NBSIR 82-2631 (AF)

Initial Graphics Exchange Specification (IGES), Version 2.0

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Manufacturing Engineering
Automated Production Technology Division
Washington, DC 20234

February 1983

U.S. DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS

ACKNOWLEDGEMENT

The IGES authors wish to acknowledge the work of Mary Marello in the word processing of this document. Her patience, dedication and expertise were invaluable. Her assistance with consistency and formatting was most appreciated. Thank you, Mary, for an outstanding job.

NBSIR 82-2631 (AF)

INITIAL GRAPHICS EXCHANGE SPECIFICATION (IGES), VERSION 2.0

Bradford M. Smith, IGES Chairman, National Bureau of Standards
Kalman M. Brauner, Ph.D., The Boeing Company
Philip R. Kennicott, Ph.D., General Electric
Michael Liewald, The Boeing Company
Joan Wellington, National Bureau of Standards

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Manufacturing Engineering
Automated Production Technology Division
Washington, DC 20234

February 1983

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

The following IGES committee members contributed to this document:

Frances Akridge, Lockheed-Georgia Co., Marietta, GA
Robert Ard, ADRC, Milford, OH
Carole Avery, The Boeing Company, Seattle, WA
Peter Benjamin, Lockheed Missiles and Space, Sunnyvale, CA
Ted Berenyi, Deere & Company, Moline, IL
Istvan Bodnar, Control Data Corp., Arden Hills, MN
Walt W. Braithwaite, The Boeing Company, Seattle, WA
Ray Brengs, David Taylor R&D Center, Bethesda, MD
David Briggs, The Boeing Company, Seattle, WA
Pat Brown, Intergraph, Huntsville, AL
William Burd, Sandia Labs., Albuquerque, NM
Mike Butler, Martin Marietta, Orlando, FL
Richard Carey, The Boeing Company, Seattle, WA
Ted Berenyi, Deere & Company, Moline, IL
Carole Avery, The Boeing Company, Seattle, WA
Noel Christensen, Bendix Corp., Kansas City, MO
Robert Colsher, Gerber, South Windsor, CT
Jason Costantino, CADAM Inc., Burbank, CA
Rick Courter, IBM, Poughkeepsie, NY
Don Crockett, Vought Corp., Dallas, TX
Ralph Dratch, Booz, Allen & Hamilton, Cleveland, Ohio
David Ellis, Holliston, MA
Joseph Finnegan, McDonnell Douglas, St. Louis, MO
Edward Fournier, Softech, Bedford, MA
David Fredricks, CADAM Inc., Burbank, CA
Richard Fuhr, The Boeing Company, Seattle, WA
Roger Gale, General Dynamics, Pomona, CA
William Geoghegan, Bath Iron Works, Bath, ME
Albert Gibbons, Westinghouse Electric Corp., Pittsburgh, PA
Brian Giles, Xerox Corp., Webster, NY
William Gruttke, McDonnell Douglas, St. Louis, MO
James Gorman, Bell Telephone Labs., North Andover, MA
Douglas Hakala, Manufacturing Data Systems, Inc., Ann Arbor, MI
William Hinton, General Motors, Pontiac, MI
Jack Horgan, Applicon Corp., Burlington, MA
Robert Houk, Martin Marietta, Denver, CO
Stephen J. Kaminski, Magnavox Research Center, Torrence, CA
J. C. Kelly, Sandia Labs., Albuquerque, NM
C. Richard Lewis, General Motors, Warren, MI
Robert Lord, McDonnell Douglas, St. Louis, MO
David Maltz, McDonnell Douglas, St. Louis, MO
William Magretta, Computervision, Bedford, MA
Ralph Mayer, Computervision, Bedford, MA
Wayne McClelland, SDRC, Cincinnati, Ohio
Don Meagher, Phoenix Data Systems, Albany, NY
William Korbholz, Xerox Corp., El Segundo, CA
James Miller, Control Data, Arden Hills, MN
David Moore, Microsoft Corp., Seattle, WA
Roger Nagel, Lehigh University, Bethlehem, PA
Paul Nelson, General Motors, Warren, MI
Janet Oakes, McDonnell Douglas, St. Louis, MO

