AD-A261 271

NBSIR 88-3728

CGM Registration for CALS Requirements

A Technical Study Completed for the Computer-aided Acquisition and Logistic Support (CALS) Program Fiscal Year 1987 Volume 3 of 4

Sharon J. Kemmerer, Editor

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Information Systems Engineering Division
Gaithersburg, MD 20899

DTIC
ELECTE
FEB 25 1993
S E D

March 1988

DISTRIBUTION STATEMENT
Approved for public release;
Distribution Unlimited

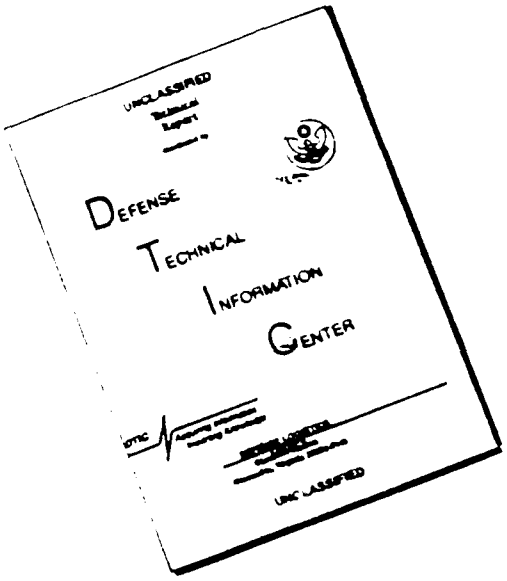
Stimulating America's Progress
1913-1988

U.S. DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS

93-03926

93 2 24 050

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

NBSIR 88-3728

CGM REGISTRATION FOR CALS REQUIREMENTS
A TECHNICAL STUDY COMPLETED FOR THE COMPUTER-AIDED ACQUISITION AND LOGISTIC SUPPORT (CALS) PROGRAM
FISCAL YEAR 1987
VOLUME 3 OF 4

Sharon J. Kemmerer, Editor

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Information Systems Engineering Division
Gaithersburg, MD 20899

March 1988

Accesion For
NTIS CRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification .....
By .....
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

DTIC QUALITY INSPECTED 3

U.S. DEPARTMENT OF COMMERCE, C. William Verity, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

EXECUTIVE SUMMARY

The overall objective of the Department of Defense Computer-aided Acquisition & 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, an OSD CALS Policy Office, and their counterparts in each Military Department and the Defense Logistics Agency. The CALS Policy Office has obtained the support of the National Bureau of Standards 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 Bureau has been funded since Spring 1986 to recommend a suite of industry standards for system integration and digital data transfer, and to accelerate their implementation. NBS activities during 1986 were primarily aimed at:

- familiarizing NBS technical staff with key DoD logistic functions and CALS demonstration projects,
- briefing DoD personnel, contractors, and other interested parties on Federal, national, and international standardization efforts that are expected to support CALS objectives,
- identifying a preliminary set of standards required for data interchange in support of CALS, and
- developing reports on the four broad categories of standards required to support the interchange of CALS digitized technical information: (1) product definition data, (2) graphics, (3) text, and (4) data management.

As a result of these efforts, NBS made a preliminary identification of several high-priority standards implementations

needed for CALS data interchange and access.[1] Building on knowledge and experience gained during FY86, NBS focused on the following activities in FY87: 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 Logistics Agency, and industry.

A major FY87 thrust was the completion of initial documentation of the high-priority standards required in the CALS environment. Some of these standards (e.g., SGML, IGES) required tailoring or enhancement. Other standards required a "push" (e.g., CGEM) for their development in a timely fashion. These four volumes are a collection of the final reports presented to the CALS Policy Office.[2] The collection is divided as follows:

VOLUME 1:

Text

- Evaluation of Text Interchange Methods
- Plan for Conformance Testing for DoD Implementation of SGML
- Guidelines for the Development of Tags for SGML
- The NBS FIPS - SGML Validation Suite
- The NBS FIPS - SGML Reference Parser
- Using SGML - Application Guidelines
- ODA/ODIF Implementation Agreement a Document Application Profile

Data Management

- CALS Report on Data Management Standards
- Supporting Logistic Support Analysis (LSA) Using the Information Resource Dictionary System (IRDS)

Media

- ICST Recommendations on Optical Disks and Interface Requirements for Planned EDMICS Procurement, Final Report

---

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 The publishing of this collection of reports does not imply the CALS Policy Office has endorsed the conclusions and recommendations presented.

Raster Compression

Report on Raster Graphics

Tiled Raster Interchange Format, TRIF Version 1.0, Rev. 1.2

Conformance Testing

NBS Plan for Validation (Conformance Testing) of Computer Products in Support of the CALS Program

VOLUME 2:

Graphics

Raster-to-Vector Conversion: A State-of-the-Art Assessment

Development of CGM Validation Routines

CALS Application Profile for CGM

CALS Requirements Reflected in the Extended CGM (CGEM) Standards Effort

A Reference Implementation for CGM, Functional Requirements and Conceptual Design

IGES to CGM Translator Design Specification

VOLUME 3:

Graphics

CGM Registration For CALS Requirements

VOLUME 4:

Product Data

Guidelines for Testing IGES Translators

Guidelines for IGES Application Subsets

The following are additional deliverables completed by NBS during FY87 but under separate cover. They are available through the CALS Policy Office.

CALS Core Requirements, Phase I.0

CALS Framework

CALS Program Integration of Logistic Support Analysis and Reliability and Maintainability Data Deliverables

CALS Current State of Digital Technology (Phase I.0)

CALS Workshop Proceedings:
- Graphics Data Interface for Engineering Design and Technical Publication Systems (January 13/14)
- Introduction to the Core Requirements Package (April 23)

MILSTD-1840A, Automated Interchange of Technical Information

MILSPEC-D-28000, Digital Representation for Communication of Product Data: Application Subsets

MILSPEC-M-28001, Manuals, Technical: Markup Requirements and Generic Style Specification for Electronic Printed Output and Exchange

CONTRIBUTIONS

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

Daniel Benigni

David Jefferson

Sharon Kemmerer

Mark Skall

FINAL REPORT

CALS SOW TASK 2.2.2.2.2

CGM REGISTRATION FOR CALS REQUIREMENTS

TABLE OF CONTENTS

		Page
I.	PURPOSE	1
II.	BACKGROUND	1
	1.0 Overview of Computer Graphics Standards	1
	1.1 Output Primitives	1
	1.2 Output Primitive Attributes	2
	2.0 Registration of Graphical Items	4
	2.1 Registration in the CGM Standard	4
	2.2 Why Registration for the CGM Alone is Insufficient	6
	2.3 The Required Approach for CALS Registration	8
	3.0 Applicable Standards	9
	4.0 Reference Model	12
III.	DISCUSSION	13
	1.0 Technical and Administrative Publishing	13
	1.1 Introduction	13
	1.2 Required Capabilities CGM Must Be Extended to Meet	13
	1.2.1 Lines	14
	1.2.1.1 Setlinecap	14
	1.2.1.2 Setlinejoin	15
	1.2.1.3 Setmiterlimit	15
	1.2.1.4 Setdash	16
	1.2.1.5 Closepath	17
	1.2.2 Symbols	17
	1.2.3 Curves	17
	1.2.4 Text	18
	1.2.4.1 Objectives and Functional Definitions	18
	1.2.4.2 Font Referencing	22
	1.2.4.3 Character Positioning	23
	1.2.4.4 Image Presentation	24
	1.2.5 Images	25
	1.2.6 Definition and Instantiation of Objects	25
	1.2.6.1 PostScript Procedures	26
	1.2.6.2 Interpress Symbols	26
	2.0 Technical Drawings and Product Data	27
	2.1 Introduction	27
	2.2 IGES Entities	28
	2.3 Required Capabilities	29
	2.3.1 Lines	29
	2.3.2 Symbols	34
	2.3.3 Curves	39
	2.3.4 Hatch Styles	39
	2.3.5 Text	41
	2.3.5.1 Y14.2 Lettering Requirements	41
	2.3.5.2 Text in IGES	43
	2.3.6 Images	44

TABLE OF CONTENTS (Continued)

		Page
IV.	CONCLUSIONS	45
	1.0 Introduction	45
	2.0 Lines	45
	3.0 Symbols	46
	4.0 Curves	46
	5.0 Hatch Styles	46
	6.0 Text	46
	7.0 Images	47
	8.0 Naming and External References	47
V.	AREAS REQUIRING FURTHER INVESTIGATION	48
	1.0 Curves	48
	2.0 Text	49
	3.0 Images	49
	4.0 Specification of Data Record Contents	49
	5.0 Support for Named Items and Symbol Libraries	50
	6.0 Definition of Registration Requirements	50
VI.	RECOMMENDATIONS	51
	1.0 Registration Proposals List	51
	2.0 Prepared Registration Proposals	52
VII.	REFERENCES	171
VIII.	APPENDICES	173
	Appendix A Extracts from Typical DoD Standards	173
	Appendix B PostScript and Interpress Capabilities	189
	Appendix C IGES Entities and Capabilities	223
	Appendix D CGM Extracts	283

FIGURES AND TABLES

			Page
Table	1	Output Primitive Attributes	3
Figure	1	Linecap and Linejoin Capabilities	15
Figure	2	Setmiterlimit Capabilities	16
Figure	3	Setdash Capabilities	16
Figure	4	Closepath Usage	17
Figure	5	ISO Font Architecture - Application Environment	19
Figure	6	A Lower-level Look at the Application Environment	20
Figure	7	Line Styles and Widths	30
Figure	8	Line in Engineering Drawings	31
Figure	9	Arrowhead Styles	33
Figure	10	Basic Symbol Pattern	36
Figure	11	Examples of Symbols	36
Figure	12	Symbol Orientation	37
Figure	13	A Typical Engineering Drawing	38
Figure	14	Some Representative Hatch Styles	40
Figure	15	Vertical Lettering	41
Figure	16	Inclined Lettering	42
Figure	17	Microform Lettering	42

I. PURPOSE

The purpose is to register extensions to the Computer Graphics Metafile (CGM) standard through the American National Standards Institute (ANSI) and International Standards Organization (ISO) processes to meet CALS requirements, to ensure that all geometric objects which are displayed graphically in CALS can be represented by the CGM. (Task 2.2.2.2.2)

II. BACKGROUND

The extensions necessary to allow the CGM to properly support CALS requirements for data interchange in the areas of Technical Publishing, Administrative Publishing, and Technical Drawings were derived fom applicable standards, commercial product descriptions, and CALS program documents. The simplest extensions are described as registered linetypes, hatchstyles, and markertypes. Such extensions add no additional "functionality" to the CGM. More complex extensions are described in the form of the Graphical Drawing Primitives (GDPs) and Escapes necessary for their implementation. NBS/ICST concludes that the general framework of the CGM is adequate to support CALS requirements, but that a large number of extensions-- some involving significant new concepts--are required. The effort expended under this year's SOW, although very comprehensive, addresses only a small portion of the CGM extensions which should be registered for CALS use. This report does indicate what should be done in follow up work. This will be discussed in Section V of the report.

1.0 Overview of Computer Graphics Standards

For completeness and to allow comparison with CALS requirements, the output facilities of the family of ISO/ANSI computer graphics standards—including the CGM—are described in this paragraph. This material is drawn from [1].

1.1 Output Primitives

Computer graphics standards provide six output primitives: one line-primitive, one point-primitive, one text-primitive, two raster-primitives and one general purpose primitive serving as an entry point for specific workstation capabilities and as a mechanism for extending the standards in a "standard way."

POLYLINE - generates a set of straight lines connecting a given point sequence.

POLYMARKER - generates symbols of some type centered at given positions. The symbols are called markers; these are glyphs (symbolic characters) with specified appearances which are used to identify a set of locations.

TEXT - generates a character string at a given position.

FILL AREA - generates a polygon which may be hollow or filled with a uniform color, a pattern, or a hatch style.

CELL ARRAY - generates an array of rectangular cells with individual colors. This is a generalization of an array of pixels on a raster device. However, the cells of this primitive need not map one-to-one with the pixels defined by the display hardware.

GENERALIZED DRAWING PRIMITIVE (GDP) - addresses special geometrical output capabilities such as the drawing of spline curves, circular arcs, and elliptic arcs. The objects are characterized by an identifier, a set of points and additional data. All transformations are applied to the points but the interpretation is left to the workstation.

1.2 Output Primitive Attributes

Table 1 gives, for every type of primitive, the set of attributes that control their appearance. The attributes describe the following aspects of output primitives.

Pick identifier - A number assigned to individual output primitives within a segment and returned by the pick device. The same pick identifier can be assigned to different output primitives. These no not apply to the CGM.

Linetype - Linetypes are used to distinguish different styles of lines. A line may be, for example, solid, dashed, or dashed-dotted.

Linewidth scale factor - The actual width of a line is given by a normal linewidth multiplied by the linewidth scale factor.

Color - The color is specified by the red-, green-, and blue-intensities defining a particular color (RGB-values).

Marker type - The marker type is a number specifying the particular glyph used for identification of the polymarker positions.

Marker size scale factor - The actual marker size of a line is given by a nominal marker size multiplied by the marker size scale factor.

Text font - The text font is a number selecting one representation for the text string characters out of the possibilities present on a workstation.

Text precision - An attribute describing the fidelity with which character position, character size, character orientation, and character font of text output match those requested by an application. In order of increasing fidelity, the precisions are string, character, and stroke.

Character height - The vertical extent of a character.

Character up vector - The "UP" -direction of a character.

Character expansion factor - The deviation of the width/height-ratio of a character from the ratio defined by the text font designer.

Text path - The writing direction of the character sequence. The normal path is "right," i.e., the text you are reading is written from left to right. The standards allow the text path values; left, up, and down also.

Character spacing - Space to be inserted between adjacent characters of a text string in addition to the space defined by the font designer.

Character alignment - A text attribute describing how the text string is positioned relative to the reference point of the text primitive (e.g., left aligned, centered).

Interior style - The interior style used to determine in what style an area should be filled. It has one of the values: hollow, solid, pattern, or hatch.

Pattern size - The pattern size specifies the size of the basic pattern rectangle.

Pattern reference point - The pattern reference point specifies the origin of the basic pattern rectangle. The lower left corner of the rectangle is placed at the reference point. Then the pattern is repeated in both directions until the whole area is filled.

Pattern array - A pattern is defined by an array of rectangular cells, each cell having a color assigned to it. These values are used to assign colors to the basic pattern rectangle and to all replications of it.

Hatch style - Hatching is specified by a number selecting one hatch style.

Primitive	Attributes	
POLYLINE	PICK IDENTIFIER LINEWIDTH SCALE FACTOR	LINETYPE COLOR
POLYMARKER	PICK IDENTIFIER MARKER SIZE SCALE FACTOR	MARKER TYPE COLOR
TEXT	PICK IDENTIFIER CHARACTER HEIGHT CHARACTER UP VECTOR CHARACTER EXPANSION FACTOR TEXT PATH CHARACTER SPACING	TEXT FONT TEXT PRECISION COLOR TEXT ALIGNMENT
FILL AREA	PICK IDENTIFIER PATTERN SIZE PATTERN REFERENCE POINT PATTERN ARRAY	INTERIOR STYLE HATCH STYLE COLOR
CELL ARRAY	PICK IDENTIFIER	COLOR
GENERALIZED DRAWING PRIMITIVE	PICK IDENTIFIER dependent on the type of GDP	

Table 1. Output Primitive Attributes

2.0 Registration of Graphical Items

The purpose of this paragraph is to define the framework through which graphics standards are extended by the procedure called **registration**. (Registration procedures do not assign values which are defined as being workstation- or implementation-dependent.) Registration applies to the registration of individual items within the following classes of graphical items as reserved for registration in computer graphics standards:

a) Generalized drawing primitive (GDP) function definitions;

b) Graphical escape function definitions;

c) Linetypes;

d) Markertypes;

e) Hatchstyles;

f) Textfont appearance;

g) Prompt and echo type definitions (for graphical interaction—does not apply to the CGM);

h) Error messages.

Linetype, markertype, and hatchstyle registration adds additional types and styles to supplement those defined in the standards. Their definition is straightforward. Textfont appearance registers the identifiers through which text fonts are accessed by graphics standards. GDP and Escape registration are the mechanisms through which additional functionality is added to a graphics standard. For example, curves can be added through GDP registration and additional line attributes - such as endcaps—can be added through Escape registration. Since this report addresses only CGM extensions, Prompt and Echo Type definition are not considered. Error messages are registered as necessary to support registered GDPs and Escapes.

2.1 Registration in the CGM Standard

The CGM standard (ANSI X3.122-1986 or FIPS 128) describes registration in this way:

> For certain elements, the CGM defines value of range of parameters as being reserved for registration. The meanings of these values will be defined using the established procedures of the ISO International Registration Authority for Graphical Items. These procedures do not apply to values and value ranges defined as being reserved for implementation-dependent or private use; these values and ranges are not standardized.
>
> Applications therefore, shall not use parameter values in the reserved ranges for implementation-dependent or private use. Those elements that will be affected by registration of graphical items are:
>
> a) LINE TYPE;
> b) MARKER TYPE;
> c) HATCH STYLE;
> d) EDGE TYPE;
> e) FONT LIST (allows the selection of named fonts via a font index);
> f) GENERALIZED DRAWING PRIMITIVE;
> g) ESCAPE.

Registration of character sets for use with the CHARACTER SET LIST element is according to the procedures established by ISO 2375, Character Set Registration.

The CGM includes the following "built-in" values for items subject to registration:

1) LINE TYPE:
   1. solid
   2. dash
   3. dot
   4. dash-dot
   5. dash-dot-dot

2) MARKER TYPE:
   1. dot (.)
   2. plus (+)
   3. asterisk (*)
   4. circle (o)
   5. cross (x)

3) HATCH TYPE (called HATCH STYLE in the above extract from the CGM standard):
   1. horizontal equally spaced parallel lines
   2. vertical equally spaced parallel lines
   3. positive slope equally spaced parallel lines
   4. negative slope equally spaced parallel lines
   5. horizontal/vertical crosshatch
   6. positive slope/negative slope crosshatch

4) EDGE TYPE (the built-in ones correspond directly to line types):
   1. solid
   2. dash
   3. dot
   4. dash-dot
   5. dash-dot-dot

5) Text Fonts:
   none

6) GENERALIZED DRAWING PRIMITIVE:
   none

7) ESCAPE:
   none

It may be necessary to extend the CGM by adding functions that cannot be accommodated in the restricted format of registered items. These extensions can be done by adding external elements. The CGM standard describes GDPs, Escapes and External elements in this way:

Generalized drawing primitives

The GENERALIZED DRAWING PRIMITIVE (GDP) is a graphical primitive element that may be used to access device (or implementation) specific graphical primitives that are not accessed by the standardized elements.

Escape elements

ESCAPE elements describe device- or system-dependent data in the CGM. ESCAPEs may be included in the metafile at the discretion of the user, but direct effects and side effects of the use of nonstandardized elements are beyond the scope of this standard. This standard imposes no constraints on the functional intent or content of data passed by the ESCAPE mechanism.

External elements

External elements communicate information not directly related to the generation of a graphical image. They may appear anywhere in the CGM.

The APPLICATION DATA element allows applications to store and access private data. This element is not a graphical element and its interpretation will have no effect on any picture produced by an interpreter.

For specification of non-standardized graphical effects, the ESCAPE and GENERALIZED DRAWING PRIMITIVE elements are provided. These elements may have an effect on the picture produced by an interpreter.

2.2 Why Registration for the CGM Alone is Insufficient

The CGM standard (deliberately) does not standardize the actions of metafile generators or interpreters. However, since metafiles are to be used in a fixed environment—such as the DoD CALS program—to transfer picture data, then the action of both metafile generators and interpreters must be precisely defined. This is done by the definition of **application profiles**. Such a profile for CALS is being developed by NBS/ICST.

Also, graphical items must be registered for use **not only** by the CGM but by all (applicable) standards. Requirements for simple extensions such as linetypes, markertypes, and hatchstyles are similar in all graphics standards. However, the application program interface standards have **more rigorous requirements** for registered Escape and GDP elements. This is illustrated by the following extracts from the Graphical Kernel System (GKS) standard (FIPS 120, ISO 7942-1985), which should be contrasted to those given in Section 2.1 from the CGM:

Escape

Parameters:

In specific escape function identification
In escape input data record
Out escape output data record

Effect: The specified non-standard specific escape function is invoked. The form of the escape data record and which of them are used may vary for different functions. Also the GKS states allowing the invocation of a specific escape function may be restricted. The following rules govern the definition of a new specific escape function:

a) the GKS design concept (see Section 0 - introduction)
b) the GKS state lists are not altered.
c) the function does not generate geometrical output.
d) any side effects are well documented.

Specific escape functions may apply to more than one workstation, for example, all open workstations or all active workstations. The escape input data record can include a workstation identifier where this is required.

Note: Examples of specific escape functions anticipated at present are:

a) support of raster devices allowing the display of more than one frame buffer,

b) use of raster or hardware to manipulate data previously output by cell array.

Where the specific escape function identification is bound to an integer in a programming language specific escape function identifications greater than 0 are received for registration or future standardization and specific escape function identifications less than 0 are implementation dependent.

Specific escape function identifications are registered in the ISO International Register of Graphical Items, which is maintained by the Registration Authority. When a specific escape function has been approved by the ISO Working Group on Computer Graphics, the specific escape function identification will be assigned to the Registration Authority.

Generalized drawing primitive (GDP)

Parameters:

In number of points
In points
In GDP identifier
In GDP data record

Effect: A Generalized Drawing Primitive (GDP) of the type indicated by the GDP identifier is generated on the basis of the given points and the GDP data record. The current values of the entities in the GKS state list ... for the set of polyline, polymarker, text or fill area attributes are bound to the primitive. These attributes are listed in ... When the GDP generates output at the workstation, zero or more of the sets of attributes are used. These are the sets of attributes most appropriate for the specified GDP function and are selected for the

GDP as part of the definition of the GDP. (They are defined in the workstation description table).

Note: The parameters are transmitted to the workstation and interpreted in a workstation dependent way that special capabilities of the workstation can be addressed. Even if errors occur, the GDP is displayed on all active workstations capable of doing so. For example, some of the parameters anticipated at present are:

a) circle points given are centre, peripheral point,
b) circular arc points are centre, start point, end point to be connected anticlockwise in world coordinates,
c) elipse points given are 2 focal points, peripheral point,
d) elliptic arc points given are 2 focal points, start point, end point to be connected anticlockwise is world coordinates,
e) interpolating curve (for example, spline) points given are interpolated.

The recommended set of attributes to use for the above GDP examples would be the polyline attributes.

It should be emphasized that the points, specified as parameters, are transformed by GKS after the interpolation of the points (as defining say, a spline curve or circle) is performed by the active workstation. For example, a GDP, which defines a circle, would appear as an ellipse when the transformation has differential scaling for the two axes. Each specific GDP definition defines how the transformation is applied to both the points and the shape of the GDP. Though the points cannot be clipped, the resulting output of the GDP is clipped against the clipping rectangle, if the "clipping indicator" entry is the GKS state list is CLIP, and the workstation window. If a specific GDP is available on a workstation but is unable to be generated because the current transformations or clipping rectangle are such that the preceding conditions would be violated, an error occurs.

The GDP data record attribute list may contain additional data for each point (for example. vertex order for splines) which remain untransformed. These have to be defined for a specific GDP. In defining a new GDP, the GKS design concepts (see Section 0) are not violated. The set of generalized drawing primitives implemented on a workstation may be empty.

Where the GDP identifier is bound to an integer in a programming language, GDP identifiers greater than 0 are reserved for registration or future standardization and GDP identifiers less than 0 are implementation dependent.

GDP identifiers are registered in the ISO Register of Graphical Items, which is maintained by the Registration Authority. When a GDP has been approved by the ISO Working Group on Computer Graphics, the GDP will be assigned by the Registration Authority.

2.3 The Required Approach for CALS Registration

Based on the above discussion, when writing actual registration proposals for graphical items required for CALS, the following must be done:

1) define how the item is implemented in all applicable graphics standards;
2) develop all necessary language bindings;
3) precisely define the semantics of the item.

Registration proposals developed contain a semantic specification that is complete and precise enough to define the required actions by metafile generators and interpreters. Portions of these descriptions may prove too explicit to be approved by the required ANSI and ISO standards committees. (These committees are accustomed to specifying items as loosely as possible to allow as many implementations as possible to conform.) In this case, some material may need to be moved from the registration proposal to the CALS application profile. In addition, some extensions may only be achievable through the use of the CGM External Data element.

3.0 Applicable Standards

The standards listed in this section were identified as being applicable to technical drawings and automated (technical and administrative) publishing. Drawing and drafting standards were included in the list since they are the best source of information on the "graphical presentation" requirements for engineering drawings. Product data standards were included since they are often (mistakenly) used for picture transfer purposes and, consequently, contain features relating to the "presentation" requirements for such data. The FIPS, ANSI and ISO standards in the areas of graphics standards and product data definition are not repeated here, since they are adequately documented in last year's final report.

Technical publishing

MIL-M-38784B - Manuals, Technical: General Style and Format Requirements

Description of drawing forms

ANSI Y14.1 - Drawing Sheet Size and Format (1980)
ANSI Y14.2M - Line Conventions and Lettering (1979)

Lists: purpose, classification, and preparation

ANSI Y14.1-1980 - Drawing, Sheet Size and Format
ANSI Y14.34M-1982 - Parts Lists, Date Lists and Index Lists
ANSI Y32.16 - Reference Designations for Electrical and Electronic Parts and Equipment

DoD-D-1000B Drawings, Engineering and Associated Lists

Printed board drafting practice

MIL-STD-100 - Engineering Drawing Practices
MIL-STD-275 - Printed Wiring for Electronic Equipment
MIL-STD-429 - Printed-Wiring and Printed-Circuit Terms and Definitions
MIL-STD-1494 - Multilayer Printed Wiring Boards for Electronic Equipment

ANSI Y14.5 - Dimensioning and Tolerancing
ANSI Y32.16 - Reference Designations for Electrical and Electronic Parts and Equipment

IPC-T-50 - Terms and Definitions
IPC-D-310 - Suggested Guidelines for Artwork Generation and Measurement Techniques for Printed Circuits
IPC-D-350 - Printed Board Description in Digital Form
IPC-D-390 - Guidelines for Design Layout and Artwork Generation on Computer Automated Equipment for Printed Wiring
IPC-CM-770 - Component Mounting Guidelines

MIL-D-8510 - Drawings, Undimensioned, Reproducible, Photographics and Contact: Preparation of
MIL-I-46058 - Insulating Compound, Electrical for Coating Printed Circuit Assemblies
MIL-P-55110 - Printed Wire Boards

Line conventions and lettering; signs and symbols

ANSI Y14.2M-1979 - Line Conventions and Lettering
ANSI Y10.3 - Quantities Used in Mechanics of Solids: Letter Symbols for
ANSI Y10.20 - Mathematic Signs and Symbols for Used in Physical Sciences and Technology (include supplement ANSI Y10.20a 1975)

Drawing preparation for microfilming

ANSI Y14.1 - Drawing Sheet Size and Format
ANSI Y14.2M - Line Conventions and Lettering

NMA - MS 100-1971 Glossary of Micrographics Terms

MIL-M-9868D Microfilming of Engineering Documents

Other drawing practices

ANSI Y14.7.1 - Gear Drawing Standards—Part 1 for Spur, Helical, Double, and Rack
ANSI Y14.7.2 - Gear and Spline Drawing Standards—Part 2, Bevel and Hypoid Gears
ANSI Y14.17 - Fluid Power Diagrams
ANSI Y14.13M - Engineering Drawing and Related Documentation Practices—Mechanical Springs

Dimensioning and tolerancing (general dimensions)

ANSI Y14.3 - Multi and Sectional View Drawings
ANSI Y14.5M - Dimensioning and Tolerancing
ANSI Y14.6 - Screw Thread Representation, Engineering Drawing and Related Documentation Practice

Reference designations for electrical parts and equipment; electrical diagrams

ANSI Y32.16 - 1975 Reference Designations for Electrical and Electronics Parts and Equipment
ANSI Y32.2 - 1975 Graphic Symbols for Electronic Diagrams
ANSI Y14.15 - Electrical and Electronics Diagrams (Includes supplements ANSI Y14.15a and Y14.15b 1973)
ANSI Y14.15b - Electrical and Electronics Diagrams (supplement to ANSI Y14.15 1966)

Note: These standards have been accepted for use by the Department of Defense.

MIL-STD-15-2 - Electrical Wiring Equipment Symbols for Ships' Plans Part 2

Surface texture

ANSI-B46.1-1978 - Surface Texture
ANSI-Y14.36-1978 - Surface Texture Symbols

Abbreviations - for use on drawings and technical documentation

ANSI Y1.1 - Abbreviations

MIL-STD 12 - Abbreviations for Use on Drawings, Specifications, Standards, and Technical Documents

Graphic symbols for metal joining and nondestructive testing symbols (welding)

AWS A2.4-76 - Symbols for Welding and Nondestructive Testing.

MIL-STD-25B - Ship Structural Symbols for Use on Ship Drawings

Specification practices

MIL-STD-490 - Specification Practices
MIL-S-83490 - Specifications, Types and Forms
DOD-STD-100 - Engineering Drawing Practices
MIL-STD-961 - Outline of Forms and Instructions for the Presentation of Specification
MIL-STD-1679 - Weapon Systems Development (Navy)
DSM 4120.3M - Standardization Policies Procedures and Instructions

ISI metric practice

ASTM E380 - Standard for Metric Practices
DOD-STD-1476 - Metric System, Application in New Design

Production identification

MIL-STD-130 - Identification Marking of US Military Property
MIL-P-15024 - Plates, Tags, and Bands for Identification of Equipment
MIL-19834 - Plates, Identification or Instruction, Metal Foil, Adhesive Backed, General Specifications for
MIL-P-83497 - Procedures for Preparation of Programmed Tapes and Cards

Communication of product data

ANSI Y14.26.3 - Dictionary of Terms for Computer-aided Preparation of Product Definition Date (including Engineering Drawings)
ANSI Y14.26 - Engineering Drawing and Related Documentation Practices—Digital Representation for Communication of Product Definition Data

DoD/CALS specific standards

Draft MIL-STD-1840A - Automated Interchange of Technical Information
DOD-D-(IGES) - Digital Representation for Communication of Product Data: Application Subsets

4.0 Reference Model

For purposes of this task, a very simple reference model has been adopted to define how the CGM is used in technical drawing production and publishing applications.

For the purposes of this work, it is assumed that the CGM will be used in the following manner in CALS:

1) CGM pictures will be used to transfer engineering drawings.

2) CGM pictures will be used to transfer pages of technical publications which contain both text and graphics; the extensions being defined herein will allow the CGM alone to be used for transferring the content of document pages—text, geometric graphics, and image (raster) graphics. The CGM may also be used to transfer pictures which are then embedded in documents by some process outside the scope of this study.

No assumptions were made in the following areas:

1) The manner in which graphical metafiles are transferred.

2) The embedding of CGM content in documents defined in other standards (for example, ODA and SGML).

3) The manner in which descriptions of externally defined items are made available to a CGM interpreter.

4) Transfer of "processible" product definition data or "processible-form" office documents. (It is assumed IGES and/or ODA/ODIF are used in these cases).

III. DISCUSSION

1.0 Technical and Administrative Publishing

1.1 Introduction

It is widely recognized that computer graphics standards, CGM in particular, will need to be extended to properly support publishing in the CALS environment. Such extensions will be initially developed in the form of GDPs and Escapes and the "folded into" the standards during their next revision cycles.

The following liaison statement (between the two relevant standards bodies) from ASC X3H3 (Computer Graphics) to ASC X3V1 (Office Systems) summarizes some of these changes:

Liaison Statement to X3V1 TG8 Concerning TPM (Text Presentation Metafile) User Requirements (X3H3/87-31)

1. Rapid soft copy display of formatted composite documents.
2. Use of internationally standardized color models in interchange.
3. Use of standard methods for image compression.
4. General fill boundary specification.
5 Wide lines with joint and end features, and patterns with anchoring.
6. Curves, including conics and Bezier curves.
7. General clip boundary.

X3H3 would like to work with the TPM developers on graphics aspects to achieve two goals:

1. Any functionality in TPM which is close to functionality in GKS or CGM should be identical.
2. Any graphic functionality required by TPM but not present in CGM should be developed jointly so that it can be used compatibly by existing and future graphics and office systems standards.

Subsequent sections describe first the methodology by which extensions in the publishing area were determined, and then the general features that must be added to the CGM. These general features are given by way of example of current proprietary products.

1.2 Required Capabilities CGM Must Be Extended to Meet

In the absence of standards in this area (the Text Presentation Metafile under development in X3V1 will not be available—even in draft form—until late 1987), the features of the most widely used commercial products have been used to define the required capabilities. In particular, the PostScript Page Description Language [5,6] of Adobe Systems has been selected to provide a baseline set of capabilities. In some cases, features from the Interpress Page Description Language [4] of Xerox Corporation have been used instead of or in place of those in PostScript.

Defining required capabilities in the publishing area is more difficult than for engineering drawing. since format and presentation are determined more by artistic considerations than slavish adherence to format standards. MIL-M-38784B contains requirements for high-level formatting and no "presentation" requirements. Unless noted, the capabilities discussed below have been made into registration proposals, and can be found in the Recommendations section.

1.2.1 Lines

"Lines" in PostScript are drawn through a two-step process. First, a **path** is constructed, corresponding to the line to be drawn. This is done by a sequence of **moveto** and **lineto** operators. (Relative operators, or curve operators can be interspersed. Appendix B provides a list of all PostScript operators, and detailed descriptions of the ones used for graphics.) Next, the **stroke** operator paints the path with the current color and line width.

The PostScript language gives complete control over how the **stroke** operator converts a path into a painted line or curve. The **setlinewidth** operators determine the width of the stroked line. There are several operators that allow other characteristics of a stroked path to be precisely determined. Among these are:

1) **setlinecap** determines the appearance of line segment ends;

2) **setlinejoin** determines the method by which different line segments are joined;

3) **setdash** determines the pattern for dashed lines.

These operators are described below in detail:

1.2.1.1 Setlinecap

The setlinecap operator takes a number from the stack and uses it as a code determining how PostScript will end stroke line segments. For example, the program line

```
1 setlinecap
```

would cause PostScript to paint all line segments with round ends.

There are three values for the line cap code:

0. Butt caps - The line segment has square ends perpendicular to the path. This is the PostScript default line cap.

1. Round caps - The line segment ends with semicircular caps with diameters equal to the widths of the line.

2. Projecting square caps - These are similar to butt caps, but extend one-half of a line width beyond the line segment's endpoint. (These three styles are illustrated in Figure 1).

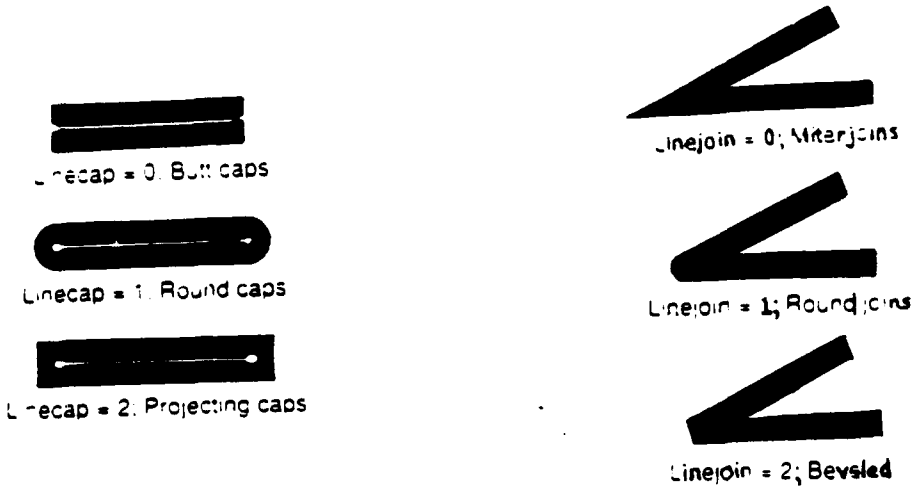


Figure 1. Linecap and Linejoin Capabilities

1.2.1.2 Setlinejoin

When two connected line segments are stroked, PostScript needs to make a decision about what type of join to use between them. The **setlinejoin** operator tell PostScript how to join connecting line segments. This operator is similar to **setlinecap**, in that it takes a code from the top of the stack. This code can have values from zero to two, corresponding to the following types of line joins:

0. Mitered join - The edges of the stroke are extended until they meet. This is the default join. This join is affected by the current *miter limit* (see below).

1. Rounded join - The segments are connected by a circular join with a diameter equal to the line width.

2. Bevel join - The segments are finished with butt end caps and the notch at the larger angle between the segments is filled with a triangle.

These three styles are illustrated in Figure 1.

1.2.1.3 Setmiterlimit

Mitered joins can present a problem. If two line segments meet at an extremely small angle, the mitered join can produce a spike that extends a considerable distance beyond the intersection of the path segments. To prevent this, the join switches from mitered to beveled when the angle between line segments becomes too acute. Figure 2 illustrates how the **setmiterlimit** operators is used to control this effect.

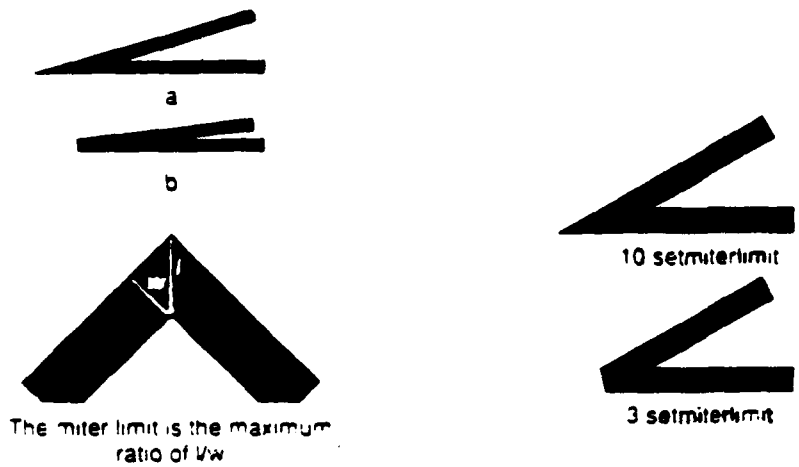


Figure 2. Setmiterlimit Capabilities

1.2.1.4 Setdash

The current path is normally stroked with a solid line. Other methods of stroking a path are possible, however, The PostScript graphics state includes a *dash array* and a *dash outset*. that together describe what pattern of alternating black and white dashes should be used to stroke paths.

This pattern is set by the **setdash** operator, which takes an array and a number from the stack and makes them the current dash array and offset. The array contains a set of numbers, such as

[3 5 1 5]

which represent the lengths of alternating black and white segments should make up a stroked line.

The array above would cause all paths to be stroked with a repeating sequence consisting of three units of black, five units of no ink, one unit black, five units no ink. This pattern will repeat along the entire stroked path. This is illustrated in Figure 3.

[3515] 0 setdash

Figure 3. Setdash Capabilities

The second argument passed to **setdash** is the offset within the dash pattern where the stroke operator is to start when it prints a line. That is, if we were to set the dash pattern with the line

[6 3] 3 setdash

stroked lines would begin three units into the pattern, or half way through the first long dash.

1.2.1.5 Closepath

A final necessary capability is provided by the **closepath** operator. This operator adds a line segment to the current path connecting the current point to the last point addressed by a **moveto** operator. Figure A shows an example of a box containing a notch in one corner since it hasn't been "closed." After the **closepath** operator is applied (with mitered joins) the result is "better box" in Figure 4.

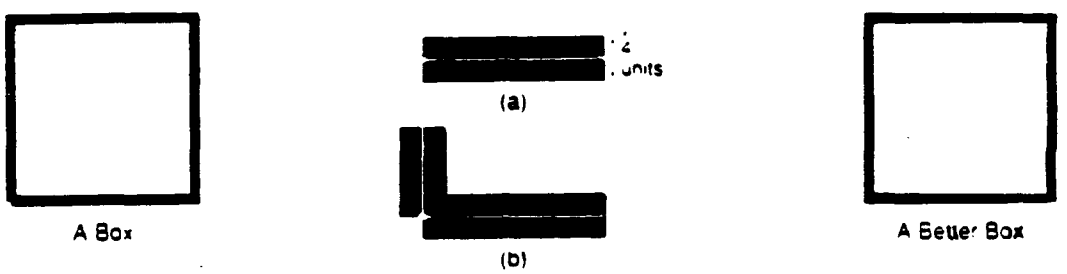


Figure 4. Closepath Usage

1.2.2 Symbols

PostScript provides no explicit modular symbol capabilities. Its program language-like structure however, allows graphical patterns to be defined once and positioned at various points to provide the necessary capability. This topic is explored further in Paragraph 3.8.

1.2.3 Curves

PostScript provides the following operations to draw curve line segments:

arc - appends a counterclockwise arc of a circle to the current path, possibly preceded by a straight line segment.

arcn - behaves like **arc**, but builds its arc segment in a clockwise direction.

arcto - appends an arc of a circle to the current path, preceded by a straight line segment.

curveto - adds a Bezier cubic section to the current path .

moveto - starts a new subpath of the current path, sets the current point in the graphics state to the user coordinate without adding any line segments to the current path.

closepath - closes the current subpath by appending a straight line segment connecting the current point to the subpaths's starting point.

rlineto - appends a straight line segment to the current path in the same manner as **lineto**; however, the number pair is interpreted as a displacement relative to the current point rather than as an absolute coordinate.

rmoveto - starts a new subpath of the current path in the same manner as moveto, however, the number pair is interpreted as a displacement relative to the current point rather than as an absolute coordinate.

rcurveto - adds a Bezier cubic section to the current path in the same manner as curveto; however, the three number pairs are interpreted as displacements relative to the current point rather than as absolute coordinates.

stroke - paints a line following the current path and using the current color.

strokepath - replaces the current path with one enclosing the shape that would result if the stroke operator were applied to the current path.

setflat - sets the flatness in the current graphics state to *num*, which must be a positive number. Flatness is used to control the accuracy with which curves are rendered on an output device

clip - intersects the inside of the current clipping path with the inside of the current path to produce a new (smaller) current clipping path.

Additional details are contained in Appendix B which provides a complete description of each operator mentioned above. [**Registration proposals have been prepared for the curve capability. Some others have equivalent facilities in the CGM. Setflat, closed figure and stroke path require further study before generalization and incorporation in the CGM.**]

1.2.4 Text

Rather than draw the required text capabilities from PostScript—which represents only one of many technologies for defining and rendering text, ISO DP 9541 Font and Character Information Interchange is used. There is agreement in principle amongst the various standards committees that the capabilities represented in this draft standard should be adopted by computer graphics standards.

