INTERNATIONAL STANDARD

ISO 8651-2

First edition 1988-02-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ

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

Part 2 :
Pascal

Systèmes de traitement de l'information — Infographie — Système graphique de base (GKS) — Interface langage

Partie 2 : Pascal

Reference number
ISO 8651-2 : 1988 (E

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work.

Draft International Standards adopted by the technical committees are circulated to the member bodies for approval before their acceptance as International Standards by the ISO Council. They are approved in accordance with ISO procedures requiring at least 75 % approval by the member bodies voting.

International Standard ISO 8651-2 was prepared by Technical Committee ISO/TC 97, *Information processing systems*.

Users should note that all International Standards undergo revision from time to time and that any reference made herein to any other International Standard implies its latest edition, unless otherwise stated.

Contents

Page

This page intentionally left blank

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

Part 2 : Pascal

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 tables 2, 3, and 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
INDICES	Ind
INDIVIDUAL	Indiv
INITIALISE	Init
INPUT	NULL
INQUIRE	Inq
INTERIOR	Int

Table 1 - Abbreviations ordered alphabetically

GKS word	Abbreviation
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

The Pascal language binding of GKS

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

Table 2 - Names ordered by Pascal name

GKS Function Name	Level	Pascal Name
INQUIRE ASPECT SOURCE FLAGS	L0a	GInqASF
INQUIRE SET OF ACTIVE WORKSTATIONS	L1a	GInqActiveWs
INQUIRE SET OF ASSOCIATED WORKSTATIONS	L1a	GInqAssocWs
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	GInqDefInputDeviceData(Choice
INQUIRE DEFAULT LOCATOR DEVICE DATA	L0b	GInqDefInputDeviceData(Locator
INQUIRE DEFAULT PICK DEVICE DATA	L1b	GInqDefInputDeviceData(Pick
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 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 - 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
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

Table 2 - Names ordered by Pascal name

GKS Function Name	Level	Pascal Name
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
REQUEST STROKE	L0b	GReqStroke
REQUEST VALUATOR	L0b	GReqValuator
SAMPLE CHOICE	L0c	GSampleChoice

Table 2 - Names ordered by Pascal name

GKS Function Name	Level	Pascal Name
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 - 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
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

Table 3 - Names ordered by GKS function name

GKS Function Name	Level	Pascal Name
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
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

Table 3 - Names ordered by GKS function name

GKS Function Name	Level	Pascal Name
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
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

Table 3 - Names ordered by GKS function name

GKS Function Name	Level	Pascal Name
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
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

Table 3 - Names ordered by GKS function name

GKS Function Name	Level	Pascal Name
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
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

Table 3 - Names ordered by GKS function name

GKS Function Name	Level	Pascal Name
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
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
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

Table 4 - Names ordered by level

GKS Function Name	Level	Pascal Name
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	GInqPatternRefPoint
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
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

Table 4 - Names ordered by level

GKS Function Name	Level	Pascal Name
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
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

Table 4 - Names ordered by level

GKS Function Name	Level	Pascal Name
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
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

Table 4 - Names ordered by level

GKS Function Name	Level	Pascal Name
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
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

Table 4 - Names ordered by level

GKS Function Name	Level	Pascal Name
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.

a) The GKS types integer and real are mapped to Pascal integer and real. Where GKS specifies a subrange of integer, that subrange is used.

b) 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 (GAString) 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 (GAString).

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, GTErrorFileName. This type and the constant GCDefErrorLog shall be defined so that GCDefErrorLog is compatible with GTErrorFileName.

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, GCDefErrorLog 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 : GEReqStatus;
     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 : GEReqStatus;
     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 : GEReqStatus;
     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 Registration [1)]

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.

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.

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.

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, 3 or 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.

GCDefErrorLog	= 1;	{Default error log identifier}
GCMaxEscapeIn	= 10;	{Maximum number of input data record items for generalized ESCAPE}
GCMaxEscapeOut	= 10;	{Maximum number of output data record items for generalized ESCAPE}
GCMaxGDP	= 10;	{Maximum number of data record items for generalized GDP}
GCMaxDX	= 100;	{Maximum size of two dimensional arrays of colour}
GCMaxDY	= 100;	{indices used in fill area and pattern representation}
GCMaxFile	= 12;	{Maximum length of strings representing file names}
GCMaxInq	= 10;	{Maximum number of elements returned by an inquiry}
GCMaxItem	= 10;	{Maximum number of data record items for generalized WRITE ITEM}
GCMaxMemory	= 32767;	{Maximum number of memory units allowed}
GCMaxName	= 32;	{Maximum length of strings representing procedure names}
GCMaxPoint	= 128;	{Maximum number of points in a point array}
GCMaxString	= 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}
GTErrorFileName	= 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;
GTGDPId           = 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               = (GVVisible,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;
GAEscapeOutString   = array[GTMaxEscapeOut] of GRString;

GAFontPrec          = array[GTInqSize] of GRFontPrec;

GAGDP               = array[GTInqSize] of GTGDPId;
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;
```

Array types

```
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;
GAString            = 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;

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

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

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

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

GRDefValuatorData   =  record
                          InitialValue         :  REAL;
                          NumPrompt            :  GTInt1;
                          PromptList           :  GAInt;
                          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              :  GAInt;
                          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          : GAString)
                  end;

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

GRIntVector     = record
                    xValue,
                    yValue              : INTEGER
                  end;
```

Record types

```
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            :  GAInt;
                        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              :  GAInt;
                                              NumWidths          :  GTInt0;
                                              NominalWidth,
                                              MinWidth,
                                              MaxWidth           :  REAL);
                          GVPolymarker    :  (Markers            :  GAInt;
                                              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            :  GAInt)
                      end;
```

Record types **Pascal GKS data structures**

```
GRPrimRep          = record
                        case  Prim        :  GEPrim of
                            GVPolyline    :  (LType          :  INTEGER;
                                              Width          :  REAL;
                                              LColr          :  GTInt0);
                            GVPolymarker  :  (MType          :  INTEGER;
                                              Size           :  REAL;
                                              MColr          :  GTInt0);
                            GVText        :  (FontPrec       :  GRFontPrec;
                                              Expan,
                                              Spacing        :  REAL;
                                              TColr          :  GTInt0);
                            GVFillArea    :  (Interior       :  GEInterior;
                                              StyleInd       :  INTEGER;
                                              FColr          :  GTInt0)
                     end;

GRSegAttr          = record
                        SegTran           :  GAMatrix;
                        Vis               :  GEVis;
                        Highlight         :  GEHighlight;
                        Priority          :  REAL;
                        Det               :  GEDet
                     end;

GRString           = record
                        StringLength      :  GTInt0;
                        CharString        :  GAString
                     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;

GRVector           = record
                        xValue,
                        yValue            :  REAL
                     end;

GRWsMaxNum         = record
                        MaxOpenWs         :  GTInt1;
                        MaxActiveWs       :  GTInt1;
                        MaxWsSeg          :  GTInt1
                     end;
```

6 GKS functions

6.1 Notational conventions

The GKS abstract function names are given followed by the corresponding Pascal procedure and associated parameters. The GKS Level is shown for each function.

6.2 Control functions

OPEN GKS **L0a**

```
procedure GOpenGKS(
            ErrorFile          : GTErrorFileName;
            MemoryUnits        : GTMemory
            );
```

CLOSE GKS **L0a**

```
procedure GCloseGKS;
```

OPEN WORKSTATION **L0a**

```
procedure GOpenWs(
            WsId               : GTWsId;
            ConnId             : GAConnId;
            WsType             : GTWsType
            );
```

CLOSE WORKSTATION **L0a**

```
procedure GCloseWs(
            WsId               : GTWsId
            );
```

ACTIVATE WORKSTATION **L0a**

```
procedure GActivateWs(
            WsId               : GTWsId
            );
```

DEACTIVATE WORKSTATION **L0a**

```
procedure GDeactivateWs(
            WsId               : GTWsId
            );
```

CLEAR WORKSTATION L0a

```
procedure GClearWs(
            WsId                 : GTWsId;
            ControlFlag          : GEControl
            );
```

REDRAW ALL SEGMENTS ON WORKSTATION L1a

```
procedure GRedrawSegWs(
            WsId                 : GTWsId
            );
```

UPDATE WORKSTATION L0a

```
procedure GUpdWs(
            WsId                 : GTWsId;
            RegenFlag            : GEUpdRegen
            );
```

SET DEFERRAL STATE L1a

```
procedure GSetDeferSt(
            WsId                 : GTWsId;
            DeferMode            : GEDefer;
            RegenMode            : GEImplicitRegen
            );
```

MESSAGE **L1a**

Pascal level 1

```
procedure GMessage(
          WsId                 : GTWsId;
          StringLength         : GTInt1;
          Message              : packed array[min..max : GTInt1] of CHAR
          );
```

The parameter StringLength shall be $\leq$ max-min+1.

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

Pascal level 0

```
procedure GMessage(
          WsId                 : GTWsId;
          StringLength         : GTMaxString;
          Message              : GAString
          );
```

ESCAPE

L0a

```
procedure GEscape(
        EscapeId                : GTEscapeId;
    VAR InputDataRec            : GREscapeDataIn;
    VAR OutputDataRec           : GREscapeDataOut
        );
```

Although the input data record is a variable parameter, the implementation shall not change the value of the data record.

Escape identifiers and parameters are reserved for registration in the ISO International Register of Graphical Items which is maintained by the Registration Authority.

The following general form of ESCAPE is defined to allow an application to use the ESCAPE function for escape identifiers which do not have data records defined in GREscapeDataIn and GREscapeDataOut. The sequence of data in each array is defined in the Register of Graphical Items and, for each ESCAPE identifier in the Register, uses the method defined for the FORTRAN binding.

Pascal level 1

```
procedure GEscapeGeneralized(
        EscapeId                : GTEscapeId;
        NumIntIn                : GTInt0;
    VAR IntDataIn               : array[min1..max1 : INTEGER] of INTEGER;
        NumRealIn               : GTInt0;
    VAR RealDataIn              : array[min2..max2 : INTEGER] of REAL;
        NumStringIn             : GTInt0;
    VAR StringDataIn            : array[min3..max3 : INTEGER] of GRString
    VAR NumIntOut               : GTInt0;
    VAR IntDataOut              : array[min4..max4 : INTEGER] of INTEGER;
    VAR NumRealOut              : GTInt0;
    VAR RealDataOut             : array[min5..max5 : INTEGER] of REAL;
    VAR NumStringOut            : GTInt0;
    VAR StringDataOut           : array[min6..max6 : INTEGER] of GRString
        );
```

The value of the parameter NumIntIn ($\leq$ max1-min1+1) shall specify the number of integers in the array IntDataIn. The value of the parameter NumRealIn ($\leq$ max2-min2+1) shall specify the number of reals in the array RealDataIn. The value of the parameter NumStringIn ($\leq$ max3-min3+1) shall specify the number of strings in the array StringDataIn. Although the arrays are variable parameters, the implementation shall not change the value of the arrays.

Errors:

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

Pascal level 0

```
procedure GEscapeGeneralized(
         EscapeId            : GTEscapeId;
         NumIntIn            : GTInt0;
     VAR IntDataIn           : GAEscapeInInt;
         NumRealIn           : GTInt0;
     VAR RealDataIn          : GAEscapeInReal;
         NumStringIn         : GTInt0;
     VAR StringDataIn        : GAEscapeInString
     VAR NumIntOut           : GTInt0;
     VAR IntDataOut          : GAEscapeOutInt;
     VAR NumRealOut          : GTInt0;
     VAR RealDataOut         : GAEscapeOutReal;
     VAR NumStringOut        : GTInt0;
     VAR StringDataOut       : GAEscapeOutString
         );
```

The value of the parameter NumIntIn shall specify the number of integers in the array IntDataIn. The value of the parameter NumRealIn shall specify the number of reals in the array RealDataIn. The value of the parameter NumStringIn shall specify the number of strings in the array StringDataIn. Although the arrays are variable parameters, the implementation shall not change the value of the arrays.

6.3 Output functions

POLYLINE **L0a**

Pascal level 1

```
procedure GPolyline(
         NumPoints           : GTInt2;
     VAR Points              : array[min..max : INTEGER] of GRPoint
         );
```

The parameter NumPoints $\leq$ max-min+1 is the number of points in the array. Although the array is a variable parameter, the implementation shall not change the value of the array.

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

Pascal level 0

