR REPRESENTATION	L0a	GInqPredColrRep
INQUIRE PREDEFINED FILL AREA REPRESENTATION	L0a	GInqPredFillRep
INQUIRE PREDEFINED PATTERN REPRESENTATION	L0a	GInqPredPatternRep
INQUIRE PREDEFINED POLYLINE REPRESENTATION	L0a	GInqPredPolylineRep
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	L0a	GInqPredPolymarkerRep
INQUIRE PREDEFINED FILL AREA REPRESENTATION	L0a	GInqPredPrimRep(FillArea
INQUIRE PREDEFINED POLYLINE REPRESENTATION	L0a	GInqPredPrimRep(Polyline

Table 2 — GKS function names and Pascal names ordered by Pascal name

GKS Function Name	Level	Pascal Name
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	L0a	GInqPredPrimRep(Polymarker
INQUIRE PREDEFINED TEXT REPRESENTATION	L0a	GInqPredPrimRep(Text
INQUIRE PREDEFINED TEXT REPRESENTATION	L0a	GInqPredTextRep
INQUIRE FILL AREA FACILITIES	L0a	GInqPrimFacil(FillArea
INQUIRE POLYLINE FACILITIES	L0a	GInqPrimFacil(Polyline
INQUIRE POLYMARKER FACILITIES	L0a	GInqPrimFacil(Polymarker
INQUIRE TEXT FACILITIES	L0a	GInqPrimFacil(Text
INQUIRE FILL AREA REPRESENTATION	L1a	GInqPrimRep(FillArea
INQUIRE POLYLINE REPRESENTATION	L1a	GInqPrimRep(Polyline
INQUIRE POLYMARKER REPRESENTATION	L1a	GInqPrimRep(Polymarker
INQUIRE TEXT REPRESENTATION	L1a	GInqPrimRep(Text
INQUIRE SEGMENT ATTRIBUTES	L1a	GInqSegAttr
INQUIRE SET OF SEGMENT NAMES IN USE	L1a	GInqSegNames
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION	L1a	GInqSegNamesWs
INQUIRE STRING DEVICE STATE	L0b	GInqStringDeviceSt
INQUIRE STROKE DEVICE STATE	L0b	GInqStrokeDeviceSt
INQUIRE TEXT ALIGNMENT	L0a	GInqTextAlign
INQUIRE TEXT COLOUR INDEX	L0a	GInqTextColrInd
INQUIRE TEXT EXTENT	L0a	GInqTextExtent
INQUIRE TEXT FACILITIES	L0a	GInqTextFacil
INQUIRE TEXT FONT AND PRECISION	L0a	GInqTextFontPrec
INQUIRE TEXT INDEX	L0a	GInqTextInd
INQUIRE TEXT PATH	L0a	GInqTextPath
INQUIRE TEXT REPRESENTATION	L1a	GInqTextRep
INQUIRE VALUATOR DEVICE STATE	L0b	GInqValuatorDeviceSt
INQUIRE WORKSTATION CATEGORY	L0a	GInqWsCategory
INQUIRE WORKSTATION CLASSIFICATION	L0a	GInqWsClass
INQUIRE WORKSTATION CONNECTION AND TYPE	L0a	GInqWsConnType
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	L0a	GInqWsDeferUpdSt
INQUIRE WORKSTATION MAXIMUM NUMBERS	L1a	GInqWsMaxNum
INQUIRE WORKSTATION STATE	L0a	GInqWsSt
INQUIRE WORKSTATION TRANSFORMATION	L0a	GInqWsTran
INSERT SEGMENT	L2a	GInsertSeg
INTERPRET ITEM	L0a	GInterpretItem
MESSAGE	L1a	GMessage
OPEN GKS	L0a	GOpenGKS
OPEN WORKSTATION	L0a	GOpenWs
POLYLINE	L0a	GPolyline
POLYMARKER	L0a	GPolymarker
READ ITEM FROM GKSM	L0a	GReadItem
REDRAW ALL SEGMENTS ON WORKSTATION	L1a	GRedrawSegWs
RENAME SEGMENT	L1a	GRenameSeg
REQUEST CHOICE	L0b	GReqChoice
REQUEST CHOICE	L0b	GReqInput(Choice
REQUEST LOCATOR	L0b	GReqInput(Locator
REQUEST PICK	L1b	GReqInput(Pick
REQUEST STRING	L0b	GReqInput(String
REQUEST STROKE	L0b	GReqInput(Stroke
REQUEST VALUATOR	L0b	GReqInput(Valuator
REQUEST LOCATOR	L0b	GReqLocator
REQUEST PICK	L1b	GReqPick
REQUEST STRING	L0b	GReqString

Table 2 — GKS function names and Pascal names ordered by Pascal name

GKS Function Name	Level	Pascal Name
REQUEST STROKE	L0b	GReqStroke
REQUEST VALUATOR	L0b	GReqValuator
SAMPLE CHOICE	L0c	GSampleChoice
SAMPLE CHOICE	L0c	GSampleInput(Choice
SAMPLE LOCATOR	L0c	GSampleInput(Locator
SAMPLE PICK	L1c	GSampleInput(Pick
SAMPLE STRING	L0c	GSampleInput(String
SAMPLE STROKE	L0c	GSampleInput(Stroke
SAMPLE VALUATOR	L0c	GSampleInput(Valuator
SAMPLE LOCATOR	L0c	GSampleLocator
SAMPLE PICK	L1c	GSamplePick
SAMPLE STRING	L0c	GSampleString
SAMPLE STROKE	L0c	GSampleStroke
SAMPLE VALUATOR	L0c	GSampleValuator
SELECT NORMALIZATION TRANSFORMATION	L0a	GSelectNormTran
SET ASPECT SOURCE FLAGS	L0a	GSetasf
SET CHARACTER EXPANSION FACTOR	L0a	GSetCharExpan
SET CHARACTER HEIGHT	L0a	GSetCharHeight
SET CHARACTER SPACING	L0a	GSetCharSpacing
SET CHARACTER UP VECTOR	L0a	GSetCharUpVector
SET CHOICE MODE	L0b	GSetChoiceMode
SET CLIPPING INDICATOR	L0a	GSetClip
SET COLOUR REPRESENTATION	L0a	GSetColrRep
SET DEFERRAL STATE	L1a	GSetDeferSt
SET DETECTABILITY	L1b	GSetDet
SET FILL AREA COLOUR INDEX	L0a	GSetFillColrInd
SET FILL AREA INDEX	L0a	GSetFillInd
SET FILL AREA INTERIOR STYLE	L0a	GSetFillIntStyle
SET FILL AREA REPRESENTATION	L1a	GSetFillRep
SET FILL AREA STYLE INDEX	L0a	GSetFillStyleInd
SET HIGHLIGHTING	L1a	GSetHighlight
SET CHOICE MODE	L0b	GSetInputMode(Choice
SET LOCATOR MODE	L0b	GSetInputMode(Locator
SET PICK MODE	L1b	GSetInputMode(Pick
SET STRING MODE	L0b	GSetInputMode(String
SET STROKE MODE	L0b	GSetInputMode(Stroke
SET VALUATOR MODE	L0b	GSetInputMode(Valuator
SET POLYLINE COLOUR INDEX	L0a	GSetLineColrInd
SET LINETYPE	L0a	GSetLineType
SET LINEWIDTH SCALE FACTOR	L0a	GSetLineWidthScale
SET LOCATOR MODE	L0b	GSetLocatorMode
SET POLYMARKER COLOUR INDEX	L0a	GSetMarkerColrInd
SET MARKER SIZE SCALE FACTOR	L0a	GSetMarkerSizeScale
SET MARKER TYPE	L0a	GSetMarkerType
SET PATTERN REFERENCE POINT	L0a	GSetPatternRefPoint
SET PATTERN REPRESENTATION	L1a	GSetPatternRep
SET PATTERN SIZE	L0a	GSetPatternSize
SET PICK IDENTIFIER	L1b	GSetPickId
SET PICK MODE	L1b	GSetPickMode
SET POLYLINE INDEX	L0a	GSetPolylineInd
SET POLYLINE REPRESENTATION	L1a	GSetPolylineRep
SET POLYMARKER INDEX	L0a	GSetPolymarkerInd
SET POLYMARKER REPRESENTATION	L1a	GSetPolymarkerRep
SET FILL AREA INDEX	L0a	GSetPrimInd(FillArea

Table 2 — GKS function names and Pascal names ordered by Pascal name

GKS Function Name	Level	Pascal Name
SET POLYLINE INDEX	L0a	GSetPrimInd(Polyline
SET POLYMARKER INDEX	L0a	GSetPrimInd(Polymarker
SET TEXT INDEX	L0a	GSetPrimInd(Text
SET FILL AREA REPRESENTATION	L1a	GSetPrimRep(FillArea
SET POLYLINE REPRESENTATION	L1a	GSetPrimRep(Polyline
SET POLYMARKER REPRESENTATION	L1a	GSetPrimRep(Polymarker
SET TEXT REPRESENTATION	L1a	GSetPrimRep(Text
SET SEGMENT PRIORITY	L1a	GSetSegPriority
SET SEGMENT TRANSFORMATION	L1a	GSetSegTran
SET STRING MODE	L0b	GSetStringMode
SET STROKE MODE	L0b	GSetStrokeMode
SET TEXT ALIGNMENT	L0a	GSetTextAlign
SET TEXT COLOUR INDEX	L0a	GSetTextColrInd
SET TEXT FONT AND PRECISION	L0a	GSetTextFontPrec
SET TEXT INDEX	L0a	GSetTextInd
SET TEXT PATH	L0a	GSetTextPath
SET TEXT REPRESENTATION	L1a	GSetTextRep
SET VALUATOR MODE	L0b	GSetValuatorMode
SET VIEWPORT	L0a	GSetViewport
SET VIEWPORT INPUT PRIORITY	L0b	GSetViewportPriority
SET VISIBILITY	L1a	GSetVis
SET WINDOW	L0a	GSetWindow
SET WORKSTATION VIEWPORT	L0a	GSetWsViewport
SET WORKSTATION WINDOW	L0a	GSetWsWindow
TEXT	L0a	GText
UPDATE WORKSTATION	L0a	GUpdWs
WRITE ITEM TO GKSM	L0a	GWriteItem
WRITE ITEM TO GKSM	L0a	GWriteItemGeneralized