ISO DP 9541 defines font referencing, positioning, and presentation. Minimal and complete capabilities are defined in each of these areas as follows:

1.2.4.1. Objectives and Functional Definitions

The objective of the Font and Character Information Interchange Standard is to define a common font resource architecture which can be used in a variety of development and application environments, for the purpose of supporting text (characters, symbols, ideograms) generation, interchange, and presentation. The font resources which are developed to this architecture may be used by: text and graphic editor, document formatters, utilities, device service programs, resource management programs, and presentation device drivers. Figure E provides a high level look at the environment in which ISO DP 9541 operates. Figure F decomposes Figure E to provide a lower level look at the publishing environment.

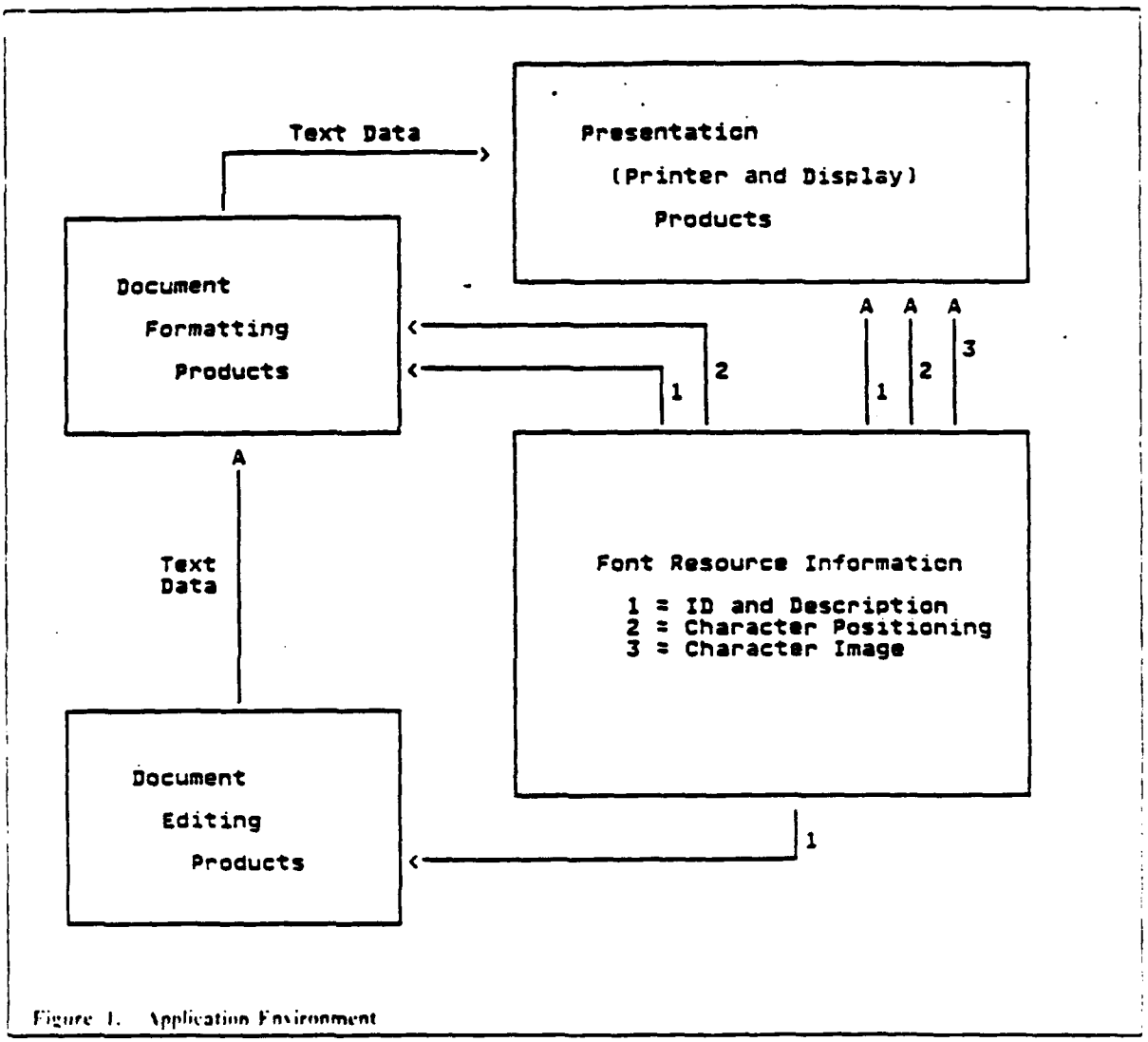
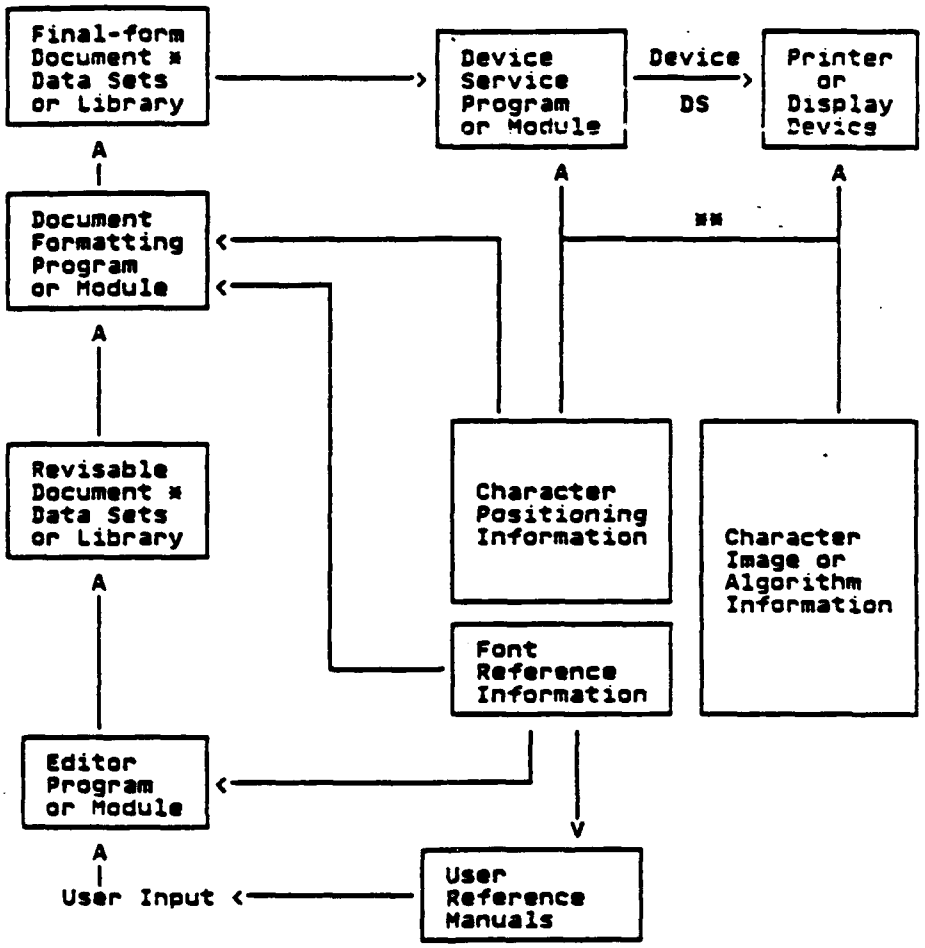


Figure 5. ISO Font Architecture - Application Environment



* : Final-form or Revisable documents may be routed to other programs, devices, systems, and sites.

** : Positioning information and shape information may be down-loaded to a device via a device service facility or may be resident in the device on a diskette, hard file, cartridge, etc.

Figure 2. Application Environment (low level view)

Figure 6. A Lower-level Look at the Font Application Environment

Consistency of font resource information is a requirement for softcopy display of documents to be typeset, and interchange of documents between systems.

The degree of consistency can be significantly improved through the use of a common font resource architecture.

A **font resource** is defined to be the total collection of information required to characterize and identify a font to compose blocks of text and to define the images of the characters, for use by an electronic data processing system. A font resource will contain information that:

1) identifies and describes itself to permit selection by users and application programs.

2) defines the character attributes required by a document formatting process to position the characters on a presentation surface.

3) defines each character's shape for generation of the character images on the presentation surface.

Font production is defined to be the process of designing the character images, converting those images into a digital technology format (bit image, vector drawing orders, outline algorithm, etc.), defining the various height, width, and escapement values for each individual character, assigning appropriate descriptive and identifying information (to the characters and the font resource in general), and creating a font resource that contains all of the required information in a format that can be used by a text processing system.

Text processing system is defined to be the total collection of hardware devices and software or firmware programs required to generate, modify, display, and/or print text. Those components (devices and programs) may be contained in a single hardware device that processes only text or may be included within a larger document processing system and/or communication network. Text processing may be the primary function of the component (text editor) or it may be only an auxiliary function of the component (graphics editor, device service program, resource management program).

Font storage and access is defined to be the process of storing the font file information on the appropriate media for use within a text processing system, and the process of accessing that information by the various components of a text processing system. When a font resource is first generated, all of the required information may be contained in a single font resource file, but that may not be the format that is most useful within a text processing system

Font referencing is defined to be the process of identifying or characterizing a font. The referencing task affects editing, formatting and presentation because it is necessary **for the user** to specify the desired fonts in the document, **for the formatter** to identify the fonts and find the required character positioning information from the appropriate font resource, and **for the presentation process** to identify the fonts and find the required character positioning and shape information from the appropriate font resource. Referencing may include identification of a specific font resource, or simply provide sufficient descriptive information about the desired font that an alternative font resource can be found if the specified one is not available to the system which will format or present the document.

Character positioning is defined to be the process of determining where a given character is to appear on the presentation surface. This function is performed by the document formatting process, which is a generic name for any process that determines the text, graphic, or image content and format of a document, including pages and line breaks and how text should flow around graphics

or image objects that also appear in the document. The document formatting process makes use of font resource information along with user, system, and document content information. Thus, font resources provide only a portion of the information required for character positioning. Characters may be positioned absolutely or relatively. That is, the formatting process may specify the precise location where each individual character is to be positioned or the formatting process may specify the content and beginning of a string of characters. In either case, the process must know the image and escapement extents for each individual character, and other character attributes dealing with coordinate system, reference point, rotation, kerning, ligatures, etc.

Image presentation is defined to be the process of forming the character image on the presentation surface. This process is actually performed by a hardware device (display or printer), but may be supported by a series of software and/or hardware processes which translate the character image information from its font resource format to the format or control codes required by the presentation device. Font resources, defined according to this standard, may support a variety of shape definition techniques, some public and some private. Some font designs are privately owned and the key to translation of the character shape information is only available through license agreements. Other font designs are in the public domain and anyone may have access to the character shape information. In addition, other practical considerations may suggest the use of other image technologies.

1.2.4.2 Font Referencing

Minimal Referencing

The minimal subset of font referencing supports simple identification of the font resource design and character content, but does not support description of the font resource to a level of detail required for support of document rendering (fidelity level) specification. The subset only includes **font name.**

Complete Referencing

The complete subset level of the font referencing supports description of the font resource to a level that permits selection or approximation of a font resource based on the description provided. The selected font must be described to a level of detail that accurately identifies the attributes of the font resource that was chosen for document formatting. If it becomes necessary for the presentation service to substitute another font resource, it must find one that most closely corresponds to both the user and formatter specified fonts.

The subset includes the following attributes (in addition to those defined for the minimal subset):

Control attributes
- body size units

Control attributes
- character collection

Format attributes
- font size
- minimum size
- maximum size

Layout attributes
- maximum height
- x height
- cap height

ascender height
descender depth
minimum escapement
average escapement
maximum escapement
weighted average escapement
lower case escapement
upper case escapement
digit escapement type
writing mode

Appearance attributes
posture
structure
mood
function
character category
font category
typeface name
design classification
escapement class

hairline weight
relative weight
absolute weight
relative width
absolute width
relative height
absolute height

1.2.4.3 Character Positioning

Minimal Positioning

The minimal subset of character positioning supports character increment fonts, but does not support proportionally spaced fonts where each character has a defined escapement value.

This subset includes the following attributes:

Control attributes
font name
escapement class
bodysize units

Positioning attributes
average escapement
font size

Complete Positioning

Given the document formatting requirements and text content, the escapement required for each character must be determined. The formatting process may designate that a character may be positioned anywhere on the presentation surface. To prevent unwanted character overlap, excessive gap between characters, or characters running off the edge of the presentation space, the values of each character's recommended escapement must be known.

This subset includes the following attributes (in addition to those defined for the minimal subset):

Control attributes
- minimum space amplification
- maximum space amplification
- pairwise escapement adjustment

These attributes are repeated for each character and for each writing mode supported by the font resource.

Positioning attributes
- character name
- writing mode
- position point
- escapement point
- extents (4 values)
- sectored space adjustment
- amplification
- correction
- white space adjustment
- op's

Formatting programs may use the positioning information to determine each character's position on a presentation surface, with each character being presented at its designated point. Or, the formulating program may designate the starting point (and the text character string, with each character providing information needed for the next character's presentation position).

1.2.4.4 Image Presentation

The information contained in these subsets may be stored and used separately from the information contained in the function sets defined for referencing and positioning. A font resource may not be identified as supporting one of these subsets unless it contains all of the attributes defined for that subset. That is, if a resource contains all font attributes except one from the minimal subset, then the resource cannot be identified as supporting either the minimal or the complete subset.

Minimal Presentation

The minimal subset level of character presentation supports only the bitmap format of character image definition. Alternate formats will not be recognized.

This subset includes the following attributes:

Control attributes
- font name
- bodysize units

Presentation attributes (per character and rotation)
- character name
- character shape information

The attributes for this format will be defined.

This is the basic format required for definition of character shapes for the registration of characters for Part 3 of the ISO DP 9541 standard.

Complete Presentation

Given the positioning point for a character, the shape of the character must be presented on the presentation surface using the device technology applicable to that device. All character shape information is defined relative to an origin, which has a defined x-y offset and rotation from the escapement box origin (current positioning point). Non-ISO character image technologies do not need to be supported by this subset level.

Control attributes
- image technology

Presentation attributes (per character and rotation)
- character-shape information for:
  - straight line outline
  - circular outline
  - conic outline
  - Bezier outline
  - composite

The shape information may be defined separately for each device technology, but there should be one set of device independent positioning information for each resource ID. Thus, the shape information may be repeated for several different technologies in any one font resource, or the shape and positioning information may be contained in separate files with cross reference pointers.

1.2.5 Images

PostScript includes a very minimal set of capabilities for handling images (raster bitmaps). The operators provided are :

image—renders a sampled image onto the current page. The sampled image is a rectangle array of *width x height* sample values, each of which consists of bits sample bits of data (1, 2, 4, or 8).

imagemask—similar to the **image** operator, however, it tracks the source image as a *mask* of 1 bit sample that is used to control where to apply paint (with the current color) and where not to. It is most useful for printing characters as bitmaps. Such bitmaps represent masks through which a color is to be transformed.

The image capabilities are too limited to meet CALS requirements (or those of the publishing industry in general). This topic requires further study. **[The CGM can be extended to incorporate a very general set of facilities compatible with the raster portions of Group 4 facsimile.]**

1.2.6 Definition and Instantiation of Objects

One capability that is missing in the CGM is the ability to define an arbitrary picture component (e.g. a symbol or non-primitive object) which can be instanced (repeated) multiple times with a single picture or over shared amongst different pictures in a document.

As will be further discussed, this capability is needed for both publishing and engineering drawing applications.

The capabilities of both PostScript (based on a programming language approach) and Interpress (based upon defining extended operators) are described below. Appendix B contains detailed descriptions of the corresponding PostScript and Interpress features.

1.2.6.1 PostScript Procedures

A PostScript **procedure** is a set of operations grouped together with a common name. This set of operations is stored with its **key** in a dictionary. When the key appears in a program, the associated set of operations is carried out.

Procedures are defined in exactly the same way as variables. The program must place the procedure name (preceded by a slash) on the stack, followed by the set of operations that make up the procedure. Then, the **def** operator is used to store the operations and name in the current dictionary.

1.2.6.2 Interpress Symbols

The concepts of **symbol** and **instance** are provided in Interpress by composed operators and transformations. A graphical symbol can be defined as a composed operator. When an instance, or copy, of the symbol is to be printed, the current transformations will be applied to all coordinates as the symbol calls imager operators. The properties of the current transformation will thus, determine the position, size, and rotation of the instance on the printed page.

The principal use of symbols and instances in Interpress is for printing characters. Each character is defined by a composed operator, called a character operator. These operators are invoked usually by SHOW, with a current transformation that achieves the proper size, orientation and position of the character.

Instancing can also be used for other purposes. Graphical objects that are repeated often on a page or throughout a document may be tracked as instances. A symbol is defined as a composed operator and called with an appropriate current transformation in order to generate each instance.

2.0 Technical Drawings and Product Data

2.1 Introduction

It is recognized that some extensions to computer graphics standards are necessary to properly support the creation of drawings and views of product model data. The following extract from the *Final NBS Report for CALS - FY86* summarize a **contractor's** findings:

> The addition of several facilities to the graphics standards would greatly improve the ability of these standards to efficiently represent the drawings and views implicit in IGES and PDES files. These additions are described in the following paragraphs.
>
> - Support for full conics including hyperbolas and parabolas
> - Support for splines, including at least one of the parametric spline and rational B-spline representations.
> - Support for surface definitions, including surfaces of revolution and cylinders.
> - Support for a ROUNDED RECTANGLE output primitive.
> - Support for many new line types, including some or all of the 11 forms of "arrows" defined in IGES through registration and subsequent standardization.
> - Support for the CENTERLINE symbol as new standardized marker type.
> - Addition of facilities to more directly control and specify such features of text strings as subscripts, superscripts, and fractions. At present, these features can be generated only by using the more primitive APPEND TEXT and by changing associated text attributes like CHARACTER HEIGHT, CHARACTER SPACING, and TEXT ALIGNMENT.
> - In IGES/PDES the slant of the text is independently specified from the type face (e.g., Helvetica Italics Bold). The Computer Graphics Interface (CGI) standard allows, via the Character Orientation element, text characters to be skewed ("slanted"). This feature is not directly available in the Graphical Kernel Sytem (GKS) standard and the Programmers Hierarchical Interface Graphics System (PHIGS) standard; it can occur only as a result of the segment or structure transformations.
> - In PDES, continuous text alignment is used to align multiple text strings. This feature is also available in CGI/CGM. It should be added to GKS, GKS-3D (GKS in three dimensions) and PHIGS.
> - The six predefined IGES SECTION entity patterns not corresponding to standardized CGI/CGM patterns should be registered.
> - Support for user-defined line types.
> - Support for multiple color tables.

NBS/ICST generally supports these conclusions, although recommendations made in this report are more comprehensive and somewhat different than those above for extensions in several areas. Further, this work is limited to extensions to the CGM as dictated by the actual requirements from standards for product data description and drawings themselves, rather than being limited to investigation of the IGES standard (which does not describe engineering drawings, per se, but rather the transfer of the complete information necessary to describe the representation of a product.)

The methodology used to define the capabilities required to define images of engineering drawings was the following:

1. Review requirements from the standards for the production of engineering drawings listed in Section 3.0 of the Background section above; then extract the required representational capabilities needed in the CGM.

2. Review the IGES standard to extract entities for which the CGM must be able to describe images; then extract the required representational capabilities needed in the CGM.

3. Consider the CALS system level requirements in areas such as use of symbol libraries. Since no such requirements exist in explicit form and there is no well-defined CALS reference model, the capabilities of commercially available products were used to derive requirements.

2.2 IGES Entities

IGES Version 3.0 contains the following types of entities. Each of these entities and suggestions on how it might be rendered by systems implementing graphics standards are described in Appendix C. This material was extracted from the *Final NBS Report for CALS - FY86*. The CGM must be capable of describing images (graphical representations) of these entities:

CIRCULAR ARC
COMPOSITE CURVE
CONIC ARC - **registration proposal prepared**
COPIOUS DATA
PLANE
LINE
PARAMETRIC SPLINE - **registration proposal prepared**
PARAMETRIC SPLINE SURFACE
POINT
RULED SURFACE
SURFACE OF REVOLUTION
TABULATED CYLINDER
TRANSFORMATION MATRIX
FLASH ENTITY
RATIONAL B-SPLINE - **registration proposal prepared**
RATIONAL B-SPLINE SURFACE
OFFSET CURVE
CONNECT POINT
NODE
FINITE ELEMENT
NODAL DISPLACEMENT AND ROTATION
OFFSET SURFACE
CURVE ON PARAMETRIC SURFACE
TRIMMED (PARAMETRIC) SURFACE
ANGULAR DIMENSION
CENTERLINE
DIAMETER
FLAG NOTE
GENERAL LABEL
GENERAL NOTE
LEADER
LINEAR DIMENSION
POINT DIMENSION
RADIUS DIMENSION
SECTION

GENERAL SYMBOL
SECTIONED AREA
WITNESS LINE
ASSOCIATIVITY DEFINITION
ASSOCIATIVITY INSTANCE
DRAWING
LINE POINT DEFINITION
MACRO CAPABILITY
PROPERTY
SUBFIGURE DEFINITION
TEXT FONT DEFINITION
VIEW ENTITY
EXTERNAL REFERENCE ENTITY
NODAL LOAD/CONSTRAINT ENTITY
COLOR DEFINITION ENTITY
TEXT DISPLAY ENTITY

2.3 Required Capabilites

2.3.1 Lines

The following material is drawn from ANSI Y14.2M-1979 (Line Conventions and Lettering) and explains how "lines" are used in product definition drawings:

Line widths - Two widths of lines are recommended for use on manually prepared drawings. (See Figure 7.) One width of line is acceptable on drawings prepared by automated methods. The ratio of line thickness should be approximately two-to-one. The thin-line width should be approximately 0.35 mm or 0.016 inch and the thick-line width approximately 0.7 mm or 0.032 inch. The actual width of each line should be governed by the sizes and styles of drawing, and the smallest size to which it is to be reduced. All lines of the same type should be uniform throughout the drawing. Spacing between parallel lines should be such that there is no fill-in when reproduced by available photographic methods. Note: spacing of no less than 1.5 mm (0.06 inch) normally meets reproduction requirements.

Line quality - All lines should be clean cut, opaque, uniform, and properly spaced for legible reproduction by all commonly used methods, including microfilming in accordance with industry and government requirements. When manually produced, there should be a distinct contrast between the two widths of lines. Contrast must be obtained only by variance in the relative widths of the lines. In no case should contrast be achieved by a difference in density, opaqueness or color.

Visible lines - The visible lines, Figure 7 and 8, should be used for representing visible edges or contours of objects. Visible lines should be drawn so that the views they outline clearly stand out on the drawing with a definite contrast between these lines and secondary lines.

Hidden lines - Hidden lines, Figures 7 and 8, consist of short evenly spaced thin dashes and are used to show the hidden features of the object. The lengths of the dashes may vary slightly in relation to the size of the drawing. Hidden lines should always begin and end with a dash in contact with the visible or hidden line from which they start or end, except when such a dash would form a continuation of a visible line. Dashes should join at corners, and arcs should start with dashes at tangent points. Hidden Lines should be omitted when their use is not required for the clarity of the drawing. Although features located transparent materials may be visible, they should be treated as concealed features and shown with hidden lines.

Section lines - Section lines, Figures 7 and 8, are used to indicate the cut surfaces of an object in a section view.

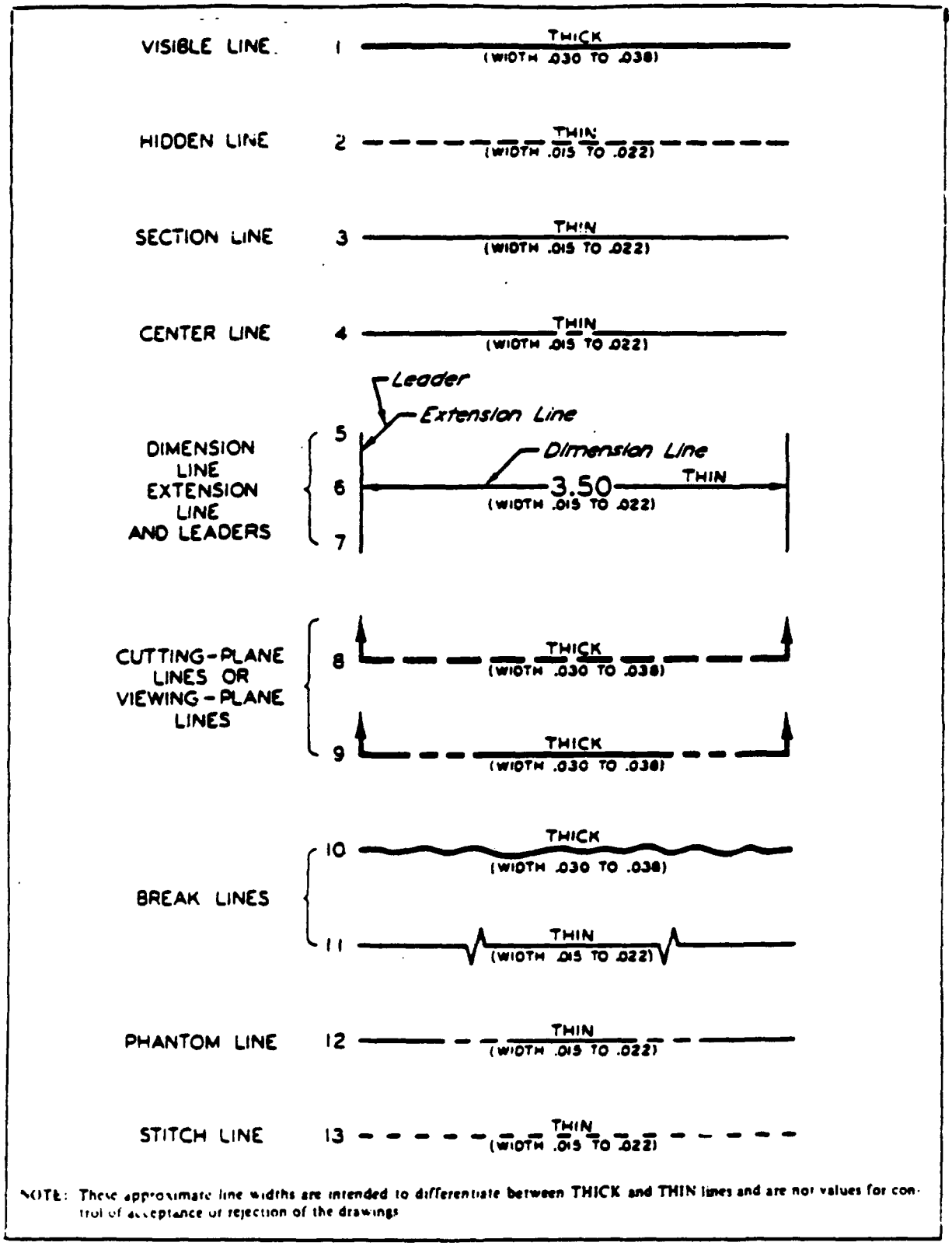


Figure 7. Line Styles and Widths

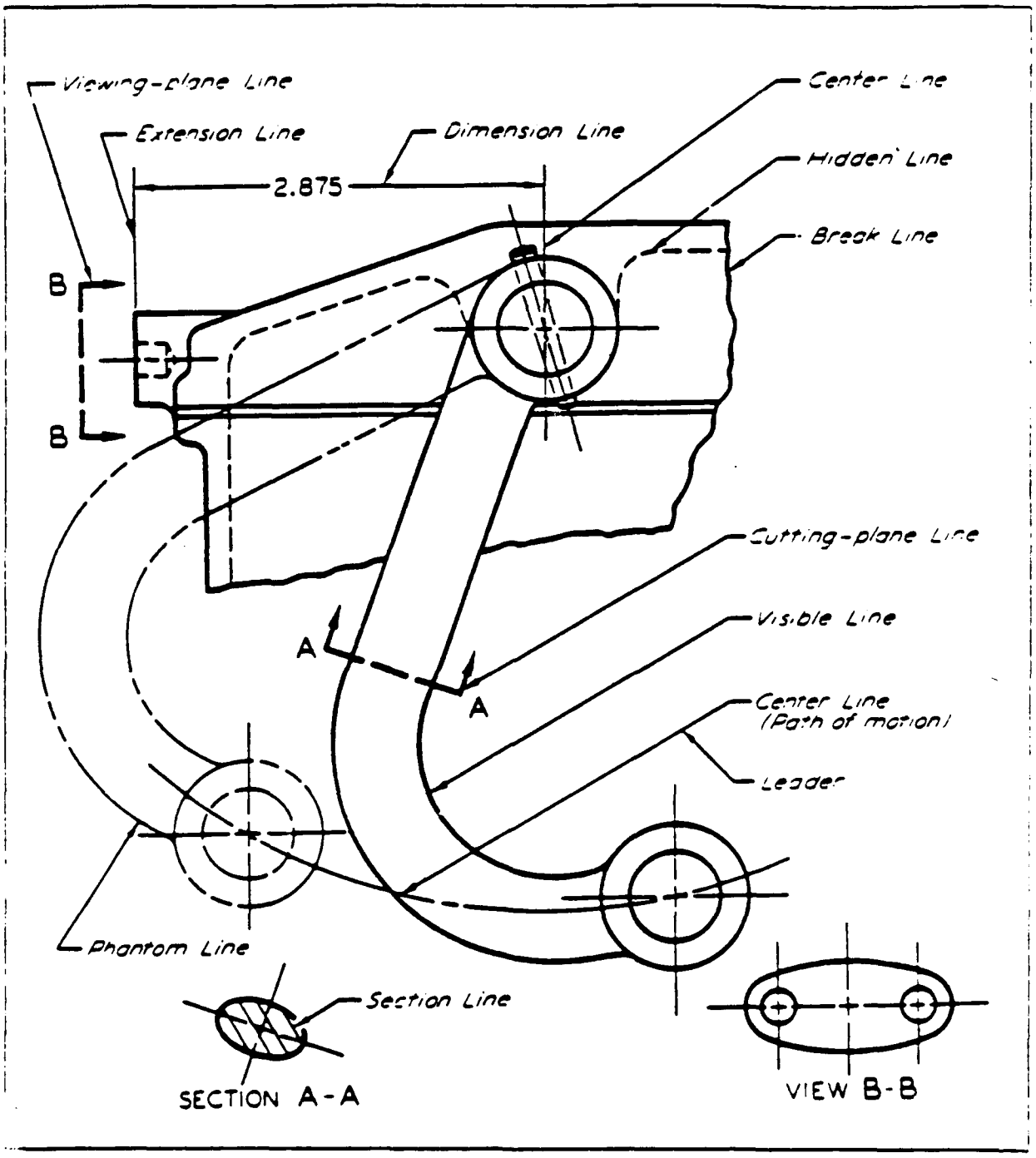


Figure 8. Lines in Engineering Drawings

Center lines - Center lines, Figures 7 and 8 consist of alternating long and short dashes. They are used to represent axes of symmetrical parts and features, bolt circles, and paths of motion. The long dashes of the center lines may vary in lengths, depending upon the size of the drawing. Center lines should start and end with long dashes. They should not intersect at the spaces between dashes or by crossing the long or short dashes. Center lines should extend uniformly and distinctly a short distance beyond the object or feature of the drawing unless a longer extension is required for dimensioning or for some other purpose. They should not terminate at other lines of the drawing nor should they extend through the space between views. Very short center lines may be unbroken if no confusion results with other lines.

Symmetry lines - This is a center lines used as an axis of symmetry for a partial view. The line of symmetry is identified by two thick short parallel lines drawn at right angles to it. They are used in representing partially drawn views and partial sections of symmetrical parts. Symmetrical view visible and hidden lines may extend past the symmetry line if clarity would be improved.

Dimension lines - Dimension lines, Figure 7 and 8 are used to indicate the extent and direction of dimensions and are terminated by neatly-made uniform arrowheads. The length of the arrowhead should be equal to the height of the dimension numerals if possible. If inadequate space is available, the arrowhead may be shown outside the dimension limit.

Extension lines - Extension (witness) lines, Figures 7 and 8, are used to indicate the point or line on the drawing to which the dimension applies. A short gap is left where the extension line would join the object, so as not to confuse extension lines with the lines of the object.

Extension lines are also used to indicate the extension of a surface to a theoretical intersection. When a point is located by extension lines, the extension lines should pass through the point.

Leaders - Leaders, Figures 7 and 8, are used to direct notes, dimensions, symbols, item numbers, or part numbers to features on the drawing. A leader should generally be a single straight inclined line (not vertical or horizontal), except for a short horizontal portion extending to the center of the height of the first or last letter or digit of the note. This horizontal portion is optional and if used it should not underline the note.

The leaders should terminate as follows:

a) Without symbol, if they end on a dimension line .

b) With a dot 1.5 mm or 0.06 inch minimum diameter, if they end within outlines of an object .

c) With an arrowhead, if they end on the outside of an object.

d) With or without a dot or arrowhead on drawings prepared by computer automated techniques.

Leaders should not be curved in any way and should not cross each other unless unavoidable. Two or more leaders to adjacent areas on the drawing should be drawn parallel.

Cutting-plane lines - Cutting-plane and viewing-plane lines, Figures 7 and 8, are used to indicate the location of cutting planes for sectional views and the viewing position for removed partial views. Two forms of cutting-plane and viewing-plane lines are approved for general use.

Break Lines - Two forms of break lines are approved for general use as follows:

a) A freehand thick line. (See Figure 7, line 10.)

b) Long ruled thin dashes joined by zigzags. (See Figure 7, line 11.)

Phantom lines - Phantom lines (Figure 7, line 12) consist of long, thin dashes separated by pairs of short, thin dashes. The long dashes may vary in length, depending on the size of the drawing. Phantom lines are used to indicate alternate positions of moving parts (Figure 8); adjacent positions of related parts; and repeated detail. These lines are also used for features such as bosses and lugs (later moved); for delineating machining stock and blanking developments; for piece parts in jigs and fixtures; and for bend lines on drawings or formed metal parts. Phantom lines should start and end with long dashes which may vary in length, depending on the size of the drawing.

Stitch lines - Stitch lines (Figure 7, lines 13 and 14) consist of short, thin dashes and spaces of equal lengths. The dots are approximately 0.016 (0.35 mm) in diameter and 0.12 inch (3 mm) apart.. Stitch lines are used for indicating a sewing or stitching process.

Chain lines - The chain line (Figure 7, line 15) consist of thick, alternating long and short dashes. This line is used to indicate that a surface or surface zone is to receive additional manufacturing treatment within limits specified on the drawing. (See Figure 8.)

Arrowheads - Arrowheads may be prepared manually or mechanically. The length and width should have a ratio of approximately 3:1. The width of the arrowhead should be proportionate to the thickness of the lines used. Consistency of the style of an arrowhead should be contained throughout the drawing. See Figure 9 for acceptable arrowhead styles.

Figure 9. Arrowhead Styles

In addition to the generic line types defined in ANSI Y14.2M, ANSI and DoD standards in specific areas require the use of additional types of lines. Insufficient time is available at present to fully investigate these and categorize them. To illustrate the general nature of the need for such lines types, Appendix A contains extracts from two Military Standards define other linestyles and specify their meanings.

In summary, many additional linestyles are required to support engineering drawings. There are additional requirements on how patterns start and end within line segments beyond those that computer graphics standards alone can satisfy. Some of these restrictions may need to be stated in the CALS application profile since they go beyond what can be specified through the registration process alone. For example, there are special rendition requirements for hidden lines that cannot be satisfied by a polyline primitive with the appropriate linestyle alone. This happens because the conformance requirements for computer graphics standards do not specify how linestyle patterns must be continued from segment to segment within a polyline, nor do they specify how such a line must end. NBS/ICST will write registration proposals for such items to include the additional conformance requirements, however, they may be rejected by the standards committees.

[Note: A linetype registration proposal has been prepared for each of the above linestypes, with the following exceptions:
- **visible lines can be done with linetype solid;**
- **symmetry lines cannot be done with a single graphical line of any type; a symbol facility could be used, or a set of individual line elements;**
- **single and double arrow linetype can be used for leaders and dimension lines;**
- **cutting plane lines must be done as a set of individual lines since they can't be represented as a single linetype.**

Some restrictions, such as allowable widths of lines and styles of text in engineering drawings, will need to be stated in the application profile for CALS.]

2.3.2 Symbols

The general intent of this section is to illustrate that the requirements for symbols in engineering drawings cannot be met by the polymarker primitives of the graphics standards.

International Standard ISO 3461 defines the general characteristics of symbols used in engineering drawings. This International Standard applies to graphics symbols which may be:

a) placed on equipment or parts of equipment of any kind in order to instruct the persons handling the equipment as to its use and operation:

b) placed on sites and ways where people may assemble or move, giving them instructions, such as prohibitions, warnings, rules or limits, regarding their behavior:

c) used in pictorial reproductions, such as plans, drawings, layouts, guidelines and similar documents.

Definitions

For the purposes of ISO 3461, the following definition applies:

Graphic symbol: A visually perceptible figure produced by means of writing, drawing, printing or other manufacturing techniques. It is used to transmit a message and represents in an understandable manner, independently of any language, an object, concept or state.

Graphics symbols stand for objects, concepts or states. (What a symbol stands for is usually known as the "referent.") This includes abstract references such as conditions, relationships, facts or actions.

Functions

As a rule, graphic symbols are used to:

a) identify (for example to describe a piece of equipment or an abstract concept);

b) qualify (for example to describe a variation or a secondary function);

c) instruct (for example to describe an operation or method of use):

d) command(that something must or must not be done);

e) warn (for example of danger);

f) indicate (for example direction, quantity).

Graphic Form

For each graphic symbol that is required an original design is prepared. An original design is a symbol design drawn and presented in the manner described below, i.e.. drawn out on the basic pattern with due regard to the principles defined below. The basic pattern described below constitutes a frame in which the original design may be inscribed. The lines indicated in the basic pattern (circles, hexagons, octagons, squares, etc.) are intended as an aid to the designer in drawing up the original designs. The form of the graphics symbols should be suitable for economical reproduction by means of commonly applied techniques, such as etching, engraving, printing, and photographic means.

Basic Pattern

The basic pattern (Figure 10) comprises:

1) a basic square of side 50 mm; this measure is equal to the nominal measure, a , of the original

2) a basic circle of 36 mm diameter having approximately the same area as the basic square;

3) a second circle of 50 mm diameter, being the inscribed circle of the basic square (1);

4) a second square of side 40 mm, which touches the basic circle (2) with its corner;

5) a rectangle of approximately the same area as the basic square (1), with the long side (62.5 mm) horizontal and symmetrical with the basic square;

6) a second rectangle having approximately the same area as the basic square (1), with its long side (62.5 mm) vertical and symmetrical with the basic square;

70 a third square formed by the lines passing through the points of intersection of the basic square (1) and the basic circle (2); the sides of this square are oriented at 45 degree to the basic square and the corners of this square define the limits of the horizontal and vertical dimensions of the basic pattern;

8) an irregular octagon formed by lines inclined at 30 degrees to the sides of the square (7);

The basic pattern is laid upon a 75 mm x 75 mm square subdivided by a 12.5 mm square grid which also coincides with the basic square (1).

The original design of a graphics symbol should be fitted into the basic pattern according to the following principles:

1) for a symbol consisting of a single geometrical form, such as a circle or a rectangle, the corresponding geometrical forms of the basic pattern should be used, in which case the lines of the basic pattern should be the center lines of the .1 inch thick lines of the symbol being designed;

2) to achieve the impression of uniform perceived size among symbols, attention should be given in the equalization of surface areas; for example, a circle without external parts should be drawn upon the basic circle (2) (see Figure 11, part C) whereas a circle with external parts should be drawn upon the smallest circle (3) (see Figure 11, part D).

Figure 11 gives several examples of symbols created according to this standard.

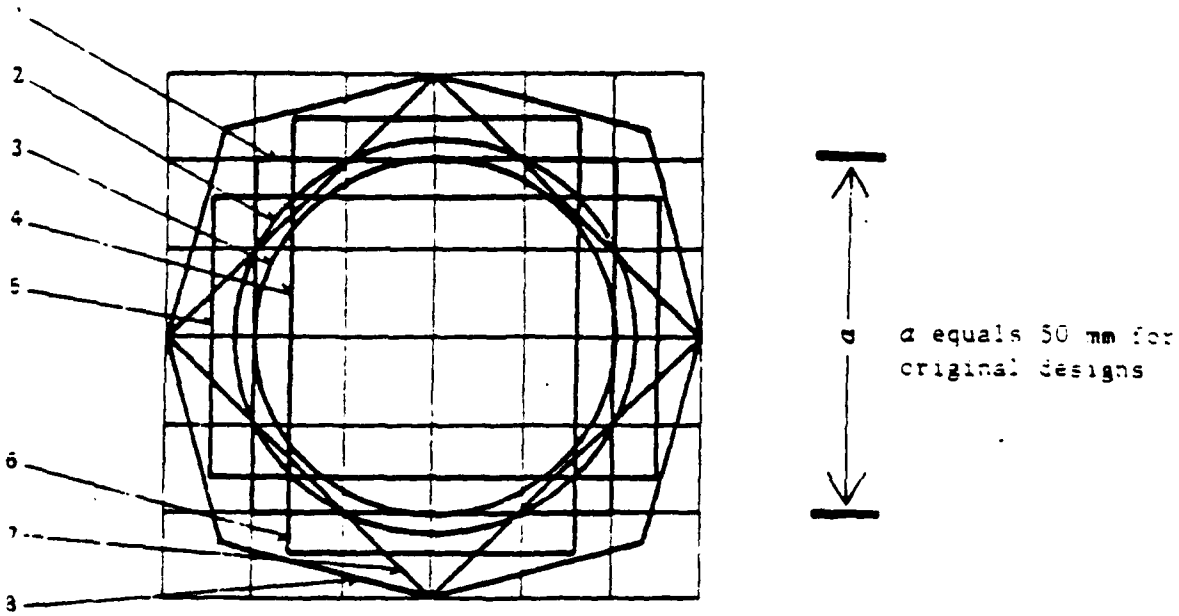


Figure 10. Basic Symbol Pattern

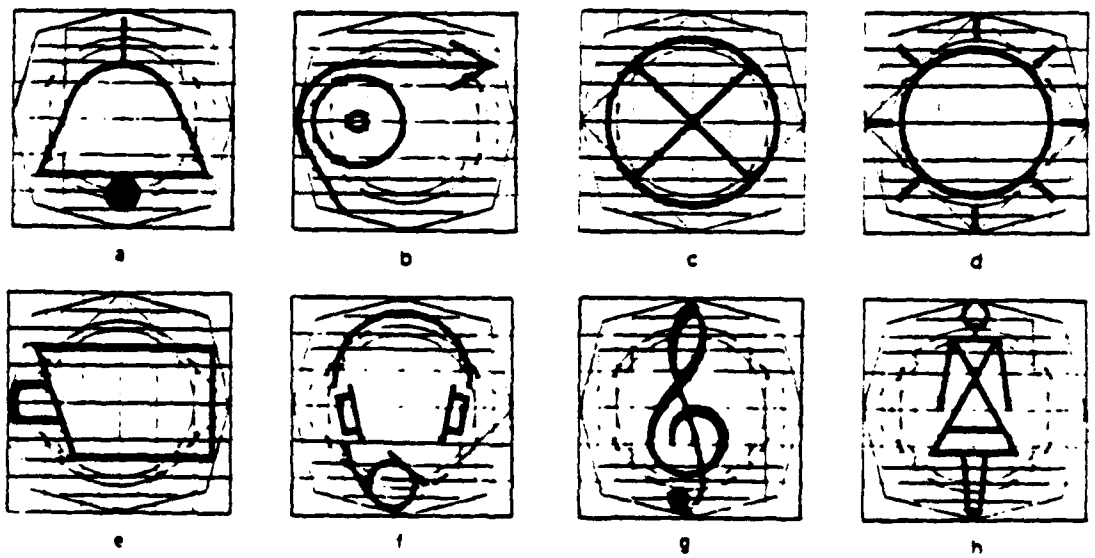


Figure 11. Examples of Symbols

Orientation of the Symbols

The majority of graphic symbols preserve their meaning in any position. However, when the meaning of a graphic symbol does depend on its orientation of position, this shall be explicitly stated. Figure 12 illustrates a graphics symbol not dependent on its position (a television receiver), and a graphic symbol dependent on its position (a "bar").