```
procedure GPolyline(
         NumPoints           : GTMaxPoint2;
     VAR Points              : GAPointArray
         );
```

The value of the parameter NumPoints shall specify the number of points in the array. Although the array is a variable parameter, the implementation shall not change the value of the array.

POLYMARKER **L0a**

Pascal level 1

```
procedure GPolymarker(
         NumPoints            : GTInt1;
   VAR Points                 : array[min..max : INTEGER] of GRPoint
         );
```

The parameter NumPoints $\leq$ max-min+1 is the number of points in the array. Although the array is a variable parameter, the implementation shall not change the value of the array.

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

Pascal level 0

```
procedure GPolymarker(
         NumPoints            : GTMaxPoint1;
   VAR Points                 : GAPointArray
         );
```

The value of the parameter NumPoints shall specify the number of points in the array. Although the array is a variable parameter, the implementation shall not change the value of the array.

TEXT **L0a**

Pascal level 1

```
procedure GText(
         TextPosition         : GRPoint;
         StringLength         : GTInt1;
         CharString           : packed array[min..max : GTInt1] of CHAR
         );
```

The parameter StringLength shall be $\leq$ max-min+1.

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

Pascal level 0

```
procedure GText(
         TextPosition         : GRPoint;
         StringLength         : GTMaxString;
         CharString           : GAString
         );
```

FILL AREA **L0a**

Pascal level 1

```
procedure GFill(
          NumPoints             : GTInt3;
    VAR Points                  : array[min..max : INTEGER] of GRPoint
          );
```

The parameter NumPoints ≤ max-min+1 is the number of points in the array. Although the array is a variable parameter, the implementation shall not change the value of the array.

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

Pascal level 0

```
procedure GFill(
          NumPoints             : GTMaxPoint3;
    VAR Points                  : GAPointArray
          );
```

The value of the parameter NumPoints shall specify the number of points in the array. Although the array is a variable parameter, the implementation shall not change the value of the array.

CELL ARRAY L0a

Pascal level 1

```
procedure GCellArray(
        PointP,
        PointQ                  : GRPoint;
        Dx,
        Dy                      : GTInt1;
    VAR ColrInds                : array[min1..max1 : INTEGER; min2..max2 : INTEGER] of GTInt0
        );
```

The cell array bounds are min1 to min1+Dx-1 ($\leq$max1) and min2 to min2+Dy-1 ($\leq$max2). Although the array is a variable parameter, the implementation shall not change the value of the array.

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

Pascal level 0

```
procedure GCellArray(
        PointP,
        PointQ                  : GRPoint;
        Dx                      : GTMaxDX;
        Dy                      : GTMaxDY;
    VAR ColrInds                : GAColrArray
        );
```

Although the array is a variable parameter, the implementation shall not change the value of the array.

GENERALIZED DRAWING PRIMITIVE (GDP) **L0a**

Pascal level 1

```
procedure GGDP(
        NumPoints              : GTInt0;
   VAR Points                  : array[min..max : INTEGER] of GRPoint;
        GDPId                  : GTGDPId;
   VAR DataRec                 : GRGDPData
        );
```

The value of the parameter NumPoints (≤ max-min+1) shall specify the number of points in the array. Although the array and data record are variable parameters, the implementation shall not change the value of the array or data record.

Errors:

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

Pascal level 0

```
procedure GGDP(
        NumPoints              : GTMaxPoint0;
   VAR Points                  : GAPointArray;
        GDPId                  : GTGDPId;
   VAR DataRec                 : GRGDPData
        );
```

The value of the parameter NumPoints shall specify the number of points in the array Points. Although the array and data record are variable parameters, the implementation shall not change the value of the array or data record.

GDP identifiers and parameters are reserved for registration in the ISO International Register of Graphical Items which is maintained by the Registration Authority.

The following general form of GDP is defined to allow an application to use the GDP function for GDP identifiers which do not have data records defined in GRGDPData. The sequence of data in each array is defined in the Register of Graphical Items and, for each GDP identifier in the Register, uses the method defined for the FORTRAN binding.

Pascal level 1

```
procedure GGDPGeneralized(
          NumPoints         : GTInt0;
    VAR Points              : array[min1..max1 : INTEGER] of GRPoint;
          GDPId             : GTGDPId;
          NumInt            : GTInt0;
    VAR IntData             : array[min2..max2 : INTEGER] of INTEGER;
          NumReal           : GTInt0;
    VAR RealData            : array[min3..max3 : INTEGER] of REAL;
          NumString         : GTInt0;
    VAR StringData          : array[min4..max4 : INTEGER] of GRString
          );
```

The value of the parameter NumPoints ($\leq$ max1-min1+1) shall specify the number of points in the array Points. The value of the parameter NumInt ($\leq$ max2-min2+1) shall specify the number of integers in the array IntData. The value of the parameter NumReal ($\leq$ max3-min3+1) shall specify the number of reals in the array RealData. The value of the parameter NumString ($\leq$ max4-min4+1) shall specify the number of strings in the array StringData. Although the arrays are variable parameters, the implementation shall not change the value of the arrays.

Errors:

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

Pascal level 0

```
procedure GGDPGeneralized(
          NumPoints         : GTMaxPoint0;
    VAR Points              : GAPointArray;
          GDPId             : GTGDPId;
          NumInt            : GTInt0;
    VAR IntData             : GAGDPInt;
          NumReal           : GTInt0;
    VAR RealData            : GAGDPReal;
          NumString         : GTInt0;
    VAR StringData          : GAGDPString
          );
```

The value of the parameter NumPoints shall specify the number of points in the array Points. The value of the parameter NumInt shall specify the number of integers in the array IntData. The value of the parameter NumReal shall specify the number of reals in the array RealData. The value of the parameter NumString shall specify the number of strings in the array StringData. Although the arrays are variable parameters, the implementation shall not change the value of the arrays.

6.4 Output attributes

6.4.1 Workstation Independent primitive attributes

SET POLYLINE INDEX **L0a**
SET POLYMARKER INDEX **L0a**
SET TEXT INDEX **L0a**
SET FILL AREA INDEX **L0a**

```
procedure GSetPrimInd(
            Prim                : GEPrim;
            PrimInd             : GTInt1
            );
```

This procedure is a general

SET <Primitive> INDEX

By substituting the components of the type GEPrim, the GKS abstract functions are obtained.

SET POLYLINE INDEX **L0a**

```
procedure GSetPolylineInd(
            Ind                 : GTInt1
            );
```

This procedure is a specific form of

SET <Primitive> INDEX

SET LINETYPE **L0a**

```
procedure GSetLineType(
            LineType            : INTEGER
            );
```

SET LINEWIDTH SCALE FACTOR **L0a**

```
procedure GSetLineWidthScale(
            LineWidthScale      : REAL
            );
```

SET POLYLINE COLOUR INDEX **L0a**

```
procedure GSetLineColrInd(
            LineColrInd         : GTInt0
            );
```

SET POLYMARKER INDEX L0a

```
procedure GSetPolymarkerInd(
          Ind                  : GTInt1
          );
```

This procedure is a specific form of

SET <Primitive> INDEX

SET MARKER TYPE L0a

```
procedure GSetMarkerType(
          MarkerType           : INTEGER
          );
```

SET MARKER SIZE SCALE FACTOR L0a

```
procedure GSetMarkerSizeScale(
          MarkerSizeScale      : REAL
          );
```

SET POLYMARKER COLOUR INDEX L0a

```
procedure GSetMarkerColrInd(
          MarkerColrInd        : GTInt0
          );
```

SET TEXT INDEX L0a

```
procedure GSetTextInd(
          Ind                  : GTInt1
          );
```

This procedure is a specific form of

SET <Primitive> INDEX

SET TEXT FONT AND PRECISION L0a

```
procedure GSetTextFontPrec(
          TextFontPrec         : GRFontPrec
          );
```

SET CHARACTER EXPANSION FACTOR **L0a**

```
procedure GSetCharExpan(
          CharExpan              : REAL
          );
```

SET CHARACTER SPACING **L0a**

```
procedure GSetCharSpacing(
          CharSpacing            : REAL
          );
```

SET TEXT COLOUR INDEX **L0a**

```
procedure GSetTextColrInd(
          TextColrInd            : GTInt0
          );
```

SET CHARACTER HEIGHT **L0a**

```
procedure GSetCharHeight(
          CharHeight             : REAL
          );
```

SET CHARACTER UP VECTOR **L0a**

```
procedure GSetCharUpVector(
          CharUpVector           : GRVector
          );
```

SET TEXT PATH **L0a**

```
procedure GSetTextPath(
          TextPath               : GEPath
          );
```

SET TEXT ALIGNMENT **L0a**

```
procedure GSetTextAlign(
          TextAlign              : GRAlign
          );
```

SET FILL AREA INDEX L0a

```
procedure GSetFillInd(
          Ind                   : GTInt1
          );
```

This procedure is a specific form of

SET <Primitive> INDEX

SET FILL AREA INTERIOR STYLE L0a

```
procedure GSetFillIntStyle(
          FillIntStyle          : GEInterior
          );
```

SET FILL AREA STYLE INDEX L0a

```
procedure GSetFillStyleInd(
          FillStyleInd          : INTEGER
          );
```

SET FILL AREA COLOUR INDEX L0a

```
procedure GSetFillColrInd(
          FillColrInd           : GTInt0
          );
```

SET PATTERN SIZE L0a

```
procedure GSetPatternSize(
          PatternSize           : GRVector
          );
```

SET PATTERN REFERENCE POINT L0a

```
procedure GSetPatternRefPoint(
          RefPoint              : GRPoint
          );
```

SET ASPECT SOURCE FLAGS L0a

```
procedure GSetASF(
          ListASF               : GAASF
          );
```

SET PICK IDENTIFIER **L1b**

```
procedure GSetPickId(
          PickId                  : GTPickId
          );
```

6.4.2 Workstation attributes (Representations)

SET POLYLINE REPRESENTATION **L1a**
SET POLYMARKER REPRESENTATION **L1a**
SET TEXT REPRESENTATION **L1a**
SET FILL AREA REPRESENTATION **L1a**

```
procedure GSetPrimRep(
          Prim                    : GEPrim;
          WsId                    : GTWsId;
          PrimInd                 : GTInt1;
          PrimRep                 : GRPrimRep
          );
```

This procedure is a general

SET <Primitive> REPRESENTATION

By substituting the components of the type GEPrim, the GKS abstract functions are obtained.

SET POLYLINE REPRESENTATION **L1a**

```
procedure GSetPolylineRep(
          WsId                    : GTWsId;
          Ind                     : GTInt1;
          Rep                     : GRLineRep
          );
```

This procedure is a specific form of

SET <Primitive> REPRESENTATION

SET POLYMARKER REPRESENTATION **L1a**

```
procedure GSetPolymarkerRep(
          WsId                    : GTWsId;
          Ind                     : GTInt1;
          Rep                     : GRMarkerRep
          );
```

This procedure is a specific form of

SET <Primitive> REPRESENTATION

SET TEXT REPRESENTATION **L1a**

```
procedure GSetTextRep(
            WsId                 : GTWsId;
            Ind                  : GTInt1;
            Rep                  : GRTextRep
            );
```

This procedure is a specific form of

SET <Primitive> REPRESENTATION

SET FILL AREA REPRESENTATION **L1a**

```
procedure GSetFillRep(
            WsId                 : GTWsId;
            Ind                  : GTInt1;
            Rep                  : GRFillRep
            );
```

This procedure is a specific form of

SET <Primitive> REPRESENTATION

SET PATTERN REPRESENTATION **L1a**

Pascal level 1

```
procedure GSetPatternRep(
            WsId                    : GTWsId;
            Ind                     : GTInt1;
            Dx,
            Dy                      : GTInt1;
    VAR PatternArray                : array[min1..max1 : INTEGER; min2..max2 : INTEGER] of GTInt0
            );
```

The pattern array bounds are min1 to min1+Dx-1 (≤max1) and min2 to min2+Dy-1 (≤max2). Although the array is a variable parameter, the implementation shall not change the value of the array.

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

Pascal level 0

```
procedure GSetPatternRep(
            WsId                    : GTWsId;
            Ind                     : GTInt1;
            Dx                      : GTMaxDX;
            Dy                      : GTMaxDY;
    VAR PatternArray                : GAColrArray
            );
```

Although the array is a variable parameter, the implementation shall not change the value of the array.

SET COLOUR REPRESENTATION **L0a**

```
procedure GSetColrRep(
            WsId                    : GTWsId;
            ColrInd                 : GTInt0;
            Colr                    : GRColr
            );
```

6.5 Transformation functions

6.5.1 Normalization transformation

SET WINDOW **L0a**