Table 3 — GKS function names and Pascal names ordered by GKS function name

GKS Function Name	Level	Pascal Name
ACCUMULATE TRANSFORMATION MATRIX	L1a	GAccumTran
ACTIVATE WORKSTATION	L0a	GActivateWs
ASSOCIATE SEGMENT WITH WORKSTATION	L2a	GAssocSegWs
AWAIT EVENT	L0c	GAwaitEvent
CELL ARRAY	L0a	GCellArray
CLEAR WORKSTATION	L0a	GClearWs
CLOSE GKS	L0a	GClosegks
CLOSE SEGMENT	L1a	GCloseSeg
CLOSE WORKSTATION	L0a	GCloseWs
COPY SEGMENT TO WORKSTATION	L2a	GCopySegWs
CREATE SEGMENT	L1a	GCreateSeg
DEACTIVATE WORKSTATION	L0a	GDeactivateWs
DELETE SEGMENT	L1a	GDelSeg
DELETE SEGMENT FROM WORKSTATION	L1a	GDelSegWs
EMERGENCY CLOSE GKS	L0a	GEmergencyClosegks
ERROR HANDLING	L0a	GErrorHandling
ERROR LOGGING	L0a	GErrorLogging
ESCAPE	L0a	GEscape
ESCAPE	L0a	GEscapeGeneralized
EVALUATE TRANSFORMATION MATRIX	L1a	GEvalTran
FILL AREA	L0a	GFill
FLUSH DEVICE EVENTS	L0c	GFlushDeviceEvents

Table 3 — GKS function names and Pascal names ordered by GKS function name

GKS Function Name	Level	Pascal Name
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDP
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDPGeneralized
GET CHOICE	L0c	GGetChoice
GET CHOICE	L0c	GGetInput(Choice
GET ITEM TYPE FROM GKSM	L0a	GGetItemType
GET LOCATOR	L0c	GGetInput(Locator
GET LOCATOR	L0c	GGetLocator
GET PICK	L1c	GGetInput(Pick
GET PICK	L1c	GGetPick
GET STRING	L0c	GGetInput(String
GET STRING	L0c	GGetString
GET STROKE	L0c	GGetInput(Stroke
GET STROKE	L0c	GGetStroke
GET VALUATOR	L0c	GGetInput(Valuator
GET VALUATOR	L0c	GGetValuator
INITIALISE CHOICE	L0b	GInitChoice
INITIALISE CHOICE	L0b	GInitInput(Choice
INITIALISE LOCATOR	L0b	GInitInput(Locator
INITIALISE LOCATOR	L0b	GInitLocator
INITIALISE PICK	L1b	GInitInput(Pick
INITIALISE PICK	L1b	GInitPick
INITIALISE STRING	L0b	GInitInput(String
INITIALISE STRING	L0b	GInitString
INITIALISE STROKE	L0b	GInitInput(Stroke
INITIALISE STROKE	L0b	GInitStroke
INITIALISE VALUATOR	L0b	GInitInput(Valuator
INITIALISE VALUATOR	L0b	GInitValuator
INQUIRE CHOICE DEVICE STATE	L0b	GInqChoiceDeviceSt
INQUIRE CHOICE DEVICE STATE	L0b	GInqInputDeviceSt(Choice
INQUIRE CLIPPING	L0a	GInqClip
INQUIRE COLOUR FACILITIES	L0a	GInqColrFacil
INQUIRE COLOUR REPRESENTATION	L0a	GInqColrRep
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	L0a	GInqCurIndivAttr
INQUIRE ASPECT SOURCE FLAGS	L0a	GInqasf
INQUIRE CHARACTER EXPANSION FACTOR	L0a	GInqCharExpan
INQUIRE CHARACTER SPACING	L0a	GInqCharSpacing
INQUIRE FILL AREA COLOUR INDEX	L0a	GInqFillColrInd
INQUIRE FILL AREA INTERIOR STYLE	L0a	GInqFillIntStyle
INQUIRE FILL AREA STYLE INDEX	L0a	GInqFillStyleInd
INQUIRE LINETYPE	L0a	GInqLineType
INQUIRE LINEWIDTH SCALE FACTOR	L0a	GInqLineWidthScale
INQUIRE POLYLINE COLOUR INDEX	L0a	GInqLineColrInd
INQUIRE POLYMARKER COLOUR INDEX	L0a	GInqMarkerColrInd
INQUIRE POLYMARKER SIZE SCALE FACTOR	L0a	GInqMarkerSizeScale
INQUIRE POLYMARKER TYPE	L0a	GInqMarkerType
INQUIRE TEXT COLOUR INDEX	L0a	GInqTextColrInd
INQUIRE TEXT FONT AND PRECISION	L0a	GInqTextFontPrec
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	L0a	GInqCurNormTranNum
INQUIRE CURRENT PICK IDENTIFIER	L1b	GInqCurPickId

Table 3 — GKS function names and Pascal names ordered by GKS function name

GKS Function Name	Level	Pascal Name
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	L0a	GInqCurPrimAttr
INQUIRE CHARACTER BASE VECTOR	L0a	GInqCharBaseVector
INQUIRE CHARACTER HEIGHT	L0a	GInqCharHeight
INQUIRE CHARACTER UP VECTOR	L0a	GInqCharUpVector
INQUIRE CHARACTER WIDTH	L0a	GInqCharWidth
INQUIRE FILL AREA INDEX	L0a	GInqFillInd
INQUIRE PATTERN REFERENCE POINT	L0a	GInqPatternRefPoint
INQUIRE PATTERN SIZE	L0a	GInqPatternSize
INQUIRE POLYLINE INDEX	L0a	GInqLineInd
INQUIRE POLYMARKER INDEX	L0a	GInqMarkerInd
INQUIRE TEXT ALIGNMENT	L0a	GInqTextAlign
INQUIRE TEXT INDEX	L0a	GInqTextInd
INQUIRE TEXT PATH	L0a	GInqTextPath
INQUIRE DEFAULT CHOICE DEVICE DATA	L0b	GInqDefChoiceDeviceData
INQUIRE DEFAULT CHOICE DEVICE DATA	L0b	GInqDefInputDeviceData(Choice
INQUIRE DEFAULT DEFERRAL STATE VALUES	L1a	GInqDefDeferSt
INQUIRE DEFAULT LOCATOR DEVICE DATA	L0b	GInqDefInputDeviceData(Locator
INQUIRE DEFAULT LOCATOR DEVICE DATA	L0b	GInqDefLocatorDeviceData
INQUIRE DEFAULT PICK DEVICE DATA	L1b	GInqDefInputDeviceData(Pick
INQUIRE DEFAULT PICK DEVICE DATA	L1b	GInqDefPickDeviceData
INQUIRE DEFAULT STRING DEVICE DATA	L0b	GInqDefInputDeviceData(String
INQUIRE DEFAULT STRING DEVICE DATA	L0b	GInqDefStringDeviceData
INQUIRE DEFAULT STROKE DEVICE DATA	L0b	GInqDefInputDeviceData(Stroke
INQUIRE DEFAULT STROKE DEVICE DATA	L0b	GInqDefStrokeDeviceData
INQUIRE DEFAULT VALUATOR DEVICE DATA	L0b	GInqDefInputDeviceData(Valuator
INQUIRE DEFAULT VALUATOR DEVICE DATA	L0b	GInqDefValuatorDeviceData
INQUIRE DISPLAY SPACE SIZE	L0a	GInqDisplaySize
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES	L1a	GInqDynModSegAttr
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	L1a	GInqDynModWsAttr
INQUIRE FILL AREA FACILITIES	L0a	GInqFillFacil
INQUIRE FILL AREA FACILITIES	L0a	GInqPrimFacil(FillArea
INQUIRE FILL AREA REPRESENTATION	L1a	GInqFillRep
INQUIRE FILL AREA REPRESENTATION	L1a	GInqPrimRep(FillArea
INQUIRE GENERALIZED DRAWING PRIMITIVE	L0a	GInqgdp
INQUIRE INPUT QUEUE OVERFLOW	L0c	GInqInputOverflow
INQUIRE LEVEL OF GKS	L0a	GInqLevelGKS
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	L0a	GInqListGDP
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	L0a	GInqListWsTypes
INQUIRE LIST OF COLOUR INDICES	L0a	GInqListColrInd
INQUIRE LIST OF FILL AREA INDICES	L1a	GInqListFillInd
INQUIRE LIST OF FILL AREA INDICES	L1a	GInqListPrimInd(FillArea
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS	L0a	GInqListNormTranNum
INQUIRE LIST OF PATTERN INDICES	L1a	GInqListPatternInd
INQUIRE LIST OF POLYLINE INDICES	L1a	GInqListPolylineInd
INQUIRE LIST OF POLYLINE INDICES	L1a	GInqListPrimInd(Polyline
INQUIRE LIST OF POLYMARKER INDICES	L1a	GInqListPolymarkerInd