The statement concerning the position dependency could read as follows:

> "The meaning of the graphic symbol depends on its position. Care shall be taken that it is not reproduced on rotating controls."

Figure 12. Symbol Orientation

Use of Symbols in Engineering Drawings

Figure 13 illustrates the typical use of symbols in engineering drawings. A large number of simple symbols are used repetitively in different locations to construct the drawing. Such drawings should not be transferred by building each symbol from primitives available in the CGM, nor is it feasible to make each required symbol a Generalized Drawing Primitive or marker symbol. It is absolutely essential that any technique used to transfer such drawings allow:

1) a symbol to be defined once in a picture and then instanced repetitively;

2) externally defined symbols from standard libraries to be included by reference.

This must be done for these reasons:

1) to reduce the required communication bandwidth;

2) to reduce the storage required for the picture;

3) to promote standardized appearance of drawings.

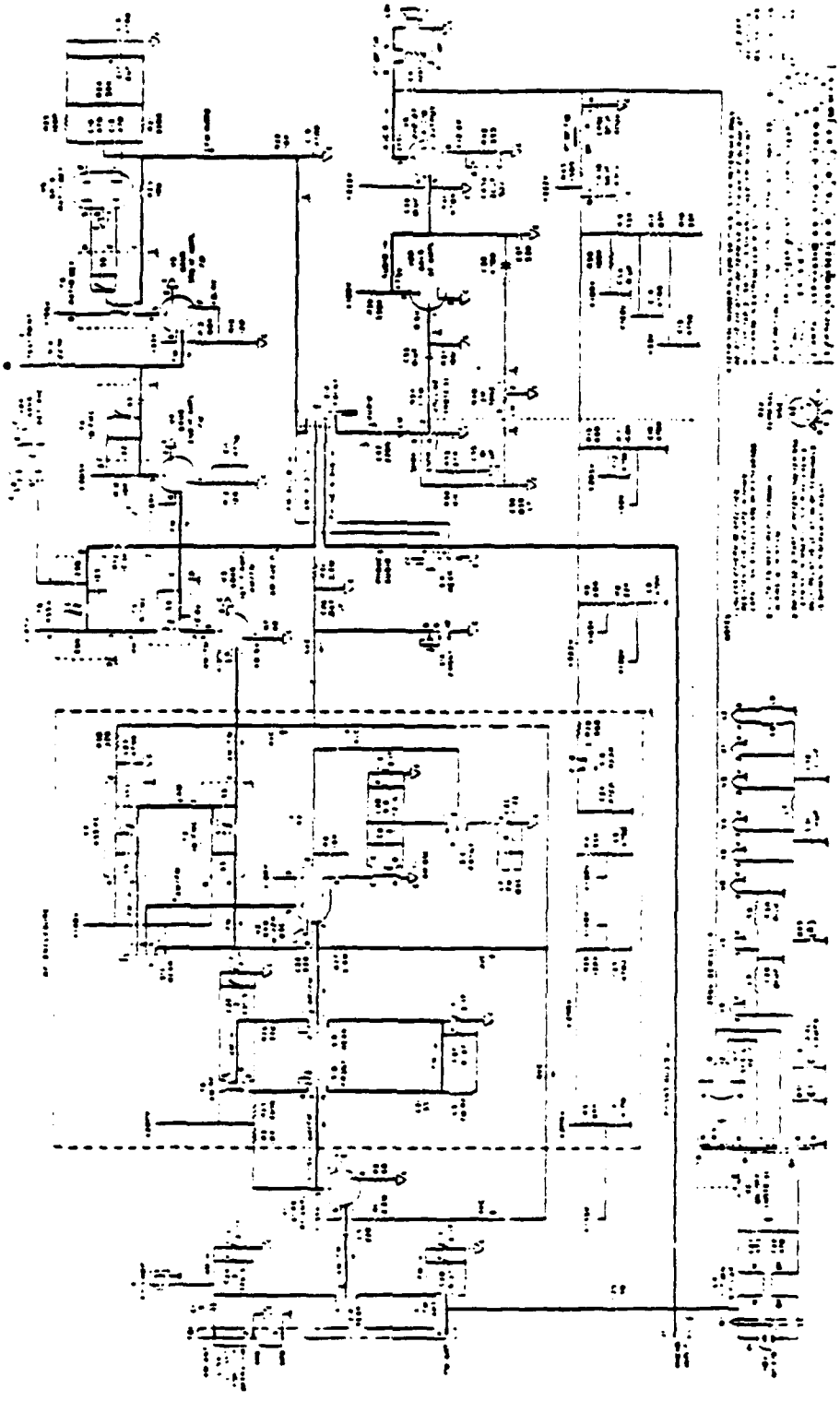


Figure 13. A Typical Engineering Drawing Showing the Use of Symbols

2.3.3 Curves

Insufficient time and funding are available in the present SOW to investigate this area thoroughly. For now, it is assumed that the IGES entities represent a sufficient set.

2.3.4 Hatch Styles

A wide variety of hatch styles are used in engineering drawings for purposes such as representing the nature of materials. Figure 14 illustrates some typical ones. Insufficient time and funding are available in the present contract to investigate this area thoroughly, but based on preliminary observations, the following capabilities appear to be needed:

1) All necessary patterns available as registered hatch styles (for reasons similar to those stated for marker symbols above, drawings containing these hatch patterns cannot be transferred between systems by decomposing the patterns using the individual primitives available in the CGM.)

2) Arbitrary fill areas and clipping regions are needed to represent drawings containing such hatch patterns. These cannot be approximated by breaking them up into simpler areas since pattern continuity cannot be maintained.

[Note: Registration proposals were prepared for all types of section lining (hatch styles) except the following, which can be done with a built-in hatch style:

- electric windings, etc.	**hatch index 6, horizontal/vertical crosshatch.**

Registration proposals were prepared for the following hatch styles to support the more exact rendition requirements of the engineering drawing standard from a similar type in the built-in CGM linestyles. In each case, the drawing standard requires "45 degree lines" while the built in hatch styles guarantee only "positive" and/or "negative" slope.

- cast iron, etc.	**similar to hatch index 3, positive slope equally spaced parallel lines**
- white metal, etc.	**similar to hatch index 6, positive slope/negative slope crosshatch]**

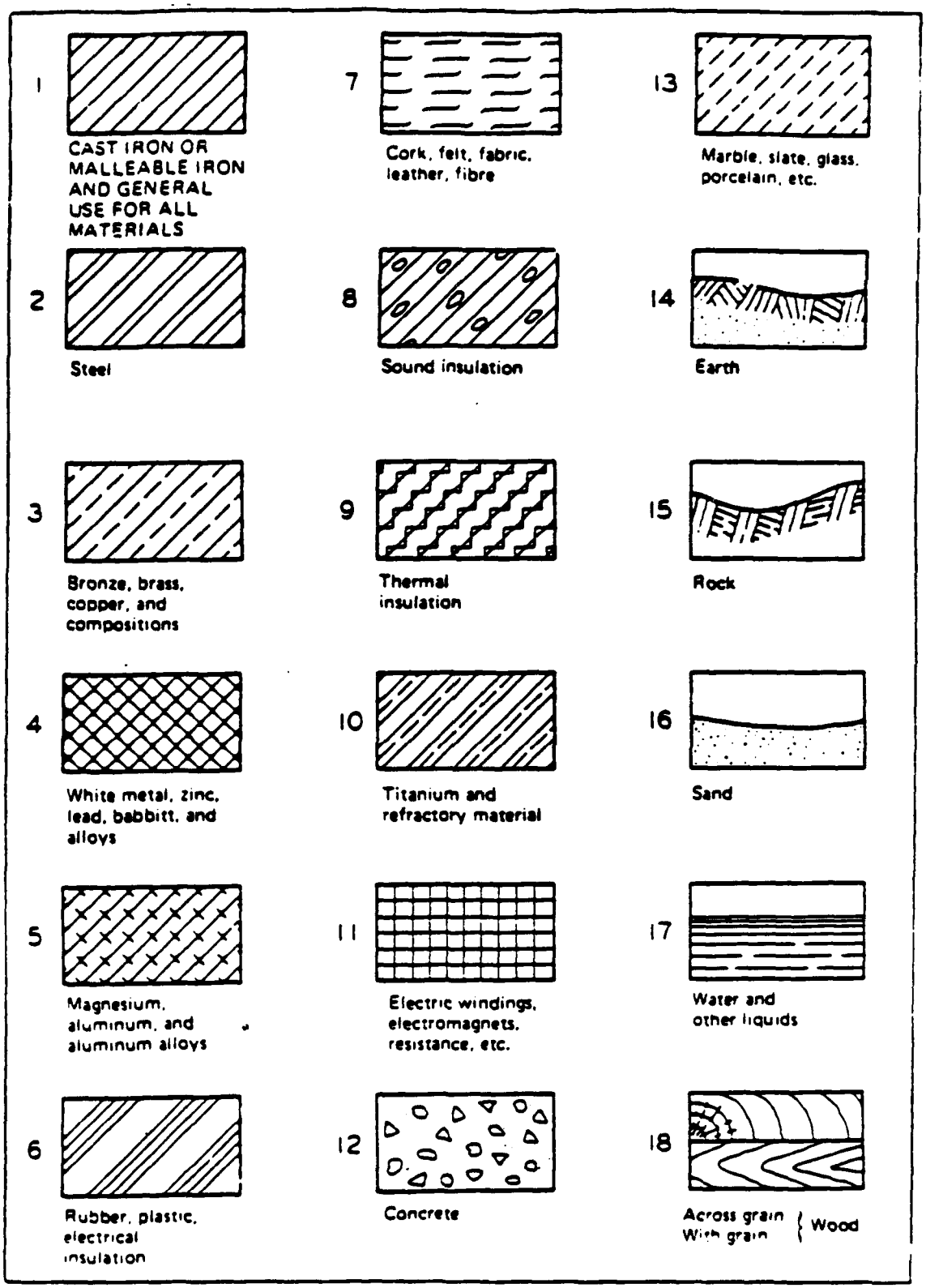


Figure 14. Hatch Styles from Y14.2M

2.3.5 Text

The text capabilities that are actually required in engineering drawings are quite simple, although most drawing systems provide more advanced features. This paragraph will first present the requirements taken from Y14.2M-1979, and then will describe the enhancements that are needed to, for example, render IGES defined data without loss of quality.

2.3.5.1 Y14.2 Lettering Requirements

Single-stroke Gothic Lettering

Lettering on drawings must be legible and suitable for easy and rapid execution. These requirements are met in the recommended single-stroke gothic characters shown in Figures 15 and 16 or adaptations thereof, which improve reproduction legibility. One such adaptation by the National Microfilm Association is the gothic style Microfont alphabet intended for general usage. (See Figure 17.) Opaque and well-spaced lettering is required on the drawing for microfilm reproduction.

Inclined or Vertical Lettering

Either inclined or vertical lettering is permissible. Only one style of lettering should be used throughout a drawing. The preferred slope for the inclined characters is 2 in 5 or approximately 68 degrees with horizontal. (See Figure 16.)

Figure 15. Vertical Lettering

Figure 16. Inclined Lettering

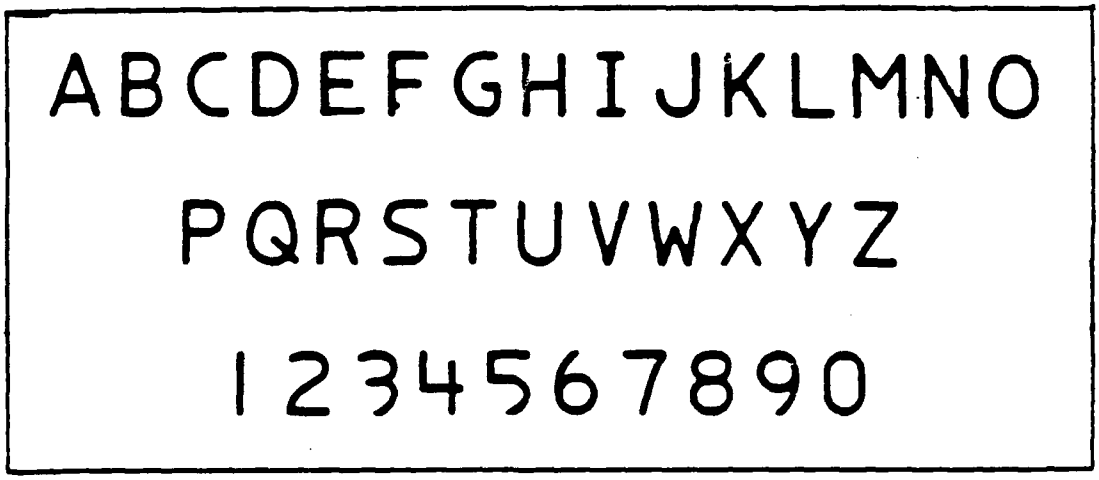


Figure 17. Microform Lettering

Use of Upper-case Letters

Upper-case letters should be used for all lettering on drawings. When additions or revisions are made, the original style of lettering should be maintained.

Lettering for titles, subtitles, drawing numbers, and other uses may be made free-hand, by typewriter, or with the aid of mechanical lettering devices such as templates and lettering machines. Regardless of the method used, all characters are to conform, in general, with the recommended gothic style and must be legible in full or reduced size copy by any accepted method of reproduction.

A type face comparable to light-line Pica Gothic, block numerals is preferred for typing on drawings.

Size and Spacing of Lettering

The recommended minimum freehand and mechanical letter height for various applications are defined in the standards as follows:

Letters in words should be spaced so that the background areas between the letters are approximately equal, and words are to be clearly separated by a space equal to the height of the lettering. For legible reproduction, a space between two letters of at least 0.06 inch (1.5 mm) is to be used whenever possible. The space between two numerals having a decimal point between them is to be a minimum of two thirds the height of the lettering. The vertical space between lines of lettering should be no more than the height of the lettering, and no less than half the height of the lettering used.

Notes should be placed horizontally on drawings and separated vertically by spaces at least equal to double the height of the character size used, to maintain the identity of each note.

The division line of a common fraction should be parallel to the direction in which the dimension reads and should be separated from the numerals by a maximum of 0.06 inch (1.5 mm) spacing. When fractions occur in notes, tables, and lists, the diagonal division line is permissible. Numerals in fractions should be the same size as other numerals.

Spacing between max/min dimensions should be one-half of the character height.

Lettering should not be underlined except when special emphasis is required. The underlining should not be less than 0.06 inch (1.5 mm) below the lettering.

The lettering height, spacing, and proportions in Figure 15, 16, and 17 normally provide acceptable reproduction or camera reduction and blow-back. However, manually, mechanically, opti-mechanically, or electro-mechanically applied lettering (typewriter, etc.) with height, spacing, and proportions less than those recommended are acceptable when the minimum reproducibility and legibility requirements of the accepted industry or military reproduction specifications are met. Therefore, the basic requirements for lettering on a drawing is that fully legible copies may be produced.

2.3.5.2 Text in IGES

The IGES standard defines an entity called **general note** which consists of strings of text that are incorporated in other entities for producing drawing labels and annotation. IGES defines the following fonts for use with this entity:

0. Symbol font (use longer recommended)
1. Standard's block
2. LeRoy
3. Future
4. Fastfont
5. Calcomp
6. Comp 80
7. Microfilm standard
8. ISO standard
9. DIN standard
10. Military standard
11. Gothic
12. New gothic
13. Lightline gothic
14. Simplex Roman
15. Italic

16. APL
17. Century schoolbook
18. Helvetica

1001. Symbol font 1
1002. Symbol font 2

In addition, the IGES text model includes several capabilities that are not present in computer graphics standards such as the CGM. Appendix C contains extracts from the IGES standard that explain these capabilities.

IGES also includes a **text font definition entity** that enables a text font to be described as a sequence of strokes. Appendix C contains extracts from IGES 3.0 that explain this entity. This capability could be built on top of the CGM by stroking out the individual characters as required. A more general solution to compatibility is described in Conclusions Section, where a common font definition mechanism is proposed, based on ISO 9541, for product data standards, graphics standards and publishing standards.

2.3.6 Images

Based upon review of project descriptions, CALS will capture and store a great deal of existing engineering drawing data in raster form. Neither the CGM nor IGES adequately address this area. It is likely that a solution developed for the use of image (raster) data in the publications area will also effectively address the requirements of engineering drawings.

IV. CONCLUSIONS

1.0 Introduction

This section presents the list of extensions that are needed to the CGM. It is anticipated that most of these extensions can be implemented through the registration process. In several areas—as described in Section V below—additional study is necessary to determine the precise nature of the required extensions.

NBS/ICST used the following criteria in defining required extensions:

1) Picture description must be as compact as is practical, consistent with ease of generation and interpretation. (This implies, in particular, that a "symbol" or "macro" capability is needed, as is access to external libraries of symbols.)

2) Only "presentation-related" relationships between objects and attributes of objects need be preserved in the transfer. (This places the transfer at an intermediate level between that provided by IGES and by the (unextended) CGM. When (1) and (2) are considered jointly, they imply that transfer by "approximation with lower-level graphical entities"—such as approximation a curve with a sequence of line segments or a centerline with a set of polylines and/or marker types—is unacceptable. A local system is free however to make such approximations in the process of imaging a picture, consistent with accuracy restrictions.)

3) Extensions must be consistent with the philosophical basis of standards in the areas of Open Systems Interconnection (naming and addressing in particular) and Office Systems (font *architecture in particular.*)

2.0 Lines

A user defined linestyle, similar to that described in Section 1.2.1 of the Discussion section is required.

Linetypes that directly represent the presentation requirements of engineering drawings must be defined. Where the limited capabilities of the built-in polyline primitive—which allows a linetype to consist only of a sequence of line segments and gaps, without precise control over its rendition—are exceeded, GDPs may be needed. Some conformance data may need to be placed in the application profile.

Some of the types needed are:

1) center line,

2) phantom line,

3) break line,

4) lines with arrows on one or both ends.

Extensions to line attributes to include line end styles and joining options.

[Note: A registration proposal for each of those from Y14.2M has been written. Investigation of other standards is needed to identify others.]

3.0 Symbols

As described above, the polymarker primitive is not particularly useful for either publishing or engineering drawing. The registration of additional marker types has little utility. Instead, a general object definition and instantiation capability as described is needed. **[No registration proposals have been completed in this area.]**

4.0 Curves

The following curves are required as GDPs. As necessary, Escapes are required to support their attributes.

1) Bezier curves,

2) B-splines,

3) Conics and conic arcs,

4) Other splines as determined by the study described in .

[Note: A registration proposal for each of these has been written.]

A closed figure primitive is required, together with:

1) Arbitrary clipping region,

2) Arbitrary fill boundary.

[No registration proposals have been completed in this area.]

5.0 Hatch styles

Registered hatch styles to support engineering drawing uses as defined in Section 2.3.4 of the Discusion section are required.

[Note: A registration proposal for each of those from Y14.2M has been written. Investigation of other standards is needed to identify others.]

Registered hatch styles to support technical and administrative publication uses are required. Insufficient data is available to specify these now.

6.0 Text

The text model must be completely replaced through the use of GDPs and Escapes to adopt the model and architecture of ISO DP 9541.

Some specific fonts identified in Section 2.3.5 of the Discussion section for use in engineering drawings must be registered.

[No registration proposals have been completed in this area.]

7.0 Images

A raster "input" primitive to accept input from scanners and files stored on disc (optical or not) is required.

The role of compression in the metafile must be clarified. Although it is best to rely on other standards for necessary compression, it may be necessary to add GDPs to cover more sophisticated compression techniques in the CGM in the short term. The CCITT Group 3 and Group 4 facsimile standards do not provide either color or grey scale capabilities.

Additional raster attributes are required to support image processing.

[No registration proposals have been completed in this area.]

8.0 Naming and External References

The naming and definition of attributes and objects must be extended from the simplistic two-dimensional (positive and negative integer indices separating the space into registered and implementation-dependent types) name space to one consistent with the "resource" view of objects seen in most modern commercial systems and exemplified in the ISO DP 9541 font work. (A limited attempt was made in the CGM by including a list of font names that are then mapped to indices. This was a first step in the right direction.) This can be done fairly easily by defining GDPs and Escapes to replace the existing output primitives and attributes. **It is absolutely essential that this be done to insure the long-term coherence of information processing standards. Neither the CGM nor other computer graphics standards will find acceptance in modern applications without these extensions.**

V. AREAS REQUIRING FURTHER INVESTIGATION

1.0 Curves

More technical work is necessary to explore:

1. Define the actual requirements in the two selected application areas for curves (as opposed to guessing them based on current practice.)

2. Investigate implementation difficulty to determine the cost of implementation.

3. Determine mathematical properties of conversions between curves (e.g. loss of accuracy, loss of desired curvature, etc.)

4) Define a minimal set of curves that:

a) have implementation difficulty appropriate for various classes (price/performance) of systems;

b) can be readily used to approximate other curves.

To illustrate some of the complexities involved, the following material—provided by a member of the committee that developed the IGES standard—shows some of the differences and similarities between two curves: B-splines and Bezier curves.

Differences between B-splines and Bezier curves

1) Bezier curve + Composite curve = B-spline

That is, the classes of functions are in fact the same.

2) Commonalities between Bezier curves and B-splines

- exact conics (rational quadratics), though the algorithms for conic to Bezier (or B-spline) conversions are not in the public domain, to the best of my knowledge.
- nonrational polynomial curves of any degree.
- nice mathematical properties: convex hull, plane intersection, etc.

3) Bezier

- C0 continuity can be explicit (common endpoint).
- Faster evaluation (by divided differences), though not as fast as power basis. Points and deviations.
- Greater storage requirements (deg+1 points per segment, while B-spline can use Const + number of segments if continuity is maximal).
- More stable (lower condition number than power basis or B-spline).
- Easy mathematics: evaluation, degree evaluation, degree evaluation, subdivision.

- Trivial conversion to B-spline.

- Immediate evaluation of points and derivations at parametric start and end.

4) B-spline

- parametric continuity given explicitly in the knot sequence. For exactness, prefer knots with explicit multiplicity to repeated knots.

- Mathematics is quite complex, but powerful: evaluation, adding and removing knots.

- For repeated evaluation or intersection algorithms, will probably want to convert to piecewise Bezier first (this amounts to making all of the interior knots of multiplicity *degree* and the two exterior knots of multiplicity *degree* + 1.)

2.0 Text

Additional time and funding will be required to develop and define the new text GDP and associated escapes (to implement attributes) that can fully implement the model of ISO DP 9541. Extensions to the "CGM environment" compatible with the ISO font description and transfer work must be developed.

3.0 Images

Additional funding will be required to develop and define raster input extensions and raster data types needed to fully support technical and administrative publications. Specifically, the following items are beyond the level of the current task:

1. Definition of standard interfaces to input scanning devices through the CGI/CGM and GKS.

2. Expansion of the attributes of raster data to accommodate the requirements of image processing algorithms. For example, data is needed on the characteristics of input scanner (resolution, frequency response characteristics, etc.) to properly process the data.

3. Families of standard compression algorithms must be developed beyond those currently supported in MIL-STD-1840 (Automated Interchange of Technical Information.) The one algorithm in that standard (CCITT group 4 facsimile) as well as the one compression technique supported in the CGM (a one-dimensional run-length encoding vaguely similar to Group 3 facsimile) are well known to be inadequate for "photographic" content. These techniques are useful for rasterized text and geometric graphics, however these contents would most likely be "compressed" by sending them in a CGM or ODIF format. The CGM can be easily extended with additional raster primitive/sencodings, thereby completely removing the requirement for the inclusion of Group 4 facsimile in MIL-STD-1840.

4.0 Specification of Data Record Contents

Escapes, GDPs, and Application data all contain information in data records. The standards do not dictate how data in such records must be formulated or encoded. It is desirable that a standard method be developed for all of these data records and promulgated throughout the standards community with the intention that all registration proposals use this same standard method. At present, we are specifying a "clear text" encoding only for all data records. This is clearly inadequate to support binary coded transfer of complex drawings due to compactness considerations.

5.0 Support for Named Items and "Symbol Libraries"

Extensions must be developed to replace the simplistic "indexed" definition of attributes in the graphics standards with named definitions based on a resource model consistent with that of the ISO Font standard. Techniques must be developed to allow picture components (symbols, macros) to be defined and instantiated in pictures.

6.0 Definition of Registration Requirements and Development of Registration Proposals

Additional work will be required to complete the definition of registration requirements and the development of registration proposals in these areas:

1) Linetypes,

2) Hatchstyles,

3) Text fonts.

This is due to these factors:

1) The large number of registration proposals to be developed.

2) The long time-frame involved in sponsoring registration proposals through the approval process and preparing necessary revisions and responding to comments from standards committees.

3) The need for a review of proposed items by the CALS community.

VI. RECOMMENDATIONS

1.0 Registration Proposals List

Recommendations for this task are the registration proposals themselves. The following list contains the categories and the names of the registration proposals developed under this CALS SOW task. They have been submitted to ANSI for formal processing through ISO. Section 2.0 below contains the actual registration proposals that have been developed and submitted for this fiscal year, and are in the order as listed below.

1) Linetypes
   - break line - style 1
   - break line - style 2
   - center line
   - chain line
   - double arrow
   - hidden line
   - phantom line
   - single arrow
   - single dot
   - stitch line
   - user specified dash pattern

2) Hatchstyles
   - across grain wood
   - bronze, brass, copper, and compositions
   - cast iron or mallable iron and general use for all materials
   - concrete
   - cork, felt, fabric, leather, and fiber
   - earth
   - magnesium, aluminum, and aluminum alloys
   - marble, slate, glass, procelain, etc.
   - rock
   - rubber, plastic, and electrical insulation
   - sand
   - sound installation
   - steel
   - thermal insulation
   - titanium and refractory material
   - water and other liquids
   - white metal, zinc, lead, babbitt, and alloys
   - with grain wood

3) Generalized drawing primitives
   - Bezier curve
   - conic arc
   - parametric spline curve
   - rational B-spiine curve

4) Escape functions
   - set conic arc transformation matrix
   - set dash
   - set line cap
   - set line join
   - set miter limit

2.0 Prepared Registration Proposals

The following registration proposals are exactly as submitted to ANSI for formal registration. They have all been submitted and are currently in the formal process.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 10 April 1987
sponsoring authority ANSI

Class of Graphical Item: LINETYPE

Name: break line - style 1

Description: A break line linetype -style 1- consists of either one of two allowable representations as specified in ANSI Y14.2M-1979 (Line Conventions and Lettering.) This is simply a line having a "freehand" appearance.

This linetype is intended for use in engineering drawings.

Additional Comments: The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype.

Justification for Inclusion in the Register:

This linetype is commonly used in engineering drawings. It is one of a set of linetypes to be registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings.

Relationship to Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: LINETYPE

Name: break line - style 2

Description: A break line linestyle consists of either one of two allowable representations as specified in ANSI Y14.2M-1979 (Line Conventions and Lettering.) This is a line consisting of long dashes joined by zigzags. Such lines have the following visual appearance:

This linetype is intended for use in engineering drawings.

Additional Comments: The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype.

Justification for Inclusion in the Register:

This linetype is commonly used in engineering drawings. It is one of a set of linetypes to be registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings.

Relationship to Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: LINETYPE

Name: center line

Description: A center line linetype consists of alternating long and short dashes as specified in ANSI Y14.2M-1979 (Line Conventions and Lettering.) Such a line has the following visual appearance:

This linetype is intended for use in engineering drawings. The long dashes may vary in length depending on the size of the drawing. Lines drawn in this linetype shall start and end with long dashes. A very short line may be unbroken.

Additional Comments: The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype. In some cases, it is necessary to exercise precise control over the manner in which two center lines intersect in a drawing. In these cases, it is appropriate to simulate this linetype by sequences of correctly placed individual line segments.

Justification for Inclusion in the Register:

This linetype is commonly used in engineering drawings. It is one of a set of linetypes to be registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings.

Relationship to Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 10 April 1987
sponsoring authority ANSI

Class of Graphical Item: LINETYPE

Name: chain line

Description: A chain line linetype consists of alternating long and short dashes as specified in ANSI Y14.2M-1979 (Line Conventions and Lettering.) Such a line has the following visual appearance:

This linetype is intended for use in engineering drawings. Its rendition is generally different from that of the dashed-dotted linestyle already present in the graphics standards.

Additional Comments: The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype. In some cases, it is necessary to exercise precise control over the manner in which two lines intersect in a drawing. In these cases it may be appropriate to simulate this linetype by using sequences of correctly placed individual line segments.

Justification for Inclusion in the Register:

This linetype is commonly used in engineering drawings. It is one of a set of linetypes to be registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings.

Relationship to Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 10 April 1987
sponsoring authority ANSI

Class of Graphical Item: LINETYPE

Name: double arrow

Description: A double arrow linetype consists of a solid line terminated by two arrowheads as specified in ANSI Y14.2M-1979 (Line Conventions and Lettering) requirements for dimension lines. The arrows are rendered so that the arrow tip occurs at the first and last points in the defining set. Such a line has the following visual appearance:

This linetype is intended for use in engineering drawings.

Additional Comments: The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype.

Justification for Inclusion in the Register:

This linetype is commonly used in engineering drawings. It is one of a set of linetypes to be registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings.

Relationship to Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: LINETYPE

Name: hidden line

Description: A hidden line linetype consists of short evenly spaced dashes as specified in ANSI Y14.2M-1979 (Line Conventions and Lettering.) Such a line has the following visual appearance:

— — — — — — — —

This linetype is intended for use in engineering drawings. The dashes may vary in length depending on the size of the drawing. Lines drawn in this linetype shall start and end with a dash. Dashes shall join at corners, and arcs drawn with this style shall start and end with dashes. These rendition requirements are different from the dashed linetype that is already defined in the graphics standards.

Additional Comments: The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype. In some cases, it is necessary to exercise precise control over the manner in which two lines intersect in a drawing. In these cases it may be appropriate to simulate this linetype by using sequences of correctly placed individual line segments.

Justification for Inclusion in the Register:

This linetype is commonly used in engineering drawings. It is one of a set of linetypes to be registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings.

Relationship to Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 10 April 1987
sponsoring authority ANSI

Class of Graphical Item: LINETYPE

Name: phantom line

Description: A phantom line linetype consists of long dashes separated by pairs of short dashes as specified in ANSI Y14.2M-1979 (Line Conventions and Lettering.) Such a line has the following visual appearance:

This linetype is intended for use in engineering drawings. The long dashes may vary in length depending on the size of the drawing. Lines drawn in this linetype shall start and end with long dashes which may vary in length depending on the size of the drawing.

Additional Comments: The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype. In some cases, it is necessary to exercise precise control over the manner in which two lines intersect in a drawing. In these cases it may be appropriate to simulate this linetype by using sequences of correctly placed individual line segments.

Justification for Inclusion in the Register:

This linetype is commonly used in engineering drawings. It is one of a set of linetypes to be registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings.

Relationship to Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 10 April 1987
sponsoring authority ANSI

Class of Graphical Item: LINETYPE

Name: single arrow

Description: A single arrow linetype consists of a solid line terminated by an arrowhead as specified in ANSI Y14.2M-1979 (Line Conventions and Lettering) requirements for dimension and leader lines. The arrow is rendered so that the arrow tip occurs at the last point in the defining set. Such a line has the following visual appearance:

This linetype is intended for use in engineering drawings.

Additional Comments: The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype.

Justification for Inclusion in the Register:

This linetype is commonly used in engineering drawings. It is one of a set of linetypes to be registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings.

Relationship to Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 10 April 1987
sponsoring authority ANSI

Class of Graphical Item: LINETYPE

Name: single dot

Description: A single dot linetype consists of a solid line terminated by a dot as specified in ANSI Y14.2M-1979 (Line Conventions and Lettering) requirements for leader lines. The dot is rendered so that the dot occurs at the last point in the defining set. Such a line has the following visual appearance:

This linetype is intended for use in engineering drawings.

Additional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype.

Justification for Inclusion in the Register:

This linetype is commonly used in engineering drawings. It is one of a set of linetypes registered use with computer graphics standards to enable compact storage and transfer of engineering drawings.

Relationship to Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 10 April 1987
sponsoring authority ANSI

Class of Graphical Item: LINETYPE

Name: stitch line

Description: A stitch line linetype consists of dashes and spaces of equal length as specified in ANSI Y14.2M-1979 (Line Conventions and Lettering.) Such a line has the following visual appearance:

This linetype is intended for use in engineering drawings. Its definition contains rendition requirements beyond those for the dashed linetype already present in the graphics standards.

Additional Comments: The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype. In some cases, it is necessary to exercise precise control over the manner in which two lines intersect in a drawing. In these cases it may be appropriate to simulate this linetype by using sequences of correctly placed individual line segments.

Justification for Inclusion in the Register:

This linetype is commonly used in engineering drawings. It is one of a set of linetypes to be registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings.

Relationship to Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation 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 linetype. 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- for the CGM that defines the current user-specified linetype. It is intended that these proposals be processed together.

Justification for Inclusion in the Register:

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 Particular 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.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: HATCHSTYLE

Name: across grain wood

Description:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of across grain wood in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:

Additional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

Justification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

Relationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

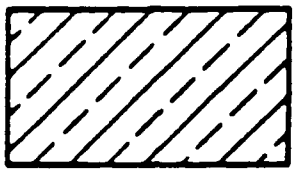
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: HATCHSTYLE

Name: bronze, brass, copper, and compositions

Description:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of bronze, brass, copper, and compositions in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



Additional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

Justification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchtypes registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized

Relationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

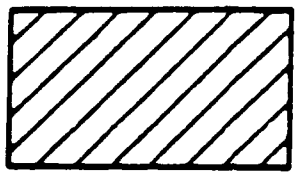
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: HATCHSTYLE

Name: cast iron or malleable iron and general use for all materials

Description:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of cast iron or malleable iron and general use for all materials in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



Additional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype. These requirements are different from those for CGM linetype 3, which requires only positive slope lines rather than 45 degree lines.

Justification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

Relationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

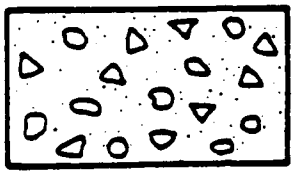
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: HATCHSTYLE

Name: concrete

Description:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of concrete sections in engineering drawings The intended visual representation of a filled-area element hatched in this style is illustrated below:



Additional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

Justification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

Relationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

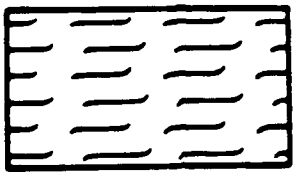
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: HATCHSTYLE

Name: cork, felt, fabric, leather, and fibre

Description:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of cork, felt, fabric, leather, and fibre in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



Additional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed when rendering this linetype.

Justification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

Relationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

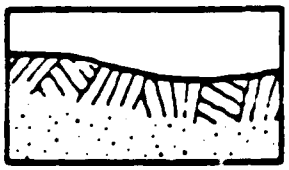
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: HATCHSTYLE

Name: earth

Description:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of earth sections in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



Additional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

Justification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

Relationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

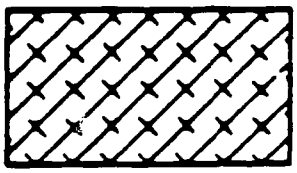
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

lass of Graphical Item: HATCHSTYLE

ame: magnesium, aluminum, and aluminum alloys

escription:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of magnesium, aluminum, and aluminum alloys in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



dditional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

ustification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

elationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

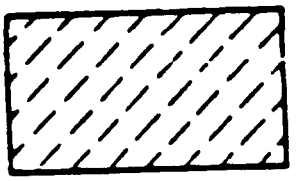
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

lass of Graphical Item: HATCHSTYLE

ame: marble, slate, glass, porcelain, etc.

escription:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of marble, slate, glass, porcelain, etc. in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



dditional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

ustification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

elationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

ıss of Graphical Item: HATCHSTYLE

ıe: rock

ıcription:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of rock sections in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:

ditional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

tification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

lationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

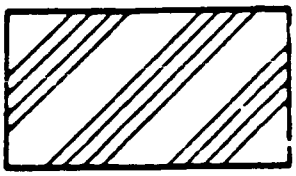
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

ass of Graphical Item: **HATCHSTYLE**

me: rubber, plastic, and electrical insulation

scription:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of rubber, plastic, and electrical insulation in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



ditional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

itification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

lationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

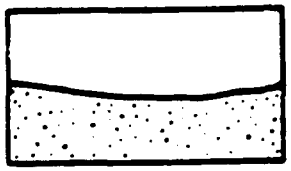
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

ss of Graphical Item: HATCHSTYLE

ne: sand

cription:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of sand sections in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



litional Comments:

The requirements stated in ANSI Y14.2M-1979shall be followed in rendering this linetype.

ification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

ationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

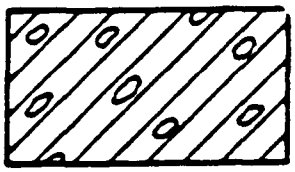
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

is of Graphical Item: HATCHSTYLE

e: sound insulation

ription:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of sound insulation in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



itional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed inrendering this linetype.

ification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

itionship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

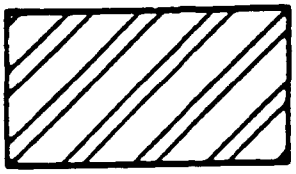
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: HATCHSTYLE

Name: steel

Description:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of steel sections in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



Additional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

Justification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

Relationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

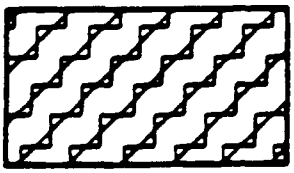
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: HATCHSTYLE

Name: thermal insulation

Description:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of thermal insulation in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



Additional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

Justification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

Relationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

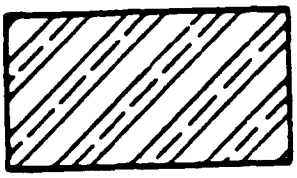
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: HATCHSTYLE

Name: titanium and refractory material

Description:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of titanium and refractory material in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



Additional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

Justification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

Relationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

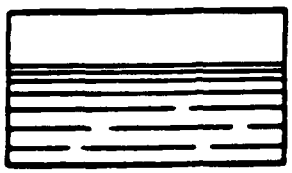
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

ass of Graphical Item: HATCHSTYLE

me: water and other liquids

scription:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of water and other liquids in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



ditional Comments:

The requirements stated in ANSI Y14.2M-1979 take precedence over those in this proposal in case of a conflict.

stification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

lationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

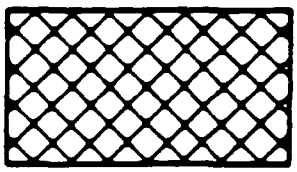
PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

:lass of Graphical Item: HATCHSTYLE

ame: white metal, zinc, lead, babbitt, and alloys

escription:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of white metal, zinc, lead, babbitt, and alloys in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:



dditional Comments:

The requirements stated in ANSI Y14.2M-1979 Shall be followed in rendering this linetype. These requirements are different from those for CGM linetype 6, which requires only positive and negative slope lines rather than 45 degree lines.

ustification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered for use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized

Relationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

lass of Graphical Item: HATCHSTYLE

ame: with grain wood

escription:

A hatchstyle conforming to the requirements of ANSI Y14.2M-1979 (Line Conventions and Lettering) for the representation of with grain wood in engineering drawings. The intended visual representation of a filled-area element hatched in this style is illustrated below:

dditional Comments:

The requirements stated in ANSI Y14.2M-1979 shall be followed in rendering this linetype.

ustification for Inclusion in the Register:

This hatchstyle is commonly used in engineering drawings. It is one of a set of hatchstyles registered use with computer graphics standards to enable compact storage and transfer of engineering drawings. The need for a compact representation of the attributes of filled areas in engineering drawings is widely recognized.

elationship to Particular Standards:

1) ISO 7942 (GKS) - Specifies a registered hatch style as defined in 5.4.1.

2) ISO 8632 (CGM) - Specifies a registered hatch style as defined in 5.7.24.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 10 April 1987
sponsoring authority ANSI

ass of Graphical Item: GDP

P Identifier: Bezier curve

ıcription:

A Bezier cubic section is generated 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 more details.

ditional Comments:

The Bezier curve capabilities proposed here are adapted from those in the PostScript language developed by Adobe Systems Incorporated.

tification for Inclusion in the Register:

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

ationship to Particular 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[1] (GKS Language Bindings) - Specifies a registered GDP. (see attached sheets).

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

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

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:

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

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

The cubic section produced by **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:

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

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

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

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 - as a clear text encoding (machine independent) of a FORTRAN-style packed data record. This is treated as a string type in each encoding and is encoded according to the rules for string in that encoding.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 10 April 1987
sponsoring authority ANSI

Class of Graphical Item: GDP

GDP Identifier: conic arc

Description:

A bounded connected portion of a parent conic curve is generated 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 more details.

Additional Comments: None

Justification for Inclusion in the Register:

Conic arcs are needed to support the requirements of office document exchange, publishing, and engineering drawing exchange. They are commonly found in proprietary graphics systems. The conic arc capabilities proposed here are adopted from the ANSI Y14.26 (IGES Version 3.0) specification.

