NISTIR 4317

A Collection of Technical Studies Completed for the Computer-Aided Acquisition and Logistic Support (CALS) Program Fiscal Year 1988 Volume 3 of 3, CGM Registration

Roy S. Morgan
Editor

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

NIST

NISTIR 4317

A Collection of Technical Studies Completed for the Computer-Aided Acquisition and Logistic Support (CALS) Program Fiscal Year 1988 Volume 3 of 3, CGM Registration

Roy S. Morgan
Editor

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Gaithersburg, MD 20899

April 1990

Issued March 1991

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

Executive Summary

The overall objective of the Department of Defense Computer-aided Acquisition and Logistic Support (CALS) Program is to integrate the design, manufacturing, and logistic functions through the efficient application of computer technology. CALS is a program to apply existing and emerging communications and computer-aided technologies in DoD and industry to:

- Integrate and improve design, manufacturing, and logistic functions; thereby bridging existing "islands of automation."
- Actively influence the design process to produce weapon systems that are more reliable and easier to support and maintain.
- Shift from current paper-intensive weapon support processes to a highly automated mode of operation, based on a unified DoD interface with industry for exchange of logistic technical information in digital form.

The CALS program was established by the Deputy Secretary of Defense in September 1985 to implement the recommendations of a Joint Industry/DoD Task Force. Management is provided by a DoD Steering Group, the OSD CALS Office, and a lead organization in each Military Department and the Defense Logistics Agency. The DoD CALS Office has obtained the support of the National Institute of Standards and Technology in the selection and implementation of CALS standards. An Industry Steering Group has also been established to focus the work of key industrial associations and the defense contractor community in CALS implementation.

The CALS strategy provides a plan for phased implementation of CALS. Phase I will apply current computer technology in existing/emerging DoD and industry systems for key logistic and design applications. Phase II will involve broad-based DoD and industry system redesign to implement advanced technology across a wider range of applications during the early 1990's. DoD is currently developing core Phase I requirements. Demonstrations and prototypes will support Phase I implementation, while advanced technology R&D continues for Phase II.

Implementation of the CALS program will result in:

- o Design of more supportable weapon systems.
- o Increased productivity and reduced cost of weapon system acquisition and logistic support.
- o Improved timeliness and accuracy of logistic technical information.
- o Enhanced operational readiness of military forces.

During FY86 NIST recommended standards to OSD which would be applicable to the DoD environment.[1] These recommendations included CALS use of standards in the areas of product definition, graphics, text, and data management.

CALS support work in FY87 focussed on the following activities: Developing a CALS framework, Development Plan and Core Requirements package; providing technical support for standards development and implementation; and conducting workshops and meetings to promote dialogue with the Services, the Defense Logistic Agency, and industry. A major thrust was the completion of the initial documentation of the high-priority standards required for CALS implementation.[2]

During FY88, a number of efforts advanced the development of technology and standards in support of CALS. These efforts were organized into the areas of Text, Graphics, and Product Data.

Text: Work on text and graphics standards in the CALS publishing environment included technology assessments, development of application guidance, conformance test plans and a draft FIPS for ODA/ODIF. Additionally, a technology assessment and proposed conformance testing strategy were developed for page description languages.

Graphics: The CALS efforts in CGM were continued and included work in the graphics standards committees and the expansion and updating of the CALS CGM Application Profile. The Application Profile was developed into a draft military specification. The draft was carried through the needed review and comment process and was published as MIL-D-28003 in December 1988. In addition, work on Extended CGM, or CGEM for CALS application was initiated. Work in

---

[1]Kemmerer, S., Editor, "Final NBS Report for CALS, FY86," U.S. Department of Commerce, National Bureau of Standards, NBSIR 87-3566, May 1987.

[2]Kemmerer, S. Editor, "A Collection of Technical Studies Completed for the Computer-aided Acquisition and Logistic Support (CALS) Program, Fiscal Year 1987," U.S. Department of Commerce, National Bureau of Standards, NBSIR 88-3726, NBSIR 88-3727, NBSIR 88-3728, and NBSIR 88-3729, March 1988.

the area of raster graphics continued. A draft MILSPEC for raster was developed which was later published as MIL-R-28002. The need for standards for the interchange of large format tiled raster documents was identified, and related technical papers were published separately as an NIST Internal Report.[3]

Product Data: The use of the Information Resource Dictionary System, (IRDS, ANSI Standard X3.138-1988) was proposed as an integration and configuration management mechanism for the Product Data Exchange Specification (PDES).

[3]Spielman, F., Editor, "Standards for the Interchange of Large Format Tiled Raster Documents," U.S. Department of Commerce, National Bureau of Standards, NBSIR 88-4017, December 1988.

These three volumes are a collection of the final reports presented to the DoD CALS Office.[4] The collection is divided as follows:

VOLUME 1.

Text, Security, and Data Management

Text and Graphics Standards in the CALS Publishing Environment

ODA/ODIF Application Guidance

Federal Information Processing Standards Publication (Draft) on Document Application Profile for the Office Document Architecture (ODA) and Interchange Format Standard

ODA/ODIF Conformance Test Plan

PDLs: A Technology Assessment

SPDL Conformance Strategy

Security

Risk Management Tools: A Guide to Selection and Use

Computer Security Issues in the Application of New and Emerging Information Technologies

Data Management

Information Resource Dictionary System: An Integration Mechanism for Product Data Exchange Specification

Using the Information Resource Dictionary System for PDES

VOLUME 2:

Graphics, CGM MIL-SPEC

CGM Conformance Testing

Final Phase I.1 CGM MILSPEC

Extended CGM MILSPEC Planning

---

[4]The publishing of this collection of reports does not imply that the CALS Office has endorsed the conclusions or recommendations presented.

VOLUME 3:

Graphics, CGM Registration

CGM Registration in Support of CALS

The following additional publications were completed by NIST during FY87 under separate cover. They are available through NTIS.

CALS Workshop Proceedings: CALS EXPO '88 'Quality and Productivity Through Integration" A DoD/Industry NIST Conference 4-6 October 1988

MIL-HDBK-59, Military Handbook: Department of Defense Computer-aided Acquisition and Logistic Support (CALS) Program Implementation Guide

MIL-STD-1840A, Automated Interchange of Technical Information

MIL-D-28000, Military Specification: Digital Representation for Communication of Product Data: IGES Application Subsets

MIL-M-28001, Military Specification: Markup Requirements and Generic Style Specification for Electronic Printed Output and Exchange of Text

MIL-R-28002, Military Specification: Raster Graphics Representation in Binary Format, Requirements for

MIL-D-28003, Military Specification: Digital Representation for Communication of Illustration Data: CGM Application Profile

CONTRIBUTIONS

NIST would like to acknowledge the major technical contributors to this volume. In alphabetical order they are:

Daniel Benigni

George Carson

Graphics, CGM Registration

CGM Registration in Support of CALS

CALS SOW TASK 3.1.3

FINAL REPORT

CALS SOW TASK 3.1.3

CGM REGISTRATION IN SUPPORT OF CALS

TABLE OF CONTENTS

I. PURPOSE 2

II. BACKGROUND 2

III. DISCUSSION 3
1.0 Raster Input Extensions 3
1.1 Introduction 4
1.2 Requirements 4
1.3 Features of Interpress 6
1.4 Features of TIFF 7
2.0 Conclusion of Registration for Line Types, Hatch Styles and Text Fonts 10
3.0 Follow Through on Previous Registration Proposals 11
3.1 Presentation of Section Lining 14
3.2 Presentation of Line Conventions 15

IV. CONCLUSIONS AND RECOMMENDATIONS 17
1.0 Revised Registration Proposals 17

APPENDIX 1 -- Responses to Ballot Comments 151

CGM REGISTRATION IN SUPPORT OF CALS

I. PURPOSE

Define the CGM extensions needed to support CALS. Support identified extensions through the registration process (CALS SOW Task 3.1.3).

II. BACKGROUND

In the previous fiscal year (refer to NBSIR 88-3728, Vol.3) NIST (the National Institute of Standards and Technology, formerly NBS) presented a list of extensions needed for the CGM to meet CALS requirements. These extensions were to be implemented through the Graphics Registration Process. In addition, a number of areas were identified as requiring further investigation, including curves, text, raster images, named items and symbol libraries. As a final product, a significant number of registration proposals were developed and submitted to ANSI for formal processing through ISO.

This task was a continuation of that work. It consisted of three items:

1) **Item One:** Develop and define Raster Encoded Extensions and Raster Data Types to fully support technical and administrative publications in CALS, and follow through to ISO acceptance.

2) **Item Two:** Define registration requirements and develop registration proposals needed to work in areas of line types, hatch styles and text fonts, and follow through to ISO acceptance.

3) **Item Three:** Follow through to ISO acceptance proposals currently in the registration process as a result of last year's work, negotiating any requirement changes.

As a result of work on this task, computer graphics standards will, for the first time, be capable of supporting real-world applications in technical and administrative publications and engineering documentation. Once the registration proposals developed under this contract are processed, it will no longer be necessary to use IGES to exchange engineering drawings, CCITT facsimile standards to exchange raster data, or PostScript to describe graphics in office documents, if so desired. Furthermore, the work done under this task has been adopted by ANSI and ISO as the basis for extensions to the family of existing Computer Graphics standards. For example, Addendum 3 to the CGM contains precisely the extensions that this task has defined as being necessary. This addendum copies material that NIST wrote almost word for word in most cases.

During this fiscal year, the ISO/IEC JTC1/SC24 meeting was attended, where the Registration proposals which were originated last fiscal year were processed. The Special Working Group meeting on the Font Architecture standard (DIS 9541) was also attended, and contributions to the development of a substantially revised architecture were made.

III. DISCUSSION

1.0 Raster Input Extensions

A set of raster extensions to the CGM that equip it with descriptive capabilities that are either substantially equivalent to or better than those in CCITT Group 3/4 facsimile, PostScript, Interpress, and TIFF have been developed. These extensions make use of the power already inherent in the available specification modes of the CGM and are both considerably simpler and more compatible with existing computer graphics standards than those developed by the CGM group of ANSI/ISO and described in the draft CGM Addendum 3 (SC24 N 52). This was accomplished by:

1) Using the P ,Q ,and R parameterization of Cell Array to provide the required pel path and line progression capabilities;

2) Using the available scaling mode of the CGM to carry any pel spacing and spacing ratio data by implication through specification of metric mode with a scale factor equating VDC units to SMUs;

3) Using the nx and ny parameterization of Cell Array to provide the required number of pels per line and number of lines, rather than inventing unnecessary new concepts;

4) Using existing CGM clipping rectangle features to designate sub-areas of tile pel array, rather than inventing an unnecessary new element.

Further, these extensions go considerably beyond those of the extended metafile work, in that they:

1) Correctly specify allowable T.4 and T.6 coding variants;

   Include photogrammetric data needed to properly image (print) and/or edit raster data.

The registration proposals generated to add these capabilities are:

1) Pel Array GDP;
2) Set Direct Colour Response escape;
3) Set Indexed Colour Response escape.

The following paragraphs explain the rationale for the selection of these extensions.

1.1 Introduction

To help derive a set of "user requirements" for raster (image) data extensions to computer graphics standards, the features of two well-developed standards with similar purposes were studied. They are the Tag Image File Format (TIFF) Specification (Version 5.0, 8 August 1988) and the Xerox Raster Encoding Standard (XNS Standard 178506, June, 1985.) Each of these standards has a publicly available specification and has been used extensively for a variety of applications for many years. The purpose of the next few paragraphs is to present the results of an evaluation of the features of two raster (image) data exchange and storage formats and show how they relate to similar proposed extensions to computer graphics standards.

1.2 Requirements

Based upon the documents reviewed, the following set of requirements for raster data were developed. These are not listed in any particular order. The extensions developed herein meet all of these requirements to a sufficient degree to make the CGM a usable raster data exchange and storage mechanism for CALS.

1. Accommodate images of different sizes.

2. Accommodate color capabilities including:
    a. bi-level
    b. gray-scale monochrome
    c. color

3. Accommodate images of different resolutions.

4. Accommodate images of different intensity quantizations.

5. Accommodate images of different encodings, including:
    a. CCITT T.4
    b. CCITT T.6
    e. "bitmap encoding" as defined in DIS 8613/7.

6. Accommodate pel paths of at least 0, 90, 180, and 270 degrees relative to the X axis.

7. Accommodate line progression of at least 90 and 270 degrees relative to the X axis.

8. Accommodate the use of different compression techniques, including:
   a. CCITT T.4
   b. CCITT T.6
   c. a simple compression technique for color photographic images

9. Accommodate different values of pel spacing (density), including absolute values of 200 pels per 25.4 mm and 1728 pels per 215mm.

10. Allow for the specification of absolute pel spacing and line spacing in metric units.

11. Allow for the specification of relative pel spacing and line spacing virtual units.

12. Accommodate different values of line spacing, including absolute values of at least 3.85 and 7.7 lines/mm.

13. Allow simple images to be encoded with little overhead.

14. Include all important features of the Tiled Raster Interchange Format (TRIF), version 1.0, as special cases, allowing its eventual replacement by an extended CGM standard.

15. Include all important features of the Raster Graphics Content Architecture of ODA (ISO/DIS 8613, Part 7) as special cases, allowing its eventual replacement by an extended CGM standard.

16. Include all important features of the Tag Image File Format (TIFF), version 5.0, as special cases.

17. Include all important features of the Cell Array primitive of existing graphics standards as special cases.

18. Allow the specification of any rectangular sub-area of a pel array as a "clipped pel array."

19. Allow the specification of minimum and preferred image sizes in absolute metric units.

20. Support the transfer of information about an image that is required for editing.

21. Support the transfer of information about an image that is required for printing/display.

22. Support a wide range of color models.

23. Allow new color models and encoding techniques to be added without interfering with the interpretation of existing files.

1.3 Features of Interpress

The table below lists the features of Interpress in terms of its operators and options for each (where applicable) and shows how each of them relates to the CGM extensions that have been defined as a result of this task. Most Interpress operators have a direct correspondence in the extended CGM. A few elements are not needed due to the nature of the CGM.

Feature	Description/Realization in CGM Extensions
imageScale	can be realized with scaling mode and metric scaling factor
xDimension	nx parameter of Pel Array GDP
yDimension	ny parameter of Pel Array GDP
maskImage	not supported; probable future CGM extension
colorImage	pel array data
colorOperator	
GrayLinear	Indexed Colour Response escape
GrayDensity (logarithmic)	Indexed Colour Response escape
GrayVisual (power law)	Indexed Colour Response escape
Map (pseudo-color)	Direct Colour Response escape
BuildMap	Direct Colour Response escape
imageProperties	
identification	
name	metafile descriptor
creationTime	metafile descriptor
modTime	not supported; the local file system is the proper location to store this data, not the CGM itself

Feature	Description/Realization in CGM Extensions
scan characteristics	all the data below could be carried in the CGM as Application Data
rasterSource	name of the scanner or raster generator
rasterSourceID	usually 48 bit Ethernet address
graphicsSource	the name of the device or system that created the image rom in which the raster was created
imageType	Text, line copy, continuous-tone, half-tone, solid
scanTransformation	describes how the image was scanned
image processing	not supported; probable future CGM extension
imageAperature	circular, elliptical, square, rectangular, other, not applicable
maskAperature	
imageThresholdPattern	described any halftoning or thresholding used to create the image
image statistics	not supported; could be carried as Application Data
compression factor	
imageHistogram	
image description	not supported; could be carried as Application Data
narrative comments	
status	
user identification	not supported; could be carried as Application Data
user identification	
signature	
encoding	
packed	equivalent to encoding of current Cell Array in the CGM and the bitmap encoding of 8613/7.
CCITT-4	
Adaptive	not supported
Compressed	obsolete; not supported

1.4 Features of TIFF

The table below lists the features of TIFF in terms of its Tag Elements and options for each (where applicable) and shows how

each of them relates to the CGM extensions defined herein. Most TIFF elements have a direct correspondence in the extended CGM. A few elements are not needed due to the nature of the CGM.

Feature	Description/Realization in CGM Extensions
Basic Fields	
BitsPerSample	local colour precision parameter of Pel Array GDP
ColorMap	Colour Table
ColorResponseCurves	Direct Colour Response escape
Compression	compression parameter of Pel Array GDP none CCITT Group 3, 1 dimensional LZW
GrayResponseCurve	Indexed Colour Response escape
GrayResponseUnit	Indexed Colour Response escape
ImageLength	ny parameter of Pel Array GDP
ImageWidth	nx parameter of Pel Array GDP
NewSubfileType	not directly supported; could do with external data a reduced resolution version of another image a single page of a multi-page image this is a transparency mask for another image
Photometric interpretation	can be realized by colour specification modes bilevel/grayscale with 0 imaged as white bilevel/grayscale with 0 imaged as black RGB
Palette Color	direct colour
Transparency mask	not supported; probable future CGM extension
PlanarConfiguration	not applicable; CGM colour coding modes preempt this

Feature	Description/Realization in CGM Extensions
Predictor	used with LZW compression; incorporated in our compression options
ResolutionUnit	can be realized with scaling mode and metric scaling factor
RowsPerStrip	ny parameter of Pel Array GDP; each strip is encoded as a separate pel array in the metafile; each may have a different number of rows, thereby offering more flexibility than TIFF
SamplesPerPixel	not needed; inherent in colour specification and coding in the CGM
StripByteCounts	not needed
StripOffsets	not needed
XResolution	can be realized with scaling mode and metric scaling factor
YResolution	can be realized with scaling mode and metric scaling factor
Informational Fields	
Artist	could be done with External Data
DateTime	metafile description
HostComputer	could be done with External Data
ImageDescription	metafile description
Make	could be done with External Data
Model	could be done with External Data
Software	could be done with External Data

Feature	Description/Realization in CGM Extensions
Facsimile Fields	
Compression CCITTGroup3 CCITTGroup4	compression parameter of Pel Array GDP
Group3Options	compression parameter of Pel Array GDP 2-dimensional coding uncompressed mode fill bits force EOL to byte boundaries
Group4Options	compression parameter of Pel Array GDP uncompressed mode
Document Storage and Retrieval Fields	
DocumentName	could be done with External Data
PageName	could be done with External Data
PageNumber	could be done with External Data
XPosition	could be done with External Data
YPosition	could be done with External Data

2.0 Conclusion of Registration for Line Types, Hatch Styles and Text Fonts

Most of the additional registered items developed during this period are described below in Section IV. Although the intention was to prepare proposals to register enough additional text fonts to support the minimum requirements of IGES and Y14.2M-1979 and for office document exchange, the entire SC18 font architecture and its relationship to character sets has undergone significant revision over the last month. These changes occurred at the SWG meeting on DIS 9541 that was held in London in late August. A compromise acceptable to SC18, SC2, and SC24 was developed.