Larry Olson, Martin Marietta, Denver, CO
Curt Parks, General Dynamics, Pomona, CA
Andrew Pauker, Computervision, Bedford, MA
Robert Peterson, ITT Research Institute, Chicago, IL
James Pierce, Xerox Corp., El Segundo, CA
Mark Quinlan, CALMA, San Diego, CA
Garry Redden, Ford Motor Co., Dearborn, MI
Robert Rosen, Harry Diamond Labs., Adelphi, MD
Patrick Rourke, Newport News Shipbuilding, VA
Wes Rutherford, The Bendix Corp., Kansas City, MO
Randy Schmid, McDonnell Douglas, St. Louis, MO
Dirk Schroeter, Martin Marietta, Orlando, FL
Gary Silverman, IBM, Los Angeles, CA
Chia-Hui Shih, General Dynamics, San Diego, CA
Paul Serednicky, IBM , Poughkeepsie, NY
Jim Snyder, Union Carbide, Oak Ridge, TN
Randy Terada, CADAM Inc., Burbank CA
David Theilen, The Bendix Corp., Kansas City, MO
Wayne Tiller, SDRC, Milford, Ohio
Mimi Vaillancourt, Computervision, Bedford, MA
Earl Weaver, USA Ballistic Research Lab., Aberdeen Proving Ground, MD
Uwe Weissflog, IBM, Los Angeles, CA
Ron Wong, Ford Motor Co., Dearborn, MI

FOREWORD

Version 2.0 of the Initial Graphics Exchange Specification (IGES) represents both a refinement and extension of the earlier published work. Clarity and precision of the Specification have been dramatically improved as the result of wider public review and comment plus feedback from an ever increasing amount of implementation and testing. In addition, many extensions and enhancements have been incorporated in the Specification to expand its capability to communicate a wider range of product data developed and used by computer aided design and manufacturing systems. Despite these extensions and enhancements, Version 2.0 remains nearly upward compatible with Version 1.0. The only exception is a change in the Text Font Definition entity. The Version 2.0 document was approved in July 1982 by the IGES committee structure.

The many changes dispersed throughout the Version 2.0 document make it difficult for a reader to compare it with the earlier work. This task is compounded by a substantial change in the basic format of presentation. Hence, it is useful here to elaborate the primary differences that do exist. In addition, the reader of the document is aided by vertical bars in the margin that identify areas of non-trivial change. Of course, a complete record of every change is available from the Extensions and Repairs Committee's formal Change Control System which documents both the request for a change and the actual text modifications to implement the change.

Changes that will be noted in the Specification can be classified into four general areas: editorial, consistency, clarification, and technical extension. Editorial changes include the usual grammar, spelling, punctuation, etc. that are discovered with each re-reading. Changes to improve the consistency of the document include the use of the same terminology throughout, the establishment of a common format, and the defining of all terms before their use. In addition, a Glossary of Terms and an Index of Topics have been added. These changes are not denoted by change bars.

Users of Version 2.0 of the IGES Specification will be pleased to see the many technical extensions which have been added to augment its capability and expand it into new areas. Many geometry entities have been enhanced in scope to be more generally applicable. Included here are the parameterization in the Ruled Surface entity, a more general form of the Tabulated Cylinder entity, and the means of relating the Surface of Revolution entity to the common geometrical surfaces like spheres and cones.

Two new geometry entities, a Rational B-Spline Surface entity and a related Rational B-Spline Curve entity, were added in Version 2.0. The addition of these entities is expected to provide a much more general approach for surface and curve representation. Algorithms were developed for an exact conversion between the Rational B-Spline method and the Bezier method of representation. New structural entities were also developed and documented for both rectangular and circular arrays of geometric entities.

In the annotation area, Version 2.0 improves on the earlier work by specifying a much larger set of text fonts, although additional work remains to be done here. Improvements have been made in the clarity of intent for positioning and scaling of text material and in a more clearly defined Angular Dimension entity.