Relationship to Particular 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.

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

The **conic arc** is a realization of the Conic Arc Entity of the IGES 3.0 standard. The attached extracts from the IGES Version 3.0 standard provide the functional specification.

A functional description of the conic arc parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

point list(nP) - contains the two start and terminate points
data record (D): - see the IGES attachments for definitions

A
B
C
D
E
F

Note: The ZT value is not included since it must be zero.

Items for Data Record:

The following values are in the same order as in the IGES standard.

Integer IL	0
Integer RL	6
Real RA(1)	A
Real RA(2)	B
Real RA(3)	C
Real RA(4)	D
Real RA(5)	E
Real RA(6)	F
Integer SL	0

....

Data Record Description:

The parameters are as defined in the attached extract from the IGES standard.

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

All encodings will be handled in the same way - as a clear text encoding (machine independent) of a FORTRAN-style packed data record. This is treated as a string type in each encoding and is encoded according to the rules for string in that encoding.

3.4 Conic Arc Entity

A conic arc is a bounded connected portion of a parent conic curve which consists of more than one point. The parent conic curve is either an ellipse, a parabola, or a hyperbola. The definition space coordinate system is always chosen so that the conic arc lies in a plane either coincident with or parallel to the XT, YT plane. Within such a plane, a conic is defined by the six coefficients in the following equation.

$$A*XT^2 + B*XT*YT + C*YT^2 + D*XT + E*YT + F = 0$$

3.4.1 Each coefficient is a real number. The definitions of ellipse, parabola, and hyperbola in terms of these six coefficients are given below.

3.4.2 A conic arc determines unique arc endpoints. A conic arc is defined within definition space by the six coefficients above and the two endpoints. By considering the conic arc endpoints to be enumerated and listed in an ordered manner, start point followed by terminate point, a direction with respect to definition space can be associated with the arc. In order for the desired elliptical arc to be distinguished from its complementary elliptical arc, the direction of the desired elliptical arc must be counterclockwise. In the case of a parabola or hyperbola, the parameters given in the parameter data section uniquely define a portion of the parabola or a portion of a branch of the hyperbola; therefore, the concept of a counterclockwise direction is not applied. (Refer to Section 3.1.2 for information concerning use of the term "counterclockwise".)

3.4.3 The direction of the conic arc with respect to model space is determined by the original direction of the arc within definition space, in conjunction with the action of the transformation matrix on the arc.

3.4.4 The definitions of the terms ellipse, parabola, and hyperbola are given in terms of the quantities Q1, Q2, and Q3. These quantities are:

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

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

$$Q3 = A + C$$

3.4.5 A parent conic curve is

An ellipse if $Q2 > 0$ and $Q1 * Q3 < 0$.

A hyperbola if $Q2 < 0$ and $Q1 \neq 0$.

A parabola if $Q2 = 0$ and $Q1 \neq 0$.

An example of each type of conic arc is shown in Figure 3-3.

3.4.6 Those entities which can be represented as various degenerate forms of a conic equation (Point and Line) must not be put into the Entity Type 104; more appropriate Entity Types exist for these forms.

Because of the numerical sensitivity of the implicit form of the conic description, a receiving system not using that form as its internal representation for conics need not be expected to correctly process conics in this form unless they are put into a standard position in definition space. A conic arc entity is said to be in a standard position in definition space provided each of its axes is parallel to either the XT axis or YT axis and provided it is centered about the ZT axis. For a parabola, use the vertex as the origin. The conic is moved from this position in definition space to the desired position in space with a transformation matrix (Entity type 124).

The form number is regarded as purely informational by such a postprocessor.

Further details may be found in Appendix E.

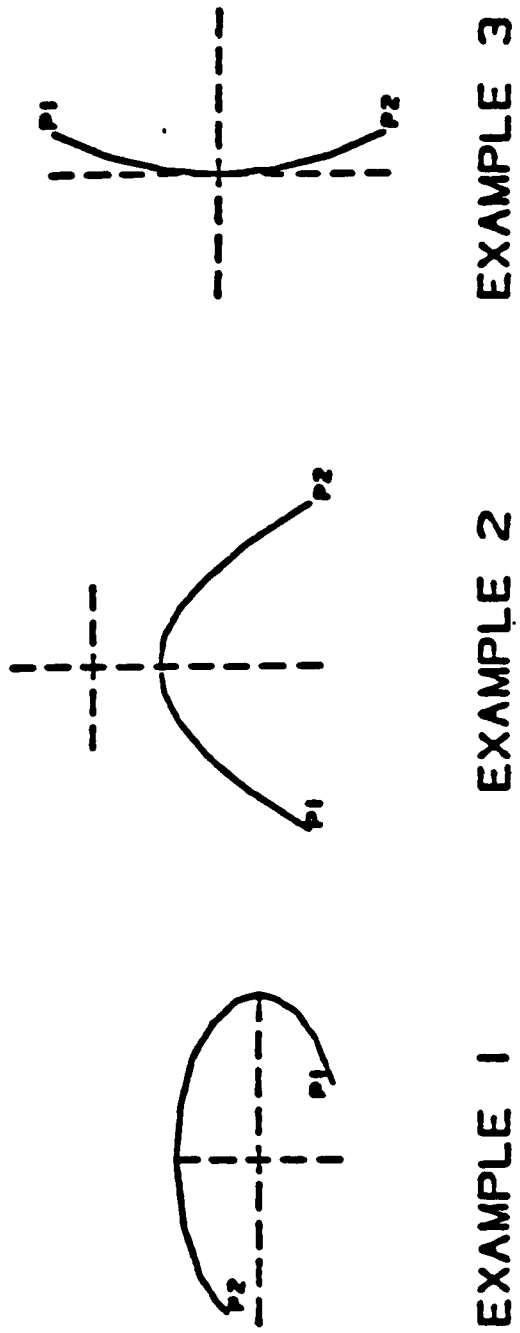


FIG. 3-3 EXAMPLES OF THE CONIC ARC ENTITY

3.4.7 In the event that a parameterization is required but not given, the default parameterization is:

Parabola

case A and E $\neq$ 0.0

if X1 < X2
C(t) = (t, -(A/E)*t**2, ZT) for t1 $\leq$ t $\leq$ t2
where, for i = 1 and 2, ti = Xi.

if X2 < X1
C(t) = (-t, -(A/E)*t**2, ZT) for t1 $\leq$ t $\leq$ t2
where, for i = 1 and 2, ti = -Xi.

case C and D $\neq$ 0.0

if Y1 < Y2
C(t) = (-(C/D)*t**((2, t, ZT) for t1 $\leq$ t $\leq$ t2
where, for i = 1 and 2, ti = Yi.

if Y2 < Y1
C(t) = (-(C/D)*t**2, -t, ZT) for t1 $\leq$ t $\leq$ t2
where, for i = 1 and 2, ti = -Yi.

Ellipse

C(t) = (a*cos t, b*sin t, ZT)

for t1 $\leq$ t $\leq$ t2

where
a = sqrt(-F/A)
b = sqrt(-F/C)

and, for i = 1 and 2, ti is such that
(i) (a*cos ti, b*sin ti, ZT) = (Xi, Yi, ZT)
(ii) 0 $\leq$ t1 $\leq$ 2*PI
(iii) 0 $\leq$ t2 - t1 $\leq$ 2*PI

Hyperbola

case F*A < 0.0 and F*C > 0.0

let
a = sqrt(-F/A)
b = sqrt(F/C)
and, for i = 1,2
ti is such that
(i) (a*sec ti, b*tan ti, ZT) = (Xi,Yi,ZT)
(ii) - PI/2 < t1,t2 < PI/2

if $t1 < t2$

$C(t) = (a*\sec t, b*\tan t, ZT)$ for $1 \leq t \leq t2$

if $t2 < t1$

$C(t) = (a*\sec(-t), b*\tan(-t), ZT)$ for $-t1 \leq t \leq -t2$

case $F*A > 0.0$ and $F*C < 0.0$

let

$a = \mathrm{sqrt}(F/A)$
$b = \mathrm{sqrt}(-F/C)$
and, for $i = 1,2$
ti is such that
(i) $(a*\tan ti, b*\sec ti, ZT) = (Xi, Yi, ZT)$
(ii) $-PI/2 < t1, t2 < PI/2$

if $t1 < t2$

$C(t) = (a*\tan t, b*\sec t, ZT)$ for $t1 \leq t \leq t2$

if $t2 < t1$

$C(t) = (a*\tan(-t), b*\sec(-t), ZT)$ for $-t1 \leq t \leq -t2$

3.4.8 Field 15 of the directory entry accommodates a Form Number. For this entity, the options are as follows:

FORM	Meaning
0	Form of parent conic curve must be determined from the general equation.
1	Parent conic curve is an ellipse (See example 1, Figure 3-3).
2	Parent conic curve is a hyperbola (See example 2, Figure 3-3).
3	Parent conic curve is a parabola (See example 3, Figure 3-3).

3.4.9 Directory Data

ENTITY TYPE NUMBER : 104

3.4.10 Parameter Data

Index	Name	Type	Description
1	A	Real	Conic Coefficient
2	B	Real	Conic Coefficient
3	C	Real	Conic Coefficient
4	D	Real	Conic Coefficient
5	E	Real	Conic Coefficient
6	F	Real	Conic Coefficient
7	ZT	Real	ZT Coordinate of plane of definition
8	X1	Real	Start Point Abscissa
9	Y1	Real	Start Point Ordinate
10	X2	Real	Terminate Point Abscissa
11	Y2	Real	Terminate Point Ordinate

Additional Pointers as required (see 2.2.4.4.2).

APPENDIX E
CONIC ARCS

Conic arcs as specified by the IGES standard are extremely sensitive to the data in two distinct ways:

(a) Accuracy

It is numerically sensitive; small changes in the coefficients can cause large changes in the locations of the points satisfying the conic equation.

(b) Stability

The determination of the conic type depends upon whether certain invariants are positive, zero or negative. Working in floating point arithmetic, a machine value of 0.0 is unlikely to be encountered. Furthermore, small changes in coefficient values can easily result in positive values when negative ones are intended and conversely.

It is assumed that data is put into a conic arc entity with the intent of preserving the geometric properties of the data (major and minor semi-axes, asymptotes, directrices, etc.) in addition to describing the points on the curve.

If the geometric properties are desired, the 104 entity should be used as described below.

This method primarily addresses the stability problem, though the accuracy of the conic should improve because the range of coefficient values will decrease. While the geometric properties are not explicitly defined in this representation, they can be obtained from it in a direct and arithmetically stable manner.

If both the sending and intended receiving system are known to use the A-F form of the 104 entity (Conic Arc) in their own databases the preprocessor may put the data into in the unchanged form. This minimizes the loss of information caused by truncation and roundoff errors as no changes are made to the data. The stability problem is presumably not of concern in this case.

Here is one suggested set of values:

(1) Ellipse

A :=AXISY^2	B := 0
C :=AXISX^2	D := 0
E := 0	F := -A*C

where AXISY and AXISX are the lengths of the major and minor semi-axes (not necessarily in order).

(2) Hyperbola

A := - AXISY^2 (or + AXISY^2)	B := 0
C := + AXISX ^{2} (or - AXISX^2, if A 0)	D := 0
E := 0	F := -A*C.

where AXISY and AXISX are the lengths of the major and minor semi-axes (not necessarily in order).

(3) Parabola

A := 0	(or 1)	B := 0
C := 1	(or 0, if A = 1)	D := 4*DIST (or 0, if A = 1)
E := 0	(or 4*DIST, if A = 1)	F := 0

where DIST is the distance of the vertex from the focus.

Preprocessor Conic Handling

The conic arc must be put into standard form, parallel to the X and/or Y axis(axes) and centered about the origin. An 124 transformation matrix must be used to move the conic arc into its desired position in space. In this form the coefficients in the format that should be 0.0 will be exactly so. In particular, for the ellipse and hyperbola B, D, and E must be 0.0, and for the parabola B and F and either A and E or C and D must be 0.0.

Determination of the conic type from the equations becomes straight forward for the postprocessor.

For further mathematical details, see (THOM60).

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 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 generated. The intended realization of this output primitive is equivalent to that intended for the Parametric Spline Curve Entity of ANSI Y14.26 (IGES Version 3.0), with the restriction that the "Z polynomial" of the IGES standard be zero. See the attached sheets for more details.

Additional Comments: None

Justification for Inclusion in the Register:

Parametric spline curves are needed to support the requirements of engineering drawing exchange. They are commonly found in proprietary graphics systems.

Relationship to Particular 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.

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

The **parametric spline curve** is a realization of the Parametric Spline Curve Entity of the IGES 3.0 standard, restricted to the two-dimensional environment of the CGM. The attached extracts from the IGES Version 3.0 standard provide the functional specification.

A functional description of the parametric spline curve parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

point list(nP) - contains the "T" values
data record (D): - see the IGES attachments for definitions
CTYPE
H
N
AX
AY
...
TPX
TPY

Items for Data Record:

If VDC TYPE integer was selected (Warning: parametric splines should not be expected to work well in this case):

The following values are in the same order as in the IGES standard with the Z values ommitted.

Integer IL	3 + 9N
Integer IA(1)	CTYPE
Integer IA(2)	H
Integer IA(3)	N
Integer IA(5)	T(1)
...	
Integer IA(5+N)	T(N+1)
Integer IA(5+N+1)	AX(1)
.....	
Integer RL	0
Integer SL	0

3.8 Parametric Spline Curve Entity

(Consult Appendix D for additional mathematical details)

The parametric spline curve is a sequence of parametric polynomial segments. The CTYPE value in Parameter 1 indicates the type of curve as it was represented in the sending (pre-processing) system before conversion to this entity.

3.8.1 The N polynomial segments are delimited by the breakpoints T(1), T(2), ...,T(N+1). The coordinates of the points in the i-th segment of the curve are given by the following cubic polynomials (the coefficients D, or C and D will be zero if the polynomials are of degrees 2 or 1, respectively):

$$X(u)=AX(i)+BX(i)\,s+CX(i)\,s^2+DX(i)\,s^3$$
$$Y(u)=AY(i)+BY(i)\,s+CY(i)\,s^2+DY(i)\,s^3$$
$$Z(u)=AZ(i)+BZ(i)\,s+CZ(i)\,s^2+DZ(i)\,s^3$$

where

$$T(i)\leq u\leq T(i+1),\ i=1,\ldots,N$$
$$s=u-T(i)$$

In order to avoid degeneracy, for each i at least one of the nine real coefficients, BX(i), CX(i), DX(i), BY(i), CY(i), DY(i), BZ(i), CZ(i), and DZ(i) must be non-zero.

3.8.2 If the spline is planar, it must be parametrized in terms of the X and Y polynomials only. The Z polynomial will then be zero except for each i, the AZ(i) term which indicates the Z-depth in definition space.

3.8.3 The parameter H is used as an indicator of the smoothness of the curve. If H=0, the curve is continuous at all breakpoints. If H=1, the curve is continuous and has slope continuity (see section 6.3 of FAUX79) at all breakpoints. If H=2, the curve is continuous and has both slope and curvature continuity at all breakpoints (see section 6.3 of Faux79).

If VDC TYPE real was selected:

```
Integer IL          3
Integer IA(1)       CTYPE
Integer IA(2)       H
Integer IA(3)       N
Integer RL          10N+1
Real RA(1)          T(1)
...
Real RA(N+1)        T(N+1)
Real RA(N+2)        AX(1)
Real RA(N+3)        BX(1)
Integer SL          0
....
```

Data Record Description:

The parameters are as defined in the attached extract from the IGES standard.

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

All encodings will be handled in the same way - as a clear text encoding (machine independent) of a FORTRAN-style packed data record. This is treated as a string type in each encoding and is encoded according to the rules for string in that encoding.

3.8.4 To enable determination of the terminate point and derivatives without computing the polynomials, the Nth polynomials and their derivatives are evaluated at u = T(N+1). These data are divided by appropriate factorials and stored following the polynomial coefficients. For example, the name TPY3 will be used to designate 1/3! times the third derivative of the Y polynomial for the Nth segment evaluated at u=T(N+1), the parameter value corresponding to the terminate point. Note that these data are redundant as they are derived from the data defining the Nth polynomial segment.

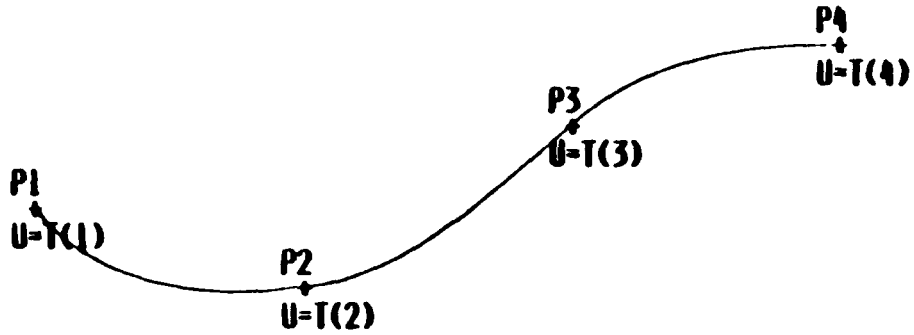
3.8.5 An example of a parametric spline is shown in Figure 3-7. Additional examples are shown in Figure 3-8.

3.8.6 Directory Data
ENTITY TYPE NUMBER : 112

3.8.7 Parameter Data

Index	Name	Type	Description
1	CTYPE	Integer	Spline Type (1=Linear 2=Quadratic 3=Cubic 4=Wilson-Fowler 5=Modified Wilson-Fowler 6=B Spline)
2	H	Integer	Degree of continuity with respect to arc length
3	NDIM	Integer	2=planar 3=non-planar
4	N	Integer	Number of segments
5	T(1)	Real	Break points of piecewise polynomial
.	.		
.	.		
.	.		
5+N	T(N+1)		

CURVE = (X(U), Y(U), Z(U)), FOR $T(1) \leq U \leq T(N+1)$
N = 3 SEGMENTS



P1 = (AX(1), AY(1), AZ(1))
P2 = (AX(2), AY(2), AZ(2))
P3 = (AX(3), AY(3), AZ(3))
P4 = TP0 = (TPX0, TPY0, TPZ0)
FIRST DERIVATIVE AT P4 = TP1 = (TPX1, TPY1, TPZ1)

FOR SEGMENT NUMBER 2:

$$X(U) = AX(2) + BX(2)*(U-T(2)) + CX(2)*(U-T(2))^2 + DX(2)*(U-T(2))^3$$
$$Y(U) = AY(2) + BY(2)*(U-T(2)) + CY(2)*(U-T(2))^2 + DY(2)*(U-T(2))^3$$
$$Z(U) = AZ(2) + BZ(2)*(U-T(2)) + CZ(2)*(U-T(2))^2 + DZ(2)*(U-T(2))^3$$

FIGURE 3-7 EXAMPLE OF THE PARAMETRIC SPLINE CURVE ENTITY

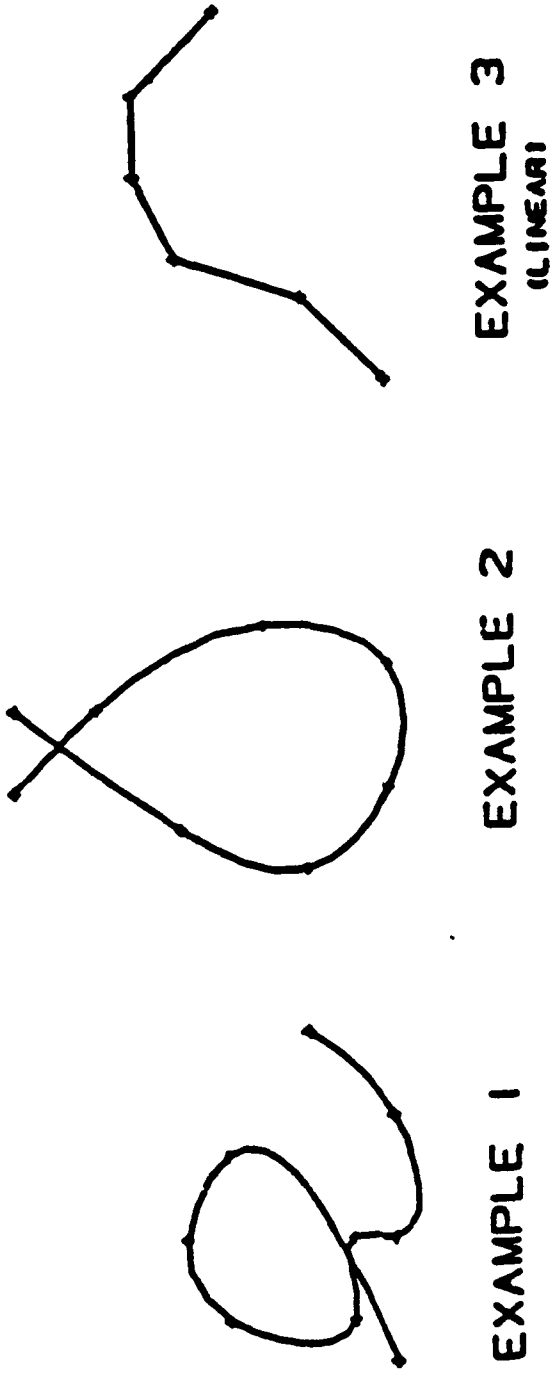


FIG. 3-8 EXAMPLES OF PARAMETRIC SPLINE CURVE ENTITY

Index	Name	Type	Description
6+N	AX(1)	Real	X coordinate polynomial
7+N	BX(1)		
8+N	CX(1)		
9+N	DX(1)		
10+N	AY(1)		Y coordinate
11+N	BY(1)		polynomial
12+N	CY(1)		
13+N	DY(1)		
14+N	AZ(1)		Z coordinate
15+N	BZ(1)		polynomial
16+N	CZ(1)		
17+N	DZ(1)		
	.		Subsequent X, Y, Z polynomials concluding with the twelve coefficients of the Nth polynomial segment.
	.		
	.		

(The parameters that follow comprise the evaluations of the polynomials of the Nth segment and their derivatives at the parameter value u=T(N+1) corresponding to the terminate point. Subsequently these evaluations are divided by appropriate factorials.)

6+13*N	TPX0	Real	X value
	TPX1		X first derivative
	TPX2		X second derivative/2!
	TPX3		X third derivative/3!
	TPY0		Y value
	TPY1		
	TPY2		
	TPY3		
	TPZ0		Z value
	TPZ1		
	TPZ2		
	TPZ3		

Additional Pointers as required (see 2.2.4.4.2)

Software to convert between parametric spline curves or surfaces and the corresponding rational B-spline curves or surfaces is available from the IGES office at the National Bureau of Standards. Materials provided include a magnetic tape of Pascal source code, a listing of the code, and accompanying documentation.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM
date of presentation of proposal 10 April 1987
sponsoring authority ANSI

Class of Graphical Item: GDP

GDP Identifier: rational B-spline curve

Description:

A planar (two dimensional) rational B-spline curve is drawn. The intended realization of this output primitive is equivalent to that intended for the Rational B-Spline Curve Entity of ANSI Y14.26 (IGES Version 3.0), with the restriction that the "Z polynomial" of the IGES standard be zero. See the attached sheets for more details.

Additional Comments: None

Justification for Inclusion in the Register:

Rational B-spline curves are needed to support the requirements of engineering drawing exchange. They are commonly found in proprietary graphics systems.

Relationship to Particular 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[1] (GKS Language Bindings) - Specifies a registered GDP. (see attached sheets).

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

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

The **rational B-spline curve** is a realization of the rational B-spline curve Entity of the IGES 3.0 standard. The attached extracts from the IGES Version 3.0 standard provide the functional specification.

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): - see the IGES attachments for definitions

K
M
PROP2
PROP3
PROP4
T
W
NORM

Note: The PROP1 value is not included since it must be 1.

Items for Data Record:

The following values are in the same order as in the IGES standard.

Integer IL	5
Integer IA(1)	K
Integer IA(2)	M
Integer IA(3)	PROP2
Integer IA(4)	PROP3
Integer IA(5)	PROP4
Integer RL	see IGES extract
Real RA(1)	T(-M)
....	
Real RA(1+A)	W(0)
Real RA(2+A+K)	XNORM
Real RA(3+A+K)	YNORM
Integer SL	0
....	

Data Record Description:

The parameters are as defined in the attached extract from the IGES standard.

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

All encodings will be handled in the same way - as a clear text encoding (machine independent) of a FORTRAN-style packed data record. This is treated as a string type in each encoding and is encoded according to the rules for string in that encoding.

3.16 Rational B-Spline Curve Entity

The rational B-spline curve may represent analytic curves of general interest. This information is important to both the sending and receiving systems. The directory entry form number parameter is provided to communicate this information. It should be emphasized that use of this curve form should be restricted to communications between systems operating directly on rational B-spline curves and not used as a replacement for the analytic forms for communication. For a brief description of a rational B-spline curves, see Section 4 of Appendix D.

If the rational B-spline curve represents a preferred curve type, the form number corresponds to the most preferred type. The preference order is from 1 through 5 followed by 0. For example, if the curve is a circle or circular arc, the form number is set to 2. If the curve is an ellipse with unequal major and minor axis lengths, the form number is set to 3. If the curve is not one of the preferred types, the form number is set to 0.

If the curve lies entirely within a unique plane, the planar flag (PROP1) is set to 1, otherwise it is set to 0. If it is set to 1, the plane normal (parameters 16+A+4K through 16+A+4K) contain a unit vector normal to the plane containing the curve. These fields exist but are ignored if the curve is non-planar.

If the beginning and ending points on the curve are identical, PROP2 is set to 1. If they are not equal, PROP2 is set to 0.

If the curve is rational (does not have all weights equal), PROP3 is set to 0. If all weights are equal to each other, the curve is polynomial and PROP3 is set to 1. The curve is polynomial since in this case all weights cancel and the denominator sums to one (see Appendix D4).

If the curve is periodic with respect to its parametric variable, set PROP4 to 1, otherwise set PROP4 to 0.

3.16.1 Directory Data

ENTITY TYPE NUMBER: 126

Form	Meaning
0	Form of curve must be determined from the rational B-spline parameters.
1	Line
2	Circular arc
3	Elliptical arc
4	Parabolic arc
5	Hyperbolic arc

3.16.2 Parameter Data

Index	Name	Type	Description
1	K	Integer	Upper index of sum. See Appendix D
2	M	Integer	Degree of basis functions
3	PROP1	Integer	=0 - non-planar =1 - planar
4	PROP2	Integer	=0 - open curve =1 - closed curve
5	PROP3	Integer	=0 - rational =1 - polynomial
6	PROP4	Integer	=0 - non-periodic =1 - periodic

Let N=K-M+1 and let A=N+2M

7	T(-M)	Real	Knot Sequence
.	.		
.	.		
.	.		
7+A	T(N+M)		
8+A	W(0)	Real	Weights
.	.		
.	.		
.	.		
8+A+K	W(K)		
9+A+K	XO	Real	Control Points
10+A+K	YO		
11+A+K	ZO		
.	.		
.	.		
.	.		
9+A+4K	XK		
10+A+4K	YK		
11+A+4K	ZK		
12+A+4K	V(0)	Real	Starting parameter value
13+A+4K	V(1)	Real	Ending parameter value
14+A+4K	XNORM	Real	Unit Normal (if curve is planar)
15+A+4K	YNORM		
16+A+4K	ZNORM		

Additional Pointers as required (see 2.2.4.4.2).

Software to convert between parametric spline curves or surfaces and the corresponding rational B-spline curves or surfaces is available from the IGES office at the National Bureau of Standards. Materials provided include a magnetic tape of Pascal source code, a listing of the code, and accompanying documentation.

D4 RATIONAL B-SPLINE CURVES

The comments in this section pertain primarily to section 3.16.

A rational B-spline curve is expressed parametrically in the form

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

where the notation is interpreted as follows.

The W(i) are the <u>weights</u> (non-zero real numbers).

The P(i) are the <u>control points</u> (points in R^3).

The b_i are the B-spline basis functions. These are defined as soon as their degree, M, and underlying knot sequence, T, are specified.

This is done as follows:

Let N = K - M + 1. Then, the knot sequence consists of the non-decreasing set of real numbers; T(-M), ..., T(0), ..., T(N), ..., T(N+M)

The curve itself is parametrized for $V(0) \leq t \leq V(1)$ where $T(0) \leq V(0) < V(1) \leq T(N)$.

The B-spline basis functions b_i are each non-negative piecewise polynomials of degree M. The function b_i is supported by the interval $[T(i-M), T(i+1)]$. Between any two adjacent knot values T(j), T(j+1) the function can be expressed as a single polynomial of degree M.

For any parameter value t between T(0) and T(N) the basis functions satisfy the identity

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

If the weights are all positive, the curve G(t) is contained within the convex hull of its control points.

There are a number of ways to precisely define the B-spline basis functions. A recursive approach proceeds as follows.

Let $N(t \mid t_{i-m}, \ldots, t_{i+1})$ denote the B-spline basis function of degree m supported by the interval $[t_{i-m}, t_{i+1}]$.

With this notation, the degree 0 functions are simply characteristic functions of a half-open interval.

$$N(t \mid a,b) = \begin{cases} 1 & \text{if } a \leq t < b \\ 0 & \text{otherwise} \end{cases}$$

The degree k functions are defined in terms of those of degree k-1.

$$N(t \mid s_0, \ldots, s_k) = \frac{(t-s_0)N(t \mid s_0, \ldots, s_{k-1})}{s_{k-1} - s_0} + \frac{(s_k-t)N(t \mid s_1, \ldots, s_k)}{s_k - s_1}$$

Since some of the denominators will be 0 in the case of multiple knots, the convention 0/0 = 0 is adopted in the above definition.

Rational Bezier curves (and surfaces) can be expressed exactly as rational B-spline curves (and surfaces). (BLOM82).

Software to convert between parametric spline curves or surfaces and the corresponding rational B-spline curves or surfaces is available from the IGES office at the National Bureau of Standards. Materials provided include a magnetic tape of Pascal source code, a listing of the code, and accompanying documentation.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	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 the ANSI Y14.26 (IGES version 3.0) specification. See attached sheets for additional details.

Additional Comments:

None

Justification for Inclusion in the Register:

Conic arcs are needed to support the requirements of engineering drawing exchange. They are commonly found in proprietary graphics systems. 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.

Relationship to Particular Standards:

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

2) See attached sheets.

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

Set Conic Arc Transformation Matrix is a realization of the Transformation Matrix Entity of the IGES 3.0 standard, restricted to the two-dimensional environment of the CGM. This matrix is a component of the graphics state and determines how subsequent conic arc output primitives are transformed from definition space to virtual device coordinates. The attached extracts from the IGES Version 3.0 standard provide the functional specification.

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): - see the IGES attachments for definitions

R_{11}
R_{12}
R_{21}
R_{22}

Items for Data Record:

If VDC TYPE integer was selected (Warning: conic arcs should not be expected to work well in this case):

Integer IL	6
Integer IA(1)	R_{11}
Integer IA(2)	R_{12}
Integer IA(3)	R_{21}
Integer IA(4)	R_{22}
Integer IA(5)	T_1
Integer IA(6)	T_2
Integer RL	0
Integer SL	0

If VDC TYPE real was selected:

Integer IL	0
Integer RL	6
Real RA(1)	R_{11}
Real RA(2)	R_{12}
Real RA(3)	R_{21}
Real RA(4)	R_{22}
Real RA(5)	T_1
Real RA(6)	T_2
Integer SL	0

Data Record Description:

The parameters are as defined in the attached extract from the ANSI Y14.26 (IGES version 3.0) standard.

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

All encodings will be handled in the same way - as a clear text encoding (machine independent) of a FORTRAN-style packed data record. This is treated as a string type in each encoding and is encoded according to the rules for string in that encoding.

3.14 Transformation Matrix Entity

The Transformation Matrix entity transforms three-row column vectors by means of a matrix multiplication and then a vector addition. The notation for this transformation is

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} XINPUT \\ YINPUT \\ ZINPUT \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} XOUTPUT \\ YOUTPUT \\ ZOUTPUT \end{bmatrix}$$

Here, col [XINPUT, YINPUT, ZINPUT] (i.e., the column vector) is the vector being transformed, and col [XOUTPUT, YOUTPUT, ZOUTPUT] is the column vector resulting from this transformation. $R = [R_{ij}]$ is a 3 row by 3 column matrix of real numbers, and T=col [T1,T2,T3] is a three-row column vector of real numbers. Thus, 12 real numbers are required for a Transformation Matrix entity. This entity can be considered to be an "operator" entity in that it starts with the input vector, operates on it as described above, and produces the output vector.

3.14.1 Frequently, the input vector lists the coordinates of some point in one coordinate system, and the output vector lists the coordinates of that same point in a second coordinate system. The matrix R and the translation vector T then express a general relationship between the two coordinate systems. By considerating special input vectors such as col [1,0,0], col [0,1,0], and col [0,0,1] and computing the corresponding output results, a geometric appreciation of the spatial relationship between the two coordinate system can be gained.

For example, for

$$R = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad T = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

the spatial relationship of the input and output cordinate systems is the following:

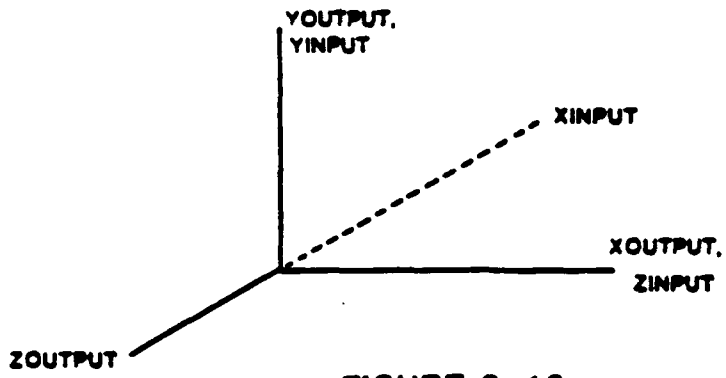


FIGURE 3-16

All coordinate systems are assumed to be orthogonal, cartesian, and right-handed unless specifically noted otherwise.

3.14.2 Following are three specific areas where the Transformation Matrix entity is used to transform coordinates between coordinate systems. Each example area illustrates a specific choice of input and output coordinate system. Other choices of coordinate systems may be appropriate in other application areas.

The usual situation for this type of use of the Transformation Matrix entity is when the input vector refers to the definition space coordinate system for a certain entity, and the output vector refers to the model space coordinate system. (See Sect. 3.1.1) In this case, the matrix R is referred to as the defining matrix, and the Transformation Matrix entity defining R and T is pointed to in field seven (transformation matrix field) of the directory entry of the entity. (See Sect. 2.2.4.3.7) In this use of the Transformation Matrix entity, the matrix R is subject to the restrictions given in Form 0 and Form 1 below.

A second situation is the case when the input vector refers to the model space coordinate system and the output vector refers to a viewing coordinate system. In this case, the matrix R is referred to as a view matrix, and is subject to the restrictions given in Form 0 below. Note that when a planar entity is viewed at true length (i.e., the viewing plane is parallel to the plane containing the entity) then the rotation matrix pointed to by DE Field 7 of the planar entity will be the inverse (=matrix transpose) of the matrix pointed to by DE Field 7 of the View entity. (See Sect. 4.3.11)

A third situation involves finite element modeling applications. Here, it may be the case that an input coordinate system is related to an output coordinate system by a particular R and T, and, in turn, the output coordinate system is then taken as an input coordinate system for a second R and T combination, and so on. These coordinate systems are frequently called local coordinate systems. Model space is frequently called the reference system. For example, the location of a finite element node may be given in one local coordinate system, which may serve as the input coordinate system for a second local coordinate system, which in turn serves as the input coordinate system for the model space coordinate system which is the reference system. Allowable forms of the matrix R for these applications are detailed in Forms 10, 11, and 12 below.

3.14.3 Whenever coordinate systems are related successively to each other as described above, a basic result is that the combined effect of the individual coordinate system changes can be expressed in terms of a single matrix R and a single translation vector T. For example, if the coordinate system change involving the matrix R2 and the translation vector T2 is to be applied following the coordinate system change involving the matrix R1 and the translation vector T1, then the matrix R and the translation vector T expressing the combined changes are R=(R2) (R1) and T = (R2) (T1) + T2.

Here, (R2) (R1) denotes matrix multiplication of 3x3 matrices, where multiplication order is important. The matrix R and the translation vector T are computed similarly whenever more than two coordinate system changes are to be applied successively.

Successive coordinate system changes are specified by allowing a Transformation Matrix entity to reference another Transformation Matrix entity through Field 7 of the Directory Entry. In the example above, the Transformation Matrix entity containing R1 and T1 would contain in its Directory Entry Field 7 a pointer to the Transformation Matrix entity containing R2 and T2. The general rule is that Transformation Matrix entities applied earlier in a succession will reference Transformation Matrix entities applied later. Note that the matrix product (R2) (R1) in the example above does not appear explicitly in the data, but, if needed, must be computed according to the usual rules of matrix multiplication.

A second example of coordinate systems being related successively (or "concatenated", or "stacked"), in addition to the finite element example mentioned above, involves one manner of locating into model space a conic arc that is in standard position in definition space. In this case, R1 and T1 move the conic arc from its standard position to an arbitrary location in any plane in definition space satisfying ZT=constant. (Therefore, $R1_{33}=1.0$, $R1_{31}=R1_{32}=R1_{13}=R1_{23}=0.0$. T1 can be an arbitrary translation vector.) R2 and T2 then position the relocated conic arc into model space. (R2 can be an arbitrary defining matrix and T2 can be an arbitrary translation vector.) Note that for R1 and T1, both the input vector and the output vector refer to the same coordinate system, namely, the definition space for the conic arc.

3.14.4 A 3x3 matrix R is called orthogonal provided its transpose, R^t, about the main diagonal yields a matrix inverse for R. The columns of an orthogonal matrix considered as vectors form an orthogonal collection of unit vectors. As $(R^t)^t=R$, the transpose of an orthogonal matrix is again an orthogonal matrix. The determinant of an orthogonal matrix is equal to either plus one or minus one. In the event R is an orthogonal matrix with determinant equal to positive one, R can be expressed as a rotation about an axis passing through the origin. In this event, R is referred to as a rotation matrix. In the event R is an orthogonal matrix with determinant equal to negative one, R can be expressed as a rotation about an axis passing through the origin followed by a reflection about a plane passing through the origin perpendicular to the axis of rotation.

3.14.5 Allowable Form Numbers. The defining matrix of an entity must use either Form zero or Form one. A defining matrix associated with a view entity must use Form zero.

Form 0: (default) R is an orthogonal matrix with determinant equal to positive one. T is arbitrary. The columns of R taken in order form a right-handed triple in the output coordinate system.

Form 1: R is an orthogonal matrix with determinant equal to negative one. T is arbitrary. The columns of R taken in order form a left-handed triple in the output coordinate system.

3.14.6 Forms 10, 11, 12. These form numbers indicate special matrices used in conjunction with the node entity (type number 134).

Form 10: This form number conveys special information when used in conjunction with the Node entity (type number 134) in Finite Element Applications.

Refer to Fig. 3-17(a) for notation. The matrix R and the vector T are used to transform coordinate data from the u1,u2,u3 coordinate system to the x,y,z local system.

The u1,u2,u3 coordinate system has its origin at an arbitrary fixed point col XOFFSET, YOFFSET, ZOFFSET in the x,y,z coordinate system and is assumed to be displaced parallel to that reference coordinate system. Thus,

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad T = \begin{bmatrix} \text{XOFFSET} \\ \text{YOFFSET} \\ \text{ZOFFSET} \end{bmatrix}$$

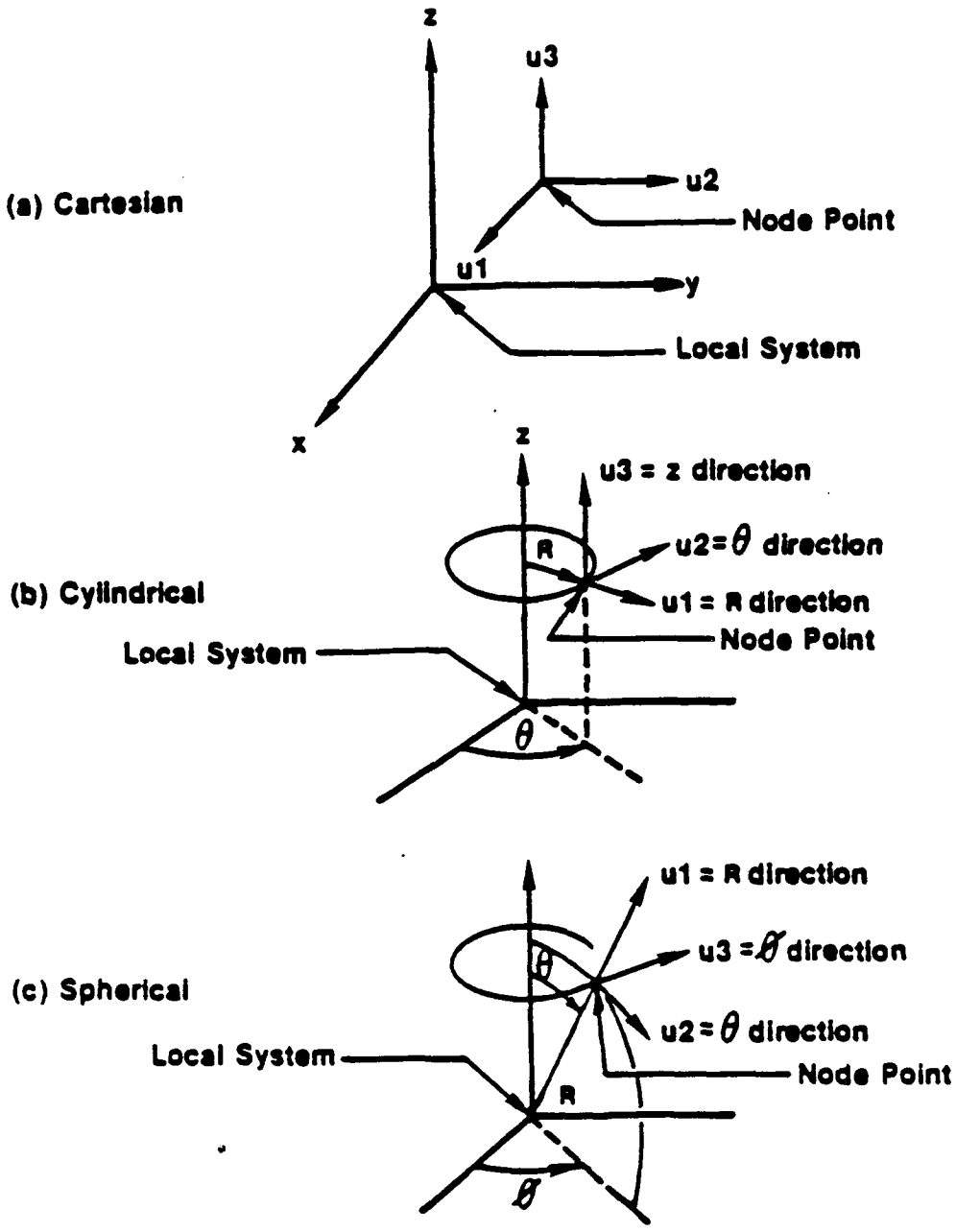


Figure 3-17 Displacement Components

so that

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u1 \\ u2 \\ u3 \end{bmatrix} + \begin{bmatrix} XOFFSET \\ YOFFSET \\ ZOFFSET \end{bmatrix} = \begin{bmatrix} XLOCAL \\ YLOCAL \\ ZLOCAL \end{bmatrix}$$

Note that the orientation of the two coordinate systems can be described by saying that the u1,u2,u3 coordinate system is the system obtained by imposing orthogonal curvilinear coordinates onto the x,y,z space and then constructing unit tangent vectors to the three curvilinear coordinate curves at the given fixed point to serve as basis vectors. In this special case of parallel displacement, the curvilinear coordinates imposed are identical to the existing x,y,z coordinates.