```
procedure GSetWindow(
            TranNum                 : GTInt1;
            WindowLimits            : GRBound
            );
```

SET VIEWPORT L0a

```
procedure GSetViewport(
        TranNum              : GTInt1;
        ViewportLimits       : GRBound
        );
```

SET VIEWPORT INPUT PRIORITY L0b

```
procedure GSetViewportPriority(
        TranNum,
        RefTranNum           : GTInt0;
        RelativePriority     : GEPriority
        );
```

SELECT NORMALIZATION TRANSFORMATION L0a

```
procedure GSelectNormTran(
        NormTranNum          : GTInt0
        );
```

SET CLIPPING INDICATOR L0a

```
procedure GSetClip(
        Clip                 : GEClip
        );
```

6.5.2 Workstation transformation

SET WORKSTATION WINDOW L0a

```
procedure GSetWsWindow(
        WsId                 : GTWsId;
        WsWindowLimits       : GRBound
        );
```

SET WORKSTATION VIEWPORT L0a

```
procedure GSetWsViewport(
        WsId                 : GTWsId;
        WsViewportLimits     : GRBound
        );
```

6.6 Segment functions

6.6.1 Segment manipulation functions

CREATE SEGMENT **L1a**

```
procedure GCreateSeg(
            SegName              : GTSeg
            );
```

CLOSE SEGMENT **L1a**

```
procedure GCloseSeg;
```

RENAME SEGMENT **L1a**

```
procedure GRenameSeg(
            OldSegName,
            NewSegName           : GTSeg
            );
```

DELETE SEGMENT **L1a**

```
procedure GDelSeg(
            SegName              : GTSeg
            );
```

DELETE SEGMENT FROM WORKSTATION **L1a**

```
procedure GDelSegWs(
            WsId                 : GTWsId;
            SegName              : GTSeg
            );
```

ASSOCIATE SEGMENT WITH WORKSTATION **L2a**

```
procedure GAssocSegWs(
            WsId                 : GTWsId;
            SegName              : GTSeg
            );
```

COPY SEGMENT TO WORKSTATION L2a

```
procedure GCopySegWs(
          WsId                    : GTWsId;
          SegName                 : GTSeg
          );
```

INSERT SEGMENT L2a

```
procedure GInsertSeg(
          SegName                 : GTSeg;
          TranMatrix              : GAMatrix
          );
```

6.6.2 Segment attributes

SET SEGMENT TRANSFORMATION L1a

```
procedure GSetSegTran(
          SegName                 : GTSeg;
          TranMatrix              : GAMatrix
          );
```

SET VISIBILITY L1a

```
procedure GSetVis(
          SegName                 : GTSeg;
          Vis                     : GEVis
          );
```

SET HIGHLIGHTING L1a

```
procedure GSetHighlight(
          SegName                 : GTSeg;
          Highlight               : GEHighlight
          );
```

SET SEGMENT PRIORITY L1a

```
procedure GSetSegPriority(
          SegName                 : GTSeg;
          SegPriority             : REAL
          );
```

SET DETECTABILITY **L1b**

```
procedure GSetDet(
            SegName            : GTSeg;
            SegDet             : GEDet
            );
```

6.7 Input functions

6.7.1 Initialisation of input devices

INITIALISE LOCATOR	**L0b**
INITIALISE STROKE	**L0b**
INITIALISE VALUATOR	**L0b**
INITIALISE CHOICE	**L0b**
INITIALISE PICK	**L1b**
INITIALISE STRING	**L0b**

```
procedure GInitInput(
            InputClass         : GEInputClass;
            WsId               : GTWsId;
            InputDeviceNum     : GTInt1;
            InitialInput       : GRInput;
            PromptEchoType     : INTEGER;
            EchoArea           : GRBound;
        VAR InputDataRecord    : GRInputData
            );
```

This procedure is a general

INITIALISE <Input>

By substituting the components of the type GEInputClass, the GKS abstract functions are obtained. Although the data record is a variable parameter, the implementation shall not change the value of the data record.

Errors:

2103 Pick is not supported in this level of GKS

INITIALISE LOCATOR **L0b**

```
procedure GInitLocator(
            WsId                : GTWsId;
            LocDeviceNum        : GTInt1;
            InitialLocator      : GRLocator;
            PromptEcho          : INTEGER;
            EchoArea            : GRBound;
        VAR LocatorDataRec      : GRLocatorData
            );
```

This procedure is a specific form of

INITIALISE <Input>

Although the data record is a variable parameter, the implementation shall not change the value of the data record.

INITIALISE STROKE **L0b**

```
procedure GInitStroke(
            WsId                : GTWsId;
            StrokeDeviceNum     : GTInt1;
            InitialStroke       : GRStroke;
            PromptEcho          : INTEGER;
            EchoArea            : GRBound;
            StrokeBufSize       : GTInt1;
        VAR StrokeDataRec       : GRStrokeData
            );
```

This procedure is a specific form of

INITIALISE <Input>

Although the data record is a variable parameter, the implementation shall not change the value of the data record.

INITIALISE VALUATOR **L0b**

```
procedure GInitValuator(
        WsId                 : GTWsId;
        ValDeviceNum         : GTInt1;
        InitialValue         : REAL;
        PromptEcho           : INTEGER;
        EchoArea             : GRBound;
        LowValue,
        HighValue            : REAL;
    VAR ValDataRec           : GRValuatorData
        );
```

This procedure is a specific form of

INITIALISE <Input>

Although the data record is a variable parameter, the implementation shall not change the value of the data record.

INITIALISE CHOICE **L0b**

```
procedure GInitChoice(
        WsId                 : GTWsId;
        ChoiceDeviceNum      : GTInt1;
        InitialChoice        : GRChoice;
        PromptEcho           : INTEGER;
        EchoArea             : GRBound;
    VAR ChoiceDataRec        : GRChoiceData
        );
```

This procedure is a specific form of

INITIALISE <Input>

Although the data record is a variable parameter, the implementation shall not change the value of the data record.

INITIALISE PICK

L1b

```
procedure GInitPick(
          WsId                : GTWsId;
          PickDeviceNum       : GTInt1;
          InitialPick         : GRPick;
          PromptEcho          : INTEGER;
          EchoArea            : GRBound;
      VAR PickDataRec         : GRPickData
          );
```

This procedure is a specific form of

INITIALISE <Input>

Although the data record is a variable parameter, the implementation shall not change the value of the data record.

INITIALISE STRING

L0b

```
procedure GInitString(
          WsId                : GTWsId;
          StringDeviceNum     : GTInt1;
          InitialString       : GRString;
          PromptEcho          : INTEGER;
          EchoArea            : GRBound;
          StringBufSize       : GTInt1;
          InitialPosition     : GTInt1;
      VAR StringDataRec       : GRStringData
          );
```

This procedure is a specific form of

INITIALISE <Input>

Although the data record is a variable parameter, the implementation shall not change the value of the data record.

6.7.2 Setting the mode of input devices

SET LOCATOR MODE	**L0b**
SET STROKE MODE	**L0b**
SET VALUATOR MODE	**L0b**
SET CHOICE MODE	**L0b**
SET PICK MODE	**L1b**
SET STRING MODE	**L0b**

```
procedure GSetInputMode(
          InputClass          : GEInputClass;
          WsId                : GTWsId;
          InputDeviceNum      : GTInt1;
          OperatingMode       : GEMode;
          EchoSwitch          : GEEcho
          );
```

This procedure is a general

SET <Input> MODE

By substituting the components of the type GEInputClass, the GKS abstract functions are obtained.

Errors:
2103 Pick is not supported in this level of GKS

SET LOCATOR MODE **L0b**

```
procedure GSetLocatorMode(
          WsId                : GTWsId;
          LocDeviceNum        : GTInt1;
          OpMode              : GEMode;
          EchoSwitch          : GEEcho
          );
```

This procedure is a specific form of

SET <Input> MODE

SET STROKE MODE **L0b**

```
procedure GSetStrokeMode(
            WsId                : GTWsId;
            StrokeDeviceNum     : GTInt1;
            OpMode              : GEMode;
            EchoSwitch          : GEEcho
            );
```

This procedure is a specific form of

SET <Input> MODE

SET VALUATOR MODE **L0b**

```
procedure GSetValuatorMode(
            WsId                : GTWsId;
            ValDeviceNum        : GTInt1;
            OpMode              : GEMode;
            EchoSwitch          : GEEcho
            );
```

This procedure is a specific form of

SET <Input> MODE

SET CHOICE MODE **L0b**

```
procedure GSetChoiceMode(
            WsId                : GTWsId;
            ChoiceDeviceNum     : GTInt1;
            OpMode              : GEMode;
            EchoSwitch          : GEEcho
            );
```

This procedure is a specific form of

SET <Input> MODE

SET PICK MODE **L1b**

```
procedure GSetPickMode(
          WsId              : GTWsId;
          PickDeviceNum     : GTInt1;
          OpMode            : GEMode;
          EchoSwitch        : GEEcho
          );
```

This procedure is a specific form of

SET <Input> MODE

SET STRING MODE **L0b**

```
procedure GSetStringMode(
          WsId              : GTWsId;
          StringDeviceNum   : GTInt1;
          OpMode            : GEMode;
          EchoSwitch        : GEEcho
          );
```

This procedure is a specific form of

SET <Input> MODE

6.7.3 Request input functions

REQUEST LOCATOR	**L0b**
REQUEST STROKE	**L0b**
REQUEST VALUATOR	**L0b**
REQUEST CHOICE	**L0b**
REQUEST PICK	**L1b**
REQUEST STRING	**L0b**

```
procedure GReqInput(
          InputClass          : GEInputClass;
          WsId                : GTWsId;
          InputDeviceNum      : GTInt1;
    VAR Status                : GEReqStatus;
    VAR InputMeasure          : GRInput
          );
```

This procedure is a general

REQUEST <Input>

By substituting the components of the type GEInputClass, the GKS abstract functions are obtained.

In the case of REQUEST CHOICE and REQUEST PICK, the status is returned in the Status parameter and also in the returned record.

Errors:
2103 Pick is not supported in this level of GKS

REQUEST LOCATOR **L0b**

```
procedure GReqLocator(
          WsId                : GTWsId;
          LocDeviceNum        : GTInt1;
    VAR Status                : GEReqStatus;
    VAR LocatorMeasure        : GRLocator
          );
```

This procedure is a specific form of

REQUEST <Input>

REQUEST STROKE **L0b**

```
procedure GReqStroke(
            WsId                : GTWsId;
            StrokeDeviceNum     : GTInt1;
    VAR Status                  : GEReqStatus;
    VAR StrokeMeasure           : GRStroke
            );
```

This procedure is a specific form of

REQUEST <Input>

REQUEST VALUATOR **L0b**

```
procedure GReqValuator(
            WsId                : GTWsId;
            ValDeviceNum        : GTInt1;
    VAR Status                  : GEReqStatus;
    VAR ValueMeasure            : REAL
            );
```

This procedure is a specific form of

REQUEST <Input>

REQUEST CHOICE **L0b**

```
procedure GReqChoice(
            WsId                : GTWsId;
            ChoiceDeviceNum     : GTInt1;
    VAR Status                  : GEReqStatus;
    VAR ChoiceMeasure           : GRChoice
            );
```

This procedure is a specific form of

REQUEST <Input>

The status is returned in the Status parameter and also in the returned record.

REQUEST PICK **L1b**

```
procedure GReqPick(
            WsId                : GTWsId;
            PickDeviceNum       : GTInt1;
    VAR Status                  : GEReqStatus;
    VAR PickMeasure             : GRPick
            );
```

This procedure is a specific form of

REQUEST <Input>

The status is returned in the Status parameter and also in the returned record.

REQUEST STRING **L0b**

```
procedure GReqString(
            WsId                : GTWsId;
            StringDeviceNum     : GTInt1;
    VAR Status                  : GEReqStatus;
    VAR StringMeasure           : GRString
            );
```

This procedure is a specific form of

REQUEST <Input>

6.7.4 Sample input functions

SAMPLE LOCATOR	**L0c**
SAMPLE STROKE	**L0c**
SAMPLE VALUATOR	**L0c**
SAMPLE CHOICE	**L0c**
SAMPLE PICK	**L1c**
SAMPLE STRING	**L0c**

```
procedure GSampleInput(
            InputClass          : GEInputClass;
            WsId                : GTWsId;
            InputDeviceNum      : GTInt1;
    VAR InputMeasure            : GRInput
            );
```

This procedure is a general

SAMPLE <Input>

By substituting the components of the type GEInputClass, the GKS abstract functions are obtained.

Errors:
2103 Pick is not supported in this level of GKS

SAMPLE LOCATOR **L0c**