As a result of this compromise, graphics standards will point to the same AFII (Association for Font Information Interchange) register as used by SC18 for "glyphs" and "glyph collections". (A glyph is somewhat like a character, only glyphs can include stylistic variations of purely typographic, not linguistic, significance. A glyph collection is like a font.) The association between character codes and glyph collections (fonts) will be made through the registration process for graphical items. (Previously, it had been agreed not to use this registration process to register "text font usage", and to wait

instead for the SC18 registration process to become effective. Unfortunately, SC18 has decided not to register any associations between glyph collections (fonts) and character codes.)

Due to the additional complexity of preparing text registration proposals under this new scheme, and its newness and lack of thorough review by all SC18 national bodies, make it unwise to attempt such registration for several more months.

One area where additional registered items are urgently needed is the area of text. Although NIST expects to eventually develop registered extensions based on the SC18 work (DP 9541), that work is not yet stable enough to use as a basis for extensions. Furthermore, such massive extensions to the text model in computer graphics standards is beyond the scope of this task. In the interim, a set of eight escape functions to meet the immediate needs of CALS applications are proposed, complete with language bindings to expedite their processing. This work will, of course, be reviewed by X3H34.

These escapes define additional text attributes and give a way to turn them on and off. They are:

Set Underline Text
Set Bold Text
Set Italic Text
Set Outline Text
Set Shadow Text
Set Condensed Text
Set Extended Text
Set Fully Justified Text

To see why these extensions are urgently needed, consider this typical situation. A picture contains eight fonts with the 7 "attributes" named above available for use in any combination (e.g. bold italic outline Times.) This requires a "font list" in the CGM that contains 40,320 attribute combinations for each of 8 fonts, for a total of 322,560 items! This is clearly unworkable. A future version of MIL-D-CGM will list only the font family name in the font list (allowable names would be those that are proposed for registration during next fiscal year) and will require the use of the escapes proposed to set additional font "style attributes." However, it is still premature to consider their inclusion in the upcoming version of MIL-D-CGM.

3.0 Follow Through on Previous Registration Proposals

The chart below gives the status of each registration proposal which is being sponsored as a result of this task and its current status. The majority of the previous submittals have completed

ANSI processing and are awaiting ISO letter ballot. Due to the work done at the SC24 Tucson meeting this month, they are expected to pass without serious problems. The work at that meeting lead to agreement with among ISO National Bodies that the line types and hatch styles would be given "generic" names, reflecting the fact that they may have other uses beyond the specific ones currently envisioned.

Several proposals were dropped due to lack of understanding by the ANSI committee. These have been re-submitted here as a result of this task, and should be balloted by X3H3 as soon as possible. Responses to ballot comments needed by the X3H3 registration task group have been prepared and are included as Appendix 1.

The name of one proposal (Bezier Curve) was revised and three new proposals were created during this task. The new proposals were devised to separate line attributes from edge attributes. This was done at the suggestion of the CGM group, who suggested that the proposals might not pass ISO letter ballot otherwise. Modifications to the other proposals that were required to respond to comments received on the last letter ballot have also been made. This took considerable effort in many cases. NIST also attempted to align the proposals as closely as possible with CGM addendum 3. This was not too difficult since CGM addendum 3 is based almost word for word on NIST registration proposals, with only a few, mostly inconsequential changes made.

The language bindings prepared by X3H34 for the Rational B-spline and Parametric Spline proposals were incomplete. Many other bindings contained what appeared to be serious technical mistakes. NIST completed the unfinished bindings and corrected the incorrect ones. These bindings will be referred to X3H34 for review.

Status of Registration Proposals

Class of Item	Name	Status
Linetype	User-specified dash pattern	included with this report
Linetype	Single arrow	SC24 Letter Ballot
Linetype	Single dot	SC24 Letter Ballot
Linetype	Double arrow	SC24 Letter Ballot
Linetype	Stitch line	SC24 Letter Ballot
Linetype	Chain line	SC24 Letter Ballot
Linetype	Center line	SC24 Letter Ballot
Linetype	Phantom line	SC24 Letter Ballot
Linetype	Break line - style 1	SC24 Letter Ballot
Linetype	Break line - style 2	SC24 Letter Ballot
Linetype	Hidden line	SC24 Letter Ballot

Status of Registration Proposals

Class of Item	Name	Status
Hatchstyle	Cast iron and general use for all materials	SC24 Letter Ballot
Hatchstyle	Steel	SC24 Letter Ballot
Hatchstyle	Bronze, brass, copper, and compositions	SC24 Letter Ballot
Hatchstyle	White medal, zinc, lead, babbitt and alloys	SC24 Letter Ballot
Hatchstyle	Magnesium, aluminum, and aluminum alloys	SC24 Letter Ballot
Hatchstyle	Rubber, plastic, and electrical insulation	SC24 Letter Ballot
Hatchstyle	Cork, felt, fabric, leather, and fibre	SC24 Letter Ballot
Hatchstyle	Sound insulation	SC24 Letter Ballot
Hatchstyle	Thermal insulation	SC24 Letter Ballot
Hatch style	Titanium, and refractory material	SC24 Letter Ballot
Hatchstyle	Concrete	SC24 Letter Ballot
Hatchstyle	Marble, slate, glass, porcelain, etc.	SC24 Letter Ballot
Hatchstyle	Earth	withdrawn
Hatch style	Rock	SC24 Letter Ballot
Hatchstyle	Sand	SC24 Letter Ballot
Hatchstyle	Water and other liquids	included with this report
Hatchstyle	With grain wood	included with this report
Hatchstyle	Across grain wood	included with this report
Escape	Set dash	included with this report
Escape	Set line miter limit	included with this report
Escape	Set line cap	included with this report
Escape	Set line join	included with this report
Escape	Set edge miter limit	included with this report
Escape	Set edge cap	included with this report
Escape	Set edge join	included with this report
Escape	Set conic arc transformation matrix	included with this report
Escape	Set direct colour response	included with this report
Escape	Set indexed colour response	included with this report
GDP	Conic arc	included with this report
GDP	Parametric spline curve	included with this report
GDP	Rational B-spline curve	included with this report
GDP	Cubic Bezier curve	included with this report
GDP	Pel array	included with this report

3.1 Presentation of Section Lining

The following table summarizes how various section lining symbols used in engineering drawings should be "presented" using hatch styles in Computer Graphics standards. NIST has used the names under which the proposals were originally submitted, but these should be replaced by the internationally registered names when they are assigned by the Registration Authority. These, too, will become part of some future version of MIL-D-CGM, but it is still premature for their inclusion at this time.

Section Lining Symbols	Presentation as Hatchstyles in Graphic Standards
Cast iron and general use for all materials	Cast iron and general use for all materials (Note: This is similar tc hatch index 3, but has additional rendition requirements beyond those specified in the graphics standards)
Steel	Steel
Bronze, brass, copper, and compositions	Bronze, brass, copper, and compositions
White medal, zinc, lead, babbitt and alloys	White medal, zinc, lead, babbitt and alloys (Note: This is similar to hatch index 6, but has additional rendition requirements beyond those specified in the graphics standards)
Magnesium, aluminum, and aluminum alloys	Magnesium, aluminum, and aluminum alloys
Rubber, plastic, and electrical insulation	Rubber, plastic, and electrical insulation
Cork, felt, fabric, leather, and fibre	Cork, felt, fabric, leather, and fibre
Sound insulation	Sound insulation
Thermal insulation	Thermal insulation
Titanium, and refractory material	Titanium, and refractory material

Section Lining Symbols	Presentation as Hatchstyles in Graphic Standards
Electric windings, electromagnets, resistance, etc.	uses hatch index 5, horizontal/vertical crosshatch
Concrete	Concrete
Marble, slate, glass, porcelain, etc.	Marble, slate, glass, porcelain, etc.
Earth	render as a combination of the Rock hatch index and the Sand hatch index
Rock	Rock
Sand	Sand
Water and other liquids	Water and other liquids
With grain wood	With grain wood
Across grain wood	Across grain wood

3.2 Presentation of Line Conventions

The following table summarizes how various line conventions are used in engineering drawings should be "presented" using linetypes in Computer Graphics standards. NIST has used tile names under which we originally submitted the proposals, but these should be replaced by the internationally registered names when they are assigned by the Registration Authority. In this table, the "thin" and "thick" linewidths refer to the two widths required by paragraph 3.1 of ANSI Y14.2M-1979. These, too, will become part of some future version of MIL-D-CGM, but it is still premature for their inclusion at this time.

ine Conventions	Presentation as Linetypes in Graphic Standards
isible lines	presented as thick line in solid linetype
idden lines	Hidden line (Note: similar to dash linetype, only with additional rendition requirements)
ection lines	presented as thin line in solid linetype
enter lines	Center linetype

Line Conventions	Presentation as Linetypes in Graphic Standards
Symmetry lines	Center linetype
Dimension lines	presented using one or two lines; if two lines are used, each ends in a Single arrow; if a single line is used (with the dimension "text" outside the line), a Double arrow type is used
Extension lines	Single arrow linetype
Leaders	Single dot or Single arrow linetype
Cutting-plane lines	presented as a set of three graphical lines, related as described in paragraph 3.7 of ANSI Y14.2M-1979; two are solid thick lines with Single arrow style (indicating direction of sight); the other is a thick line using Hidden linetype
Viewing plane lines	presented as a set of three graphical lines, related as described in paragraph 3.7 of ANSI Y14.2M-1979; two are solid thick lines with Single arrow style (indicating direction of sight); the other is a thick line using Phantom linetype
Break lines	both styles of break line specified in paragraph 3.7 of ANSI Y14.2M- 1979 are allowable presentation styles; lines as described in 3.8(a) are presented using Break line - style 1; lines as described in 3.8(b) are presented using Break line-style 2
Phantom lines	Phantom line linetype
Stitch lines	Stitch line linetype
Chain lines	Chain line linetype

IV. CONCLUSIONS AND RECOMMENDATIONS

It must be stressed that timely processing of the registration proposals developed this fiscal year requires sponsorship through the ISO process and response to requests for changes as the result of standards committee ballots.

Successful completion of the necessary text font usage registration proposals should be undertaken as soon as possible. Without these proposals, the other items developed thus far will be less useful as high-quality text cannot be transferred in metafiles.

1.0 Revised Registration Proposals

The following pages describe all the revised registration proposals as a result of this task.

Proposal Number:	1

Presentation date of proposal:	10 April 1987
Sponsoring Authority:	ANSI
Class of Graphical Item:	LINETYPE
Name:	user-specified dash pattern

Description

The user-specified dash pattern linetype consists of alternating dashes and spaces as specified in the current user-specified dash pattern. This linetype is intended for use in high-quality graphical applications where the user of the standard maintains precise control over the manner in which the linetype is rendered by the use of individually specified attributes. Although its use is not precluded in applications that choose to use bundled attributes, the intent of the user to exercise a high degree of control over the rendition of graphical output will be compromised, especially in metafile applications.

Additional Comments

This registration proposal is accompanied by a proposal to register an escape function - Set Dash - that defines the current user-specified dash pattern. It is intended that these proposals be processed together.

This linetype is not intended to be used as an edgetype.

Justification for Inclusion

User specified linetypes are needed to support the requirements of office document exchange and publishing. They are commonly found in widely available proprietary graphics systems.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered linetype to supplement those defined in 5.4.1.
2) ISO 8632 (CGM) - Specifies a registered linetype to supplement those defined in 5.7.2.
3) ANSI Y14.2M-1979 - Line Conventions and Lettering.

Proposal Number:	36

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Dash

Description

This escape function sets the value for the user-specified (registered) linetype. This pattern is used during subsequent interpretation of graphical primitives that use linetype attributes. See attached sheet for additional details.

Additional Comments

None.

Justification for Inclusion

Required to support the user-specified dash pattern linetype. User specified line types are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing. The linetype capabilities proposed here are adapted from those in the PostScript language, which was developed by Adobe Systems Incorporated, and whose specification has been placed in the public domain.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Dash sets a dash pattern state value in the graphics state, controlling the dash pattern used during subsequent interpretation of graphics primitives that are drawn with the registered linetype value of "user-specified dash pattern" (linetype TBD). If the array of *dash pattern lengths* is empty (i.e, the *number of lengths* is zero), the linetype is equivalent to solid. This is the default value. If the array of *dash pattern lengths* is not empty, the affected primitives are drawn with dashed lines whose pattern is given by the elements of the array, which must be non-negative numbers and not all zero.

The elements of the array of *dash pattern lengths* are interpreted in sequence as relative distances along the primitive. These distances alternately specify the length of a gap between dashes. The contents of the array are used cyclically, that is when the end of the array is reached, the pattern starts over at the beginning.

Dashed lines wrap around curves and corners just as solid lines do. The ends of each dash are treated with current line cap, corners within a dash are treated with current line join. No measures are provided to coordinate the dash pattern with features of an output primitive.

The *offset* value may be thought of as the "phase" or the dash pattern relative to the start of the path. It is interpreted as a distance into the dash pattern at which the pattern should be started. Before beginning output of the dash pattern, the elements of the array of *dash pattern lengths* are cycled through, and the distances of alternating dashes and gaps added up, but without generating any output. When the *offset* distance into dash pattern has been reached, the primitive is drawn (from its beginning) using the dash pattern from the point that has been reached.

When *continuity* is set to *restart*, each portion of a primitive (e.g. each line segment within a polyline) is treated independently; i.e. the dash pattern is restarted (and *offset* applied) at the beginning of each portion. When *continuity* is set to *continuous*, the dash pattern is not restarted in going from one portion of a primitive to the next.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

The elements of the array of dash pattern lengths are interpreted in sequence as distances in VDC units along the primitive. The offset value is in VDC units. A functional description of the Set Dash escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
offset (VDC)
continuity (one of: restart, continuous) (E)
list of lengths (nVDC)
dash pattern lengths array

Items for Data Record:

If VDC TYPE integer was selected:

Integer IL	3 + number of lengths
Integer IA(1)	offset
Integer IA(2)	continuity
Integer IA(3)	number of lengths
Integer IA(4)	first length
Integer IA(5)	second length
...	
Integer IA(2+number of lengths)	last length
Integer RL	0
Integer SL	0

If VDC TYPE real was selected:

Integer IL	2
Integer IA(1)	number of lengths
Integer IA(2)	continuity
Integer RL	1 + number of lengths
Real RA(1)	offset
Real RA(2)	first length
Real RA(3)	second length
...	
Real RA(1+number of lengths)	last length
Integer SL	0

Data Record Description:

The parameters define the offset, continuity, number of dash pattern lengths, and the dash pattern lengths.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

The set dash escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
continuity (RESTART, CONTINUOUS)		E
number of lengths		I
dash pattern lengths array		nxR
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(CONT, DIMLEN, OFFSET, LEN)

Input Parameters:

INTEGER CONT          continuity (RESTART, CONTINUOUS)
INTEGER DIMLEN        dimension of the dash pattern lengths
                      array
REAL OFFSET           offset into dash pattern
REAL LEN(DIMLEN)      dash pattern lengths array

Output Parameters:

NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL            2
INTEGER IA( 1 )       continuity (RESTART, CONTINUOUS)
INTEGER IA( 2 )       number of dash pattern lengths
INTEGER RL            1+ number of dash pattern lengths
REAL RA( 1 )          offset
REAL RA( 2 )          first length
REAL RA( 3 )          second length
...
REAL RA(1 + number of dash pattern lengths)
                      last length
INTEGER SL            0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL            0
INTEGER RL            0
INTEGER SL            0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEscapeContinuity = (GVEscapeRestart, GVEscapeContinuous);

GREscapeDataIn = RECORD
                 CASE EscapeId : GTEscapeDataTag of
                      1:(
                           Continuity     : GEscapeContinuity;
                           NumberLengths  : INTEGER;
                           Patterns       : REAL );
                  END;

GREscapeDataOut = RECORD
                  CASE EscapeID : GTEscapeDataTag of
                       1: () ;              (*Null Record*)
                  END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_DASH" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a user specified dash pattern.
--Data types ESCAPE_ID and ESCAPE_FLOAT are defined in package
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type CONTINUITY_CHOICE is (RESTART,CONTINUOUS);
  type DASH_PATTERN_LENGTHS_ARRAY is array
       (SMALL_NATURAL range <>)of ESCAPE_FLOAT:
  type SET_DASH_DATA_RECORD(NUMBER_OF_LENGTHS:SMALL_NATURAL:=())is
    record
      CONTINUITY                :in CONTINUITY_CHOICE:
      DASH_PATTERN_LENGTHS      :in DASH_PATTERN_LENGTH_ARRAY
                                   (1..NUMBER_OF_LENGTHS);
    end record;

procedure SET_DASH
       (ESCAPE_IDENTIFIER  :in ESCAPE_ID;
        DASH_RECORD        :in SET_DASH_DATA_RECORD);

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

Proposal Number:	37

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape	Function Identifier:	Set Line Cap

Description

This escape function sets a value for the current line cap. This value is to determine the shape put at the ends of portions of lines and curves during subsequent interpretation of graphical primitives that use linetype attributes. See attached sheet for additional details.

Additional Comments

None.

Justification for Inclusion

User specified line caps are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing. The linetype capabilities proposed here are adapted from those in the PostScript language, which was developed by Adobe Systems Incorporated, and whose specification has been placed in the public domain.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Line Cap sets the current line cap value in the graphics state to the value specified by the *line cap indicator*. This establishes the shape to be put at the ends of open subportions of graphics output primitives whose components are lines or curves. The following line cap values are supported:

butt cap : the line is squared off at the endpoint; there is no projection beyond the endpoint.

round cap : a semicircular arc with diameter equal to the line width is drawn around the endpoint and filled. The drawn line thus projects beyond the endpoint.

projecting square cap : the line is squared off at a distance equal to half the line width beyond the endpoint.

Other values are reserved for future registration and standardization. The default value is butt cap.

Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Line Cap escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
line cap indicator (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	line cap indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the line cap index.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
line cap indicator	(BUTT, ROUND, PROJECTING SQUARE)	E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(LNCAP)

Input Parameters:
INTEGER LNCAP          line cap indicator (BUTT, ROUND,
                       PROJECTING SQUARE)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL              1
INTEGER IA( 1 )         line cap indicator (BUTT, ROUND,
                        PROJECTING SQUARE)
INTEGER RL 0
INTEGER SL 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL              0
INTEGER RL              0
INTEGER SL              0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEscapeLineEndType = (GVEscapeButt, GVEscapeRound,
                      GVEscapeProjectingSquare);

GREscapeDataIn = RECORD

            CASE EscapeID : GTEscapeDataTag of
                 1:(
                    Linecap       : GEscapeLineEndType);
            End;

GREscapeDataOut = RECORD
            CASE EscapeId  :  GTEscapeDataTag of
                1: () ; (*Null Record *)
            END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_LINE_CAP" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set line cap.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type LINE_CAP_INDICATOR_TYPE is (BUTT, ROUND, PROJECTING SQUARE);
     LINE_CAP_INDICATOR_VALUE     :in LINE_CAP_INDICATOR_TYPE;

procedure SET_LINE_CAP

       (ESCAPE_IDENTIFIER     :in ESCAPE_ID:
        LINE_CAP_INDICATOR_VALUE    :in LINE_CAP_INDICATOR TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS ESCAPE;
```

Proposal Number:	38

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Line Mitre Limit

Description

This escape function sets a value for the current line mitre limit. This value helps determine the shape put at corners between portions of lines and curves during subsequent interpretation of graphical primitives that use linetype attributes. Its purpose is to place a limit on how long a "spike" can emanate from the join of two portions of a line or curve primitive by "truncating" long mitre joins into bevel joins. See attached sheets for additional details.

Additional Comments

None.

Justification for Inclusion

User specified mitre limits are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing. The linetype capabilities proposed here are adapted from those in the PostScript language, which was developed by Adobe Systems Incorporated, and whose specification has been placed in the public domain.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Line Mitre Limit sets the current line mitre limit value in the graphics state to *mitre length specifier*, which must be a number greater than or equal to 1. Mitre limit is a dimensionless number that controls the treatment of corners between portions of line and curve output primitives when mitre joins have been specified (see **Set Line Join**). (**Set Edge Mitre Limit** and **Set Edge Join** provide similar capabilities for the edges of filled primitives.) When portions connect at a sharp angle, a mitre join results in a spike that extends well beyond the connection point. The purpose of the mitre limit is to cut off such spikes when when they become objectionably long.

At any given corner, the *mitre length* is the distance from the point at which the inner edges of the curve or line portions intersect to the point in which the outside edges of the portions intersect (i.e., the diagonal length of the mitre). This distance increases as the angle between the portions decreases. Whenever the ratio of the mitre length to the line width exceeds the line mitre limit parameter *and* is also greater than 1.415, a bevel is introduced at the join perpendicular to the angle bisector and at the mitre limit. (Note that this is *not*, in general equivalent to introducing a bevel join when this limit is reached. Introducing such a join causes discontinous behaviour, where small changes in the angle between the segments results in radically different appearances.) Whenever the ratio of tne line mitre length to the line width exceeds the line mitre limit parameter *and* is also less than or equal to 1.415 (that is, when the line mitre limit parameter is between 1 and 1.415 inclusive), a bevel join is implemented at the mitre limit.

The ratio of line mitre length to line width is directly related to the angle ϕ between the segments by the formula:

$$\textit{mitre length} / \textit{line width} = 1 / \sin(\phi/2)$$

Examples of line mitre limit values are: 1.415 cuts off miters (converts them to bevels) at angles less than 90 degrees, 2.0 cuts off miters at angles less than 60 degrees, and 10.0 cuts miters off at angles less than 11 degrees. The default value of line mitre limit is 10. Setting the line mitre limit to 1 cuts off miters at all angles so that bevels are always produced even when miters are specified. The lengths in the above formula must be in the same units (WC, VDC, etc.) and cancel out to yield a dimensionless line mitre limit value. Line mitre limit applies to line and curve elements.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Line Mitre Limit escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
mitre length (R)

Items for Data Record:

Integer IL	0
Integer RL	1
Real RA(1)	mitre length
Integer SL	0

Data Record Description:

The parameter defines the line mitre length.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Description (reference ISO 7942 GKS functional description)

This escape is applicable at level 0a of GKS. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
mitre length		R
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(MITRE)

Input Parameters:
REAL MITRE              mitre length

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL              0
INTEGER RL              1
REAL RA( 1 )            mitre length
INTEGER SL              0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL              0
INTEGER RL              0
INTEGER SL              0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GREscapeDataIn = Record
        CASE EscapeID : GTEscapeDataTag of
            1: (
                    MitreLength      : REAL );
        END;

GREscapeDataOut = RECORD
        CASE EscapeId : GTEscapeDataTag of
            1: () ;                  (*Null Record *)
        END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_LINE_MITRE_LIMIT" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set line mitre limit.
--Data types ESCAPE_ID and ESCAPE_FLOAT are defined in package GKS
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--


with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
     MITRE_LENGTH          :in ESCAPE_FLOAT;

procedure SET_LINE_MITRE_LIMIT
      (ESCAPE_IDENTIFIER   :in ESCAPE_ID;
       MITRE_LIMIT         :in MITRE_LENGTH;

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

Proposal Number:	39

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Line Join

Description

This escape function sets a value for the current line join. This value is to determine the shape put at corners between portions of lines and curves during subsequent interpretation of graphical primitives that use linetype attributes. See attached sheet for additional details.

Additional Comments

None.

Justification for Inclusion

User specified line joins are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing. The linetype capabilities proposed here are adapted from those in the PostScript language, which was developed by Adobe Systems Incorporated, and whose specification has been placed in the public domain.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Line Join sets the current line join state in the graphics state to *line join indicator*. This establishes the shape to be put at the corners between portions (line or curve segments) of line and curve graphical output primitives. The following line join values are supported:

mitre join : the outer edge of the two portions are extended until they meet at a point. (Note that the mitre limit value may affect the appearance of these joins.)

round join : a circular arc with diameter equal to the line width is drawn around the vertex between the adjoining segments and is filled in, producing a rounded corner.

bevel join : the meeting portions are finished with butt end cap and the resulting triangular notch is filled in.

Other values are reserved for future registration and standardization.

Join styles are significant only at points where consecutive portions of a line or curve connect at an angle; portions that meet or intersect fortuitously receive no special treatment.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Line Join escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
line join indicator (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	line join indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the line join index.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Description (reference ISO 7942 GKS functional description)

This escape is applicable at level 0a of GKS. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
line join indicator	(MITRE, ROUND, BEVEL)	E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(LNJOIN)

Input Parameters:
INTEGER LNJOIN          line join indicator (MITRE, .ROUND,
  BEVEL)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL              1
INTEGER IA( 1 )         line join indicator (MITRE, ROUND,
                        BEVEL)
INTEGER RL              0
INTEGER SL              0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL              0
INTEGER RL              0
INTEGER SL              0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEscapeLineJoin = (Mitre, Round, Bevel);

GREscapeDataIn = RECORD
       CASE EscapeId   :  GTEscapeDataTag of
          1:(
             Linejoin :  GEscapeLineJoin);
       END;

GREscapeDataOut = RECORD
       CASE EscapeId   :  GTEscapeDataTag of
           1:  () ;              (*Null Record*)
       END;
```

6) GKS Ada Language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

Registered ESCAPE's are in a library package name GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_LINE_JOIN" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set line join
--Data type ESCAPE_ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPES is
     type LINE_JOIN_INDICATOR_TYPE is (MITRE, ROUND, BEVEL);
     LINE_JOIN_INDICATOR_VALUE : in LINE_JOIN_INDICTOR_TYPE;

procedure SET_LINE_JOIN
        (ESCAPE_IDENTIFIER   :in ESCAPE_ID;
         LINE_JOIN_INDICATOR_VALUE:in LINE_JOIN_INDICTOR_TYPE );

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	40

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	GDP

GDP Identifier:	Cubic Bezier Curve.

Description

A Bezier cubic seldom is drawn using the four points specified. The curve starts at the first point and ends at the fourth point; the second and third point are used as control points. See the attached sheets for a detailed description.

Additional Comments

None.

Justification for Inclusion

Bezier curves are widely available in proprietary graphics system. They are needed to support the requirements of office document exchange and publishing.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.

2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered GDP. (See attached sheets).

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Cubic Bezier Curve adds a Bezier cubic curve between the first point, referred to here as (X_0, Y_0) and the fourth point (X_3, Y_3), using (X_1, Y_1) and (X_2, Y_2) as the Bezier cubic points.

The four points define the shape of the curve geometrically. The curve starts at (X_0, Y_0), it is tangent to the line from (X_0, Y_0) to (X_1, Y_1) at that point, and it leaves the point in that direction. The curve ends at (X_3, Y_3), it is tangent to the line from (X_2, Y_2) to (X_0, Y_0) at that point, and it approaches the point from that direction. The lengths of the lines (X_0, Y_0) to (X_1, Y_1) and (X_2, Y_2) to (X_3, Y_3) represent in some sense the "velocity" of the path at the endpoints. The curve is always entirely enclosed by the convex quadrilateral defined by the four points.

The mathematical foundation of a Bezier cubic curve is derived from a pair of parametric cubic equations:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + x_0$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + y_0$$

The cubic section produced by **Cubic Bezier Curve** is the path traced by $x(t)$ and $y(t)$ as t ranges from 0 to 1. The Bezier control points corresponding to this curve are:

$$x_1 = x_0 + c_x/3 \qquad y_1 = y_0 + c_y/3$$

$$x_2 = x_1 + (c_x + b_x)/3 \qquad y_2 = y_1 + (c_y + b_y)/3$$

$$x_3 = x_0 + c_x + b_x + a_x \qquad y_3 = y_0 + c_y + b_y + a_y$$

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Bezier curve generalized drawing primitive parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

point list(nP)
data record(D)

Items for Data Record:

Integer IL	0
Integer RL	0
Integer SL	0

Data Record Description:

The data record is empty.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This GDP is applicable at level 0a of GKS. It will use the polyline attribute set. The control points are transformed to NDC and the Bezier curve through those points is then drawn. A functional description of its input parameters is:

Name	Coordinate System	Values	Data Type
number of points		(4)	I
Bezier control points	WC		4xP
GDP identifier		as assigned	N
GDP data record:		empty	

Errors:

5 GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP
100 Number of points is invalid

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GDpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the GDP (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GDpqrs( N, PXA, PYA )

Input Parameters:
INTEGER N                number of points      (4)
REAL PXA(4), PYA(4)      Bezier control points
```

b) The following parameters are proposed for use when accessing this GDP through the GGDP function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL 0
INTEGER RL 0
INTEGER SL 0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
NumPoints = 4;
Points : GAPointArray;

GRGDPData = RECORD
       CASE GDPId : GTGDPDataTag OF
          1: (); (* null data record*)
       END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

Registered GDP's are in a library package named GKS_GDP. GKS Ada provides a data type package, GKS_TYPES which provides types declarations.

The binding for the "procedure BEZIER_CURVE" form (as defined in paragraph 4.1 of the GKS Ada language binding) of the GDP is:

```
--
--GDP function for a Bezier curve
--Data type GDP_ DATA_RECORD is defined in package GKS_GDP.
--GDP_ID and other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_GDP is
  type BEZIER_POINTS is new WC.POINT_ARRAY (1..4);

procedure BEZIER_CURVE
       (BEZIER_CONTROL_POINTS       :in BEZIER_POINTS;
        GDP_IDENTIFIER              :in GDP_ID);

--
--more GDP procedures can be inserted here
--

end GKS_GDP;
```

Proposal Number:	41

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	GDP

Specific Escape Function Identifier:	Conic Arc

Description

A bounded connected portion of a parent conic curve is drawn in a definition space and then transformed to world coordinates by the current conic arc transformation matrix. The intended realization of this output primitive is equivalent to that intended for the Conic Arc Entity of IGES Version 3.0. See the attached sheets for a detailed description.

Additional Comments

None.

Justification for Inclusion

Conic arcs are commonly found in proprietary graphics system. They are needed to support the requirements of office document and engineering drawing exchange. The conic arc capabilities proposed here are adapted from the IGES Version 3.0 specification.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.

2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.

3) *ISO 8651 (GKS Language Bindings) - Specifies a registered GDP. (See attached sheets).

*At present at the stage of draft. The status of this is provisional unit until this standard has been approved by ISO council.

Description:

[Note: The following description is adapted from the IGES 3.0 specification.]

A conic arc is generated which is defined as follows:

A conic arc is a bounded connected portion of a parent conic curve which consists of more than one point. The parent arc is either an ellipse, a parabola, or a hyperbola. The conic arc is defined by a start point, *P*, an end point, *Q*, and six parameters, *A,B,C,D,E,* and *F*. The conic arc itself is defined by the six parameters and the following equation:

$$A*X_t^2 + B*X_tY_t + C*Y_t^2 + D*X_t + E*Y_t + F = 0$$

where (X_t, Y_t) is an abstract Cartesian coordinate system called "definition space". In order for the conic arc to be processed correctly by the receiving system given the above presentation, the conic arc entity must be positioned such that each of its axes is parellel to either the X_t axis or Y_t axis. The arc is then positioned correctly in the appropriate computer graphics coordinate system by using the value of the current Conic Arc Transformation Matrix.

To determine the form of the conic arc, the quantities *Q1*, *Q2* and *Q3* are defined as follows:

$$Q1 = \text{determinant of } \begin{vmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{vmatrix}$$

$$Q2 = \text{determinant of } \begin{vmatrix} A & B/2 \\ B/2 & C \end{vmatrix}$$

$$Q3 = A + C$$

If $Q2>0$ and $(Q1*Q3)<0$, then arc is an ellipse.
If $Q2<0$ and $Q1<>0$, then the arc is a hyperbola.
If $Q2=0$ and $Q1<>0$, the the arc is a parabola.

In the case where the conic arc is elliptical, to distinquish the arc in question from its complement, the direction of the arc with respect to the definition space must be from start point to end point in a counterclockwise direction.

In the case where the conic arc is parabolic or hyperbolic, the parameterization defines a unique portion of the parabola or a unique portion of a branch of the hyperbola, thus, the direction is irrelevant.

The direction of the conic arc with respect to the space where it is drawn is determined by the original direction of the arc in definition space, in conjunction with the action of the current Conic Arc Transformation Matrix.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

The start and end points are in definition space and are always encoded as real numbers. The arc is transformed to VDC using the current Conic Arc Transformation Matrix and is then drawn. A functional description of the conic arc parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

point list(nP) - none
data record (D):
P
Q
A
B
C
D
E
F

Items for Data Record:

Integer IL	0
Integer RL	10
Real RA(1)	PX
Real RA(2)	PY
Real RA(3)	QX
Real RA(4)	QY
Real RA(5)	A
Real RA(6)	B
Real RA(7)	C
Real RA(8)	D
Real RA(9)	E
Real RA(10)	F
Integer SL	0
....	

Data Record Description:

The data record contains the start point and the end point (in definition space) and the six coefficients of the defining equation.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This GDP is applicable at level 0a of GKS. It will use the polyline attribute set. The arc is transformed to NDC using the current Conic Arc Transformation Matrix and then drawn. A functional description of its input parameters is:

Name	Coordinate System	Values	Data Type
number of points		(0)	I
GPD identifier		as assigned	N
GDP data record			
P(conic arc start point):			2xR
Q(conic arc end point):			2xR
(conic arc coefficients):			
A			R
B			R
C			R
D			R
E			R
F			R

Errors:

5 GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP

100 Number of points is invalid

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GDpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the GDP (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GDpqrs (P, Q, A, B, C, D, E, F)

Input Parameters:
REAL P(2), Q(2)        conic arc endpoints
REAL A, B, C, D, E, F  conic arc coefficients
```

b) The following parameters are proposed for use when accessing this GDP through the GGDP function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL          0
INTEGER RL          10
REAL RA( 1 )        PX
REAL RA( 2 )        PY
REAL RA( 3 )        QX
REAL RA( 4 )        QY
REAL RA( 5 )        A
REAL RA( 6 )        B
REAL RA( 7 )        C
REAL RA( 8 )        D
REAL RA( 9 )        E
REAL RA( 10 )       F
INTEGER SL          0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
NumPoints = 0;

GRGDPData = RECORD
       CASE GDPId   :  GTGDPDataTag OF
            1:(
              StartX : REAL;
              StartY : REAL;
              EndX   : REAL;
              EndY   : REAL;
              CoefA  : REAL;
              CoefB  : REAL;
              CoefC  : REAL;
              CoefD  : REAL;
              CoefE  : REAL;
              CoefF  : REAL);

       END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered GDP's are in a library package named GKS_GDP. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure CONIC_ARC" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the GDP is:

```
--
--GDP function for a conic arc.
--Data type GDP_DATA_RECORD is defined in package GKS_GDP.
--GDP_ID and other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_GDP is
  type CONIC_ARC_DATA_RECORD is new GDP_DATA_RECORD (0,10,0);

procedure CONIC_ARC
        GDP_IDENTIFIER              :in GPD_ID;
        CONIC_ARC_RECORD            :in CONIC_ARC_DATA_RECORD);

--The components of the CONIC_ARC_DATA_RECORD are the start and end
--points as well as the coefficients specified in the registration
--procedure.
--
--more GDP procedures can be inserted here
--
end GKS_GDP;
```

Proposal Number:	42

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Conic Arc Transformation Matrix

Description

This escape function sets a value of the transformation matrix needed to describe how a conic arc described by the conic arc GDP is moved from "definition space" to world coordinates (called "model space" in the IGES standard.) It is modelled on the Transformation Matrix Entity of IGES Version 3.0. See attached sheets for a detailed description.

Additional Comments

None.

Justification for Inclusion

Conic arcs are commonly found in proprietary graphics system. They are needed to support the requirements of engineering drawing exchange. Due to various numerical problems, such curves are best specified in a "definition space" and then transformed to their final location by applying a transformation matrix. This escape function is needed to supply values for the required "modelling" or transformation matrix. The capabilities proposed here are adapted from the IGES Version 3.0 specification, and are specilized to the case of two-dimensions.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.

2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.

*3) ISO 8651 (GKS Language Bindings) - Specifies a GDP. See attached sheets.

*At present at the stage of draft. The status of this is provisional unit until this standard has been approved by ISO council.

Description:

[Note: This escape is adapted from the Transformation Matrix Entity of the IGES version 3.0 specification.]

This escape is intended to work in conjunction with the conic arc generalized drawing primitive to transform the conic arc from definition space to drawing space. The **conic arc transformation matrix** transforms definition space point coordinates by means of a matrix multiplication. This transformation is performed by applying the following matrix multiplication to coordinates:

$$\begin{vmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \end{vmatrix} \times \begin{vmatrix} X_{in} \\ Y_{in} \\ 1 \end{vmatrix} = \begin{vmatrix} X_{out} \\ Y_{out} \end{vmatrix}$$

where $|R_{ij}|$ is the transformation matrix, (X_{in}, Y_{in}) is the corrdinate to be transformed and (X_{out}, Y_{out}) is the coordinate resulting from the transformation. Both the input and output coordinate systems are assumed to be orthogonal, Cartesian and right-handed.

[Note: IGES version 3.0 defines this transformation as a matrix multiplication followed by a vector addition (translation). To be consistent with the way transformations are defined in computer graphics standards, an equivalent homogeneous coordinate formulation is used here instead. To translate this form to the IGES form, simply use the first two columns of the above matrix as the IGES matrix and the last column as the "T-vector" values T1 and T2 in IGES.]