Form 11: This form number conveys special information when used in conjunction with the Node entity (type number 134) in Finite Element applications.

Refer to Figure 3-17(b) for notation. The matrix R and the vector T are used to transform coordinate data from the u1, u2, u3 coordinate system to the x,y,z local system.

The u1, u2, u3 coordinate system has its origin at an arbitrary fixed point

$$\begin{aligned} XOFFSET &= r_o \cos \theta_o & r_o \quad 0 \\ YOFFSET &= r_o \sin \theta_o & 0 \quad \theta_o \quad 360^\circ \\ ZOFFSET &= z_o & -\infty < z_o < \infty \end{aligned}$$

for $r_o = 0$, take $\theta = 0^\circ$

in the x,y,z coordinate system. The u1,u2,u3 system is the system obtained by imposing orthogonal curvilinear coordinates onto the x,y,z space which are the cylindrical coordinates (r,θ,z) with

$$x = r \cos \theta$$
$$y = r \sin \theta$$
$$z = z,$$

and then constructing unit tangent vectors to the three curvilinear coordinate curves at the given fixed point to serve as basis vectors.

Thus, the relationship between the u1, u2, u3 and the x,y,z local coordinate system is given by

$$\begin{bmatrix} \cos\theta_0 & -\sin\theta_0 & 0 \\ \sin\theta_0 & \cos\theta_0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u1 \\ u2 \\ u3 \end{bmatrix} + \begin{bmatrix} XOFFSET \\ YOFFSET \\ ZOFFSET \end{bmatrix} = \begin{bmatrix} XLOCAL \\ YLOCAL \\ ZLOCAL \end{bmatrix}$$

Form 12: This form number conveys special information when used in conjunction with the Node entity (type number 134) in Finite Element applications.

Refer to Fig. 3-17(c) for notation. The matrix R and the vector T are used to transform coordinate data from the u1, u2, u3 coordinate system to the x,y,z local system.

The u1, u2, u3 coordinate system has its origin at an arbitrary fixed point

$$XOFFSET = r_0 \sin\theta_0 \sin\phi_0 \qquad r_0 \geq 0$$
$$YOFFSET = r_0 \sin\theta_0 \cos\phi_0 \qquad 0 \leq \theta_0 \leq 180^\circ$$
$$ZOFFSET = r_0 \cos\theta_0 \qquad 0 \leq \phi_0 < 360^\circ$$

for $r_0 = 0$, take $\theta_0 = \phi_0 = 0^\circ$

for $\theta_0 = 0^\circ$ or 180°, take $\phi_0 = 0^\circ$

in the x,y,z coordinate system. The u1, u2, u3 system is the system obtained by imposing orthogonal curvilinear coordinates onto the x,y,z space which are the spherical coordinates (r, θ, ϕ) with

$$X = r \sin\theta \cos\phi$$
$$Y = r \sin\theta \sin\phi$$
$$Z = r \cos\theta$$

and then constructing unit tangent vectors to the three curvilinear coordinate curves at the given fixed point to serve as basis vectors.

Thus, the relationship between the u1, u2, u3 and the x,y,z local coordinate systems is given by

$$\begin{bmatrix} \sin\theta_0 & \cos\phi_0 & \cos\theta_0 & \cos\phi_0 & -\sin\phi_0 \\ \sin\theta_0 & \sin\phi_0 & \cos\theta_0 & \sin\phi_0 & \cos\phi_0 \\ \cos\theta_0 & & -\sin\theta_0 & & 0 \end{bmatrix} \begin{bmatrix} u1 \\ u2 \\ u3 \end{bmatrix} + \begin{bmatrix} XOFFSET \\ YOFFSET \\ ZOFFSET \end{bmatrix} = \begin{bmatrix} XLOCAL \\ YLOCAL \\ ZLOCAL \end{bmatrix}$$

See, Kaplan, (KAPL52) or Hildebrand, (HILD76) for a discussion of orthogonal curvilinear coordinate systems.

3.14.7 **Directory Data**

ENTITY TYPE NUMBER: 124

3.14.8 **Parameter Data**

Index	Name	Type	Description
1	R11	Real	Top Row
2	R12	Real	
3	R13	Real	
4	T1	Real	
5	R21	Real	Second Row
6	R22	Real	
7	R23	Real	
8	T2	Real	
9	R31	Real	Third Row
10	R32	Real	
11	R33	Real	
12	T3	Real	

Additional Pointers as required (see 2.2.4.4.2).

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: ESCAPE

Specific Escape Function Identifier: Set Dash

Description:

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

Additional Comments:

The line type capability proposed here is adapted from those in the PostScript language developed by Adobe Systems Incorporated.

Justification for Inclusion in the Register:

The Set Dash function is needed to support the user-specified dash pattern linetype. This linetype is needed to support the requirements of office document exchange and publishing. Similiar capabilities are commonly found in widely available proprietary graphics systems.

Relationship to Particular Standards:

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

2) See attached sheets.

1) CGM Functional Specification (reference ISO 8632 CGM; Part 1: Functional 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. If 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 distances in VDC units along the path of the primitive. These distances alternately specify the length of a gap between dashes. The contents of the array are used cyclically. 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 required to coordinate the dash pattern with features of an output primitive.

The *offset* value, in VDC units, 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.

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)
number of lengths (I)
(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 - as a clear text encoding (machine independent) of a FORTRAN-style packed data record. This is treated as a string type in each encoding and is encoded according to the rules for string in that encoding.

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	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:

The line cap capabilities proposed here are adapted from those in the PostScript language developed by Adobe Systems Incorporated.

Justification for Inclusion in the Register:

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

Relationship to Particular Standards:

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

2) See attached sheet.

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

1: butt cap; the portion is squared off at the endpoint of the path; there is no projection beyond the end of the path.

2: round cap; a semicircular arc with diameter equal to the line width is drawn around the endpoint and filled in with the current line colour.

3: projecting square cap; the portion continues beyond the endpoint of the path for the distance equal to half the line width and is squared off.

Values above 3 are reserved for future registration and standardization, and negative values are available for implementation-dependent use.

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 (IX)

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 - as a clear text encoding (machine independent) of a FORTRAN-style packed data record. This is treated as a string type in each encoding and is encoded according to the rules for string in that encoding.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	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:

The line join capabilities proposed here are adapted from those in the PostScript language developed by Adobe Systems Incorporated.

Justification for Inclusion in the Register:

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

Relationship to Particular Standards:

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

2) See attached sheet.

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

1 miter join; the outer edge of the two portions are extended until they meet at an angle, as in a picture frame. (If the portions meet at too sharp an angle, a bevel join is used instead; this is controlled by the miter limit state established by **Set Miter Limit** escape function).

2: round join; a circular arc with diameter equal to the line width is drawn around the point where the portions meet and is filled in with the current line colour, producing a rounded corner.

3: bevel join; the meeting portions are finished with butt end cap (see the **Set Line Cap** escape function); then the resulting notch beyond the ends of the portions is filled with a triangle in the current line colour.

Values above 3 are reserved for future registration and standardization, and negative values are available for implementation-dependent use.

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

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 (IX)

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 - as a clear text encoding (machine independent) of a FORTRAN-style packed data record. This is treated as a string type in each encoding and is encoded according to the rules for string in that encoding.
Data Record Description:

This page left intentionally blank.

PROPOSAL FOR REGISTRATION OF GRAPHICAL ITEM	
date of presentation of proposal	10 April 1987
sponsoring authority	ANSI

Class of Graphical Item: ESCAPE

Specific Escape Function Identifier: Set Miter Limit

Description:

This escape function sets a value for the current miter 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 miter joins into bevel joins. See attached sheets for additional details.

Additional Comments:

The line join capabilities proposed here are adapted from those in the PostScript language developed by Adobe Systems Incorporated.

Justification for Inclusion in the Register:

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

Relationship to Particular Standards:

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

2) See attached sheet.

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

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

At any given corner, the *miter 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 miter). This distance increases as the angle between the portions decreases. If the ratio of the miter length to the line width exceeds the miter limit parameter, the corner is treated with a bevel join instead of a miter join.

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

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

Examples of miter 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 the miter limit is 10. Setting the miter limit to 1 cuts off miters at all angles so that bevels are always produced even when miters are specified.

A functional description of the Set Miter Limit escape parameters is:

Parameters:

function identifier (I) as assigned by the Registration Authority

data record (D):
miter length (VDC)

Items for Data Record:

If VDC TYPE integer was selected:

Integer IL	1
Integer IA(1)	miter length
Integer RL	0
Integer SL	0

If VDC TYPE real was selected:

Integer IL	0
Integer RL	1
Real RA(1)	miter limit
Integer SL	0

Data Record Description:

The parameter defines the miter length.

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

All encodings will be handled in the same way - as a clear text encoding (machine independent) of a FORTRAN-style packed data record. This is treated as a string type in each encoding and is encoded according to the rules for string in that encoding.

This page left intentionally blank.

VII. REFERENCES

1. G. Enderle, K. Kansy, G. Pfaff, *Computer Graphics Programming*, Spring Verlag, Berlin, 1984.

2. ANSI X3.122-1986, Computer Graphics - Metafile for the Storage and Transfer of Picture Description Information.

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

4. Robert F. Sproul and Brian K. Reid, Introduction to Interpress, Xerox System Integration Guide 038404, Xexox Corporation, El Segundo, CA, April 1984.

5. Adobe Systems Inc., *Postscript Language Reference Manual* . Addison Wesley, Reading, MA, 1985.

6. Adobe Systems Inc., *Postscript Language Tutorial and Cookbook* , Addison Wesley, Reading, MA, 1985.

7. Brad Smith and Joan Wellington, *Initial Graphics Exchange Specification (IGES)*, Version 3.0, National Bureau of Standards, April 1986.

8. *Modern Drafting Practices and Standards Manual*, Genium Publishing, Schenectudy, NY, 1986.

This page left intentionally blank.

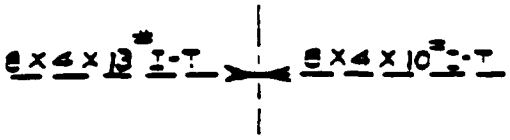
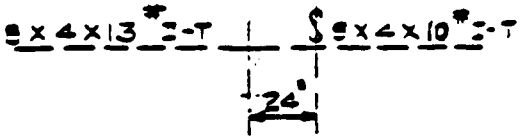
VIII. APPENDICES

Appendix A. Extracts from Typical DoD Standards

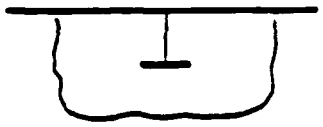
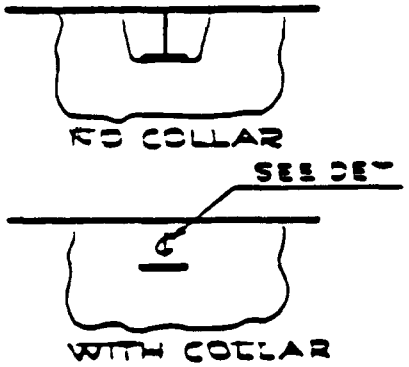
SYMBOL	INTERPRETATION
	DIMENSION LINES, PLATE EDGES AND STIFFENERS, NEAR SIDE
	BULKHEADS, DECKS OR OTHER PLATING IN SECTION
	PLATE EDGES, FLOORS, GIRDERS, BEAMS, STIFFENERS, ETC., FAR SIDE OR BELOW
	DECKS AND BULKHEADS, FAR SIDE OR BELOW
	CENTERLINE
	OPENING OR SPECIFIC AREA OR BOUNDARY
	WELDED SEAM OR BUTT
	SINGLE FASTENED LAP
	DOUBLE FASTENED LAP
	TRIPLE FASTENED LAP
	FASTENED SINGLE BUTT OR SEAM STRAP
	FASTENED DOUBLE BUTT OR SEAM STRAP

SH 13003

FIGURE 1. Conventional delineation.

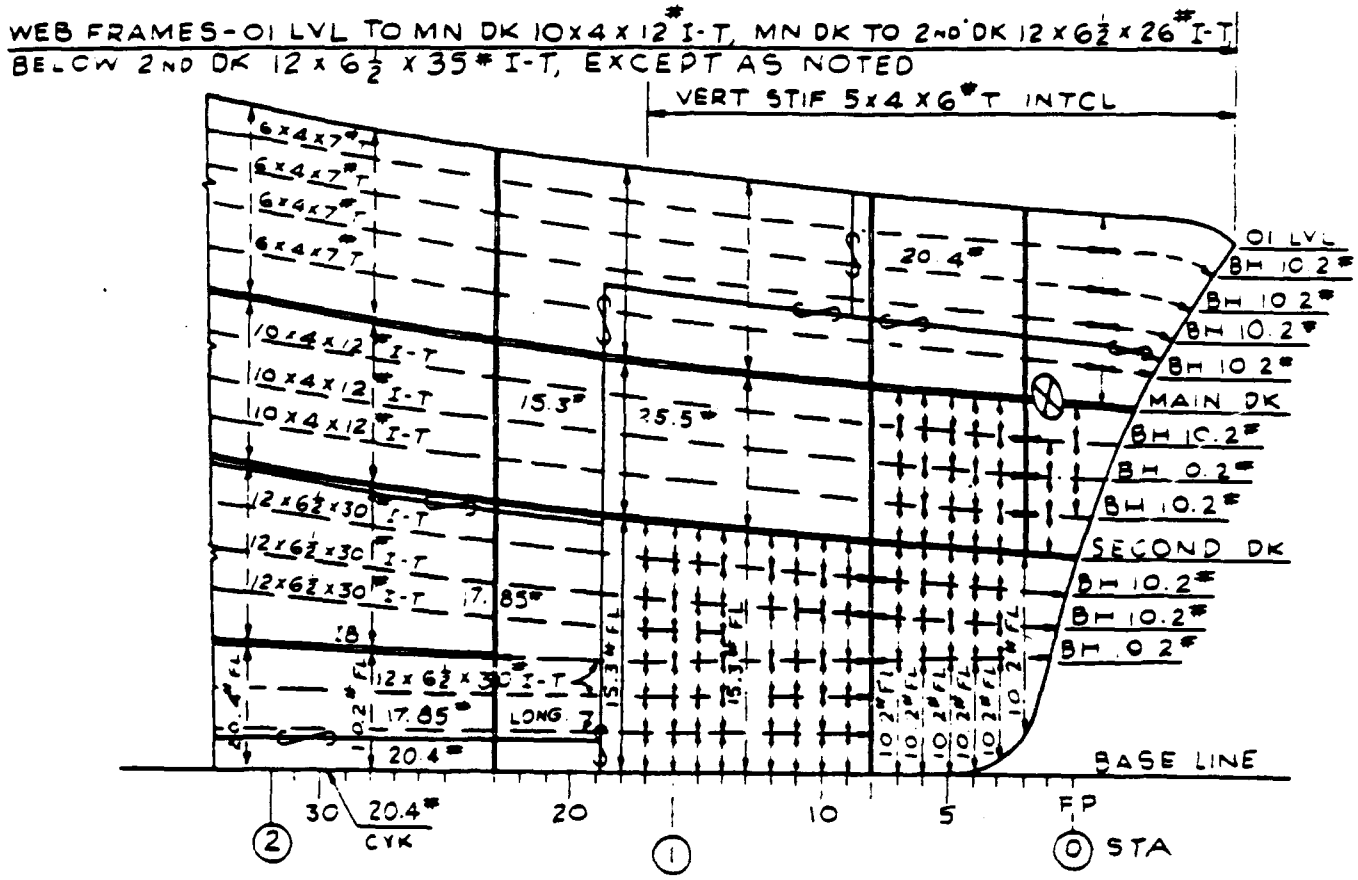
SYMBOL	INTERPRETATION
	LIMITS OF BEAMS - CONTINUOUS OR INTERCOSTAL (CONTRACT DRAWINGS) OR LIMITS OF INTERCOSTAL BEAMS (SHIP CONSTRUCTION DRAWINGS)
	LIMITS OF CONTINUOUS BEAMS (SHIP CONSTRUCTION DRAWINGS)

CONTINUOUS SHAPE THROUGH PLATE

	CONTRACT DRAWINGS: COLLAR DETAILS SPECIFIED IN APPROPRIATE SPECIFICATION
	SHIP CONSTRUCTION DRAWINGS: COLLAR DETAILS SHOWN ON DETAIL VIEW AS REQUIRED

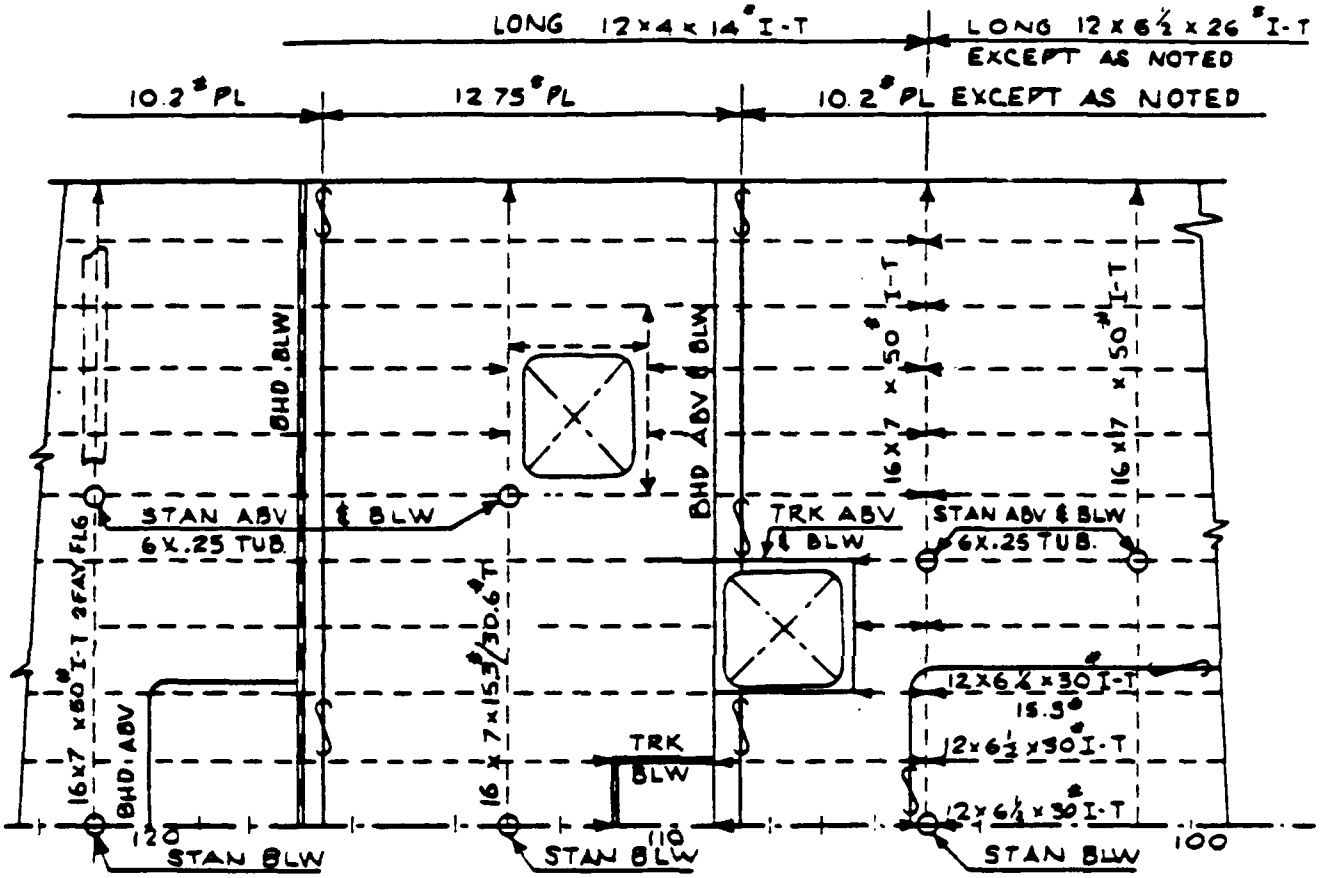
SH 13028-1

FIGURE 1. Conventional delineation. - Continued



SH 13029

FIGURE 2. Contract drawing or contract guidance drawing - shell expansion.



SH 13030

FIGURE 3. Contract drawing or contract guidance drawing - deck plan.

MIL-STD-17B-1
23 January 1963
SUPERSEDING
MIL-STD-17A
11 September 1961

MILITARY STANDARD

MECHANICAL SYMBOLS

(OTHER THAN AERONAUTICAL, AEROSPACECRAFT AND SPACECRAFT USE)

PART-1

DEPARTMENT OF DEFENSE

WASHINGTON 25, D. C.

Mechanical Symbols (Other
Than Aeronautical, Aerospacecraft
and Spacecraft Use)

MIL-STD-17B-1

1. This standard has been approved by the Department of Defense and is mandatory for use by the Departments of the Army, the Navy, and the Air Force, effective 23 January 1963.

2. Recommended corrections, additions, or deletions should be addressed to the Standardization Division, Defense Supply Agency, Alexandria, Virginia.

FOREWORD

By permission of the American Standards Association and the American Society of Mechanical Engineers, the symbols contained herein have been adopted without change from the following American Standards:

ASA Z32.2.3-1949 (Reaffirmed 1953) - Graphical Symbols for Pipe Fittings, Valves and Piping.

ASA Y32.4-1955 - Graphical Symbols for Plumbing.

ASA Z32.2.4-1949 (Reaffirmed 1954) - Graphical Symbols for Heating, Ventilating, and Air Conditioning.

ASA Z32.2.6-1950 (Reaffirmed 1956) - Graphical Symbols for Heat-Power Apparatus.

ASA Y32.10-1958 - Graphical Symbols for Fluid Power Diagrams.

CONTENTS

Page

1.1 Scope ---------- 1

1.2 Limitations ---------- 1

1.3 Application ---------- 1

1.4 Use ---------- 1

2. Graphical symbols for pipe fittings, valves and piping ---------- 2

3. Graphical symbols for plumbing ---------- 10

4. Graphical symbols for heating, ventilating and air conditioning ---------- 13

5. Graphical symbols for heat power apparatus ---------- 22

6. Graphical symbols for fluid power diagrams ---------- 26

Section 1, Scope.

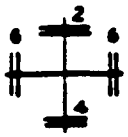
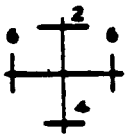
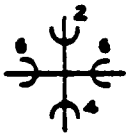
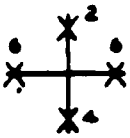
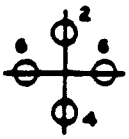
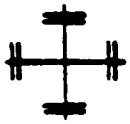
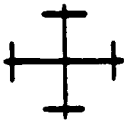
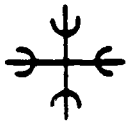
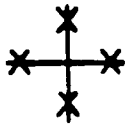
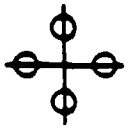
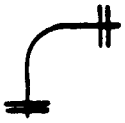
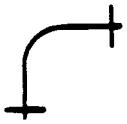
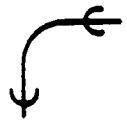
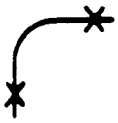
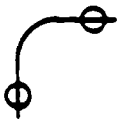
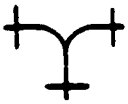
1.1 This standard establishes an approved listing of mechanical symbols that shall be used in the preparation of drawings when graphic or symbolic representation is desired, for all items except aircraft, aerospacecraft, or spacecraft. The symbols listed are those most commonly employed on engineering drawings. They are for exterior and interior services.

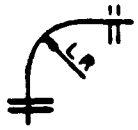
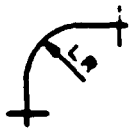
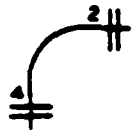
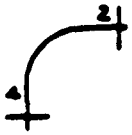
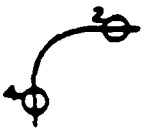
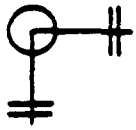
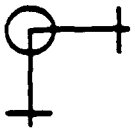
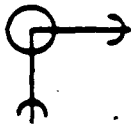
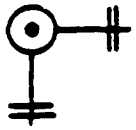
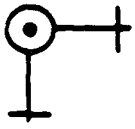
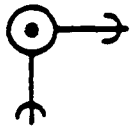
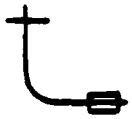
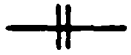
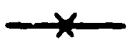
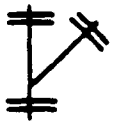
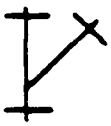
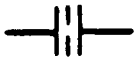
1.2 Limitations. - Where graphical symbols are required for an item or equipment not covered in this standard, the form and character of the symbol will be left to the discretion of the activity concerned, provided that the symbol used does not duplicate any of those contained herein, and is clearly understandable, subject to one interpretation only, or explained by a suitable note on the drawing when necessary.

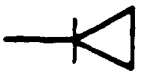
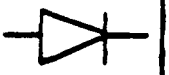
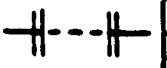
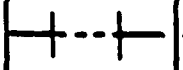
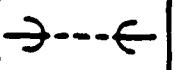
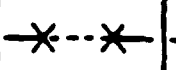
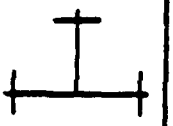
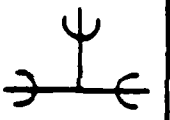
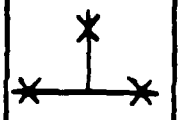
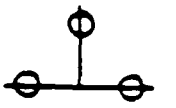
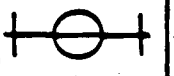
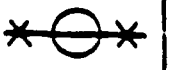
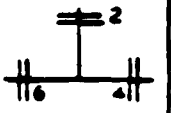
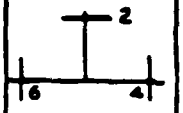
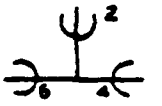
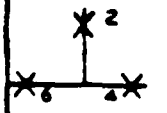
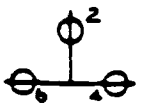
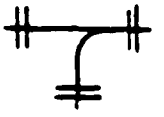
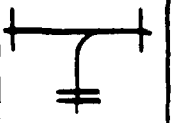
1.3 Application. - Pipe fittings, valves and piping symbols (see section 2) shall be used for all systems, except where so desired, the symbols shown for fluid power systems (see section 6) may be used for fluid power systems. All other symbols shall be used as applicable.

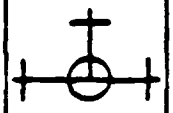
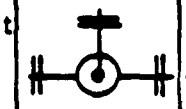
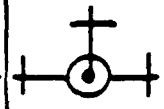
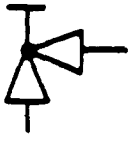
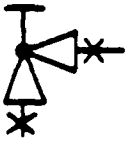
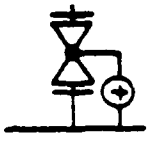
1.4 Use. - Each drawing or one sheet of each set of drawings on which symbols are used shall show a legend which shall identify the symbols. However, those symbols which are unmistakably identified on the drawing by the text or note at or near the symbols, may be omitted in the legend. In lieu of identifying each symbol, it is permissible to state that symbols used are in accordance with this Standard or the appropriate American Standards. Since symbolic representation does not usually involve exact or scale layout or the actual run or leads of piping, the same symbol may be used for all projections of the system (plan, elevations and sections) except where specific symbols for the various views are included in this standard.

Section 2 - Graphical symbols for pipe fittings, valves, and piping extracted from Publication ASA Z32.2.3:

Pipe fittings, valves and piping	Flanged	Screwed	Bell and spigot	Welded	Soldered
2.1 Bushing					
2.2 Cap					
2.3 Cross					
2.3.1 Reducing					
2.3.2 Straight size					
2.4 Crossover					
2.5 Elbow					
2.5.1 45-Degree					
2.5.2 90-Degree					
2.5.3 Turned down					
2.5.4 Turned up					
2.5.5 Base					
2.5.6 Double branch					

Pipe fittings valves and piping (cont'd.)	Flanged	Screwed	Bell and spigot	Welded	Soldered
2.5.7 Long radius					
2.5.8 Reducing					
2.5.9 Side outlet (outlet down)					
2.5.10 Side outlet (outlet up)					
2.5.11 Street					
2.6 Joint					
2.6.1 Connecting pipe					
2.6.2 Expansion					
2.7 Lateral					
2.8 Orifice flange					
2.9 Reducing flange					

Pipe fittings, valves and piping (cont'd.)	Flanged	Screwed	Bell and spigot	Welded	Soldered
2.10 Plugs					
2.10.1 Bull plug					
2.10.2 Pipe plug					
2.11 Reducer					
2.11.1 Concentric					
2.11.2 Eccentric					
2.12 Sleeve					
2.13 Tee					
2.13.1 (Straight size)					
2.13.2 (Outlet up)					
2.13.3 (Outlet down)					
2.13.4 (Double sweep)					
2.13.5 Reducing					
2.13.6 (Single sweep)					

Pipe fittings, valves and piping (cont'd.)	Flanged	Screwed	Bell and spigot	Welded	Soldered
2.13.7 Side outlet (outlet down)					
2.13.8 Side outlet (outlet up)					
2.14 Union					
2.15 Anglevalve					
2.15.1 Check					
2.15.2 Gate (elevation)					
2.15.3 Gate (plan)					
2.15.4 Globe (elevation)					
2.15.5 Globe (plan)					
2.15.6 Hose angle	same as	symbol	2.23.1		
2.16 Automatic valve					
2.16.1 By-pass					

Section 6 - Graphical symbols for fluid power diagrams, extracted from ASA Publication Y32.10:

6.1 This section shows the basic symbols, describes the principles on which the symbols are based, and illustrates some representative composite symbols. Composite symbols can be devised for any fluid power component by combining basic symbols. A symbol is considered to be the lines, letters and abbreviations which identify purpose and method of operation of component represented, and symbols only are standardized in this section. Component data are added to symbols when used in circuit diagrams and consist of names of ports and control elements, notes regarding pressure and flow rate settings and other explanatory data.

6.2 Symbol rules. The following rules are applicable to graphical symbols for fluid power diagrams:

(a) Symbols show connections, flow paths and function of component represented. They do not indicate conditions occuring during transition from one flow path arrangement to another. Further, they do not indicate construction, or values such as pressure, flow rate, and other component settings.
(b) Symbols do not indicate location of ports, direction of shifting of spools, or position of control elements on actual component.
(c) Symbols may be rotated or reversed without altering their meaning except in the cases of:
    (1) Lines to reservoir, 6.3.7.1 and 6.3.7.2
    (2) Vented manifold, 6.8.3
(d) Line width does not alter meaning of symbols.
(e) Symbols may be drawn any suitable size. Size may be varied for emphasis or clarity.
(f) Letter combinations used as parts of graphical symbols are not necessarily abbreviations, provided their meaning is clearly understandable and subject to one interpretation only or explained by a suitable note or tabulated legend on the drawing when necessary.
(g) Where flow lines cross, a loop shall be used except within a symbol envelope. Loop may be used in this case if clarity is improved.
(h) In multiple envelope symbols, flow condition shown nearest a control symbol takes place when that control is caused or permitted to actuate.
(i) Each symbol is drawn to show normal or neutral condition of component unless multiple circuit diagrams are furnished showing various phases of circuit operation.
(j) Arrows shall be used within symbol envelopes to show direction of flow path in component as used in the application represented. Double-end arrow shall be used to indicate reversing flow.
(k) External ports are where flow lines connect to basic symbol, except where component enclosure is used.
External ports are at intersections of flow lines and component enclosure symbol when enclosure is used.

Fluid power diagram symbols

6.3 Flow Lines And Flow Line Functions

6.3.1 Lines, Working

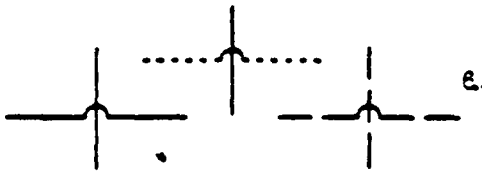
6.3.2 Lines, Pilot

Length of dash shall be at least 20 line widths with space approximately 5 line widths.

6.3.3 Lines, Liquid Drain or Air Exhaust

Length of dash and space shall be approximately equal, each less than 5 line widths.

6.3.4 Lines, Crossing



6.3.5 Lines, Joining

Connector dot shall be approximately 5 widths of associated lines.

6.3.6 Lines, Flexible

6.3.7 Lines to Reservoir

6.3.7.1 *Below Fluid Level*

6.3.7.2 *Above Fluid Level*

6.3.8 Flow, Direction of

6.3.9 Plug or Plugged Connection

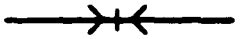
6.3.9.1 *Testing Station*

6.3.9.2 *Fluid Power Take-off Station*

6.3.10 Restriction, Fixed

6.3.11 Quick Disconnect

6.3.11.1 *Without Checks*



6.3.11.2 *With Checks*

6.3.11.21 Disconnected

6.3.11.22 With One Check

6.3.11.23 With Two Checks

6.4 Pumps, Compressors And Rotary Motors

6.4.1 Basic Symbol

6.4.1.1 *Envelope*

Appendix B. PostScript and Interpress Capabilities

PostScript Procedures and Interpress Symbols

POSTSCRIPT Procedures

A POSTSCRIPT procedure is a set of operations grouped together with a common name. This set of operations is stored with its key in a dictionary. When the key appears in a program, the associated set of operations is carried out.

POSTSCRIPT procedures are defined in exactly the same way as variables. The program must place the procedure name (preceded by a slash) on the stack, followed by the set of operations that make up the procedure. Then the **def** operator is used to store the operations and name in the current dictionary. The set of operations making up the procedure must be enclosed in braces.

For example, the following line defines a procedure named *inch*.

```
/inch {72 mul} def
```

Any appearances of the word *inch* following this line will cause the interpreter to carry out the operations inside the braces. That is, the interpreter will put *72* on the stack and then multiply the top two numbers, the *72* and whatever was on the stack when *inch* was called. Thus, the program lines

```
5 inch
5 72 mul
```

have identical results; both leave *360*, the product of *5* and *72*, on the stack.

The *inch* procedure is a useful tool in many programs, since it translates inches into the 1/72-inch units of the POSTSCRIPT coordinate system.

4.3 USING PROCEDURES AND VARIABLES

The use of procedures and variables can make an enormous difference in the readability of a program, the ease with which it can be modified, and its length.

Three Boxes Again

As an example, let us take the last program from chapter two, the three overlapping boxes, and rewrite it. Looking over the program, we see that the set of instructions that construct a one-inch-square path is repeated three times. Let us define these instructions to be a procedure named *box* and then incorporate this procedure into the program.

Overlapping Boxes

```
% ----- Define box procedure ---
/box
 { 72 0 rlineto
   0 72 rlineto
   -72 0 rlineto
   closepath } def
% --------- Begin Program ----------
newpath          % First box
  252 324 moveto box
  0 setgray fill
newpath          % Second box
  270 360 moveto box
  .4 setgray fill
newpath          % Third box
  288 396 moveto box
  .8 setgray fill
showpage
```

Here we start by defining our new procedure, *box*, to be the set of operators that create a square path. We then use that procedure three times to make three filled boxes. First we move to a starting point on the current page. Then we call the *box* procedure, which contructs a path starting at that point. Finally, we set the gray value and fill the path we contructed. These steps are repeated two more times, once for each box in our image.

14

Instancing

The concepts of *symbol* and *instance* are provided in Interpress by composed operators and transformations. A graphical symbol can be defined as a composed operator. When an instance, or copy, of the symbol is to be printed, the current transformation will be applied to all coordinates as the symbol calls imager operators. The properties of the current transformation will thus determine the position, size, and rotation of the instance on the printed page.

The principal use of symbols and instances in Interpress is for printing characters. Each character is defined by a composed operator, called a *character operator*. These operators are invoked, usually by SHOW, with a current transformation that achieves the proper size, orientation, and position of the character.

Instancing can also be used for other purposes. Graphical objects that are repeated often on a page or throughout a document may be treated as instances. A symbol is defined as a composed operator and called with an appropriate current transformation in order to generate each instance. Since SHOW may not be the best operator to effect these calls, other primitives are available as well.

14.1 Defining symbols

A symbol is simply a composed operator that calls imager operators to construct a graphical symbol. The symbol expresses coordinates in a coordinate system of its own choice, sometimes called the *symbol coordinate system*. Figure 14.1 shows a stick figure and an associated coordinate system. Example 14.1 shows how this symbol could be defined as a composed operator. To Interpress, this symbol is simply a composed operator. Its effective use as a symbol depends on the ways in which the master calls the composed operator.

```
--Example 14.1--
--0--     {                                  --a symbol is a composed operator--
--1--     0 2 15 ISET                        --set strokeWidth to 0.1 units--
--2--     2 10 ISET                          --set strokeEnd to 2 (round) --
--3--     -2 0 0 4 MASKVECTOR                --left leg--
--4--      2 0 0 4 MASKVECTOR                --right leg--
--5--      0 4 0 8 MASKVECTOR                --torso--
--6--     -2 4 0 6 MASKVECTOR                --left arm--
--7--      2 4 0 6 MASKVECTOR                --right arm--
--8--     -1 9 1 9 MASKVECTOR                --head--
--9--     } MAKESIMPLECO 13 FSET             --make composed operator and save it in frame[13]--
```

14.1

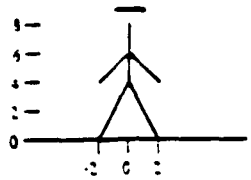


Figure 14.1. A symbol defined in its coordinate system

14.2 Making instances

In order to place an instance of a symbol on the page, we will need to use a transformation to control the size, rotation, and location of the instance. This transformation is responsible for mapping from the symbol coordinate system to the current coordinate system, the one established by T. As we observed in Section 6.2, it's common to establish a page coordinate system that has a convenient origin and units of measurement. The examples in this section will assume a page coordinate system as in Example 6.1, which uses units of 10^{-5} meter.

When we establish the symbol-to-page transformation, it is usually helpful to break the transformation down into two components: (1) a translation, which is used to determine where the origin of the symbol coordinate system will be on the page coordinate system, and (2) a scaling and/or rotation transformation that determines the size and rotation of the instance with respect to its origin.

Suppose, for example, that we want instances of the stick figure in Example 14.1 to be 10 cm. high from toe to head. Since 10 cm. is 10000 units in the page coordinate system, and since the head-to-toe distance is 9 units in the symbol coordinate system, the scale to convert from symbol to page coordinates will be 10000/9.

The following master prints two instances of the symbol, with the same sizes, with origins at $x = 5$ cm., $y = 8$ cm., and at $x = 15$ cm., $y = 8$ cm.:

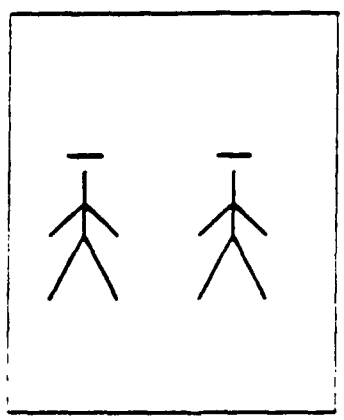


Figure 14.2. Two instances of a symbol.

```
--Example 14.2--
-- 1--    BEGIN                                  --begin preamble--
-- 2--                                           --A symbol is a composed operator--
-- 3--    2 0 15 SETW                            --set strokewidth to 2.1 units--
-- 3--    2 16 15SETC                            --set strokeend to 2 round.--
-- 4--    -2 0 0 4 MASKVECTOR                    --left leg--
-- 5--     2 0 0 4 MASKVECTOR                    --right leg--
-- 6--     0 4 0 6 MASKVECTOR                    --torso--
-- 7--    -2 4 0 5 MASKVECTOR                    --left arm--
-- 8--     2 4 0 5 MASKVECTOR                    --right arm--
-- 9--    -1 9 1 9 MASKVECTOR                    --head--
--10--    } MAKESIMPLECO 13 FSET                 --make composed operator and save it in frame[13]--
--11--    }                                      --end preamble--
--12--    { 0.00001 SCALE CONCATT                --page coordinate system in units of 0.00001 meter--
--13--    { 10000/9 SCALE  5000 8000 TRANSLATE CONCAT CONCATT 13 FGET DO } DOSAVESIMPLEBODY
--14--    { 10000/9 SCALE 15000 8000 TRANSLATE CONCAT CONCATT 13 FGET DO } DOSAVESIMPLEBODY
--15--    }                                      --end of page body--
--16--    END
```

Line 13 alters the current transformation so that coordinates in the symbol will be subjected first to a scaling transformation, then a translation, and finally, to the page transformation. Line 13 could equally well be written as <{ 10000/9 SCALE 5000 8000 TRANSLATE CONCATT CONCATT 13 FGET DO }>, but could *not* be written as <{ 10000/9 SCALE CONCATT 5000 8000 TRANSLATE CONCATT 13 FGET DO }> because this would change the order of application of transformations. However, the following variant would do:

```
--Example 14.3, replacement for line 13 in Example 14.2--
--13--    { 5000 8000 TRANSLATE CONCATT 10000/9 SCALE CONCATT 13 FGET DO } DOSAVESIMPLEBODY
```

You should see clearly how this is equivalent to the original line 13. Understanding the order of application of transformations, and how transformations are concatenated onto *T*, is extremely important.

Note the use of DOSAVESIMPLEBODY to save and restore the current transformation. At the beginning of line 13, the current transformation establishes the page coordinate system. Because of the saving and restoring, this same system will also be in effect after line 13 is executed. If the transformation were not restored, the CONCATT operation on line 13 would have a permanent effect on the current transformation, leaving it in the symbol coordinate system associated with the first instance of the stick figure.