```
procedure GSampleLocator(
         WsId                 : GTWsId;
         LocDeviceNum         : GTInt1;
     VAR LocatorMeasure       : GRLocator
         );
```

This procedure is a specific form of

SAMPLE <Input>

SAMPLE STROKE **L0c**

```
procedure GSampleStroke(
         WsId                 : GTWsId;
         StrokeDeviceNum      : GTInt1;
     VAR StrokeMeasure        : GRStroke
         );
```

This procedure is a specific form of

SAMPLE <Input>

SAMPLE VALUATOR **L0c**

```
procedure GSampleValuator(
         WsId                 : GTWsId;
         ValDeviceNum         : GTInt1;
     VAR ValueMeasure         : REAL
         );
```

This procedure is a specific form of

SAMPLE <Input>

SAMPLE CHOICE **L0c**

```
procedure GSampleChoice(
         WsId                 : GTWsId;
         ChoiceDeviceNum      : GTInt1;
     VAR ChoiceMeasure        : GRChoice
         );
```

This procedure is a specific form of

SAMPLE <Input>

SAMPLE PICK **L1c**

```
procedure GSamplePick(
            WsId               : GTWsId;
            PickDeviceNum      : GTInt1;
    VAR PickMeasure            : GRPick
            );
```

This procedure is a specific form of

SAMPLE <Input>

SAMPLE STRING **L0c**

```
procedure GSampleString(
            WsId               : GTWsId;
            StringDeviceNum    : GTInt1;
    VAR StringMeasure          : GRString
            );
```

This procedure is a specific form of

SAMPLE <Input>

6.7.5 Event input functions

AWAIT EVENT **L0c**

```
procedure GAwaitEvent(
            Timeout            : REAL;
    VAR WsId                   : GTWsId;
    VAR Class                  : GEEventClass;
    VAR InputDeviceNum         : GTInt1
            );
```

FLUSH DEVICE EVENTS **L0c**

```
procedure GFlushDeviceEvents(
            WsId               : GTWsId;
            InputClass         : GEInputClass;
            InputDeviceNum     : GTInt1
            );
```

GET LOCATOR L0c
GET STROKE L0c
GET VALUATOR L0c
GET CHOICE L0c
GET PICK L1c
GET STRING L0c

```
procedure GGetInput(
            InputClass           : GEInputClass;
        VAR InputMeasure         : GRInput
            );
```

This procedure is a general

GET <Input>

By substituting the components of the type GEInputClass, the GKS abstract functions are obtained.

Errors:
2103 Pick is not supported in this level of GKS

GET LOCATOR L0c

```
procedure GGetLocator(
        VAR LocatorMeasure       : GRLocator
            );
```

This procedure is a specific form of

GET <Input>

GET STROKE L0c

```
procedure GGetStroke(
        VAR StrokeMeasure        : GRStroke
            );
```

This procedure is a specific form of

GET <Input>

GET VALUATOR L0c

```
procedure GGetValuator(
        VAR ValueMeasure         : REAL
            );
```

This procedure is a specific form of

GET <Input>

GET CHOICE L0c

```
procedure GGetChoice(
    VAR ChoiceMeasure          : GRChoice
        );
```

This procedure is a specific form of

GET <Input>

GET PICK L1c

```
procedure GGetPick(
    VAR PickMeasure            : GRPick
        );
```

This procedure is a specific form of

GET <Input>

GET STRING L0c

```
procedure GGetString(
    VAR StringMeasure          : GRString
        );
```

This procedure is a specific form of

GET <Input>

6.8 Metafile functions

WRITE ITEM TO GKSM

L0a

```
procedure GWriteItem(
            WsId                : GTWsId;
            ItemType            : INTEGER;
            ItemDataRecLength   : GTInt0;
        VAR DataRec             : GRFileData
            );
```

Although the data record is a variable parameter, the implementation shall not change the value of the data record.

The following general form of WRITE ITEM is defined to allow an application to use the WRITE ITEM function for application defined item types which do not have data records defined in GRFileData.

Pascal level 1

```
procedure GWriteItemGeneralized(
            WsId                : GTWsId;
            ItemType            : INTEGER;
            NumIntIn            : GTInt0;
        VAR IntDataIn           : array[min1..max1 : INTEGER] of INTEGER;
            NumRealIn           : GTInt0;
        VAR RealDataIn          : array[min2..max2 : INTEGER] of REAL;
            NumStringIn         : GTInt0;
        VAR StringDataIn        : array[min3..max3 : INTEGER] of GRString
            );
```

The value of the parameter NumIntIn (≤ max1-min1+1) shall specify the number of integers in the array IntDataIn. The value of the parameter NumRealIn (≤ max2-min2+1) shall specify the number of reals in the array RealDataIn. The value of the parameter NumStringIn (≤ max3-min3+1) shall specify the number of strings in the array StringDataIn. Although the arrays are variable parameters, the implementation shall not change the value of the arrays.

Errors:

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

Pascal level 0

```
procedure GWriteItemGeneralized(
          WsId                : GTWsId;
          ItemType            : INTEGER;
          NumIntIn            : GTInt0;
      VAR IntDataIn           : GAItemInt;
          NumRealIn           : GTInt0;
      VAR RealDataIn          : GAItemReal;
          NumStringIn         : GTInt0;
      VAR StringDataIn        : GAItemString
          );
```

The value of the parameter NumIntIn shall specify the number of integers in the array IntDataIn. The value of the parameter NumRealIn shall specify the number of reals in the array RealDataIn. The value of the parameter NumStringIn shall specify the number of strings in the array StringDataIn. Although the arrays are variable parameters, the implementation shall not change the value of the arrays.

GET ITEM TYPE FROM GKSM **L0a**

```
procedure GGetItemType(
          WsId                : GTWsId;
      VAR ItemType            : GTInt0;
      VAR ItemDataRecLength   : GTInt0
          );
```

Further information about possible values of the item type is provided in Annex B.

READ ITEM FROM GKSM **L0a**

```
procedure GReadItem(
          WsId                : GTWsId;
          MaxDataRecLength    : GTInt0;
      VAR DataRec             : GRFileData
          );
```

INTERPRET ITEM **L0a**

```
procedure GInterpretItem(
          ItemType            : INTEGER;
          ItemDataRecLength   : GTInt0;
      VAR DataRec             : GRFileData
          );
```

Although the data record is a variable parameter, the implementation shall not change the value of the data record.

6.9 Inquiry functions

6.9.1 Convention

Where a GKS abstract inquiry function returns many values, these are represented in Pascal as a record, the fields of which deliver the values required.

6.9.2 Inquiry function for operating state value

INQUIRE OPERATING STATE VALUE **L0a**

```
procedure GInqOpSt(
    VAR OpSt                  : GEOpSt
        );
```

6.9.3 Inquiry functions for GKS description table

INQUIRE LEVEL OF GKS **L0a**

```
procedure GInqLevelGKS(
    VAR ErrorInd              : INTEGER;
    VAR LevelGKS              : GELevel
        );
```

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES L0a

Pascal level 1

```
procedure GInqListWsTypes(
          Start             : GTInt1;
          Size              : INTEGER;
    VAR Done                : Boolean;
    VAR ErrorInd            : INTEGER;
    VAR NumWsTypes          : GTInt1;
    VAR WsTypes             : array [min..max:INTEGER] of GTWsType
          );
```

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:
2100 *There is an incompatibility between the bounds of the array and the actual size parameters specified*
2102 *List element or set member not available*

Pascal level 0

```
procedure GInqListWsTypes(
          Start             : GTInt1;
          Size              : GTInqSize;
    VAR Done                : Boolean;
    VAR ErrorInd            : INTEGER;
    VAR NumWsTypes          : GTInt1;
    VAR WsTypes             : GAWsType
          );
```

The parameters Start, Size and Done are defined in 3.11.

Errors:
2102 *List element or set member not available*

INQUIRE WORKSTATION MAXIMUM NUMBERS L1a

```
procedure GInqWsMaxNum(
    VAR ErrorInd            : INTEGER;
    VAR MaxNum              : GRWsMaxNum
          );
```

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER L0a

```
procedure GInqMaxNormTranNum(
    VAR ErrorInd            : INTEGER;
    VAR MaxNormTran         : GTInt1
          );
```

6.9.4 Inquiry functions for GKS state list

INQUIRE SET OF OPEN WORKSTATIONS L0a

Pascal level 1

```
procedure GInqOpenWs(
          Start                : GTInt1;
          Size                 : INTEGER;
    VAR Done                   : Boolean;
    VAR ErrorInd               : INTEGER;
    VAR NumOpenWs              : GTInt0;
    VAR OpenWs                 : array [min..max:INTEGER] of GTWsId
          );
```

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:
2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqOpenWs(
          Start                : GTInt1;
          Size                 : GTInqSize;
    VAR Done                   : Boolean;
    VAR ErrorInd               : INTEGER;
    VAR NumOpenWs              : GTInt0;
    VAR OpenWs                 : GAWsId
          );
```

The parameters Start, Size and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE SET OF ACTIVE WORKSTATIONS **L1a**

Pascal level 1

```
procedure GInqActiveWs(
          Start             : GTInt1;
          Size              : INTEGER;
    VAR Done                : Boolean;
    VAR ErrorInd            : INTEGER;
    VAR NumActiveWs         : GTInt0;
    VAR ActiveWs            : array [min..max:INTEGER] of GTWsId
          );
```

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:
2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqActiveWs(
          Start             : GTInt1;
          Size              : GTInqSize;
    VAR Done                : Boolean;
    VAR ErrorInd            : INTEGER;
    VAR NumActiveWs         : GTInt0;
    VAR ActiveWs            : GAWsId
          );
```

The parameters Start, Size and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES **L0a**

```
procedure GInqCurPrimAttr(
    VAR ErrorInd            : INTEGER;
    VAR PrimAttr            : GRPrimAttr
          );
```

INQUIRE POLYLINE INDEX **L0a**

```
procedure GInqLineInd(
    VAR ErrorInd            : INTEGER;
    VAR PolyLineInd         : GTInt1
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE POLYMARKER INDEX **L0a**

```
procedure GInqMarkerInd(
    VAR ErrorInd            : INTEGER;
    VAR PolyMarkerInd       : GTInt1
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE TEXT INDEX **L0a**

```
procedure GInqTextInd(
    VAR ErrorInd            : INTEGER;
    VAR TextInd             : GTInt1
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE CHARACTER HEIGHT **L0a**

```
procedure GInqCharHeight(
    VAR ErrorInd            : INTEGER;
    VAR Height              : REAL
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE CHARACTER UP VECTOR L0a

```
procedure GInqCharUpVector(
    VAR ErrorInd                : INTEGER;
    VAR UpVector                : GRVector
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE CHARACTER WIDTH L0a

```
procedure GInqCharWidth(
    VAR ErrorInd                : INTEGER;
    VAR Width                   : REAL
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE CHARACTER BASE VECTOR L0a

```
procedure GInqCharBaseVector(
    VAR ErrorInd                : INTEGER;
    VAR BaseVector              : GRVector
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE TEXT PATH L0a

```
procedure GInqTextPath(
    VAR ErrorInd                : INTEGER;
    VAR Path                    : GEPath
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE TEXT ALIGNMENT L0a

```
procedure GInqTextAlign(
    VAR ErrorInd              : INTEGER;
    VAR Align                 : GRAlign
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE FILL AREA INDEX L0a

```
procedure GInqFillInd(
    VAR ErrorInd              : INTEGER;
    VAR FillInd               : GTInt1
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE PATTERN SIZE L0a

```
procedure GInqPatternSize(
    VAR ErrorInd              : INTEGER;
    VAR PatternWidth          : GRVector;
    VAR PatternHeight         : GRVector
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE PATTERN REFERENCE POINT L0a

```
procedure GInqPatternRefPoint(
    VAR ErrorInd              : INTEGER;
    VAR PatternRefPoint       : GRPoint
        );
```

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE CURRENT PICK IDENTIFIER **L1b**

```
procedure GInqCurPickId(
    VAR ErrorInd            : INTEGER;
    VAR PickId              : GTPickId
        );
```

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES **L0a**

```
procedure GInqCurIndivAttr(
    VAR ErrorInd            : INTEGER;
    VAR IndivAttr           : GAPrimRep;
    VAR ListASF             : GAASF
        );
```