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

This matrix is used to transform a Conic Arc element from its definition space into VDC where it is drawn. A functional description of the Set Conic Arc Transformation Matrix escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
R_{11}
R_{12}
R_{13}
R_{21}
R_{22}
R_{23}

Items for Data Record:

Integer IL	0
Integer RL	6
Real RA(1)	R_{11}
Real RA(2)	R_{12}
Real RA(3)	R_{13}
Real RA(4)	R_{21}
Real RA(5)	R_{22}
Real RA(6)	R_{23}
Integer SL	0

Data Record Description:

The parameters define a 2X3 matrix used to transform a Conic Arc element from its definition space into VDC where it is drawn.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Description (reference ISO 7942 GKS functional description)

This escape is applicable at level 0a of GKS. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
(transformation matrix)		2X3XR
R11		R
R12		R
R13		R
R21		R
R22		R
R23		R

output data record:
none

Errors:

5 GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP
100 Number of points is invalid

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(R11, R12, R13, R21, R22, R23)

Input Parameters:
REAL R11                2X3 transformation matrix
REAL R12
REAL R13
REAL R21
REAL R22
REAL R23

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

INTEGER IL	0
INTEGER RL	6
REAL RA(1)	R11
REAL RA(2)	R12
REAL RA(3)	R13
REAL RA(4)	R21
REAL RA(5)	R22
REAL RA(6)	R23
INTEGER SL	0

Output parameters to the Unpack Data Record function (GUREC):

INTEGER IL	0
INTEGER RL	0
INTEGER SL	0

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GREscapeDataIn = RECORD
        CASE EscapeID : GTEscapeDataTag of
             1:(
                   MatrixReal11  : REAL;
                   MatrixReal12  : REAL;
                   MatrixReal13  : REAL;
                   MatrixReal21  : REAL;
                   MatrixReal22  : REAL;
                   MatrixReal23  : REAL);
         END;

GREscapeDataOut = RECORD
        CASE EscapeId : GTEscapeDataTag of
          1: () ;              (* Null Record*)
        END;
```

6) GKS Ada Language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_CONIC_ARC_TRANSFORMATION_ MATRIX" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set conic arc transformation.
--Data type ESCAPE_ID and ESCAPE_FLOAT are defined in package
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is

 type CONIC_TRANSFORM_MATRIX is array (1..2,1..3) of ESCAPE_FLOAT;

procedure SET_CONIC_ARC_TRANSFORMATION_MATRIX
       (ESCAPE_IDENTIFIER    :in ESCAPE_ID;
        CONIC_ARC_MATRIX     :in CONIC_TRANSFORM_MATRIX );

--The components of CONIC_ARC_DATA are the 2X3 transformation
--matrix specified in the registration proposal

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	43

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	GDP

GDP Identifier:	Parametric Spline Curve

Description

A planar (two dimensional) parametric spline curve is drawn. The intended realization of this output primitive is equivalent to that intended for the Parametric Spline Curve Entitty of IGES Version 3.0, with the restriction that the "Z polynomial" of the IGES standard be zero. See attached sheets for a detailed description.

Additional Comments

None.

Justification for Inclusion

Parametrics spline curves are commonly found in proprietary graphics system. They are needed to support the requirements of engineering drawing exchange. The capabilities proposed here are adapted from the IGES Version 3.0 specification, and are specilized to the case of two-dimensions.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.

2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered GDP. (See attached sheets).

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

A parametric spline curve as defined below is drawn. A parametric spline curve is a sequence of parametric polynomial segments. The curve is defined using the following parameters:

curve type (E)
H (degree of continuity) (I)
N (number of segments) (I)
T (break point list for polynomial) ((N+1)R)
X coordinate polynomial list (N sets of four)

A_X, B_X, C_X, D_X ((4*N)R)

Y coordinate polynomial list (N sets of four)

A_Y, B_Y, C_Y, D_Y ((4*N)R).

This parameterization is generalized to allow for the representation of many different parametric spline curves using this one element. The curve type parameter indicates the type of parametric curve as it was represented in the sending system before being converted to this generic form. The following curve types have been assigned:

1) linear
2) quadratic
3) cubic
4) Wilson-Fowler
5) modified Wilson-Fowler
6) B spline

Additional curve types are reserved for future registration and standardization.

The degree of continuity parameter, *H* indicates the smoothness, or continuity of the curve with respect to arc length. If *H=0*, the curve is continuous at all break points. If *H=1*, the curve is continuous and has slope continuity at all break points. If *H=2*, the curve is continuous and has both slope and curvature continuity at all break points.

The number of segment parameters, *N*, is the number of polynomial segments to be used to define the curve. Each segment is defined by a cubic polynomial in X and Y that is evaluated using the eight polynomial coefficients associated with that segment: $A_X, B_X, C_X, D_X, A_Y, B_Y, C_Y, D_Y$. Segment *i* is delimited by its breakpoints, *T(i) and T(i+1)*.

The coordinates of the points of the i^{th} segment of the curve are given by the following cubic polynomial equations. Note that coefficients *D*, or *C* and *D* will be zero if the polynomials are of degrees 2 or 1, respectively:

$$X(u) = A_X(i) + B_X(i)*s + C_X(i)*s^2 + D_X(i)*s^3$$
$$Y(u) = A_Y(i) + B_Y(i)*s + C_Y(i)*s^2 + D_Y(i)*s^3$$

where $T(i) <= u <= T(i+1), i=1,...,N$ and $s = u - T(i)$. In order to avoid degeneracy, for each *i* at least one of the six real coefficients $B_X, C_X, D_X, B_Y, C_Y, D_Y$ must be non-zero.

To enable determination of the terminate point and derivatives without computing the polynomials, the N^{th} polynomials and derivatives are evaluated at $u = T(N+1)$. These data, divided by the appropriate factorial (i.e. the second derivative divided by *2!*, the third by *3!*, etc.), are used as the $N+1^{st}$ or terminate point values.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

The parameters of the curve are transformed to VDC and the curve is drawn. A functional description of the parametric spline curve parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

point list - empty
data record (D): -

C (curve type) (Note 1) (E)
H (degree of continuity) (I)
N (number of segments) (I)
T (break point list for polynomial) ((N+1)R)
X coordinate polynomial list (N sets of four)

AX,BX,CX,DX ((4*N)R)

Y coordinate polynomial list (N sets of four)

AY,BY,CY,DY ((4*N)R).

Note 1: Curve type is one of (linear, quadratic, cubic, Wilson-Fowler, modified Wilson-Fowler, B spline)

Items for Data Record:

Integer IL	3
Integer IA(1)	C
Integer IA(2)	H
Integer IA(3)	N
Integer RL	(9N+1)R
Real RA(1)	T(1)
....	
Real RA(N+1)	T(N+1)
Real RA(N+2)	AX(1)
Real RA(N+3)	BX(1)
Real RA(N+4)	CX(1)
Real RA(N+5)	DX(1)
Real RA(N+6)	AX(2)
Real RA(N+7)	BX(2)
....	
Real RA(5N+2)	AY(1)
Real RA(5N+3)	BY(1)
....	
Integer SL	0

Data Record Description:

The data record contains the parameters needed to define the parametric spline curve: curve type, degree of continuity, number of segments, break point list, and the coefficients of the two polynomials that define the X and Y coordinate values, respectively, for the curve.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This GDP is applicable at level 0a of GKS. It will use the polyline attribute set. The parameters of the curve are transformed toi NDC and the curve is drawn. A functional description of its input parameters is:

Name	Coordinate System	Values	Data Type
number of points		(0)	I
GDP identifier		as assigned	N
GDP data record: see IGES attachments for definitions			
CTYPE		Note 1	I
H		(0,1,2)	I
NSEG			I
breakpoints, "T" values			(N+1)R
X coordinate polynomial coefficients			((4*N)R)
Y coordinate polynomial coefficients			((4*N)R).

Note 1: Curve type is one of (linear, quadratic, cubic, Wilson-Fowler, modified Wilson-Fowler, B spline)

Errors:

5 GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP
100 Number of points is invalid

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GDpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the GDP (pqrs is to be assigned by the Registraton Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GDpqrs( CTYPE, H, NSEG, T, X, Y )

Input Parameters:
INTEGER CTYPE          type of spline curve
INTEGER H              degree of continuity
INTEGER NSEG           number of polynomial segments
REAL T(NSEG+1)         break point list
REAL X(NSEG,4)         X coordinate polynomial coefficients
REAL Y(NSEG,4)         Y coordinate polynomial coefficients
```

b) The following parameters are proposed for use when accessing this GDP through the GGDP function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
Integer IL          3
Integer IA(1)       C
Integer IA(2)       H
Integer IA(3)       N
Integer RL          (9N+1)R
Real RA(1)          T(1)
...
Real RA(N+1)        T(N+1)
Real RA(N+2)        AX(1)
Real RA(N+3)        BX(1)
Real RA(N+4)        CX(1)
Real RA(N+5)        DX(1)
Real RA(N+6)        AX(2)
Real RA(N+7)        BX(2)
....
Real RA(5N+2)       AY(1)
Real RA(5N+3)       BY(1)
....
Integer SL          0
```

5) **Pascal language binding** (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEGDPCurveProperty5 = ( linear, quadratic, cubic,
                        Wilson-Fowler, modified Wilson-Fowler,
                        B spline);


GRGDPData = RECORD
        CASE GDPId : GTGDPDataTag OF
             1:(
                  CType   : GEGDPCurveProperty5;
                  H       : INTEGER;
                  NSeg    : INTEGER;
                  T       : array [1..(NSeg+1)] of REAL;
                  X       : array [1..(4*NSeg)] of REAL;
                  Y       : array [1..(4*NSeg)] of REAL);
      END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

Registered GDP's are in a library package named GKS_GDP. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure PARAMETRIC_SPLINE_CURVE" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the GDP is:

```
--
--GDP function for a parametric spline curve gap.
--Data type GDP_DATA_RECORD is defined in package GKS_GDP.
--GDP_ID and other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_GDP is
   type CURVE_TYPE is (linear, quadratic, cubic, Wilson-Fowler,
        modified Wilson-Fowler, B spline);
   type DEGREE_OF_CONTINUITY is new INTEGER range 0..2;
   type BREAKPOINT_ARRAY is array (SMALL_NATURAL range<>) of
        ESCAPE_FLOAT;
   type SPLINE_CURVE_DATA_RECORD (NUM_SEG : SMALL_NATURAL := 0)is
      record
       SPLINE_TYPE      : CURVE_TYPE;
       CONTINUITY       : DEGREE_OF_CONTINUITY;
       BREAKPOINTS      : BREAKPOINT_ARRAY (1..(NUM_SEG+1));
       POLYNOMIALS      : POLYNOMIAL_ARRAY (1..NUM_SEG,1..4);
      end record;

procedure PARAMETRIC_SPLINE_CURVE
    (GDP_IDENTIFIER              :in GDP_ID;
    SPLINE_CURVE_RECORD          :in SPLINE_CURVE_DATA_RECORD);

--The components of the SPLINE_CURVE_DATA_RECORD are those
--specified in the registration proposal.
--
--more GDP procedures can be inserted here
--
end GKS_GDP;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	44

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	GDP

GDP Identifier:	Rational B-Spline Curve

Description

A planar (two dimensional) rational spline B-spline curve is drawn. The intended realization of this output primitive is equivalent to that intended for the Rational B-Spline Curve Entitty of IGES Version 3.0, with the restriction that the "Z polynomial" of the IGES standard be zero. See attached sheets for a detailed description.

Additional Comments

None.

Justification for Inclusion

Rational B-spline curves are commonly found in proprietary graphics system. They are needed to support the requirements of engineering drawing exchange. The capabilities proposed here are adapted from the IGES Version 3.0 specification, and are specilized to the case of two-dimensions.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.

2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered GDP. (See attached sheets).

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

A rational B-spline curve as defined below is drawn. The curve is defined using the following parameters:

K (upper index of summation) (I)
M (degree of the basis functions) (I)
curve_open flag (one of: *open, closed*) (E)
equation_type flag (one of: *rational, polynomial*) (E)
periodic flag (one of: *non-periodic, periodic*) (E)
T (knot sequence) ((K+M+1)R)
W (weights) ((K+1)R)
P (control points) ((K+1)P)
start_param ,end_param (2R)

The parametric equation governing the definition of the rational B-spline curve is shown in the following expression:

$$G(t) = \frac{\sum_{i=0}^{K} W(i)P(i)b_i(t)}{\sum_{i=0}^{K} W(i)b_i(t)}$$

where $W(i)$ are the weights, $P(i)$ are the control points and b_i are the basis functions.

The B-spline basis functions, b_i, are all non-negative piecewise polynomials of degree *M*. *T* is a non-decreasing sequence of real numbers $T(-M),\ldots,T(0),..T(K+1)$. Each function b_i is supported by the knot sequence interval [T(i-M),T(i+1)]. Between any two adjacent knot values, T(j) and T(j+1), the corresponding basis function can be expressed as a single polynomial of degree *M*.

The curve itself is parameterized where:

$start\_param <= t <= end\_param,$
$T(0) < start\_param < end\_param <= T(N)$

Thus, for any parameter value *t* between $T(0)$ and $T(K+1)$, the sum of the basis functions satisfies the following identity:

$$\sum_{i=0}^{K} b_i(t) = 1.$$

If the beginning and ending points of the curve are identical, then the curve_open flag is set to *closed*, otherwise, it is set to *open*.

If all of the weights, *W*, are not equal, then the equation_type flag is set to *rational*. Otherwise, if all of the weights are equal, then all of the weights cancel, the denominators sum to one and the equation_type becomes *polynomial*.

Relationship to particular standard:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

The parameters of the curve are transformed to VDC and the curve is then drawn. A functional description of the rational B-spline parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

point list(nP) - contains the control points
data record (D):

K (upper index of sum) (I)
M (degree of the basic functions) (I)
curve_open flag (one of: open, closed) (E)
equation_type flag (one of: rational, polynomial (E)
periodic flag (one of: non-periodic, periodic) (E)
T (knot sequence) ((K+M+1)R)
W (weights) ((K+1)R)
P (control points) ((K+1)P)
start_param ,end_param (2R)

Items for Data Record:

Integer IL	5
Integer IA(1)	K
Integer IA(2)	M
Integer IA(3)	curve_open flag
Integer IA(4)	equation_type flag
Integer IA(5)	periodic flag
Integer RL	2K + M + 4
Real RA(1)	T(-M)
Real RA(2)	T(-M+1)
....	
Real RA(K+M+1)	T(K+1)
Real RA(K+M+2)	W(0)
....	
Real RA(2K+M+2)	W(K+1)
Real RA(2K+M+3)	start_param
Real RA(2K+M+4)	end_param
Integer SL	0

Data Record Description:

The data record contains the parameters that define the rational B-spline curve: the upper index of summation, the basis functions, the curve_open flag, the equation_type flag, the periodic flag, the knot sequence, the weights, and two parameters -- start_param and end_param -- that determine the beginning and end of the curve.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This GDP is applicable at level 0a of GKS. It will use the polyline attribute set. The control points are transformed to NDC and the curve is drawn. A functional description of its input parameters is:

Name	Coordinate System	Values	Data Type
number of points			I
control points	WC		nxP
GDP identifier		as assigned	N
GDP data record:			
K (upper index of summation)			I
M (degree of basis functions)			I
COFLAG		(open,closed)	E
ETFLAG		(rational,polynomial)	E
PFLAG		(non-periodic,periodic)	E
T			R
W			R
SPARAM, EPARAM			R

Errors:

5 *GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP*

100 *Number of points is invalid*

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GDpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the GDP (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
      SUBROUTINE GDpqrs( N, PXA, PYA, K, M, COFLAG, ETFLAG, PFLAG, T,
     +                   W, SPARAM, EPARAM)

      Input Parameters:
      INTEGER N                 number of control points
      REAL PXA(*), PYA(*)       control points
      INTEGER K                 upper index of summation
      INTEGER M                 degree of basis functions
      INTEGER COFLAG
      INTEGER ETFLAG
      INTEGER PFLAG
      REAL T( N+2M )            knot sequence
      REAL W( K )               weight
      REAL SPARAM,EPARAM        determine start and end points
```

b) The following parameters are proposed for use when accessing this GDP through the GGDP function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
      Integer IL              5
      Integer IA(1)           K
      Integer IA(2)           M
      Integer IA(3)           COFLAG
      Integer IA(4)           ETFLAG
      Integer IA(5)           PFLAG
      Integer RL              2K + M + 4
      Real RA(1)              T(-M)
      Real RA(2)              T(-M+1)
      ....
      Real RA(K+M+1)          T(K+1)
      Real RA(K+M+2)          W(0)
      ....
      Real RA(2K+M+2)         W(K+1)
      Real RA(2K+M+3)         SPARAM
      Real RA(2K+M+4)         EPARAM
      Integer SL              0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEGDPCurveProperty2 = (GVGDPOpen, GVGDPClosed);
GEGDPCurveProperty3 = (GVGDPRational, GVGDPPolynomical);
GEGDPCurveProperty4 = (GVGDPNonPeriodic, GDPPeriodic);

Points              :  GAPointArray;     (*control points*)

GRGDPData = RECORD
      CASE GDPId :  GTGDPDataTag OF
           1:(
              K                    : INTEGER;
              M                    : INTEGER;
              CurveOpenFlag        : GEGDPCurveProperty2;
              EquationTypeFlag     : GEGDPCurveProperty3;
              PeriodicFlag         : GEGDPCurveProperty4;
              T                    : REAL;
              W                    : REAL;
              StartParam           : REAL;
              EndParam             : REAL;)

      END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

a) Registered GDP's are in a library package named GKS_GDP. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure Rational_B_SPLINE_CURVE" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the GDP is:

```
--
--GDP function for a rational B_Spline curve.
--Data type GDP_DATA_RECORD is defined in package GKS_GDP.
--GDP_ID and other data types are defined in package GKS_TYPES.
--


with GKS-TYPES;
use GKS-TYPES;
package GKS_GDP is
   type CURVE_OPEN_TYPE is (OPEN, CLOSED);
   CURVE_OPEN_VALUE : in CURVE_OPEN_TYPE;

   type EQUATION_TYPE is (RATIONAL, POLYNOMIAL);
   EQUATION_VALUE : in EQUATION_TYPE;

   type PERIODIC_TYPE is (NON-PERIODIC, PERIODIC);
   PERIODIC_VALUE : in PERIODIC_TYPE;

   type KNOT_SEQUENCE_ARRAY is array (SMALL NATURAL range<>) of
      ESCAPE_FLOAT;

   type WEIGHT_SEQUENCE_ARRAY is array (SMALL NATURAL range<>) of
      ESCAPE_FLOAT;

   type RATIONAL_B_SPLINE_DATA_RECORD is
      record
        UPPER_INDEX_OF_SUM          : INTEGER;
        DEGREE_OF_BASIS_FUNCTIONS   : INTEGER;
        CURVE_OPEN_VALUE            : CURVE_OPEN_TYPE;
        EQUATION_VALUE              : EQUATION_TYPE;
        PERIODIC_VALUE              : PERIODIC_TYPE;
        KNOT_SEQUENCE               : KNOT_SEQUENCE_ARRAY;
        WEIGHT_SEQUENCE             : WEIGHT_SEQUENCE_ARRAY;
        START_PARAM                 : ESCAPE_FLOAT;
        END_PARAM                   : ESCAPE_FLOAT;
      end record;

procedure RATIONAL_B_SPLINE_CURVE
   (CONTROL_POINTS                  :in WC.POINT_ARRAY;
    GDP_IDENTIFIER                  :in GDP_ID;
    RATIONAL_B_SPLINE_DATA_RECORD   :in GDP_DATA_RECORD);

--The components of the RATIONAL_B_SPLINE_DATA_RECORD are those
--specified in the registration proposal.
--
--more GDP procedures can be inserted here
--
end GKS_GDP;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape	Function Identifier:	Set Edge Mitre Limit

Description

This escape function sets a value for the current edge mitre limit. This value helps determine the shape put at corners between portions of edges during subsequent interpretation of filled area graphical primitives. Its purpose is to place a limit on how long a "spike" can emanate from the join of two portions of an edge of a filled area primitive by "truncating" long mitre joins into bevel joins. See attached sheets for additional details.