Table 3 — GKS function names and Pascal names ordered by G KS function name

GKS Function Name	Level	Pascal Name
INQUIRE LIST OF POLYMARKER INDICES	L1a	GInqListPrimInd(Polymarker
INQUIRE LIST OF TEXT INDICES	L1a	GInqListPrimInd(Text
INQUIRE LIST OF TEXT INDICES	L1a	GInqListTextInd
INQUIRE LOCATOR DEVICE STATE	L0b	GInqInputDeviceSt(Locator
INQUIRE LOCATOR DEVICE STATE	L0b	GInqLocatorDeviceSt
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	L0a	GInqMaxWsSt
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	L0a	GInqMaxNormTranNum
INQUIRE MORE SIMULTANEOUS EVENTS	L0c	GInqMoreEvents
INQUIRE NAME OF OPEN SEGMENT	L1a	GInqOpenSeg
INQUIRE NORMALIZATION TRANSFORMATION	L0a	GInqNormTran
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	L0b	GInqNumInputDevices
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED	L1a	GInqNumSegPriorities
INQUIRE OPERATING STATE VALUE	L0a	GInqOpSt
INQUIRE PATTERN FACILITIES	L0a	GInqPatternFacil
INQUIRE PATTERN REPRESENTATION	L1a	GInqPatternRep
INQUIRE PICK DEVICE STATE	L1b	GInqInputDeviceSt(Pick
INQUIRE PICK DEVICE STATE	L1b	GInqPickDeviceSt
INQUIRE PIXEL	L0a	GInqPixel
INQUIRE PIXEL ARRAY	L0a	GInqPixelArray
INQUIRE PIXEL ARRAY DIMENSIONS	L0a	GInqPixelArrayDim
INQUIRE POLYLINE FACILITIES	L0a	GInqPolylineFacil
INQUIRE POLYLINE FACILITIES	L0a	GInqPrimFacil(Polyline
INQUIRE POLYLINE REPRESENTATION	L1a	GInqPolylineRep
INQUIRE POLYLINE REPRESENTATION	L1a	GInqPrimRep(Polyline
INQUIRE POLYMARKER FACILITIES	L0a	GInqPolymarkerFacil
INQUIRE POLYMARKER FACILITIES	L0a	GInqPrimFacil(Polymarker
INQUIRE POLYMARKER REPRESENTATION	L1a	GInqPolymarkerRep
INQUIRE POLYMARKER REPRESENTATION	L1a	GInqPrimRep(Polymarker
INQUIRE PREDEFINED COLOUR REPRESENTATION	L0a	GInqPredColrRep
INQUIRE PREDEFINED FILL AREA REPRESENTATION	L0a	GInqPredFillRep
INQUIRE PREDEFINED FILL AREA REPRESENTATION	L0a	GInqPredPrimRep(FillArea
INQUIRE PREDEFINED PATTERN REPRESENTATION	L0a	GInqPredPatternRep
INQUIRE PREDEFINED POLYLINE REPRESENTATION	L0a	GInqPredPolylineRep
INQUIRE PREDEFINED POLYLINE REPRESENTATION	L0a	GInqPredPrimRep(Polyline
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	L0a	GInqPredPolymarkerRep
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	L0a	GInqPredPrimRep(Polymarker
INQUIRE PREDEFINED TEXT REPRESENTATION	L0a	GInqPredPrimRep(Text
INQUIRE PREDEFINED TEXT REPRESENTATION	L0a	GInqPredTextRep
INQUIRE SEGMENT ATTRIBUTES	L1a	GInqSegAttr
INQUIRE SET OF ACTIVE WORKSTATIONS	L1a	GInqActiveWs
INQUIRE SET OF ASSOCIATED WORKSTATIONS	L1a	GInqAssocWs
INQUIRE SET OF OPEN WORKSTATIONS	L0a	GInqOpenWs
INQUIRE SET OF SEGMENT NAMES IN USE	L1a	GInqSegNames
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION	L1a	GInqSegNamesWs
INQUIRE STRING DEVICE STATE	L0b	GInqInputDeviceSt(String

Table 3 — GKS function names and Pascal names ordered by GKS function name

GKS Function Name	Level	Pascal Name
INQUIRE STRING DEVICE STATE	L0b	GInqStringDeviceSt
INQUIRE STROKE DEVICE STATE	L0b	GInqInputDeviceSt(Stroke
INQUIRE STROKE DEVICE STATE	L0b	GInqStrokeDeviceSt
INQUIRE TEXT EXTENT	L0a	GInqTextExtent
INQUIRE TEXT FACILITIES	L0a	GInqPrimFacil(Text
INQUIRE TEXT FACILITIES	L0a	GInqTextFacil
INQUIRE TEXT REPRESENTATION	L1a	GInqPrimRep(Text
INQUIRE TEXT REPRESENTATION	L1a	GInqTextRep
INQUIRE VALUATOR DEVICE STATE	L0b	GInqInputDeviceSt(Valuator
INQUIRE VALUATOR DEVICE STATE	L0b	GInqValuatorDeviceSt
INQUIRE WORKSTATION CATEGORY	L0a	GInqWsCategory
INQUIRE WORKSTATION CLASSIFICATION	L0a	GInqWsClass
INQUIRE WORKSTATION CONNECTION AND TYPE	L0a	GInqWsConnType
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	L0a	GInqWsDeferUpdSt
INQUIRE WORKSTATION MAXIMUM NUMBERS	L1a	GInqWsMaxNum
INQUIRE WORKSTATION STATE	L0a	GInqWsSt
INQUIRE WORKSTATION TRANSFORMATION	L0a	GInqWsTran
INSERT SEGMENT	L2a	GInsertSeg
INTERPRET ITEM	L0a	GInterpretItem
MESSAGE	L1a	GMessage
OPEN GKS	L0a	GOpengks
OPEN WORKSTATION	L0a	GOpenWs
POLYLINE	L0a	GPolyline
POLYMARKER	L0a	GPolymarker
READ ITEM FROM GKSM	L0a	GReadItem
REDRAW ALL SEGMENTS ON WORKSTATION	L1a	GRedrawSegWs
RENAME SEGMENT	L1a	GRenameSeg
REQUEST CHOICE	L0b	GReqChoice
REQUEST CHOICE	L0b	GReqInput(Choice
REQUEST LOCATOR	L0b	GReqInput(Locator
REQUEST LOCATOR	L0b	GReqLocator
REQUEST PICK	L1b	GReqInput(Pick
REQUEST PICK	L1b	GReqPick
REQUEST STRING	L0b	GReqInput(String
REQUEST STRING	L0b	GReqString
REQUEST STROKE	L0b	GReqInput(Stroke
REQUEST STROKE	L0b	GReqStroke
REQUEST VALUATOR	L0b	GReqInput(Valuator
REQUEST VALUATOR	L0b	GReqValuator
SAMPLE CHOICE	L0c	GSampleChoice
SAMPLE CHOICE	L0c	GSampleInput(Choice
SAMPLE LOCATOR	L0c	GSampleInput(Locator
SAMPLE LOCATOR	L0c	GSampleLocator
SAMPLE PICK	L1c	GSampleInput(Pick
SAMPLE PICK	L1c	GSamplePick
SAMPLE STRING	L0c	GSampleInput(String
SAMPLE STRING	L0c	GSampleString
SAMPLE STROKE	L0c	GSampleInput(Stroke
SAMPLE STROKE	L0c	GSampleStroke
SAMPLE VALUATOR	L0c	GSampleInput(Valuator

Table 3 — GKS function names and Pascal names ordered by GKS function name

GKS Function Name	Level	Pascal Name
SAMPLE VALUATOR	L0c	GSampleValuator
SELECT NORMALIZATION TRANSFORMATION	L0a	GSelectNormTran
SET ASPECT SOURCE FLAGS	L0a	GSetasf
SET CHARACTER EXPANSION FACTOR	L0a	GSetCharExpan
SET CHARACTER HEIGHT	L0a	GSetCharHeight
SET CHARACTER SPACING	L0a	GSetCharSpacing
SET CHARACTER UP VECTOR	L0a	GSetCharUpVector
SET CHOICE MODE	L0b	GSetChoiceMode
SET CHOICE MODE	L0b	GSetInputMode(Choice
SET CLIPPING INDICATOR	L0a	GSetClip
SET COLOUR REPRESENTATION	L0a	GSetColrRep
SET DEFERRAL STATE	L1a	GSetDeferSt
SET DETECTABILITY	L1b	GSetDet
SET FILL AREA COLOUR INDEX	L0a	GSetFillColrInd
SET FILL AREA INDEX	L0a	GSetFillInd
SET FILL AREA INDEX	L0a	GSetPrimInd(FillArea
SET FILL AREA INTERIOR STYLE	L0a	GSetFillIntStyle
SET FILL AREA REPRESENTATION	L1a	GSetFillRep
SET FILL AREA REPRESENTATION	L1a	GSetPrimRep(FillArea
SET FILL AREA STYLE INDEX	L0a	GSetFillStyleInd
SET HIGHLIGHTING	L1a	GSetHighlight
SET LINETYPE	L0a	GSetLineType
SET LINEWIDTH SCALE FACTOR	L0a	GSetLineWidthScale
SET LOCATOR MODE	L0b	GSetInputMode(Locator
SET LOCATOR MODE	L0b	GSetLocatorMode
SET MARKER SIZE SCALE FACTOR	L0a	GSetMarkerSizeScale
SET MARKER TYPE	L0a	GSetMarkerType
SET PATTERN REFERENCE POINT	L0a	GSetPatternRefPoint
SET PATTERN REPRESENTATION	L1a	GSetPatternRep
SET PATTERN SIZE	L0a	GSetPatternSize
SET PICK IDENTIFIER	L1b	GSetPickId
SET PICK MODE	L1b	GSetInputMode(Pick
SET PICK MODE	L1b	GSetPickMode
SET POLYLINE COLOUR INDEX	L0a	GSetLineColrInd
SET POLYLINE INDEX	L0a	GSetPolylineInd
SET POLYLINE INDEX	L0a	GSetPrimInd(Polyline
SET POLYLINE REPRESENTATION	L1a	GSetPolylineRep
SET POLYLINE REPRESENTATION	L1a	GSetPrimRep(Polyline
SET POLYMARKER COLOUR INDEX	L0a	GSetMarkerColrInd
SET POLYMARKER INDEX	L0a	GSetPolymarkerInd
SET POLYMARKER INDEX	L0a	GSetPrimInd(Polymarker
SET POLYMARKER REPRESENTATION	L1a	GSetPolymarkerRep
SET POLYMARKER REPRESENTATION	L1a	GSetPrimRep(Polymarker
SET SEGMENT PRIORITY	L1a	GSetSegPriority
SET SEGMENT TRANSFORMATION	L1a	GSetSegTran
SET STRING MODE	L0b	GSetInputMode(String
SET STRING MODE	L0b	GSetStringMode
SET STROKE MODE	L0b	GSetInputMode(Stroke
SET STROKE MODE	L0b	GSetStrokeMode
SET TEXT ALIGNMENT	L0a	GSetTextAlign
SET TEXT COLOUR INDEX	L0a	GSetTextColrInd
SET TEXT FONT AND PRECISION	L0a	GSetTextFontPrec

