American National Standard

for information systems –

computer graphics – graphical kernel system (GKS) pascal binding

ADOPTED FOR USE BY THE FEDERAL GOVERNMENT

FIPS

PUB 120-1

SEE NOTICE ON INSIDE

american national standards institute, inc.
1430 broadway, new york, new york 10018

ANSI X3.124.2-1988

This standard has been adopted for Federal Government use.

Details concerning its use within the Federal Government are contained in Federal Information Processing Standards Publication 120-1, Graphical Kernel System (GKS). For a complete list of the publications available in the Federal Information Processing Standards Series, write to the Standards Processing Coordinator (ADP), National Institute of Standards and Technology, Gaithersburg, MD 20899.

ANSI®
X3.124.2-1988

American National Standard
for Information Systems –

Computer Graphics – Graphical Kernel System (GKS) Pascal Binding

Secretariat

Computer and Business Equipment Manufacturers Association

Approved February 18, 1988

American National Standards Institute, Inc

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made toward their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

Published by

American National Standards Institute
1430 Broadway, New York, New York 10018

Printed in the United States of America

PCIM688/28

Foreword (This Foreword is not part of American National Standard X3.124.2-1988.)

This American National Standard provides access to a set of basic functions for computer graphics programming in American National Standard Programming Language Pascal, ANSI/IEEE 770X3.97-1983. These graphics functions taken as a whole are called the Pascal language binding of the Graphical Kernel System (GKS).

This standard defines a Pascal application level programming interface to a graphics system. Hence, it contains functions for (1) outputting graphical primitives, (2) controlling the appearance of graphical primitives with attributes, (3) controlling graphical workstations, (4) controlling transformation and coordinate systems, (5) generating and controlling groups of primitives called segments, (6) obtaining graphical input, (7) manipulating groups of device-independent instructions called metafiles, (8) inquiring the capabilities and states of the graphics system, and (9) handling errors.

For each GKS function, a Pascal procedure declaration is given. In addition, any special errors associated only with the Pascal language binding of GKS are specified and assigned unique error numbers. Finally, all of the data types and constants necessary to access the procedures are defined.

Twelve upwardly compatible levels of conformance are defined, addressing the most common classes of equipment and applications.

American National Standard for Information Systems - Computer Graphics - Graphical Kernel System (GKS) Functional Description, ANSI X3.124-1985, is supplemented by this derivative standard. ANSI X3.124-1985 corresponds to ISO 7942-1985 in that it represents the functional aspects of GKS. ANSI X3.124.2-1988 contains specifications not present in ANSI X3.124-1985, namely, the syntax for using GKS functions and data types from Pascal.

This standard was developed by Technical Committee X3H3 of American National Standards Committee X3 under project 531-D authorized by X3 (described in document X3H3/85-116).

This standard was approved as an American National Standard by the American National Standards Institute on February 18, 1988.

Suggestions for improvement of this standard will be welcome. They should be sent to Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001.

This standard was processed and approved for submittal to ANSI by Accredited Standards Committee on Information Processing Systems, X3. Committee approval of the standard does not necessarily imply that all committee members voted for its approval. At the time it approved this standard, the X3 committee had the following members:

Richard Gibson, Chair
Donald C. Loughry, Vice-Chair
Catherine A. Kachurik, Administrative Secretary

Organization Represented	*Name of Representative*
American Library Association	Paul Peters
American Nuclear Society	Geraldine C. Main

Organization Represented	*Name of Representative*
	Sally Hartzell (Alt)
AMP Incorporated	Edward Kelly
Apple	Karl Kimball
	Michael J. Lawler (Alt)
Association of the Institute for Certification of Computer Professionals	Thomas M. Kurihara
AT&T	Paul D. Bartoli
	Thomas F. Frost (Alt)
Control Data Corporation	Charles E. Cooper
	Keith Lucke (Alt)
Cooperating Users of Burroughs Equipment	Thomas Easterday
	Donald Miller (Alt)
Data General Corporation	Lee Schiller
	Lyman Chapin (Alt)
Data Processing Management Association	Ward Arrington
	Wallace R. McPherson (Alt)
Digital Equipment Computer Users Society	Dennis Perry
Digital Equipment Corporation	Gary S. Robinson
	Delbert L. Shoemaker (Alt)
Eastman Kodak	Gary Haines
	Charleton C. Bard (Alt)
General Electric Company	Richard W. Signor
	William R. Kruesi (Alt)
GUIDE International	Frank Kirshenbaum
	Sandra Swartz Abraham (Alt)
Hewlett-Packard	Donald C. Loughry
Honeywell Bull	David M. Taylor
IBM Corporation	Mary Anne Gray
	Robert H. Follett (Alt)
IEEE Computer Society	Sava I. Sherr
	Thomas A. Varetoni (Alt)
	Helen Wood (Alt)
Lawrence Berkeley Laboratory	David F. Stevens
	Robert L. Fink (Alt)
MAP/TOP	James D. Converse
	Mike Kaminski (Alt)
Moore Business Forms	Delmer H. Oddy
National Bureau of Standards	Robert E. Rountree
	James H. Burrows (Alt)
National Communications System	Dennis Bodson
	George W. White (Alt)
NCR Corporation	William E. Snyder
	A. Raymond Daniels (Alt)
OMNICOM	Harold C. Folts
	Catherine Howells (Alt)
Prime Computer, Inc	Arthur Norton
	Donna A. Poulack (Alt)
Railinc Corporation	Moncure N. Lyon
Recognition Technology Users Association	Herbert F. Schantz

Organization Represented	*Name of Representative*
	G. W. Wetzel (Alt)
Scientific Computer Systems Corporation	James A. Baker
	Carl Haberland (Alt)
SHARE, Inc	Thomas B. Steel
	Robert A. Rannie (Alt)
3M Company	Paul D. Jahnke
Travelers Insurance Companies, Inc	Joseph T. Brophy
Unisys	Marvin W. Bass
	Stanley Fenner (Alt)
U.S. Department of Defense	Fred Virtue
	Belkis Leong-Hong (Alt)
U.S. General Services Administration	William C. Rinehuls
	Larry L. Jackson (Alt)
VIM	Chris Tanner
	Madeleine Sparks (Alt)
VISA U.S.A.	Jean T. McKenna
	Patty Greenhalgh (Alt)
Wang	J. J. Cinecoe
	Sarah Wagner (Alt)
Xerox Corporation	John L. Wheeler
	Roy Pierce (Alt)

Technical Committee X3H3 on Computer Graphics, which developed and reviewed the draft proposals, had the following members:

Peter R. Bono, Chair
John McConnell, Vice-Chair
James Michener, Secretary
Janet Chin, International Representative
David Larson, Document Editor

Salim Abi-Ezzi
David Ambrose
Roxann Arey
David Bagby
David Bailey
Bob Benson
John Blair
Loren Buchanan
John Butler
Debbie Cahn
George Carson
Janet Chin
Bruce Coughran
Geraldine R. Cuthbert
Frank Dawson
Sahib A. Dudani
Richard Ehlers
Ted Fisher
Jim Flatten
Zhanna Gelman-Reyf
Jeffrey Gishen
Wolfgang Gnettner
Richard Greco
Jack Grimes

Giora Guth
Richard Hagy
Daniel Hahn
Jim Hargrove
Terry Harney
Mike Heck
Lofton Henderson
Raymond Hoffman
S. Hoshimoto
Joyce Howell
John Jeter
Peter Jones
Jim Kearney
Joseph Kowalik
Joseph Lamm
Mark Landguth
Franklin Lim
Magsood Mahmud
Peter Masters
Russell McDowell
Eileen B. McGinnis
James Michener
T. Motoyama
David A. Nation

Christopher Nelson
Steven Olson
Dino Pavlakos
Burt Perry
Neil Pierman
Mike Pittman
Brian Plunkett
Paul Porter
Tom Powers
Richard Puk
Ted Reed
Brad Rice
Reed Rinn
Cynthia Rodriguez
Russel Rogers
Jeff Rowe
Daryl Salmons
Gregory Saunders
George Schaeffer
Michael Schulman
Charles Seum
Niru Shah
Shreyas Shah
Barry Shepherd
Mark Skall
Norman Soong
Madeleine R. Sparks
Johathan Steinhart
David Straayer
Daniel Strom
George Sutty
Jon Turner
Karla Veechiet
Lynn Whittington
Meredith L. Whyles
John Yambor

The document editor for this standard was David Larson (Hewlett-Packard Co.). The camera-ready masters for the body of this standard were provided by Hewlett-Packard Company, Fort Collins, Colorado, U.S.A.

Contents

Appendixes

American National Standard
for Information Systems –

Computer Graphics – Graphical Kernel System (GKS) Pascal Binding

0. Introduction

The Graphical Kernel System (GKS), the functional description of which is given in ANSI X3.124-1985, 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 standard is to define a standardized binding for the Pascal computer programming language.

1. Scope and Field of Application

The Graphical Kernel System (GKS), as described in ANSI X3.124-1985, 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 ANSI X3.124 specifies such a language-dependent layer for the Pascal language.

2. Referenced Standards

2.1 Referenced American National Standards

This standard is intended for use with the following American National Standards:

ANSI X3.124-1985, Information Systems - Computer Graphics - Graphical Kernel System (GKS) Functional Description.

ANSI/IEEE 770X3.97-1983, Pascal Computer Programming Language.

2.2 Other Referenced Standards

This standard is also intended for use with the following International Standards:

ISO 2382-13:1984, Data processing - Vocabulary - Part 13: Computer Graphics.

ISO 7185:1983, Programming languages - Pascal.

3. The Pascal Language Binding of GKS

3.1 Specification

The GKS language binding interface for Pascal, as described in ANSI/IEEE 770X3.97-1983, 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.

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
LENGTH	NULL
LIST	NULL
LOGICAL	NULL
MATRIX	NULL
MAXIMUM	Max
MODIFICATION	Mod
NAME	NULL
NORMALIZATION	Norm

Table 1 - Abbreviations Ordered Alphabetically

GKS word	Abbreviation
NUMBER	Num
NUMBERS	Num
OF	NULL
ON	NULL
OPERATING	Op
POLYLINE	Line
POLYMARKER	Markcr
PRECISION	Prec
PREDEFINED	Pred
PRIMITIVE	Prim
QUEUE	NULL
REFERENCE	Ref
REPRESENTATION	Rep
REQUEST	Req
SEGMENT	Seg
SEGMENTS	Seg
SET	NULL
SIMULTANEOUS	NULL
SPACE	NULL
STATE	St
SUPPORTED	NULL
TABLES	NULL
TO	NULL
TRANSFORMATION	Tran
UPDATE	Upd
USE	NULL
VALUE	NULL
VALUES	NULL
VISIBILITY	Vis
WITH	NULL
WORKSTATION	Ws

NOTE - NULL represents the null string

Table 2 - GKS Function Names and Pascal Names Ordered by Pascal Name

GKS Function Name	Level	Pascal Name
ACCUMULATE TRANSFORMATION MATRIX	L1a	GAccumTran
ACTIVATE WORKSTATION	Lma	GActivateWs
ASSOCIATE SEGMENT WITH WORKSTATION	L2a	GAssocSegWs
AWAIT EVENT	Lmc	GAwaitEvent
CELL ARRAY	L0a	GCellArray
CLEAR WORKSTATION	Lma	GClearWs
CLOSE GKS	Lma	GCloseGKS
CLOSE SEGMENT	L1a	GCloseSeg
CLOSE WORKSTATION	Lma	GCloseWs
COPY SEGMENT TO WORKSTATION	L2a	GCopySegWs
CREATE SEGMENT	L1a	GCreateSeg
DEACTIVATE WORKSTATION	Lma	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	Lma	GEscape
ESCAPE	Lma	GEscapeGeneralized
EVALUATE TRANSFORMATION MATRIX	L1a	GEvalTran
FILL AREA	Lma	GFill
FLUSH DEVICE EVENTS	Lmc	GFlushDeviceEvents
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDP
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDPGeneralized
GET CHOICE	Lmc	GGetChoice
GET ITEM TYPE FROM GKSM	L0a	GGetItemType
GET LOCATOR	Lmc	GGetLocator
GET PICK	L1c	GGetPick
GET STRING	Lmc	GGetString
GET STROKE	Lmc	GGetStroke
GET VALUATOR	Lmc	GGetValuator
INITIALISE CHOICE	Lmb	GInitChoice
INITIALISE LOCATOR	Lmb	GInitLocator
INITIALISE PICK	L1b	GInitPick
INITIALISE STRING	Lmb	GInitString
INITIALISE STROKE	Lmb	GInitStroke
INITIALISE VALUATOR	Lmb	GInitValuator
INQUIRE ASPECT SOURCE FLAGS	Lma	GInqASF
INQUIRE SET OF ACTIVE WORKSTATIONS	L1a	GInqActiveWs
INQUIRE SET OF ASSOCIATED WORKSTATIONS	L1a	GInqAssocWs
INQUIRE CHARACTER BASE VECTOR	Lma	GInqCharBaseVector
INQUIRE CHARACTER EXPANSION FACTOR	Lma	GInqCharExpan
INQUIRE CHARACTER HEIGHT	Lma	GInqCharHeight
INQUIRE CHARACTER SPACING	Lma	GInqCharSpacing
INQUIRE CHARACTER UP VECTOR	Lma	GInqCharUpVector
INQUIRE CHARACTER WIDTH	Lma	GInqCharWidth
INQUIRE CHOICE DEVICE STATE	Lmb	GInqChoiceDeviceSt
INQUIRE CLIPPING	Lma	GInqClip
INQUIRE COLOUR FACILITIES	Lma	GInqColrFacil

Table 2 - GKS Function Names and Pascal Names Ordered by Pascal Name

GKS Function Name	Level	Pascal Name
INQUIRE COLOUR REPRESENTATION	Lma	GInqColrRep
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	Lma	GInqCurIndivAttr
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	Lma	GInqCurNormTranNum
INQUIRE CURRENT PICK IDENTIFIER	L1b	GInqCurPickId
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	Lma	GInqCurPrimAttr
INQUIRE DEFAULT CHOICE DEVICE DATA	Lmb	GInqDefChoiceDeviceData
INQUIRE DEFAULT DEFERRAL STATE VALUES	L1a	GInqDefDeferSt
INQUIRE DEFAULT LOCATOR DEVICE DATA	Lmb	GInqDefLocatorDeviceData
INQUIRE DEFAULT PICK DEVICE DATA	L1b	GInqDefPickDeviceData
INQUIRE DEFAULT STRING DEVICE DATA	Lmb	GInqDefStringDeviceData
INQUIRE DEFAULT STROKE DEVICE DATA	Lmb	GInqDefStrokeDeviceData
INQUIRE DEFAULT VALUATOR DEVICE DATA	Lmb	GInqDefValuatorDeviceData
INQUIRE DISPLAY SPACE SIZE	Lma	GInqDisplaySize
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES	L1a	GInqDynModSegAttr
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	L1a	GInqDynModWsAttr
INQUIRE FILL AREA COLOUR INDEX	Lma	GInqFillColrInd
INQUIRE FILL AREA FACILITIES	Lma	GInqFillFacil
INQUIRE FILL AREA INDEX	Lma	GInqFillInd
INQUIRE FILL AREA INTERIOR STYLE	Lma	GInqFillIntStyle
INQUIRE FILL AREA REPRESENTATION	L1a	GInqFillRep
INQUIRE FILL AREA STYLE INDEX	Lma	GInqFillStyleInd
INQUIRE GENERALIZED DRAWING PRIMITIVE	L0a	GInqGDP
INQUIRE INPUT QUEUE OVERFLOW	Lmc	GInqInputOverflow
INQUIRE LEVEL OF GKS	Lma	GInqLevelGKS
INQUIRE POLYLINE COLOUR INDEX	Lma	GInqLineColrInd
INQUIRE POLYLINE INDEX	Lma	GInqLineInd
INQUIRE LINETYPE	Lma	GInqLineType
INQUIRE LINEWIDTH SCALE FACTOR	Lma	GInqLineWidthScale
INQUIRE LIST OF COLOUR INDICES	Lma	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 TEXT INDICES	L1a	GInqListTextInd
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	L0a	GInqListWsTypes
INQUIRE LOCATOR DEVICE STATE	Lmb	GInqLocatorDeviceSt
INQUIRE POLYMARKER COLOUR INDEX	Lma	GInqMarkerColrInd
INQUIRE POLYMARKER INDEX	Lma	GInqMarkerInd
INQUIRE POLYMARKER SIZE SCALE FACTOR	Lma	GInqMarkerSizeScale
INQUIRE POLYMARKER TYPE	Lma	GInqMarkerType
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	L0a	GInqMaxNormTranNum
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	Lma	GInqMaxWsSt
INQUIRE MORE SIMULTANEOUS EVENTS	Lmc	GInqMoreEvents
INQUIRE NORMALIZATION TRANSFORMATION	Lma	GInqNormTran
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	Lmb	GInqNumInputDevices
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED	L1a	GInqNumSegPriorities
INQUIRE OPERATING STATE VALUE	L0a	GInqOpSt