Two major applications areas have been addressed by Version 2.0: finite element modeling data and electronics printed wiring board product data. The earlier IGES Specification contained no means of handling this data, yet both are widely used applications on CAD/CAM systems.

The geometry of a finite element is defined by the ordered connection of geometric points called nodes. Communication of this data through IGES Version 2.0 is handled by defining the nodes with the Node entity. The node connectivity is defined by the Finite Element entity. To complete the finite element definition, new properties will be defined to communicate such items as modulus of elasticity, Poisson's ratio, thermal conductivity, moment of inertia, sheer modulus, and physical constraints.

The second major applications area addressed in the extended Specification is the communication of printed wiring board product data. Extensions in this area are intended to provide for transferring the physical shape of metallization on each layer, the location and size of drilled holes, the location and identification of components and their pins, the connectivity of certain component pins and their associated named signals, and the functional use of entities by graphics systems level. Some beginning is made into the transfer of design rules and in transferring schematic drawings, but only the physical design transfer is thought to approach complete coverage.

Altogether, four new geometric entities are provided for efficient transfer of commonly used printed wiring board graphical elements, eight new properties are defined to preserve design characteristics, one new associativity is specified for signal connectivity, and one change is introduced so that an IGES property may apply to levels as well as to individual entities.

A frequent criticism of the IGES format has been the anticipated large file lengths due primarily to the ASCII character representation. Included in Version 2.0 are the details of an optional or alternate binary format representation which addresses the problems of file size and processing speed. While efficiency improvements vary with word length and other variables, analysis of 20 IGES production files has estimated the savings in file size of 50 - 68%.

In total, the IGES Version 2.0 document is a major improvement in the Specification. It refines and more precisely describes the Version 1.0 capability as well as extends IGES into new geometry and application areas. Altogether, the document represents the discussions from 98 Change Requests which generated 56 documented Change Orders from the earlier work.

TABLE OF CONTENTS

LIST OF FIGURES

Figure No.

LIST OF TABLES

Table No.

INITIAL GRAPHICS EXCHANGE SPECIFICATION (IGES)
Version 2.0

1 GENERAL

1.1 Purpose

This document establishes information structures to be used for the digital representation and communication of product definition data. Use of the specification established herein permits the compatible exchange of product definition data used by various CAD/CAM (Computer Aided Design and Computer Aided Manufacturing) systems.

1.2 Field of Application

This Specification specifies a file structure format, a language format, and the representation of geometric, topological, and non-geometric product definition data in these formats. Product definition data represented in these formats will be exchanged via a variety of physical media. The specific features and protocols for the communications media are the subject of other standards. The methodology for representing product definition data in this Specification is extensible and independent of the geometric modeling methods used.

Section 2 defines the communications file structure and format. It explains the function of each of the five major segments of an IGES file. The geometry data representation in Section 3 deals with two- and three-dimensional edge-vertex models and with simple surface representations. Section 4 specifies non-geometric representations, including common drafting practices, data organization methods, and data definition methods.

In Sections 3 and 4, the product is described in terms of geometric and non-geometric information, with non-geometric information being divided into annotation, definition, and organization. The geometry category consists of elements such as points, lines, arcs, cubic splines and parametric surfaces which model the product. The annotation category consists of those elements which are used to clarify or enhance the geometry, including dimensions, drafting notation, and text. The definition category provides the ability to define specific properties or characteristics of individual or collections of data entities. The structure category identifies groupings of elements from geometric, annotation, or property data which are to be evaluated and manipulated as single items.

1.3 Concepts of Product Definition

This Specification is concerned with the data required to describe and communicate the essential engineering characteristics of physical objects as manufactured products. Such products are described in terms of their physical shape, their dimensions, and information which further describes or explains the product. The processes which generate or utilize the product definition data typically include design, engineering analysis, production planning, fabrication, material handling, assembly, inspection, marketing, and field service.