Table 3 — GKS function names and Pascal names ordered by GKS function name

GKS Function Name	Level	Pascal Name
SET TEXT INDEX	L0a	GSetPrimInd(Text
SET TEXT INDEX	L0a	GSetTextInd
SET TEXT PATH	L0a	GSetTextPath
SET TEXT REPRESENTATION	L1a	GSetPrimRep(Text
SET TEXT REPRESENTATION	L1a	GSetTextRep
SET VALUATOR MODE	L0b	GSetInputMode(Valuator
SET VALUATOR MODE	L0b	GSetValuatorMode
SET VIEWPORT	L0a	GSetViewport
SET VIEWPORT INPUT PRIORITY	L0b	GSetViewportPriority
SET VISIBILITY	L1a	GSetVis
SET WINDOW	L0a	GSetWindow
SET WORKSTATION VIEWPORT	L0a	GSetWsViewport
SET WORKSTATION WINDOW	L0a	GSetWsWindow
TEXT	L0a	GText
UPDATE WORKSTATION	L0a	GUpdWs
WRITE ITEM TO GKSM	L0a	GWriteItem
WRITE ITEM TO GKSM	L0a	GWriteItemGeneralized

Table 4 — GKS function names and Pascal names ordered by level

GKS Function Name	Level	Pascal Name
ACTIVATE WORKSTATION	L0a	GActivateWs
CELL ARRAY	L0a	GCellArray
CLEAR WORKSTATION	L0a	GClearWs
CLOSE GKS	L0a	GClosegks
CLOSE WORKSTATION	L0a	GCloseWs
DEACTIVATE WORKSTATION	L0a	GDeactivateWs
EMERGENCY CLOSE GKS	L0a	GEmergencyClosegks
ERROR HANDLING	L0a	GErrorHandling
ERROR LOGGING	L0a	GErrorLogging
ESCAPE	L0a	GEscape
ESCAPE	L0a	GEscapeGeneralized
FILL AREA	L0a	GFill
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDP
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDPGeneralized
GET ITEM TYPE FROM GKSM	L0a	GGetItemType
INQUIRE ASPECT SOURCE FLAGS	L0a	GInqasf
INQUIRE CHARACTER BASE VECTOR	L0a	GInqCharBaseVector
INQUIRE CHARACTER EXPANSION FACTOR	L0a	GInqCharExpan
INQUIRE CHARACTER HEIGHT	L0a	GInqCharHeight
INQUIRE CHARACTER SPACING	L0a	GInqCharSpacing
INQUIRE CHARACTER UP VECTOR	L0a	GInqCharUpVector
INQUIRE CHARACTER WIDTH	L0a	GInqCharWidth
INQUIRE CLIPPING	L0a	GInqClip
INQUIRE COLOUR FACILITIES	L0a	GInqColrFacil
INQUIRE COLOUR REPRESENTATION	L0a	GInqColrRep
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	L0a	GInqCurIndivAttr
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	L0a	GInqCurNormTranNum
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	L0a	GInqCurPrimAttr
INQUIRE DISPLAY SPACE SIZE	L0a	GInqDisplaySize
INQUIRE FILL AREA COLOUR INDEX	L0a	GInqFillColrInd
INQUIRE FILL AREA FACILITIES	L0a	GInqFillFacil
INQUIRE FILL AREA INDEX	L0a	GInqFillInd
INQUIRE FILL AREA INTERIOR STYLE	L0a	GInqFillIntStyle

Table 4 — GKS function names and Pascal names ordered by level

GKS Function Name	Level	Pascal Name
INQUIRE FILL AREA STYLE INDEX	L0a	GInqFillStyleInd
INQUIRE GENERALIZED DRAWING PRIMITIVE	L0a	GInqgdp
INQUIRE LEVEL OF GKS	L0a	GInqLevelgks
INQUIRE POLYLINE COLOUR INDEX	L0a	GInqLineColrInd
INQUIRE POLYLINE INDEX	L0a	GInqLineInd
INQUIRE LINETYPE	L0a	GInqLineType
INQUIRE LINEWIDTH SCALE FACTOR	L0a	GInqLineWidthScale
INQUIRE LIST OF COLOUR INDICES	L0a	GInqListColrInd
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	L0a	GInqListgdp
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS	L0a	GInqListNormTranNum
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	L0a	GInqListWsTypes
INQUIRE POLYMARKER COLOUR INDEX	L0a	GInqMarkerColrInd
INQUIRE POLYMARKER INDEX	L0a	GInqMarkerInd
INQUIRE POLYMARKER SIZE SCALE FACTOR	L0a	GInqMarkerSizeScale
INQUIRE POLYMARKER TYPE	L0a	GInqMarkerType
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	L0a	GInqMaxNormTranNum
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	L0a	GInqMaxWsSt
INQUIRE NORMALIZATION TRANSFORMATION	L0a	GInqNormTran
INQUIRE OPERATING STATE VALUE	L0a	GInqOpSt
INQUIRE SET OF OPEN WORKSTATIONS	L0a	GInqOpenWs
INQUIRE PATTERN FACILITIES	L0a	GInqPatternFacil
INQUIRE PATTERN REFERENCE POINT	L0a	GInqPatternReffPoint
INQUIRE PATTERN SIZE	L0a	GInqPatternSize
INQUIRE PIXEL	L0a	GInqPixel
INQUIRE PIXEL ARRAY	L0a	GInqPixelArray
INQUIRE PIXEL ARRAY DIMENSIONS	L0a	GInqPixelArrayDim
INQUIRE POLYLINE FACILITIES	L0a	GInqPolylineFacil
INQUIRE POLYMARKER FACILITIES	L0a	GInqPolymarkerFacil
INQUIRE PREDEFINED COLOUR REPRESENTATION	L0a	GInqPredColrRep
INQUIRE PREDEFINED FILL AREA REPRESENTATION	L0a	GInqPredFillRep
INQUIRE PREDEFINED PATTERN REPRESENTATION	L0a	GInqPredPatternRep
INQUIRE PREDEFINED POLYLINE REPRESENTATION	L0a	GInqPredPolylineRep
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	L0a	GInqPredPolymarkerRep
INQUIRE PREDEFINED FILL AREA REPRESENTATION	L0a	GInqPredPrimRep(FillArea
INQUIRE PREDEFINED POLYLINE REPRESENTATION	L0a	GInqPredPrimRep(Polyline
INQUIRE PREDEFINED POLYMARKER REPRESENTATION	L0a	GInqPredPrimRep(Polymarker
INQUIRE PREDEFINED TEXT REPRESENTATION	L0a	GInqPredPrimRep(Text
INQUIRE PREDEFINED TEXT REPRESENTATION	L0a	GInqPredTextRep
INQUIRE FILL AREA FACILITIES	L0a	GInqPrimFacil(FillArea
INQUIRE POLYLINE FACILITIES	L0a	GInqPrimFacil(Polyline
INQUIRE POLYMARKER FACILITIES	L0a	GInqPrimFacil(Polymarker
INQUIRE TEXT FACILITIES	L0a	GInqPrimFacil(Text
INQUIRE TEXT ALIGNMENT	L0a	GInqTextAlign
INQUIRE TEXT COLOUR INDEX	L0a	GInqTextColrInd
INQUIRE TEXT EXTENT	L0a	GInqTextExtent
INQUIRE TEXT FACILITIES	L0a	GInqTextFacil
INQUIRE TEXT FONT AND PRECISION	L0a	GInqTextFontPrec
INQUIRE TEXT INDEX	L0a	GInqTextInd
INQUIRE TEXT PATH	L0a	GInqTextPath

Table 4 — GKS function names and Pascal names ordered by level

GKS Function Name	Level	Pascal Name
INQUIRE WORKSTATION CATEGORY	L0a	GInqWsCategory
INQUIRE WORKSTATION CLASSIFICATION	L0a	GInqWsClass
INQUIRE WORKSTATION CONNECTION AND TYPE	L0a	GInqWsConnType
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	L0a	GInqWsDeferUpdSt
INQUIRE WORKSTATION STATE	L0a	GInqWsSt
INQUIRE WORKSTATION TRANSFORMATION	L0a	GInqWsTran
INTERPRET ITEM	L0a	GInterpretItem
OPEN GKS	L0a	GOpenGKS
OPEN WORKSTATION	L0a	GOpenWs
POLYLINE	L0a	GPolyline
POLYMARKER	L0a	GPolymarker
READ ITEM FROM GKSM	L0a	GReadItem
SELECT NORMALIZATION TRANSFORMATION	L0a	GSelectNormTran
SET ASPECT SOURCE FLAGS	L0a	GSetASF
SET CHARACTER EXPANSION FACTOR	L0a	GSetCharExpan
SET CHARACTER HEIGHT	L0a	GSetCharHeight
SET CHARACTER SPACING	L0a	GSetCharSpacing
SET CHARACTER UP VECTOR	L0a	GSetCharUpVector
SET CLIPPING INDICATOR	L0a	GSetClip
SET COLOUR REPRESENTATION	L0a	GSetColrRep
SET FILL AREA COLOUR INDEX	L0a	GSetFillColrInd
SET FILL AREA INDEX	L0a	GSetFillInd
SET FILL AREA INTERIOR STYLE	L0a	GSetFillIntStyle
SET FILL AREA STYLE INDEX	L0a	GSetFillStyleInd
SET POLYLINE COLOUR INDEX	L0a	GSetLineColrInd
SET LINETYPE	L0a	GSetLineType
SET LINEWIDTH SCALE FACTOR	L0a	GSetLineWidthScale
SET POLYMARKER COLOUR INDEX	L0a	GSetMarkerColrInd
SET MARKER SIZE SCALE FACTOR	L0a	GSetMarkerSizeScale
SET MARKER TYPE	L0a	GSetMarkerType
SET PATTERN REFERENCE POINT	L0a	GSetPatternRefPoint
SET PATTERN SIZE	L0a	GSetPatternSize
SET POLYLINE INDEX	L0a	GSetPolylineInd
SET POLYMARKER INDEX	L0a	GSetPolymarkerInd
SET FILL AREA INDEX	L0a	GSetPrimInd(FillArea)
SET POLYLINE INDEX	L0a	GSetPrimInd(Polyline)
SET POLYMARKER INDEX	L0a	GSetPrimInd(Polymarker)
SET TEXT INDEX	L0a	GSetPrimInd(Text)
SET TEXT ALIGNMENT	L0a	GSetTextAlign
SET TEXT COLOUR INDEX	L0a	GSetTextColrInd
SET TEXT FONT AND PRECISION	L0a	GSetTextFontPrec
SET TEXT INDEX	L0a	GSetTextInd
SET TEXT PATH	L0a	GSetTextPath
SET VIEWPORT	L0a	GSetViewport
SET WINDOW	L0a	GSetWindow
SET WORKSTATION VIEWPORT	L0a	GSetWsViewport
SET WORKSTATION WINDOW	L0a	GSetWsWindow
TEXT	L0a	GText
UPDATE WORKSTATION	L0a	GUpdWs
WRITE ITEM TO GKSM	L0a	GWriteItem
WRITE ITEM TO GKSM	L0a	GWriteItemGeneralized
INITIALISE CHOICE	L0b	GInitChoice