This procedure delivers an array IndivAttr with elements IndivAttr[GVPolyline], IndivAttr[GVPolymarker], IndivAttr[GVText], IndivAttr[GVFill]. Each array element is a variant record of the type GRPrimRep with a tag field corresponding to the value of the array index. For example, the following would be true for the element IndivAttr[GVPolyline]:

```
Prim   = GVPolyline
Ltype  = the current line type
Width  = the current line width scale factor
LColr  = the current polyline colour Index
```

INQUIRE LINETYPE **L0a**

```
procedure GInqLineType(
    VAR ErrorInd            : INTEGER;
    VAR LType               : INTEGER
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE LINEWIDTH SCALE FACTOR **L0a**

```
procedure GInqLineWidthScale(
    VAR ErrorInd            : INTEGER;
    VAR Width               : REAL
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE POLYLINE COLOUR INDEX **L0a**

```
procedure GInqLineColrInd(
    VAR ErrorInd             : INTEGER;
    VAR LColr                : GTInt0
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE POLYMARKER TYPE **L0a**

```
procedure GInqMarkerType(
    VAR ErrorInd             : INTEGER;
    VAR MType                : INTEGER
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE POLYMARKER SIZE SCALE FACTOR **L0a**

```
procedure GInqMarkerSizeScale(
    VAR ErrorInd             : INTEGER;
    VAR Size                 : REAL
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE POLYMARKER COLOUR INDEX **L0a**

```
procedure GInqMarkerColrInd(
    VAR ErrorInd             : INTEGER;
    VAR MColr                : GTInt0
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE TEXT FONT AND PRECISION **L0a**

```
procedure GInqTextFontPrec(
    VAR ErrorInd              : INTEGER;
    VAR FontPrec              : GRFontPrec
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE CHARACTER EXPANSION FACTOR **L0a**

```
procedure GInqCharExpan(
    VAR ErrorInd              : INTEGER;
    VAR Expan                 : REAL
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE CHARACTER SPACING **L0a**

```
procedure GInqCharSpacing(
    VAR ErrorInd              : INTEGER;
    VAR Spacing               : REAL
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE TEXT COLOUR INDEX **L0a**

```
procedure GInqTextColrInd(
    VAR ErrorInd              : INTEGER;
    VAR TColr                 : GTInt0
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE FILL AREA INTERIOR STYLE **L0a**

```
procedure GInqFillIntStyle(
    VAR ErrorInd            : INTEGER;
    VAR Interior            : GEInterior
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE FILL AREA STYLE INDEX **L0a**

```
procedure GInqFillStyleInd(
    VAR ErrorInd            : INTEGER;
    VAR StyleInd            : INTEGER
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE FILL AREA COLOUR INDEX **L0a**

```
procedure GInqFillColrInd(
    VAR ErrorInd            : INTEGER;
    VAR FColr               : GTInt0
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE ASPECT SOURCE FLAGS **L0a**

```
procedure GInqASF(
    VAR ErrorInd            : INTEGER;
    VAR ListASF             : GAASF
        );
```

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER **L0a**

```
procedure GInqCurNormTranNum(
    VAR ErrorInd            : INTEGER;
    VAR NormTranNum         : GTInt0
        );
```

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS L0a

Pascal level 1

```
procedure GInqListNormTranNum(
            Start              : GTInt1;
            Size               : INTEGER;
    VAR Done                   : Boolean;
    VAR ErrorInd               : INTEGER;
    VAR NumNormTran            : INTEGER;
    VAR NormTran               : array [min..max:INTEGER] of INTEGER
            );
```

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:
2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqListNormTranNum(
            Start              : GTInt1;
            Size               : GTInqSize;
    VAR Done                   : Boolean;
    VAR ErrorInd               : INTEGER;
    VAR NumNormTran            : INTEGER;
    VAR NormTran               : GAInt
            );
```

The parameters Start, Size and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE NORMALIZATION TRANSFORMATION L0a

```
procedure GInqNormTran(
            NormTranNum        : GTInt0;
    VAR ErrorInd               : INTEGER;
    VAR WindowLimits,
            ViewportLimits     : GRBound
            );
```

INQUIRE CLIPPING L0a

```
procedure GInqClip(
    VAR ErrorInd              : INTEGER;
    VAR Indicator             : GEClip;
    VAR ClippingRectangle     : GRBound
        );
```

INQUIRE NAME OF OPEN SEGMENT L1a

```
procedure GInqOpenSeg(
    VAR ErrorInd              : INTEGER;
    VAR SegName               : GTSeg
        );
```

INQUIRE SET OF SEGMENT NAMES IN USE L1a

Pascal level 1

```
procedure GInqSegNames(
          Start              : GTInt1;
          Size               : INTEGER;
   VAR Done                  : Boolean;
   VAR ErrorInd              : INTEGER;
   VAR NumSegNames           : GTInt0;
   VAR SegNames              : array [min..max:INTEGER] of GTSeg
          );
```

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:
2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqSegNames(
          Start              : GTInt1;
          Size               : GTInqSize;
   VAR Done                  : Boolean;
   VAR ErrorInd              : INTEGER;
   VAR NumSegNames           : GTInt0;
   VAR SegNames              : GASeg
          );
```

The parameters Start, Size and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE MORE SIMULTANEOUS EVENTS L0c

```
procedure GInqMoreEvents(
   VAR ErrorInd              : INTEGER;
   VAR MoreEvents            : GEEvents
          );
```

6.9.5 Inquiry functions for workstation state list

INQUIRE WORKSTATION CONNECTION AND TYPE **L0a**

```
procedure GInqWsConnType(
          WsId                   : GTWsId;
      VAR ErrorInd               : INTEGER;
      VAR ConnId                 : GAConnId;
      VAR WsType                 : GTWsType
          );
```

INQUIRE WORKSTATION STATE **L0a**

```
procedure GInqWsSt(
          WsId                   : GTWsId;
      VAR ErrorInd               : INTEGER;
      VAR WsSt                   : GEWsSt
          );
```

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES **L0a**

```
procedure GInqWsDeferUpdSt(
          WsId                   : GTWsId;
      VAR ErrorInd               : INTEGER;
      VAR WsSts                  : GRDeferUpd
          );
```

INQUIRE LIST OF POLYLINE INDICES L1a
INQUIRE LIST OF POLYMARKER INDICES L1a
INQUIRE LIST OF TEXT INDICES L1a
INQUIRE LIST OF FILL AREA INDICES L1a

Pascal level 1

```
procedure GInqListPrimInd(
          Prim                 : GEPrim;
          WsId                 : GTWsId;
          Start                : GTInt1;
          Size                 : INTEGER;
   VAR Done                    : Boolean;
   VAR ErrorInd                : INTEGER;
   VAR NumPrimEntries          : GTInt1;
   VAR DefinedPrimInd          : array [min..max:INTEGER] of INTEGER
          );
```

This procedure is a general

INQUIRE LIST OF <Primitive> INDICES

By substituting the components of the type GEPrim, the GKS abstract functions are obtained. The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:
2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqListPrimInd(
          Prim                 : GEPrim;
          WsId                 : GTWsId;
          Start                : GTInt1;
          Size                 : GTInqSize;
   VAR Done                    : Boolean;
   VAR ErrorInd                : INTEGER;
   VAR NumPrimEntries          : GTInt1;
   VAR ListDefinedPrimInd      : GAInt
          );
```

This procedure is a general

INQUIRE LIST OF <Primitive> INDICES

By substituting the components of the type GEPrim, the GKS abstract functions are obtained. The parameters Start, Size and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE POLYLINE REPRESENTATION L1a
INQUIRE POLYMARKER REPRESENTATION L1a
INQUIRE TEXT REPRESENTATION L1a
INQUIRE FILL AREA REPRESENTATION L1a

```
procedure GInqPrimRep(
          Prim                : GEPrim;
          WsId                : GTWsId;
          PrimInd             : GTInt1;
          TypeReturn          : GEReturn;
      VAR ErrorInd            : INTEGER;
      VAR PrimRep             : GRPrimRep
          );
```

This procedure is a general

INQUIRE <Primitive> REPRESENTATION

By substituting the components of the type GEPrim, the GKS abstract functions are obtained.

INQUIRE LIST OF POLYLINE INDICES

L1a

Pascal level 1

```
procedure GInqListPolylineInd(
            WsId              : GTWsId;
            Start             : GTInt1;
            Size              : INTEGER;
    VAR Done                  : Boolean;
    VAR ErrorInd              : INTEGER;
    VAR NumEntries            : GTInt1;
    VAR DefinedEntries        : array [min..max:INTEGER] of INTEGER
            );
```

This procedure is a specific form of

INQUIRE LIST OF <Primitive> INDICES

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:

2100 *There is an incompatibility between the bounds of the array and the actual size parameters specified*
2102 *List element or set member not available*

Pascal level 0

```
procedure GInqListPolylineInd(
            WsId              : GTWsId;
            Start             : GTInt1;
            Size              : GTInqSize;
    VAR Done                  : Boolean;
    VAR ErrorInd              : INTEGER;
    VAR NumEntries            : GTInt1;
    VAR DefinedEntries        : GAInt
            );
```

This procedure is a specific form of

INQUIRE LIST OF <Primitive> INDICES

The parameters Start, Size and Done are defined in 3.11.

Errors:

2102 *List element or set member not available*

INQUIRE POLYLINE REPRESENTATION L1a

```
procedure GInqPolylineRep(
            WsId                  :  GTWsId;
            PolylineInd           :  GTInt1;
            TypeReturn            :  GEReturn;
        VAR ErrorInd              :  INTEGER;
        VAR LineRep               :  GRLineRep
            );
```

This procedure is a specific form of

INQUIRE <Primitive> REPRESENTATION

INQUIRE LIST OF POLYMARKER INDICES L1a

Pascal level 1

```
procedure GInqListPolymarkerInd(
            WsId                  : GTWsId;
            Start                 : GTInt1;
            Size                  : INTEGER;
        VAR Done                  : Boolean;
        VAR ErrorInd              : INTEGER;
        VAR NumEntries            : GTInt1;
        VAR DefinedEntries        : array [min..max:INTEGER] of INTEGER
            );
```

This procedure is a specific form of

INQUIRE LIST OF <Primitive> INDICES

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:
2100 *There is an incompatibility between the bounds of the array and the actual size parameters specified*
2102 *List element or set member not available*

Pascal level 0

```
procedure GInqListPolymarkerInd(
            WsId                  : GTWsId;
            Start                 : GTInt1;
            Size                  : GTInqSize;
        VAR Done                  : Boolean;
        VAR ErrorInd              : INTEGER;
        VAR NumEntries            : GTInt1;
        VAR DefinedEntries        : GAInt
            );
```

This procedure is a specific form of

INQUIRE LIST OF <Primitive> INDICES

The parameters Start, Size and Done are defined in 3.11.