The requirements for a common data communication format for product definition can be understood in terms of today's CAD/CAM environment. Traditionally, engineering drawings and associated documentation are used to communicate product definition data. Commercial interactive graphics systems, originally developed as aids to producing these two-dimensional drawings, are rapidly developing sophisticated three-dimensional edge-vertex modeling capability. In parallel, extensive research work is being conducted in advanced geometric modeling techniques (e.g., parametric representations and solid primitives) and in CAM applications utilizing product definition data in manufacturing (e.g., NC machining and computer-controlled coordinate measurement). The result is rapid growth of CAD/CAM applications which should be able to exchange product definition data, but which usually employ incompatible data representations and formats. In addressing this compatibility problem, this Specification is concerned with needs and capabilities of current and advanced methods of CAD/CAM product definition development.

Product definition data may be categorized by their principal roles in defining a product. An example of such a categorization is presented in Figure 1-1. This Specification specifies communication formats (information structures) for subsets of the product definition.

1.4 Concepts of the File Structure

A format to allow the exchange of a product definition between CAD/CAM systems must, as a minimum, support the communication of geometric data, annotation, and organization of the data. The file format defined by this Specification treats the product definition as a file of entities, each entity being represented in an application-independent format, to and from which the native representation of a specific CAD/CAM system can be mapped. The entity representations provided in this Specification include forms common to the CAD/CAM systems currently available and forms which support the system technologies currently emerging.

- ADMINISTRATIVE
  - Product Identification
  - Product Structure
- DESIGN/ANALYSIS
  - Idealized Models
- BASIC SHAPE
  - Geometric
  - Topological
- AUGMENTING PHYSICAL CHARACTERISTICS
  - Dimensions and Tolerances
  - Intrinsic Properties
- PROCESSING INFORMATION
- PRESENTATIONAL INFORMATION

FIG. 1-1 CATEGORIES OF PRODUCT DEFINITION

The fundamental unit of information in the file is the entity. Entities are categorized as geometric and non-geometric. Geometric entities represent the definition of the physical shape and include points, curves, surfaces, and relations which are collections of similarly structured entities. Non-geometric entities typically serve to enrich the model by providing a viewing perspective in which a planar drawing may be composed and by providing annotation and dimensioning appropriate to the drawing. Non-geometric entities further serve to provide specific attributes or characteristics for individual or groups of entities and to provide definitions and instances for groupings of entities. Typical non-geometric entities for drawing definition, annotation, and dimensioning are the view, drawing, general note, witness line, and leader. Typical non-geometric entities for attributes and groupings are the property and the associativity entities.

A file consists of five or six sections: a binary section in the case of the binary format, start, global conditions, directory data, parameter data, and terminator. A file may include any number of entities of any type as required to represent the product definition. Each entity occurrence consists of a directory entry and a parameter data entry. The directory entry provides an index and includes descriptive attributes about the data, while the parameter data provides the specific entity definition. The directory data are organized in fixed fields and are consistent for all entities to provide simple access to frequently used descriptive data. The parameter data are entity specific and are variable in length and format. The directory data and parameter data for all entities in the file are organized into separate sections, with pointers providing bi-directional links between the directory entry and parameter data for each entity.

Each entity defined by the file structure of Section 2 has a specific assigned entity type number. While not all are assigned at this time, entity numbers 0001 through 5000 are allocated for specific assignment. Entity type numbers 5001 through 9999 are available for user specified assignment. The Index of Terms (Appendix E) includes an alphabetical listing of entity types.

Some entity types include a form number as an attribute. The form number serves to further define or classify an entity within its specific type.

The entity set includes provision for associativities and properties. The associativity provides a mechanism to establish relationships among entities, and to define the meaning of the relationship. The property allows specific characteristics, such as color, to be assigned to an entity or collection of entities. Each entity format includes structure for an arbitrary number of pointers to associativities and properties. The file structure provides for both standard associativities and properties to be included in the Specification, and unique definitions which will be defined by the user.

1.5 Concepts of the Information Structures for Wire-Frame Model Descriptions

The wire-frame model refers to the entity set defined by Sections 3 and 4, and comprises an entity-based product definition file. The entity types, as described in 1.4, are categorized as geometric and non-geometric. In general, the geometric entities are defined independently of one another (surfaces are an exception). Features have been provided to define and compose relationships among entities to enhance the model. The non-geometric entities include structures in which an entity may be defined by a collection of other entities and structures which are independent.