Table 4 — GKS function names and Pascal names ordered by level

GKS Function Name	Level	Pascal Name
INITIALISE CHOICE	L0b	GInitInput(Choice
INITIALISE LOCATOR	L0b	GInitInput(Locator
INITIALISE STRING	L0b	GInitInput(String
INITIALISE STROKE	L0b	GInitInput(Stroke
INITIALISE VALUATOR	L0b	GInitInput(Valuator
INITIALISE LOCATOR	L0b	GInitLocator
INITIALISE STRING	L0b	GInitString
INITIALISE STROKE	L0b	GInitStroke
INITIALISE VALUATOR	L0b	GInitValuator
INQUIRE CHOICE DEVICE STATE	L0b	GInqChoiceDeviceSt
INQUIRE DEFAULT CHOICE DEVICE DATA	L0b	GInqDefChoiceDeviceData
INQUIRE DEFAULT CHOICE DEVICE DATA	L0b	GInqDefInputDeviceData(Choice
INQUIRE DEFAULT LOCATOR DEVICE DATA	L0b	GInqDefInputDeviceData(Locator
INQUIRE DEFAULT STRING DEVICE DATA	L0b	GInqDefInputDeviceData(String
INQUIRE DEFAULT STROKE DEVICE DATA	L0b	GInqDefInputDeviceData(Stroke
INQUIRE DEFAULT VALUATOR DEVICE DATA	L0b	GInqDefInputDeviceData(Valuator
INQUIRE DEFAULT LOCATOR DEVICE DATA	L0b	GInqDefLocatorDeviceData
INQUIRE DEFAULT STRING DEVICE DATA	L0b	GInqDefStringDeviceData
INQUIRE DEFAULT STROKE DEVICE DATA	L0b	GInqDefStrokeDeviceData
INQUIRE DEFAULT VALUATOR DEVICE DATA	L0b	GInqDefValuatorDeviceData
INQUIRE CHOICE DEVICE STATE	L0b	GInqInputDeviceSt(Choice
INQUIRE LOCATOR DEVICE STATE	L0b	GInqInputDeviceSt(Locator
INQUIRE STRING DEVICE STATE	L0b	GInqInputDeviceSt(String
INQUIRE STROKE DEVICE STATE	L0b	GInqInputDeviceSt(Stroke
INQUIRE VALUATOR DEVICE STATE	L0b	GInqInputDeviceSt(Valuator
INQUIRE LOCATOR DEVICE STATE	L0b	GInqLocatorDeviceSt
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	L0b	GInqNumInputDevices
INQUIRE STRING DEVICE STATE	L0b	GInqStringDeviceSt
INQUIRE STROKE DEVICE STATE	L0b	GInqStrokeDeviceSt
INQUIRE VALUATOR DEVICE STATE	L0b	GInqValuatorDeviceSt
REQUEST CHOICE	L0b	GReqChoice
REQUEST CHOICE	L0b	GReqInput(Choice
REQUEST LOCATOR	L0b	GReqInput(Locator
REQUEST STRING	L0b	GReqInput(String
REQUEST STROKE	L0b	GReqInput(Stroke
REQUEST VALUATOR	L0b	GReqInput(Valuator
REQUEST LOCATOR	L0b	GReqLocator
REQUEST STRING	L0b	GReqString
REQUEST STROKE	L0b	GReqStroke
REQUEST VALUATOR	L0b	GReqValuator
SET CHOICE MODE	L0b	GSetChoiceMode
SET CHOICE MODE	L0b	GSetInputMode(Choice
SET LOCATOR MODE	L0b	GSetInputMode(Locator
SET STRING MODE	L0b	GSetInputMode(String
SET STROKE MODE	L0b	GSetInputMode(Stroke
SET VALUATOR MODE	L0b	GSetInputMode(Valuator
SET LOCATOR MODE	L0b	GSetLocatorMode
SET STRING MODE	L0b	GSetStringMode
SET STROKE MODE	L0b	GSetStrokeMode
SET VALUATOR MODE	L0b	GSetValuatorMode
SET VIEWPORT INPUT PRIORITY	L0b	GSetViewportPriority
AWAIT EVENT	L0c	GAwaitEvent
FLUSH DEVICE EVENTS	L0c	GFlushDeviceEvents
GET CHOICE	L0c	GGetChoice

Table 4 — GKS function names and Pascal names ordered by level

GKS Function Name	Level	Pascal Name
GET CHOICE	L0c	GGetInput(Choice
GET LOCATOR	L0c	GGetInput(Locator
GET STRING	L0c	GGetInput(String
GET STROKE	L0c	GGetInput(Stroke
GET VALUATOR	L0c	GGetInput(Valuator
GET LOCATOR	L0c	GGetLocator
GET STRING	L0c	GGetString
GET STROKE	L0c	GGetStroke
GET VALUATOR	L0c	GGetValuator
INQUIRE INPUT QUEUE OVERFLOW	L0c	GInqInputOverflow
INQUIRE MORE SIMULTANEOUS EVENTS	L0c	GInqMoreEvents
SAMPLE CHOICE	L0c	GSampleChoice
SAMPLE CHOICE	L0c	GSampleInput(Choice
SAMPLE LOCATOR	L0c	GSampleInput(Locator
SAMPLE STRING	L0c	GSampleInput(String
SAMPLE STROKE	L0c	GSampleInput(Stroke
SAMPLE VALUATOR	L0c	GSampleInput(Valuator
SAMPLE LOCATOR	L0c	GSampleLocator
SAMPLE STRING	L0c	GSampleString
SAMPLE STROKE	L0c	GSampleStroke
SAMPLE VALUATOR	L0c	GSampleValuator
ACCUMULATE TRANSFORMATION MATRIX	L1a	GAccumTran
CLOSE SEGMENT	L1a	GCloseSeg
CREATE SEGMENT	L1a	GCreateSeg
DELETE SEGMENT	L1a	GDelSeg
DELETE SEGMENT FROM WORKSTATION	L1a	GDelSegWs
EVALUATE TRANSFORMATION MATRIX	L1a	GEvalTran
INQUIRE SET OF ACTIVE WORKSTATIONS	L1a	GInqActiveWs
INQUIRE SET OF ASSOCIATED WORKSTATIONS	L1a	GInqAssocWs
INQUIRE DEFAULT DEFERRAL STATE VALUES	L1a	GInqDefDeferSt
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES	L1a	GInqDynModSegAttr
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	L1a	GInqDynModWsAttr
INQUIRE FILL AREA REPRESENTATION	L1a	GInqFillRep
INQUIRE LIST OF FILL AREA INDICES	L1a	GInqListFillInd
INQUIRE LIST OF PATTERN INDICES	L1a	GInqListPatternInd
INQUIRE LIST OF POLYLINE INDICES	L1a	GInqListPolylineInd
INQUIRE LIST OF POLYMARKER INDICES	L1a	GInqListPolymarkerInd
INQUIRE LIST OF FILL AREA INDICES	L1a	GInqListPrimInd(FillArea
INQUIRE LIST OF POLYLINE INDICES	L1a	GInqListPrimInd(Polyline
INQUIRE LIST OF POLYMARKER INDICES	L1a	GInqListPrimInd(Polymarker
INQUIRE LIST OF TEXT INDICES	L1a	GInqListPrimInd(Text
INQUIRE LIST OF TEXT INDICES	L1a	GInqListTextInd
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED	L1a	GInqNumSegPriorities
INQUIRE NAME OF OPEN SEGMENT	L1a	GInqOpenSeg
INQUIRE PATTERN REPRESENTATION	L1a	GInqPatternRep
INQUIRE POLYLINE REPRESENTATION	L1a	GInqPolylineRep
INQUIRE POLYMARKER REPRESENTATION	L1a	GInqPolymarkerRep
INQUIRE FILL AREA REPRESENTATION	L1a	GInqPrimRep(FillArea
INQUIRE POLYLINE REPRESENTATION	L1a	GInqPrimRep(Polyline
INQUIRE POLYMARKER REPRESENTATION	L1a	GInqPrimRep(Polymarker

Table 4 — GKS function names and Pascal names ordered by level

GKS Function Name	Level	Pascal Name
INQUIRE TEXT REPRESENTATION	L1a	GInqPrimRep(Text
INQUIRE SEGMENT ATTRIBUTES	L1a	GInqSegAttr
INQUIRE SET OF SEGMENT NAMES IN USE	L1a	GInqSegNames
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION	L1a	GInqSegNamesWs
INQUIRE TEXT REPRESENTATION	L1a	GInqTextRep
INQUIRE WORKSTATION MAXIMUM NUMBERS	L1a	GInqWsMaxNum
MESSAGE	L1a	GMessage
REDRAW ALL SEGMENTS ON WORKSTATION	L1a	GRedrawSegWs
RENAME SEGMENT	L1a	GRenameSeg
SET DEFERRAL STATE	L1a	GSetDeferSt
SET FILL AREA REPRESENTATION	L1a	GSetFillRep
SET HIGHLIGHTING	L1a	GSetHighlight
SET PATTERN REPRESENTATION	L1a	GSetPatternRep
SET POLYLINE REPRESENTATION	L1a	GSetPolylineRep
SET POLYMARKER REPRESENTATION	L1a	GSetPolymarkerRep
SET FILL AREA REPRESENTATION	L1a	GSetPrimRep(FillArea
SET POLYLINE REPRESENTATION	L1a	GSetPrimRep(Polyline
SET POLYMARKER REPRESENTATION	L1a	GSetPrimRep(Polymarker
SET TEXT REPRESENTATION	L1a	GSetPrimRep(Text
SET SEGMENT PRIORITY	L1a	GSetSegPriority
SET SEGMENT TRANSFORMATION	L1a	GSetSegTran
SET TEXT REPRESENTATION	L1a	GSetTextRep
SET VISIBILITY	L1a	GSetVis
INITIALISE PICK	L1b	GInitInput(Pick
INITIALISE PICK	L1b	GInitPick
INQUIRE CURRENT PICK IDENTIFIER	L1b	GInqCurPickId
INQUIRE DEFAULT PICK DEVICE DATA	L1b	GInqDefInputDeviceData(Pick
INQUIRE DEFAULT PICK DEVICE DATA	L1b	GInqDefPickDeviceData
INQUIRE PICK DEVICE STATE	L1b	GInqInputDeviceSt(Pick
INQUIRE PICK DEVICE STATE	L1b	GInqPickDeviceSt
REQUEST PICK	L1b	GReqInput(Pick
REQUEST PICK	L1b	GReqPick
SET DETECTABILITY	L1b	GSetDet
SET PICK MODE	L1b	GSetInputMode(Pick
SET PICK IDENTIFIER	L1b	GSetPickId
SET PICK MODE	L1b	GSetPickMode
GET PICK	L1c	GGetInput(Pick
GET PICK	L1c	GGetPick
SAMPLE PICK	L1c	GSampleInput(Pick
SAMPLE PICK	L1c	GSamplePick
ASSOCIATE SEGMENT WITH WORKSTATION	L2a	GAssocSegWs
COPY SEGMENT TO WORKSTATION	L2a	GCopySegWs
INSERT SEGMENT	L2a	GInsertSeg