Table 2 - GKS Function Names and Pascal Names Ordered by Pascal Name

GKS Function Name	Level	Pascal Name
INQUIRE NAME OF OPEN SEGMENT	L1a	GInqOpenSeg
INQUIRE SET OF OPEN WORKSTATIONS	L0a	GInqOpenWs
INQUIRE PATTERN FACILITIES	L0a	GInqPatternFacil
INQUIRE PATTERN REFERENCE POINT	Lma	GInqPatternRefPoint
INQUIRE PATTERN REPRESENTATION	L1a	GInqPatternRep
INQUIRE PATTERN SIZE	Lma	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	Lma	GInqPolylineFacil
INQUIRE POLYLINE REPRESENTATION	L1a	GInqPolylineRep
INQUIRE POLYMARKER FACILITIES	Lma	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 TEXT REPRESENTATION	L0a	GInqPredTextRep
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	Lmb	GInqStringDeviceSt
INQUIRE STROKE DEVICE STATE	Lmb	GInqStrokeDeviceSt
INQUIRE TEXT ALIGNMENT	Lma	GInqTextAlign
INQUIRE TEXT COLOUR INDEX	Lma	GInqTextColrInd
INQUIRE TEXT EXTENT	Lma	GInqTextExtent
INQUIRE TEXT FACILITIES	Lma	GInqTextFacil
INQUIRE TEXT FONT AND PRECISION	Lma	GInqTextFontPrec
INQUIRE TEXT INDEX	Lma	GInqTextInd
INQUIRE TEXT PATH	Lma	GInqTextPath
INQUIRE TEXT REPRESENTATION	L1a	GInqTextRep
INQUIRE VALUATOR DEVICE STATE	Lmb	GInqValuatorDeviceSt
INQUIRE WORKSTATION CATEGORY	L0a	GInqWsCategory
INQUIRE WORKSTATION CLASSIFICATION	L0a	GInqWsClass
INQUIRE WORKSTATION CONNECTION AND TYPE	Lma	GInqWsConnType
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	L0a	GInqWsDeferUpdSt
INQUIRE WORKSTATION MAXIMUM NUMBERS	L1a	GInqWsMaxNum
INQUIRE WORKSTATION STATE	L0a	GInqWsSt
INQUIRE WORKSTATION TRANSFORMATION	Lma	GInqWsTran
INSERT SEGMENT	L2a	GInsertSeg
INTERPRET ITEM	L0a	GInterpretItem
MESSAGE	L1a	GMessage
OPEN GKS	Lma	GOpenGKS
OPEN WORKSTATION	Lma	GOpenWs
POLYLINE	Lma	GPolyline
POLYMARKER	Lma	GPolymarker
READ ITEM FROM GKSM	L0a	GReadItem

Table 2 - GKS Function Names and Pascal Names Ordered by Pascal Name

GKS Function Name	Level	Pascal Name
REDRAW ALL SEGMENTS ON WORKSTATION	L1a	GRedrawSegWs
RENAME SEGMENT	L1a	GRenameSeg
REQUEST CHOICE	Lmb	GReqChoice
REQUEST LOCATOR	Lmb	GReqLocator
REQUEST PICK	L1b	GReqPick
REQUEST STRING	Lmb	GReqString
REQUEST STROKE	Lmb	GReqStroke
REQUEST VALUATOR	Lmb	GReqValuator
SAMPLE CHOICE	Lmc	GSampleChoice
SAMPLE LOCATOR	Lmc	GSampleLocator
SAMPLE PICK	L1c	GSamplePick
SAMPLE STRING	Lmc	GSampleString
SAMPLE STROKE	Lmc	GSampleStroke
SAMPLE VALUATOR	Lmc	GSampleValuator
SELECT NORMALIZATION TRANSFORMATION	Lma	GSelectNormTran
SET ASPECT SOURCE FLAGS	L0a	GSetASF
SET CHARACTER EXPANSION FACTOR	L0a	GSetCharExpan
SET CHARACTER HEIGHT	Lma	GSetCharHeight
SET CHARACTER SPACING	L0a	GSetCharSpacing
SET CHARACTER UP VECTOR	Lma	GSetCharUpVector
SET CHOICE MODE	Lmb	GSetChoiceMode
SET CLIPPING INDICATOR	Lma	GSetClip
SET COLOUR REPRESENTATION	Lma	GSetColrRep
SET DEFERRAL STATE	L1a	GSetDeferSt
SET DETECTABILITY	L1b	GSetDet
SET FILL AREA COLOUR INDEX	Lma	GSetFillColrInd
SET FILL AREA INDEX	L0a	GSetFillInd
SET FILL AREA INTERIOR STYLE	Lma	GSetFillIntStyle
SET FILL AREA REPRESENTATION	L1a	GSetFillRep
SET FILL AREA STYLE INDEX	L0a	GSetFillStyleInd
SET HIGHLIGHTING	L1a	GSetHighlight
SET POLYLINE COLOUR INDEX	Lma	GSetLineColrInd
SET LINETYPE	Lma	GSetLineType
SET LINEWIDTH SCALE FACTOR	L0a	GSetLineWidthScale
SET LOCATOR MODE	Lmb	GSetLocatorMode
SET POLYMARKER COLOUR INDEX	Lma	GSetMarkerColrInd
SET MARKER SIZE SCALE FACTOR	L0a	GSetMarkerSizeScale
SET MARKER TYPE	Lma	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 SEGMENT PRIORITY	L1a	GSetSegPriority
SET SEGMENT TRANSFORMATION	L1a	GSetSegTran

Table 2 - GKS Function Names and Pascal Names Ordered by Pascal Name

GKS Function Name	Level	Pascal Name
SET STRING MODE	Lmb	GSetStringMode
SET STROKE MODE	Lmb	GSetStrokeMode
SET TEXT ALIGNMENT	Lma	GSetTextAlign
SET TEXT COLOUR INDEX	Lma	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	Lmb	GSetValuatorMode
SET VIEWPORT	Lma	GSetViewport
SET VIEWPORT INPUT PRIORITY	Lmb	GSetViewportPriority
SET VISIBILITY	L1a	GSetVis
SET WINDOW	Lma	GSetWindow
SET WORKSTATION VIEWPORT	Lma	GSetWsViewport
SET WORKSTATION WINDOW	Lma	GSetWsWindow
TEXT	Lma	GText
UPDATE WORKSTATION	Lma	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	Lma	GActivateWs
ASSOCIATE SEGMENT WITH WORKSTATION	L2a	GAssocSegWs
AWAIT EVENT	Lmc	GAwaitEvent
CELL ARRAY	L0a	GCellArray
CLEAR WORKSTATION	Lma	GClearWs
CLOSE GKS	Lma	GCloseGKS
CLOSE SEGMENT	L1a	GCloseSeg
CLOSE WORKSTATION	Lma	GCloseWs
COPY SEGMENT TO WORKSTATION	L2a	GCopySegWs
CREATE SEGMENT	L1a	GCreateSeg
DEACTIVATE WORKSTATION	Lma	GDeactivateWs
DELETE SEGMENT	L1a	GDelSeg
DELETE SEGMENT FROM WORKSTATION	L1a	GDelScgWs
EMERGENCY CLOSE GKS	L0a	GEmergencyCloseGKS
ERROR HANDLING	L0a	GErrorHandling
ERROR LOGGING	L0a	GErrorLogging
ESCAPE	Lma	GEscape
ESCAPE	Lma	GEscapeGeneralized
EVALUATE TRANSFORMATION MATRIX	L1a	GEvalTran
FILL AREA	Lma	GFill
FLUSH DEVICE EVENTS	Lmc	GFlushDeviceEvents
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDP
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDPGeneralized
GET CHOICE	Lmc	GGetChoice
GET ITEM TYPE FROM GKSM	L0a	GGetItemType
GET LOCATOR	Lmc	GGetLocator
GET PICK	L1c	GGetPick
GET STRING	Lmc	GGetString
GET STROKE	Lmc	GGetStroke
GET VALUATOR	Lmc	GGetValuator
INITIALISE CHOICE	Lmb	GInitChoice
INITIALISE LOCATOR	Lmb	GInitLocator
INITIALISE PICK	L1b	GInitPick
INITIALISE STRING	Lmb	GInitString
INITIALISE STROKE	Lmb	GInitStroke
INITIALISE VALUATOR	Lmb	GInitValuator
INQUIRE CHOICE DEVICE STATE	Lmb	GInqChoiceDeviceSt
INQUIRE CLIPPING	Lma	GInqClip
INQUIRE COLOUR FACILITIES	Lma	GInqColrFacil
INQUIRE COLOUR REPRESENTATION	Lma	GInqColrRep
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	Lma	GInqCurIndivAttr
INQUIRE ASPECT SOURCE FLAGS	Lma	GInqASF
INQUIRE CHARACTER EXPANSION FACTOR	Lma	GInqCharExpan
INQUIRE CHARACTER SPACING	Lma	GInqCharSpacing
INQUIRE FILL AREA COLOUR INDEX	Lma	GInqFillColrInd
INQUIRE FILL AREA INTERIOR STYLE	Lma	GInqFillIntStyle
INQUIRE FILL AREA STYLE INDEX	Lma	GInqFillStyleInd
INQUIRE LINETYPE	Lma	GInqLineType

Table 3 - GKS Function Names and Pascal Names Ordered by GKS Function Name

GKS Function Name	Level	Pascal Name
INQUIRE LINEWIDTH SCALE FACTOR	Lma	GInqLineWidthScale
INQUIRE POLYLINE COLOUR INDEX	Lma	GInqLineColrInd
INQUIRE POLYMARKER COLOUR INDEX	Lma	GInqMarkerColrInd
INQUIRE POLYMARKER SIZE SCALE FACTOR	Lma	GInqMarkerSizeScale
INQUIRE POLYMARKER TYPE	Lma	GInqMarkerType
INQUIRE TEXT COLOUR INDEX	Lma	GInqTextColrInd
INQUIRE TEXT FONT AND PRECISION	Lma	GInqTextFontPrec
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	Lma	GInqCurNormTranNum
INQUIRE CURRENT PICK IDENTIFIER	L1b	GInqCurPickId
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	Lma	GInqCurPrimAttr
INQUIRE CHARACTER BASE VECTOR	Lma	GInqCharBaseVector
INQUIRE CHARACTER HEIGHT	Lma	GInqCharHeight
INQUIRE CHARACTER UP VECTOR	Lma	GInqCharUpVector
INQUIRE CHARACTER WIDTH	Lma	GInqCharWidth
INQUIRE FILL AREA INDEX	Lma	GInqFillInd
INQUIRE PATTERN REFERENCE POINT	Lma	GInqPatternRefPoint
INQUIRE PATTERN SIZE	Lma	GInqPatternSize
INQUIRE POLYLINE INDEX	Lma	GInqLineInd
INQUIRE POLYMARKER INDEX	Lma	GInqMarkerInd
INQUIRE TEXT ALIGNMENT	Lma	GInqTextAlign
INQUIRE TEXT INDEX	Lma	GInqTextInd
INQUIRE TEXT PATH	Lma	GInqTextPath
INQUIRE DEFAULT CHOICE DEVICE DATA	Lmb	GInqDefChoiceDeviceData
INQUIRE DEFAULT DEFERRAL STATE VALUES	L1a	GInqDefDeferSt
INQUIRE DEFAULT LOCATOR DEVICE DATA	Lmb	GInqDefLocatorDeviceData
INQUIRE DEFAULT PICK DEVICE DATA	L1b	GInqDefPickDeviceData
INQUIRE DEFAULT STRING DEVICE DATA	Lmb	GInqDefStringDeviceData
INQUIRE DEFAULT STROKE DEVICE DATA	Lmb	GInqDefStrokeDeviceData
INQUIRE DEFAULT VALUATOR DEVICE DATA	Lmb	GInqDefValuatorDeviceData
INQUIRE DISPLAY SPACE SIZE	Lma	GInqDisplaySize
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES	L1a	GInqDynModSegAttr
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES	L1a	GInqDynModWsAttr
INQUIRE FILL AREA FACILITIES	Lma	GInqFillFacil
INQUIRE FILL AREA REPRESENTATION	L1a	GInqFillRep
INQUIRE GENERALIZED DRAWING PRIMITIVE	L0a	GInqGDP
INQUIRE INPUT QUEUE OVERFLOW	Lmc	GInqInputOverflow
INQUIRE LEVEL OF GKS	Lma	GInqLevelGKS
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	L0a	GInqListGDP
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	L0a	GInqListWsTypes
INQUIRE LIST OF COLOUR INDICES	Lma	GInqListColrInd
INQUIRE LIST OF FILL AREA INDICES	L1a	GInqListFillInd
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 TEXT INDICES	L1a	GInqListTextInd
INQUIRE LOCATOR DEVICE STATE	Lmb	GInqLocatorDeviceSt
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	Lma	GInqMaxWsSt
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	L0a	GInqMaxNormTranNum

Table 3 - GKS Function Names and Pascal Names Ordered by GKS Function Name

GKS Function Name	Level	Pascal Name
INQUIRE MORE SIMULTANEOUS EVENTS	Lmc	GInqMoreEvents
INQUIRE NAME OF OPEN SEGMENT	L1a	GInqOpenSeg
INQUIRE NORMALIZATION TRANSFORMATION	Lma	GInqNormTran
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	Lmb	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	GInqPickDeviceSt
INQUIRE PIXEL	L0a	GInqPixel
INQUIRE PIXEL ARRAY	L0a	GInqPixelArray
INQUIRE PIXEL ARRAY DIMENSIONS	L0a	GInqPixelArrayDim
INQUIRE POLYLINE FACILITIES	Lma	GInqPolylineFacil
INQUIRE POLYLINE REPRESENTATION	L1a	GInqPolylineRep
INQUIRE POLYMARKER FACILITIES	Lma	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 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	Lmb	GInqStringDeviceSt
INQUIRE STROKE DEVICE STATE	Lmb	GInqStrokeDeviceSt
INQUIRE TEXT EXTENT	Lma	GInqTextExtent
INQUIRE TEXT FACILITIES	Lma	GInqTextFacil
INQUIRE TEXT REPRESENTATION	L1a	GInqTextRep
INQUIRE VALUATOR DEVICE STATE	Lmb	GInqValuatorDeviceSt
INQUIRE WORKSTATION CATEGORY	L0a	GInqWsCategory
INQUIRE WORKSTATION CLASSIFICATION	L0a	GInqWsClass
INQUIRE WORKSTATION CONNECTION AND TYPE	Lma	GInqWsConnType
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	L0a	GInqWsDeferUpdSt
INQUIRE WORKSTATION MAXIMUM NUMBERS	L1a	GInqWsMaxNum
INQUIRE WORKSTATION STATE	L0a	GInqWsSt
INQUIRE WORKSTATION TRANSFORMATION	Lma	GInqWsTran
INSERT SEGMENT	L2a	GInsertSeg
INTERPRET ITEM	L0a	GInterpretItem
MESSAGE	L1a	GMessage
OPEN GKS	Lma	GOpenGKS
OPEN WORKSTATION	Lma	GOpenWs
POLYLINE	Lma	GPolyline
POLYMARKER	Lma	GPolymarker
READ ITEM FROM GKSM	L0a	GReadItem