An alternative way to call an instance is to use the current position and MOVE or TRANS to accomplish the translation transformation. Example 14.3 could be modified to read:

```
--Example 14.4, replacement for line 13 in Example 14.2--
--13--    { 5000 8000 SETXY MOVE 10000/9 SCALE CONCATT 13 FGET DO } DOSAVESIMPLEBODY
```

Or, after one minor change, to read:

```
--Example 14.5, replacement for line 13 in Example 14.2--
--13a--   5000 8000 SETXY
--13b--   { MOVE 10000/9 SCALE CONCATT 13 FGET DO } DOSAVESIMPLEBODY
```

If we're going to make lots of instances of the symbol with the same size and orientation, this form has a nice property: line 13b will be the same for each instance because it contains no positioning information. This suggests defining a composed operator to achieve the effect of line 13b. Example 14.6 restates the master in Example 14.2 to use this idea.

```
--Example 14 8  same image as Example 14 2--
-- 0--    BEGIN {                                  --begin preamble--
-- 1--    {                                        --a symbol is a composed operator--
-- 2--    0 2 15 ISET                              --set strokewidth to 0 1 units--
-- 3--    2 18 ISET                                --set strokeEnd to 2 (round) --
-- 4--    -2 0 3 4 MASKVECTOR                      --left leg--
-- 5--     2 0 3 4 MASKVECTOR                      --right leg--
-- 6--     0 4 0 8 MASKVECTOR                      --torso--
-- 7--    -2 4 3 8 MASKVECTOR                      --left arm--
-- 8--     2 4 3 8 MASKVECTOR                      --right arm--
-- 9--    -1 9 1 9 MASKVECTOR                      --head--
--10--    } MAKESIMPLECO 13 FSET                   --make composed operator and save it in /Fsmd[13]--
--11--    {                                        --another composed operator--
--12--    MOVE 10000/9 SCALE CONCATT 13 FGET DO
--13--    } MAKESIMPLECO 14 FSET                   --make composed operator and save it in /Fsmd[14]--
--14--    }                                        --end preamble--
--15--    { 0.00001 SCALE CONCATT                  --page coordinate system in units of 0 00001 meter--
--16--    5000 8000 SETXY                          --set current position to x=5 cm, y=8 cm--
--17--    14 FGET DOSAVE                           --print first instance--
--18--    15000 8000 SETXY
--19--    14 FGET DOSAVE                           --print second instance--
--20--    }                                        --end of page body--
--21--    END
```

This example looks a lot like the way character operators work! The composed operator constructed in lines 1 to 10 corresponds to a character operator, defined in the character coordinate system. The composed operator constructed in lines 11 to 13 corresponds roughly to the operator created by MODIFYFONT, which controls the scale and rotation of the instance. The invocation of the operator by first setting the current position to the desired origin of the instance and then calling an appropriate operator (lines 16 and 17) corresponds roughly to the way SETXY and SHOW are used for characters.

The basic Interpress facilities of composed operators and transformations allow a great deal of flexibility in defining and using symbols, including applications we have not illustrated. Symbols may of course be rotated as well as scaled. Symbols can call other symbols, nesting to an arbitrary depth.

PostScript Graphical Operators

CHAPTER 6

OPERATORS

6.1 INTRODUCTION

This chapter contains detailed information about all the standard operators in the POSTSCRIPT language. It is divided into two parts.

First, there is a summary of the operators, organized into groups of related functions. The summary is intended to assist in locating the operators needed to perform specific tasks.

Second, there are detailed descriptions of all the operators, organized alphabetically by operator name. Each operator description is presented in the following format:

operator $operand_1$ $operand_2$ ... $operand_n$ **operator** $result_1$ ... $result_m$

Detailed explanation of the operator

EXAMPLE:
An example of the use of this operator. The symbol ⇒ designates values left on the operand stack by the example.

ERRORS:
A list of the errors that this operator might execute

SEE ALSO:
A list of related operator names

At the head of an operator description, $operand_1$ through $operand_n$ are the operands that the operator requires, with $operand_n$ being the topmost element on the operand stack. The operator pops these objects from the operand stack and consumes them. After executing, the operator leaves the objects $result_1$ through $result_m$ on the stack, with $result_m$ being the topmost element.

Normally the operand and result names suggest their types. For example, *num* indicates that the operand or result is a number, *int* indicates an integer number, *any* indicates a value of any type, and *proc* indicates a POSTSCRIPT procedure (i.e., an executable array). The notation '*bool|int*' indicates a value that is either a boolean or an integer. Names representing numbers sometimes suggest their purpose, e.g., *x*, *y*, or *angle*. A *matrix* is an array of six numbers describing a transformation matrix (see section 4.4). A *font* is a dictionary constructed according to the rules for font dictionaries (see section 5.3).

The notation '⊢' indicates the bottom of the stack. The notation '–' in the operand position indicates that the operator expects no operands, and a '–' in the result position indicates that the operator returns no results.

The documented effects on the operand stack and the possible errors are those produced directly by the operator itself. Many operators cause arbitrary POSTSCRIPT procedures to be invoked. Obviously, such procedures can have arbitrary effects that are not mentioned in the operator descriptions.

6.2 OPERATOR SUMMARY

Operand stack manipulation operators

any	**pop**	–	discard top element *193*
any_1 any_2	**exch**	any_2 any_1	exchange top two elements *151*
any	**dup**	any any	duplicate top element *148*
any_1 ... any_n n	**copy**	any_1 ... any_n any_1 ... any_n	duplicate top *n* elements *131*
any_n ... any_0 n	**index**	any_n ... any_0 any_n	duplicate arbitrary element *172*
a_{n-1} ... a_0 n j	**roll**	$a_{j-1 \bmod n}$ ... a_0 a_{n-1} ... $a_{j \bmod n}$	roll *n* elements up *j* times *205*
⊢ any ... any_n	**clear**	⊢	discard all elements *127*
⊢ any_1 ... any_n	**count**	⊢ any_1 ... any_n n	count elements on stack *132*
–	**mark**	mark	push mark on stack *184*

mark obj_1..obj_n	**cleartomark**	–	discard elements down through *mark* *127*
mark obj_1..obj_n	**counttomark**	mark obj_1..obj_n	count elements down to mark *133*

Arithmetic and math operators

num, num_2	**add**	sum	*num*, plus num_2 *115*
num, num_2	**div**	quotient	*num*, divided by num_2 *148*
int_1 int_2	**idiv**	quotient	integer divide *168*
int_1 int_2	**mod**	remainder	*int*, mod int_2 *185*
num, num_2	**mul**	product	*num*, times num_2 *186*
num, num_2	**sub**	difference	*num*, minus num_2 *230*
num_1	**abs**	num_2	absolute value of *num*, *115*
num_1	**neg**	num_2	negative of *num*, *187*
num_1	**ceiling**	num_2	ceiling of *num*, *126*
num_1	**floor**	num_2	floor of *num*, *157*
num_1	**round**	num_2	round *num*, to nearest integer *206*
num_1	**truncate**	num_2	remove fractional part of *num*, *234*
num	**sqrt**	real	square root of *num* *224*
num den	**atan**	angle	arctangent of *num/den* in degrees *121*
angle	**cos**	real	cosine of *angle* (degrees) *132*
angle	**sin**	real	sine of *angle* (degrees) *223*
base exponent	**exp**	real	raise *base* to *exponent* power *154*
num	**ln**	real	natural logarithm (base *e*) *180*
num	**log**	real	logarithm (base 10) *181*
–	**rand**	int	generate pseudo-random integer *197*
int	**srand**	–	set random number seed *224*
–	**rrand**	int	return random number seed *207*

Array operators

int	**array**	array	create array of length *int* *120*
–	**[**	mark	start array construction *113*
mark obj_0..obj_{n-1}	**]**	array	end array construction *113*
array	**length**	int	number of elements in *array* *179*
array index	**get**	any	get array element indexed by *index* *164*
array index any	**put**	–	put *any* into *array* at *index* *195*
array index count	**getinterval**	subarray	subarray of *array* starting at *index* for *count* elements *165*
array, index $array_2$	**putinterval**	–	replace subarray of *array*, starting at *index* by $array_2$ *196*
array	**aload**	a_0..a_{n-1}, array	push all elements of *array* on stack *115*
any_0..any_{n-1}, array	**astore**	array	pop elements from stack into *array* *121*
$array_1$ $array_2$	**copy**	$subarray_2$	copy elements of *array*, to initial subarray of $array_2$ *131*
array proc	**forall**	–	execute *proc* for each element of *array* *161*

Dictionary operators

int	**dict**	dict	create dictionary with capacity for *int* elements *146*
dict	**length**	int	number of key-value pairs in *dict* *179*
dict	**maxlength**	int	capacity of *dict* *185*
dict	**begin**	–	push *dict* on dict stack *124*
–	**end**	–	pop dict stack *149*
key value	**def**	–	associate *key* and *value* in current dict *145*
key	**load**	value	search dict stack for *key* and return associated *value* *181*
key value	**store**	–	replace topmost definition of *key* *227*
dict key	**get**	any	get value associated with *key* in *dict* *164*
dict key value	**put**	–	associate *key* with *value* in *dict* *195*
dict key	**known**	bool	test whether *key* is in *dict* *177*
key	**where**	dict true *or* false	find dict in which *key* is defined *238*
dict, $dict_2$	**copy**	$dict_2$	copy contents of *dict*, to $dict_2$ *131*
dict proc	**forall**	–	execute *proc* for each element of *dict* *161*
–	**errordict**	dict	push **errordict** on operand stack *151*
–	**systemdict**	dict	push **systemdict** on operand stack *231*
–	**userdict**	dict	push **userdict** on operand stack *236*
–	**currentdict**	dict	push current dict on operand stack *134*
–	**countdictstack**	int	count elements on dict stack *133*
array	**dictstack**	subarray	copy dict stack into *array* *147*

String operators

int	**string**	string	create string of length *int* *228*
string	**length**	int	number of elements in *string* *179*
string index	**get**	int	get string element indexed by *index* *164*
string index int	**put**	–	put *int* into *string* at *index* *195*
string index count	**getinterval**	substring	substring of *string* starting at *index* for *count* elements *165*
string, index $string_2$	**putinterval**	–	replace substring of *string*, starting at *index* by $string_2$ *196*
string, $string_2$	**copy**	$substring_2$	copy elements of *string*, to initial substring of $string_2$ *131*
string proc	**forall**	–	execute *proc* for each element of *string* *161*
string seek	**anchorsearch**	post match true *or* string false	determine if *seek* is initial substring of *string* *116*
string seek	**search**	post match pre true *or* string false	search for *seek* in *string* *211*
string	**token**	post token true *or* false	read token from start of *string* *232*

Relational, boolean, and bitwise operators

any, any_2	**eq**	bool	test equal *150*
any, any_2	**ne**	bool	test not equal *186*
num, str, num_2,str_2	**ge**	bool	test greater or equal *163*
num, str, num_2,str_2	**gt**	bool	test greater than *167*
num,|str, num_2|str_2	**le**	bool	test less or equal *179*
num,|str, num_2|str_2	**lt**	bool	test less than *182*
bool,|int, $bool_2$|int_2	**and**	$bool_3$|int_3	logical	bitwise and *116*
bool,|int,	**not**	$bool_2$|int_2	logical	bitwise not *189*
bool,|int, $bool_2$|int_2	**or**	$bool_3$|int_3	logical	bitwise inclusive or *191*
bool,|int, $bool_2$|int_2	**xor**	$bool_3$|int_3	logical	bitwise exclusive or *241*
–	**true**	true	push boolean value *true* *234*
–	**false**	false	push boolean value *false* *155*
int_1 shift	**bitshift**	int_2	bitwise shift of int_1 (positive is left) *125*

Control operators

any	**exec**	–	execute arbitrary object *152*
bool proc	**if**	–	execute *proc* if *bool* is true *169*
bool $proc_1$ $proc_2$	**ifelse**	–	execute $proc_1$ if *bool* is true, $proc_2$ if *bool* is false *169*
init incr limit proc	**for**	–	execute *proc* with values from *init* by steps of *incr* to *limit* *160*
int proc	**repeat**	–	execute *proc* *int* times *202*
proc	**loop**	–	execute *proc* an indefinite number of times *182*
–	**exit**	–	exit innermost active loop *154*
–	**stop**	–	terminate stopped context *228*
any	**stopped**	bool	establish context for catching stop *227*
–	**countexecstack**	int	count elements on exec stack *133*
array	**execstack**	subarray	copy exec stack into *array* *152*
–	**quit**	–	terminate interpreter *197*
–	**start**	–	executed at interpreter startup *225*

Type, attribute, and conversion operators

any	**type**	name	return name identifying *any*'s type *235*
any	**cvlit**	any	make object be literal *141*
any	**cvx**	any	make object be executable *144*
any	**xcheck**	bool	test executable attribute *240*
array|file|string	**executeonly**	array|file|string	reduce access to execute-only *153*
array|dict|file|string	**noaccess**	array|dict|file|string	disallow any access *188*
array|dict|file|string	**readonly**	array|dict|file|string	reduce access to read-only *200*
array|dict|file|string	**rcheck**	bool	test read access *198*
array|dict|file|string	**wcheck**	bool	test write access *238*
num|string	**cvi**	int	convert to integer *141*
string	**cvn**	name	convert to name *142*
num|string	**cvr**	real	convert to real *142*
num radix string	**cvrs**	substring	convert to string with radix *143*

any string	**cvs**	substring	convert to string *144*

File operators

$string_1$ $string_2$	**file**	file	open file identified by $string_1$ with access $string_2$ *155*
file	**closefile**	–	close file *129*
file	**read**	int true *or* false	read one character from *file* *198*
file int	**write**	–	write one character to *file* *239*
file string	**readhexstring**	substring bool	read hex from *file* into *string* *199*
file string	**writehexstring**	–	write *string* to *file* as hex *240*
file string	**readstring**	substring bool	read string from *file* *201*
file string	**writestring**	–	write characters of *string* to *file* *240*
file string	**readline**	substring bool	read line from *file* into *string* *200*
file	**token**	token true *or* false	read token from *file* *232*
file	**bytesavailable**	int	number of bytes available to read *125*
–	**flush**	–	send buffered data to standard output file *158*
file	**flushfile**	–	send buffered data or read to EOF *158*
file	**resetfile**	–	discard buffered characters *202*
file	**status**	bool	return status of *file* *226*
string	**run**	–	execute contents of named file *207*
–	**currentfile**	file	return file currently being executed *135*
string	**print**	–	write characters of *string* to standard output file *193*
any	**=**	–	write text representation of *any* to standard output file *114*
⊢ any_1 .. any_n	**stack**	⊢ any_1 .. any_n	print stack nondestructively using = *224*
any	**==**	–	write syntactic representation of *any* to standard output file *114*
⊢ any_1 .. any_n	**pstack**	⊢ any_1 .. any_n	print stack nondestructively using == *194*
–	**prompt**	–	executed when ready for interactive input *194*
bool	**echo**	–	turn on/off echoing *149*

Virtual memory operators

–	**save**	save	create VM snapshot *208*
save	**restore**	–	restore VM snapshot *203*
–	**vmstatus**	level used maximum	report VM status *238*

Miscellaneous operators

proc	**bind**	proc	replace operator names in *proc* by operators *124*
–	**null**	null	push null on operand stack *189*
–	**usertime**	int	return time in milliseconds *237*
–	**version**	string	interpreter version *237*

Graphics state operators

–	**gsave**	–	save graphics state *166*
–	**grestore**	–	restore graphics state *165*
–	**grestoreall**	–	restore to bottommost graphics state *166*
–	**initgraphics**	–	reset graphics state parameters *173*
num	**setlinewidth**	–	set line width *219*
–	**currentlinewidth**	num	return current line width *137*
int	**setlinecap**	–	set shape of line ends for stroke (0=butt, 1=round, 2=square) *217*
–	**currentlinecap**	int	return current line cap *136*
int	**setlinejoin**	–	set shape of corners for stroke (0=miter, 1=round, 2=bevel) *218*
–	**currentlinejoin**	int	return current line join *137*
num	**setmiterlimit**	–	set miter length limit *220*
–	**currentmiterlimit**	num	return current miter limit *137*
array offset	**setdash**	–	set dash pattern for stroking *214*
–	**currentdash**	array offset	return current dash pattern *134*
num	**setflat**	–	set flatness tolerance *215*
–	**currentflat**	num	return current flatness *135*
num	**setgray**	–	set color to gray value from 0 (black) to 1 (white) *216*
–	**currentgray**	num	return current gray *136*
hue sat brt	**sethsbcolor**	–	set color given hue, saturation, brightness *216*
–	**currenthsbcolor**	hue sat brt	return current color hue, saturation, brightness *136*
red green blue	**setrgbcolor**	–	set color given red, green, blue *221*
–	**currentrgbcolor**	red green blue	return current color red, green, blue *138*
freq angle proc	**setscreen**	–	set halftone screen *221*
–	**currentscreen**	freq angle proc	return current halftone screen *139*
proc	**settransfer**	–	set gray transfer function *222*
–	**currenttransfer**	proc	return current transfer function *139*

Coordinate system and matrix operators

–	**matrix**	matrix	create identity matrix *184*
–	**initmatrix**	–	set CTM to device default *174*
matrix	**identmatrix**	matrix	fill *matrix* with identity transform *167*
matrix	**defaultmatrix**	matrix	fill *matrix* with device default matrix *145*
matrix	**currentmatrix**	matrix	fill *matrix* with CTM *137*
matrix	**setmatrix**	–	replace CTM by *matrix* *219*
t_x t_y	**translate**	–	translate user space by (t_x, t_y) *233*
t_x t_y matrix	**translate**	matrix	define translation by (t_x, t_y) *233*
s_x s_y	**scale**	–	scale user space by s_x and s_y *209*
s_x s_y matrix	**scale**	matrix	define scaling by s_x and s_y *209*
angle	**rotate**	–	rotate user space by *angle* degrees *206*
angle *matrix*	**rotate**	*matrix*	define rotation by *angle* degrees *206*
matrix	**concat**	–	replace CTM by *matrix* × CTM *130*

matrix, $matrix_2$ $matrix_3$	**concatmatrix**	$matrix_3$	fill $matrix_3$ with *matrix*, × $matrix_2$ *130*
x y	**transform**	x′ y′	transform (x, y) by CTM *233*
x y matrix	**transform**	x′ y′	transform (x, y) by *matrix* *233*
dx dy	**dtransform**	dx′ dy′	transform distance (dx, dy) by CTM *148*
dx dy matrix	**dtransform**	dx′ dy′	transform distance (dx, dy) by *matrix* *148*
x′ y′	**itransform**	x y	inverse transform (x′, y′) by CTM *176*
x′ y′ matrix	**itransform**	x y	inverse transform (x′, y′) by *matrix* *176*
dx′ dy′	**idtransform**	dx dy	inverse transform distance (dx′, dy′) by CTM *168*
dx′ dy′ matrix	**idtransform**	dx dy	inverse transform distance (dx′, dy′) by *matrix* *168*
$matrix_1$ $matrix_2$	**invertmatrix**	$matrix_2$	fill $matrix_2$ with inverse of $matrix_1$ *175*

Path construction operators

–	**newpath**	–	initialize current path to be empty *187*
–	**currentpoint**	x y	return current point coordinate *138*
x y	**moveto**	–	set current point to (x, y) *186*
dx dy	**rmoveto**	–	relative moveto *204*
x y	**lineto**	–	append straight line to (x, y) *180*
dx dy	**rlineto**	–	relative lineto *204*
x y r ang, ang_2	**arc**	–	append counterclockwise arc *117*
x y r ang, ang_2	**arcn**	–	append clockwise arc *118*
x_1 y_1 x_2 y_2 r	**arcto**	xt_1 yt_1 xt_2 yt_2	append tangent arc *119*
x_1 y_1 x_2 y_2 x_3 y_3	**curveto**	–	append Bezier cubic section *140*
dx_1 dy_1 dx_2 dy_2 dx_3 dy_3	**rcurveto**	–	relative curveto *198*
–	**closepath**	–	connect subpath back to its starting point *130*
–	**flattenpath**	–	convert curves to sequences of straight lines *157*
–	**reversepath**	–	reverse direction of current path *203*
–	**strokepath**	–	compute outline of stroked path *229*
string bool	**charpath**	–	append character outline to current path *127*
–	**clippath**	–	set current path to clipping path *129*
–	**pathbbox**	ll_x ll_y ur_x ur_y	return bounding box of current path *191*
move line curve close	**pathforall**	–	enumerate current path *192*
–	**initclip**	–	set clip path to device default *172*
–	**clip**	–	establish new clipping path *128*
–	**eoclip**	–	clip using even-odd inside rule *149*

Painting operators

–	**erasepage**	–	paint current page white *151*
–	**fill**	–	fill current path with current color *156*
–	**eofill**	–	fill using even-odd rule *150*
–	**stroke**	–	draw line along current path *229*
width height bits/sample matrix proc	**image**	–	render sampled image onto current page *170*

width height invert matrix proc	**imagemask**	–	render mask onto current page *171*

Device setup and output operators

–	**showpage**	–	output and reset current page *223*
–	**copypage**	–	output current page *132*
matrix width height proc	**banddevice**	–	install band buffer device *123*
matrix width height proc	**framedevice**	–	install frame buffer device *162*
–	**nulldevice**	–	install no-output device *190*
proc	**renderbands**	–	enumerate bands for output to device *201*

Character and font operators

key font	**definefont**	font	register *font* as a font dictionary *146*
key	**findfont**	font	return font dict identified by *key* *156*
font scale	**scalefont**	font'	scale *font* by *scale* to produce new *font'* *210*
font matrix	**makefont**	font'	transform *font* by *matrix* to produce new *font'* *183*
font	**setfont**	–	set font dictionary *215*
–	**currentfont**	font	return current font dictionary *136*
string	**show**	–	print characters of *string* on page *222*
a_x a_y string	**ashow**	–	add (a_x, a_y) to width of each char while showing *string* *120*
c_x c_y char string	**widthshow**	–	add (c_x, c_y) to width of *char* while showing *string* *239*
c_x c_y char a_x a_y string	**awidthshow**	–	combined effects of ashow and widthshow *122*
proc string	**kshow**	–	execute *proc* between characters shown from *string* *178*
string	**stringwidth**	w_x w_y	width of *string* in current font *228*
–	**FontDirectory**	dict	dictionary of font dictionaries *159*
–	**StandardEncoding**	array	standard font encoding vector *225*

Font cache operators

–	**cachestatus**	bsize bmax msize mmax csize cmax blimit	return cache status and parameters *126*
w_x w_y ll_x ll_y ur_x ur_y	**setcachedevice**	–	declare cached character metrics *212*
w_x w_y	**setcharwidth**	–	declare uncached character metrics *213*
num	**setcachelimit**	–	set max bytes in cached character *213*

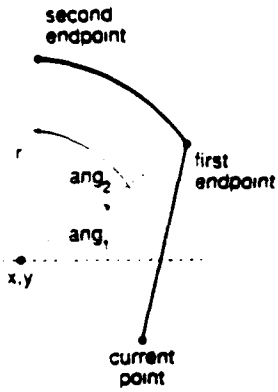
Errors

dictfull	no more room in dictionary *146*
dictstackoverflow	too many begins *147*
dictstackunderflow	too many ends *147*
execstackoverflow	exec nesting too deep *153*
handleerror	called to report error information *167*

interrupt	external interrupt request e.g. control-C) *174*
invalidaccess	attempt to violate access attribute *174*
invalidexit	exit not in loop *175*
invalidfileaccess	unacceptable access string *175*
invalidfont	invalid font name or dict *175*
invalidrestore	improper restore *175*
ioerror	input/output error occurred *176*
limitcheck	implementation limit exceeded *180*
nocurrentpoint	current point is undefined *188*
rangecheck	operand out of bounds *197*
stackoverflow	operand stack overflow *224*
stackunderflow	operand stack underflow *225*
syntaxerror	syntax error in PostScript program text *230*
timeout	time limit exceeded *231*
typecheck	operand of wrong type *235*
undefined	name not known *235*
undefinedfilename	file not found *236*
undefinedresult	over/underflow or meaningless result *236*
unmatchedmark	expected mark not on stack *236*
unregistered	internal error *236*
VMerror	VM exhausted *237*

arc *x y r* ang, ang_2 **arc** –

appends a counterclockwise arc of a circle to the current path, possibly preceded by a straight line segment. The arc has (*x*, *y*) as center, *r* as radius, ang_1 the angle of a vector from (*x*, *y*) of length *r* to the first endpoint of the arc, and ang_2 the angle of a vector from (*x*, *y*) of length *r* to the second endpoint of the arc.



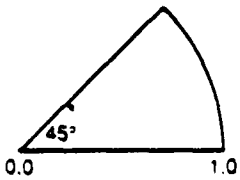
If there is a current point, the **arc** operator includes a straight line segment from the current point to the first endpoint of this arc and then adds the arc itself into the current path. If the current path is empty, the **arc** operator does not produce the initial straight line segment. In any event, the second endpoint of the arc becomes the new current point.

Angles are measured in degrees counterclockwise from the positive *x*-axis of the current user coordinate system. The curve produced is circular in user space. If user space is scaled non-uniformly (i.e., differently in *x* and *y*), **arc** will produce elliptical curves on the output device.

The operators that produce arcs (**arc**, **arcn**, and **arcto**) represent them internally as one or more Bézier cubic curves (see **curveto**) that approximate the required shape. This is done with sufficient accuracy that a faithful rendition of an arc is produced. However, a program that reads the constructed path (using **pathforall**) will encounter **curveto** segments where arcs were specified originally.

EXAMPLE:

```
newpath 0 0 moveto 0 0 1 0 45 arc closepath
```



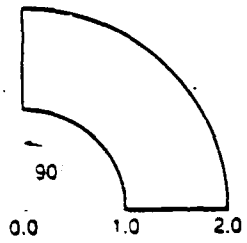
This constructs a unit radius 45 degree 'pie slice'.

ERRORS:
limitcheck, rangecheck, stackunderflow, typecheck

SEE ALSO:
arcn, arcto, curveto

arcn *x y r ang₁ ang₂* **arcn** –

(arc negative) behaves like **arc**, but **arcn** builds its arc segment in a clockwise direction (in user space).



EXAMPLE:

```
newpath 0 0 2 0 90 arc 0 0 1 90 0 arcn closepath
```

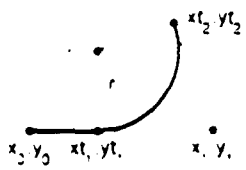
This constructs a 2 unit radius, 1 unit wide, 90 degree 'windshield wiper swath'.

ERRORS:
limitcheck, rangecheck, stackunderflow, typecheck

SEE ALSO:
arc, arcto, curveto

arcto $x_1\ y_1\ x_2\ y_2\ r$ **arcto** $xt_1\ yt_1\ xt_2\ yt_2$

appends an arc of a circle to the current path, possibly preceded by a straight line segment. The arc is defined by a radius r and two tangent lines. The tangent lines are those drawn from the current point, here called (x_0, y_0), to (x_1, y_1) and from (x_1, y_1) to (x_2, y_2). (If the current point is undefined, **arcto** executes the error **nocurrentpoint**.)



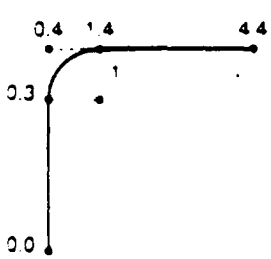
The center of the arc is located within the inner angle between the tangent lines; it is the only point located at distance r in a direction perpendicular to both lines. The arc begins at the first tangent point (xt_1, yt_1) on the first tangent line, passes between its center and the point (x_1, y_1), and ends at the second tangent point (xt_2, yt_2) on the second tangent line.

Before constructing the arc, **arcto** adds a straight line segment from the current point (x_0, y_0) to (xt_1, yt_1), unless those points are the same. In any event, (xt_2, yt_2) becomes the new current point.

The curve produced is circular in user space. If user space is scaled non-uniformly (i.e., differently in x and y), **arcto** will produce elliptical curves on the output device.

If the two tangent lines are collinear, **arcto** merely appends a straight line segment from (x_0, y_0) to (x_1, y_1), considering the arc to be part of a degenerate circle (with radius 0) at that point.

The values returned by **arcto** are for information only; they are the two tangent points in user space.



EXAMPLE:

```
newpath 0 0 moveto
0 4 4 4 1 arcto
4 {pop} repeat
4 4 lineto
```

This constructs a 4 unit wide, 4 unit high right angle with a 1 unit radius 'rounded corner.'

ERRORS:
limitcheck, nocurrentpoint, stackunderflow, typecheck, undefinedresult

SEE ALSO:
arc, arcn, curveto

clip – **clip** –

intersects the inside of the current clipping path with the inside of the current path to produce a new (smaller) current clipping path. The inside of the current path is determined by the normal PostScript non-zero winding number rule (see section 4.6), while the inside of the current clipping path is determined by whatever rule was used at the time that path was created. Before computing the intersection, the **clip** operator implicitly closes any open subpaths of the current path.

In general, **clip** produces a new path whose inside (according to the non-zero winding number rule) consists of all areas that are inside both of the original paths. The manner in which this new path is constructed (order of its segments, whether or not it self-intersects, etc.) is not specified.

There is no way to enlarge the current clipping path (other than by **initclip** or **initgraphics**) or to set a new path without reference to the current one. The recommended way of using **clip** is to bracket the **clip** and the sequence of graphics to be clipped with **gsave** and **grestore**. The **grestore** will restore the clipping path that was in effect before the **gsave**.

Unlike **fill** and **stroke**, **clip** does not implicitly perform a **newpath** after it has finished using the current path. Any subsequent path construction operators will append to the current path unless **newpath** is executed explicitly. This can be a source of unexpected behavior.

ERRORS:
limitcheck

SEE ALSO:
eoclip, **clippath**, **initclip**

closepath – **closepath** –

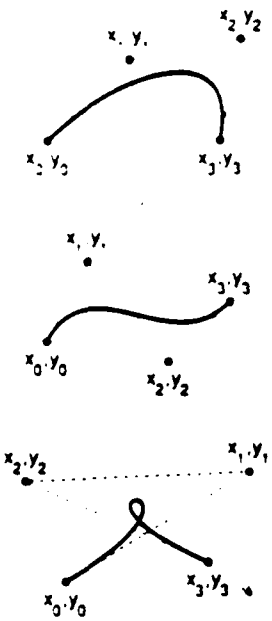
closes the current subpath by appending a straight line segment connecting the current point to the subpath's starting point (generally the point most recently specified by **moveto**). If the current subpath is already closed or the current path is empty, **closepath** does nothing. (See section 4.5.)

closepath terminates the current subpath. Appending another segment to the current path will begin a new subpath, even if it is drawn from the endpoint reached by the **closepath**.

ERRORS:
limitcheck

SEE ALSO:
fill, stroke

curveto $x_1\ y_1\ x_2\ y_2\ x_3\ y_3$ **curveto** –



adds a Bézier cubic section to the current path between the current point, referred to here as (x_0, y_0), and the point (x_3, y_3), using (x_1, y_1) and (x_2, y_2) as the Bézier cubic control points. After constructing the curve, **curveto** makes (x_3, y_3) become the new current point. If the current point is undefined (because the current path is empty), **curveto** executes the error **nocurrentpoint**.

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_3, y_3) 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 formulation of a Bézier 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 **curveto** is the path traced by $x(t)$ and $y(t)$ as t ranges from 0 to 1. The Bézier control points corresponding to this curve are:[1]

$$x_1 = x_0 + c_x/3 \qquad y_1 = y_0 + c_y/3$$
$$x_2 = x_1 + (c_x + b_x)/3 \qquad y_2 = y_1 + (c_y + b_y)/3$$
$$x_3 = x_0 + c_x + b_x + a_x \qquad y_3 = y_0 + c_y + b_y + a_y$$

ERRORS:
limitcheck, nocurrentpoint, stackunderflow, typecheck

SEE ALSO:
lineto, moveto, arc, arcn, arcto

For a more thorough treatment of the mathematics of Bézier cubics, see J. D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982.

moveto x y **moveto** –

starts a new subpath of the current path. **moveto** sets the current point in the graphics state to the user space coordinate (x, y) without adding any line segments to the current path.

If the previous path operation in the current path was also a **moveto** (or **rmoveto**), that point is deleted from the current path and the new **moveto** point replaces it.

ERRORS:
limitcheck, stackunderflow, typecheck

SEE ALSO:
rmoveto, lineto, curveto, arc, closepath

rcurveto $dx_1\ dy_1\ dx_2\ dy_2\ dx_3\ dy_3$ **rcurveto** –

(relative curveto) adds a Bézier cubic section to the current path in the same manner as **curveto**: however, the three number pairs are interpreted as displacements relative to the current point (x_0, y_0) rather than as absolute coordinates. That is, **rcurveto** constructs a curve from (x_0, y_0) to (x_0+dx_3, y_0+dy_3), using (x_0+dx_1, y_0+dy_1) and (x_0+dx_2, y_0+dy_2) as Bézier control points. See the description of **curveto** for complete information.

ERRORS:
limitcheck, nocurrentpoint, stackunderflow, typecheck, undefinedresult

SEE ALSO:
curveto, rlineto, rmoveto

rlineto dx dy **rlineto** –

(relative lineto) appends a straight line segment to the current path in the same manner as **lineto**; however, the number pair is interpreted as a displacement relative to the current point (x, y) rather than as an absolute coordinate. That is, **rlineto** constructs a line from (x, y) to $(x+dx, y+dy)$ and makes $(x+dx, y+dy)$ be the new current point.

ERRORS:
limitcheck, nocurrentpoint, stackunderflow, typecheck

SEE ALSO:
lineto, rmoveto, rcurveto

rmoveto dx dy **rmoveto** –

(relative moveto) starts a new subpath of the current path in the same manner as **moveto**; however, the number pair is interpreted as a displacement relative to the current point (x, y) rather than as an absolute coordinate. That is, **rmoveto** makes $(x+dx, y+dy)$ be the new current point, without connecting it to the previous point. If the current point is undefined (because the current path is empty), **rmoveto** executes the error operator **nocurrentpoint**.

ERRORS:
limitcheck, nocurrentpoint, stackunderflow, typecheck

SEE ALSO:
moveto, rlineto, rcurveto

stroke – **stroke** –

paints a line following the current path and using the current color. This line is centered on the path, has sides parallel to the path segments, and has a width (thickness) given by the current line width parameter in the graphics state (see **setlinewidth**). **stroke** paints the joints between connected path segments with the current line join (see **setlinejoin**) and the ends of open subpaths with the current line cap (see **setlinecap**). The line is either solid or broken according to the dash pattern established by **setdash**.

The parameters in the graphics state controlling line rendition (line width, line join, and so forth) are consulted at the time **stroke** is executed: their values during the time the path is being constructed are irrelevant.

If a subpath is degenerate (consists of a single point), **stroke** paints it only if round line caps have been specified, producing a filled circle centered at that point. If butt or projecting square line caps have been specified, **stroke** produces no output, since the orientation of the caps would be indeterminate.

stroke implicitly performs a **newpath** after it has finished painting the current path. To preserve the current path across a **stroke** operation, use the sequence: **gsave stroke grestore**.

ERRORS:
limitcheck

SEE ALSO:
setlinewidth, setlinejoin, setmiterlimit, setlinecap, setdash

strokepath – **strokepath** –

replaces the current path with one enclosing the shape that would result if the **stroke** operator were applied to the current path. The path resulting from **strokepath** is suitable as the implicit operand to **fill**, **clip**, or **pathbbox**. In general, this path is not suitable for **stroke**, as it may contain interior segments or disconnected subpaths produced by **strokepath**'s stroke to outline conversion process.

ERRORS:
limitcheck

SEE ALSO:
fill, clip, stroke, pathbbox, charpath

setflat num **setflat** –

sets the flatness parameter in the current graphics state to *num*, which must be a positive number. This controls the accuracy with which curved path segments are to be rendered on the raster output device by the **stroke**, **fill**, and **clip** operators. Those operators render curves by approximating them with a series of straight line segments. 'Flatness' is an informal term for the error tolerance of this approximation: it is the maximum distance of any point of the approximation from the corresponding point on the true curve, measured in output device pixels.

The choice of flatness value is a tradeoff between accuracy and execution efficiency. Very small values (less than 1 device pixel) produce very accurate curves at high cost, since enormous numbers of tiny line segments must be produced. Larger values produce cruder approximations with substantially less computation. A default value of the flatness parameter is established by the device setup routine for each raster output device; this value is based on characteristics of that device and is the one suitable for most applications.

ERRORS:
stackunderflow, **typecheck**

SEE ALSO:
currentflat, **flattenpath**, **stroke**, **fill**

PostScript Image Capabilities

image width height bits/sample matrix proc **image** –

renders a sampled image onto the current page. The description here only summarizes the **image** operator; see section 4.7 for full details.

The sampled image is a rectangular array of *width* × *height* sample values, each of which consists of *bits/sample* bits of data (1, 2, 4, or 8). The data is received as a sequence of characters, i.e., 8-bit integers in the range 0 to 255. If *bits/sample* is less than 8, sample values are packed left to right within a character (but see the note in section 4.7).

The image is considered to exist in its own coordinate system. The rectangular boundary of the image has its lower left corner at (0, 0) and its upper right corner at (*width*, *height*). The *matrix* operand specifies a transformation from user space to the image coordinate system.

image executes *proc* repeatedly to obtain the actual image data. *proc* must return (on the operand stack) a string containing any number of additional characters of sample data. (If *proc* returns a string of length zero, **image** will terminate execution prematurely.) The sample values are assumed to be received in a fixed order: (0, 0) through (*width*–1, 0), then (0, 1) through (*width*–1, 1), etc.

Use of the **image** operator after a **setcachedevice** within the context of a **BuildChar** procedure is not permitted (an **undefined** error results); however, use of **imagemask** in that context is permitted (see the **setcachedevice** operator and section 5.7).

EXAMPLE:

```
/picstr 256 string def          % string to hold image data
45 140 translate                % locate lower left corner of image
132 132 scale                   % map image to 132 point square

256 256 8                       % dimensions of source image
[256 0 0 -256 0 256]            % map unit square to source
{currentfile                    % read image data from program file
 picstr readhexstring pop}
image

4c47494b4d4c524c4d50535051554c5152 ...
5959565c5 ... (131072 hex digits of image data)
```

ERRORS:
stackunderflow, typecheck, undefinedresult, undefined

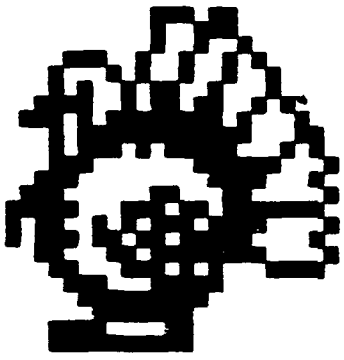
SEE ALSO:
imagemask

imagemask *width height invert matrix proc* **imagemask** –

is similar to the **image** operator; however, it treats the source image as a *mask* of 1 bit samples that are used to control where to apply paint (with the current color) and where not to. See the description of the **image** operator and the presentation of sampled images in section 4.7.

imagemask uses the *width*, *height*, *matrix*, and *proc* operands in precisely the same way as **image**. The *invert* operand is a boolean that determines the polarity of the mask. If *invert* is *false*, portions of the image corresponding to source sample values of 0 are painted, while those corresponding to sample values of 1 are left unchanged. If *invert* is *true*, sample values of 1 are painted and sample values of 0 are left unchanged.

imagemask is most useful for printing characters represented as bitmaps. Such bitmaps represent masks through which a color is to be transferred; the bitmaps themselves do not have a color (see section 4.7).



EXAMPLE:

```
45 228 translate              % locate lower left corner of square
132 132 scale                 % scale 1 unit to 132 points

0 0 moveto 0 1 lineto         % fill square with gray background
1 1 lineto 1 0 lineto closepath
.9 setgray fill

0 setgray                     % paint mask black
24 23                         % dimensions of source mask
true                          % paint the 1 bits
[24 0 0 -23 0 23]             % map unit square to mask
{<003800 002700 002480 0E4940 114920
  14B220 3CB650 75FE88 17FF8C 175F14
  1C07E2 3803C4 703182 F8EDFC B2BBC2
  BB6F84 31BFC2 18EA3C 0E3E00 07FC00
  03F800 1E1800 1FF800>}      % mask data
imagemask
```

ERRORS:
stackunderflow, typecheck, undefinedresult

SEE ALSO:
image

This page left intentionally blank.

Appendix C. IGES Entities and Capabilities

APPENDIX 3

IGES ENTITIES AS RENDERED BY GRAPHICS STANDARDS SYSTEM

This Appendix describes each of the IGES entities in turn, noting how well these entities could be rendered by a system based upon the graphics standards.

1. Geometric Entities

CIRCULAR ARC. Not available in GKS/PHIGS. Present as CIRCLE and 3-POINT ARC in CGI/CGM.

COMPOSITE CURVE. Rendered by POLYLINE in GKS/PHIGS; by POLYLINE, ARC, and ELLIPTICAL ARC in CGI/CGM. The full conics and splines of IGES are not directly supported, but would have to be approximated by POLYLINE.

CONIC ARC. Not available in GKS/PHIGS. Present as CIRCLE, ELLIPSE, 3-POINT ARC, and ELLIPTICAL ARC in CGI/CGM. Parabolas and hyperbolas are not directly supported.

COPIUS DATA. Directly available as POLYLINE with different line types.

PLANE. Unbounded planes need no direct visualization. Bounded planes correspond to POLYGON SET in GKS/PHIGS/CGM and are supplemented by CLOSED FIGURE in CGI.

LINE. Available as a two-point POLYLINE.

PARAMETRIC SPLINE. Can be visualized by lines, arcs, etc. However, information about the shape is lost.

PARAMETRIC SPLINE SURFACE. Can be visualized by rectangles, lines, arcs, etc. However, information about the shape is lost.

POINT. Available as POLYMARKERS for pre-defined symbols and positioned SEGMENTS (GKS/CGI but not CGM) or STRUCTURES (PHIGS) for user-defined symbols.

RULED SURFACE. Can be visualized by lines, etc. However, information about the shape is lost.

SURFACE OF REVOLUTION. Can be visualized by lines, arcs, etc. However, information about the shape is lost.

TABULATED CYLINDER. Can be visualized by lines, arcs, etc. However, information about the shape is lost.

TRANSFORMATION MATRIX. Really used like an attribute in IGES. Also special form numbers are used with certain contructs for Finite Element Analysis and viewing. Closely related to SEGMENT TRANSFORMATIONS of GKS and PHIGS transformation matrix structure elements used for modelling and viewing. No direct parallel in CGI or CGM.