3.7 Representation of GKS data types

Clause 6 of ISO 7942 defines GKS data structures which are an integral part of the system. These data structures are bound as the following Pascal data structures.

- The GKS types integer and real are mapped to Pascal integer and real. Where GKS specifies a subrange of integer, that subrange is used.
- GKS enumeration types are mapped to Pascal enumerated types (or subranges of enumerated types), often with the addition of sentinel words or characters to avoid name clashes. For example, the GKS type (HIGHER, LOWER) in the SET VIEWPORT INPUT PRIORITY function is in Pascal the type GEPriority with elements (GVHigher, GVLower).

c) String is represented as a fixed length string (GAStrng) when it is a component of another data structure. In Pascal Level 1, string is represented as a packed conformant array of characters when the string is a parameter to a Pascal procedure. In Pascal Level 0, string is mapped to the fixed length type (GAStrng).

d) A GKS point is mapped to the type GRPoint in Pascal, which is a record with x,y(: real) fields. Conformant arrays of GRPoint are used in Level 1 procedure definitions, and fixed length arrays of GRPoint (GAPointArray) in Level 0.

e) Sets in GKS are represented as arrays in Pascal when the cardinality of the set is potentially infinite (for example, the set of workstation identifiers). Otherwise GKS sets are Pascal sets — the only two being the set of primitives associated with a GENERALIZED DRAWING PRIMITIVE, and the set of fill area interior styles.

f) The GKS data records are represented as Pascal records.

g) More complex data structures are represented as Pascal records and arrays.

The definition of all of the data structures is given in clause 5.

3.8 Naming conventions for data types

A consistent naming scheme for data types is used.

- a) Constants are prefixed with “GC”.
- b) Scalar types are prefixed with “GT”.
- c) Enumerated types are prefixed with “GE”.
- d) Values of enumerated types are prefixed with “GV”.
- e) Arrays are prefixed with “GA”.
- f) Records are prefixed with “GR”.
- g) Sets are prefixed with “GS”.

Mixtures of upper and lower case letters are used extensively in the type and procedure names. This is to aid readability and is not otherwise significant.

3.9 Implementation-dependent characteristics

There are a number of implementation-defined characteristics necessary in a Pascal interface.

Resolutions of such implementation-dependent features shall be detailed in the documentation of a Pascal implementation.

- a) The error reporting file name specified by GKS as a parameter to the OPEN GKS function is represented as an implementation-defined type, GTErrrFileNm. This type and the constant GCDefErrrLog shall be defined so that GCDefErrrLog is compatible with GTErrrFileNm.
- b) The workstation connection identifier, specified by GKS as a parameter to the OPEN WORKSTATION function, is represented in Pascal as a string. The implementation documentation shall describe the meaning and use of the string.
- c) There are a number of implementation-defined constants, which are shown in 5.1, and implementation-defined types, which are shown in 5.2.
- d) The GKS Data Record is represented as a number of types depending on the purpose of the records. GRLocatorData, GRStrokeData, GRValuatorData, GRChoiceData, GRPickData and GRStringData are used in the INITIALISE INPUT functions and the corresponding inquiry functions. GREscapeDataIn and GREscapeDataOut are for use with the ESCAPE function. GRGDpData is for use with the GENERALIZED DRAWING PRIMITIVE. The form of these records is described in 3.10.
- e) ISO 7185 does not prohibit separate compilation, but does not give a standard manner for achieving this. Separate compilation is crucial for GKS, since an implementation is not feasible unless the underlying Pascal supports separate compilation. The implementation documentation shall specify how data types and procedure definitions are to be imported into the application environment and the means of binding the physical GKS implementation with the program.
- f) The OPEN GKS function (GOpenGKS) has parameters “ErrorFile” and “MemoryUnits”. Constant values, GCDefErrrLog and GCDefMemory, exist which will evoke the default behaviour of the implementation. The implementation documentation shall describe the meaning and behaviour associated with these default values.

3.10 Data Records Subject to Registration

The GKS Data Records described above are all defined in a similar manner. The implementation documentation shall detail the structure of these records, and the implementation may provide procedures for constructing and manipulating such records.

The form for each of these records is:

```
record
  case tag : TagType of
    - 99 : (U0099field1; U0099field2; U0099field3);
    - 27 : (U0027field1; U0027field2);
    15 : (R0015field1; R0015field2; R0015field3; R0015field4);
    42 : (R0042field1)
  end;
```

NOTE This record does not reflect exact Pascal syntax, but indicates the type of information which the record shall contain. Each field identifier is to be prefixed with a representation of the tag value. The representation is constructed as:

<reg/unreg> <tag value>

Where <reg/unreg> is “R” for a registered entry or “U” for an unregistered entry, and <tag value> is a string representing the absolute value of the tag.

3.11 Return Parameter Arrays

Several inquiry functions return a variable amount of information depending on the implementation, the workstations in use, or the characteristics of the calling program. The corresponding Pascal procedures have additional parameters which allow the calling program to specify a subset of information to be returned.

Three parameters are added to these procedures. They are defined as follows:

- a) Start: specifies an index into the list of available values. This may take on the values 1 to the size of the list. If the value of Start is outside of this range, an error is generated.
- b) Size: specifies the number of values to be returned. Size is constrained to be less than or equal to the size of the array used to pass information.
- c) Done: is set to TRUE if the last of the values is returned by the call. It is set to FALSE if additional values exist beyond the last value returned by the call.

3.12 Level of Pascal

ISO 7185 has two levels of conformance. Level 0 Pascal does not have conformant array parameters whereas Level 1 does.

This Pascal binding to GKS may be used with Level 0 or Level 1 Pascal. For Level 0, the conformant array parameters are replaced by parameters of the appropriate fixed length array type. All of the fixed length types needed are defined in this document. The names of the procedures are the same under Level 0 and Level 1 Pascal. However, those procedures which use the fixed length array types are headed in clause 6 as “Pascal Level 0”. A Pascal binding implementation shall provide all of the procedures and data types defined in clauses 5 and 6.

An implementation which does not use conformant arrays, that is, which is based on a Level 0 Pascal, is called a Level 0 implementation. Likewise, a Level 1 implementation is one which makes use of conformant arrays. An applications program written for a Level 0 Pascal implementation also works on a Level 1 Pascal implementation. In addition an applications program written for a Level 1 Pascal implementation works on a Level 0 implementation provided that the following rules are adopted:

- a) The fixed length arrays provided in the binding must be used in the application. For example, in calls to GPolyline, the type GAPointArray must be used.
- b) The size of the arrays must not exceed the maximum sizes as specified by the constants (for example, GCMaxPoint for arrays of points).

Some examples are given to clarify these points:

Example 1

This only works with Level 1.

```
var p : array[1..MAX] of GRPoint;
    n : GTInt2;
GPolyline(n,p);
```

Example 2

This works with both levels.

```
var p : GAPointArray;
    n : GTInt2;
GPolyline(n,p);
```

Example 3

This works with both levels.

```
var status : GReqStatus;
    return : GRInput;
GReqInput(GVStroke,DISPLAY,1,status,return);
if status = GVStatusOK then
with return do GFill(Num,Points);
```

Example 4

This works only with Level 1.

```
var status : GReqStatus;
    return : GRInput;
GReqInput(GVLocator,DISPLAY,1,status,return);
if status = GVStatusOK then
with return do GText(Position,11,"Hello world");
```

Example 5

This works with both levels.

```
var LocStatus,StrStatus : GReqStatus;
    LocReturn,StrReturn : GRInput;
GReqInput(GVString,DISPLAY,1,StrStatus,StrReturn);
GReqInput(GVLocator,DISPLAY,1,LocStatus,LocReturn);
if((LocStatus=GVStatusOK) and (StrStatus=GVStatusOK)) then
with StrReturn do
with LocReturn do
GText(Position,StringLength,CharString);
```

NOTE These examples do not reflect exact Pascal syntax, but indicate how to use the Level 0 and Level 1 interfaces.

3.13 Registration¹⁾

ISO 7942 reserves certain value ranges for registration as graphical items. The registered graphical items will be bound to Pascal (and other programming languages). The registered item bindings will be consistent with the binding presented in this part of ISO 8651.

4 Error handling**4.1 The error handling function**

Error handling shall be carried out as described in subclause 4.12 of ISO 7942. The implementation documentation shall detail the method for an application program to provide an ERROR HANDLING procedure. The default ERROR HANDLING procedure shall be available.

¹⁾ For the purpose of this International Standard and according to the rules for the designation and operation of registration authorities in the ISO Directives, the ISO Council has designated the National Bureau of Standards (Institute of Computer Sciences and Technology), B-154 Technology Building, Gaithersburg, MD 20899, USA to act as registration authority.