Table 3 - GKS Function Names and Pascal Names Ordered by GKS Function Name

GKS Function Name	Level	Pascal Name
REDRAW ALL SEGMENTS ON WORKSTATION	L1a	GRedrawSegWs
RENAME SEGMENT	L1a	GRenameSeg
REQUEST CHOICE	Lmb	GReqChoice
REQUEST LOCATOR	Lmb	GReqLocator
REQUEST PICK	L1b	GReqPick
REQUEST STRING	Lmb	GReqString
REQUEST STROKE	Lmb	GReqStroke
REQUEST VALUATOR	Lmb	GReqValuator
SAMPLE CHOICE	Lmc	GSampleChoice
SAMPLE LOCATOR	Lmc	GSampleLocator
SAMPLE PICK	L1c	GSamplePick
SAMPLE STRING	Lmc	GSampleString
SAMPLE STROKE	Lmc	GSampleStroke
SAMPLE VALUATOR	Lmc	GSampleValuator
SELECT NORMALIZATION TRANSFORMATION	Lma	GSelectNormTran
SET ASPECT SOURCE FLAGS	L0a	GSetASF
SET CHARACTER EXPANSION FACTOR	L0a	GSetCharExpan
SET CHARACTER HEIGHT	Lma	GSetCharHeight
SET CHARACTER SPACING	L0a	GSetCharSpacing
SET CHARACTER UP VECTOR	Lma	GSetCharUpVector
SET CHOICE MODE	Lmb	GSetChoiceMode
SET CLIPPING INDICATOR	Lma	GSetClip
SET COLOUR REPRESENTATION	Lma	GSetColrRep
SET DEFERRAL STATE	L1a	GSetDeferSt
SET DETECTABILITY	L1b	GSetDet
SET FILL AREA COLOUR INDEX	Lma	GSetFillColrInd
SET FILL AREA INDEX	L0a	GSetFillInd
SET FILL AREA INTERIOR STYLE	Lma	GSetFillIntStyle
SET FILL AREA REPRESENTATION	L1a	GSetFillRep
SET FILL AREA STYLE INDEX	L0a	GSetFillStyleInd
SET HIGHLIGHTING	L1a	GSetHighlight
SET LINETYPE	Lma	GSetLineType
SET LINEWIDTH SCALE FACTOR	L0a	GSetLineWidthScale
SET LOCATOR MODE	Lmb	GSetLocatorMode
SET MARKER SIZE SCALE FACTOR	L0a	GSetMarkerSizeScale
SET MARKER TYPE	Lma	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 COLOUR INDEX	Lma	GSetLineColrInd
SET POLYLINE INDEX	L0a	GSetPolylineInd
SET POLYLINE REPRESENTATION	L1a	GSetPolylineRep
SET POLYMARKER COLOUR INDEX	Lma	GSetMarkerColrInd
SET POLYMARKER INDEX	L0a	GSetPolymarkerInd
SET POLYMARKER REPRESENTATION	L1a	GSetPolymarkerRep
SET SEGMENT PRIORITY	L1a	GSetSegPriority
SET SEGMENT TRANSFORMATION	L1a	GSetSegTran

Table 3 - GKS Function Names and Pascal Names Ordered by GKS Function Name

GKS Function Name	Level	Pascal Name
SET STRING MODE	Lmb	GSetStringMode
SET STROKE MODE	Lmb	GSetStrokeMode
SET TEXT ALIGNMENT	Lma	GSetTextAlign
SET TEXT COLOUR INDEX	Lma	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	Lmb	GSetValuatorMode
SET VIEWPORT	Lma	GSetViewport
SET VIEWPORT INPUT PRIORITY	Lmb	GSetViewportPriority
SET VISIBILITY	L1a	GSetVis
SET WINDOW	Lma	GSetWindow
SET WORKSTATION VIEWPORT	Lma	GSetWsViewport
SET WORKSTATION WINDOW	Lma	GSetWsWindow
TEXT	Lma	GText
UPDATE WORKSTATION	Lma	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	Lma	GActivateWs
CLEAR WORKSTATION	Lma	GClearWs
CLOSE GKS	Lma	GCloseGKS
CLOSE WORKSTATION	Lma	GCloseWs
DEACTIVATE WORKSTATION	Lma	GDeactivateWs
ESCAPE	Lma	GEscape
ESCAPE	Lma	GEscapeGeneralized
FILL AREA	Lma	GFill
INQUIRE ASPECT SOURCE FLAGS	Lma	GInqASF
INQUIRE CHARACTER BASE VECTOR	Lma	GInqCharBaseVector
INQUIRE CHARACTER EXPANSION FACTOR	Lma	GInqCharExpan
INQUIRE CHARACTER HEIGHT	Lma	GInqCharHeight
INQUIRE CHARACTER SPACING	Lma	GInqCharSpacing
INQUIRE CHARACTER UP VECTOR	Lma	GInqCharUpVector
INQUIRE CHARACTER WIDTH	Lma	GInqCharWidth
INQUIRE CLIPPING	Lma	GInqClip
INQUIRE COLOUR FACILITIES	Lma	GInqColrFacil
INQUIRE COLOUR REPRESENTATION	Lma	GInqColrRep
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES	Lma	GInqCurIndivAttr
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER	Lma	GInqCurNormTranNum
INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES	Lma	GInqCurPrimAttr
INQUIRE DISPLAY SPACE SIZE	Lma	GInqDisplaySize
INQUIRE FILL AREA COLOUR INDEX	Lma	GInqFillColrInd
INQUIRE FILL AREA FACILITIES	Lma	GInqFillFacil
INQUIRE FILL AREA INDEX	Lma	GInqFillInd
INQUIRE FILL AREA INTERIOR STYLE	Lma	GInqFillIntStyle
INQUIRE FILL AREA STYLE INDEX	Lma	GInqFillStyleInd
INQUIRE LEVEL OF GKS	Lma	GInqLevelGKS
INQUIRE LINETYPE	Lma	GInqLineType
INQUIRE LINEWIDTH SCALE FACTOR	Lma	GInqLineWidthScale
INQUIRE LIST OF COLOUR INDICES	Lma	GInqListColrInd
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES	Lma	GInqMaxWsSt
INQUIRE NORMALIZATION TRANSFORMATION	Lma	GInqNormTran
INQUIRE PATTERN REFERENCE POINT	Lma	GInqPatternRefPoint
INQUIRE PATTERN SIZE	Lma	GInqPatternSize
INQUIRE POLYLINE COLOUR INDEX	Lma	GInqLineColrInd
INQUIRE POLYLINE FACILITIES	Lma	GInqPolylineFacil
INQUIRE POLYLINE INDEX	Lma	GInqLineInd
INQUIRE POLYMARKER COLOUR INDEX	Lma	GInqMarkerColrInd
INQUIRE POLYMARKER FACILITIES	Lma	GInqPolymarkerFacil
INQUIRE POLYMARKER INDEX	Lma	GInqMarkerInd
INQUIRE POLYMARKER SIZE SCALE FACTOR	Lma	GInqMarkerSizeScale
INQUIRE POLYMARKER TYPE	Lma	GInqMarkerType
INQUIRE TEXT ALIGNMENT	Lma	GInqTextAlign
INQUIRE TEXT COLOUR INDEX	Lma	GInqTextColrInd
INQUIRE TEXT EXTENT	Lma	GInqTextExtent
INQUIRE TEXT FACILITIES	Lma	GInqTextFacil
INQUIRE TEXT FONT AND PRECISION	Lma	GInqTextFontPrec
INQUIRE TEXT INDEX	Lma	GInqTextInd

Table 4 - GKS Function Names and Pascal Names Ordered by Level

GKS Function Name	Level	Pascal Name
INQUIRE TEXT PATH	Lma	GInqTextPath
INQUIRE WORKSTATION CONNECTION AND TYPE	Lma	GInqWsConnType
INQUIRE WORKSTATION TRANSFORMATION	Lma	GInqWsTran
OPEN GKS	Lma	GOpenGKS
OPEN WORKSTATION	Lma	GOpenWs
POLYLINE	Lma	GPolyline
POLYMARKER	Lma	GPolymarker
SELECT NORMALIZATION TRANSFORMATION	Lma	GSelectNormTran
SET CHARACTER HEIGHT	Lma	GSetCharHeight
SET CHARACTER UP VECTOR	Lma	GSetCharUpVector
SET CLIPPING INDICATOR	Lma	GSetClip
SET COLOUR REPRESENTATION	Lma	GSetColrRep
SET FILL AREA COLOUR INDEX	Lma	GSetFillColrInd
SET FILL AREA INTERIOR STYLE	Lma	GSetFillIntStyle
SET LINETYPE	Lma	GSetLineType
SET MARKER TYPE	Lma	GSetMarkerType
SET POLYLINE COLOUR INDEX	Lma	GSetLineColrInd
SET POLYMARKER COLOUR INDEX	Lma	GSetMarkerColrInd
SET TEXT ALIGNMENT	Lma	GSetTextAlign
SET TEXT COLOUR INDEX	Lma	GSetTextColrInd
SET VIEWPORT	Lma	GSetViewport
SET WINDOW	Lma	GSetWindow
SET WORKSTATION VIEWPORT	Lma	GSetWsViewport
SET WORKSTATION WINDOW	Lma	GSetWsWindow
TEXT	Lma	GText
UPDATE WORKSTATION	Lma	GUpdWs
INITIALISE CHOICE	Lmb	GInitChoice
INITIALISE LOCATOR	Lmb	GInitLocator
INITIALISE STRING	Lmb	GInitString
INITIALISE STROKE	Lmb	GInitStroke
INITIALISE VALUATOR	Lmb	GInitValuator
INQUIRE CHOICE DEVICE STATE	Lmb	GInqChoiceDeviceSt
INQUIRE DEFAULT CHOICE DEVICE DATA	Lmb	GInqDefChoiceDeviceData
INQUIRE DEFAULT LOCATOR DEVICE DATA	Lmb	GInqDefLocatorDeviceData
INQUIRE DEFAULT STRING DEVICE DATA	Lmb	GInqDefStringDeviceData
INQUIRE DEFAULT STROKE DEVICE DATA	Lmb	GInqDefStrokeDeviceData
INQUIRE DEFAULT VALUATOR DEVICE DATA	Lmb	GInqDefValuatorDeviceData
INQUIRE LOCATOR DEVICE STATE	Lmb	GInqLocatorDeviceSt
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES	Lmb	GInqNumInputDevices
INQUIRE STRING DEVICE STATE	Lmb	GInqStringDeviceSt
INQUIRE STROKE DEVICE STATE	Lmb	GInqStrokeDeviceSt
INQUIRE VALUATOR DEVICE STATE	Lmb	GInqValuatorDeviceSt
REQUEST CHOICE	Lmb	GReqChoice
REQUEST LOCATOR	Lmb	GReqLocator
REQUEST STRING	Lmb	GReqString
REQUEST STROKE	Lmb	GReqStroke
REQUEST VALUATOR	Lmb	GReqValuator
SET CHOICE MODE	Lmb	GSetChoiceMode
SET LOCATOR MODE	Lmb	GSetLocatorMode

Table 4 - GKS Function Names and Pascal Names Ordered by Level

GKS Function Name	Level	Pascal Name
SET STRING MODE	Lmb	GSetStringMode
SET STROKE MODE	Lmb	GSetStrokeMode
SET VALUATOR MODE	Lmb	GSetValuatorMode
SET VIEWPORT INPUT PRIORITY	Lmb	GSetViewportPriority
AWAIT EVENT	Lmc	GAwaitEvent
FLUSH DEVICE EVENTS	Lmc	GFlushDeviceEvents
GET CHOICE	Lmc	GGetChoice
GET LOCATOR	Lmc	GGetLocator
GET STRING	Lmc	GGetString
GET STROKE	Lmc	GGetStroke
GET VALUATOR	Lmc	GGetValuator
INQUIRE INPUT QUEUE OVERFLOW	Lmc	GInqInputOverflow
INQUIRE MORE SIMULTANEOUS EVENTS	Lmc	GInqMoreEvents
SAMPLE CHOICE	Lmc	GSampleChoice
SAMPLE LOCATOR	Lmc	GSampleLocator
SAMPLE STRING	Lmc	GSampleString
SAMPLE STROKE	Lmc	GSampleStroke
SAMPLE VALUATOR	Lmc	GSampleValuator
CELL ARRAY	L0a	GCellArray
EMERGENCY CLOSE GKS	L0a	GEmergencyCloseGKS
ERROR HANDLING	L0a	GErrorHandling
ERROR LOGGING	L0a	GErrorLogging
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDP
GENERALIZED DRAWING PRIMITIVE (GDP)	L0a	GGDPGeneralized
GET ITEM TYPE FROM GKSM	L0a	GGetItemType
INQUIRE GENERALIZED DRAWING PRIMITIVE	L0a	GInqGDP
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES	L0a	GInqListGDP
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES	L0a	GInqListWsTypes
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS	L0a	GInqListNormTranNum
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER	L0a	GInqMaxNormTranNum
INQUIRE OPERATING STATE VALUE	L0a	GInqOpSt
INQUIRE PATTERN FACILITIES	L0a	GInqPatternFacil
INQUIRE PIXEL	L0a	GInqPixel
INQUIRE PIXEL ARRAY	L0a	GInqPixelArray
INQUIRE PIXEL ARRAY DIMENSIONS	L0a	GInqPixelArrayDim
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 TEXT REPRESENTATION	L0a	GInqPredTextRep
INQUIRE SET OF OPEN WORKSTATIONS	L0a	GInqOpenWs
INQUIRE WORKSTATION CATEGORY	L0a	GInqWsCategory
INQUIRE WORKSTATION CLASSIFICATION	L0a	GInqWsClass
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES	L0a	GInqWsDeferUpdSt
INQUIRE WORKSTATION STATE	L0a	GInqWsSt
INTERPRET ITEM	L0a	GInterpretItem
READ ITEM FROM GKSM	L0a	GReadItem
SET ASPECT SOURCE FLAGS	L0a	GSetASF