Errors:
2102 *List element or set member not available*

INQUIRE POLYMARKER REPRESENTATION

L1a

```
procedure GInqPolymarkerRep(
            WsId                : GTWsId;
            PolymarkerInd       : GTInt1;
            TypeReturn          : GEReturn;
    VAR ErrorInd               : INTEGER;
    VAR MarkerRep              : GRMarkerRep
            );
```

This procedure is a specific form of

INQUIRE <Primitive> REPRESENTATION

INQUIRE LIST OF TEXT INDICES L1a

Pascal level 1

```
procedure GInqListTextInd(
          WsId              : GTWsId;
          Start             : GTInt1;
          Size              : INTEGER;
    VAR Done                : Boolean;
    VAR ErrorInd            : INTEGER;
    VAR NumEntries          : GTInt1;
    VAR DefinedEntries      : array [min..max:INTEGER] of INTEGER
          );
```

This procedure is a specific form of

INQUIRE LIST OF <Primitive> INDICES

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:

2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqListTextInd(
          WsId              : GTWsId;
          Start             : GTInt1;
          Size              : GTInqSize;
    VAR Done                : Boolean;
    VAR ErrorInd            : INTEGER;
    VAR NumEntries          : GTInt1;
    VAR DefinedEntries      : GAInt
          );
```

This procedure is a specific form of

INQUIRE LIST OF <Primitive> INDICES

The parameters Start, Size and Done are defined in 3.11.

Errors:

2102 List element or set member not available

INQUIRE TEXT REPRESENTATION **L1a**

```
procedure GInqTextRep(
          WsId                : GTWsId;
          TextInd             : GTInt1;
          TypeReturn          : GEReturn;
   VAR ErrorInd               : INTEGER;
   VAR TextRep                : GRTextRep
          );
```

This procedure is a specific form of

INQUIRE <Primitive> REPRESENTATION

INQUIRE TEXT EXTENT **L0a**

Pascal level 1

```
procedure GInqTextExtent(
          WsId                : GTWsId;
          TextPosition        : GRPoint;
          StringLength        : GTInt1;
          CharString          : packed array[min..max : GTInt1] of CHAR;
   VAR ErrorInd               : INTEGER;
   VAR ConcatPoint            : GRPoint;
   VAR TextExtent             : GATextExtent
          );
```

The parameter StringLength shall be ≤ max-min+1.

Errors:

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

Pascal level 0

```
procedure GInqTextExtent(
          WsId                : GTWsId;
          TextPosition        : GRPoint;
          StringLength        : GTMaxString;
          CharString          : GAString;
   VAR ErrorInd               : INTEGER;
   VAR ConcatPoint            : GRPoint;
   VAR TextExtent             : GATextExtent
          );
```

INQUIRE LIST OF FILL AREA INDICES

L1a

Pascal level 1

```
procedure GInqListFillInd(
          WsId                : GTWsId;
          Start               : GTInt1;
          Size                : INTEGER;
   VAR Done                   : Boolean;
   VAR ErrorInd               : INTEGER;
   VAR NumEntries             : GTInt1;
   VAR DefinedEntries         : array [min..max:INTEGER] of INTEGER
          );
```

This procedure is a specific form of

INQUIRE LIST OF <Primitive> INDICES

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:

2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqListFillInd(
          WsId                : GTWsId;
          Start               : GTInt1;
          Size                : GTInqSize;
   VAR Done                   : Boolean;
   VAR ErrorInd               : INTEGER;
   VAR NumEntries             : GTInt1;
   VAR DefinedEntries         : GAInt
          );
```

This procedure is a specific form of

INQUIRE LIST OF <Primitive> INDICES

The parameters Start, Size and Done are defined in 3.11.

Errors:

2102 List element or set member not available

INQUIRE FILL AREA REPRESENTATION L1a

```
procedure GInqFillRep(
          WsId                 : GTWsId;
          FillInd              : GTInt1;
          TypeReturn           : GEReturn;
    VAR ErrorInd               : INTEGER;
    VAR FillRep                : GRFillRep
          );
```

This procedure is a specific form of

INQUIRE <Primitive> REPRESENTATION

INQUIRE LIST OF PATTERN INDICES L1a

Pascal level 1

```
procedure GInqListPatternInd(
          WsId                 : GTWsId;
          Start                : GTInt1;
          Size                 : INTEGER;
    VAR Done                   : Boolean;
    VAR ErrorInd               : INTEGER;
    VAR NumEntries             : GTInt0;
    VAR DefinedInd             : array [min..max:INTEGER] of INTEGER
          );
```

The parameter Size shall be $\leq$ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:

2100 *There is an incompatibility between the bounds of the array and the actual size parameters specified*
2102 *List element or set member not available*

Pascal level 0

```
procedure GInqListPatternInd(
          WsId                 : GTWsId;
          Start                : GTInt1;
          Size                 : GTInqSize;
    VAR Done                   : Boolean;
    VAR ErrorInd               : INTEGER;
    VAR NumEntries             : GTInt0;
    VAR DefinedInd             : GAInt
          );
```

The parameters Start, Size and Done are defined in 3.11.

Errors:

2102 *List element or set member not available*

INQUIRE PATTERN REPRESENTATION

L1a

Pascal level 1

```
procedure GInqPatternRep(
            WsId                    : GTWsId;
            PatternInd              : GTInt1;
            TypeReturn              : GEReturn;
    VAR ErrorInd                    : INTEGER;
    VAR Dx,
            Dy                      : GTInt1;
    VAR PatternArray                : array[min1..max1:INTEGER; min2..max2:INTEGER] of INTEGER
            );
```

The pattern array is within the bounds min1 to min1+Dx-1 (≤max1) and min2 to min2+Dy-1 (≤max2).

Errors:

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

Pascal level 0

```
procedure GInqPatternRep(
            WsId                    : GTWsId;
            PatternInd              : GTInt1;
            TypeReturn              : GEReturn;
    VAR ErrorInd                    : INTEGER;
    VAR Dx                          : GTMaxDX;
    VAR Dy                          : GTMaxDY;
    VAR PatternArray                : GAColrInqArray
            );
```

INQUIRE LIST OF COLOUR INDICES L0a

Pascal level 1

```
procedure GInqListColrInd(
         WsId              : GTWsId;
         Start             : GTInt1;
         Size              : INTEGER;
      VAR Done             : Boolean;
      VAR ErrorInd         : INTEGER;
      VAR NumEntries       : GTInt2;
      VAR DefinedInd       : array [min..max:INTEGER] of INTEGER
         );
```

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:

2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqListColrInd(
         WsId              : GTWsId;
         Start             : GTInt1;
         Size              : GTInqSize;
      VAR Done             : Boolean;
      VAR ErrorInd         : INTEGER;
      VAR NumEntries       : GTInt2;
      VAR DefinedInd       : GAInt
         );
```

The parameters Start, Size and Done are defined in 3.11.

Errors:

2102 List element or set member not available

INQUIRE COLOUR REPRESENTATION L0a

```
procedure GInqColrRep(
         WsId              : GTWsId;
         Ind               : GTInt0;
         TypeReturn        : GEReturn;
      VAR ErrorInd         : INTEGER;
      VAR Colr             : GRColr
         );
```

INQUIRE WORKSTATION TRANSFORMATION **L0a**

```
procedure GInqWsTran(
          WsId                : GTWsId;
   VAR ErrorInd               : INTEGER;
   VAR WsTranUpdSt            : GEWsTran;
   VAR ReqWsWindow,
          CurWsWindow,
          ReqWsViewport,
          CurWsViewport       : GRBound
          );
```

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION **L1a**

Pascal level 1

```
procedure GInqSegNamesWs(
          WsId                : GTWsId;
          Start               : GTInt1;
          Size                : INTEGER;
   VAR Done                   : Boolean;
   VAR ErrorInd               : INTEGER;
   VAR NumSegNames            : GTInt0;
   VAR StoredSegsWs           : array [min..max:INTEGER] of GTSeg
          );
```

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:
2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqSegNamesWs(
          WsId                : GTWsId;
          Start               : GTInt1;
          Size                : GTInqSize;
   VAR Done                   : Boolean;
   VAR ErrorInd               : INTEGER;
   VAR NumSegNames            : GTInt0;
   VAR StoredSegsWs           : GASeg
          );
```

The parameters Start, Size and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE LOCATOR DEVICE STATE	**L0b**
INQUIRE STROKE DEVICE STATE	**L0b**
INQUIRE VALUATOR DEVICE STATE	**L0b**
INQUIRE CHOICE DEVICE STATE	**L0b**
INQUIRE PICK DEVICE STATE	**L1b**
INQUIRE STRING DEVICE STATE	**L0b**

```
procedure GInqInputDeviceSt(
            InputClass            : GEInputClass;
            WsId                  : GTWsId;
            InputDeviceNum        : GTInt1;
            TypeReturn            : GEReturn;
    VAR ErrorInd                  : INTEGER;
    VAR OpMode                    : GEMode;
    VAR EchoSwitch                : GEEcho;
    VAR InputDeviceSt             : GRInput;
    VAR PromptEcho                : INTEGER;
    VAR EchoArea                  : GRBound;
    VAR InputDataRecord           : GRInputData
            );
```

This procedure is a general

INQUIRE <Input> DEVICE STATE

By substituting the components of the type GEInputClass, the GKS abstract functions are obtained.

Where GEReturn is not specified as required by the corresponding abstract GKS function, that is for the cases Valuator, Choice and String, then the supplied value of TypeReturn can be GVSet or GVRealized.

Errors:
2103 Pick is not supported in this level of GKS

INQUIRE LOCATOR DEVICE STATE L0b

```
procedure GInqLocatorDeviceSt(
           WsId                   : GTWsId;
           LocatorDeviceNum       : GTInt1;
           TypeReturn             : GEReturn;
       VAR ErrorInd               : INTEGER;
       VAR OpMode                 : GEMode;
       VAR EchoSwitch             : GEEcho;
       VAR InitialLocator         : GRLocator;
       VAR PromptEcho             : INTEGER;
       VAR EchoArea               : GRBound;
       VAR LocatorDataRec         : GRLocatorData
           );
```

This procedure is a specific form of

INQUIRE <Input> DEVICE STATE

INQUIRE STROKE DEVICE STATE L0b

```
procedure GInqStrokeDeviceSt(
           WsId                   : GTWsId;
           StrokeDeviceNum        : GTInt1;
           TypeReturn             : GEReturn;
       VAR ErrorInd               : INTEGER;
       VAR OpMode                 : GEMode;
       VAR EchoSwitch             : GEEcho;
       VAR InitialStroke          : GRStroke;
       VAR PromptEcho             : INTEGER;
       VAR EchoArea               : GRBound;
       VAR StrokeBufSize          : GTInt1;
       VAR StrokeDataRec          : GRStrokeData
           );
```

This procedure is a specific form of

INQUIRE <Input> DEVICE STATE

INQUIRE VALUATOR DEVICE STATE L0b

```
procedure GInqValuatorDeviceSt(
              WsId                : GTWsId;
              ValuatorDeviceNum   : GTInt1;
      VAR ErrorInd                : INTEGER;
      VAR OpMode                  : GEMode;
      VAR EchoSwitch              : GEEcho;
      VAR InitialValue            : REAL;
      VAR PromptEcho              : INTEGER;
      VAR EchoArea                : GRBound;
      VAR LowValue,
              HighValue           : REAL;
      VAR ValuatorDataRec         : GRValuatorData
              );
```

This procedure is a specific form of

INQUIRE <Input> DEVICE STATE

INQUIRE CHOICE DEVICE STATE L0b

```
procedure GInqChoiceDeviceSt(
              WsId                : GTWsId;
              ChoiceDeviceNum     : GTInt1;
      VAR ErrorInd                : INTEGER;
      VAR OpMode                  : GEMode;
      VAR EchoSwitch              : GEEcho;
      VAR InitialChoice           : GRChoice;
      VAR PromptEcho              : INTEGER;
      VAR EchoArea                : GRBound;
      VAR ChoiceDataRec           : GRChoiceData
              );
```

This procedure is a specific form of

INQUIRE <Input> DEVICE STATE

INQUIRE PICK DEVICE STATE L1b

```
procedure GInqPickDeviceSt(
          WsId                   : GTWsId;
          PickDeviceNum          : GTInt1;
          TypeReturn             : GEReturn;
   VAR ErrorInd                  : INTEGER;
   VAR OpMode                    : GEMode;
   VAR EchoSwitch                : GEEcho;
   VAR InitialPick               : GRPick;
   VAR PromptEcho                : INTEGER;
   VAR EchoArea                  : GRBound;
   VAR PickDataRec               : GRPickData
          );
```

This procedure is a specific form of

INQUIRE <Input> DEVICE STATE

INQUIRE STRING DEVICE STATE L0b

```
procedure GInqStringDeviceSt(
          WsId                   : GTWsId;
          StringDeviceNum        : GTInt1;
   VAR ErrorInd                  : INTEGER;
   VAR OpMode                    : GEMode;
   VAR EchoSwitch                : GEEcho;
   VAR InitialString             : GRString;
   VAR PromptEcho                : INTEGER;
   VAR EchoArea                  : GRBound;
   VAR StringBufSize,
          InitialPosition        : GTInt1;
   VAR StringDataRec             : GRStringData
          );
```

This procedure is a specific form of

INQUIRE <Input> DEVICE STATE

6.9.6 Inquiry functions for workstation description table

INQUIRE WORKSTATION CATEGORY L0a

```
procedure GInqWsCategory(
          WsType                 : GTWsType;
   VAR ErrorInd                  : INTEGER;
   VAR WsCategory                : GEWsCategory
          );
```

INQUIRE WORKSTATION CLASSIFICATION **L0a**

```
procedure GInqWsClass(
        WsType              : GTWsType;
    VAR ErrorInd            : INTEGER;
    VAR WsClass             : GEWsClass
        );
```

INQUIRE DISPLAY SPACE SIZE **L0a**

```
procedure GInqDisplaySize(
        WsType              : GTWsType;
    VAR ErrorInd            : INTEGER;
    VAR DisplaySize         : GRDisplaySize
        );
```

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES **L1a**

```
procedure GInqDynModWsAttr(
        WsType              : GTWsType;
    VAR ErrorInd            : INTEGER;
    VAR DynMod              : GRDynModWsAttr
        );
```

INQUIRE DEFAULT DEFERRAL STATE VALUES **L1a**

```
procedure GInqDefDeferSt(
        WsType              : GTWsType;
    VAR ErrorInd            : INTEGER;
    VAR DefDeferMode        : GEDefer;
    VAR DefRegenMode        : GEImplicitRegen
        );
```

INQUIRE POLYLINE FACILITIES **L0a**
INQUIRE POLYMARKER FACILITIES **L0a**
INQUIRE TEXT FACILITIES **L0a**
INQUIRE FILL AREA FACILITIES **L0a**

```
procedure GInqPrimFacil(
          Prim             : GEPrim;
          WsType           : GTWsType;
          Start            : GTInt1;
   VAR Done                : Boolean;
   VAR ErrorInd            : INTEGER;
   VAR PrimFacil           : GRPrimFacil
          );
```

This procedure is a general

INQUIRE <Primitive> FACILITIES

By substituting the components of the type GEPrim, the GKS abstract functions are obtained. The parameters Start and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE PREDEFINED POLYLINE REPRESENTATION **L0a**
INQUIRE PREDEFINED POLYMARKER REPRESENTATION **L0a**
INQUIRE PREDEFINED TEXT REPRESENTATION **L0a**
INQUIRE PREDEFINED FILL AREA REPRESENTATION **L0a**

```
procedure GInqPredPrimRep(
          Prim             : GEPrim;
          WsType           : GTWsType;
          PredPrimInd      : GTInt1;
   VAR ErrorInd            : INTEGER;
   VAR PrimRep             : GRPrimRep
          );
```

This procedure is a general

INQUIRE PREDEFINED <Primitive> REPRESENTATION

By substituting the components of the type GEPrim, the GKS abstract functions are obtained.