The ERROR HANDLING and ERROR LOGGING procedures each have a parameter “identification of the GKS procedure which caused the error detection”. This is represented in Pascal as a string parameter. The values of the strings identifying Pascal GKS procedures are those strings from the column headed “Pascal Name” in Table 2, Table 3 or Table 4.

4.2 Pascal specific GKS errors

Certain features of the Pascal language make additional errors (beyond the ones described in ISO 7942) possible. Specifically, these new errors are defined:

2100 There is an incompatibility between the bounds of the array and the actual size parameters specified

This error is invoked when the parameter which specifies size or length is not compatible with the bounds of a conformant array. For example, in Level 1 GPolyline, NumPoints must not be greater than (max-min + 1).

2101 The supplied array is too small to store the required data

This error is invoked when the data requested in an inquiry exceeds the size of the array passed to the procedure.

2102 List element or set member not available

This error is invoked when a value greater than the size of a list or set was passed as the requested start element in an inquiry routine.

2103 Pick is not supported in this level of GKS

This error is invoked when GVPick is passed as a parameter to a Level 0 implementation.

5 Pascal GKS data structures

The following sections contain the data structures defined for the Pascal interface to GKS. The definitions are given in alphabetical order for each of the sections: implementation-defined constants and types, constants, enumerated types, array types, set types and record types.

For some of those data types involving a reference to pick input, the pick input field is valid only at certain levels of GKS.

5.1 Implementation-defined constants

These are suggested minimum values for the required implementation-defined constants. The implementation shall provide values appropriate for that specific implementation.

GCDerErrorLog	= 1;	{Default error log identifier}
GCMaEscapeIn	= 10;	{Maximum number of input data record items for generalized ESCAPE}
GCMaEscapeOut	= 10;	{Maximum number of output data record items for generalized ESCAPE}
GCMaGDP	= 10;	{Maximum number of data record items for generalized GDP}
GCMaDX	= 100;	{Maximum size of two dimensional arrays of colour}
GCMaDY	= 100;	{indices used in fill area and pattern representation}
GCMaFile	= 12;	{Maximum length of strings representing file names}
GCMaInq	= 10;	{Maximum number of elements returned by an inquiry}
GCMaItem	= 10;	{Maximum number of data record items for generalized WRITE ITEM}
GCMaMemory	= 32767;	{Maximum number of memory units allowed}
GCMaName	= 32;	{Maximum length of strings representing procedure names}
GCMaPoint	= 128;	{Maximum number of points in a point array}
GCMaString	= 80;	{Maximum length of fixed length strings}

5.2 Implementation-defined types

These types shall be fully defined or modified to match the capabilities of the implementation.

5.2.1 General types

GTChoiceDataTag	= 1..1;	{Range of valid Choice prompt and echo types}
GTErrrorFileName	= INTEGER;	{Implementation-defined type for defining error log}
GTEscapeDataTag	= 0..0;	{Range of valid Escape ID's}
GTGDPDataTag	= 0..0;	{Range of valid GDP ID's}
GTLocatorDataTag	= 1..1;	{Range of valid Locator prompt and echo types}
GTPickDataTag	= 1..1;	{Range of valid Pick prompt and echo types}
GTStringDataTag	= 1..1;	{Range of valid String prompt and echo types}
GTStrokeDataTag	= 1..1;	{Range of valid Stroke prompt and echo types}
GTValuatorDataTag	= 1..1;	{Range of valid Valuator prompt and echo types}

5.2.2 Record types

The fields of registered data records are defined in the ISO International Register of Graphical Items which is maintained by the Registration Authority. The form of these data records is described in 3.10. GRFileData is not a registered data record.

GRChoiceData	= record	
	case Prompt	: GTChoiceDataTag of
	1	: ();{Implementation-defined}
		{Implementation-defined record for initialising choice input}
	end;	
GREscapeDataIn	= record	
	case EscapeId	: GTEscapeDataTag of
	0	: (); {The value 0 will never be used}
		{Implementation-defined record for use with the Escape function}
	end;	
GREscapeDataOut	= record	
	case EscapeId	: GTEscapeDataTag of
	0	: (); {The value 0 will never be used}
		{Implementation-defined record for use with the Escape function}
	end;	
GRFileData	= record	
	Dummy	: BOOLEAN; {Replace with data record}
		{Implementation-defined record for use with the Metafile functions}
	end;	
GRGDPData	= record	
	case GDPId	: GTGDPDataTag of
	0	: (); {The value 0 will never be used}
		{Implementation-defined record for use with GDP function}
	end;	
GRLocatorData	= record	
	case Prompt	: GTLocatorDataTag of
	1	: ();{Implementation-defined}
		{Implementation-defined record for initialising locator input}
	end;	
GRPickData	= record	
	case Prompt	: GTPickDataTag of
	1	: ();{Implementation-defined}
		{Implementation-defined record for initialising pick input}
	end;	

```

GRStringData      = record
    case Prompt          : GTStringDataTag of
        1                : ();{Implementation-defined}
    {Implementation-defined record for initialising string input}
end;

GRStrokeData      = record
    case Prompt          : GTStrokeDataTag of
        1                : ();{Implementation-defined}
    {Implementation-defined record for initialising stroke input}
end;

GRValuatorData    = record
    case Prompt          : GTValuatorDataTag of
        1                : ();{Implementation-defined}
    {Implementation-defined record for initialising valuator input}
end;

```

5.3 Required constants

```

GCCircleMarker    = 4;
GCCrossMarker     = 5;
GCDashDotLine     = 4;
GCDashedLine      = 2;
GCDefMemory       = - 1;
GCDotMarker       = 1;
GCDottedLine      = 3;
GCPlusMarker      = 2;
GCSolidLine       = 1;
GCStarMarker      = 3;

```

5.4 General types

```

GTInqSize         = 1..GCMaxInq;
GTInt0            = 0..MAXINT;
GTInt1            = 1..MAXINT;
GTInt2            = 2..MAXINT;
GTInt3            = 3..MAXINT;
GTMaxDX           = 1..GCMaxDX;
GTMaxDY           = 1..GCMaxDY;
GTMaxEscapeIn     = 1..GCMaxEscapeIn;
GTMaxEscapeOut    = 1..GCMaxEscapeOut;
GTMaxGDP          = 1..GCMaxGDP;
GTMaxItem         = 1..GCMaxItem;
GTMaxPoint0       = 0..GCMaxPoint;
GTMaxPoint1       = 1..GCMaxPoint;
GTMaxPoint2       = 2..GCMaxPoint;
GTMaxPoint3       = 3..GCMaxPoint;
GTMaxString       = 1..GCMaxString;
GTMemory          = GCDefMemory..GCMaxMemory;

```

5.5 Names used by GKS

```

GTEscapeId        = INTEGER;
GTGDPIId          = INTEGER;
GTPickId          = INTEGER;
GTSeg             = INTEGER;
GTWsType          = INTEGER;
GTWsId            = INTEGER;

```

5.6 GKS enumerated types

GEASF	= (GVBundled,GVIndividual);
GEClip	= (GVClip,GVNoClip);
GEControl	= (GVConditionally,GVAlways);
GECoordSwitch	= (GVwc,GVndc);
GEDefer	= (GVasap,GVbnig,GVbnil,GVasti);
GEDet	= (GVUndetectable,GVDetectable);
GEDeviceUnits	= (GVMetres,GVOtherUnits);
GEDisplay	= (GVColour,GVMonochrome);
GEDynMod	= (GVirg,GVimm);
GEEcho	= (GVEcho,GVNoEcho);
GEEventClass	= (GVNone,GVLocator,GVStroke,GVValuator, GVChoice,GVPick,GVString);
GEEvents	= (GVNoMore,GVMore);
GEHighlight	= (GVNormal,GVHighlighted);
GEHorizontal	= (GVHnormal,GVHleft,GVHcentre,GVHright);
GEImplicitRegen	= (GVSuppressed,GVAllowed);
GEInputClass	= GVLocator..GVString;
GEInputStatus	= GVStatusOk..GVNoInput;
GEInterior	= (GVHollow,GVSolid,GVPattern,GVHatch);
GEInvPixel	= (GVAbsent,GVPresent);
GELevel	= (GVL0a,GVL0b,GVL0c, GVL1a,GVL1b,GVL1c, GVL2a,GVL2b,GVL2c);
GEMode	= (GVRequest,GVSample,GVEvent);
GENfan	= (GVNfanNo,GVNfanYes);
GEOpSt	= (GVgkcl,GVgkop,GVwsop,GVwsac,GVsgop);
GEPath	= (GVRight,GVLeft,GVUp,GVDown);
GEPrec	= (GVStringPrec,GVCharPrec,GVStrokePrec);
GEPrim	= (GVPolyline,GVPolymarker,GVText,GVFillArea);
GEPrimAttr	= (GVLineType,GVLineWidth,GVLineColr, GVMarkerType,GVMarkerSize,GVMarkerColr, GVFontPrec,GVExpan,GVSpacing,GVTextColr, GVFillInterior,GVFillStyleInd,GVFillColr);
GEPriority	= (GVHigher,GVLower);
GEReqStatus	= (GVStatusOk,GVNoInput,GVStatusNone);
GEReturn	= (GVSet,GVRealised);
GESegAttr	= (GVTran,GVToInvis,GVToVis, GVHighlight,GVPriority, GVAdd,GVRemove);
GESurface	= (GVEmpty,GVNotEmpty);
GEUpdRegen	= (GVPerform,GVPostpone);

GEVertical	= (GVVnormal, GVVtop, GVVcap, GVVhalf, GVVbase, GVVbottom);
GEVis	= (GVVvisible, GVInvisible);
GEWsCategory	= (GVOutput, GVInput, GVOutIn, GVWISS, GVMO, GVMI);
GEWsClass	= (GVVector, GVRaster, GVOther);
GEWsSt	= (GVInactive, GVActive);
GEWsTran	= (GVNotPending, GVPending);