Table 4 - GKS Function Names and Pascal Names Ordered by Level

GKS Function Name	Level	Pascal Name
SET CHARACTER EXPANSION FACTOR	L0a	GSetCharExpan
SET CHARACTER SPACING	L0a	GSetCharSpacing
SET FILL AREA INDEX	L0a	GSetFillInd
SET FILL AREA STYLE INDEX	L0a	GSetFillStyleInd
SET LINEWIDTH SCALE FACTOR	L0a	GSetLineWidthScale
SET MARKER SIZE SCALE FACTOR	L0a	GSetMarkerSizeScale
SET PATTERN REFERENCE POINT	L0a	GSetPatternRefPoint
SET PATTERN SIZE	L0a	GSetPatternSize
SET POLYLINE INDEX	L0a	GSetPolylineInd
SET POLYMARKER INDEX	L0a	GSetPolymarkerInd
SET TEXT FONT AND PRECISION	L0a	GSetTextFontPrec
SET TEXT INDEX	L0a	GSetTextInd
SET TEXT PATH	L0a	GSetTextPath
WRITE ITEM TO GKSM	L0a	GWriteItem
WRITE ITEM TO GKSM	L0a	GWriteItemGeneralized
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 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 TEXT INDICES	L1a	GInqListTextInd
INQUIRE NAME OF OPEN SEGMENT	L1a	GInqOpenSeg
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED	L1a	GInqNumSegPriorities
INQUIRE PATTERN REPRESENTATION	L1a	GInqPatternRep
INQUIRE POLYLINE REPRESENTATION	L1a	GInqPolylineRep
INQUIRE POLYMARKER REPRESENTATION	L1a	GInqPolymarkerRep
INQUIRE SEGMENT ATTRIBUTES	L1a	GInqSegAttr
INQUIRE SET OF ACTIVE WORKSTATIONS	L1a	GInqActiveWs
INQUIRE SET OF ASSOCIATED WORKSTATIONS	L1a	GInqAssocWs
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

Table 4 - GKS Function Names and Pascal Names Ordered by Level

GKS Function Name	Level	Pascal Name
SET POLYMARKER REPRESENTATION	L1a	GSetPolymarkerRep
SET SEGMENT PRIORITY	L1a	GSetSegPriority
SET SEGMENT TRANSFORMATION	L1a	GSetSegTran
SET TEXT REPRESENTATION	L1a	GSetTextRep
SET VISIBILITY	L1a	GSetVis
INITIALISE PICK	L1b	GInitPiek
INQUIRE CURRENT PICK IDENTIFIER	L1b	GInqCurPickId
INQUIRE DEFAULT PICK DEVICE DATA	L1b	GInqDefPickDeviceData
INQUIRE PICK DEVICE STATE	L1b	GInqPiekDeviceSt
REQUEST PICK	L1b	GReqPick
SET DETECTABILITY	L1b	GSetDet
SET PICK IDENTIFIER	L1b	GSetPiekId
SET PICK MODE	L1b	GSetPiekMode
GET PICK	L1c	GGetPick
SAMPLE PICK	L1e	GSamplePick
ASSOCIATE SEGMENT WITH WORKSTATION	L2a	GAssocSegWs
COPY SEGMENT TO WORKSTATION	L2a	GCopySegWs
INSERT SEGMENT	L2a	GInsertSeg

3.3 Representation of GKS Data Types

Clause 6 of ANSI X3.124-1985 defines GKS data structures which are an integral part of the system. These data structures are bound as the following Pascal data structures.

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

(2) 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).

(3) String is represented as a fixed length type (GAString).

(4) A GKS point is mapped to the type GRPoint in Pascal, which is a record with x,y(: real) fields. A fixed length array of GRPoint (GAPointArray) is used in procedure definitions.

(5) 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.

(6) The GKS data records are represented as Pascal records.

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

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

3.4 Naming Conventions for Data Types

A consistent naming scheme for data types is used.

(1) Constants are prefixed with "GC".

(2) Scalar types are prefixed with "GT".

(3) Enumerated types are prefixed with "GE".

(4) Values of enumerated types are prefixed with "GV".

(5) Arrays are prefixed with "GA".

(6) Records are prefixed with "GR".

(7) Sets are prefixed with "GS".

Mixtures of upper- and lowercase letters are used extensively in the type and procedure names. This is to aid readability and is not otherwise significant. International spelling is used in all instances where there is a conflict with U.S. spelling.

3.5 Implementation-Dependent Characteristics

There are a number of implementation-defined characteristics necessary in a Pascal interface. Resolutions of such implementation-dependent features should be detailed in the documentation of a Pascal implementation.

(1) 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.

(2) 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 should describe the meaning and use of the string.

(3) There are a number of implementation-defined constants, which are shown in 5.1, and implementation-defined types, which are shown in 5.2.

(4) 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 3.6.

(5) In ANSI/IEEE 770X3.97-1983, separate compilation is not prohibited, but a standard manner for achieving this compilation is not given. Separate compilation is crucial for GKS, since an implementation is not feasible unless the underlying Pascal supports separate compilation. The implementation documentation should 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.

(6) 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 should describe the meaning and behaviour associated with these default values.

3.6 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.

3.7 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:

(1) 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.

(2) 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.

(3) 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.

3.8 Registration [1)]

In the GKS standard, ANSI X3.124-1985, certain value ranges are reserved 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 ANSI X3.124.

1) For the purpose of this 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), A-266 Technology Building, Gaithersburg, MD 20899, USA to act as registration authority. Information concerning the Registration Authority and its procedures may be obtained on request from the National Bureau of Standards quoting ISO 8651-2:1988, the number of the related International Standard.

4. Error Handling

4.1 The Error Handling Function

Error handling shall be carried out as described in subclause 4.12 of ANSI X3.124-1985. 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 Tables 2, 3 or 4.

4.2 Pascal-Specific GKS Errors

Certain features of the Pascal language make additional errors (beyond the ones described in ANSI X3.124-1985) possible. Specifically, these new errors are defined:

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.

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}
GCMaxChoicePicks    = 10;     {Maximum number of pick ids for choice data record}
GCMaxChoicePrompts  = 10;     {Maximum number of prompts for choice data record}
GCMaxChoiceStrings  = 10;     {Maximum number of strings for choice data record}
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..5;   {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..5;   {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..4;   {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 3.6. GRFileData is not a registered data record.

```
GRChoiceData        = record
                        case Prompt      : GTChoiceDataTag of
                              1          : ( );{Implementation-defined}
                              2          : (R0002NumAlternatives : GTMaxChoicePrompt;
                                           R0002AltPrompts       : GAPrompt);
                              3          : (R0003NumStrings      : GTMaxChoiceStrings;
                                           R0003ChoiceStrings    : GAStringList);
                              4          : (R0004NumStrings      : GTMaxChoiceStrings;
                                           R0004ChoiceStrings    : GAStringList);
                              5          : (R0005Segment         : GTSeg;
                                           R0005NumChoices       : GTMaxChoicePicks;
                                           R0005PickIds          : GAPickId);
                        {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}
                        2     : ( );{Implementation-defined}
                        3     : ( );{Implementation-defined}
                        4     : (R0004LAttrControl     : GEAttrControl;
                                 R0004LineAttributes   : GRLineAttr);
                        5     : (R0005RAttrControl     : GEAttrControl;
                                 case LineFillControl  : GELineFillControl of
                                    GVLineControl      :
                                    (R0005RectAttributes: GRLineAttr);
                                    GVFillControl      :
                                    (R0005FillAttributes : GRFillAttr));
                  {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
                  InitialPos  : GTInt1;
                  XInterval,
                  YInterval,
                  TimeInterval : REAL;
                  case Prompt : GTStrokeDataTag of
                        1     : ( );{Implementation-defined}
                        2     : ( );{Implementation-defined}
                        3     : (R0003MAttrControl     : GEAttrControl;
                                 R0003MarkerAttributes : GRMarkerAttr);
                        4     : (R0004LAttrControl     : GEAttrControl;
                                 R0004LineAttributes   : GRLineAttr);
                  {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;
GTMaxChoicePicks    = 1..GCMaxChoicePicks;
GTMaxChoicePrompts  = 1..GCMaxChoicePrompts;
GTMaxChoiceStrings  = 1..GCMaxChoiceStrings;
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);
GEAttrControl       = (GVCurrent,GVSpecified);

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            = (GVLma,GVLmb,GVLmc,
                        GVL0a,GVL0b,GVL0c,
                        GVL1a,GVL1b,GVL1c,
                        GVL2a,GVL2b,GVL2c);
GELineFillControl  = (GVLineControl,GVFillControl);

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);
GEPrompt           = (GVOff,GVOn);

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;

GAPickId         = array[GTMaxChoicePicks] of GTPickId;
GAPointArray     = array[GTMaxPoint1] of GRPoint;
GAPrim           = array[GEPrim] of INTEGER;
GAPrimRep        = array[GEPrim] of GRPrimRep;
GAProcName       = packed array[1..GCMaxName] of CHAR;
GAPrompt         = array[GTMaxChoicePrompts] of GEPrompt;

GASeg            = array[GTInqSize] of GTSeg;
GASegMod         = array[GESegAttr] of GEDynMod;
GAString         = packed array[GTMaxString] of CHAR;
GAStringList     = array[GTMaxChoiceStrings] of GRString;

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;

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

Pascal GKS Data Structures Record Types

```
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;
```

```
GRFillAttr      = record
                    InteriorASF,
                    StyleASF,
                    ColourASF       : GEASF;
                    FillIndex       : GTInt1;
                    FillBundle      : GRFillRep
                  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;

GRIntVector     = record
                    xValue,
                    yValue          : INTEGER
                  end;

GRLineAttr      = record
                    TypeASF,
                    WidthASF,
                    ColourASF       : GEASF;
                    LineIndex       : GTInt1;
                    LineBundle      : GRLineRep
                  end;

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

Pascal GKS Data Structures Record Types

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

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

GRMarkerAttr    = record
                    TypeASF,
                    SizeASF,
                    ColourASF       : GEASF;
                    MarkerIndex     : GTInt1;
                    MarkerBundle    : GRMarkerRep
                  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;
```

Record Types **Pascal GKS Data Structures**

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

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 **Lma**

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

CLOSE GKS **Lma**

```
procedure GCloseGKS;
```

OPEN WORKSTATION **Lma**

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

CLOSE WORKSTATION **Lma**

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

ACTIVATE WORKSTATION **Lma**

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

DEACTIVATE WORKSTATION **Lma**

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

CLEAR WORKSTATION Lma

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

REDRAW ALL SEGMENTS ON WORKSTATION L1a

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

UPDATE WORKSTATION Lma

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

SET DEFERRAL STATE L1a

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

MESSAGE L1a

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

ESCAPE **Lma**

```
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.

```
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 Lma

```
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 Lma

```
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 Lma

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

FILL AREA Lma

```
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**

```
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**

```
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.

```
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

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

This procedure is a specific form of

SET <Primitive> INDEX

SET LINETYPE Lma

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

SET LINEWIDTH SCALE FACTOR L0a

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

SET POLYLINE COLOUR INDEX Lma

```
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 Lma

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

SET MARKER SIZE SCALE FACTOR **L0a**

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

SET POLYMARKER COLOUR INDEX **Lma**

```
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 **Lma**

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

SET CHARACTER HEIGHT Lma

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

SET CHARACTER UP VECTOR Lma

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

SET TEXT PATH L0a

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

SET TEXT ALIGNMENT Lma

```
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 Lma

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

SET FILL AREA STYLE INDEX L0a

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

Output Attributes — GKS Functions

SET FILL AREA COLOUR INDEX — **Lma**

```
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**

```
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

```
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 Lma

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

6.5 Transformation Functions

6.5.1 Normalization Transformation

SET WINDOW Lma

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

SET VIEWPORT Lma

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

SET VIEWPORT INPUT PRIORITY Lmb

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

SELECT NORMALIZATION TRANSFORMATION Lma

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

SET CLIPPING INDICATOR Lma

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

6.5.2 Workstation Transformation

SET WORKSTATION WINDOW Lma

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

SET WORKSTATION VIEWPORT Lma

```
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
        );
```

Segment Functions — GKS Functions

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 Lmb

```
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 **Lmb**

```
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 **Lmb**

```
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 Lmb

```
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 **Lmb**

```
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 **Lmb**

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

This procedure is a specific form of

SET <Input> MODE

SET STROKE MODE **Lmb**

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

This procedure is a specific form of

SET <Input> MODE

SET VALUATOR MODE Lmb

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

This procedure is a specific form of

SET <Input> MODE

SET CHOICE MODE Lmb

```
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 **Lmb**

```
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 **Lmb**

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

This procedure is a specific form of

REQUEST <Input>

REQUEST STROKE **Lmb**

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

This procedure is a specific form of

REQUEST <Input>

REQUEST VALUATOR Lmb

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

This procedure is a specific form of

REQUEST <Input>

REQUEST CHOICE Lmb

```
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 **Lmb**

```
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 **Lmc**

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

This procedure is a specific form of

SAMPLE <Input>

SAMPLE STROKE **Lmc**

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

This procedure is a specific form of

SAMPLE <Input>

SAMPLE VALUATOR **Lmc**

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

This procedure is a specific form of

SAMPLE <Input>

SAMPLE CHOICE Lmc

```
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 Lmc

```
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 Lmc

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

FLUSH DEVICE EVENTS **Lmc**

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

GET LOCATOR **Lmc**

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

This procedure is a specific form of

GET <Input>

GET STROKE **Lmc**

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

This procedure is a specific form of

GET <Input>

GET VALUATOR **Lmc**

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

This procedure is a specific form of

GET <Input>

GET CHOICE **Lmc**

```
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 Lmc

```
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.

```
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 Appendix 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 Lma

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

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES L0a

```
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 3.7.

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**

```
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 3.7.

Errors:
2102 List element or set member not available

INQUIRE SET OF ACTIVE WORKSTATIONS **L1a**

```
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 3.7.

Errors:
2102 List element or set member not available

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES **Lma**

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

INQUIRE POLYLINE INDEX Lma

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

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE POLYMARKER INDEX Lma

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

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE TEXT INDEX Lma

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

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE CHARACTER HEIGHT Lma

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

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE CHARACTER UP VECTOR **Lma**

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

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE CHARACTER WIDTH **Lma**

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

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE CHARACTER BASE VECTOR **Lma**

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

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE TEXT PATH **Lma**

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

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE TEXT ALIGNMENT Lma

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

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE FILL AREA INDEX Lma

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

This procedure is a specific form of

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES

INQUIRE PATTERN SIZE Lma

```
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 Lma

```
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 **Lma**

```
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 **Lma**

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

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE LINEWIDTH SCALE FACTOR **Lma**

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

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE POLYLINE COLOUR INDEX Lma

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

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE POLYMARKER TYPE Lma

```
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 Lma

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

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE POLYMARKER COLOUR INDEX Lma

```
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 **Lma**

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

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE CHARACTER EXPANSION FACTOR **Lma**

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

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE CHARACTER SPACING **Lma**

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

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE TEXT COLOUR INDEX **Lma**

```
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 Lma

```
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 Lma

```
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 Lma

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

This procedure is a specific form of

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES

INQUIRE ASPECT SOURCE FLAGS Lma

```
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 Lma

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

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS **L0a**

```
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 3.7.

Errors:
2102 List element or set member not available

INQUIRE NORMALIZATION TRANSFORMATION **Lma**

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

INQUIRE CLIPPING **Lma**

```
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

```
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 3.7.

Errors:
2102 List element or set member not available

INQUIRE MORE SIMULTANEOUS EVENTS Lmc

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

6.9.5 Inquiry Functions for Workstation State List

INQUIRE WORKSTATION CONNECTION AND TYPE Lma

```
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

```
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 3.7.

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

```
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 3.7.

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**

```
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 3.7.

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 **Lma**

```
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

```
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 3.7.

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

```
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 3.7.

Errors:
2102 List element or set member not available

INQUIRE PATTERN REPRESENTATION L1a

```
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 Lma

```
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 3.7.

Errors:
2102 List element or set member not available

INQUIRE COLOUR REPRESENTATION Lma

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

INQUIRE WORKSTATION TRANSFORMATION Lma

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

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION L1a

```
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 3.7.