Additional Comments

None.

Justification for Inclusion

User specified mitre limits are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing. The edgetype capabilities proposed here are adapted from those in the PostScript language, which was developed by Adobe Systems Incorporated, and whose specification has been placed in the public domain.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Edge Mitre Limit sets the current edge mitre limit value in the graphics state to *mitre length specifier*, which must be a number greater than or equal to 1. Mitre limit is a dimensionless number that controls the treatment of corners between portions of the edges of filled area output primitives when mitre joins have been specified (see **Set Line Join**). (**Set Line Mitre Limit** and **Set Line Join** provide similar capabilities for the edges of line and curve primitives.) When portions connect at a sharp angle, a mitre join results in a spike that extends well beyond the connection point. The purpose of the mitre limit is to cut off such spikes when when they become objectionably long.

At any given corner, the *mitre length* is the distance from the point at which the inner edges of the curve or line portions intersect to the point in which the outside edges of the portions intersect (i.e., the diagonal length of the mitre). This distance increases as the angle between the portions decreases. Whenever the ratio of the mitre length to the edge width exceeds the edge mitre limit parameter *and* is also greater than 1.415, a bevel is introduced at the join perpendicular to the angle bisector and at the mitre limit. (Note that this is *not*, in general equivalent to introducing a bevel join when this limit is reached. Introducing such a join causes discontinous behaviour, where small changes in the angle between the segments results in radically different appearances.) Whenever the ratio of the edge mitre length to the edge width exceeds the edge mitre limit parameter *and* is also less than or equal to 1.415 (that is, when the edge mitre limit parameter is between 1 and 1.415 inclusive), a bevel join is implemented at the mitre limit.

The ratio of edge mitre length to edge width is directly related to the angle ϕ between the segments by the formula:

$$\textit{mitre length} / \textit{edge width} = 1 / \sin(\phi/2)$$

Examples of edge mitre limit values are: 1.415 cuts off miters (converts them to bevels) at angles less than 90 degrees, 2.0 cuts off miters at angles less than 60 degrees, and 10.0 cuts miters off at angles less than 11 degrees. The default value of edge mitre limit is 10. Setting the edge mitre limit to 1 cuts off miters at all angles so that bevels are always produced even when miters are specified. The lengths in the above formula must be in the same units (WC, VDC, etc.) and cancel out to yield a dimensionless edge mitre limit value. Edge mitre limit applies to the edges of filled area primitives

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Edge Mitre Limit escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
mitre length (R)

Items for Data Record:

Integer IL	0
Integer RL	1
Real RA(1)	mitre length
Integer SL	0

Data Record Description:

The parameter defines the edge mitre length.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Description (reference ISO 7942 GKS functional description)

This escape is applicable at level 0a of GKS. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
mitre length		R
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(MITRE)

Input Parameters:
REAL MITRE              mitre length

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL              0
INTEGER RL              1
REAL RA( 1 )            mitre length
INTEGER SL              0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL              0
INTEGER RL              0
INTEGER SL              0
```

5) **Pascal language binding** (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GREscapeDataIn = Record
        CASE EscapeID : GTEscapeDataTag of
            1: (
                    MitreLength      : REAL );
        END;

GREscapeDataOut = RECORD
        CASE EscapeId : GTEscapeDataTag of
            1: () ;                  (*Null Record *)
        END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_EDGE_MITRE_LIMIT" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set edge mitre limit.
--Data types ESCAPE_ID and ESCAPE_FLOAT are defined in package GKS
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--


with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
     MITRE_LENGTH          :in ESCAPE_FLOAT;

procedure SET_EDGE_MITRE_LIMIT
      (ESCAPE_IDENTIFIER   :in ESCAPE_ID;
       MITRE_LIMIT         :in MITRE_LENGTH;

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Edge Cap

Description

This escape function sets a value for the current edge cap. This value is to determine the shape put at the ends of portions of edges during subsequent interpretation of filled area graphical primitives . See attached sheet for additional details.

Additional Comments

None.

Justification for Inclusion

User specified edge caps are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing. The edgetype capabilities proposed here are adapted from those in the PostScript language, which was developed by Adobe Systems Incorporated, and whose specification has been placed in the public domain.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Edge Cap sets the current edge cap value in the graphics state to the value specified by the *edge cap indicator*. This establishes the shape to be put at the ends of edge elements of filled primitives. The following edge cap values are supported:

butt cap : the edge is squared off at the endpoint; there is no projection beyond the endpoint.

round cap : a semicircular arc with diameter equal to the line width is drawn around the endpoint and filled. The drawn edge thus projects beyond the endpoint.

projecting square cap : the edge is squared off at a distance equal to half the line width beyond the endpoint.

Other values are reserved for future registration and standardization. The default value is butt cap.

Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Edge Cap escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
edge cap indicator (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	edge cap indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the edge cap index.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
edge cap indicator	(BUTT, ROUND, PROJECTING SQUARE)	E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(EDCAP)

Input Parameters:
INTEGER EDCAP          edge cap indicator (BUTT, ROUND,
                       PROJECTING SQUARE)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL              1
INTEGER IA( 1 )         edge cap indicator (BUTT, ROUND,
                        PROJECTING SQUARE)
INTEGER RL 0
INTEGER SL 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL              0
INTEGER RL              0
INTEGER SL              0
```

5) **Pascal language binding** (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEscapeEdgeEndType = (GVEscapeButt, GVEscapeRound,
                      GVEscapeProjectingSquare);

GREscapeDataIn = RECORD

            CASE EscapeID : GTEscapeDataTag of
                 1:(
                    EdgeCap       : GEscapeEdgeEndType);
            End;

GREscapeDataOut = RECORD
            CASE EscapeId  :  GTEscapeDataTag of
                1: () ; (*Null Record *)
            END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_EDGE_CAP" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set line cap.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type EDGE_CAP_INDICATOR_TYPE is (BUTT, ROUND, PROJECTING SQUARE);
     EDGE_CAP_INDICATOR_VALUE     :in EDGE_CAP_INDICATOR_TYPE;

procedure SET_EDGE_CAP

       (ESCAPE_IDENTIFIER     :in ESCAPE_ID:
        EDGE_CAP_INDICATOR_VALUE    :in EDGE_CAP_INDICATOR TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Edge Join

Description

This escape function sets a value for the current edge join. This value is to determine the shape put at corners between portions of edges during subsequent interpretation of filled area graphical primitives. See attached sheet for additional details.

Additional Comments

None.

Justification for Inclusion

User specified edge joins are commonly found in proprietary graphics systems. They are needed to support the requirements of office document exchange and publishing. The edge capabilities proposed here are adapted from those in the PostScript language, which was developed by Adobe Systems Incorporated, and whose specification has been placed in the public domain.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Edge Join sets the current edge join state in the graphics state to *edge join indicator*. This establishes the shape to be put at the corners between portions (line or curve segments) of the edges of filled area primitives. The following edge join values are supported:

mitre join : the outer edge of the two portions are extended until they meet at a point. (Note that the mitre limit value may affect the appearance of these joins.)

round join : a circular arc with diameter equal to the edge width is drawn around the vertex between the adjoining segments and is filled in, producing a rounded corner.

bevel join : the meeting portions are finished with butt end cap and the resulting triangular notch is filled in.

Other values are reserved for future registration and standardization.

Join styles are significant only at points where consecutive portions of an edge connect at an angle; portions that meet or intersect fortuitously receive no special treatment.

Relationship to particular standards:

1) **CGM Functional Specification** (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Edge Join escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
edge join indicator (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	edge join indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the edge join index.

2) **CGM Encodings** (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Description (reference ISO 7942 GKS functional description)

This escape is applicable at level 0a of GKS. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
edge join indicator	(MITRE, ROUND, BEVEL)	E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(EDJOIN)

Input Parameters:
INTEGER EDJOIN          edge join indicator (MITRE, ROUND,
   BEVEL)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

INTEGER IL	1
INTEGER IA(1)	edge join indicator (MITRE, ROUND, BEVEL)
INTEGER RL	0
INTEGER SL	0

Output parameters to the Unpack Data Record function (GUREC):

INTEGER IL	0
INTEGER RL	0
INTEGER SL	0

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GEscapeEdgeJoin = (Mitre, Round, Bevel);

GREscapeDataIn = RECORD
       CASE EscapeId   :  GTEscapeDataTag of
          1:(
             EdgeJoin :  GEscapeEdgeJoin);
       END;