INQUIRE POLYLINE FACILITIES **L0a**

```
procedure GInqPolylineFacil(
         WsType                 : GTWsType;
         Start                  : GTInt1;
   VAR Done                     : Boolean;
   VAR ErrorInd                 : INTEGER;
   VAR Facil                    : GRLineFacil
         );
```

This procedure is a specific form of

INQUIRE <Primitive> FACILITIES

The parameters Start and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE PREDEFINED POLYLINE REPRESENTATION **L0a**

```
procedure GInqPredPolylineRep(
         WsType                 : GTWsType;
         PredInd                : GTInt1;
   VAR ErrorInd                 : INTEGER;
   VAR LineRep                  : GRLineRep
         );
```

This procedure is a specific form of

INQUIRE PREDEFINED <Primitive> REPRESENTATION

INQUIRE POLYMARKER FACILITIES **L0a**

```
procedure GInqPolymarkerFacil(
         WsType                 : GTWsType;
         Start                  : GTInt1;
   VAR Done                     : Boolean;
   VAR ErrorInd                 : INTEGER;
   VAR Facil                    : GRMarkerFacil
         );
```

This procedure is a specific form of

INQUIRE <Primitive> FACILITIES

The parameters Start and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE PREDEFINED POLYMARKER REPRESENTATION L0a

```
procedure GInqPredPolymarkerRep(
          WsType              : GTWsType;
          PredInd             : GTInt1;
      VAR ErrorInd            : INTEGER;
      VAR MarkerRep           : GRMarkerRep
          );
```

This procedure is a specific form of

INQUIRE PREDEFINED <Primitive> REPRESENTATION

INQUIRE TEXT FACILITIES L0a

```
procedure GInqTextFacil(
          WsType              : GTWsType;
          Start               : GTInt1;
      VAR Done                : Boolean;
      VAR ErrorInd            : INTEGER;
      VAR Facil               : GRTextFacil
          );
```

This procedure is a specific form of

INQUIRE <Primitive> FACILITIES

The parameters Start and Done are defined in 3.11.

Errors:

2102 List element or set member not available

INQUIRE PREDEFINED TEXT REPRESENTATION L0a

```
procedure GInqPredTextRep(
          WsType              : GTWsType;
          PredInd             : GTInt1;
      VAR ErrorInd            : INTEGER;
      VAR TextRep             : GRTextRep
          );
```

This procedure is a specific form of

INQUIRE PREDEFINED <Primitive> REPRESENTATION

INQUIRE FILL AREA FACILITIES **L0a**

```
procedure GInqFillFacil(
            WsType               : GTWsType;
            Start                : GTInt1;
    VAR Done                     : Boolean;
    VAR ErrorInd                 : INTEGER;
    VAR Facil                    : GRFillFacil
            );
```

This procedure is a specific form of

INQUIRE <Primitive> FACILITIES

The parameters Start and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE PREDEFINED FILL AREA REPRESENTATION **L0a**

```
procedure GInqPredFillRep(
            WsType               : GTWsType;
            PredInd              : GTInt1;
    VAR ErrorInd                 : INTEGER;
    VAR FillRep                  : GRFillRep
            );
```

This procedure is a specific form of

INQUIRE PREDEFINED <Primitive> REPRESENTATION

INQUIRE PATTERN FACILITIES **L0a**

```
procedure GInqPatternFacil(
            WsType               : GTWsType;
    VAR ErrorInd                 : INTEGER;
    VAR NumPredPatternInd        : GTInt0
            );
```

INQUIRE PREDEFINED PATTERN REPRESENTATION

L0a

Pascal level 1

```
procedure GInqPredPatternRep(
          WsType               : GTWsType;
          PredInd              : GTInt1;
    VAR ErrorInd               : INTEGER;
    VAR Dx,
          Dy                   : GTInt1;
    VAR PatternArray           : array[min1..max1:INTEGER; min2..max2:INTEGER] of INTEGER
          );
```

The pattern array bounds are min1 to min1+Dx-1 (≤max1) and min2 to min2+Dy-1 (≤max2).

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

Pascal level 0

```
procedure GInqPredPatternRep(
          WsType               : GTWsType;
          PredInd              : GTInt1;
    VAR ErrorInd               : INTEGER;
    VAR Dx                     : GTMaxDX;
    VAR Dy                     : GTMaxDY;
    VAR PatternArray           : GAColrInqArray
          );
```

INQUIRE COLOUR FACILITIES

L0a

```
procedure GInqColrFacil(
          WsType               : GTWsType;
    VAR ErrorInd               : INTEGER;
    VAR NumColrs               : GTInt0;
    VAR ColrAvail              : GEDisplay;
    VAR NumPredColrInd         : GTInt2
          );
```

INQUIRE PREDEFINED COLOUR REPRESENTATION

L0a

```
procedure GInqPredColrRep(
          WsType               : GTWsType;
          PredInd              : GTInt0;
    VAR ErrorInd               : INTEGER;
    VAR Colr                   : GRColr
          );
```

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES L0a

Pascal level 1

```
procedure GInqListGDP(
          WsType           : GTWsType;
          Start            : GTInt1;
          Size             : INTEGER;
   VAR Done                : Boolean;
   VAR ErrorInd            : INTEGER;
   VAR NumGDP              : GTInt0;
   VAR GDPId               : array [min..max:INTEGER] of GTGDPId
          );
```

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:
2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqListGDP(
          WsType           : GTWsType;
          Start            : GTInt1;
          Size             : GTInqSize;
   VAR Done                : Boolean;
   VAR ErrorInd            : INTEGER;
   VAR NumGDP              : GTInt0;
   VAR ListGDPId           : GAGDP
          );
```

The parameters Start, Size and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE GENERALIZED DRAWING PRIMITIVE L0a

```
procedure GInqGDP(
          WsType           : GTWsType;
          GDPId            : GTGDPId;
   VAR ErrorInd            : INTEGER;
   VAR NumAttr             : GTInt0;
   VAR ListAttr            : GSPrim
          );
```

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES **L0a**

```
procedure GInqMaxWsSt(
         WsType              : GTWsType;
     VAR ErrorInd            : INTEGER;
     VAR MaxNumPrim          : GAPrim;
     VAR MaxNumPattern       : GTInt0;
     VAR MaxNumColr          : GTInt2
         );
```

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED **L1a**

```
procedure GInqNumSegPriorities(
         WsType              : GTWsType;
     VAR ErrorInd            : INTEGER;
     VAR NumSegPriorities    : GTInt0
         );
```

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES **L1a**

```
procedure GInqDynModSegAttr(
         WsType              : GTWsType;
     VAR ErrorInd            : INTEGER;
     VAR SegAttrChangeable   : GASegMod
         );
```

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES **L0b**

```
procedure GInqNumInputDevices(
         WsType              : GTWsType;
     VAR ErrorInd            : INTEGER;
     VAR NumInputDevices     : GAInputClass
         );
```

INQUIRE DEFAULT LOCATOR DEVICE DATA	**L0b**
INQUIRE DEFAULT STROKE DEVICE DATA	**L0b**
INQUIRE DEFAULT VALUATOR DEVICE DATA	**L0b**
INQUIRE DEFAULT CHOICE DEVICE DATA	**L0b**
INQUIRE DEFAULT PICK DEVICE DATA	**L1b**
INQUIRE DEFAULT STRING DEVICE DATA	**L0b**

```
procedure GInqDefInputDeviceData(
            InputClass           : GEInputClass;
            WsType               : GTWsType;
            InputDeviceNum       : GTInt1;
            Start                : GTInt1;
    VAR Done                     : Boolean;
    VAR ErrorInd                 : INTEGER;
    VAR DefInputData             : GRDefInput
            );
```

This procedure is a general

INQUIRE DEFAULT <Input> DEVICE DATA

By substituting the components of the type GEInputClass, the GKS abstract functions are obtained. The parameters Start and Done are defined in 3.11.

Errors:
2102 List element or set member not available
2103 Pick is not supported in this level of GKS

INQUIRE DEFAULT LOCATOR DEVICE DATA **L0b**

```
procedure GInqDefLocatorDeviceData(
            WsType               : GTWsType;
            InputDeviceNum       : GTInt1;
            Start                : GTInt1;
    VAR Done                     : Boolean;
    VAR ErrorInd                 : INTEGER;
    VAR DefData                  : GRDefLocatorData
            );
```

This procedure is a specific form of

INQUIRE DEFAULT <Input> DEVICE DATA

The parameters Start and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE DEFAULT STROKE DEVICE DATA **L0b**

```
procedure GInqDefStrokeDeviceData(
          WsType          : GTWsType;
          InputDeviceNum  : GTInt1;
          Start           : GTInt1;
      VAR Done            : Boolean;
      VAR ErrorInd        : INTEGER;
      VAR DefData         : GRDefStrokeData
          );
```

This procedure is a specific form of

INQUIRE DEFAULT <Input> DEVICE DATA

The parameters Start and Done are defined in 3.11.

Errors:

2102 List element or set member not available

INQUIRE DEFAULT VALUATOR DEVICE DATA **L0b**

```
procedure GInqDefValuatorDeviceData(
          WsType          : GTWsType;
          InputDeviceNum  : GTInt1;
          Start           : GTInt1;
      VAR Done            : Boolean;
      VAR ErrorInd        : INTEGER;
      VAR DefData         : GRDefValuatorData
          );
```

This procedure is a specific form of

INQUIRE DEFAULT <Input> DEVICE DATA

The parameters Start and Done are defined in 3.11.

Errors:

2102 List element or set member not available

INQUIRE DEFAULT CHOICE DEVICE DATA L0b

```
procedure GInqDefChoiceDeviceData(
          WsType                : GTWsType;
          InputDeviceNum        : GTInt1;
          Start                 : GTInt1;
      VAR Done                  : Boolean;
      VAR ErrorInd              : INTEGER;
      VAR DefData               : GRDefChoiceData
          );
```

This procedure is a specific form of

INQUIRE DEFAULT <Input> DEVICE DATA

The parameters Start and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE DEFAULT PICK DEVICE DATA L1b

```
procedure GInqDefPickDeviceData(
          WsType                : GTWsType;
          InputDeviceNum        : GTInt1;
          Start                 : GTInt1;
      VAR Done                  : Boolean;
      VAR ErrorInd              : INTEGER;
      VAR DefData               : GRDefPickData
          );
```

This procedure is a specific form of

INQUIRE DEFAULT <Input> DEVICE DATA

The parameters Start and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE DEFAULT STRING DEVICE DATA

L0b