Errors:
2102 List element or set member not available

INQUIRE LOCATOR DEVICE STATE Lmb

```
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 **Lmb**

```
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 **Lmb**

```
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 Lmb

```
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 **Lmb**

```
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 **Lma**

```
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 Lma

```
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 3.7.

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 **Lma**

```
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 3.7.

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 **Lma**

```
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 3.7.

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 Lma

```
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 3.7.

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

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

INQUIRE COLOUR FACILITIES Lma

```
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

```
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 3.7.

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 Lma

```
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 **Lmb**

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

INQUIRE DEFAULT LOCATOR DEVICE DATA **Lmb**

```
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 3.7.

Errors:
2102 List element or set member not available

INQUIRE DEFAULT STROKE DEVICE DATA Lmb

```
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 3.7.

Errors:
2102 List element or set member not available

INQUIRE DEFAULT VALUATOR DEVICE DATA Lmb

```
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 3.7.

Errors:
2102 List element or set member not available

INQUIRE DEFAULT CHOICE DEVICE DATA **Lmb**

```
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 3.7.

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 3.7.

Errors:
2102 List element or set member not available

INQUIRE DEFAULT STRING DEVICE DATA Lmb

```
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 3.7.

Errors:
2102 List element or set member not available

6.9.7 Inquiry Functions for Segment State List

INQUIRE SET OF ASSOCIATED WORKSTATIONS L1a

```
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 3.7.

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

```
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 Lmc

```
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
        );
```

Appendix A

Data Types in Compilation Order

(This Appendix is not part of ANSI X3.124.2-1988, but is included for information only.)

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

A1. Implementation-Defined Constants

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

A2. Required Constants

GCCircleMarker
GCCrossMarker
GCDashDotLine
GCDashedLine
GCDefMemory
GCDotMarker
GCDottedLine
GCPlusMarker
GCSolidLine
GCStarMarker

A3. Implementation-Defined Tag Types

GTChoiceDataTag
GTEscapeDataTag
GTGDPDataTag
GTLocatorDataTag
GTPickDataTag
GTStringDataTag
GTStrokeDataTag
GTValuatorDataTag

A4. Error Logging and Connection Files

GTErrorFileName
GAConnId

A5. General Types

GTInqSize
GTInt0
GTInt1
GTInt2
GTInt3
GTMaxChoicePicks
GTMaxChoicePrompts
GTMaxChoiceStrings
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

A6. Types Applicable to Workstation Control Procedures

GEControl
GEDefer
GEImplicitRegen
GEUpdRegen
GEWsSt

A7. Types Applicable to Transformation Procedures

GEClip
GEPriority

A8. Types Applicable to Attribute Setting Procedures

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

A9. Types Applicable to Segment Procedures

GECoordSwitch
GEDet
GEHighlight
GEVis
GAMatrix

A10. Types Applicable to Input Procedures

GEEcho
GEEvents
GEEventClass
GEMode
GEPrompt
GEReqStatus
GEInputClass
GEInputStatus
GAPrompt

A11. Types Applicable to GKS Description

GELevel
GRWsMaxNum

A12. Types Applicable to GKS State

GEOpSt
GRPrimAttr

A13. Types Applicable to Workstation State

GENfan
GEReturn
GESurface
GEWsTran
GRDeferUpd

A14. Types Applicable to Workstation Description

GEDeviceUnits
GEDisplay
GEDynMod
GESegAttr
GEWsCategory
GEWsClass
GAInputClass
GAPrim
GASegMod
GRDisplaySize
GRDynModWsAttr

A15. Types Applicable to Segment State

GRSegAttr

A16. GKS Data Records

GRChoiceData
GREscapeDataIn
GREscapeDataOut
GAEscapeInInt
GAEscapeInReal
GAEscapeInString

GKS Data Records

GAEscapeOutInt
GAEscapeOutReal
GAEscapeOutString
GRFileData
GRGDPData
GAGDPInt
GAGDPReal
GAGDPString
GAItemInt
GAItemReal
GAItemString
GRLocatorData
GRPickData
GRStringData
GRStrokeData
GRValuatorData

A17. Types Applicable to the One-One Procedures

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

A18. Types Applicable to the Many-One Procedures

GRInput
GRInputData
GRDefInput
GRPrimFacil

Appendix B

Metafile Item Types

(This Appendix is not part of ANSI X3.124.2-1988, but is included for information only.)

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 B1 - 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

Table B1 - Pascal Item Type Names

GKSM Item Type	Pascal Name
POLYLINE	GITPolyline
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

Appendix C

Conformant Arrays

(This Appendix is not part of ANSI X3.124.2-1988, but is included for information only.)

The Pascal language, as described in ANSI/IEEE 770X3.97-1983, does not define a method for passing arrays of varying size as actual parameters to procedures. However, many compilers have implemented a conformant array mechanism in conformance with ISO 7185-1982, Programming languages - Pascal. Since several procedures within the GKS binding to Pascal benefit from the use of conformant arrays, this appendix specifies a binding which may be used when conformant array capabilities are available.

C1. Conformant-Array-Specific GKS Errors

The use of conformant arrays in Pascal makes additional errors (beyond the ones described in clause 4) 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). This error is also defined in Appendix C.

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.

C2. Conformant Array Procedures

MESSAGE **L1a**

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

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

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

ESCAPE **Lma**

```
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 (≤ max1-min1+1) must specify the number of integers in the array IntDataIn. The value of the parameter NumRealIn (≤ max2-min2+1) must specify the number of reals in the array RealDataIn. The value of the parameter NumStringIn (≤ max3-min3+1) must specify the number of strings in the array StringDataIn. Although the arrays are variable parameters, the implementation must 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

POLYLINE **Lma**

```
procedure GPolyline(
        NumPoints         : GTInt2;
    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 must 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

POLYMARKER Lma

```
procedure GPolymarker(
        NumPoints       : GTInt1;
    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 must 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

TEXT Lma

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

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

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

FILL AREA Lma

```
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 must 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

CELL ARRAY L0a

```
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 (≤max1) and min2 to min2+Dy-1 (≤max2). Although the array is a variable parameter, the implementation must 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

GENERALIZED DRAWING PRIMITIVE (GDP) L0a

```
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) must specify the number of points in the array Points. Although the array and data record are variable parameters, the implementation must 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

```
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 (≤ max1-min1+1) must specify the number of points in the array Points. The value of the parameter NumInt (≤ max2-min2+1) must specify the number of integers in the array IntData. The value of the parameter NumReal (≤ max3-min3+1) must specify the number of reals in the array RealData. The value of the parameter NumString (≤ max4-min4+1) must specify the number of strings in the array StringData. Although the arrays are variable parameters, the implementation must 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

SET PATTERN REPRESENTATION **L1a**

```
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 must 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

WRITE ITEM TO GKSM **L0a**

```
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) must specify the number of integers in the array IntDataIn. The value of the parameter NumRealIn (≤ max2-min2+1) must specify the number of reals in the array RealDataIn. The value of the parameter NumStringIn (≤ max3-min3+1) must specify the number of strings in the array StringDataIn. Although the arrays are variable parameters, the implementation must 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

INQUIRE LIST OF AVAILABLE WORKSTATION TYPES **L0a**

```
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 must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE SET OF OPEN WORKSTATIONS L0a

```
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 must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE SET OF ACTIVE WORKSTATIONS L1a

```
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 must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS L0a

```
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 must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE SET OF SEGMENT NAMES IN USE **L1a**

```
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 must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE LIST OF POLYLINE INDICES **L1a**

```
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
        );
```

The parameter Size must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE LIST OF POLYMARKER INDICES L1a

```
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
        );
```

The parameter Size must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE LIST OF TEXT INDICES L1a

```
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
        );
```

The parameter Size must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE TEXT EXTENT **Lma**

```
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 must be ≤ max-min+1.

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

INQUIRE LIST OF FILL AREA INDICES **L1a**

```
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
        );
```

The parameter Size must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE LIST OF PATTERN INDICES L1a

```
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 must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE PATTERN REPRESENTATION L1a

```
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

INQUIRE LIST OF COLOUR INDICES **Lma**

```
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 must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

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

```
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 must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE PREDEFINED PATTERN REPRESENTATION L0a

```
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

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES L0a

```
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 must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE SET OF ASSOCIATED WORKSTATIONS **L1a**

```
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 must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE PIXEL ARRAY **L0a**

```
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

Appendix D

The Many-One Pascal Interface

(This Appendix is not part of ANSI X3.124.2-1988, but is included for information only.)

There may not be a strict one-to-one correspondence between GKS abstract functions and Pascal procedures. A method employing variant records may be 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:

(1) Output Primitive Representations

(a) The GKS words Polyline, Polymarker, Text, and Fill Area are replaced by "Prim" (which is the abbreviation for PRIMITIVE).

(b) The first parameter of the function is an enumerated type (GEPrim) which has one of the values GVPolyline, GVPolymarker, GVText, GVFillArea.

(2) Input Classes

(a) The GKS words Locator, Stroke, Valuator, Choice, Pick, and String are replaced by "Input".

(b) The first parameter of the function is an enumerated type (GEInputClass) which has one of the values GVLocator, GVStroke, GVValuator, GVChoice, GVPick, GVString.

This interface may exist in addition to the interface defined in clauses 3, 4, 5, and 6, but the many-one interface is not permitted to replace what is defined in clauses 3, 4, 5, and 6.

D1. Many-One-Interface-Specific GKS Errors

The use of the many-one interface in Pascal makes additional errors (beyond the ones described in clause 4) 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). This error is also defined in Appendix B.

2103 Pick is not supported in this level of GKS

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

D2. Additional Data Types

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

```
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;
```

D3. Output Attributes

D3.1 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 for the components of the type GEPrim, the GKS abstract functions are obtained.

D3.2 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 for the components of the type GEPrim, the GKS abstract functions are obtained.

D4. Input Functions

D4.1 Initialisation of Input Devices

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

```
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 must not change the value of the data record.

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

D4.2 Setting the Mode of Input Devices

SET LOCATOR MODE Lmb
SET STROKE MODE Lmb
SET VALUATOR MODE Lmb
SET CHOICE MODE Lmb
SET PICK MODE L1b
SET STRING MODE Lmb

```
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

D4.3 Request Input Functions

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

```
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

D4.4 Sample Input Functions

SAMPLE LOCATOR Lmc
SAMPLE STROKE Lmc
SAMPLE VALUATOR Lmc
SAMPLE CHOICE Lmc
SAMPLE PICK L1c
SAMPLE STRING Lmc

```
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

D4.5 Event Input Functions

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

```
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

D5. Inquiry Functions

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 for the components of the type GEPrim, the GKS abstract functions are obtained. The parameter Size must be ≤ max-min+1. The parameters Start, Size, and Done are defined in 3.7.

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 DefinedPrimInd      : GAInt
        );
```

This procedure is a general

INQUIRE LIST OF <Primitive> INDICES

By substituting for the components of the type GEPrim, the GKS abstract functions are obtained. The parameters Start, Size, and Done are defined in 3.7.

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

INQUIRE LOCATOR DEVICE STATE Lmb
INQUIRE STROKE DEVICE STATE Lmb
INQUIRE VALUATOR DEVICE STATE Lmb
INQUIRE CHOICE DEVICE STATE Lmb
INQUIRE PICK DEVICE STATE L1b
INQUIRE STRING DEVICE STATE Lmb

```
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 POLYLINE FACILITIES **Lma**
INQUIRE POLYMARKER FACILITIES **Lma**
INQUIRE TEXT FACILITIES **Lma**
INQUIRE FILL AREA FACILITIES **Lma**

```
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 for the components of the type GEPrim, the GKS abstract functions are obtained. The parameters Start and Done are defined in 3.7.

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

INQUIRE DEFAULT LOCATOR DEVICE DATA Lmb
INQUIRE DEFAULT STROKE DEVICE DATA Lmb
INQUIRE DEFAULT VALUATOR DEVICE DATA Lmb
INQUIRE DEFAULT CHOICE DEVICE DATA Lmb
INQUIRE DEFAULT PICK DEVICE DATA L1b
INQUIRE DEFAULT STRING DEVICE DATA Lmb

```
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 3.7.

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

Appendix E

Example Programs

(This Appendix is not part of ANSI X3.124.2-1988, but is included for information only.)

This Appendix gives example programs using the language binding defined in this part of ANSI X3.124-1985.

E1. Program STAR

```
{
     PROGRAM STAR

         DESCRIPTION:
           This program draws a yellow star on a blue background
           and writes the title 'Star' in green under the star.

         CONFORMANCE:
           GKS level: ma
           The implementation must support at least one
           workstation of category output or outin

}

PROGRAM STAR (output);

CONST

{ Implementation-specific method to define the
implementation's constants.  }
$INCLUDE 'gks.p1'$

{ User defined constants  }
    wsid     = 1;
    wstype   = 1;

TYPE

{ Implementation-specific method to define the GKS data types. }
$INCLUDE 'gks.p2'$

VAR
    align        : GRAlign;
    colr         : GRColr;
    conid        : GAConnId;
    fnam         : GTErrorFileName;
    length       : INTEGER;
    position     : GRPoint;
    starpoints   : GAPointArray;
```

```
    title       : GAString;
    winlimits   : GRBound;

{ Implementation-specific method to define the interface to
the GKS routines  }
$INCLUDE 'gks.p3'$

{************************************************************}

BEGIN
    { Open GKS and activate a workstation.      }

    fnam    := 1;
    conid   := '/dev/display';

    GOpenGKS ( fnam, GCDefMemory );
    GOpenWs ( wsid, conid, wstype );
    GActivateWs ( wsid );

    { Centre the window around the origin.      }

    WITH winlimits DO
        BEGIN
        LeftBound   := -1.25;
        RightBound  :=  1.25;
        LowerBound  := -1.25;
        UpperBound  :=  1.25;
        END; { with }

    GSetWindow ( 1, winlimits );
    GSelectNormTran ( 1 );

    { Define the colours to be used.            }

    WITH colr DO
        BEGIN
        Red     := 0.0;
        Green   := 0.0;
        Blue    := 1.0;
        END; { with }
    GSetColrRep ( wsid, 0, colr );

    WITH colr DO
        BEGIN
        Red     := 1.0;
        Green   := 1.0;
        Blue    := 0.0;
        END; { with }
    GSetColrRep ( wsid, 1, colr );

    WITH colr DO
        BEGIN
```

```
        Red     := 0.0;
        Green   := 1.0;
        Blue    := 0.0;
        END; { with }
     GSetColrRep ( wsid, 2, colr );

    { Fill the star with solid yellow.      }

    GSetFillIntStyle ( GVSolid );
    GSetFillColrInd ( 1 );

    { Draw the star.                        }

    starpoints[1].x :=  0.951057;
    starpoints[2].x := -0.951057;
    starpoints[3].x :=  0.587785;
    starpoints[4].x :=  0.0;
    starpoints[5].x := -0.587785;

    starpoints[1].y :=  0.309017;
    starpoints[2].y :=  0.309017;
    starpoints[3].y := -0.951057;
    starpoints[4].y :=  1.0;
    starpoints[5].y :=  0.951057;

    GFill ( 5, starpoints );

    { Select large characters centred under the star. }

    align.Horizontal    := GVHcentre;
    align.Vertical      := GVVhalf;

    GSetCharHeight ( 0.15 );
    GSetTextAlign  ( align );
    GSetTextColrInd ( 2 );

    { Draw the title.                       }

    position.x  :=  0.0;
    position.y  := -1.1;
    length      :=  4;
    title       := 'Star                                        ';
    GText ( position, length, title );

    { Close the workstation and shut down GKS.      }

    GDeactivateWs ( wsid );
    GCloseWs ( wsid );
    GCloseGKS;

END. { main }
```