GREscapeDataOut = RECORD
       CASE EscapeId   :  GTEscapeDataTag of
           1:  () ;                (*Null Record*)
       END;
```

6) GKS Ada Language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

Registered ESCAPE's are in a library package name GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_EDGE_JOIN" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for a set line join
--Data type ESCAPE_ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPES is
     type EDGE_JOIN_INDICATOR_TYPE is (MITRE, ROUND, BEVEL);
     EDGE_JOIN_INDICATOR_VALUE : in EDGE_JOIN_INDICTOR_TYPE;

procedure SET_EDGE_JOIN
        (ESCAPE_IDENTIFIER   :in ESCAPE_ID;
         EDGE_JOIN_INDICATOR_VALUE:in EDGE_JOIN_INDICTOR_TYPE );

--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	18 July 1988

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Italic Text

Description

This escape function sets a value for the italic attribute of text. This value may be either on, indicating that text is drawn in an italic style, or off, indicating that text is not drawn in an italic style.
See attached sheet for additional details.

Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable for most applications. Most proprietary graphics systems currently use a text model that names a basic font family (such as Gothic) and the identifies members of that family by additional syle attributes (such as underline, bold, or italic.) This is one in a set of escapes that provide similar facilities for the family of computer graphics standards.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Italic Text sets the current italic text value in the graphics state to the value specified by the *italic text indicator*. This establishes whether text is to be drawn in a italic style. The following italic text values are supported:

OFF: italic text is set to OFF, indicating that text is not drawn in an italic style.

ON: italic text is set to ON, indicating that text is drawn in an italic style.

Values other than OFF and ON are reserved for future registration and standardization.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Italic Text escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
italic text indicator(OFF,ON) (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	italic text indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the italic text indicator.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
italic text indicator	(OFF, ON)	E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(ITTXT)

Input Parameters:
INTEGER ITTXT          italic text indicator (OFF, ON)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL             1
INTEGER IA( 1 )        italic text indicator (OFF, ON)
INTEGER RL 0
INTEGER SL 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL             0
INTEGER RL             0
INTEGER SL             0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
ItalicTextType = (GVEscapeOn,
                  GVEscapeOff);

GREscapeDataIn = RECORD

            CASE EscapeID : GTEscapeDataTag of
                 1:(
                    ItalicText     : GEscapeItalicTextType);
            End;

GREscapeDataOut = RECORD
            CASE EscapeId  :  GTEscapeDataTag of
                1: () ; (*Null Record *)
            END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_ITALIC_TEXT" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for italic text.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type ITALIC_TEXT_INDICATOR_TYPE is (OFF, ON);
  type ITALIC_TEXT_DATA_RECORD is
     ITALIC_TEXT_INDICATOR_VALUE    :in ITALIC_TEXT_INDICATOR_TYPE;

procedure SET_ ITALIC_TEXT

      (ESCAPE_IDENTIFIER     :in ESCAPE_ID:
       ITALIC_TEXT_INDICATOR_VALUE:in ITALIC_TEXT_INDICATOR_TYPE);

--
--more ESCAPE procedures can be inserted here     .
--
end GKS ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	18 July 1988

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Outline Text

Description

This escape function sets a value for the outline attribute of text. This value may be either on, indicating that text is drawn in an outline style, or off, indicating that text is not drawn in an outline style.
See attached sheet for additional details.

Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable for most applications. Most proprietary graphics systems currently use a text model that names a basic font family (such as Gothic) and the identifies members of that family by additional syle attributes (such as underline, bold, or italic.) This is one in a set of escapes that provide similar facilities for the family of computer graphics standards.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Outline Text sets the current outline text value in the graphics state to the value specified by the *outline text indicator*. This establishes whether text is to be drawn in a outline style. The following Outline text values are supported:

OFF: outline text is set to OFF, indicating that text is not drawn in an outline style.

ON: outline text is set to ON, indicating that text is drawn in an outline style.

Values other than ON and OFF are reserved for future registration and standardization.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Outline Text escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
outline text indicator (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	outline text indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the outline text indicator.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
outline text indicator	(OFF, ON)	E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(OLTXT)

Input Parameters:
INTEGER OLTXT          outline text indicator (OFF, ON)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL             1
INTEGER IA( 1 )        outline text indicator (OFF, ON)
INTEGER RL 0
INTEGER SL 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL             0
INTEGER RL             0
INTEGER SL             0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
ExtendedTextType = (GVEscapeOn,
                    GVEscapeOff);

GREscapeDataIn = RECORD

            CASE EscapeID : GTEscapeDataTag of
                 1:(
                    ExtendedText   : GEscapeOutlineTextType);
            End;

GREscapeDataOut = RECORD
            CASE EscapeId  :  GTEscapeDataTag of
                1: () ; (*Null Record *)
            END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_OUTLINE_TEXT" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for Outline text.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type OUTLINE_TEXT_INDICATOR_TYPE is (OFF, ON);
  type OUTLINE_TEXT_DATA_RECORD is
     OUTLINE_TEXT_INDICATOR_VALUE  :in OUTLINE_TEXT_INDICATOR_TYPE;

procedure SET_OUTLINE_TEXT

    (ESCAPE_IDENTIFIER       :in ESCAPE_ID:
     OUTLINE_TEXT_INDICATOR_VALUE :in OUTLINE_TEXT_INDICATOR_TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	18 July 1988

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape	Function Identifier:	Set Shadow Text

Description

This escape function sets a value for the shadow attribute of text. This value may be either on, indicating that text is drawn in a shadow style, or off, indicating that text is not drawn in a shadow style.
See attached sheet for additional details.

Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable for most applications. Most proprietary graphics systems currently use a text model that names a basic font family (such as Gothic) and the identifies members of that family by additional syle attributes (such as underline, bold, or italic.) This is one in a set of escapes that provide similar facilities for the family of computer graphics standards.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Shadow Text sets the current shadow text value in the graphics state to the value specified by the *shadow text indicator*. This establishes whether text is to be drawn in a shadow style. The following shadow text values are supported:

OFF: shadow text is set to OFF, indicating that text is not drawn in an shadow style.

ON: shadow text is set to ON, indicating that text is drawn in an shadow style.

Values other than OFF and ON are reserved for future registration and standardization.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Shadow Text escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
shadow text indicator (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	shadow text indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the shadow text indicator.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
shadow text indicator	(OFF, ON)	E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(SHTXT)

Input Parameters:
INTEGER SHTXT          shadow text indicator (OFF, ON)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL             1
INTEGER IA( 1 )        shadow text indicator (OFF, ON)
INTEGER RL 0
INTEGER SL 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL             0
INTEGER RL             0
INTEGER SL             0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
ShadowTextType = (GVEscapeOn,
                  GVEscapeOff);

GREscapeDataIn = RECORD

            CASE EscapeID : GTEscapeDataTag of
                 1:(
                    ShadowText     : GEscapeShadowTextType);
            End;

GREscapeDataOut = RECORD
            CASE EscapeId  :  GTEscapeDataTag of
                1: () ; (*Null Record *)
            END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_SHADOW_TEXT" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for shadow text.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type SHADOW_TEXT_INDICATOR_TYPE is (OFF, ON);
  type SHADOW_TEXT_DATA_RECORD is
     SHADOW_TEXT_INDICATOR_VALUE    :in SHADOW_TEXT_INDICATOR_TYPE;

procedure SET_SHADOW_TEXT

      (ESCAPE_IDENTIFIER      :in ESCAPE_ID:
       SHADOW_TEXT_INDICATOR_VALUE:in SHADOW_TEXT_INDICATOR_TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	18 July 1988

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Underline Text

Description

This escape function sets a value for the underline attribute of text. This value may be either on, indicating that text is drawn in an underline style, or off, indicating that text is not drawn in an underline style.
See attached sheet for additional details.

Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable for most applications. Most proprietary graphics systems currently use a text model that names a basic font family (such as Gothic) and the identifies members of that family by additional syle attributes (such as underline, bold, or italic.) This is one in a set of escapes that provide similar facilities for the family of computer graphics standards.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Underline Text sets the current underline text value in the graphics state to the value specified by the *underline text indicator*. This establishes whether text is to be drawn in a underline style. The following underline text values are supported:

OFF: underline text is set to OFF, indicating that text is not drawn in an underline style.

ON: underline text is set to ON, indicating that text is drawn in an underline style.

Values other than ON and OFF are reserved for future registration and standardization.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Underline Text escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
underline text indicator (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	underline text indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the underline text indicator.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
underline text indicator (OFF, ON)		E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(ULTXT)

Input Parameters:
INTEGER ULTXT          underline text indicator (OFF, ON)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL             1
INTEGER IA( 1 )        underline text indicator (OFF, ON)
INTEGER RL 0
INTEGER SL 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL             0
INTEGER RL             0
INTEGER SL             0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
UnderlineTextType = (GVEscapeOn,
                     GVEscapeOff);

GREscapeDataIn = RECORD

            CASE EscapeID : GTEscapeDataTag of
                 1:(
                    UnderlineText : GEscapeUnderlineTextType);
            End;

GREscapeDataOut = RECORD
            CASE EscapeId  :  GTEscapeDataTag of
                1: () ; (*Null Record *)
            END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_UNDERLINE_TEXT" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for underline text.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type UNDERLINE_TEXT_INDICATOR_TYPE is (OFF, ON);
  type UNDERLINE_TEXT_DATA_RECORD is
    UNDERLINE_TEXT_INDICATOR_VALUE    :in
                UNDERLINE_TEXT_INDICATOR_TYPE;

procedure SET_UNDERLINE_TEXT

        (ESCAPE_IDENTIFIER    :in ESCAPE_ID:
         UNDERLINE_TEXT_INDICATOR_VALUE
                :in UNDERLINE_TEXT_INDICATOR_TYPE );

--
--more ESCAPE procedures can be inserted here
--
end GKS ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

te of Presentation:	18 July 1988

onsoring Authority:	ANSI

ass of Graphical Item:	ESCAPE

ecific Escape	Function Identifier:	Set Bold Text

scription

is escape function sets a value for the bold attribute of text. This value may be either on, indicating that text drawn in a bold style, or off, indicating that text is not drawn in a bold style.
e attached sheet for additional details.

ditional Comments

is escape is intended for interim use, pending the revision of the text model in computer graphics standards sed upon the developing ISO font architecture (DP 9541).

stification for Inclusion

lensions to the text model in computer graphics standards are urgently needed if such standards are to be able for most applications. Most proprietary graphics systems currently use a text model that names a basic t family (such as Gothic) and the identifies members of that family by additional syle attributes (such as derline, bold, or italic.) This is one in a set of escapes that provide similar facilities for the family of nputer graphics standards.

lationship to Standards

) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

present at the stage of draft. The status of this relationship is provisional until this standard has been proved by ISO council.

Description:

Set Bold Text sets the current bold text value in the graphics state to the value specified by the *bold text indicator*. This establishes whether text is to be drawn in a bold style. The following bold text values are supported:

OFF: bold text is set to OFF, indicating that text is not drawn in an bold style.

ON: bold text is set to ON, indicating that text is drawn in an bold style.

Values other OFF and ON are reserved for future registration and standardization.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Bold Text escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
bold text indicator(OFF,ON) (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	bold text indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the bold text indicator.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
bold text indicator (OFF, ON)		E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(BLTXT)

Input Parameters:
INTEGER BLTXT          bold text indicator (OFF, ON)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL             1
INTEGER IA( 1 )        bold text indicator (OFF, ON)
INTEGER RL 0
INTEGER SL 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL             0
INTEGER RL             0
INTEGER SL             0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
BoldTextType = (GVEscapeOn,
                GVEscapeOff);

GREscapeDataIn = RECORD

            CASE EscapeID : GTEscapeDataTag of
                 1:(
                    BoldText       : GEscapeBoldTextType);
            End;

GREscapeDataOut = RECORD
            CASE EscapeId  :  GTEscapeDataTag of
                1: () ; (*Null Record *)
            END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_BOLD_TEXT" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for bold text.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type BOLD_TEXT_INDICATOR_TYPE is (OFF, ON);
  type BOLD_TEXT_DATA_RECORD is
    BOLD_TEXT_INDICATOR_VALUE    :in BOLD_TEXT_INDICATOR_TYPE;

procedure SET_BOLD_TEXT

       (ESCAPE_IDENTIFIER    :in ESCAPE_ID:
        BOLD_TEXT_INDICATOR_VALUE  :in BOLD_TEXT_INDICATOR_TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	9 September 1988

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Fully Justified Text

Description

This escape function sets a value for the fully justified attribute of Text. This value may be either on, indicating that Restricted Text is drawn in a fully justified style, or off, indicating that Restricted Text is not drawn in a fully justified style. This attribute has no affect on the drawing of Text or Append Text primitives.
See attached sheet for additional details.

Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable for most applications. Most proprietary graphics systems currently allow fully justified text. Such text is drawn by the target system by varying the spacing between words in a text string to insure that the string starts at the beginning of a restricted region and ends at the end of the region. The Restricted Text primitive in the existing graphics standards partially meets this goal, but its semantics allows the target system to adjust text by many different means to meet this restriction. This escape requires that the text be fit by adjusting only the inter-word spacing where this is possible. Based on "minimality", it was decided to simply add an attribute

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Fully Justified Text sets the current fully justified text value in the graphics state to the value specified by the *fully justified text indicator*. This establishes whether Restricted Text primitives are to be drawn in a fully justified style.

Fully Justified text is defined as follows. First, just as with Restricted Text, the fully justified text is constrained to be within a parallelogram as defined for Restriced Text. Second, in fitting the text within this parallelogram, only the value of Character Spacing for the spaces between words may be adjusted. (Restricted Text allows the values of Character Height, Character Expansion Factor, Text Precision, and TextFont Index to be adjusted also, all at the sole discretion of the system drawing the text.) Third, the text is drawn in such a way that the symbols drawn appear to "fill" the parallelogram along the text path, with the first and last symbols drawn against the sides of the parallelogram. In case the text cannot be fit within the parallelogram by adjusting only the Character Spacing value for the spaces between words, then the text will be drawn using the rules for Restructed Text in general.

The following fully justified text values are supported:

OFF: fully justified text is set to OFF, indicating that Restricted Text is not drawn in a fully justified style.

ON: fully justified text is set to ON, indicating that Restricted Text is drawn in a fully justified style.

Values other than ON and OFF are reserved for future registration and standardization.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Fully Justified Text escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
fully justified text indicator(OFF, ON) (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	fully justified text indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the fully justified text indicator.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
fully justified text indicator	(OFF, ON)	E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(FJTXT)

Input Parameters:
INTEGER FJTXT          fully justified text indicator(OFF, ON)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL             1
INTEGER IA( 1 )        fully justified text indicator(OFF, ON)
INTEGER RL 0
INTEGER SL 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL             0
INTEGER RL             0
INTEGER SL             0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
FullyJustifiedTextType = (GVEscapeOn,
                          GVEscapeOff);

GREscapeDataIn = RECORD

            CASE EscapeID : GTEscapeDataTag of
                 1:(
                    FullyJustifiedText
                        : GEscape FullyJustifiedTextType);
            End;

GREscapeDataOut = RECORD
            CASE EscapeId  :  GTEscapeDataTag of
                1: () ; (*Null Record *)
            END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_FULLY_JUSTIFIED_TEXT" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for fully justified text.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type FULLY_JUSTIFIED_TEXT_INDICATOR_TYPE is (OFF, ON);
  type FULLY_JUSTIFIED_TEXT_DATA_RECORD is
    FULLY_JUSTIFIED_TEXT_INDICATOR_VALUE    :in
                FULLY_JUSTIFIED_TEXT_INDICATOR_TYPE;

procedure SET_FULLY_JUSTIFIED_TEXT

       (ESCAPE_IDENTIFIER    :in ESCAPE_ID:
        FULLY_JUSTIFIED_TEXT_INDICATOR_VALUE
              :in FULLY_JUSTIFIED_TEXT_INDICATOR_TYPE );

--
--more ESCAPE procedures can be inserted here
--
end GKS ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	18 July 1988

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Condensed Text

Description

This escape function sets a value for the condensed attribute of text. This value may be either on, indicating that text is drawn in a condensed style, or off, indicating that text is not drawn in a condensed style.
See attached sheet for additional details.

Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable for most applications. Most proprietary graphics systems currently use a text model that names a basic font family (such as Gothic) and the identifies members of that family by additional syle attributes (such as underline, bold, or italic.) Many systems include extended and condensed styes of text, where the font designer specifies changes in character spacing to achieve the condensed or extended appearance. Although these effects could be crudely approximated by varying the Character Spacing attribute of graphical text, such an approximation is inappropriate for systems that require high-quality text. This is one in a set of escapes that

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Condensed Text sets the current condensed text value in the graphics state to the value specified by the *condensed text indicator*. This establishes whether text is to be drawn in a condensed style. The following Condensed text values are supported:

OFF: condensed text is set to OFF, indicating that text is not drawn in an condensed style.

ON: condensed text is set to ON, indicating that text is drawn in an condensed style.

Values other than OFF and ON are reserved for future registration and standardization.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Condensed Text escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
condensed text indicator(OFF,ON) (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	condensed text indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the condensed text indicator.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
condensed text indicator	(OFF, ON)	E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(CNTXT)

Input Parameters:
INTEGER CNTXT          condensed text indicator (OFF, ON)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL             1
INTEGER IA( 1 )        condensed text indicator (OFF, ON)
INTEGER RL 0
INTEGER SL 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL             0
INTEGER RL             0
INTEGER SL             0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
CondensedTextType = (GVEscapeOn,
                     GVEscapeOff);

GREscapeDataIn = RECORD

                 CASE EscapeID : GTEscapeDataTag of
                      1:(
                         CondensedText : GEscapeCondensedTextType);
                 End;

GREscapeDataOut = RECORD
                 CASE EscapeId  :  GTEscapeDataTag of
                     1: () ; (*Null Record *)
                 END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_CONDENSED_TEXT" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for condensed text.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type CONDENSED_TEXT_INDICATOR_TYPE is (OFF, ON);
  type CONDENSED_TEXT_DATA_RECORD is
     CONDENSED_TEXT_INDICATOR_VALUE    :in
                 CONDENSED_TEXT_INDICATOR_TYPE;

procedure SET_CONDENSED_TEXT

        (ESCAPE_IDENTIFIER     :in ESCAPE_ID:
         CONDENSED_TEXT_INDICATOR_VALUE
        :in CONDENSED_TEXT_INDICATOR_TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	18 July 1988

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Extended Text

Description

This escape function sets a value for the extended attribute of text. This value may be either on, indicating that text is drawn in an extended style, or off, indicating that text is not drawn in an extended style.
See attached sheet for additional details.

Additional Comments

This escape is intended for interim use, pending the revision of the text model in computer graphics standards based upon the developing ISO font architecture (DP 9541).

Justification for Inclusion

Extensions to the text model in computer graphics standards are urgently needed if such standards are to be usable for most applications. Most proprietary graphics systems currently use a text model that names a basic font family (such as Gothic) and the identifies members of that family by additional syle attributes (such as underline, bold, or italic.) Many systems include extended and condensed styes of text, where the font designer specifies changes in character spacing to achieve the condensed or extended appearance. Although these effects could be crudely approximated by varying the Character Spacing attribute of graphical text, such an approximation is inappropriate for systems that require high-quality text. This is one in a set of escapes that

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

Set Extended Text sets the current extended text value in the graphics state to the value specified by the *extended text indicator*. This establishes whether text is to be drawn in an extended style. The following extended text values are supported:

OFF: extended text is set to OFF, indicating that text is not drawn in an extended style.

ON: extended text is set to ON, indicating that text is drawn in an extended style.

Values other than OFF and ON are reserved for future registration and standardization.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

A functional description of the Set Extended Text escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
extended text indicator(OFF,ON) (E)

Items for Data Record:

Integer IL	1
Integer IA(1)	extended text indicator
Integer RL	0
Integer SL	0

Data Record Description:

The parameter defines the extended text indicator.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
extended text indicator	(OFF, ON)	E
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(EXTXT)

Input Parameters:
INTEGER EXTXT          extended text indicator (OFF, ON)

Output Parameters:
NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
INTEGER IL             1
INTEGER IA( 1 )        extended text indicator (OFF, ON)
INTEGER RL 0
INTEGER SL 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL             0
INTEGER RL             0
INTEGER SL             0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GESCAPE" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
ExtendedTextType = (GVEscapeOn,
                    GVEscapeOff);

GREscapeDataIn = RECORD

            CASE EscapeID : GTEscapeDataTag of
                 1:(
                    ExtendedText   : GEscapeExtendedTextType);
            End;

GREscapeDataOut = RECORD
            CASE EscapeId   :  GTEscapeDataTag of
                1: () ; (*Null Record *)
            END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3:Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_EXTENDED_TEXT" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function for extended text.
--Data type ESCAPE ID is defined in package GKS_ESCAPE.
--Other data types are defined in package GKS-TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type EXTENDED_TEXT_INDICATOR_TYPE is (OFF, ON);
  type EXTENDED_TEXT_DATA_RECORD is
    EXTENDED_TEXT_INDICATOR_VALUE :in EXTENDED_TEXT_INDICATOR_TYPE;

procedure SET_EXTENDED_TEXT

   (ESCAPE_IDENTIFIER        :in ESCAPE_ID:
    EXTENDED_TEXT_INDICATOR_VALUE:in EXTENDED_TEXT_INDICATOR_TYPE);

--
--more ESCAPE procedures can be inserted here
--
end GKS ESCAPE;
```

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEMS

Proposal Number:	

Date of Presentation:	10 April 1987

Sponsoring Authority:	ANSI

Class of Graphical Item:	GDP

GDP Identifier:	Pel Array

Description

This Generalized Drawing Primitive provides a very flexible mechanism for describing raster (image) data within computer graphics standards. It can be viewed as an extension of the Cell Array primitive to allow alternate encoding and compression schemes. See attached sheet for additional details.

Additional Comments

The features in this GDP are based upon those in the Cell Array primitive, with extensions adopted from CCITT T.4 and T.6. Tag Image File Format(TIFF), and Tiled Raster Interchange Format (TRIF) as needed. to meet the requirements of intended applications.

Justification for Inclusion

The raster data capabilities of existing graphics standards provide a rich and flexible set of facilities for dealing with raster (image) data. These facilities do, however, lack a few features that limit their wider applicability and often force users to utilize techniques outside of computer graphics standards to describe/transfer raster data. This GDP is intended to add the missing facilities. With it, a single file format—the CGM—can meet the needs of both raster and "vector" graphics storage and exchange in office, publishing, and other applications.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered GDP as defined in 5.3.

2) ISO 8632 (CGM) - Specifies a registered GDP as defined in 5.6.10.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered GDP. (See attached sheets).

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

The Pel Array generalized drawing primitive is an extended version of the Cell Array primitive designed to support the transfer, storage, and printing of raster image data from a variety of sources.

Parameters:

3 corner points P,Q, and R (3P)

nx,ny (number of pels per line; number of lines) (2I)

encoding (one of: packed, run-length, T.4, T.6, LZW) (E)

local colour precision

pel array colour specifiers (nx*ny*local colour precision)

In the general case, P,Q,R can delimit an arbitrary pallelogram. P and Q delimit the end points of a diagonal of the parallelogram, and R defines a third corner.

In the simplest case, the three corner points, P,Q,R, define a rectangular area in the coordinate space. This area is subdivided into nx*ny contiguous rectangles as follows. The edge from P to R is subdivided into nx equal intervals, and the edge from R to Q is subdivided into ny equal intervals. The grid implied consists of nx*ny identical pels. The colour list consists of nx*ny colour specifications, conceptually an array of dimensions nx and ny representing respectively the column and row dimensions. Array element (1,1) is mapped to the pel at corner P, and array element (nx,1) is mapped to pel at corner R. Array element (nx,ny) is mapped to the pel at corner Q. Hence, the colour elements are mapped within rows running from P to R, and with the rows incrementing in order from R to Q. [Note that all four traditional values of pel path -- 0, 90, 180, and 270 -- as well as both traditional values of line progression -- 90 and 270 -- can be realized by varying the inter-relationships among P, Q, and R]

The encoding parameter specifies how the pel array values are encoded. This parameter allows an API standard to pick up a pel array (image) obtained from an external device, such as a scanner, and image(print) it without having to interpret the data in it. Similarly, it allows a graphical metafile or interface standard to transfer such data without having to interpret it. The allowable values of encoding are:

Packed : No compression, but pack colour values as tightly as possible, with no unused bits except at the end of a row. The colour values are represented by rows of values, each row starting on a (16 bit) word boundary. [Note: This encoding is identical to the "packed list" mode in the binary binding of the CGM. It is also equivalent to the "packed" mode of TIFF, with byte order restricted to "MM".]

Run-length : The colour list values are represented by rows broken into runs of constant colour; each row starts on a (16 bit) word boundary. [Note: This encoding is identical to the "packed list" mode in the binary binding of the CGM.]

T.4 one dimensional : Facsimile-compatible CCITT Group 3, exactly as specified in "Standardization of Group 3 facsimile apparatus for document transmission" Recommendation T.4 Volume VII, Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services, The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, 1985, pages 16 through 21. All rows must begin on a byte boundary. The restrictions in T.4 for the number of pels per line (nx), the number of lines per pel array (ny), and the size, position, and orientation of an pel array within a picture do not apply. [Note that the addition of fill bits before EOLs to force them to end on word boundaries is not required but is allowed.]

T.4 two dimensional : Facsimile-compatible CCITT Group 3 two dimensional, exactly as specified in "Standardization of Group 3 facsimile apparatus for document transmission" Recommendation T.4 Volume VII, Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services, The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, 1985, pages 21 through 28. All rows must begin on a byte boundary. The restrictions in T.4 for the number of pels per line (nx), the number of lines per pel array (ny), and the size, position, and orientation of an pel array within a picture do not apply. [Note that the addition of fill bits before EOLs to force them to end on word boundaries is not required but is allowed.]

T.6 : Facsimile-compatible CCITT Group 4, exactly as specified in "Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus", Recommendation T.6, Volume VII, Fascicle VII.3. Terminal Equipment and Protocols for Telematic Services, The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, 1985, pages 40 through 48. The restrictions in T.4 for the number of pels per line (nx), the number of lines per pel array (ny), and the size, position, and orientation of an pel array within a picture do not apply. [Note that the addition of fill bits before EOLs to force them to end on word boundaries is not required but is allowed.] [Note: This encoding is identical to that of the T.4 two dimensional with the single exception that the "K" parameter which controls the number of consecutive two dimensional lines without an intervening one dimensional line has the value infinity rather than a small integer.]

LZW Compression : An adaptive compression for raster images as defined in an article by Terry A. Welch, entitled "A Technique for High Performance Data Compression", IEEE Computer , vol. 17 no. 6 (June 1984), and called the basic Lempel-Ziv & Welch (LZW) algorithm. [Note: The author's goal in that article was to describe a hardware-based compressor that could be built into disk controller or database engine, and used on all types of data. There is no specific discussion of raster images.]

LZW is fully reversible. All information is preserved, but if noise or information is removed from an image, perhaps by smoothing or zeroing some low-order bitplanes, LZW compresses images into a smaller size. Thus, 5-bit, 6-bit, or 7-bit data masquerading as

8-bit data compresses better than true 8-bit data. Smooth images also compress better than noisy images, and simple images compress better than complex images. LZW works well on bilevel images too. It generally ties T.4 one dimensional (Modified Huffman) compression, on test images while LZW seems to be considerably fa

LZW Encoding

The LZW algorithm is based on a translation table, or string table, that maps strings of input characters into codes. Variable-length codes are used, with a maximum code length of 12 bits. This string table is different for every pel array, and, remarkably, does not need to be kept for the decompressor. The trick is to make the decompressor automatically build the same table as is built when compressing the data. The following C-like pseudocode describes the coding scheme:

```
InitializeStringTable();
WriteCode(ClearCode);
Ω = the empty string
for each character in the pel array{
  K = GetNextOctet();
  if Ω+K is the string table{
     Ω = Ω+K; /* string concatenation*/
  }else{
     WriteCode(CodeFromString(Ω));
     AddTableEntry(Ω);
     Ω = K;
  }
}/*end of for loop*/
WriteCode(CodeFromString(Ω));
WriteCode(EndofInformation);
```

The "characters" that make up LZW strings are octets of uncompressed pel array data. InitializeStringTable() initializes the string table to contain all 256 possible single octet codes, numbered 0 through 255. WriteCode() writes a code to the output stream. The first code written is a Clear code, which is defined to be code #256. Ω represents the "prefix" string. GetNextOctet() retrieves the next octet from the input stream. The "+" sign indicates string concatenation.

AddTableEntry() adds a table entry. Since InitializeStringTable has already added 256 entries to the table, and since entry 256 is reserved for a special "clear code", and entry #257 is reserved for a special "End of Information" code, therefore the first multi-octet entry to the table is made at position 258.