FLASH ENTITY. In GKS, only have POLYMARKER or segments to realize the forms. In CGI/CGM, you also have CIRCLE and RECTANGLE. IGES flash entity form 4, rounded rectangle, is not directly available in any of the graphics standards.

RATIONAL B-SPLINE CURVE. Can be visualized by lines, arcs, etc. However, information about the shape is lost. Parabolas and hyperbolas not directly supported by the graphics standards.

RATIONAL B-SPLINE SURFACE. Can be visualized by lines, arcs, etc. However, information about the shape is lost. Parabolas and hyperbolas not directly supported by the graphics standards.

OFFSET CURVE. Can be visualized by lines, arcs, etc. However, information about the relationship between the two entities is lost.

CONNECT POINT. Can be visualized by lines. However, information about the relationship between the entities is lost.

NODE. Not directly visualized, so no need for support from the graphics standards. However, not unlike a PHIGS structure label.

FINITE ELEMENT. Thirty-three topologies are currently defined by IGES version 3. All can be visualized using the graphics primitives of line, arc, ellipse, etc., with loss of information concerning the relationships between the lines and curves making up the wire-frame model. Furthermore, the IGES specification is much more compact.

NODAL DISPLACEMENT AND ROTATION. These are attributes that are used to communicate finite element post processing data.

OFFSET SURFACE. Can be visualized by lines, arcs, etc. However, information about the relationship between the two entities is lost.

CURVE ON PARAMETRIC SURFACE. Can be visualized by lines, arcs, etc. However, information about the relationship between the curve and the surface is lost.

TRIMMED (PARAMETRIC) SURFACE. Can be visualized by lines, arcs, etc. However, information about the relationships among the boundary lines and other curved lines used to represent the surface is lost.

ANGULAR DIMENSION. Visualize using text, lines, arrows, and arcs. Only arrows not directly supported in the graphics standards. See discussion under LEADER entity below.

CENTERLINE. Use graphics circles (in CGI/CGM) only and special line types. Could be made available in GKS/PHIGS through special marker types.

DIAMETER. Visualize using text, lines, arrows, and arcs.

FLAG NOTE. Visualize using text and fill area (polygon) primitives.

GENERAL LABEL. Visualize using text, lines, and arrows.

GENERAL NOTE. A general note entity consists of one or more text strings. Each text string contains text, a starting point, text size, and angle of rotation of the text. A single font number applies to the whole note and incorporates the separate concepts of type face (appearance of the characters; e.g., bold Helvetica, italic Futura) and character set (shape of the characters; e.g., ASCII, German National Set, Math, Greek). Only 7-bit character codes are supported. In addition, a form number is used to designate the layout of the (possibly multiple) text strings, the justification of the strings within a text rectangle, and whether there are subscripts, superscripts, fractions, and embedded font changes. The GKS/PHIGS TEXT and CGI/CGM TEXT, APPEND TEXT, and RESTRICTED TEXT primitives with their numerous text attributes are capable of visualizing any general note entity. With RESTRICTED TEXT and APPEND TEXT, CGI/CGM are a bit more capable than GKS/PHIGS.

LEADER. Eleven arrow head types are specified in IGES version 3. Although all can easily be rendered using more primitive lines and polgons, the information that the arrowhead belongs with the leader line is lost when described using graphics standards primitives.

LINEAR DIMENSION. Visualized using text, lines, and arrows.

ORDINATE DIMENSION. Visualized using text, lines, and arrows.

POINT DIMENSION. Visualized using text, lines, arrows, circles, and polylines/polygons (hexagons).

RADIUS DIMENSION. Visualized using text, lines, arrows, and arcs.

SECTION. Visualized using FILL AREA with HATCH patterns. Of the eight pre-defined IGES patterns, two correspond to CGI/CGM hatch patterns; namely, IGES form 31 is CGM hatch style 3 and IGES form 37 is CGM hatch style 6.

GENERAL SYMBOL. Visualized using text, lines, arrows, etc., and possibly grouped in SEGMENTS (GKS or CGI but not CGM) or STRUCTURES (PHIGS).

SECTIONED AREA. Visualized with POLYLINE and FILL AREA in GKS/PHIGS/CGM and also using CLOSED FIGURE with fill in CGI. However, the IGES representation is generally more compact, because there is control over the angle and spacing of the lines that make up the cross-hatched fill pattern. In these cases, the DISJOINT POLYLINE element of CGI/CGM may help. In its default state, IGES line pattern code 1 corresponds to CGM hatch style 3, line pattern code 16 to CGM hatch style 6, and line pattern code 18 to CGM hatch style 5.

WITNESS LINE. Visualized using POLYLINES, although the DISJOINT POLYLINE of CGI/CGM may be useful in certain special cases.

ASSOCIATIVITY DEFINITION. Defines the relationship among entities. These relationships are lost in CGM, occasionally can be represented by GKS/CGI segments, and with somewhat greater likelihood can be represented by PHIGS structures.

ASSOCIATIVITY INSTANCE. Many predefined associativity relationships exist in IGES. These include simple and ordered GROUPS (both doubly linked and forward-only linked entities), external references and external reference files, labels, and parent-child structures. These have some parallels in PHIGS structures, but in general need not be directly handled by a graphics viewing system like CGI/GKS. Likewise, the PLANAR and FLOW instance types do not require direct support from the graphics system. The VIEWS VISIBLE and VIEWS VISIBLE, COLOR, LINE WEIGHT instance type have an effect on the visualization of the IGES model. The necessary controls for proper visualization are available in the graphics standards, using viewports, color tables, and line width specification mode. Finally, the DIMENSIONED GEOMETRY instance type has already been discussed in Section 1 above, under the separate elements for ANGULAR DIMENSION, LINEAR DIMENSION, POINT DIMENSION, DIAMETER DIMENSION, RADIUS DIMENSION, and ORDINATE DIMENSION.

DRAWING. A drawing is a collection of annotation entities and views. Multiple drawings can be included in a single file. Drawings have names and size; units may be specified for the drawing. A drawing closely corresponds to a CGM or to any image displayed on a graphics view surface by any of the programmer interface standards: GKS, CGI, or PHIGS.

LINE FONT DEFINITION. Two types of line fonts may be defined. One type considers a line font as a repetition of a basic pattern of visible-blanked segments superimposed upon a straight line or a curve. The other type considers a line font as a repetition of a template figure that is displayed at regularly spaced locations along a planar anchoring curve. None of the graphics standards, at present, support user-defined line types: the type 1 capability would have to be visualized by POLYLINE (in GKS/PHIGS) and DISJOINT POLYLINE (in CGI/CGM); the type two capability by using segments or structures in PHIGS/GKS/CGI and only lines, rectangles, etc. in CGM.

MACRO Capability. This facility allows the IGES specification to be extended beyond the common entity subset, utilizing a formal mechanism which is a part of the IGES Specification. This facility is available in an extremely limited fashion in the graphics standards as ESCAPE and GENERALIZED DRAWING PRIMITIVE (GDP).

PROPERTY. This facility is available in the graphics standards as APPLICATION DATA and MESSAGE. Many engineering specific properties are defined in the IGES specification. A few of them correspond to graphics elements, generally useful for rendering a picture. These are Region Restriction, Hierarchy (partial), Name, Drawing Size and Drawing Units.

SUBFIGURE Definition. All such relationships can be visualized fairly straightforwardly using PHIGS structures, somewhat more difficultly using GKS and CGI segments, and much more difficultly in CGM.

TEXT FONT Definition. This IGES facility provides for the exchange of font definitions, which are limited to a model of the motion of an imaginary pen moving between the points of an integer Cartesian "font coordinate system." None of the graphics standards provide support for "user-defined" fonts. However, such a stroke-precision text capability can easily be built on top of all the graphics standards. Compactness of representation is lost, but speed of picture generation should not be much worse.

VIEW ENTITY. This IGES entity defines a framework for specifying a viewing orientation of an object in three dimensional model space. The framework is also used to support the projection of all or part of model space onto a view plane. One type of projection, an orthographic parallel projection, can be specified. Clipping to a view volume is supported.

The 3-D graphics standards, GKS-3D and PHIGS, have a much richer viewing model. They allow a full 4x4 transformation matrix to be used, thus obtaining perspective projections and various oblique parallel projections as well.

EXTERNAL REFERENCE ENTITY. PHIGS Archive Files would directly support this element. The other standards (GKS/CGI/CGM) would have to do a lot more processing to directly support this feature.

NODAL LOAD/CONSTRAINT ENTITY. A special element to support Finite Element Modelling.

COLOR DEFINITION ENTITY. Not directly available in GKS/PHIGS, but is present in CGI/CGM as the MAXIMUM COLOR EXTENT element. In GKS/PHIGS, the application would query the workstation description tables to gather sufficient information to adjust for this entity before rendering the picture with a graphics system.

TEXT DISPLAY ENTITY. See the earlier discussion under the IGES GENERAL NOTE ENTITY. The attributes of text include height and width, font index, slant angle, rotation angle, mirror flag, and horizontal/vertical text path. All such entities can be correctly displayed by a proper setting of the graphics standards text attributes, which are common to PHIGS/GKS/CGI/CGM.

This page left intentionally blank.

IGES Text Capabilities

4.3.10 Text Font Definition Entity.

This entity defines the appearance of characters in a text font. The data describing the appearance of a character may be located by the Font Code (FC) and the ASCII character code. This entity may describe any or all the characters in a character set. Thus, this entity may be used to describe a complete font or a modification to a subset of characters in another font. Font Number and Font Name are the number and name used to reference the font on the originating system. When this entity is a modification to another font, the Supersedes Font value (Parameter 3) indicates which font the entity modifies. This value is an integer which indicates the font number to be modified or the negative of the pointer value to the directory entry of another text font definition entity. When this entity modifies another font, i.e., Parameter 3 references another font, the definitions in this entity supersede the definition in the original font. For example, a complete set of characters may have their font definition specified by this entity. Another text font definition entity could reference the first definition and modify a subset of the characters.

Each character is defined by overlaying an equally spaced square grid over the character. The character is decomposed into straight line segments which connect grid points. Grid points are referenced by standard Cartesian coordinates. The position of the character relative to the grid is defined by two points. The character's origin point is placed at the origin (0,0) of the grid and defines the position of the character relative to the text origin of that character. The second point defines the origin point of the character following the character being defined. This allows the spacing between characters to be specified. Construction of text strings consists of placing the character origin of the first character at the text string origin and placing subsequent character origins at the location specified in the previous character as the location of the next character's origin.

The parameterization of the character appearance is described by the motion of an imaginary pen moving between grid points. Commands to move the pen reference the grid location to which the pen is to move. The pen may be "lifted" such that its movement is not displayed. The representation of the movement of the pen is a sequence of pen commands and grid locations. Each movement of the pen is represented by a pen updown flag and a pair of integer grid coordinates. The pen up/down flag defaults to pen down. A flag value of 1 means the pen is to be lifted (i.e., display off) and moved to the next location in the sequence. Upon arrival at this location the pen is returned to a "down" position (i.e., display on)

The grid size is related to the text height through the scale parameter. This parameter defines how many grid units equal one text height unit.

4.3.10.1 Directory Data
ENTITY TYPE NUMBER : 310

4.3.10.2 Parameter Data

Index	Name	Type	Description
1	FC	Integer	Font Number
2	FNAME	String	Font Name
3	SF	Integer	Number of the font which this definition supersedes
4	SCALE	Integer	Number of grid units which equal one text height unit
5	N	Integer	Number of characters in this definition
6	AC1	Integer	ASCII code for first character
7	NX1	Integer	Grid location of the next character's origin

Index	Name	Type	Description
8	NY1	Integer	
9	NM1	Integer	Number of pen motions for first character
10	$PF1_1$	Integer	Pen up flag 0 = Down, 1 = Up
11	$X1_1$	Integer	Grid location to which the pen is to move
12	$Y1_1$	Integer	
.	.	.	
.	.	.	
.	.	.	
9+NM1*3	AC2	Integer	ASCII code for second character
10+NM1*3	NX2	Integer	Grid location of the next character origin
11+NM1*3	NY2	Integer	"
12+NM1*3	NM2	Integer	Number of pen motions for second character
.	.	.	
.	.	.	
$5+4N+\sum_{i=1}^{N} 3*NMi$	.	Integer	Last grid location of last character

Additional Pointers as required (see 2.2.4.4.2).

An example of this entity using the character in Figure 4-41 is

FC	1
FNAME	8H STANDARD
SF	
SCALE	8
N	60
AC1	65
NX1	11
NY1	0
NM1	4
PF1	0
X1	4
Y1	8
PF2	0
X2	8
Y2	0
PF3	1
X3	2
Y3	4
PF4	0
X4	6
Y4	4
.	
.	
.	

In the parameter section of the IGES file it would look like:

1,8HSTANDARD,,8,60,65,11,0,4,,4,8,,8,0,1,2,4,,6,4....

Figure 4-42 provides another example.

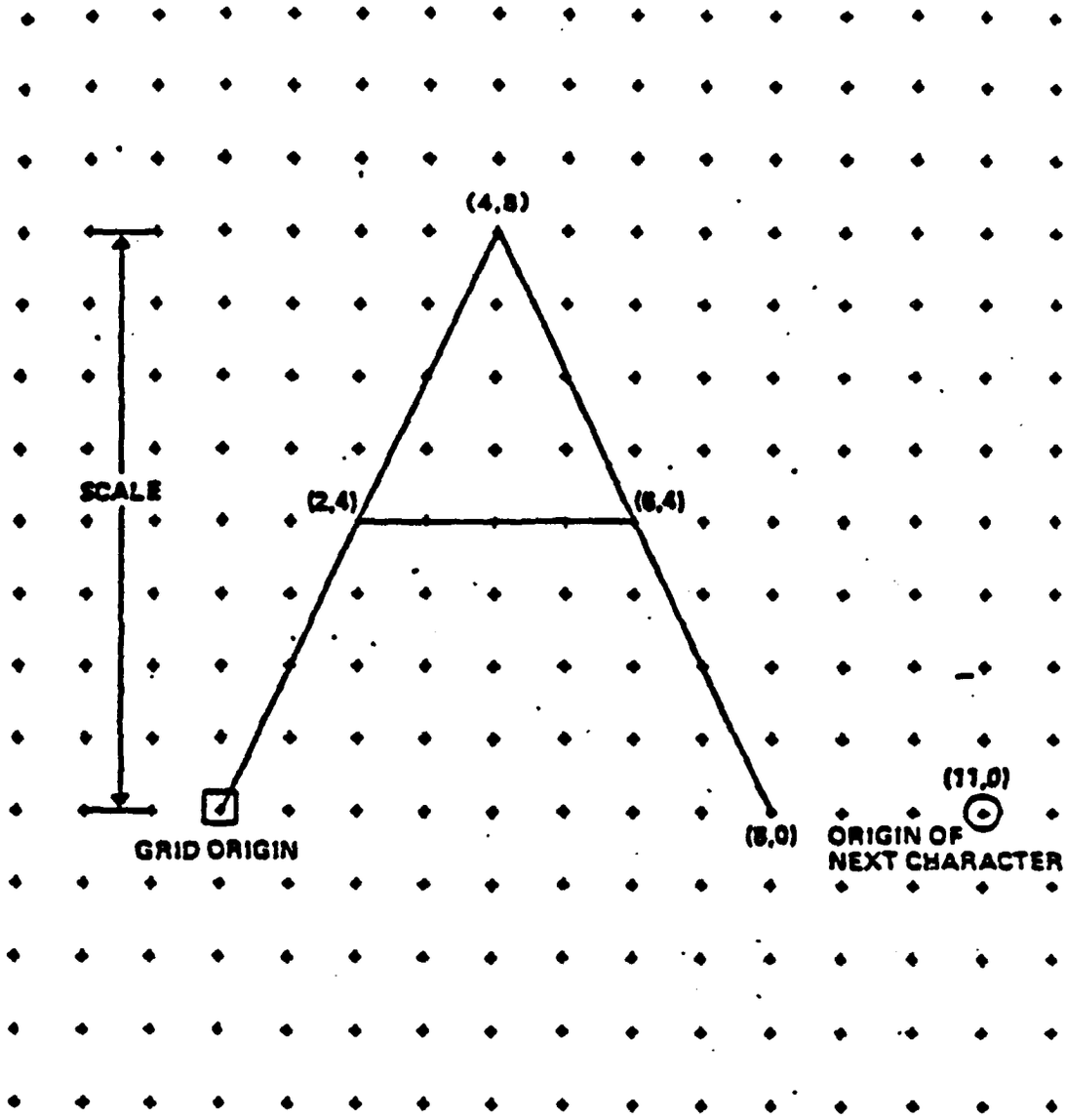


FIGURE 4-41 EXAMPLE OF A CHARACTER DEFINITION

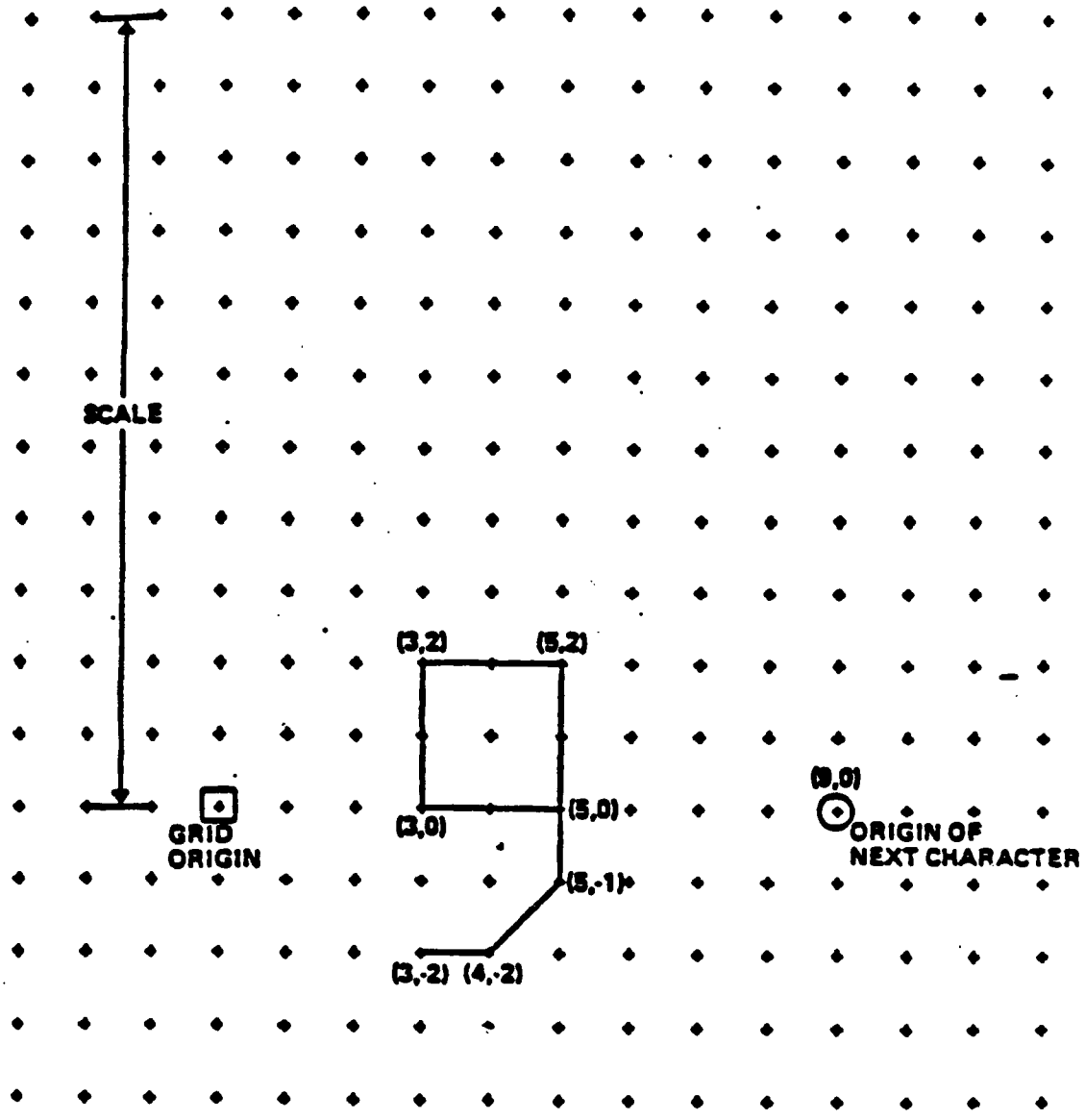


FIGURE 4-42 SECOND CHARACTER DEFINITION EXAMPLE

4.2.9 General Note Entity.

A general note entity consists of one or more text strings. Each text string contains text, a starting point, text size, and angle of rotation of the text. Examples of general notes are shown in Figure 4-9. The FC value indicates the font number and is an integer. Positive values are pre-defined fonts. Negative values point to user defined fonts or modifications to a pre-defined font.

The following fonts will be defined:

0. Symbol Font (use no longer recommended)
1. Standard Block
2. LeRoy
3. Futura
4. Fastfont
5. Calcomp
6. Comp 80
7. Micro-Film Standard
8. ISO Standard
9. DIN Standard
10. Military Standard
11. Gothic
12. News Gothic
13. Lightline Gothic
14. Simplex Roman
15. Italic
16. APL
17. Century Schoolbook
18. Helvetica

1001. Symbol Font 1
1002. Symbol Font 2

Fonts in the 1000 series display symbols mapped onto ASCII characters as shown in Figures 4 - 10 and 4 - 11. They do not specify a character display font.

GENERAL NOTE EXAMPLE

EXAMPLE 1

GENERAL NOTE ROTATED 195°

EXAMPLE 2

A GENERAL NOTE
WITH TWO LINES

EXAMPLE 3

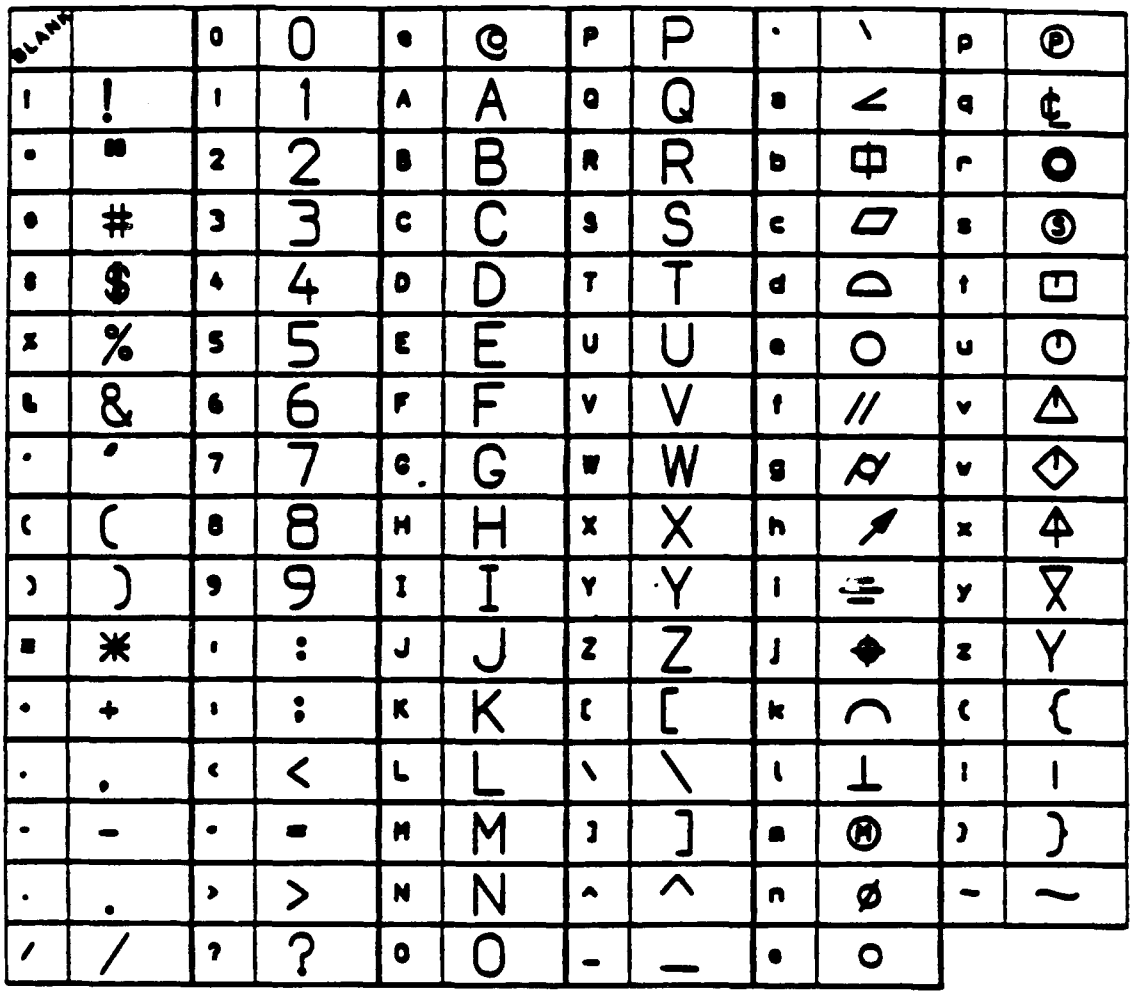
MIRRORED TEXT

EXAMPLE 4

GENERAL NOTE WITH FRAME

EXAMPLE 5

FIG. 4-9 EXAMPLES OF THE GENERAL NOTE ENTITY



FONT CODE 1001

FIGURE 4-10

BLANK		0	0	@	@	P	P	`	\	p	↑
!	!	1	1	A	A	Q	Q	a	⧖	q	↓
"	"	2	2	B	B	R	R	b	÷	r	→
#	±	3	3	C	C	S	S	c	≤	s	←
$	°	4	4	D	D	T	T	d	≥	t	φ
%	%	5	5	E	E	U	U	e	Δ	u	θ
&	&	6	6	F	F	V	V	f	√	v	Γ
'	′	7	7	G	G	W	W	g	X	w	ψ
(	(	8	8	H	H	X	X	h	≡	x	ω
)	)	9	9	I	I	Y	Y	i	≠	y	λ
*	✳	:	:	J	J	Z	Z	j	∫	z	α
+	+	;	;	K	K	[	[	k	⊃	{	δ
,	,	<	<	L	L	\	\	l	∨	|	μ
-	-	=	=	M	M	]	]	m	∧	}	Π
.	.	>	>	N	N	^	^	n	~	~	—
/	/	?	?	O	O	_	_	o	Σ		

FONT CODE 1002

FIGURE 4-11

Font 1 does not have a defined display. Use of Font 1 implies the receiving system may use any font which displays the appropriate ASCII characters. The intent of this font is for usage when the actual display of the characters is not critical for the application.

Font 0 is an old symbol font and should no longer be used. Figure 4 - 12 is a mapping symbol definition for font 0.

If the FC number is not sufficient to describe the font, a text font definition entity may be used to define the font. If a text font definition is being used, the negative of the pointer value for the directory entry of the text font definition entity is placed in the FC parameter. The use of the values WT, HT, SL, A, and text start point are shown in Figure 4-13.

Within definition space, the parameters for the text block are applied in the following order (See Figure 4-14):

1) Define the box height (HT) and box width (WT).

   The rotate internal text flag indicates whether the text box is filled with horizontal text or vertical text. The box width is measured from the text start point in the positive XT direction and the box height is measured in the positive YT direction from the text start point, before the rotation angle (A) is applied.

2) The slant angle is then applied to each individual character. For horizontal text it is measured from the XT axis in a counterclockwise direction. For vertical text the slant angle is measured from the YT axis.

3) The rotation angle is then applied to the text block. This rotation is applied in a counterclockwise direction about the text start point. The plane of rotation is the XT, YT plane at the depth ZSn (where ZSn is the value given for the text start point).

0	⧖	26	ψ	54	.	102	B	130	X	156	ø
1	÷	27	ω	55	-	103	C	131	Y	157	○
2	≤	30	λ	56	.	104	D	132	Z	160	Ⓟ
3	≥	31	α	57	/	105	E	133	[	161	¢
4	△	32	δ	60	0	106	F	134	\	162	ⓞ
5	√	33	μ	61	1	107	G	135	]	163	Ⓢ
6	×	34	π	62	2	110	H	136	^	164	⊡
7	≡	35	—	63	3	111	I	137	_	165	⏲
10	≠	36	±	64	4	112	J	140	`	166	⚠
11	∫	37	•	65	5	113	K	141	∠	167	◇
12	⊃	40		66	6	114	L	142	⊕	170	♠
13	∨	41	!	67	7	115	M	143	▱	171	⧖
14	∧	42	"	70	8	116	N	144	⌓	172	Y
15	~	43	#	71	9	117	O	145	○	173	{
16	Σ	44	$	72	:	120	P	146	//	174	|
17	↑	45	%	73	:	121	Q	147	⌀	175	}
20	↓	46	&	74	<	122	R	150	↗	176	~
21	→	47	'	75	=	123	S	151	≡	177	Z
22	←	50	(	76	>	124	T	152	◆		
23	φ	51	)	77	?	125	U	153	⌒		
24	θ	52	*	100	@	126	V	154	⊥		
25	⊤	53	+	101	A	127	W	155	Ⓜ		

FIGURE 4-12 CHARACTER SET & OCTAL CODE FOR FONT CODE ZERO

4) The mirror operation is performed next. The value 1 indicates the mirror axis is the (rotated) line perpendicular to the text base line and through the text start point. The value 2 indicates the mirror axis is the (rotated) test base line.

Finally, the Transformation Matrix entity is used to specify the relative position of definition space within model space.

The number of characters (NCn) must always be equal to the character count in its corresponding text string (TEXTn).

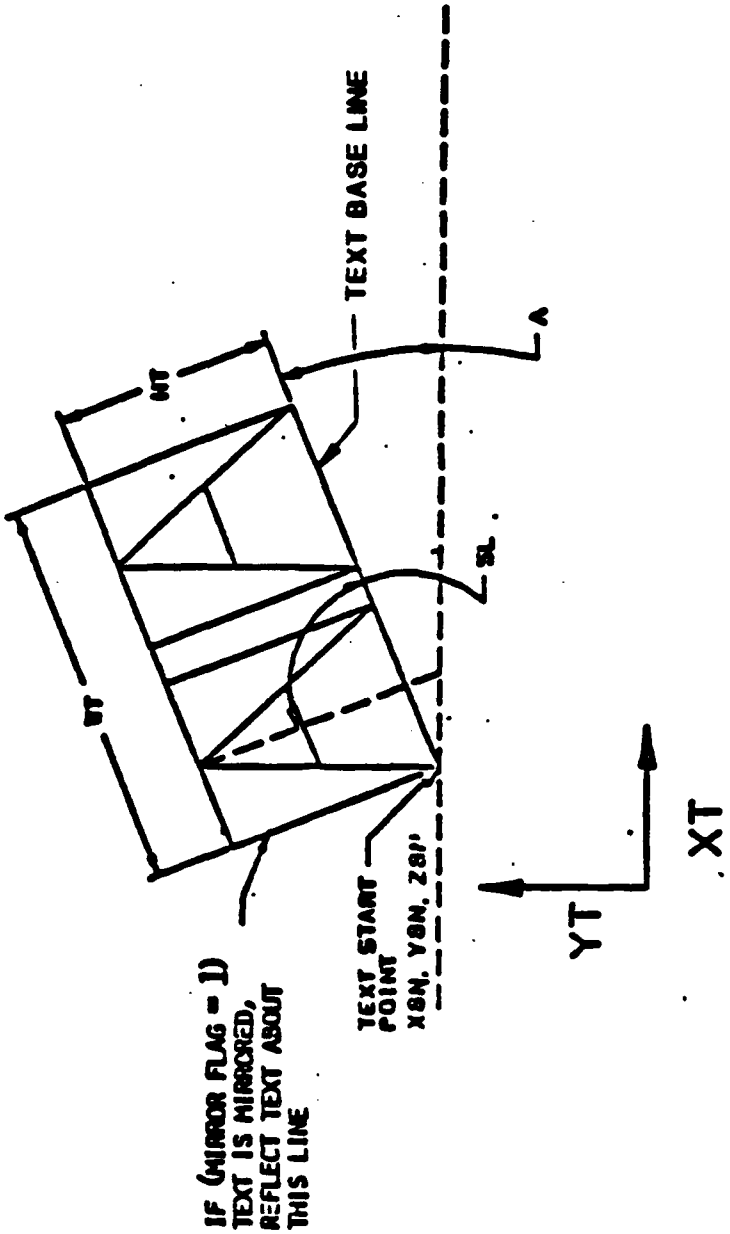


FIGURE 4-13 GENERAL NOTE TEXT CONSTRUCTION

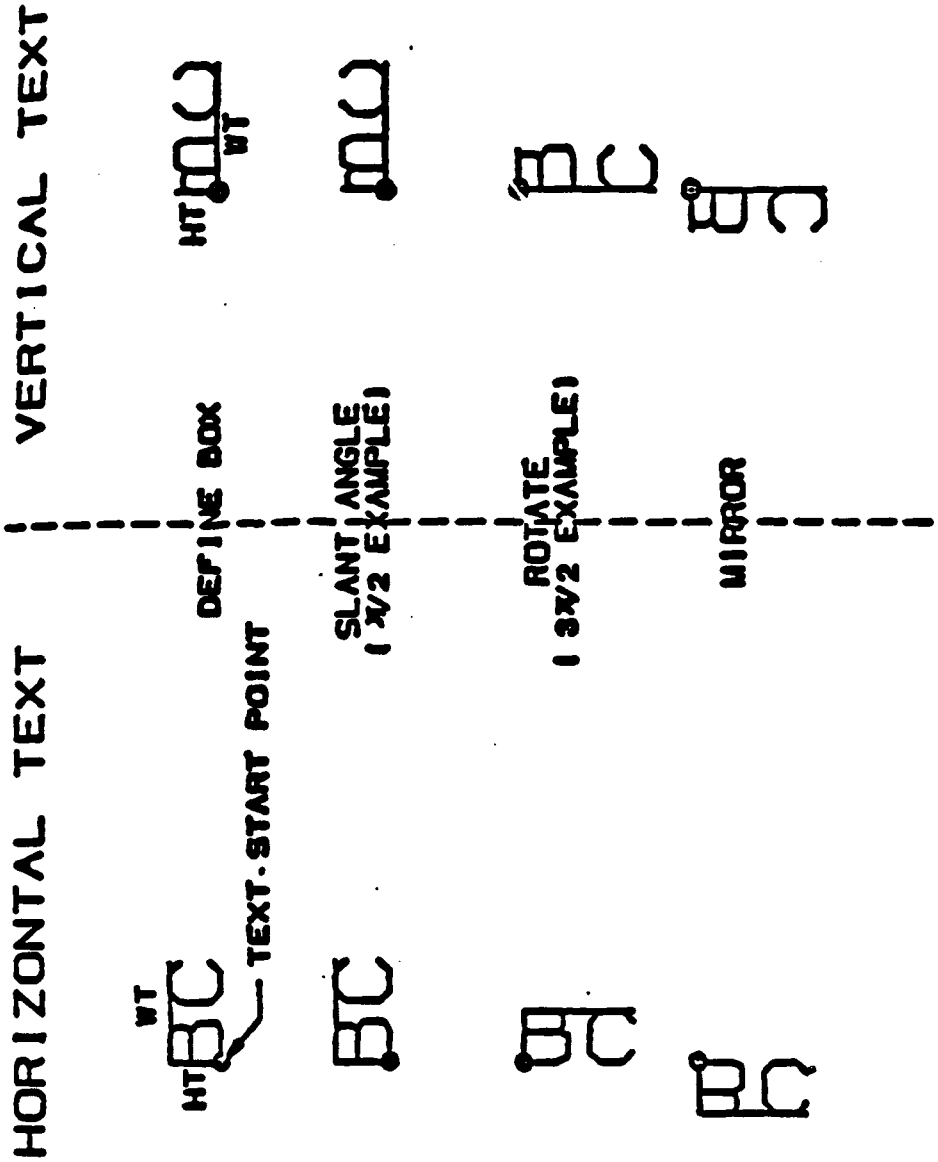


FIG. 4-14 GENERAL NOTE EXAMPLE OF TEXT OPERATIONS

4.2.9.1 The graphical representation and recreation of notes with a special structure are handled by the use of the Form Number in Field 15 of the Directory Entry for this entity. A system to accommodate these notes is outlined below. Any strings after those specified by the form number are considered additional, appended strings that are not related in any particular manner to the previously referenced strings.

4.2.9.2 In the event that a string necessary for the defined structure is not present in the originating system's note, a null string shall be inserted in the general note entity to take the place of the non-existent string to maintain the structure of the data.

4.2.9.3 Notes that contain fractional notation will be represented as mixed numerals. This is done through the use of four consecutive strings representing the whole number, the numerator, the denominator, and the divisor bar. These are examples of the divisor bar string:

1H/ 1H- 2H-- 1H__

4.2.9.4 The following form numbers for the general note are used to maintain the graphical representation of the originating system's note:

4.2.9.4.1 Form 0: Simple Note (default) - A general note of one or more strings such that a text string is not related in any manner to another string in the same general note entity.

4.2.9.4.2 Form 1: Dual Stack - A general note of two or more strings where the first two are related in a manner such that they are both left justified and the second string is displayed 'below' the first.

```
xxxxxx
yyyyy
```

4.2.9.4.3 Form 2: Imbedded Font Change - A general note of two or more strings that is intended as a single string but was divided to accommodate a font change in the string.

xxxx yyy xxxx

4.2.9.4.4 Form 3: Superscript - A general note of two or more strings where the second string is a superscript of the first string.

yyy^{xxx}

4.2.9.4.5 Form 4: Subscript - A general note of two or more strings where the second string is a subscript of the first string.

xxx_{yyy}

4.2.9.4.6 Form 5: Super-/Sub-script - A general note of three or more strings where the second string is a superscript of the first string and the third string is a subscript of the first string.

yyy^{xxx}_{zzz}

4.2.9.4.7 Form 6: Multiple Stack/Left Justified - A general note where all strings are left justified to a common margin. These strings originated as a "paragraphed" note.

xxxxxxxxxx
yyyyyyyyy
zzzzzzzzzzz

4.2.9.4.8 Form 7: Multiple Stack/Center Justified - A general note where all strings are center justified to a common axis.

xxxxxxx
yyyy
zzzzzzz

4.2.9.4.9 Form 8: Multiple Stack/Right Justified - A general note where all strings are right justified to a common margin.

```
xxxxxxxxxx
  yyyyyyy
 zzzzzzzz
```

4.2.9.4.10 Form 100: Simple Fraction - A general note of four or more strings where the first four strings define a mixed numeral as defined in 4.2.9.3.

$$xx\ \frac{yyy}{zzz}$$

4.2.9.4.11 Form 101: Dual Stack Fraction - A general note of eight or more strings which represent two mixed numerals as defined in 4.2.9.3. These mixed numerals are related such that the fifth through the eighth strings are displayed below the first through the fourth strings respectively.

$$xx\ \frac{yy}{zz}$$

$$ii\ \frac{jj}{kk}$$

4.2.9.4.12 Form 102: Imbedded Font Change/Double Fraction - This general note originated as a single string but was split to accommodate a font change for a special character in the fifth string. This is a general note of nine or more strings where the first and sixth strings represent the whole number string of a mixed numeral as defined in 4.2.9.3. The fifth string is a character (or characters) that was set apart to accommodate the font change.

$$xx\ \frac{yy}{zz}\ -\ ii\ \frac{jj}{kk}$$

4.2.9.4.13 Form 105: Super-/Subscript Fraction - A general note of twelve or more strings where the first, fifth, and ninth strings represent the whole number string of a mixed numeral as defined in 4.2.9.3. The second and third mixed numerals are the superscript and subscript respectively of the first mixed numeral.

$$\mathrm{ii}\,{}^{\mathrm{jj}}_{\mathrm{kk}}$$

$$\mathrm{xx}\,{}^{\mathrm{yy}}_{\mathrm{zz}}$$

$$\mathrm{rr}\,{}^{\mathrm{ss}}_{\mathrm{tt}}$$

4.2.9.5 Directory Data
ENTITY TYPE NUMBER : 212

4.2.9.6 Parameter Data

Index	Name	Type	Description
1	NS	Integer	Number of text strings in general note
2	NC1	Integer	Number of characters in first string (TEXT1) or zero. The number of characters (NCn) must always be equal to the character count of its corresponding text string (TEXTn)
3	WT1	Real	Box width
4	HT1	Real	Box height
5	FC	Integer or Pointer	Font characteristic (Default = 1)
6	SL1	Real	Slant angle of TEXT1 in radians (pi/2 is the value for no slant angle and is the default value)
7	A1	Real	Rotation angle in radians for TEXT1
8	M1	Integer	Mirror flag 0-no mirroring 1 - mirror axis is perpendicular to text base line 2 - mirror axis is text base line
9	VH1	Integer	Rotate internal text flag (0-text horizontal, 1-text vertical)
10	XS1	Real	First text start point
11	YS1	Real	

12	ZS1	Real	Z depth from XT, YT plane
13	TEXT1	String	First text string
14	NC2	Integer	Number of characters in second text string
1+NS*12	TEXTNS	String	Last text string
2+NS*12	NCNS	Integer	Number of characters in last text string

Additional Pointers as required (see 2.2.4.4.2)

IGES Curve Capabilities

3.2 Circular Arc Entity

A circular arc is a connected portion of a parent circle which consists of more than one point. The definition space coordinate system is always chosen so that the circular arc lies in a plane either coincident with or parallel to the XT, YT plane.

3.2.1 A circular arc determines unique arc end points and an arc center point (the center of the parent circle). By considering the arc end points to be enumerated and listed in an ordered manner, start point first, followed by terminate point, a direction with respect to definition space can be associated with the arc. The ordering of the end points corresponds to the ordering necessary for the arc to be traced out in a counterclockwise manner. This convention serves to distinguish the desired circular arc from its complementary arc (complementary with respect to the parent circle). Refer to Section 3.1.2 for information relating to use of the term counterclockwise.

3.2.2 The direction of the arc with respect to model space is determined by the original counterclockwise direction of the arc within definition space, in conjunction with the action of the transformation matrix on the arc.

3.2.3 In the event that a parameterization is required but not given, the default parameterization is

$$C(t) = (X1 + R*\cos t,\ Y1 + R*\sin t,\ ZT)$$
$$\text{for } t2 \leq t \leq t3$$

where, for i = 2 and 3,

(i) $R = \mathrm{sqrt}((Xi-X1)**2 + (Yi-Y1)**2)$

(ii) ti is such that $(R*\cos ti,\ R*\sin ti) = (Xi - X1,\ Yi - Y1)$

and

$$0 \leq t2 < 2*PI$$
$$0 \leq t3 - t2 \leq 2*PI$$

3.2.4 Examples of the circular arc entity are shown in Figure 3-1. In Example 3 of Figure 3-1, the solid arc is defined using point A as the start point and point B as the terminate point. If the complementary dashed arc were desired, the first endpoint listed in the parameter data entry would be B, and the second would be A.