Appendix E

E2. Program IRON

```
{
     PROGRAM IRON

     DESCRIPTION:
       This program draws a horizontal bar chart illustrating
       costs within the iron industry.  The user can select the
       data to be displayed using a GKS choice device.  The plot
       is adapted from 'Scientific American' May 1984 page 39.

     CONFORMANCE:
       GKS level: 0b
       Choice device must support prompt and echo type 3

}

PROGRAM IRON ( output );


CONST

{ Implementation-specific method to define the
implementation's constants.  }
$INCLUDE 'gks.p1'$

{ User defined constants  }
     errout  = 1;
     wsid    = 1;
     wstype  = 1;

TYPE

{ Implementation-specific method to define the GKS data types. }
$INCLUDE 'gks.p2'$

     bardata = ARRAY [1..6] of REAL;

VAR
     area    : GRBound;
     asflist : GAASF;
     choice  : GRChoice;
     colr    : GRColr;
     conid   : GAConnId;
     datarec : GRChoiceData;
     echosw  : GEEcho;
     error   : INTEGER;
     gdat1,
     gdat2,
     jdat1,
     jdat2   : bardata;
     opmode  : GEMode;
     prompt  : INTEGER;
```

```
    status  : GEReqStatus;
    usdat1,
    usdat2  : bardata;
    window  : GRBound;

{ Implementation-specific method to define the interface to
the GKS routines  }
$INCLUDE 'gks.p3'$

PROCEDURE bar ( length, pos : REAL );

VAR
    align   : GRAlign;
    bar     : GAPointArray;
    textpos : GRPoint;

BEGIN
    { If the value is too small, print a zero }

    IF (length <= 0.0) THEN
        BEGIN
        align.Horizontal := GVHLeft;
        align.Vertical := GVVhalf;
        GSetTextAlign ( align );

        textpos.x := 0.0;
        textpos.y := pos;
        GText ( textpos, 1, '0');
        END { if }
    ELSE
    { Otherwise draw the bar }

        BEGIN
        bar[1].x := 0.0;
        bar[2].x := length;
        bar[3].x := length;
        bar[4].x := 0.0;

        bar[1].y := pos + 0.4;
        bar[2].y := pos + 0.4;
        bar[3].y := pos - 0.4;
        bar[4].y := pos - 0.4;

        GFill ( 4, bar );
        END; { else }
END; { bar }

PROCEDURE ticks ( first, last : REAL );

VAR
    index   : INTEGER;
    textpos : GRPoint;
```

```
    tick    : GAPointArray;

BEGIN
    { Determine the direction of the tick }

    tick[1].x := 0.0;
    tick[2].x := 0.0;
    tick[1].y := first;
    tick[2].y := last;

    { Draw each tick mark }

    FOR index := 1 TO 4 DO
        BEGIN
        GPolyline ( 2, tick );
        tick[1].x := tick[1].x + 50.0;
        tick[2].x := tick[2].x + 50.0;
        END; { for }

    { Draw the tick mark labels }

    textpos.x := 0.0;
    textpos.y := first;
    GText ( textpos, 1, '0');

    textpos.x := 50.0;
    GText ( textpos, 2, '50');

    textpos.x := 100.0;
    GText ( textpos, 3, '100');

    textpos.x := 150.0;
    GText ( textpos, 3, '150');
END; { ticks }

PROCEDURE border ( length : INTEGER; nation : GAString );

VAR
    align   : GRAlign;
    box     : GAPointArray;
    expan   : REAL;
    textpos : GRPoint;

BEGIN
    { Clear the screen (if not already clear) }

    GClearWs (wsid, GVConditionally);

    { Draw the box surrounding the chart area }

    box[1].x := 0.0;
    box[2].x := 150.0;
```

```
box[3].x := 150.0;
box[4].x := 0.0;
box[5].x := 0.0;
box[1].y := 0.0;
box[2].y := 0.0;
box[3].y := 12.0;
box[4].y := 12.0;
box[5].y := 0.0;

GPolyline ( 5, box );

{ Draw the labels centred on the bar and flush left }

align.Horizontal := GVHleft;
align.Vertical := GVVhalf;
GSetTextAlign ( align );
GSetCharHeight ( 0.5 );
GSetTextColrInd ( 1 );

WITH window DO
    expan := (RightBound - LeftBound) /
             (UpperBound - LowerBound);
GSetCharExpan ( expan );    { Default may be insufficient }

textpos.x := -114.0;
textpos.y := 1.2;
GText (textpos, 5, 'LABOR');

textpos.y := 3.2;
GText (textpos, 8, 'IRON ORE');

textpos.y := 5.2;
GText (textpos, 12, 'COKE OR COAL');

textpos.y := 7.2;
GText (textpos, 15, 'PURCHASED SCRAP');

textpos.y := 9.2;
GText (textpos, 11, 'OTHER COSTS');

textpos.y := 11.2;
GText (textpos, 12, 'OTHER ENERGY');

{ Draw the top and bottom tick marks (bottom in red) }

align.Horizontal := GVHcentre;
align.Vertical := GVVtop;
GSetTextAlign ( align );
GSetTextColrInd (2);
ticks ( 0.0, -0.2 );

align.Horizontal := GVHcentre;
```

```
    align.Vertical := GVVbottom;
    GSetTextAlign ( align );
    GSetTextColrInd (1);
    ticks ( 12.0, 12.2 );

    { Draw the title text in bigger characters }

    textpos.x := 0.0;
    textpos.y := -2.0;
    GText (textpos, 15, 'Production Cost');

    align.Vertical := GVVtop;
    GSetTextAlign ( align );
    GSetCharHeight ( 0.7 );
    GText (textpos, length, nation);
    GSetCharHeight ( 0.5 );
END; { border }

PROCEDURE draw (data1, data2 : bardata;
                size : INTEGER; nation : GAString );

VAR
    index   : INTEGER;
    pos     : REAL;

BEGIN
    { Draw the border }

    border ( size, nation );

    { Draw the black bars }

    GSetTextColrInd ( 1 );
    GSetFillColrInd ( 1 );
    GSetFillInd ( 1 );

    FOR index := 1 TO 6 DO
        BEGIN
        pos := 2.0 * (index-1) + 1.6;
        bar ( data1[index], pos );
        END; { for }

    { Draw the red bars }

    GSetTextColrInd ( 2 );
    GSetFillColrInd ( 2 );
    GSetFillInd ( 2 );

    FOR index := 1 TO 6 DO
        BEGIN
        pos := 2.0 * (index-1) + 0.8;
        bar ( data2[index], pos );
```

```
        END; { for }
END; { draw }

BEGIN { main }
     conid := '/dev/tty    ';

    GOpenGKS ( errout, GCDefMemory );
    GOpenWs  ( wsid, conid, wstype );
    GActivateWs ( wsid );

    { Specify the window for the chart }

    WITH window DO
        BEGIN
        LeftBound   := -115.0;
        RightBound  := 160.0;
        LowerBound  := -2.0;
        UpperBound  := 14.0;
        END; { with }

    GSetWindow ( 1, window );
    GSelectNormTran ( 1 );

    { Define the colours to be used    }

    WITH colr DO
        BEGIN
        Red     := 0.0;
        Green   := 1.0;
        Blue    := 0.0;
        END; { with }
    GSetColrRep (1,1,colr);

    WITH colr DO
        BEGIN
        Red     := 1.0;
        Green   := 0.0;
        END; { with }
    GSetColrRep (1,2,colr);

    { Use bundled attributes except for fill area interior
      style and index }

    asflist[GVLineType]     := GVBundled;
    asflist[GVLineWidth]    := GVBundled;
    asflist[GVLineColr]     := GVBundled;
    asflist[GVMarkerType]   := GVBundled;
    asflist[GVMarkerSize]   := GVBundled;
    asflist[GVMarkerColr]   := GVBundled;
    asflist[GVFontPrec]     := GVBundled;
    asflist[GVExpan]        := GVIndividual;
    asflist[GVSpacing]      := GVBundled;
```

```
asflist[GVTextColr]     := GVBundled;
asflist[GVFillColr]     := GVBundled;
asflist[GVFillInterior] := GVIndividual;
asflist[GVFillStyleInd] := GVIndividual;

GSetASF (asflist);

{ Initialise the choice device }

GInqChoiceDeviceSt (wsid, 1, error, opmode, echosw,
          choice, prompt, area, datarec);
prompt := 3;
WITH datarec DO
    BEGIN
    Prompt := 3;
    R0003NumStrings := 4;
    R0003ChoiceStrings[1].StringLength  := 4;
    R0003ChoiceStrings[1].CharString        :=
        'U.S.                                   ';
    R0003ChoiceStrings[2].StringLength  := 10;
    R0003ChoiceStrings[2].CharString        :=
        'W. GERMANY                             ';
    R0003ChoiceStrings[3].StringLength  := 5;
    R0003ChoiceStrings[3].CharString        :=
        'JAPAN                                  ';
    R0003ChoiceStrings[4].StringLength  := 4;
    R0003ChoiceStrings[4].CharString        :=
        'EXIT                                   ';
    END; { with }

GInitChoice (wsid, 1, choice, prompt, area, datarec);

{ Define the data }

usdat1[1] := 69.0;  usdat2[1] := 72.0;
usdat1[2] := 50.0;  usdat2[2] := 50.0;
usdat1[3] := 15.0;  usdat2[3] := 103.0;
usdat1[4] := 53.0;  usdat2[4] := 0.0;
usdat1[5] := 57.0;  usdat2[5] := 0.0;
usdat1[6] := 150.0; usdat2[6] := 56.0;

gdat1[1]  := 65.0;  gdat2[1]  := 70.0;
gdat1[2]  := 42.0;  gdat2[2]  := 53.0;
gdat1[3]  := 3.0;   gdat2[3]  := 102.0;
gdat1[4]  := 89.0;  gdat2[4]  := 0.0;
gdat1[5]  := 52.0;  gdat2[5]  := 0.0;
gdat1[6]  := 93.0;  gdat2[6]  := 49.0;

jdat1[1]  := 65.0;  jdat2[1]  := 70.0;
jdat1[2]  := 47.0;  jdat2[2]  := 57.0;
jdat1[3]  := 2.0;   jdat2[3]  := 105.0;
jdat1[4]  := 60.0;  jdat2[4]  := 0.0;
```

```
    jdat1[5]  := 52.0;  jdat2[5]  := 0.0;
    jdat1[6]  := 55.0;  jdat2[6]  := 41.0;

    { Get the users choice (u.s., w. germany, japan, or exit ) }

    REPEAT
        GReqChoice (wsid,1,status,choice);
        CASE choice.ChoiceNum OF
        1: draw (usdat1, usdat2, 4,
            'U.S.                                                ';
        2: draw (gdat1, gdat2, 10,
            'W. Germany                                          ';
        3: draw (jdat1, jdat2, 5,
            'Japan                                               ';
        4: ;
        END; { case }
    UNTIL (choice.ChoiceNum > 3);

    { Close the workstation and shut down GKS }

    GDeactivateWs ( wsid );
    GCloseWs ( wsid );
    GCloseGKS;
END. { main }
```

Appendix E

E3. Program MAP

```
{
      PROGRAM MAP

          DESCRIPTION:
            This program reads a GKS metafile to draw a map. The
            primitives in each region are in a separate segment.
            The user can then use a pick device to select the
            various regions. A sampled choice device determines
            the action taken with the selected region.  The
            choice number assignments are:
                 1. highlight the region
                 2. turn off highlighting
                 3. make the region visible
                 4. make the region invisible
                 5. exit

          CONFORMANCE:
            GKS level: 1c
            The implementation must support at least one
            workstation of category outin and one of category mi
            (metafile input). The default choice device must
            support at least five choices.

 ************************************************************
}

PROGRAM MAP (output);

CONST

{ Implementation-specific method to define the
implementation's constants.  }
$INCLUDE 'gks.p1'$

{ User defined constants  }
    metaid  = 2;
    wsid    = 1;
    wstype  = 1;
    mi      = 8;

TYPE

{ Implementation-specific method to define the GKS data types. }
$INCLUDE 'gks.p2'$

VAR
    choiceval   : GRChoice;
    conid       : GAConnId;
    datrec      : GRFileData;
    fnam        : GTErrorFileName;
    idrl        : INTEGER;
```

```
    itype       : INTEGER;
    miid        : GAConnId;
    pickval     : GRPick;
    stat        : GEReqStatus;

{ Implementation-specific method to define the interface to
the GKS routines  }
$INCLUDE 'gks.p3'$

{************************************************************}

BEGIN

    { Open GKS and activate a workstation.      }

    fnam    := 1;
    conid   := '/dev/display';

    GOpenGKS ( fnam, GCDefMemory );
    GOpenWs ( wsid, conid, wstype );
    GActivateWs ( wsid );

    { Set the choice device to sample mode.     }

    GSetChoiceMode ( wsid, 1, GVSample, GVNoEcho );

    { Open the metafile input workstation.      }

    miid    := 'metainput   ';

    GOpenWs ( metaid, miid, mi );

    { Interpret items until eof item is read.   }

    REPEAT
        GGetItemType ( metaid, itype, idrl );
        IF ( itype <> 0 ) THEN
            BEGIN
            GReadItem ( metaid, idrl, datrec );
            GInterpretItem ( itype, idrl, datrec );
            END; { if }
    UNTIL (itype = 0);

    { Close the metafile.  }

    GCloseWs ( metaid );

    { Allow the user to select states until they select
      the 'exit' choice. }

    REPEAT
        GReqPick ( wsid, 1, stat, pickval );
```

```
        IF ( stat = GVStatusOk ) THEN
            BEGIN
            { See which choice is in effect.  }

            GSampleChoice ( wsid, 1, choiceval );

            IF ( choiceval.ChoiceStatus = GVStatusOk ) THEN
                BEGIN
                CASE choiceval.ChoiceNum OF
                1: GSetHighlight
                    ( pickval.Segment, GVHighlighted );
                2: GSetHighlight
                    ( pickval.Segment, GVNormal );
                3: GSetVis
                    ( pickval.Segment, GVInvisible );
                4: GSetVis
                    ( pickval.Segment, GVVisible );
                5: ;
                END; { case }
                END; { if   }
            END; { if }
    UNTIL (choiceval.ChoiceNum = 5);

    { Close the workstation and shut down GKS.  }

    GDeactivateWs ( wsid );
    GCloseWs ( wsid );
    GCloseGKS;

END. { main }
```

E4. Program MANIPULATE

```
{
    PROGRAM MANIPULATE

    DESCRIPTION:
     This program allows the user to create an object and
     then manipulate the object by changing the segment
     transformation.

    CONFORMANCE:
     GKS level: 2b
     This program uses the conformant array versions of
     GMessage and GText.
}

PROGRAM MANIPULATE ( output );

CONST

{ Implementation-specific method to define the implementation's
constants.  }
$INCLUDE 'gks.p1'$

{ User defined constants  }

    DeviceNum    = 1;                {device number for input}
    Display      = 1;                {ws identifier for display}
    DisConn      = 'port1       ';   {connection for display}
    DisType      = 510;              {type number for display}
    errorfile    = 1;                {file for error handling}
    LocatorDevice   = 1;             {device number for input}
    Plotter      = 2;                {ws identifier for plotter}
    PloConn      = 'port2       ';   {connection for plotter}
    PloType      = 1000;             {type number for plotter}
    segstore     = 3;                {ws identifier for segstore}
    SegConn      = 'null        ';   {connection for segstore}
    SegType      = 0;                {type number for segstore}

TYPE

{ Implementation-specific method to define the GKS data types. }
$INCLUDE 'gks.p2'$

VAR

    {global transformation matrix}
    matrix : GAMatrix;

{ Implementation-specific method to define the interface to the
GKS routines  }
$INCLUDE 'gks.p3'$
```