Since codes are written using as few bits as possible, WriteCode() starts out with 9 bit codes since the new entries are greater than 255 but less than 512. When table entry 512 is added, WriteCode switches to 10 bit codes. Likewise, it switches to 11 bit codes at 1024 and to 12 bit codes at 2048. The table is limited to 12 bit codes, so when entry 4094 is reached, a ClearCode is written and the compresor re-initilaizes the table and starts writing out 9 bit codes again. Each encoded pel array begins with a Clear code and ends with an End of Information code.

LZW Decoding

The procedure for decompression is described by the following pseudocode:

```
while ((CodeNextCode()!=End of Information code){
   if ((Code == Clear code){
      InitializeTable();
      Code = GetNextCode();
      if (Code == End of Information code )
         break;
      WriteCode(StringFromCode(Code));
      OldCode = Code;
   } */end of Clear code case*/

   else{
        if (IsInTable(Code)){
           WriteString(StringFromCode(Code));
           AddStringToTable(StringFromCode(OldCode)+FirstChar
            (StringFromCode(Code)));
           WriteString(OutString);
           AddStringToTable(OutString);
           OldCode = Code;

        }else{
           OutString=StringFromCode(OldCode)
              + FirstChar(StringFromCode(Code));
           AddStringToTable(OutString);
           OldCode = Code;
           WriteString(OutString);
        }
     }/*end of not-Clear code case*/
 }/*end of while loop*/
```

The function GetNextCode() retrieves the next code from the LZW-coded data. It must keep track of bit boundaries. It knows that the first code that it gets will be 9-bit code. We add a table entry each time we get a code, so GetNextCode() must switch over to 10-bit as soon as string #511 is stored into the table.

The function StringFromCode() gets the string associated with a particular code from the string table.

The function AddStringToTable() adds a string to the string table. The "+" sign joining the two parts of the argument to AddStringToTable indicate the string concatenation.

StringFromCode() looks up the string associated with a given code.

WriteString() adds a string to the output stream.

The 'local colour precision' parameter declares the precision of 'cell colour specifiers'. It applies only to computer graphics standards that support the both direct and indexed specification of colour. If indexed colour selection is used, then this parameter specifies the colour index precision. If direct colour selection is used, then then this parameter specifies the colour precision.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

The LCP (local color precision) parameter declares the precision of the cell colour specifiers. The precision is for either indexed or direct colour, according to the COLOUR SELECTION MODE of the picture. The form of the parameter is encoding dependent. If the picture uses indexed colour selection, then the form of the parameter is the same as that of COLOUR INDEX PRECISION. If the picture uses direct colour selection, then the form of the parameter is the same as that of COLOUR PRECISION. Since the array may be compressed, its length may not be able to be calculated directly from the number of pel array colour specifier values. Consequently, the pel array is treated as an array of 16 bit integer words and its length is specified by the pel array length parameter.

A functional description of the Pel Array generalized drawing primitive parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

point list(nP) contains the three points P, Q, and R
data record(D)

Items for Data Record:

Integer IL	5 + pel array length (which depends the relationship of LCP to integer precision integer precision and encoding)
Integer IA(1)	nx
Integer IA(2)	ny
Integer IA(3)	encoding
Integer IA(4)	LCP
Integer IA(5)	pel array length
Integer IA(6)	start of pel array colour specifiers
....	
Integer RL	0
Integer SL	0

Data Record Description:

The data record contains the dimensions of the pel array, its encoding type, its local colour precision, and the pel array itself.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

This GDP is applicable at level 0a of GKS. The corner points are transformed to NDC and the pel array through those points is then drawn. The pel array colour specifiers may be either index or direct colour values, as determined by the value of the colour specification type parameter. If the pel array uses indexed colour selection, then the local colour precision specifies the length of the index values in bits. If the pel array uses direct colour selection, then the local colour precision specifies the length of the direct RGB colour values in bits. Since the array may be compressed, its length may not be able to be calculated directly from the number of pel array colour specifier values. Consequently, the pel array is treated as an array of 16 bit integer words and its length is specified by the pel array length parameter.

A functional description of all input parameters is:

Name	Coordinate System	Values	Data Type
number of points		(3)	I
corner points (P,Q, and R)	WC		3xP
GDP identifier		as assigned	N
GDP data record: nx,ny (2I)			
encoding	(packed, run-length, T.4, T.6, LZW)		E
colour specification type	(indexed, direct)		E
local colour precision			I
pel array length			I
pel array colour specifiers	(nx*ny*local colour precision)		I*pel array length

Errors:

5 GKS not in proper state: GKS shall be either in the state WSAC or in the state SGOP
100 Number of points is invalid

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GDpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the GDP (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
 SUBROUTINE GDpqrs( N, PXA, PYA, NX, NY, IENCODE, ICSPEC, LCP,
+IPAL, IPACS)
```

Input Parameters:

INTEGER N	number of points (3)
REAL PXA(3), PYA(3)	P,Q, and R corner points
INTEGER NX, NY	number of pels per line; number of lines
INTEGER IENCODE	encoding (packed, run-length, T.4, T.6, LZW)
INTEGER ICSPEC	colour specification type(indexed, direct)
INTEGER LCP	local colour precision
INTEGER IPEL	pel array length
INTEGER IPACS(IPEL)	pel array colour specifiers (number of values is nx*ny*local colour precision)

b) The following parameters are proposed for use when accessing this GDP through the GGDP function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

Integer IL	5 + pel array length (which depends the relationship of LCP to integer precision integer precision and encoding)
Integer IA(1)	nx
Integer IA(2)	ny
Integer IA(3)	encoding
Integer IA(4)	LCP
Integer IA(5)	pel array length
Integer IA(6)	start of pel array colour specifiers
....	
Integer RL	0
Integer SL	0

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
NumPoints = 3;
Points : GAPointArray;
GEGDPCurveProperty6 = (packed, run-length, T.4, T.6, LZW);
GEGDPCurveProperty7 = (indexed, direct);


GRGDPData = RECORD
       CASE GDPId : GTGDPDataTag OF
          1: (
               PelsPerLine                : INTEGER;
               LinesPerArray              : INTEGER;
               EncodingType               : GEGDPCurveProperty6;
               ColourSpecificationType    : GEGDPCurveProperty7;
               LocalColourPrecision       : INTEGER;
               PelArrayLength             : INTEGER;
               PelArrayColourSpecifiers   : array
                                              [1..PelArrayLength]
                                            of INTEGER);
       END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

Registered GDP's are in a library package named GKS_GDP. GKS Ada provides a data type package, GKS_TYPES which provides types declarations.

The binding for the "procedure PEL_ARRAY" form (as defined in paragraph 4.1 of the GKS Ada language binding) of the GDP is:

```
--
--GDP function for a Pel Array.
--Data type GDP_DATA_RECORD is defined in package GKS_GDP.
--GDP_ID and other data types are defined in package GKS_TYPES.
--

with GKS_TYPES;
use GKS_TYPES;
package GKS_GDP is
  type CORNER_POINTS is new WC.POINT_ARRAY (1..3);
  type ENCODING_TYPE is (packed, run-length, T.4, T.6, LZW);
  type COLOUR_SPECIFICATION_TYPE is (indexed, direct);
 type PEL_ARRAY_COLOUR_SPECIFIERS is array (NATURAL range<>) of
    ESCAPE_INTEGER);.
  type PEL_ARRAY_DATA_RECORD is
    record
         PELS_PER_LINE                : ESCAPE_INTEGER;
         LINES_PER_ARRAY              : ESCAPE_ INTEGER;
         ENCODING                     : ENCODING_TYPE;
         COLOUR_SPECIFICATION         : COLOUR_SPECIFICATION_TYPE;
         LOCAL_COLOUR_PRECISION       : ESCAPE_ INTEGER;
         PEL_ARRAY_LENGTH             : ESCAPE_ INTEGER;
         PEL_ARRAY                    : PEL_ARRAY_COLOUR_SPECIFIERS;
    end;

procedure PEL_ARRAY
     ( CORNER_POINTS                 :in BEZIER_POINTS;
       GDP_IDENTIFIER                :in GDP_ID;
       PEL_ARRAY_RECORD              :in PEL_ARRAY_DATA_RECORD;
     );

--
--more GDP procedures can be inserted here
--

end GKS_GDP;
```

Proposal Number:	

Date of Presentation:	9 September 1988

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Indexed Colour Response

Description

This escape function sets a value for the indexed colour response curve. This curve is used to provide exact photometric information in terms of density for indexed colour specifications contained in pel array primitives. See attached sheet for additional details.

Additional Comments

This escape is intended for use in conjunction with the pel array generalized drawing primitive, although it could be used for rendering other primitives as well.

Justification for Inclusion

Photometric information in terms of density for indexed colour specifications contained in pel array primitives is needed if output devices are to reproduce input pel arrays precisely. Many proprietary systems for defining/storing/transferring raster image data provide this capability. This is one in a set of escapes that provide extended raster/image data capabilities enabeling the family of computer graphics standards to meet the requirements of office document generation/exchange and technical publications.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

The purpose of the Set Indexed Colour Response escape function is to define a "curve" providing exact photometric interpretation information in terms of optical density for indexed colour image data contained in pel array primitives. In particular, if the index values represent ranges of a monochrome value such as gray, this escape can be used to provide more exact photometric interpretation information for gray scale image data. The default curve is linear in intensity/reflectance.

Since optical density is specified in terms of fractional numbers, Real numbers are used for these values. *Optical densitometers* typically measure densities within the range of 0.0 to 2.0. If the indexed colour response curve is known for the data in a pel array, and if the indexed colour response of the output device is known, then an intelligent conversion can be made between the input data and the output device. For example, the output can be made to look just like the input. In addition, if the input image lacks contrast (as can be seen from the response curve), then appropriate contrast enhancements can be made.

The purpose of the indexed colour response curve is to act as a "lookup" table mapping values from 0 to 2**(local colour precision)-1 into specific density values. The *0th* element of the indexed colour response curve array is used to define the colour response value for all pels having an index value of 0, the *1st* element of the indexed colour response curve array is used to define the colour response value for all pels having a value of 1, and so on, up to 2**(local colour precision)-1. It is permissible to have a indexed colour response curve even for bilevel (1-bit) pel arrays. The indexed colour response curve will have 2 values.

Implementers may wish to purphase a Kodak Reflection Density Guide, catalog number 146,5947, available for $10 or so at prepress supply houses, and use it to determine reasonable density values for their scanner or frame grabber. If this is not practical the default curve that is linear in intensity/reflectance can be used.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

The local colour precision value in this escape should match that used in subsequent Pel Array GDPs when index colour is used. A functional description of the Set Indexed Colour Response escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
LCP (local colour precision) (I)
colour response curve ((2**LCP) * R)

Items for Data Record:

Integer IL	1
Integer IA(1)	LCP (local colour precision)
Integer RL	2**LCP
Real RA(1)	colour response curve (0)
Real RA(2)	colour response curve (1)
...	
Real RA(2**LCP)	colour response curve ((2**LCP)-1)
Integer SL	0

Data Record Description:

The parameters define the local colour precision and indexed colour response curve for pel array data.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

The set dash escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
LCP (local colour precision)		I
colour response curve		(2**LCP)*R
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(LCP, CRC)

Input Parameters:

INTEGER LCP          local colour precision
REAL CRC(2**LCP)     colour response curve

Output Parameters:

NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

```
Integer IL           1
Integer IA(1)        LCP (local colour precision)
Integer RL           2**LCP
Real RA(1)           colour response curve (0)
Real RA(2)           colour response curve (1)
...
Real RA(2**LCP)      colour response curve ((2**LCP)-1 )
Integer SL           0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL           0
INTEGER RL           0
INTEGER SL           0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
  GREscapeDataIn = RECORD
     CASE EscapeId : GTEscapeDataTag of
     1:(
        LocalColourPrecision : INTEGER;
        ColourResponseCurve  : array [1..(2**LocalColourPrecision)]
                               of REAL);
     END;

    GREscapeDataOut = RECORD
        CASE EscapeID : GTEscapeDataTag of
             1: () ;          (*Null Record*)
        END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_INDEXED_COLOUR_RESPONSE" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function to set the indexed colour response curve for a
--pel array.
--Data types ESCAPE_ID and ESCAPE_FLOAT are defined in package
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type COLOUR_RESPONSE_CURVE is array
       (SMALL_NATURAL range <>)of ESCAPE_FLOAT:
  type INDEXED_COLOUR_RESPONSE_RECORD is
    record
     LOCAL_COLOUR_PRECISION  :in INTEGER;
     COLOUR_RESPONSE         :in COLOUR_RESPONSE_CURVE
                               (0..(2**LOCAL_COLOUR_PRECISION-1));
    end record;

procedure SET_INDEXED_COLOUR_RESPONSE
       (ESCAPE_IDENTIFIER    :in ESCAPE_ID;
        COLOUR_RESPONSE      :in INDEXED_COLOUR_RESPONSE_RECORD );
--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

Proposal Number:	

Date of Presentation:	9 September 1988

Sponsoring Authority:	ANSI

Class of Graphical Item:	ESCAPE

Specific Escape Function Identifier:	Set Direct Colour Response

Description

This escape function sets a value for the direct colour response curve. This curve is used to provide exact photometric information in terms of intensity for direct colour specifications contained in pel array primitives. See attached sheet for additional details.

Additional Comments

This escape is intended for use in conjunction with the pel array generalized drawing primitive, although it could be used for rendering other primitives as well.

Justification for Inclusion

Photometric information in terms of intensity for directly specified colour contained in pel array primitives is needed if output devices are to reproduce input pel arrays precisely. Many proprietary systems for defining/storing/transferring/viewing raster image data provide this capability. This is one in a set of escapes that provide extended raster/image data capabilities enabeling the family of computer graphics standards to meet the requirements of office document generation/exchange and technical publications.

Relationship to Standards

1) ISO 7942 (GKS) - Specifies a registered escape as defined in 5.2.

2) ISO 8632 (CGM) - Specifies a registered escape as defined in 5.8.1.

*3) ISO 8651 (GKS Language Bindings) - Specifies a registered escape.

*At present at the stage of draft. The status of this relationship is provisional until this standard has been approved by ISO council.

Description:

The purpose of the Set Direct Colour Response escape function is to define a "curve" providing exact photometric interpretation information in terms of intensity for direct colour image data contained in pel array primitives. Three colour response curves, one each for Red, Green and Blue color information are defined. The Red entries come first, followed by the Green entries, followed by the Blue entries. The default curves are linear in intensity/reflectance. The length of each subcurve is 2**(local colour precision), using the same local colour precision value as subsequent pel array generalized drawing primitives. Each entry is a 16 integer value. 0 represents the minimum intensity, and 65535 represents the maximum intensity. Black is represented by (0,0,0), and white by (65535, 65535, 65535). Therefore, a color response curve entry for direct (RGB) colour data with a local colour precision of 8 bits would have 3 * 256 entries, each consisting of a 16 bit integer.

Relationship to particular standards:

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional Description)

The local colour precision value in this escape should match that used in subsequent Pel Array GDPs when index colour is used. A functional description of the Set Indexed Colour Response escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
LCP (local colour precision) (I)
red response curve ((2**LCP) * I)
green response curve ((2**LCP) * I)
blue response curve ((2**LCP) * I)

Items for Data Record:

Integer IL	1 + 3*(2**LCP)
Integer IA(1)	LCP (local colour precision)
Integer IA(2)	red response curve (0)
Integer IA(3)	red response curve (1)
...	
Integer IA(2**LCP+1)	red response curve ((2**LCP)-1)
Integer IA(2**LCP+2)	green response curve (0)
Integer IA(2**LCP+3)	green response curve (1)
...	
Integer IA(2*2**LCP+1)	green response curve ((2**LCP)-1)
Integer IA(2*2**LCP+2)	blue response curve (0)
Integer IA(2*2**LCP+3)	blue response curve (1)
...	
Integer IA(3*2**LCP+1)	blue response curve ((2**LCP)-1)
Integer RL	0
Integer SL	0

Data Record Description:

The parameters define the local colour precision and indexed colour response curve for pel array data.

2) CGM Encodings (reference ISO 8632 CGM; Parts 2,3,4)

All encodings will be handled in the same way. The entire data record, containing the data record items in sequential order from first to last, will be treated as a string. Ignoring (for the moment) the contents of the "string", the entire data record will be encoded according to the rules for string in that encoding. Considering the string contents (that is, the data record items), the base data types in the data record are encoded according to the encoding rules for that type in that encoding. For example, in the binary encoding, a data record that contains two 16 bit integers would be coded as if it were a string of length 4. Within the four octets that comprise the string's contents, the two 16 bit integers would be coded using the binary coding for 16 bit binary integers.

3) GKS Functional Specification (reference ISO 7942 GKS Functional Description)

The set dash escape is applicable at GKS level 0a. A functional description of its parameters is given below:

Name	Values	Data Type
escape function identifier	as assigned	N
input data record:		
LCP (local colour precision)		I
red response curve		(2**LCP)*I
green response curve		(2**LCP)*I
blue response curve		(2**LCP)*I
output data record:		
none		

Errors:

8 GKS not in proper state: GKS shall be either in one of the states GKOP,WSOP, WSAC, or SGOP

4) GKS FORTRAN language binding (reference ISO DIS 8651/1 GKS Language Bindings; Part 1: FORTRAN)

a) The following language binding is proposed for the "GEpqrs" form (as defined in Paragraph 9.1 of the GKS FORTRAN language binding) of the escape (pqrs is to be assigned by the Registration Authority to correspond to the assigned Register Identifier):

```
SUBROUTINE GEpqrs(LCP, RCRC, GCRC, BCRC)

Input Parameters:

INTEGER LCP           local colour precision
INTEGER RCRC(2**LCP)  red response curve
INTEGER GCRC(2**LCP)  green response curve
INTEGER BCRC(2**LCP)  blue response curve

Output Parameters:

NONE
```

b) The following parameters are proposed for use when accessing this escape through the GESC function of Paragraph 9.3 of the GKS FORTRAN language binding standard:

Input parameters to the Pack Data Record function (GPREC):

Items for Data Record:

```
Integer IL                 1 + 3*(2**LCP)
Integer IA(1)              LCP (local colour precision)
Integer IA(2)              red response curve (0)
Integer IA(3)              red response curve (1)
...
Integer IA(2**LCP+1)       red response curve ((2**LCP)-1 )
Integer IA(2**LCP+2)       green response curve (0)
Integer IA(2**LCP+3)       green response curve (1)
...
Integer IA(2*2**LCP+1)     green response curve ((2**LCP)-1 )
Integer IA(2*2**LCP+2)     blue response curve (0)
Integer IA(2*2**LCP+3)     blue response curve (1)
...
Integer IA(3*2**LCP+1)     blue response curve ((2**LCP)-1 )
Integer RL                 0
Integer SL                 0
```

Output parameters to the Unpack Data Record function (GUREC):

```
INTEGER IL                 0
INTEGER RL                 0
INTEGER SL                 0
```

5) Pascal language binding (reference: ISO DIS 8651 GKS Language Bindings; Part 2: Pascal)

The following Pascal language binding is proposed for the procedure "GEscape" as defined in paragraph 6.2 of the GKS Pascal language binding (note the case variant "1" will be replaced with the actual ESCAPE identifier at registration):

```
GREscapeDataIn = RECORD
   CASE EscapeId : GTEscapeDataTag of
   1:(
      LocalColourPrecision : INTEGER;
      RedResponseCurve     : array [1..(2**LocalColourPrecision)]
                             of INTEGER);
      GreenResponseCurve   : array [1..(2**LocalColourPrecision)]
                             of INTEGER);
      BlueResponseCurve    : array [1..(2**LocalColourPrecision)]
                             of INTEGER);
   END;

 GREscapeDataOut = RECORD
      CASE EscapeID : GTEscapeDataTag of
           1: () ;          (*Null Record*)
      END;
```

6) GKS Ada language binding (reference ISO DIS 8651/3 GKS Language Bindings; Part3: Ada)

Registered ESCAPE's are in a library package named GKS_ESCAPE. GKS Ada provides a data type package, GKS_TYPES which provides type declarations.