```
procedure GInqDefStringDeviceData(
         WsType                : GTWsType;
         InputDeviceNum        : GTInt1;
         Start                 : GTInt1;
   VAR Done                    : Boolean;
   VAR ErrorInd                : INTEGER;
   VAR DefData                 : GRDefStringData
         );
```

This procedure is a specific form of

INQUIRE DEFAULT <Input> DEVICE DATA

The parameters Start and Done are defined in 3.11.

Errors:

2102 List element or set member not available

6.9.7 Inquiry functions for segment state list

INQUIRE SET OF ASSOCIATED WORKSTATIONS L1a

Pascal level 1

```
procedure GInqAssocWs(
          SegName              : GTSeg;
          Start                : GTInt1;
          Size                 : INTEGER;
   VAR Done                    : Boolean;
   VAR ErrorInd                : INTEGER;
   VAR NumberAssocWs           : GTInt1;
   VAR AssocWsId               : array [min..max:INTEGER] of GTWsId
          );
```

The parameter Size shall be ≤ max-min+1. The parameters Start, Size and Done are defined in 3.11.

Errors:
2100 There is an incompatibility between the bounds of the array and the actual size parameters specified
2102 List element or set member not available

Pascal level 0

```
procedure GInqAssocWs(
          SegName              : GTSeg;
          Start                : GTInt1;
          Size                 : GTInqSize;
   VAR Done                    : Boolean;
   VAR ErrorInd                : INTEGER;
   VAR NumberAssocWs           : GTInt1;
   VAR AssocWsId               : GAWsId
          );
```

The parameters Start, Size and Done are defined in 3.11.

Errors:
2102 List element or set member not available

INQUIRE SEGMENT ATTRIBUTES L1a

```
procedure GInqSegAttr(
          SegName              : GTSeg;
   VAR ErrorInd                : INTEGER;
   VAR SegAttr                 : GRSegAttr
          );
```

6.9.8 Pixel inquiries

INQUIRE PIXEL ARRAY DIMENSIONS **L0a**

```
procedure GInqPixelArrayDim(
            WsId                   : GTWsId;
            p,
            q                      : GRPoint;
        VAR ErrorInd               : INTEGER;
        VAR Dx,
            Dy                     : GTInt1
            );
```

INQUIRE PIXEL ARRAY **L0a**

Pascal level 1

```
procedure GInqPixelArray(
            WsId                   : GTWsId;
            p                      : GRPoint;
            Dx,
            Dy                     : GTInt1;
        VAR ErrorInd               : INTEGER;
        VAR InvalidValues          : GEInvPixel;
        VAR ColrIndArray           : array[min1..max1 : INTEGER; min2..max2:INTEGER] of INTEGER
            );
```

The colour index array bounds are min1 to min1+Dx-1 (≤max1) and min2 to min2+Dy-1 (≤max2).

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

Pascal level 0

```
procedure GInqPixelArray(
            WsId                   : GTWsId;
            p                      : GRPoint;
            Dx                     : GTMaxDX;
            Dy                     : GTMaxDY;
        VAR ErrorInd               : INTEGER;
        VAR InvalidValues          : GEInvPixel;
        VAR ColrIndArray           : GAColrInqArray
            );
```

INQUIRE PIXEL L0a

```
procedure GInqPixel(
        WsId                   : GTWsId;
        p                      : GRPoint;
    VAR ErrorInd               : INTEGER;
    VAR ColrInd                : INTEGER
        );
```

6.9.9 Inquiry function for GKS error state list

INQUIRE INPUT QUEUE OVERFLOW L0c

```
procedure GInqInputOverflow(
    VAR ErrorInd               : INTEGER;
    VAR WsId                   : GTWsId;
    VAR InputClass             : GEInputClass;
    VAR InputDeviceNum         : GTInt1
        );
```

6.10 Utility functions

EVALUATE TRANSFORMATION MATRIX L1a

```
procedure GEvalTran(
        FixedPoint             : GRPoint;
        ShiftVector            : GRVector;
        RotationAngle          : REAL;
        Scale                  : GRVector;
        CoordinateSwitch       : GECoordSwitch;
    VAR SegTranMatrix          : GAMatrix
        );
```

ACCUMULATE TRANSFORMATION MATRIX L1a

```
procedure GAccumTran(
        InsegTran              : GAMatrix;
        FixedPoint             : GRPoint;
        ShiftVector            : GRVector;
        RotationAngle          : REAL;
        Scale                  : GRVector;
        CoordinateSwitch       : GECoordSwitch;
    VAR SegTran                : GAMatrix
        );
```

6.11 Error handling

EMERGENCY CLOSE GKS **L0a**

```
procedure GEmergencyCloseGKS;
```

ERROR HANDLING **L0a**

```
procedure GErrorHandling(
        ErrorNum        : INTEGER;
        ProcId          : GAProcName;
        ErrorFile       : GTErrorFileName
        );
```

ERROR LOGGING **L0a**

```
procedure GErrorLogging(
        ErrorNum        : INTEGER;
        ProcId          : GAProcName;
        ErrorFile       : GTErrorFileName
        );
```

Annex A

Data types in compilation order

(This annex does not form an integral part of the standard, but provides additional information.)

This annex categorizes the constants and types of the binding, and gives a possible ordering of type definitions that meets the Pascal requirements.

A.1 Implementation defined constants

GCDefErrorLog
GCMaxDX
GCMaxDY
GCMaxEscapeIn
GCMaxEscapeOut
GCMaxFile
GCMaxGDP
GCMaxInq
GCMaxItem
GCMaxMemory
GCMaxName
GCMaxPoint
GCMaxString

A.2 Required constants

GCCircleMarker
GCCrossMarker
GCDashDotLine
GCDashedLine
GCDefMemory
GCDotMarker
GCDottedLine
GCPlusMarker
GCSolidLine
GCStarMarker

A.3 Implementation defined tag types

GTChoiceDataTag
GTEscapeDataTag
GTGDPDataTag
GTLocatorDataTag
GTPickDataTag
GTStringDataTag
GTStrokeDataTag
GTValuatorDataTag

A.4 Error logging and connection files

GTErrorFileName
GAConnId

A.5 General types

GTInqSize
GTInt0
GTInt1
GTInt2
GTInt3
GTMaxDX
GTMaxDY
GTMaxEscapeIn
GTMaxEscapeOut
GTMaxGDP
GTMaxItem
GTMaxPoint0
GTMaxPoint1
GTMaxPoint2
GTMaxPoint3
GTMaxString
GTMemory
GTEscapeId
GTGDPId
GTPickId
GTSeg
GTWsId
GTWsType
GRBound
GRIntVector
GRVector
GRPoint
GAInt
GAColrArray
GAColrInqArray
GAGDP
GAPickId
GAPointArray
GASeg
GAString
GATextExtent
GAProcName
GAWsId
GAWsType
GEInvPixel

A.6 Types applicable to workstation control procedures

GEControl
GEDefer
GEImplicitRegen
GEUpdRegen

GEWsSt

A.7 Types applicable to transformation procedures

GEClip
GEPriority

A.8 Types applicable to attribute setting procedures

GEASF
GEHorizontal
GEInterior
GEPath
GEPrec
GEPrim
GEPrimAttr
GEVertical
GRAlign
GRColr
GRFontPrec
GRPrimRep
GAASF
GAFontPrec
GAPrimRep
GRText
GSInterior
GSPrim

A.9 Types applicable to segment procedures

GECoordSwitch
GEDet
GEHighlight
GEVis
GAMatrix

A.10 Types applicable to input procedures

GEEcho
GEEvents
GEEventClass
GEMode
GEPrompt
GEReqStatus
GEInputClass
GEInputStatus

A.11 Types applicable to GKS description

GELevel
GRWsMaxNum

A.12 Types applicable to GKS state

GEOpSt
GRPrimAttr

A.13 Types applicable to workstation state

GENfan
GEReturn
GESurface
GEWsTran
GRDeferUpd

A.14 Types applicable to workstation description

GEDeviceUnits
GEDisplay
GEDynMod
GESegAttr
GEWsCategory
GEWsClass
GAInputClass
GAPrim
GASegMod
GRDisplaySize
GRDynModWsAttr

A.15 Types applicable to segment state

GRSegAttr

A.16 GKS data records

GRChoiceData
GREscapeDataIn
GREscapeDataOut
GAEscapeInInt
GAEscapeInReal
GAEscapeInString
GAEscapeOutInt
GAEscapeOutReal
GAEscapeOutString
GRFileData
GRGDPData
GAGDPInt
GAGDPReal
GAGDPString
GAItemInt
GAItemReal
GAItemString
GRLocatorData
GRPickData
GRStringData
GRStrokeData
GRValuatorData

A.17 Types applicable to the one-one procedures

GRChoice
GRDefChoiceData
GRDefLocatorData
GRDefPickData
GRDefStringData
GRDefStrokeData
GRDefValuatorData
GRFillFacil
GRFillRep
GRLineFacil
GRLineRep
GRLocator
GRMarkerFacil
GRMarkerRep
GRPick
GRString
GRStroke
GRTextFacil
GRTextRep

A.18 Types applicable to the many-one procedures

GRInput
GRInputData
GRDefInput
GRPrimFacil

Annex B

Metafile Item Types

(This annex does not form an integral part of the standard, but provides additional information.)

The GET ITEM TYPE FROM GKSM function returns an integer value which uniquely identifies the type of the next metafile item. However, the value of the item type may vary depending on the metafile implementation. In order to allow application programs to be written in a manner which is independent of the metafile implementation, the following Pascal names are suggested. The implementation should define these names with values which match the values returned by the GET ITEM TYPE FROM GKSM procedure. The USER ITEM START item is used to indicate the first user item type. All of the integers corresponding to the other items in the table are less than the value of USER ITEM START. All user items should have a value greater than or equal to the value of USER ITEM START.

Table B.1 - Pascal Item Type Names

GKSM Item Type	Pascal Name
ASPECT SOURCE FLAGS	GITASF
CELL ARRAY	GITCellArray
CHARACTER EXPANSION FACTOR	GITCharExpan
CHARACTER SPACING	GITCharSpacing
CHARACTER VECTORS	GITCharVectors
CLEAR WORKSTATION	GITClearWs
CLIPPING RECTANGLE	GITClipRect
CLOSE SEGMENT	GITCloseSeg
COLOUR REPRESENTATION	GITColrRep
CREATE SEGMENT	GITCreateSeg
DEFERRAL STATE	GITDeferSt
DELETE SEGMENT	GITDelSeg
END ITEM	GITEndItem
ESCAPE	GITEscape
FILL AREA	GITFill
FILL AREA COLOUR INDEX	GITFillColrInd
FILL AREA INDEX	GITFillInd
FILL AREA INTERIOR STYLE	GITFillIntStyle
FILL AREA REPRESENTATION	GITFillRep
FILL AREA STYLE INDEX	GITFillStyleInd
GENERALIZED DRAWING PRIMITIVE (GDP)	GITGDP
LINETYPE	GITLineType
LINEWIDTH SCALE FACTOR	GITLineWidthScale
MARKER SIZE SCALE FACTOR	GITMarkerSizeScale
MARKER TYPE	GITMarkerType
MESSAGE	GITMessage
PATTERN REFERENCE POINT	GITPatternRefPoint
PATTERN REPRESENTATION	GITPatternRep
PATTERN VECTORS	GITPatternVectors
PICK IDENTIFIER	GITPickId
POLYLINE	GITPolyline

GKSM Item Type	Pascal Name
POLYLINE COLOUR INDEX	GITLineColrInd
POLYLINE INDEX	GITPolylineInd
POLYLINE REPRESENTATION	GITPolylineRep
POLYMARKER	GITPolymarker
POLYMARKER COLOUR INDEX	GITMarkerColrInd
POLYMARKER INDEX	GITPolymarkerInd
POLYMARKER REPRESENTATION	GITPolymarkerRep
REDRAW ALL SEGMENTS ON WORKSTATION	GITRedrawSegWs
RENAME SEGMENT	GITRenameSeg
SET DETECTABILITY	GITSetDet
SET HIGHLIGHTING	GITSetHighlight
SET SEGMENT PRIORITY	GITSetSegPriority
SET SEGMENT TRANSFORMATION	GITSetSegTran
SET VISIBILITY	GITSetVis
TEXT	GITText
TEXT ALIGNMENT	GITTextAlign
TEXT COLOUR INDEX	GITTextColrInd
TEXT FONT AND PRECISION	GITTextFontPrec
TEXT INDEX	GITTextInd
TEXT PATH	GITTextPath
TEXT REPRESENTATION	GITTextRep
UPDATE WORKSTATION	GITUpdWs
USER ITEM START	GITUserItem
WORKSTATION VIEWPORT	GITWsViewport
WORKSTATION WINDOW	GITWsWindow