Appendix E Program MANIPULATE

```
{************************************************************}

PROCEDURE Initialise;

{initialises GKS and workstations}

BEGIN
    GOpenGKS ( errorfile, GCDefMemory );
    GOpenWs  ( Display, DisConn, DisType );
    GOpenWs  ( segstore, SegConn, SegType );
    GOpenWs  ( Plotter, PloConn, PloType );
    GActivateWs ( Display );
    GActivateWs ( segstore );
END; { initialise }

{************************************************************}

PROCEDURE Close;

{ Close the workstations and shut down GKS }

BEGIN
    GDeactivateWs ( Display );
    GDeactivateWs ( segstore );
    GCloseWs ( Plotter );
    GCloseWs ( segstore );
    GCloseWs ( Display );
    GCloseGKS;
END; { close }

{************************************************************}

PROCEDURE SetUpTrans;

VAR
    window,
    viewport : GRBound;

BEGIN

    {initialise window data}

    WITH window DO
        BEGIN
        LeftBound   := 0.0;
        RightBound  := 100.0;
        LowerBound  := 0.0;
        UpperBound  := 100.0;
        END; { with }

    {initialise viewport data}
```

```
    WITH viewport DO
        BEGIN
        LeftBound   := 0.05;
        RightBound  := 0.95;
        LowerBound  := 0.05;
        UpperBound  := 0.95;
        END; { with }

    {set window, viewport and viewport input priority}

    GSetWindow   ( 1, window );
    GSetViewport ( 1, viewport );
    GSetViewportPriority ( 1, 0, GVHigher );

end; { SetUpTrans }

{*********************************************************}

PROCEDURE CreateObject ( polygon : GTSeg );

{create an object interactively}

VAR
    status          : GEReqStatus;
    stroke          : GRInput;
    strokefailed    : boolean;

BEGIN

    {set polyline index to 3}

    GSetPrimInd ( GVPolyline, 3 );

    {create the required polygon}

    strokefailed := TRUE;
    WHILE strokefailed DO
        BEGIN
        GReqInput ( GVStroke, Display, DeviceNum,
                    status, stroke );
        strokefailed := (status=GVStatusNone) OR
                    (stroke.Num>=GCMaxPoint);

        IF strokefailed THEN
            BEGIN
            GMessage ( Display, 33,
                    'stroke not successful - try again');
            GClearWs ( Display, GVConditionally );
            END; { if }
        END; { while }

    GCreateSeg ( polygon );
```

```
    WITH stroke DO
        BEGIN
        GSelectNormTran ( NormTranStroke );
        Points[Num+1] := Points[1];
        GPolyline ( Num+1, Points );
        END; { with }
    GCloseSeg;

END; { CreateObject }

{*************************************************************}


PROCEDURE ShiftObject ( polygon : GTseg; var continue : boolean );

{does segment transformation on polygon resulting in a shift}

VAR
    fixedpoint  : GRPoint;
    locator1,
    locator2    : GRInput;
    outmatrix   : GAMatrix;
    scale       : GRVector;
    shift       : GRVector;
    status1,
    status2     : GEReqStatus;

BEGIN
    GReqInput ( GVLocator, Display, LocatorDevice,
                        status1, locator1 );
    GReqInput ( GVLocator, Display, LocatorDevice,
                        status2, locator2 );

    continue := (status1 = GVStatusOK) AND
                (status2 = GVStatusOK);
    IF continue THEN
        BEGIN
        GSelectNormTran ( locator1.NormTranLocator );

        WITH shift DO
            BEGIN
            xValue := locator2.position.x - locator1.position.x;
            yValue := locator2.position.y - locator1.position.y
            END; { with }

        WITH fixedpoint DO
            BEGIN
            x := 0.0;
            y := 0.0
            END; { with }

        WITH scale DO
```

```
            BEGIN
            xValue := 1.0;
            yValue := 1.0;
            END; { with }

        GAccumTran ( matrix, fixedpoint, shift, 0.0, scale,
                                        GVwc, outmatrix );
        GSetSegTran ( polygon, outmatrix );
        matrix := outmatrix;
        END; { if }

END; { ShiftObject }

{*********************************************************}

PROCEDURE ScaleObject ( polygon : GTseg;
                        var continue : boolean );

VAR
    fixedpoint      : GRPoint;
    outmatrix       : GAMatrix;
    returnvalue,
    returnposition  : GRInput;
    scale           : GRVector;
    shift           : GRVector;
    status1,
    status2         : GEReqStatus;

BEGIN
    {choose scaling from valuator input}

    GReqInput ( GVValuator, Display, DeviceNum,
                        status1, returnvalue );

    {choose fixed point from request locator}

    GReqInput ( GVLocator, Display, LocatorDevice,
                        status2, returnposition );

    continue := (status1 = GVStatusOK) AND
                (status2 = GVStatusOK);
    IF continue THEN
        BEGIN

        WITH shift DO
            BEGIN
            xValue := 0.0;
            yValue := 0.0
            END; { with }

        WITH returnposition DO
            BEGIN
```

```
            GSelectNormTran ( NormTranLocator );
            fixedpoint := Position;
            END; { with }

        WITH scale DO
            BEGIN
            xValue := returnvalue.Value;
            yValue := returnvalue.Value;
            END; { with }

        GAccumTran ( matrix, fixedpoint, shift, 0.0, scale,
                                        GVwc, outmatrix );
        GSetSegTran ( polygon, outmatrix );
        matrix := outmatrix;
        END; { if }

END; { ScaleObject }
{*********************************************************}

PROCEDURE InitValuator ( low, high : real );

{Initialises Valuator Device to required range}

VAR
    area         : GRBound;
    datarecord   : GRInputData;
    echosw       : GEEcho;
    error        : INTEGER;
    initvalue    : GRInput;
    mode         : GEMode;
    prompt       : INTEGER;
    typereturn   : GEReturn;

BEGIN

    {inquire valuator device state}

    GInqInputDeviceSt ( GVValuator, Display, DeviceNum,
                    typereturn, error, mode, echosw, initvalue,
                    prompt, area, datarecord );

    {reset data record}

    WITH datarecord DO
        BEGIN
        LowValue    := low;
        HighValue   := high;
        END; { with }

    {reset initial input}

    WITH initvalue DO
```

```
        Value := 0.5 * (low+high);

    {initialise valuator}

    GInitInput ( GVValuator, Display, DeviceNum,
                    initvalue, prompt, area, datarecord );
END; { InitValuator }

{*************************************************************}

PROCEDURE InitTranMatrix;

{Initialise tranformation matrix}

VAR
    angle       : real;
    fixedpoint  : GRPoint;
    scale       : GRVector;
    shift       : GRVector;

BEGIN
    WITH fixedpoint DO
        BEGIN
        x := 0.0;
        y := 0.0
        END; { with }

    WITH shift DO
        BEGIN
        xValue := 0.0;
        yValue := 0.0
        END; { with }

    WITH scale DO
        BEGIN
        xValue := 1.0;
        yValue := 1.0;
        END; { with }

    GEvalTran ( fixedpoint, shift, 0.0, scale, GVwc, matrix );

END; { InitTranMatrix }

{*************************************************************}

PROCEDURE Interaction ( polygon : GTseg );

CONST
    SHIFT   = 1;
    SCALE   = 2;
    QUIT    = 3;
```

Appendix E Program MANIPULATE

```
VAR
    choice   : GRInput;
    continue: BOOLEAN;
    status   : GEReqStatus;

BEGIN
    InitTranMatrix;

    {Initialise the valuator device to return values
    between 0.5 and 2.5}

    InitValuator ( 0.5, 2.5 );
    REPEAT
        GReqInput ( GVChoice, Display, DeviceNum,
                              status, choice );

        continue := status=GVStatusOK;
        IF continue THEN
        CASE choice.ChoiceNum OF
            SHIFT   :   ShiftObject ( polygon, continue );

            SCALE   :   ScaleObject ( polygon, continue );

            QUIT    :   continue := false;
        END; { case }
    UNTIL (not continue);
END; { Interaction }

{*************************************************************}


PROCEDURE UsePlotter ( polygon : GTseg );

VAR
    colour      : GRColr;
    linerep     : GRPrimRep;
    position    : GRPoint;
    textrep     : GRPrimRep;

BEGIN
    GDeactivateWs ( Display );
    GDeactivateWs ( segstore );
    GActivateWs ( Plotter );

    WITH colour DO
        BEGIN
        Red     := 1.0;
        Green   := 0.0;
        Blue    := 0.0
        END; { with }
    GSetColrRep ( Plotter, 1, colour );
```

```
    WITH linerep DO
        BEGIN
        Prim    := GVPolyline;
        Ltype   := 1;
        Width   := 1.5;
        LColr   := 1;
        END; { with }
    GSetPrimRep ( GVPolyline, Plotter, 3, linerep );

    WITH textrep DO
        BEGIN
        Prim := GVText;
        WITH fontprec DO
            BEGIN
            font := 1;
            prec := GVStringPrec;
            END; { with }
        Expan   := 0.0;
        Spacing := 0.0;
        TColr   := 1;
        END; { with textrep }
    GSetPrimRep ( GVText, Plotter, 2, textrep );

    GCopySegWs ( Plotter, polygon );
    GSetPrimInd ( GVText, 2 );
    GSetCharHeight ( 0.1 );

    WITH position DO
        BEGIN
        x := 0.5;
        y := 0.5
        END; { with }
    GText ( position, 17, 'This is a polygon' );

    GDeactivateWs ( Plotter );
    GActivateWs ( Display );
    GActivateWs ( segstore );

END; { UsePlotter }

{*********************************************************}

PROCEDURE GeneratePolygon;

BEGIN
    Initialise;
    SetUpTrans;
    CreateObject ( 1 );
    Interaction ( 1 );
    UsePlotter ( 1 );
    Close;
END; { GeneratePolygon }
```

Appendix E Program MANIPULATE

```
{*************************************************************}
BEGIN { Main }

    GeneratePolygon;

END. { Main }
```

E5. Program SHOWLN

```
{
     PROGRAM SHOWLN

          DESCRIPTION:
            This program contains a typical GKS initialisation
            module and demonstrates how to program modules which
            do not change any state list entries.

          CONFORMANCE:
            GKS level: ma

 ************************************************************
}

PROGRAM SHOWLN (input, output);

LABEL 99;

CONST

{ Implementation-specific method to define the
implementation's constants.  }
$INCLUDE 'gks.p1'$

{ User defined constants  }
    wsid    = 1;

TYPE

{ Implementation-specific method to define the GKS data types. }
$INCLUDE 'gks.p2'$

VAR
    wstype      : GTWsType;
    error_ind   : INTEGER;

{ Implementation-specific method to define the interface to
the GKS routines  }
$INCLUDE 'gks.p3'$

{************************************************************}

PROCEDURE init_gks ( VAR wstype : GTWsType;
                VAR error_ind : INTEGER);

VAR
    category    : GEWsCategory;
    conid       : GAConnId;
    done        : BOOLEAN;
    index       : INTEGER;
    num_types   : GTInt1;
```

```
    opstate      : GEOpSt;
    size         : INTEGER;
    ws_types     : GAWsType;

BEGIN { init_gks }

    { Open GKS and activate a workstation.       }

    GOpenGKS ( GCDefErrorLog, GCDefMemory );

    {Inquire the available workstation types and print them }

    writeln
      ('The available output and outin workstation types are:');
    GInqListWsTypes (1, size, done, error_ind,
                              num_types, ws_types);
    FOR index := 1 TO num_types DO
        BEGIN
        GInqWsCategory (ws_types[index], error_ind, category);
        IF ((category = GVOutput) or (category = GVOutIn)) THEN
            writeln (ws_types[index]);
        END; { for }

    { Choose one workstation and open and activate it }

    writeln
      ('Please enter connection identifier and workstation type');
    readln (conid, wstype);

    GOpenWs ( wsid, conid, wstype );
    GActivateWs ( wsid );

    { Check the operating state to ensure successful
        opening and activating }

    GInqOpSt (opstate);
    IF (opstate <> GVwsac) THEN
        error_ind := 3
    ELSE
        error_ind := 0;

END; { init_gks }

{*************************************************************}

PROCEDURE line_demo ( wstype : GTWsType;
                VAR errorind : INTEGER);

LABEL 98;

VAR
    align        : GRAlign;
```

```
    asf_indiv,
    asf_list     : GAASF;
    char_count   : GTMaxString;
    char_list    : GAString;
    dist         : REAL;
    font_prec    : GRFontPrec;
    height       : REAL;
    index        : INTEGER;
    indiv_attr   : GAPrimRep;
    line_type    : INTEGER;
    last         : BOOLEAN;
    line_facil   : GRLineFacil;
    norm_tran    : GTInt0;
    opstate      : GEOpSt;
    prim_attr    : GRPrimAttr;
    up_vect      : GRVector;
    upd_state    : GEWsTran;
    req_win,
    cur_win,
    req_view,
    cur_view     : GRBound;
    point_list   : GAPointArray;

BEGIN { line_demo }

    { Check the operating state value }

    GInqOpSt (opstate);
    IF (opstate < GVwsac) THEN
        BEGIN
        error_ind := 5;
        goto 98;
        END;

    { Inquire workstation transformation }

    GInqWsTran ( wsid, error_ind, upd_state, req_win, cur_win,
            req_view, cur_view );
    IF (error_ind <> 0) THEN goto 98;

    { Inquire polyline facilities }

    GInqPolylineFacil ( wsid, 1, last, error_ind, line_facil );

    { Save normalization transformation number,
      relevant polyline and text attributes }

    GInqCurNormTranNum ( error_ind, norm_tran );
    GInqCurIndivAttr ( error_ind, indiv_attr, asf_list );
    GInqCurPrimAttr ( error_ind, prim_attr );

    { Set unity normalization transformation,
```

```
    individual aspect source flags,
    linewidth scale factor 1.0 and polyline colour index 1,
    reasonable text attributes }

  asf_indiv[GVLineType]       := GVIndividual;
  asf_indiv[GVLineWidth]      := GVIndividual;
  asf_indiv[GVLineColr]       := GVIndividual;
  asf_indiv[GVMarkerType]     := GVIndividual;
  asf_indiv[GVMarkerSize]     := GVIndividual;
  asf_indiv[GVMarkerColr]     := GVIndividual;
  asf_indiv[GVFontPrec]       := GVIndividual;
  asf_indiv[GVExpan]          := GVIndividual;
  asf_indiv[GVSpacing]        := GVIndividual;
  asf_indiv[GVTextColr]       := GVIndividual;
  asf_indiv[GVFillColr]       := GVIndividual;
  asf_indiv[GVFillInterior]   := GVIndividual;
  asf_indiv[GVFillStyleInd]   := GVIndividual;

  up_vect.xValue := 0.0;
  up_vect.yValue := 1.0;

  align.Horizontal := GVHleft;
  align.Vertical := GVVhalf;

  font_prec.Font := 1;
  font_prec.Prec := GVCharPrec;

  GSelectNormTran ( 0 );
  GSetASF ( asf_indiv );
  GSetLineWidthScale ( 1.0 );
  GSetLineColrInd ( 1 );
  GSetCharUpVector ( up_vect );
  GSetTextPath ( GVRight );
  GSetTextAlign ( align );
  GSetTextFontPrec ( font_prec );
  GSetCharExpan ( 1.0 );
  GSetCharSpacing ( 0.0 );
  GSetTextColrInd ( 1 );

  { Compute distance between lines }

  dist := ( cur_win.UpperBound - cur_win.LowerBound ) /
            line_facil.NumTypes;

  { Set character height to half of the distance between lines,
    but not more than 1/20th of the height of the current
    workstation viewport }

  height := ( cur_win.UpperBound - cur_win.LowerBound ) / 20.0;
  IF ( height > (dist / 2.0) ) THEN height := dist/2.0;

  GSetCharHeight ( height );
```

```
{ Lines stretch from the left bound to the middle of the
  current workstation viewport }

point_list[1].X := cur_win.LeftBound;
point_list[2].X := (cur_win.LeftBound + cur_win.RightBound) /
                                2.0;
point_list[1].Y := cur_win.UpperBound - dist/2.0;

{ Loop over available linetypes }

FOR index := 1 TO line_facil.NumTypes DO
    BEGIN

    { Draw line }

    line_type := line_facil.lines[index];
    GSetLineType ( line_type );

    point_list[2].Y := point_list[1].Y;
    GPolyline ( 2, point_list );
    point_list[1].Y := point_list[1].Y - dist;

    { Annotate linetype }
    char_count := 1;
    IF (line_type < 0) THEN
        BEGIN
        char_list[char_count] := '-';
        char_count := char_count + 1;
        line_type := -line_type;
        END; { if }
    REPEAT
        char_list[char_count] := chr(line_type + ord('0'));
        char_count := char_count + 1;
        line_type := line_type DIV 10;
    UNTIL (line_type = 0);

    GText ( point_list[2], char_count, char_list );
    END; { for }

{ Restore normalization transformation and attributes }

GSelectNormTran ( norm_tran );
GSetASF ( asf_list );
GSetLineWidthScale ( indiv_attr[GVPolyline].Width );
GSetLineColrInd ( indiv_attr[GVPolyline].LColr );
GSetCharHeight ( prim_attr.text.Height );
GSetCharUpVector ( prim_attr.text.UpVector );
GSetTextPath ( prim_attr.text.Path );
GSetTextAlign ( prim_attr.text.Align );
GSetTextFontPrec ( indiv_attr[GVText].FontPrec );
GSetCharExpan ( indiv_attr[GVText].Expan );
GSetCharSpacing ( indiv_attr[GVText].Spacing );
```

```
    GSetTextColrInd ( indiv_attr[GVText].TColr );

    error_ind := 0;

98: ;
END; { line_demo }

{*************************************************************}

PROCEDURE close_gks;

BEGIN { close_gks }

    { Close the workstation and shut down GKS.      }

    GDeactivateWs ( wsid );
    GCloseWs ( wsid );
    GCloseGKS;
END; { close_gks }

{*************************************************************}

BEGIN { main }

    { Call the initialisation module }

    init_gks (wstype, error_ind);
    if (error_ind <> 0) THEN
        BEGIN
        GEmergencyCloseGKS;
        goto 99;
        END;

    { Call demonstration module for linetype capabilities }

    line_demo (wstype, error_ind);
    if (error_ind <> 0) THEN
        BEGIN
        GEmergencyCloseGKS;
        goto 99;
        END;

    { Shut down GKS }

    close_gks;
99: ;
END. { main }
```