The binding for the "procedure SET_DIRECT_COLOUR_RESPONSE" form (as defined in Paragraph 4.1 of the GKS Ada language binding) of the ESCAPE is:

```
--
--Escape function to set the indexed colour response curve for a
--pel array.
--Data types ESCAPE_ID and ESCAPE_FLOAT are defined in package
--GKS_ESCAPE.
--Other data types are defined in package GKS_TYPES.

with GKS_TYPES;
use GKS_TYPES;
package GKS_ESCAPE is
  type DIRECT_COLOUR_RESPONSE_CURVE is array
       (SMALL_NATURAL range <>)of INTEGER:
  type DIRECT_COLOUR_RESPONSE_RECORD is
    record
     LOCAL_COLOUR_PRECISION  :in INTEGER;
     COLOUR_RESPONSE         :in DIRECT_COLOUR_RESPONSE_CURVE
                                (0..(2**LOCAL_COLOUR_PRECISION-1));
    end record;

procedure SET_DIRECT_COLOUR_RESPONSE
       (ESCAPE_IDENTIFIER    :in ESCAPE_ID;
        COLOUR_RESPONSE      :in DIRECT_COLOUR_RESPONSE_RECORD );
--
--more ESCAPE procedures can be inserted here
--
end GKS_ESCAPE;
```

This page left intentionally blank.

APPENDIX 1

RESPONSES TO BALLOT COMMENTS

PROPOSAL 36-SET DASH

1) GSS #5: Your comment is not accepted. The dash pattern length should be specified in VDC to match the way that the line width specifier for the Line Width CGM element is specified when line-width specification mode is absolute. This gives a level of control similar to that PostScript gives, where both line width and dash pattern are specified in the same coordinate system ("user coordinates".)

2) DRI #25: Your comment is accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string.

3) Apollo #34: Your comments are accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string. Further, we agree with you that all the escapes in the set should be "workstation independent", and have removed the workstation ID from all the proposals. Thank you for your excellent and insightful comments.

4) System One Software #53: Your comment is accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string.

5) Henderson Software #52: Your comments are accepted. Where the parameters of our proposed "direct" line attribute specifications overlap those of the "indexed" ones of the "initial text" of CGEM Addendum 3, we have changed ours to match yours. We would like to point out several facts, however:

a) the Line Type Definition in tile "CGEM" is indexed, not 'direct; our requirements dictate a direct specification;

b) the Line Type Definition in the "CGEM" is too complex for our purposes;

c) the Line Type Definition in the "CGEM" has no counterpart to the "continuity" parameter m our proposal;

d) "CGEM" text is in a very preliminary state, with no consensus reached even within ANSI, let alone

internationally. Following the 1 year SC24 study period in this area, work may start within 150 that could produce a stable set of extensions in 2-3 years. Our registration proposals are based on current commercial practice and are urgently needed to meet present needs. While we will endeavor to make every attempt to be compatible with early CGEM text, we can neither take the level of consensus represented by that text too seriously nor can we wait 3-4 years for DIS "CGEM" text to base our proposals on. To put the shoe on the other foot, "are you willing to freeze your CGEM draft if we agree to make our proposals totally compatible?" Even if you are, you can't because of the way in which the consensus standards making process works. Registration is a "weaker" mechanism than consensus standardization. It is designed to rapidly "register" (not standardize) standards-related items of use to more than one implementation. We do not expect the CGEM work to be "bound" in any way by the items we register. We fully expect that 3-4 years of consensus standardization work will result in CGM extensions that are superior to, and that will eventually replace, ours.

6) **Sun #12:** Your comment is accepted. We have added default values to proposals 36, 37, and 39.

7) **HP #4:** Your comment is accepted. We have changed the word "required" to "provided in the suggested location.

PROPOSAL 37-SET LINE CAP

1) **Apollo #34:** Your comments are accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string. Further, we agree with you that all the escapes in the set should be "workstation independent", and have removed the workstation ID from all the proposals. Thank you for your excellent and insightful comments.

2) **System One Software #53:** Your comment is accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string.

3) **Henderson Software #52:** Your comments are accepted. We have changed the text of this proposal to match that of the Line Cap element of the "CGEM"" document (SC24 N52).

4) **Sun #12:** Your comment is accepted. We have added default values to proposals 36, 37, and 39.

5) **Nova Graphics #13:** In answer to your comment (1), the PostScript Specification has been placed in the public domain, and we have informed Adobe Systems of the extensions to Computer Graphics standards that we are writing. In answer to your comment (2), we have adopted the same association of this attribute to primitives as the CGEM has. There are no compound objects in our approved standards to worry about applying them to.

PROPOSAL 38-SET MITRE LIMIT

1) **Apollo #34:** Your comments are accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string. Further, we agree with you that all the escapes in the set should be "workstation independent", and have removed the workstation ID front all the proposals. Thank you for your excellent and insightful comments.

2) **System One Software #53:** Your comment is accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string.

3) **Henderson Software #52:** Your comments were partially accepted. Where the parameters of our proposed attribute specifications overlap those of the "initial text" of CGEM Addendum 3 (SC24 N52), we have changed ours to match yours. We have not adopted the Mitre Limit definition from the CGEM however, opting instead for a reformulation proposed by IIP and Nova Graphics. We suggest that the CGEM do likewise.

4) **Nova Graphics #13:** Your comments are accepted. We have changed the word spelling of "miter" to "mitre" throughout the proposal. We have modified the mitre limit definition as you suggested to make it more "continuous." In answer to your comment (I), the PostScript Specification has been placed in the public domain, and we have informed Adobe Systems of the extensions to Computer Graphics standards that we are writing. In answer to your comment (2), we have adopted the same association of this attribute to primitives as the CGEM has. There are no compound objects in our approved standards to worry about applying them to.

PROPOSAL 39-SET LINE JOIN

1) Apollo #34: Your comments are accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string. Further, we agree with you that all the escapes in the set should be "workstation independent", and have removed the workstation ID from all the proposals. Thank you for your excellent and insightful comments.

2) System One Software #53: Your comment is accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string.

3) Henderson Software #52: Your comments are accepted. We have changed the text of this proposal to match that of the Line Join element of the "CGEM"" document (SC24 N52).

4) Sun #12: Your comment is accepted. We have added default values to proposals 36, 37, and 39.

5) Nova Graphics #13: In answer to your comment (1), the PostScript Specification has been placed in the public domain, and we have informed Adobe Systems of the extensions to Computer Graphics standards that we are writing. In answer to your comment (2), we have adopted the same association of this attribute to primitives as the CGEM has. There are no compound objects in our approved standards to worry about applying it to.

PROPOSAL 40-CUBIC BEZIER CURVE

1) System One Software #53: Your comment is accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string.

2) Henderson Software #52: We are gratified that the CGM group has chosen our publications on required extensions to the CGM as the basis for their work. We note with pleasure that CGM addendum 3 is based word-for-word on our proposals in most cases. In so far as possible, we will work with the CGM group to insure that registered items and CGM extensions are as compatible as possible given the different time frames for these two projects.

3) US Navy #50: The name has been changed to Cubic Bezier Curve.

4) **Apollo #34:** Your comments are accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string. Further, we agree with you that all the escapes in the set should be "workstation independent", and have removed the workstation ID from all the proposals. A description of allowable errors has been added to each escape and GDP. We have moved the material describing the general functionality of each GDP and Escape to the "description" section.

Thank you for your excellent and insightful comments.

5) **3M #44:** We have revised the proposal to incorporate your suggested replacement wording.

PROPOSAL 41-CONIC ARC

1) **System One Software #53:** Your comment is accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string.

2) **Henderson Software #52:** We are gratified that the CGM group has chosen our publications on required extensions to the CGM as the basis for their work. We note with pleasure that CGM addendum 3 is based word-for-word on our proposals in most cases. In so far as possible, we will work with the CGM group to insure that registered items and CGM extensions are as compatible as possible given the different time frames for these two projects.

3) **3M #44:** We have revised the proposal to incorporate your suggested replacement wording.

4) **DRI #25:** Apparently developers of the CGM felt that is was reasonable to require the use of real parameters in an integer VDC system. For examples, check the binary encodings of both Character Spacing and Character Expansion Factor, both of which are type Real irrespective of the VDC Type. We feel that the use of Real is also justified in this case for similar reasons, consequently we cannot accept your comment.

5) **Hewlett-Packard #4:** We have modified our proposal to be compatible with CGM Addendum 3, consequently we cannot accept your comments as that conflict with that addendum. The term "counterclockwise" is now defined in the proposal.

6) **McDonnell Douglas Corporation #19:** We have revised the proposals as you suggested and have extracted material from the IGES specifications, rather than incorporating that material by reference. Although we are concerned about compatibility issues when this is done, it is clear to us that the graphics standards community is not yet mature enough to be able to adopt material "not invented here" without getting their hands into modifying it. The changes required were slight, since the CGM Addendum 3 material is only slightly re-worded from our proposals and made only insignificant deletions to the referenced IGES material.

We recommended carrying tile TPX and TPT parameters for the same reasons that IGES does, and we certainly realized that the receiving system can derive them from other information. We have accepted your comment and have removed them.

7) **Apollo #34:** We have revised our proposals based on comments by others to be more compatible with cGM Addendum 3. This requires that certain "points" in definition space be treated like "VDC' points for coding purposes. This also suggests that we ignore the issue of the start and end points not being on the curve since CGM Addendum 3 ignores this problem. We did add text to require these points to be on the curve, and will suggest that the CGM group do likewise.

We have modified the descriptions of our proposed GDPs to make their adaption into different standards easier. One flaw in the registration process is that a single "description" is supposed to apply to all standards. This is difficult, since coordinates are often in WC in standards like GKS and in VDC in the CGM. We have tried to work around this by describing things in generic terms and then specializing in the "Relationship to Standards" section.

We agree with you that the start and end points in the GKS binding do not belong in "WC" and are not transformed as usual GDP "points." We have moved them to the data record as you suggested. We have made similar changes to the GKS FORTRAN binding to eliminate the N, PXA and PYA values.

8) **Nova Graphics #13:** The proposed escape does not violate minimality since it is more general that' the elliptical arc already in tile CGM. On the issue of parameterization, we have revised our parameterization to be consistent with CGM Addendum 3. (While we are sympathetic with your observations on parameterization and numerical sensitivities, we chose to remain compatible with IGES usage. Clearly, this is not acceptable to most graphics folks, so we have changed.)

PROPOSAL 42-SET CONIC ARC TRANSFORMATION MATRIX

1) System One Software #53: Your comment is accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string.

2) Henderson Software #52: We are gratified that the CGM group has chosen our publications on required extensions to the CGM as the basis for their work. We note with pleasure that CGM addendum 3 is based word-for-word on our proposals in most cases. In so far as possible, we will work with the CGM group to insure that registered items and CGM extensions are as compatible as possible given the different time frames for these two projects. We have made several changes to this particular proposal based on letter ballot comments. For example, we converted to the use of a "homogeneous coordinates" formulation of the transformation/translation process. We suggest that you make similar changes to CGM Addendum 3.

3) Pansophic #24: Your comment is accepted. We removed the offending part of the functional specification.

4) Hewlett-Packard #4: While we are sympathetic to your concerns about the introduction of a new "pipeline" element with the Conic Arc Transformation Matrix, such a separate matrix element has been chosen by both IGES and CGM Addendum 3 (SC24 N52.) Our choice was made to be as compatible as possible, consequently we cannot accept your comment.

We have modified our proposal to pass the coordinates of the transformation matrix in "homogeneous" format as you requested. We note that this is not the parameterization used in CGM Addendum 3.

After reviewing the arithmetic stability involved in the computations against the requirements of our constituency, we have concluded that only real matrix values are acceptably for this element. Thus, we accept your comment about the use of real values only in this element.

5) DRI #25: Developers of the CGM apparently felt that is was reasonable to require the use of real parameters in an integer VDC system. For examples, look at the binary encodings of both the Character Spacing and Character Expansion Factor elements, both of which are type Real irrespective of the VDC Type. We feel that the use of Real is justified in this case for similar reasons, consequently we cannot accept your comment.

6) Apollo #34: We have removed workstation identifier as a parameter in the GKS version of this escape.

We agree with your assessment that the parameters Rij should be reals independent of the VDC type in the CGM encoding and have made this change.

We have modified the descriptions to make the coordinate systems clearer. The transformation now maps a definition space into VDC in the case of the CGM, and into WC in the case of GKS. In the GKS case, the current normalization transformation would map to NDC.

7) Nova Graphics #13: This escape defines a "modelling", not a "viewing" transform. Our concept of doing it this way has been adopted by CGM Addendum 3. (We have now modified our parameterizations to be consistent with theirs.) We don't see that your comments apply in this case.

8) McDonnell Douglas Corporation #19: We have revised the proposals as you suggested and have extracted material from the IGES specifications, rather than incorporating that material by reference. Although we are concerned about compatibility issues when this is done, it is clear to us that the graphics standards community is not yet mature enough to be able to adopt material "not invented here" without getting their hands into modifying it. The changes required were slight, since the CGM Addendum 3 material is only slightly re-worded from our proposals and made only insignificant deletions to the referenced IGES material.

PROPOSAL 43-PARAMETRIC SPLINE CURVE

1) System One Software #53: Your comment is accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string.

2) Henderson Software #52: We are gratified that the CGM group has chosen our publications on required extensions to the CGM as the basis for their work. We note with pleasure that CGM addendum 3 is based word-for-word on our proposals in most cases. In so far as possible, we will work with the CGM group to insure that registered items and CGM extensions are as compatible as possible given the different time frames for these two projects.

3) 3M #44: We have revised the proposal to incorporate your suggested replacement wording.

4) Puk #26: The omission of IA(4) was a typographical error that

has been corrected.

5) E&S #36: Your comment concerns a note in the IGES specification regarding the availability of certain conversion software. We merely repeated the IGES wording. You will have to take your question up with NBS. We have removed the offending paragraph from our revised proposal, so your objection has been answered.

6) Hewlett Packard #4: After reviewing the arithmetic stability involved in the computations against the requirements of our constituency, we have concluded that only real values are acceptable for this element. Thus, we accept your comment about the use of real ~C only in this element.

We have removed VDC type integer, so this satisfies your objections to various typographical errors. We wish to point out that these errors resulted when GSC Associates inputs were transcribed by others and we were afforded no opportunity to review the work.

We have corrected the value of RL. We have reversed the lines "INTEGER SL" and "...".

Thank you for pointing out errors in the Rational B-spline language bindings. These bindings were done by X3H34, not by us as the proposer. We have attempted to correct them however, but our work should be reviewed. Further, we will refer the incomplete bindings back to X3H34 for completion.

7) McDonnell Douglas Corporation #19: We have revised the proposals as you suggested and have extracted material from the IGES specifications, rather than incorporating that material by reference. Although we are concerned about compatibility issues when this is done, it is clear to us that the graphics standards community is not yet mature enough to be able to adopt material "not invented here" without getting their hands into modifying it. The changes required were slight, since the CGM Addendum 3 material is only slightly re-worded from our proposals and made only insignificant deletions to the referenced IGES material.

8) Apollo #34: That you for pointing out the numerous errors that resulted when our material was transcribed without our knowledge. We have corrected those that were typographical.

We have removed the "I"' values from the point lists as you requested.

We have dropped NORJ4 from the specification

We have modified the primitives to use only real values as you requested.

We have corrected the numbers of control points and weights.

9) **Nova Graphics #13:** We have removed the offending values from the point list.

PROPOSAL 44-RATIONAL B-SPLINE CURVE

1) **System One Software #53:** Your comment is accepted. We have completely changed the description of the encoding of escape elements in all the proposals. After surveying commercial practice, we have settled on a coding as a "string" using the base types for each encoding within the string.

2) **Henderson Software #52:** We are gratified that the CGM group has chosen our publications on required extensions to the CGM as the basis for their work. We note with pleasure that CGM addendum 3 is based word-for-word on our proposals in most cases. In so far as possible, we will work with the CGM group to insure that registered items and CGM extensions are as compatible as possible given the different time frames for these two projects.

3) **3M #44:** We have revised the proposal to incorporate your suggested replacement wording.

4) **McDonnell Douglas Corporation #19:** We have revised the proposals as you suggested and have extracted material from the IGES specifications, rather than incorporating that material by reference. Although we are concerned about compatibility issues when this is done, it is clear to us that the graphics standards community is not yet mature enough to be able to adopt material "not invented here" without getting their hands into modifying it. The changes required were slight, since the CGM Addendum 3 material is only slightly re-worded from our proposals and made only insignificant deletions to the referenced IGES material.

NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4317

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
MARCH 1991

4. TITLE AND SUBTITLE
A Collection of Technical Studies Completed for the Computer-Aided Acquisition and Logistic Support (CALS) Program Fiscal Year 1988 Volume 3 of 3, CGM Registration

5. AUTHOR(S)
Roy S. Morgan, Editor

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)
U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED
NISTIR 10/87 - 9/88

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)
Office of the Secretary of Defense
Production and Logistics/Systems/CALS
Room 3B322, Pentagon
Washington, D.C. 20301-8000

10. SUPPLEMENTARY NOTES

[ ] DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

Computer-aided Acquisition and Logistic Support (CALS) Program is a DoD and Industry strategy to transition from paper-intensive acquisition and logistic processes to a highly automated and integrated mode of operation for the weapon systems of the 1990s. These volumes document the accomplishments of the National Institute of Standards and Technology to advance the development of technology and standards in support of CALS. These reports are divided into three volumes: 1, Text, Security, and Data Management; 2, Graphics, CGM MIL-SPEC; and 3, Graphics, CGM Registration.

Volume 3. CGM Registration: This report documents work accomplished to meet CALS needs for CGM functionality through the registration of graphical elements within the standardization process.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)
CGM; graphical element; graphics; graphics metafile; metafile; registration; standardization

13. AVAILABILITY

[X] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[ ] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
173

15. PRICE
A08

ELECTRONIC FORM