5.7 Array types

GAASF	= array[GEPrimAttr] of GEASF;
GAColrArray	= array[GTMaxDX, GTMaxDY] of GTInt0;
GAColrInqArray	= array[GTMaxDX, GTMaxDY] of INTEGER;
GAConnId	= packed array[1..GCMaxFile] of CHAR;
GAEscapeInInt	= array[GTMaxEscapeIn] of INTEGER;
GAEscapeInReal	= array[GTMaxEscapeIn] of REAL;
GAEscapeInString	= array[GTMaxEscapeIn] of GRString;
GAEscapeOutInt	= array[GTMaxEscapeOut] of INTEGER;
GAEscapeOutReal	= array[GTMaxEscapeOut] of REAL;
CAEscapeOutString	= array[GTMaxEscapeOut] of GRString;
GAFontPrec	= array[GTInqSize] of GRFontPrec;
GAGDP	= array[GTInqSize] of GTGDPI;
GAGDPInt	= array[GTMaxGDP] of INTEGER;
GAGDPReal	= array[GTMaxGDP] of REAL;
GAGDPString	= array[GTMaxGDP] of GRString;
GAInputClass	= array[GEInputClass] of INTEGER;
GAInt	= array[GTInqSize] of INTEGER;
GAItemInt	= array[GTMaxItem] of INTEGER;
GAItemReal	= array[GTMaxItem] of REAL;
GAItemString	= array[GTMaxItem] of GRString;
GAMatrix	= array[1..2, 1..3] of REAL;
GAPointArray	= array[GTMaxPoint1] of GRPoint;
GAPrim	= array[GEPrim] of INTEGER;
GAPrimRep	= array[GEPrim] of GRPrimRep;
GAProcName	= packed array[1..GCMaxName] of CHAR;
GASeg	= array[GTInqSize] of GTSeg;
GASegMod	= array[GESegAttr] of GEDynMod;
GAStrng	= packed array[GTMaxString] of CHAR;
GATextExtent	= array[1..4] of GRPoint;
GAWSId	= array[GTInqSize] of GTWSId;
GAWSType	= array[GTInqSize] of GTWSType;

5.8 Set types

GSInterior	= SET of GEInterior;
GSPrim	= SET of GEPrim;

5.9 Record types

```

GRAlign      = record
                Horizontal      : GEHorizontal;
                Vertical        : GEVertical;
            end;

GRBound      = record
                LeftBound,
                RightBound,
                LowerBound,
                UpperBound      : REAL;
            end;

GRChoice     = record
                ChoiceStatus    : GEInputStatus;
                ChoiceNum       : GTInt1;
            end;

GRColr      = record
                Red,
                Green,
                Blue            : REAL;
            end;

GRDefChoiceData = record
                MaxChoice      : GTInt1;
                NumPrompt      : GTInt1;
                PromptList     : GAInt;
                Area           : GRBound;
                Data           : GRChoiceData;
            end;

GRDeferUpd   = record
                Defer          : GEDefer;
                ImplicitRegen   : GEImplicitRegen;
                Surface         : GESurface;
                Nfan           : GENfan;
            end;

GRDefInput   = record
                NumPrompt      : GTInt1;
                PromptList     : GAInt;
                Area           : GRBound;
                Data           : GRInputData;
                case Class     : GEInputClass of
                    GVLocator   : (InitialPosition : GRPoint);
                    GVStroke    : (MaxStroke       : GTInt1);
                    GVValuator   : (InitialValue    : REAL);
                    GVChoice     : (MaxChoice       : GTInt1);
                    GVPick       : ();
                    GVString     : (MaxString       : GTInt1);
                end;
            end;

GRDefLocatorData = record
                InitialPosition : GRPoint;
                NumPrompt      : GTInt1;
                PromptList     : GAInt;
                Area           : GRBound;
                Data           : GRLocatorData;
            end;

```

```

GRDefPickData      = record
    NumPrompt      : GTInt1;
    PromptList     : GInt;
    Area           : GRBound;
    Data           : GRPickData
end;

GRDefStringData    = record
    MaxSize        : GTInt1;
    NumPrompt      : GTInt1;
    PromptList     : GInt;
    Area           : GRBound;
    StringBufSize,
    InitialPosition : GTInt1;
    Data           : GRStringData
end;

GRDefStrokeData    = record
    MaxSize        : GTInt1;
    NumPrompt      : GTInt1;
    PromptList     : GInt;
    Area           : GRBound;
    StrokeBufSize  : GTInt1;
    Data           : GRStrokeData
end;

GRDefValuatorData  = record
    InitialValue   : REAL;
    NumPrompt      : GTInt1;
    PromptList     : GInt;
    Area           : GRBound;
    LowValue,
    HighValue      : REAL;
    Data           : GRValuatorData
end;

GRDisplaySize      = record
    DeviceUnits    : GEDeviceUnits;
    DcSize         : GRVector;
    RasterSize     : GRIntVector
end;

GRDynModWsAttr     = record
    LineBundleMod,
    MarkerBundleMod,
    TextBundleMod,
    FillBundleMod,
    PatternMod,
    ColrMod,
    WsTranMod      : GEDynMod
end;

GRFillFacil        = record
    NumTypes       : GTInt1;
    Styles         : GSInterior;
    NumHatch       : GTInt0;
    Hatches        : GInt;
    NumPredInd     : GTInt0
end;

GRFillRep          = record
    Interior       : GEInterior;
    StyleInd       : INTEGER;
    FColr          : GTInt0
end;

```

```

GRFontPrec      = record
    Font          : INTEGER;
    Prec          : GEPrec
end;

GRInput         = record
    case InputClass : GEInputClass of
        GVLocator   : (NormTranLocator : GTInt0;
                        Position         : GRPoint);
        GVStroke    : (NormTranStroke  : GTInt0;
                        Num              : GTInt0;
                        Points           : GAPointArray);
        GVValuator  : (Value            : REAL);
        GVChoice    : (ChoiceStatus    : GEInputStatus;
                        ChoiceNum       : GTInt1);
        GVPick      : (PickStatus       : GEInputStatus;
                        Segment         : GTSeg;
                        PickId          : GTPickId);
        GVString    : (StringLength    : GTInt0;
                        CharString      : GAStrng)
    end;

GRInputData     = record
    case InputClass : GEInputClass of
        GVLocator   : (LocatorData      : GRLocatorData);
        GVStroke    : (StrokeBufSize    : GTInt1;
                        StrokeData       : GRStrokeData);
        GVValuator  : (LowValue,        : REAL;
                        HighValue        : GRValuatorData);
        GVChoice    : (ChoiceData       : GRChoiceData);
        GVPick      : (PickData         : GRPickData);
        GVString    : (StringBufSize    : GTInt1;
                        InitialPosition  : GTInt1;
                        StringData       : GRStringData)
    end;

GRIntVector     = record
    xValue,
    yValue       : INTEGER
end;

GRLineFacil     = record
    NumTypes     : GTInt1;
    Lines        : GAInt;
    NumWidths    : GTInt0;
    NominalWidth,
    MinWidth,
    MaxWidth     : REAL;
    NumPredInd   : GTInt0
end;

GRLineRep       = record
    LType        : INTEGER;
    Width        : REAL;
    LColr        : GTInt0
end;

GRLocator       = record
    NormTranLocator : GTInt0;
    Position        : GRPoint
end;

```

```

GRMarkerFacil      = record
    NumTypes          : GTInt1;
    Markers            : GAIInt;
    NumSizes          : GTInt0;
    NominalSize,
    MinSize,
    MaxSize           : REAL;
    NumPredInd        : GTInt0
end;

GRMarkerRep        = record
    MType             : INTEGER;
    Size              : REAL;
    MColr             : GTInt0
end;

GRPICK             = record
    PickStatus        : GEInputStatus;
    Segment           : GTSeg;
    PickId            : GTPickId
end;

GRPoint            = record
    x,
    y                 : REAL
end;

GRPrimAttr         = record
    PolylineInd       : GTInt1;
    PolymarkerInd     : GTInt1;
    TextInd           : GTInt1;
    Text              : GRText;
    FillInd           : GTInt1;
    PatternWidth,
    PatternHeight     : GRVector;
    PatternRefPoint   : GRPoint
end;

GRPrimFacil        = record
    NumTypes          : GTInt1;
    NumPredInd        : GTInt0;
    case Prim         : GEPrim of
        GVPolyline    : (Lines           : GAIInt;
                        NumWidths        : GTInt0;
                        NominalWidth,
                        MinWidth,
                        MaxWidth         : REAL);
        GVPolymarker  : (Markers         : GAIInt;
                        NumSizes         : GTInt0;
                        NominalSize,
                        MinSize,
                        MaxSize          : REAL);
        GVText        : (FontPrecs       : GAFontPrec;
                        NumHeights       : GTInt0;
                        MinHeight,
                        MaxHeight        : REAL;
                        NumExpan         : GTInt0;
                        MinExpan,
                        MaxExpan         : REAL);
        GVFillArea    : (Styles          : GSInterior;
                        NumHatch         : GTInt0;
                        Hatches          : GAIInt)
    end;
end;

```

```

GRPrimRep      = record
    case Prim
        GVPolyline      : GEPrim of
            LType        : INTEGER;
            Width        : REAL;
            LColr        : GTInt0;
            GVPolymarker : (MType
                Size      : REAL;
                MColr     : GTInt0;
            GVText      : (FontPrec
                Expan,    : GRFontPrec;
                Spacing   : REAL;
                TColr     : GTInt0;
            GVFillArea  : (Interior
                StyleInd  : GEInterior;
                StyleInd  : INTEGER;
                FColr     : GTInt0)
        end;
GRSegAttr      = record
    SegTran      : GAMatrix;
    Vis          : GEVis;
    Highlight     : GEHighlight;
    Priority      : REAL;
    Det          : GEDet
end;
GRString       = record
    StringLength : GTInt0;
    CharString   : GASString
end;
GRStroke       = record
    NormTranStroke : GTInt0;
    Num            : GTInt0;
    Points         : GAPointArray
end;
GRText         = record
    Height        : REAL;
    UpVector      : GRVector;
    Width         : REAL;
    BaseVector    : GRVector;
    Path          : GEPATH;
    Align         : GRAlign
end;
GRTextFacil    = record
    NumTypes      : GTInt1;
    FontPrecs     : GAFontPrec;
    NumHeights    : GTInt0;
    MinHeight,
    MaxHeight     : REAL;
    NumExpan      : GTInt0;
    MinExpan,
    MaxExpan      : REAL;
    NumPredInd    : GTInt0
end;
GRTextRep      = record
    FontPrec      : GRFontPrec;
    Expan,
    Spacing       : REAL;
    TColr         : GTInt0
end;

```