Several entity types which are used to provide relations or definitions are essential to the file structure methodology of this Specification and are described below.

1.5.1 Property Entity. The PROPERTY entity allows non-geometric numeric or textual information to be related to any entity. Any entity occurrence may reference one or more property entity occurrences as required.

Property entities themselves may exist independently of other entities. In this case the property is defined to be a property of the level indicated in the level field of the directory entry (DE) of the property. This allows for a general property to apply to all entities of a given level or for the assignment of an applications function to a level. Because the level field in DE is also allowed to point to an associativity of levels, properties could be applied to multiple levels.

1.5.2 Associativity Entities. The Associativity Entities are designed for use when several entities must be logically related to one another. Two types of entities are involved here: ASSOCIATIVITY DEFINITION and ASSOCIATIVITY INSTANCE. The associativity definition entity is used to specify the structure of the logical relationship, and the associativity instance entity is used to specify the information involved in a particular occurrance of the relationship.

Some associativities are defined as part of this Specification. These intrinsic definitions include GROUP and DEFINITION LEVEL associativities, and are defined in Section 4.3.

1.5.3 View Entity. A drawing or equivalent human-readable representation of the geometric model of a product is a two-dimensional projection of a selected subset of the model, together with non-geometric information such as text. The VIEW entity and VIEWS VISIBLE form of associativity control such representations. These provide information for orientation, clipping, line removal, and other characteristics associated with individual views rather than with the model itself.

1.5.4 Drawing Entity. The DRAWING entity allows a set of views to be identified and arranged for human presentation. Note that the view and drawing entities contain only the rules and parameters for extracting drawings from the geometric model. The actual product definition is not duplicated in various views, eliminating risk of conflicting or ambiguous information.

1.5.5 Transformation Matrix Entity. The TRANSFORMATION MATRIX entity allows translation and rotation to be applied to any geometric entity in the construction of the model and to the development of views of the model.

1.5.6 Macro Entities. This Specification includes a MACRO DEFINITION entity for defining new entity types which may then be used in the defining file in the same manner as the intrinsically defined entities. A language for defining these new entity types is specified in 4.3.6.

1.6 Appendices

As an aid to the implementor/user, a series of appendices is included with this Specification. The first three appendices provide examples of specific utilization. Appendix A provides explanation of spline representation and approaches for conversion techniques. Appendix B provides an example of an electrical application, and Appendix C, two mechanical examples. The last two appendices, a Glossary and an Index of Terms have been added as references for the user.

2 DATA FORM

2.1 General

Two different formats are defined to represent IGES data. These formats are ASCII and Binary, where the ASCII format utilizes a character oriented (card image) structure and the Binary format utilizes a byte oriented (bit string) structure. In each case, the parameter definitions in the file sections are identical. The two formats provide alternatives for ease of use and file size trade-offs. The ASCII format is comparatively simple but can yield excessive data volumes. The Binary format is more complex but offers a data volume reduction of approximately 60% (as compared to ASCII).

The constants, free format rules, and file structure are discussed in terms of the ASCII format. Following this discussion, the binary format is introduced together with necessary changes in constants and file structure.

2.2 ASCII Format

The file is written on 80 column records, using the ASCII (Code Extension Techniques for Use with the 7-bit Coded Character Set of American National Standard Code for Information Interchange (ASCII) X3.4-1968, ANSI X3.41-1974.) character set.

2.2.1 Constants

This Specification defines five types of constants: integer, floating point, string, pointer, and language statement.

2.2.1.1 Integer Constants. An integer constant is composed of one or more numerical characters. Although formally called an integer constant, it is more commonly called a fixed-point or integer number because of the fact that the decimal point is always assumed to be located to the right of the last numerical character of the number.