3.2.5 **Directory Data**

ENTITY TYPE NUMBER : 100

3.2.6 **Parameter Data**

Index	Name	Type	Description
1	ZT	Real	Parallel ZT displacement of arc from XT, YT plane
2	X1	Real	Arc center abscissa
3	Y1	Real	Arc center ordinate
4	X2	Real	Start point abscissa
5	Y2	Real	Start point ordinate
6	X3	Real	Terminate point abscissa
7	Y3	Real	Terminate point ordinate

Additional Pointers as required (see 2.2.4.4.2).

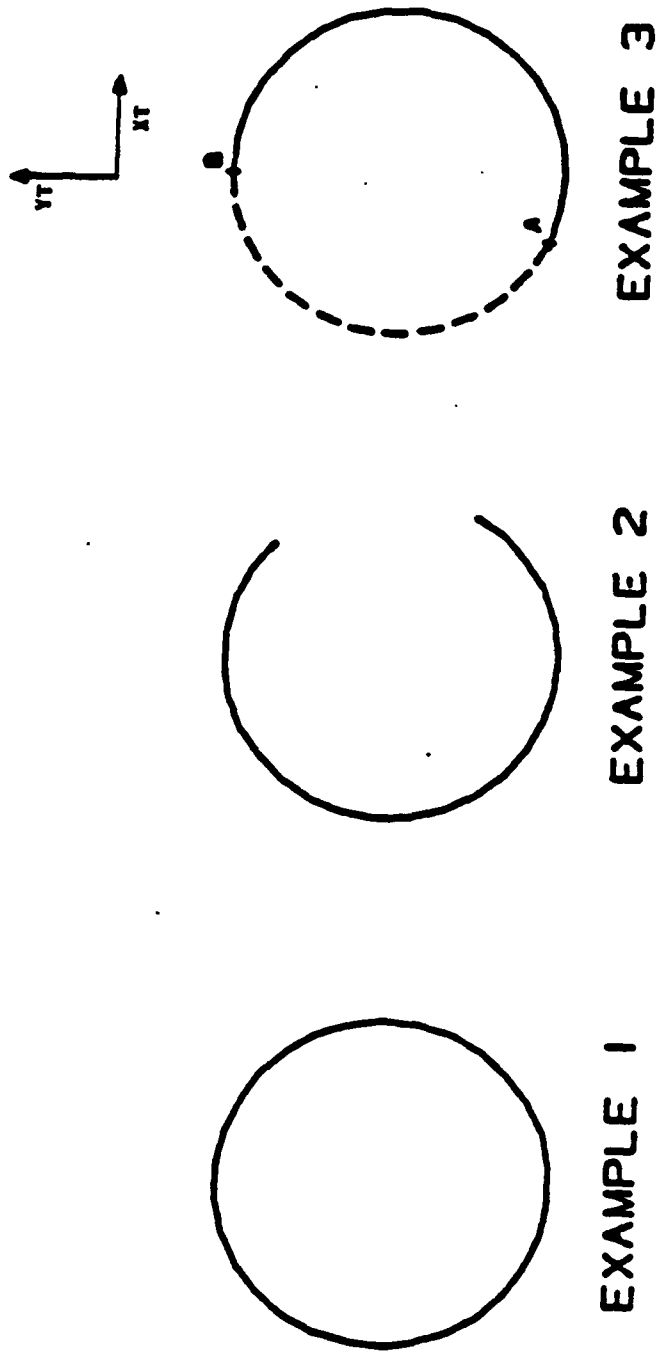


FIG. 3-1 EXAMPLES OF THE CIRCULAR ARC ENTITY

3.3 Composite Curve Entity

A composite curve is a connected curve that results from the grouping of certain individual constituent entities into a logical unit.

3.3.1 A composite curve is defined as an ordered list of entities of the following types: point, line, circular arc, conic arc, parametric spline, rational B-spline, and connect point. The list of entities appears in the parameter data entry. There, each entity to appear in the defining list is indicated by means of a pointer to the directory entry of that entity. The order within the defining list is derived from the order of the listing of these pointers.

3.3.2 Each constituent entity has its own transformation matrix and display attributes. Each constituent entity may have text or properties associated with it. Because the constituent entities are subordinate to the composite entity, the Subordinate Entity Switch (digits 3-4 in directory entry field 9) of each constituent entity should indicate a physical dependency.

3.3.3 A composite curve is a directed curve, having a start point and a terminate point. The direction of the composite curve is induced by the direction of the constituent curve entities (i.e., those constituent entities other than the point entity) in the following way: The start point for the composite curve is the start point of the first curve entity appearing in the defining list. The terminate point for the composite curve is the terminate point of the last curve entity appearing in the defining list. Within the defining list itself, the terminate point of each constituent curve entity has the same coordinates as the start point of the succeeding curve entity.

3.3.4 The point and connect point entities are included as allowable entity types so that properties or general notes can be attached to either the start point or the terminate point of any constituent curve entities in the defining list.

A logical connection relationship can be indicated by having two composite curves or a composite curve and a network subfigure reference the connect point entity. For the special case of the logical connection of a connect point on one subfigure instance to a connect point on another subfigure instance a

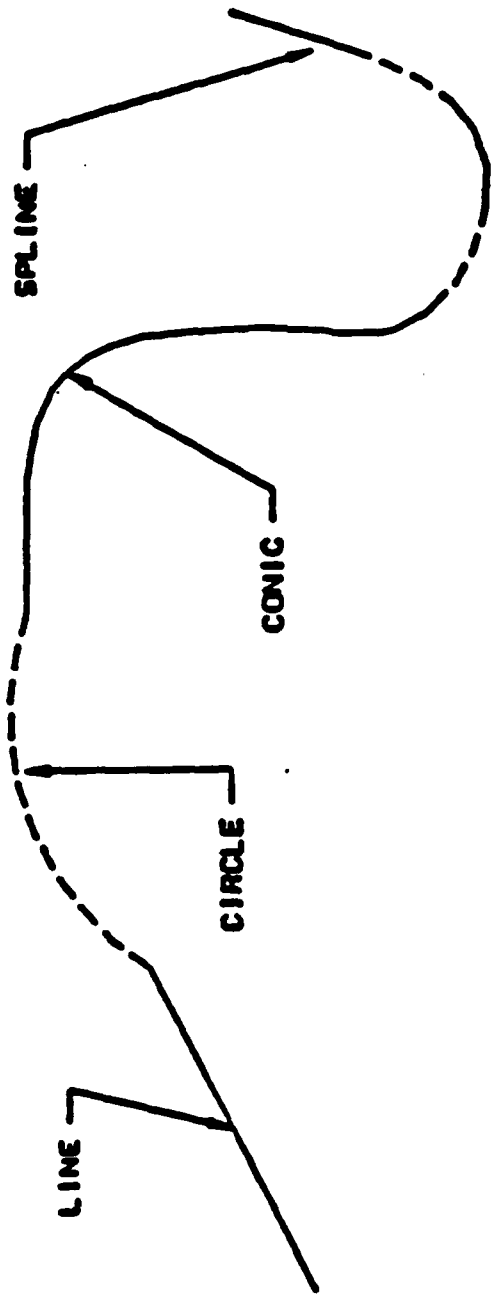


FIG. 3-2 EXAMPLE OF THE COMPOSITE CURVE ENTITY

composite curve is allowed whose list contains only two connect point entities with no intervening curve entity. There are certain restrictions regarding the use of the point entity in a composite entity. They are:

a. Two point or connect point entities cannot appear consecutively in the defining list unless they are the only entities in the composite curve.

b. If a point or connect point entity and a curve entity are adjacent in the defining list, then the coordinates of the point or connect point entity must agree with the coordinates of the terminate point of the curve entity whenever the curve entity precedes the point or connect point entity, and must agree with the coordinates of the start point of the curve entity whenever the curve entity follows the point or connect point entity.

c. A composite curve cannot consist of a point entity alone or a single connect point.

3.3.5 In the event that a parametrization is required but not given, the default parametrization of the composite curve is obtained from the parametrization of the constituent curves as defined below. As point and connect point entities do not contribute to the parametrization of a composite curve, they are not considered in the definition below.

Let

C	be the composite curve;
N	be the number of constituent curves ($N \geq 1$);
CC(i)	be the i-th constituent curve, for each i such that $1 \leq i \leq N$;
PS(i)	be the parametric value of the start of CC(i);
PE(i)	be the parametric value of the end of CC(i);
T(0)	be 0.0;
T(i)	be the sum from j=1 to j=i of (PE(j) - PS(j)), for each i such that $1 \leq i \leq N$.

Then

(1) the parametric values of C range from T(0) to T(N) ; and

(2) C(u) = CC(i) (u - T(i-1) + PS (i)) where u is a parametric value such that $T(i-1) \leq u \leq T(i)$.

A composite curve consisting solely of point and/or connect point entities, will not be given a parametrization.

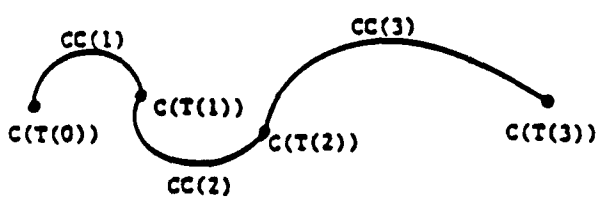
3.3.6 In this section, an example of a parametrization of a composite curve entity is given.

Let N=3 and for each i such that $1 \leq i \leq 3$, let CC(i) be the i-th constituent curve of the composite curve C. Assume the parametric values of the start and end points of each CC(i) are given by the table

i	PS(i)	PE(i)
1	0.0	0.4
2	3.3	3.5
3	0.0	0.3

Then T(0) = 0.0, T(1) = 0.4, T(2) = 0.6, T(3) = 0.9, and the composite curve C, is defined from 0.0 to 0.9.

This situation described is illustrated by



The curve combining CC(1), CC(2), and CC(3) represents the composite curve C.

3.3.7 An example of a composite curve entity is shown in Figure 3-2

3.3.8 **Directory Data**

ENTITY TYPE NUMBER : 102

3.3.9 **Parameter Data**

Index	Name	Type	Description
1	N	Integer	Number of entities
2	DE	Pointer	Pointers to directory entries for the constituent entities
.	.	.	
.	.	.	
.	.	.	
N+1	DE	Pointer	

Additional Pointers as required (see 2.2.4.4.2).

3.4 Conic Arc Entity

A conic arc is a bounded connected portion of a parent conic curve which consists of more than one point. The parent conic curve is either an ellipse, a parabola, or a hyperbola. The definition space coordinate system is always chosen so that the conic arc lies in a plane either coincident with or parallel to the XT, YT plane. Within such a plane, a conic is defined by the six coefficients in the following equation.

$$A*XT^2 + B*XT*YT + C*YT^2 + D*XT + E*YT + F = 0$$

3.4.1 Each coefficient is a real number. The definitions of ellipse, parabola, and hyperbola in terms of these six coefficients are given below.

3.4.2 A conic arc determines unique arc endpoints. A conic arc is defined within definition space by the six coefficients above and the two endpoints. By considering the conic arc endpoints to be enumerated and listed in an ordered manner, start point followed by terminate point, a direction with respect to definition space can be associated with the arc. In order for the desired elliptical arc to be distinguished from its complementary elliptical arc, the direction of the desired elliptical arc must be counterclockwise. In the case of a parabola or hyperbola, the parameters given in the parameter data section uniquely define a portion of the parabola or a portion of a branch of the hyperbola; therefore, the concept of a counterclockwise direction is not applied. (Refer to Section 3.1.2 for information concerning use of the term "counterclockwise".)

3.4.3 The direction of the conic arc with respect to model space is determined by the original direction of the arc within definition space, in conjunction with the action of the transformation matrix on the arc.

3.4.4 The definitions of the terms ellipse, parabola, and hyperbola are given in terms of the quantities Q1, Q2, and Q3. These quantities are:

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

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

$$Q3 = A + C$$

3.4.5 A parent conic curve is

An ellipse if Q2>0 and Q1 * Q3<0.

A hyperbola if Q2<0 and Q1 ≠ 0.

A parabola if Q2 = 0 and Q1 ≠ 0.

An example of each type of conic arc is shown in Figure 3-3.

3.4.6 Those entities which can be represented as various degenerate forms of a conic equation (Point and Line) must not be put into the Entity Type 104; more appropriate Entity Types exist for these forms.

Because of the numerical sensitivity of the implicit form of the conic description, a receiving system not using that form as its internal representation for conics need not be expected to correctly process conics in this form unless they are put into a standard position in definition space. A conic arc entity is said to be in a standard position in definition space provided each of its axes is parallel to either the XT axis or YT axis and provided it is centered about the ZT axis. For a parabola, use the vertex as the origin. The conic is moved from this position in definition space to the desired position in space with a transformation matrix (Entity type 124).

The form number is regarded as purely informational by such a postprocessor.

Further details may be found in Appendix E.

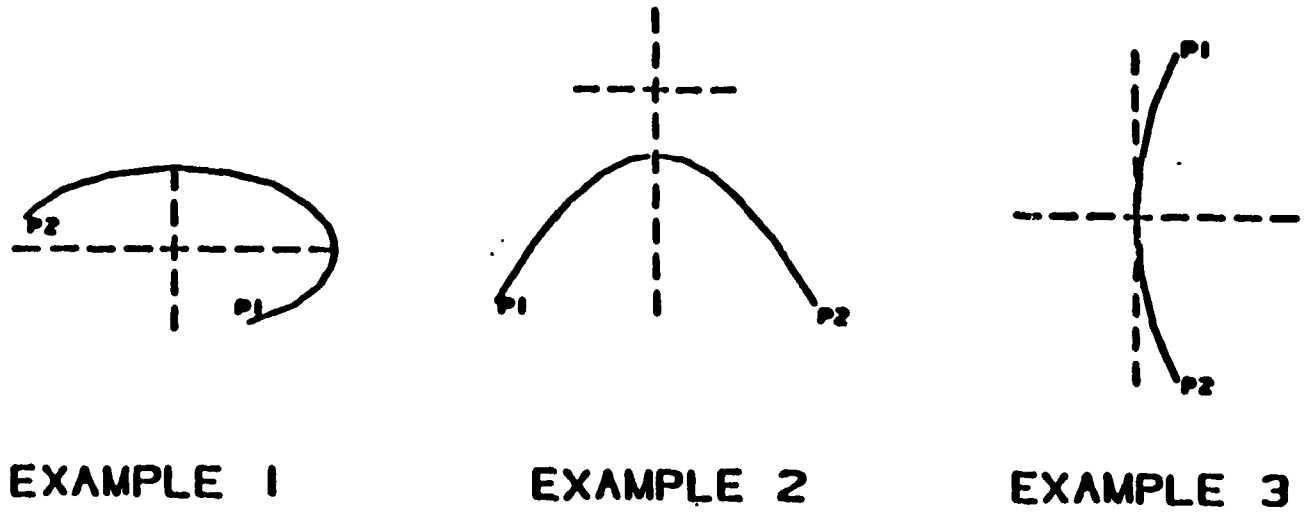


FIG. 3-3 EXAMPLES OF THE CONIC ARC ENTITY

3.4.7 In the event that a parameterization is required but not given, the default parameterization is:

Parabola

case A and E $\neq$ 0.0

if X1 < X2
C(t) = (t, -(A/E)*t**2, ZT) for t1 $\leq$ t $\leq$ t2
where, for i = 1 and 2, ti = Xi.

if X2 < X1
C(t) = (-t, -(A/E)*t**2, ZT) for t1 $\leq$ t $\leq$ t2
where, for i = 1 and 2, ti = -Xi.

case C and D $\neq$ 0.0

if Y1 < Y2
C(t) = (-(C/D)*t**((2, t, ZT) for t1 $\leq$ t $\leq$ t2
where, for i = 1 and 2, ti = Yi.

if Y2 < Y1
C(t) = (-(C/D)*t**2, -t, ZT) for t1 $\leq$ t $\leq$ t2
where, for i = 1 and 2, ti = -Yi.

Ellipse

C(t) = (a*cos t, b*sin t, ZT)

for t1 $\leq$ t $\leq$ t2

where
a = sqrt(-F/A)
b = sqrt(-F/C)

and, for i = 1 and 2, ti is such that
(i) (a*cos ti, b*sin ti, ZT) = (Xi, Yi, ZT)
(ii) 0 $\leq$ t1 $\leq$ 2*PI
(iii) 0 $\leq$ t2 - t1 $\leq$ 2*PI

Hyperbola

case F*A < 0.0 and F*C > 0.0

let
a = sqrt(-F/A)
b = sqrt(F/C)
and, for i = 1,2
ti is such that
(i) (a*sec ti, b*tan ti, ZT) = (Xi,Yi,ZT)
(ii) - PI/2 < t1,t2 < PI/2

if $t1 < t2$

$C(t) = (a*\sec t,\ b*\tan t,\ ZT)$ for $1 \leq t \leq t2$

if $t2 < t1$

$C(t) = (a*\sec(-t),\ b*\tan(-t),\ ZT)$ for $-t1 \leq t \leq -t2$

case $F*A > 0.0$ and $F*C < 0.0$

let

$a = \mathrm{sqrt}(F/A)$
$b = \mathrm{sqrt}(-F/C)$
and, for $i = 1,2$
ti is such that
(i) $(a*\tan ti,\ b*\sec ti,\ ZT) = (Xi, Yi, ZT)$
(ii) $-PI/2 < t1, t2 < PI/2$

if $t1 < t2$

$C(t) = (a*\tan t,\ b*\sec t,\ ZT)$ for $t1 \leq t \leq t2$

if $t2 < t1$

$C(t) = (a*\tan(-t),\ b*\sec(-t),\ ZT)$ for $-t1 \leq t \leq -t2$

3.4.8 Field 15 of the directory entry accommodates a Form Number. For this entity, the options are as follows:

FORM	Meaning
0	Form of parent conic curve must be determined from the general equation.
1	Parent conic curve is an ellipse (See example 1, Figure 3-3).
2	Parent conic curve is a hyperbola (See example 2, Figure 3-3).
3	Parent conic curve is a parabola (See example 3, Figure 3-3).

3.4.9 **Directory Data**

ENTITY TYPE NUMBER : 104

3.4.10 **Parameter Data**

Index	Name	Type	Description
1	A	Real	Conic Coefficient
2	B	Real	Conic Coefficient
3	C	Real	Conic Coefficient
4	D	Real	Conic Coefficient
5	E	Real	Conic Coefficient
6	F	Real	Conic Coefficient
7	ZT	Real	ZT Coordinate of plane of definition
8	X1	Real	Start Point Abscissa
9	Y1	Real	Start Point Ordinate
10	X2	Real	Terminate Point Abscissa
11	Y2	Real	Terminate Point Ordinate

Additional Pointers as required (see 2.2.4.4.2).

3.5 Copious Data Entity

This entity stores data points in the form of pairs, triples, or sextuples. An interpretation flag value signifies which of these forms is being used. This value is one of the parameter data entries. The interpretation flag is abbreviated below by the letters IP.

Data points within definition space which lie within a single plane are specified in the form of XT, YT coordinate pairs. In this case, the common ZT value is also needed. Data points arbitrarily located within definition space are specified in the form of XT, YT, ZT coordinate triples. Data points within definition space which have an associated vector are specified in the form of sextuples; the XT, YT, ZT coordinates are specified first, followed by the i, j, k coordinates of the vector associated with the point. (Note that, for an associated vector, no special meaning is implicit.)

Field 15 of the directory entry accommodates a Form Number. For this entity, the options are as follows:

FORM	Meaning
1	Data points in the form of coordinate pairs. All data points lie in a plane ZT= constant. (IP=1)
2	Data points in the form of coordinate triples. (IP=2)
3	Data points in the form of sextuples. (IP=3)
11	Data points in the form of coordinate pairs which represent the vertices of a planar, piecewise linear curve (piecewise linear string is sometimes used). All data points lie in a plane ZT=constant. (IP=1)
12	Data points in the form of coordinate triples which represent the vertices of a piecewise linear curve (piecewise linear string is sometimes used). (IP=2)
13	Data points in the form of sextuples. The first triple of each sextuple represents the vertices of a piecewise linear curve (piecewise linear string is sometimes used). The second triple is an associated vector. (IP=3)
20	Centerline Entity through points (IP=1)

21	Centerline Entity through circle centers (IP=1)
31	Section Entity Form 31 (IP=1)
32	Section Entity Form 32 (IP=1)
33	Section Entity Form 33 (IP=1)
34	Section Entity Form 34 (IP=1)
35	Section Entity Form 35 (IP=1)
36	Section Entity Form 36 (IP=1)
37	Section Entity Form 37 (IP=1)
38	Section Entity Form 38 (IP=1)
40	Witness Line Entity (IP=1)
63	Simple Closed Area Entity (IP=1)

The linear path is an ordered set of points in either 2- or 3-dimensional space. These points define a series of linear segments along the consecutive points of the path. The segments may cross or be coincident with each other. Paths may close, i.e., the first path point may be identical to the last.

The linear path is implemented as two forms of the copious data block (entity number 106). Form 11 is for 2-dimensional paths and form 12 is for 3-dimensional paths. This entity will be closely associated with properties indicating functionality and fabrication parameters, such as Line Widening.

Refer to the centerline and witness line entities in Section 4 of this specification for examples of Form Numbers 20, 21 and 40. Each of these annotation entities contains a description of how the associated copious data are to be interpreted. Forms 31-38 provide for the transfer of graphical information and are defined here for compatability with previous versions of the specification. The Sectioned Area Entity (type 230) provides a more compact method for transferring this information.

A simple closed area is a bounded region of XY coordinate space represented by a set of points that forms a series of connected linear segments. These segments must form a closed loop, i.e., the first point of the boundary of the area and the last point must be identical. No segments of this entity are allowed to intersect or be coincident except for the closing of the entity at the initial and final

points. This entity will be closely related to properties that indicate functionality of closed regions, such as Region Fill and Region Restriction.

The area is implemented as Form 63 of entity 106, the copious data block.

3.5.1 Directory Data

ENTITY TYPE NUMBER : 106

3.5.2 Parameter Data

Index	Name	Type	Description
1	IP	Integer	Interpretation Flag IP=1 x,y pairs, common z IP=2 x,y,z coordinates IP=3 x,y,z coordinates and i,j,k vectors
2	N1	Integer	Number of n-tuples
For IP=1 (x,y pairs, common z):			
3	ZT	Real	Common z displacement
4	X1	Real	First data point abecissa
5	Y1	Real	First data point ordinate
.	.	.	.
.	.	.	.
.	.	.	.
3+2N	YN	Real	Last data point ordinate
For IP=2 (x,y,z triples):			
3	X1	Real	First data point x value
4	Y1	Real	First data point y value
5	Z1	Real	First data point z value
.	.	.	.
.	.	.	.
.	.	.	.
2+3N	ZN	Real	Last data point z value

For IP=3 (x,y,z,i,j,k sextuples):

3	X1	Real	First data point x value
4	Y1	Real	First data point y value
5	Z1	Real	First data point z value
6	I1	Real	First data point i value
7	J1	Real	First data point j value
8	K1	Real	First data point k value
.	.	.	.
.	.	.	.
.	.	.	.
2+6N	KN	Real	Last data point k value

Additional pointers as required (see sec. 2.2.4.4.2).

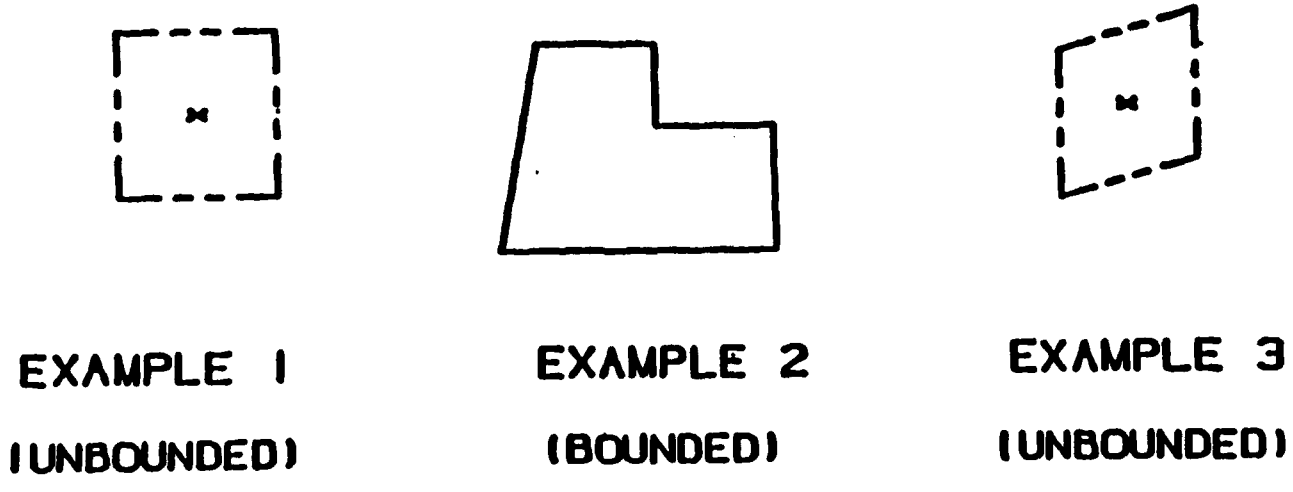


FIG. 3-4 EXAMPLES OF THE PLANE ENTITY

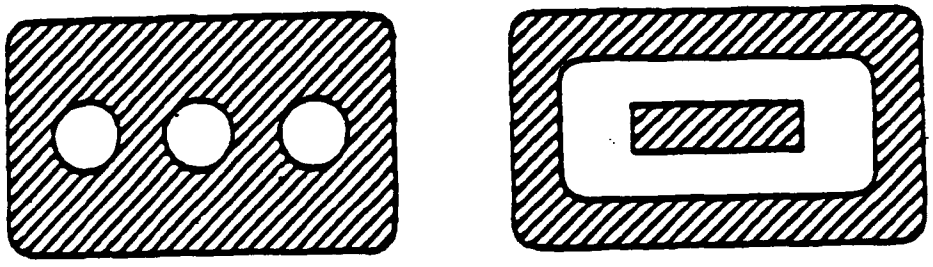


FIG. 3-5 SINGLE PARENT ASSOCIATIVITY AS USED WITH A COLLECTION OF BOUNDED PLANES

3.8 Parametric Spline Curve Entity

(Consult Appendix D for additional mathematical details)

The parametric spline curve is a sequence of parametric polynomial segments. The CTYPE value in Parameter 1 indicates the type of curve as it was represented in the sending (pre-processing) system before conversion to this entity.

3.8.1 The N polynomial segments are delimited by the breakpoints T(1), T(2), ...,T(N+1). The coordinates of the points in the i-th segment of the curve are given by the following cubic polynomials (the coefficients D, or C and D will be zero if the polynomials are of degrees 2 or 1, respectively):

$$X(u)=AX(i)+BX(i)s+CX(i)s^2+DX(i)s^3$$
$$Y(u)=AY(i)+BY(i)s+CY(i)s^2+DY(i)s^3$$
$$Z(u)=AZ(i)+BZ(i)s+CZ(i)s^2+DZ(i)s^3$$

where

$$T(i)\leq u\leq T(i+1), i=1,\ldots,N$$
$$s=u-T(i)$$

In order to avoid degeneracy, for each i at least one of the nine real coefficients, BX(i), CX(i), DX(i), BY(i), CY(i), DY(i), BZ(i), CZ(i), and DZ(i) must be non-zero.

3.8.2 If the spline is planar, it must be parametrized in terms of the X and Y polynomials only. The Z polynomial will then be zero except for each i, the AZ(i) term which indicates the Z-depth in definition space.

3.8.3 The parameter H is used as an indicator of the smoothness of the curve. If H=0, the curve is continuous at all breakpoints. If H=1, the curve is continuous and has slope continuity (see section 6.3 of FAUX79) at all breakpoints. If H=2, the curve is continuous and has both slope and curvature continuity at all breakpoints (see section 6.3 of Faux79).

3.8.4 To enable determination of the terminate point and derivatives without computing the polynomials, the Nth polynomials and their derivatives are evaluated at u = T(N+1). These data are divided by appropriate factorials and stored following the polynomial coefficients. For example, the name TPY3 will be used to designate 1/3! times the third derivative of the Y polynomial for the Nth segment evaluated at u=T(N+1), the parameter value corresponding to the terminate point. Note that these data are redundant as they are derived from the data defining the Nth polynomial segment.

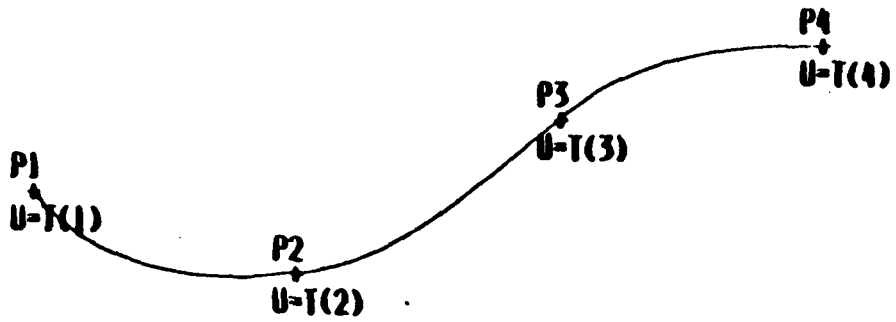
3.8.5 An example of a parametric spline is shown in Figure 3-7. Additional examples are shown in Figure 3-8.

3.8.6 Directory Data
ENTITY TYPE NUMBER : 112

3.8.7 Parameter Data

Index	Name	Type	Description
1	CTYPE	Integer	Spline Type (1=Linear 2=Quadratic 3=Cubic 4=Wilson-Fowler 5=Modified Wilson-Fowler 6=B Spline)
2	H	Integer	Degree of continuity with respect to arc length
3	NDIM	Integer	2=planar 3=non-planar
4	N	Integer	Number of segments
5	T(1)	Real	Break points of piecewise polynomial
.	.		
.	.		
.	.		
5+N	T(N+1)		

CURVE = (X(U), Y(U), Z(U)), FOR T(1) ≤ U ≤ T(N+1)
N = 3 SEGMENTS



P1 = (AX(1), AY(1), AZ(1))
P2 = (AX(2), AY(2), AZ(2))
P3 = (AX(3), AY(3), AZ(3))
P4 = TP0 = (TPX0, TPY0, TPZ0)
FIRST DERIVATIVE AT P4 = TP1 = (TPX1, TPY1, TPZ1)

FOR SEGMENT NUMBER 2:

$$X(U) = AX(2) + BX(2)*(U-T(2)) + CX(2)*(U-T(2))^2 + DX(2)*(U-T(2))^3$$
$$Y(U) = AY(2) + BY(2)*(U-T(2)) + CY(2)*(U-T(2))^2 + DY(2)*(U-T(2))^3$$
$$Z(U) = AZ(2) + BZ(2)*(U-T(2)) + CZ(2)*(U-T(2))^2 + DZ(2)*(U-T(2))^3$$

FIGURE 3-7 EXAMPLE OF THE PARAMETRIC SPLINE CURVE ENTITY

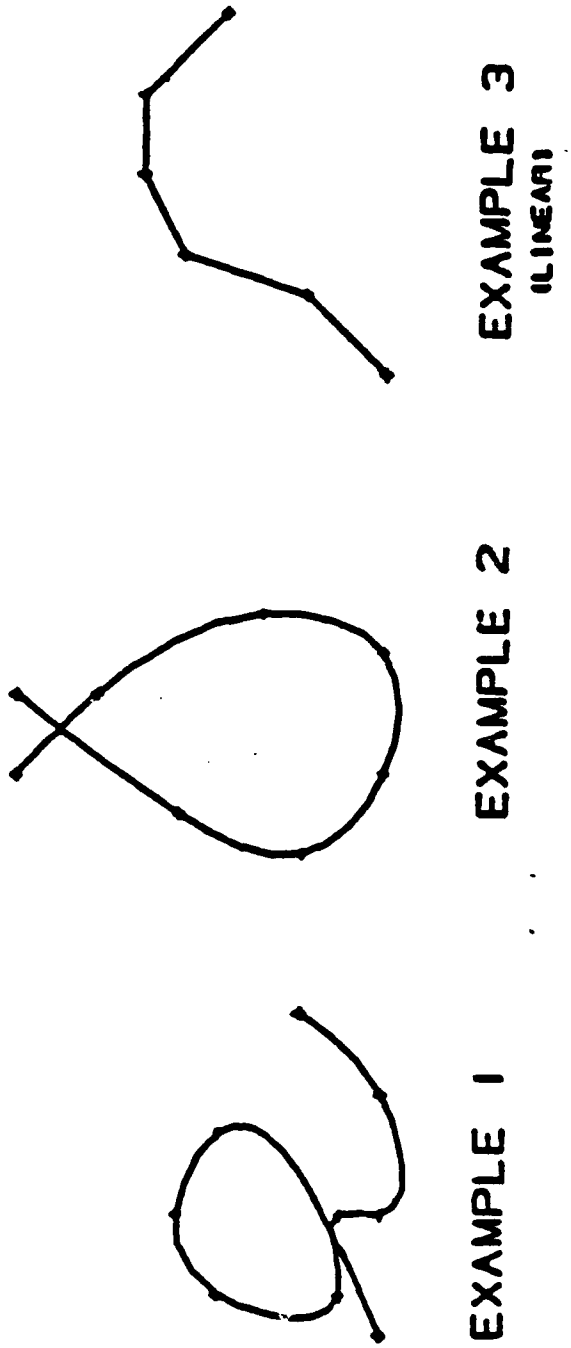


FIG. 3-8 EXAMPLES OF PARAMETRIC SPLINE CURVE ENTITY

Index	Name	Type	Description
6+N	AX(1)	Real	X coordinate polynomial
7+N	BX(1)		
8+N	CX(1)		
9+N	DX(1)		
10+N	AY(1)		Y coordinate
11+N	BY(1)		polynomial
12+N	CY(1)		
13+N	DY(1)		
14+N	AZ(1)		Z coordinate
15+N	BZ(1)		polynomial
16+N	CZ(1)		
17+N	DZ(1)		
	.		Subsequent X, Y, Z polynomials concluding with the twelve coefficients of the Nth polynomial segment.
	.		
	.		

(The parameters that follow comprise the evaluations of the polynomials of the Nth segment and their derivatives at the parameter value u=T(N+1) corresponding to the terminate point. Subsequently these evaluations are divided by appropriate factorials.)

6+13*N	TPX0	Real	X value
	TPX1		X first derivative
	TPX2		X second derivative/2!
	TPX3		X third derivative/3!
	TPY0		Y value
	TPY1		
	TPY2		
	TPY3		
	TPZ0		Z value
	TPZ1		
	TPZ2		
	TPZ3		

Additional Pointers as required (see 2.2.4.4.2)

Software to convert between parametric spline curves or surfaces and the corresponding rational B-spline curves or surfaces is available from the IGES office at the National Bureau of Standards. Materials provided include a magnetic tape of Pascal source code, a listing of the code, and accompanying documentation.

3.16 Rational B-Spline Curve Entity

The rational B-spline curve may represent analytic curves of general interest. This information is important to both the sending and receiving systems. The directory entry form number parameter is provided to communicate this information. It should be emphasized that use of this curve form should be restricted to communications between systems operating directly on rational B-spline curves and not used as a replacement for the analytic forms for communication. For a brief description of a rational B-spline curves, see Section 4 of Appendix D.

If the rational B-spline curve represents a preferred curve type, the form number corresponds to the most preferred type. The preference order is from 1 through 5 followed by 0. For example, if the curve is a circle or circular arc, the form number is set to 2. If the curve is an ellipse with unequal major and minor axis lengths, the form number is set to 3. If the curve is not one of the preferred types, the form number is set to 0.

If the curve lies entirely within a unique plane, the planar flag (PROP1) is set to 1, otherwise it is set to 0. If it is set to 1, the plane normal (parameters 18+A+4K through 16+A+4K) contain a unit vector normal to the plane containing the curve. These fields exist but are ignored if the curve is non-planar.

If the beginning and ending points on the curve are identical, PROP2 is set to 1. If they are not equal, PROP2 is set to 0.

If the curve is rational (does not have all weights equal), PROP3 is set to 0. If all weights are equal to each other, the curve is polynomial and PROP3 is set to 1. The curve is polynomial since in this case all weights cancel and the denominator sums to one (see Appendix D4).

If the curve is periodic with respect to its parametric variable, set PROP4 to 1, otherwise set PROP4 to 0.

3.16.1 Directory Data

ENTITY TYPE NUMBER: 126

Form	Meaning
0	Form of curve must be determined from the rational B-spline parameters.
1	Line
2	Circular arc
3	Elliptical arc
4	Parabolic arc
5	Hyperbolic arc

3.16.2 Parameter Data

Index	Name	Type	Description
1	K	Integer	Upper index of sum. See Appendix D
2	M	Integer	Degree of basis functions
3	PROP1	Integer	=0 - non-planar =1 - planar
4	PROP2	Integer	=0 - open curve =1 - closed curve
5	PROP3	Integer	=0 - rational =1 - polynomial
6	PROP4	Integer	=0 - non-periodic =1 - periodic

Let N=K-M+1 and let A=N+2M

7	T(-M)	Real	Knot Sequence
.	.		
.	.		
.	.		
7+A	T(N+M)		
8+A	W(0)	Real	Weights
.	.		
.	.		
.	.		
8+A+K	W(K)		
9+A+K	XO	Real	Control Points
10+A+K	YO		
11+A+K	ZO		
.	.		
.	.		
.	.		
9+A+4K	XK		
10+A+4K	YK		
11+A+4K	ZK		
12+A+4K	V(0)	Real	Starting parameter value
13+A+4K	V(1)	Real	Ending parameter value
14+A+4K	XNORM	Real	Unit Normal (if curve is planar)
15+A+4K	YNORM		
16+A+4K	ZNORM		

Additional Pointers as required (see 2.2.4.4.2).

Software to convert between parametric spline curves or surfaces and the corresponding rational B-spline curves or surfaces is available from the IGES office at the National Bureau of Standards. Materials provided include a magnetic tape of Pascal source code, a listing of the code, and accompanying documentation.

Appendix D. CGM Extracts

5.8.10 GENERALIZED DRAWING PRIMITIVE (GDP)

Parameters:

identifier (I)
point list (nP)
data record (D)

Description:

A Generalized Drawing Primitive (GDP) of the type specified by the identifier is generated on the basis of the given points and the data record.

Non-negative values of the identifier are reserved for registration and future standardization, and negative values are available for private use.

The appearance of the GDP is determined by zero or more of the attribute sets of the standardized graphical primitive elements, depending on the particular GDP. The parameters of the GDP are interpreted and utilized in an interpreter dependent manner.

NOTE - GDP provides convenient access to non-standardized graphical primitives that a device may support. GDP is similar to ESCAPE in this sense, but GDP provides a mechanism for handling of coordinate data whereas ESCAPE does not. GDP is thus preferable for generating graphical output, and ESCAPE is designed for such applications as non-standardized control functions.

When registration of GDPs occurs there may be registered GDPs which correspond with some of the standardized metafile graphical primitive elements, e.g., CIRCLE.

GDP identifiers are registered in the ISO International Register of Graphical Items, which is maintained by the Registration Authority. When a GDP identifier has been approved by the ISO Working Group on Computer Graphics, the GDP identifier value will be assigned by the Registration Authority.

References:

4.6

5.8 Escape Elements

5.8.1 ESCAPE

Parameters:

function identifier (I)
data record (D)

Description:

ESCAPE provides access to device capabilities not specified by this Standard. The function identifier parameter specifies the particular escape function. Non-negative values are reserved for registration and future standardization, and negative values are available for implementation dependent use.

NOTE - This element has been deliberately underspecified. Software making use of the ESCAPE element is less portable.

ESCAPE is designed for access to non-standardized control features of graphics devices, as opposed to non-standardized geometric primitives. The GENERALIZED DRAWING PRIMITIVE element is designed for specification of non-standardized primitives.

Function identifiers are registered in the ISO International Register of Graphical Items, which is maintained by the Registration Authority. When a function identifier has been approved by the ISO Working Group on Computer Graphics, the function identifier value will be assigned by the Registration Authority.

References:

4.8

5.9 External Elements

5.9.1 MESSAGE

Parameters:

action required flag (one of no action, action) (E)
text (S)

Description:

The MESSAGE element specifies a string of characters used to communicate information to operators at Metafile interpretation time through a path separate from normal graphical output.

If the action required flag parameter is action, the metafile interpreter may need to pause to wait for an operator response. Because the message and an associated pause may be directed at a particular device, only the interpreter may determine if a pause is appropriate. Character set selection for the text parameter is independent of any character set selection specified by this standard.

References:

4.9

5.9.2 APPLICATION DATA

Parameters:

identifier (I)
data record (D)

Description:

This element supplements the information in the metafile in an application-dependent way. It has no effect on the picture generated by interpreting the metafile, or on the states of the metafile generator or interpreter.

The content of the identifier and data record parameters is not standardized.

NOTE - The contents of the data record may include such information as history data associated with pictures, description of algorithms used, etc.

References:

4.9

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 88-3728	2. Performing Organ. Report No.	3. Publication Date MARCH 1988

4. TITLE AND SUBTITLE

CGM REGISTRATION FOR CALS REQUIREMENTS

A Technical Study Completed for the Computer-aided Acquisition & Logistic Support (CALS) Program - Fiscal Year 1987 (Vol. 3 of 4)

5. AUTHOR(S)

Edited by Sharon J. Kemmerer

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

NBSIR, 10/86 through 9/87

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Office of Assistant Secretary of Defense (A&P)/WSIG
Department of Defense, Pentagon
Washington, DC 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)

The overall objective of the Department of Defense Computer-aided Acquisition & Logistic Support (CALS) Program is to integrate the design, manufacturing, and logistic functions through the efficient application of computer technology. The National Bureau of Standards has been funded since Spring 1986 to recommend a suite of industry standards for system integration and digital data transfer, and to accelerate their implementation. A major FY87 thrust was the completion of initial documentation of the high-priority standards required in the CALS environment. Volumes one, two and four provide a collection of the final reports presented to the CALS Policy Office. They cover such topics as text, product data, data management, raster compression, media, conformance testing, and graphics. This volume specifcally addresses Computer Graphics Metafile (CGM) registration requirements for CALS.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

CALS; CGEM; CGM; DoD; graphics; interpress; PostScript; registration

13. AVAILABILITY

☐ Unlimited
☒ For Official Distribution. Do Not Release to NTIS
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

297

15. Price

$24.95

USCOMM-DC 6043-P80