Appendix F

Function Lists

(This Appendix is not part of ANSI X3.124.2-1988, but is included for information only.)

F1. GKS Functions

F2. Pascal Functions

X3.113-1987 Programming Language FULL BASIC
X3.114-1984 Alphanumeric Machines; Coded Character Sets for Keyboard Arrangements in ANSI X4.23-1982 and X4.22-1983
X3.115-1984 Unformatted 80 Megabyte Trident Pack for Use at 370 tpi and 6000 bpi (General, Physical, and Magnetic Characteristics)
X3.116-1986 Recorded Magnetic Tape Cartridge, 4-Track, Serial 0.250 Inch (6.30 mm) 6400 bpi (252 bpmm), Inverted Modified Frequency Modulation Encoded
X3.117-1984 Printable/Image Areas for Text and Facsimile Communication Equipment
X3.118-1984 Financial Services — Personal Identification Number — PIN Pad
X3.119-1984 Contact Start/Stop Storage Disk, 158361 Flux Transitions per Track, 8.268 Inch (210 mm) Outer Diameter and 3.937 inch (100 mm) Inner Diameter
X3.120-1984 Contact Start/Stop Storage Disk
X3.121-1984 Two-Sided, Unformatted, 8-Inch (200-mm), 48-tpi, Double-Density, Flexible Disk Cartridge for 13 262 ftpr Two-Headed Application
X3.122-1986 Computer Graphics Metafile for the Storage and Transfer of Picture Description Information
X3.124-1985 Graphical Kernel System (GKS) Functional Description
X3.124.1-1985 Graphical Kernel System (GKS) FORTRAN Binding
X3.124.2-1988 Graphical Kernel System (GKS) PASCAL Binding
X3.125-1985 Two-Sided, Double-Density, Unformatted 5.25-Inch (130-mm), 48-tpi (1,9-tpmm), Flexible Disk Cartridge for 7958 bpr Use
X3.126-1986 One- or Two-Sided Double-Density Unformatted 5.25-Inch (130-mm), 96 Tracks per Inch, Flexible Disk Cartridge
X3.127-1987 Unrecorded Magnetic Tape Cartridge for Information Interchange
X3.128-1986 Contact Start-Stop Storage Disk — 83 000 Flux Transitions per Track, 130-mm (5.118-Inch) Outer Diameter and 40-mm (1.575-Inch) Inner Diameter
X3.129-1986 Intelligent Peripheral Interface, Physical Level
X3.130-1986 Intelligent Peripheral Interface, Logical Device Specific Command Sets for Magnetic Disk Drive
X3.131-1986 Small Computer Systems Interface
X3.132-1987 Intelligent Peripheral Interface — Logical Device Generic Command Set for Optical and Magnetic Disks
X3.133-1986 Database Language — NDL
X3.135-1986 Database Language — SQL
X3.136-1988 Serial Recorded Magnetic Tape Cartridge for Information Interchange, Four and Nine Track
X3.137-1988 Unformatted Flexible Disk Cartridge, 90 mm (3.5 Inch) 5.3 tpmm (135 tpi) for 7958 bpr Use
X3.139-1987 Fiber Distributed Data Interface (FDDI) Token Ring Media Access Control (MAC)
X3.140-1986 Open Systems Interconnection — Connection Oriented Transport Layer Protocol Specification
X3.141-1987 Data Communication Systems and Services — Measurement Methods for User-Oriented Performance Evaluation
X3.146-1987 Device Level Interface for Streaming Cartridge and Cassette Tape Drives
X3.147—1988 Intelligent Peripheral Interface — Device Generic Command Set for Magnetic Tape Drives
X3.153-1987 Open Systems Interconnection — Basic Connection Oriented Session Protocol Specification
X3.156-1987 Nominal 8-Inch Rigid Disk Removable Cartridge
X3.157-1987 Recorded Magnetic Tape for Information Interchange, 3200 CPI
X3.158-1987 Serial Recorded Magnetic Tape Cassette for Information Interchange, 0.150 Inch (3.81 mm), 8000 bpi (315 bpmm), Group Code Recording
X3.162-1988 Two-Sided, High-Density, Unformatted, 5.25-Inch (130-mm), 96 tpi, Flexible Disk Cartridge for 13 262 ftpr Use
X11.1-1977 Programming Language MUMPS
IEEE 416-1978 Abbreviated Test Language for All Systems (ATLAS)
IEEE 716-1982 Standard C/ATLAS Language
IEEE 717-1982 Standard C/ATLAS Syntax
IEEE 770X3.97-1983 Programming Language PASCAL
IEEE 771-1980 Guide to the Use of ATLAS
ISO 8211-1986 Specifications for a Data Descriptive File for Information Interchange
MIL-STD-1815A-1983 Reference Manual for the Ada Programming Language
NBS-ICST 1-1986 Fingerprint Identification — Data Format for Information Interchange

---

X3/TRI-82 Dictionary for Information Processing Systems (Technical Report)

American National Standards for Information Processing

X3.1-1987 Synchronous Signaling Rates for Data Transmission
X3.2-1970 Print Specifications for Magnetic Ink Character Recognition
X3.4-1986 Coded Character Sets — 7-Bit ASCII
X3.5-1970 Flowchart Symbols and Their Usage
X3.6-1965 Perforated Tape Code
X3.9-1978 Programming Language FORTRAN
X3.11-1969 General Purpose Paper Cards
X3.14-1983 Recorded Magnetic Tape (200 CPI, NRZI)
X3.15-1976 Bit Sequencing of the American National Standard Code for Information Interchange in Serial-by-Bit Data Transmission
X3.16-1976 Character Structure and Character Parity Sense for Serial-by-Bit Data Communication in the American National Standard Code for Information Interchange
X3.17-1981 Character Set for Optical Character Recognition (OCR-A)
X3.18-1974 One-Inch Perforated Paper Tape
X3.19-1974 Eleven-Sixteenths-Inch Perforated Paper Tape
X3.20-1967 Take-Up Reels for One-Inch Perforated Tape
X3.21-1967 Rectangular Holes in Twelve-Row Punched Cards
X3.22-1983 Recorded Magnetic Tape (800 CPI, NRZI)
X3.23-1985 Programming Language COBOL
X3.25-1976 Character Structure and Character Parity Sense for Parallel-by-Bit Data Communication in the American National Standard Code for Information Interchange
X3.26-1980 Hollerith Punched Card Code
X3.27-1987 Magnetic Tape Labels and File Structure
X3.28-1976 Procedures for the Use of the Communication Control Characters of American National Standard Code for Information Interchange in Specified Data Communication Links
X3.29-1971 Specifications for Properties of Unpunched Oiled Paper Perforator Tape
X3.30-1986 Representation for Calendar Date and Ordinal Date
X3.31-1988 Identification of the Counties of the United States
X3.32-1973 Graphic Representation of the Control Characters of American National Standard Code for Information Interchange
X3.34-1972 Interchange Rolls of Perforated Tape
X3.37-1987 Programming Language APT
X3.38-1988 Identification of States of the United States (Including the District of Columbia)
X3.39-1986 Recorded Magnetic Tape (1600 CPI, PE)
X3.40-1983 Unrecorded Magnetic Tape (9-Track 800 CPI, NRZI; 1600 CPI, PE; and 6250 CPI, GCR)
X3.41-1974 Code Extension Techniques for Use with the 7-Bit Coded Character Set of American National Standard Code for Information Interchange
X3.42-1975 Representation of Numeric Values in Character Strings
X3.43-1986 Representations of Local Time of Day
X3.44-1974 Determination of the Performance of Data Communication Systems
X3.45-1982 Character Set for Handprinting
X3.46-1974 Unrecorded Magnetic Six-Disk Pack (General, Physical, and Magnetic Characteristics)
X3.47-1988 Identification of Named Populated Places, Primary County Divisions, and Other Locational Entities of the United States
X3.48-1986 Magnetic Tape Cassettes (3.81-mm [0.150-Inch] Tape at 32 bpmm [800 bpi], PE)
X3.49-1975 Character Set for Optical Character Recognition (OCR-B)
X3.50-1986 Representations for U.S. Customary, SI, and Other Units to Be Used in Systems with Limited Character Sets
X3.51-1986 Representations of Universal Time, Local Time Differentials, and United States Time Zone References
X3.52-1976 Unrecorded Single-Disk Cartridge (Front Loading, 2200 BPI) (General, Physical, and Magnetic Requirements)
X3.53-1976 Programming Language PL/I
X3.54-1986 Recorded Magnetic Tape (6250 CPI, Group Coded Recording)
X3.55-1982 Unrecorded Magnetic Tape Cartridge, 0.250 Inch (6.30 mm), 1600 bpi (63 bpmm), Phase encoded
X3.56-1986 Recorded Magnetic Tape Cartridge, 4 Track, 0.250 Inch (6.30 mm), 1600 bpi (63 bpmm), Phase Encoded
X3.57-1977 Structure for Formatting Message Headings Using the American National Standard Code for Information Interchange for Data Communication Systems Control
X3.58-1977 Unrecorded Eleven-Disk Pack (General, Physical, and Magnetic Requirements)
X3.60-1978 Programming Language Minimal BASIC
X3.61-1986 Representation of Geographic Point Locations
X3.62-1987 Paper Used in Optical Character Recognition (OCR) Systems
X3.63-1981 Unrecorded Twelve-Disk Pack (100 Megabytes) (General, Physical, and Magnetic Requirements)
X3.64-1979 Additional Controls for Use with American National Standard Code for Information Interchange
X3.66-1979 Advanced Data Communication Control Procedures (ADCCP)
X3.72-1981 Parallel Recorded Magnetic Tape Cartridge, 4 Track, 0.250 Inch (6.30 mm), 1600 bpi (63 bpmm), Phase Encoded
X3.73-1980 Single-Sided Unformatted Flexible Disk Cartridge (for 6631-BPR Use)
X3.74-1981 Programming Language PL/I, General-Purpose Subset
X3.76-1981 Unformatted Single-Disk Cartridge (Top Loading, 200 tpi 4400 bpi) (General, Physical, and Magnetic Requirements)
X3.77-1980 Representation of Pocket Select Characters
X3.78-1981 Representation of Vertical Carriage Positioning Characters in Information Interchange
X3.79-1981 Determination of Performance of Data Communications Systems That Use Bit-Oriented Communication Procedures
X3.80-1981 Interfaces between Flexible Disk Cartridge Drives and Their Host Controllers
X3.82-1980 One-Sided Single-Density Unformatted 5.25-Inch Flexible Disk Cartridge (for 3979-BPR Use)
X3.83-1980 ANSI Sponsorship Procedures for ISO Registration According to ISO 2375
X3.84-1981 Unformatted Twelve-Disk Pack (200 Megabytes)(General, Physical, and Magnetic Requirements
X3.85-1981 1/2-Inch Magnetic Tape Interchange Using a Self Loading Cartridge
X3.86-1980 Optical Character Recognition (OCR) Inks
X3.88-1981 Computer Program Abstracts
X3.89-1981 Unrecorded Single-Disk, Double-Density Cartridge (Front Loading, 2200 bpi, 200 tpi) (General, Physical, and Magnetic Requirements)
X3.91M-1987 Storage Module Interfaces
X3.92-1981 Data Encryption Algorithm
X3.93M-1981 OCR Character Positioning
X3.94-1985 Programming Language PANCM
X3.95-1982 Microprocessors — Hexadecimal Input/Output, Using 5-Bit and 7-Bit Teleprinters
X3.96-1983 Continuous Business Forms (Single-Part)
X3.98-1983 Text Information Interchange in Page Image Format (PIF)
X3.99-1983 Print Quality Guideline for Optical Character Recognition (OCR)
X3.100-1983 Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment for Packet Mode Operation with Packet Switched Data Communications Network
X3.101-1984 Interfaces Between Rigid Disk Drive(s) and Host(s)
X3.102-1983 Data Communication Systems and Services — User-Oriented Performance Parameters
X3.103-1983 Unrecorded Magnetic Tape Minicassette for Information Interchange, Coplanar 3.81 mm (0.150 Inch)
X3.104-1983 Recorded Magnetic Tape Minicassette for Information Interchange, Coplanar 3.81 mm (0.150 in), Phase Encoded
X3.105-1983 Data Link Encryption
X3.106-1983 Modes of Operation for the Data Encryption Algorithm
X3.108-1988 Physical Layer Interface for Local Distributed Data Interfaces to a Nonbranching Coaxial Cable Bus
X3.110-1983 Videotex/Teletext Presentation Level Protocol Syntax
X3.111-1986 Optical Character Recognition (OCR) Matrix Character Sets for OCR-M
X3.112-1984 14-in (356-mm) Diameter Low-Surface-Friction Magnetic Storage Disk

(Continued on reverse)

June 1988