An integer constant may be positive, zero, or negative. While a positive integer number can have the special character plus (+) as its leading character, if an integer number is unsigned and nonzero, the Specification assumes it to be positive. An integer number must comply with the following four rules:

a. It must be a whole number. That is, it cannot contain a decimal point.

b. If negative, the special character minus (-) must be the leading character.

c. It cannot contain embedded commas.

d. Its maximum magnitude can be either plus or minus 2**(N-1) -1 (where N is parameter seven from the global section).

The following are examples of valid integer constants (assuming N is 32).

```
1
150
2147483647
0
-10
-2147483647
+3451
```

2.2.1.2 Floating-Point Constants. This Specification permits both single and double precision floating-point constants. The precision of these constants is specified in the global section, in parameters 8 through 11.

A single precision floating-point number may be expressed with or without an exponent. Double precision constants must be in exponential form.

A floating-point constant without an exponent is composed of one or more numerical characters and the special character period (.) that may be followed by one or more of these numerical characters to form what is called the fractional part of the constant. Sometimes called a real constant, it is more commonly called a floating-point constant to reflect the fact that the decimal point can be moved or floated to the beginning, middle, or end of the numerical characters forming the number. Floating-point constants may be positive, zero, or negative. A positive floating-point constant can have the special character plus (+) as its leading character. If a floating-point constant is unsigned and nonzero, the Spectification assumes it to be positive. A floating-point constant must comply with the following four rules:

a. If negative, the special character minus (-) must be the leading character.

b. It must contain a decimal point.

c. It cannot contain embedded commas.

d. The size of the number must be compatible with the parameters in the global section.

A floating-point constant may be expressed in exponential form. Single precision floating-point numbers use the letter "E" in the exponent, while double precision floating-point constants use the letter "D" in the exponent.

A floating-point constant in exponential form begins with a constant (real or integer) followed by an exponent letter ("E" or "D") followed by an integer constant. The first constant is called the mantissa and the second constant, the exponent. The value of the resultant floating-point constant is the value of the mantissa multiplied by ten raised to the power specified in the exponent. The precision of allowable numbers for the mantissa and exponent are given in the global section. Examples of floating-point constants are below:

Single precision non-exponent form:

```
264.091
0.
-0.58
+4.21
```

Single precision exponent form:

```
1.36 E 01
12.943E1
-13.09E-2
123.409E-4
0.1E-3
1.0E+4
```

Double precision exponent form:

```
145.98763D+04
2145.980001D-5
0.123456789D 9
```

Note:

Double precision floating-point constants must use the exponential form.

2.2.1.3 String Constants. A string constant in this Specification uses the Hollerith form as found in the ANSI specification of Fortran (Programming Language Fortran ANSI X3.9-1978). A string constant is preceded by an unsigned integer, and the letter "H". String constants have the following rules:

a. The string is preceded by a count of characters and the letter "H".

b. Any character from the ASCII set may appear in the string. (Blanks and the field and record delimiters have no special meaning within a string constant.)

c. String constants may cross record boundaries in the file (other constants may not). When a string constant does cross a record boundary, the last usable column on the current record is concatenated with column one on the succeeding record. The last usable column on parameter records is column 64; on other records it is column 72.

d. There is no limit on the size of a string constant.

Examples of valid string constants are:

```
3H123
10HABC.,:ABCD
12H HELLO THERE
8H0.457E03
```

2.2.1.4 Pointer Constants. A pointer constant is a one to seven digit integer identifying a record in the same subsection or an alternate subsection. The pointer value corresponds to the sequence number of the target record. The subsection of the target record is context determined. Where the pointer is an optional parameter, its use is denoted by a leading minus (-) sign. All other instances are unsigned. Pointers requiring less than seven non-zero digits are valid with or without the leading zeros. The pointer specification may not exist across record boundaries.

The sequence number for each section begins with 0000001 and numbering continues sequentially until ending at the appropriate number for the section.

Leading zeros in the sequence field may be optionally replaced with blanks. The number must be right justified. The letter codes for column 73 are as follows:

SECTION		LETTER CODE
a.	Start Section	S
b.	Global Section	G
c.	Directory Entry Section	D
d.	Parameter Data Section	P
e.	Terminate Section	T

2.2.1.5 Language Statement Constant. The language statement constant is an arbitrary string of alpha-numerics, punctuation, and blanks. The string is not preceded by the character count and hollerith delimiter 'H'. Language statements have a syntax which is fully defined in 4.3.6. The length of the string may be determined through the parameter data record count in the directory entry for the entity.

2.2.2 Free Format Rules

The data in several sections of the file may be entered in free format. The free format feature allows the specification of parameters in a prescribed order but does not specify a location on the record image. When free format is permitted, the following rules apply:

a. Blanks are ignored.

b. The field delimiter (default is comma) is used to separate parameters.

c. The record delimiter (default is semicolon) is used to terminate the list of parameters.

d. When two commas appear adjacent to each other (or separated only by blanks) the parameter is not specified in the file and should be given a default value.

e. If a semicolon appears before the list of parameters is complete, all remaining parameters should be given default values.

f. Blanks are not ignored in string constants. In addition the comma and semicolon are treated as characters in a string constant and do not have the meaning specified in (b) through (e).

g. Text parameters may be split between two records if necessary, whereas numerical parameters and pointers together with their accompanying delimiters are not to be so split.

h. Unless otherwise specified, the default values for a numeric argument and for a text argument are zero and a null string respectively. It is the responsibility of the pre-processor which creates a standard file to make sure that the default value is a reasonable one for the particular parameter.

2.2.3 File Structure

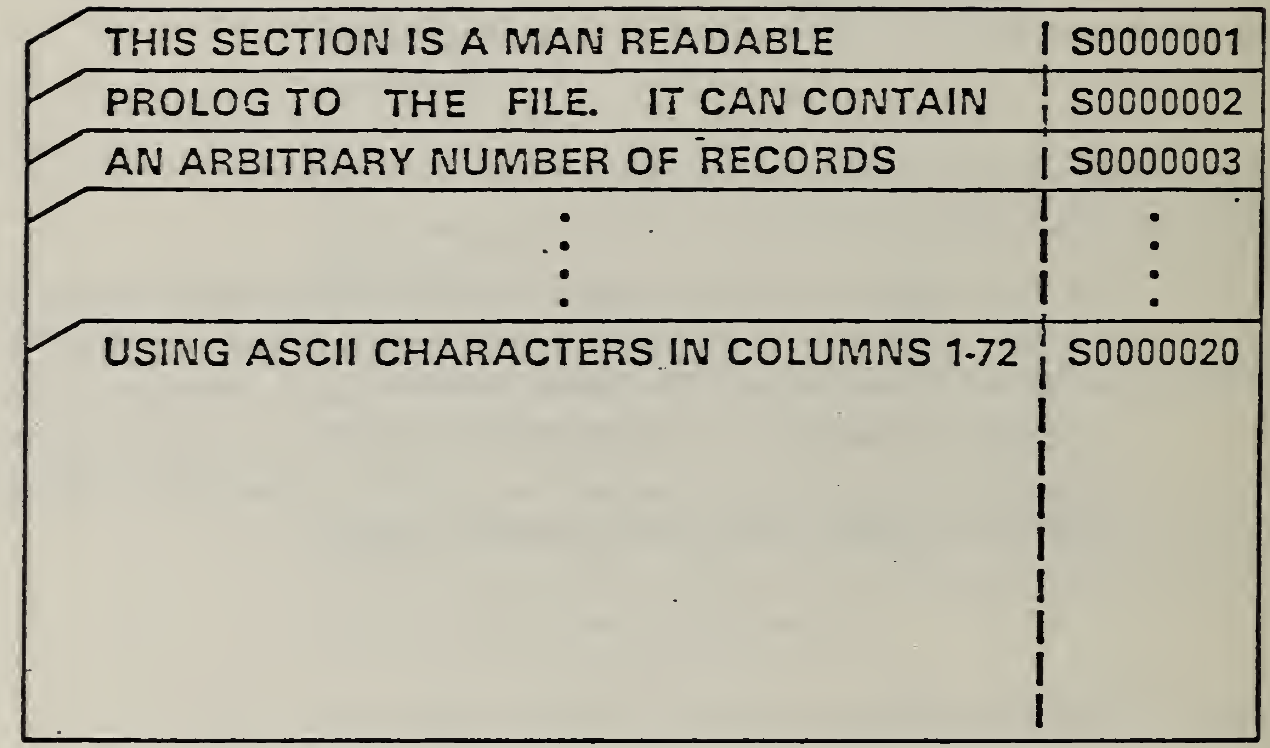
The file contains five subsections which must appear in order as follows:

a. Start Section
b. Global Section
c. Directory Entry Section
d. Parameter Data Section
e. Terminate Section.

2.2.3.1 Start Section

The start section of the file is designed to provide a man-readable prolog to the file. There must be at least one start record, and all records in the section must have the letter S in column 73 and a sequence number in column 74 through 80 (See 2.2.1.4). The information in columns 1 through 72 is not formatted in any special way except that the ASCII character set must be used. An example of a start section is shown in Figure 2-1.

START SECTION



```
THIS SECTION IS A MAN READABLE               | S0000001
PROLOG TO  THE  FILE.  IT CAN CONTAIN        | S0000002
AN ARBITRARY NUMBER OF RECORDS               | S0000003
                   .                         |    .
                   .                         |    .
                   .                         |    .
                   .                         |    .
USING ASCII CHARACTERS IN COLUMNS 1-72       | S0000020
```

FIG. 2-1 START SECTION

2.2.3.2 Global Section

The global section of the file contains the information describing the pre-processor and information needed by the post-processor to handle the file. All records in the global section shall contain the letter G in column 73 and a sequence number (See 2.2.1.4). The first two global parameters are used to redefine the delimiter and end of record characters if necessary. The default characters are "comma" and "semicolon" respectively.

The parameters for the global section are input in free format as described in 2.2.2. As implied in 2.2.2, the global parameters will end with the end of record delimiter. If the global section specifies new delimiter characters, they take over immediately and are used in the global section as well as the rest of the file. This is possible, because the comma and semicolon delimiter functions are the first two global parameters. The form of a new delimiter character is similar to any text string: 1H character . It is expected that if the comma and semicolon delimiters are not to be changed, the global section will begin with ",," to indicate the default values are desired.

The parameters in the global section are described in Table 2-1 and the paragraphs that follow. Unless explicitly stated, no defaults are provided.

TABLE 2-1 PARAMETERS IN THE GLOBAL SECTION

PARAMETER	FIELD TYPE	DESCRIPTION
1	String	Delimiter character (default=,)
2	String	End of record delimiter (default=;)
3	String	Product identification from sending system
4	String	File name
5	String	System ID . Vendor . Software version
6	String	ANSI Standard translator version
7	Integer	Number of bits for integer representation
8	Integer	Number of bits in a single precision floating point exponent

9	Integer	Number of bits in a single precision floating point mantissa
10	Integer	Number of bits in a double precision exponent
11	Integer	Number of bits in a double precision mantissa
(TO SYSTEM)		
12	String	Product identification for the receiving system
(FILE INFORMATION)		
13	Floating point	Model space scale (example: .125 indicates a value 1.0:8.0 real world)
14	Integer	Unit flag
15	String	Units. Two units have been defined: 4H INCH for unit flag = 1 and 2HMM for unit flag = 2
16	Integer	Maximum number of line weight gradations (1-32768). Refer to the directory entry parameter 12 (See 2.2.3.3) for use of this parameter.
17	Floating point	Size of maximum line width in units. Refer to the directory entry parameter 12 (See 2.2.3.3) for use of this parameter.
18	String	Date & time of file generation 13HYYMMDD.HHNNSS where: YY is year (last 2 digits) MM is month (01-12) DD is day (01-31) HH is hour (00-23) NN is minute (00-59) SS is second (00-59)
19	Floating Point	Minimum user-intended resolution or granularity of the model expressed in units defined by parameter 15 (example .0001)

20	Floating point	Approximate maximum coordinate value occurring in the model expressed in units defined by parameter 15. (Example: 1000.0 means for all coordinates $|X|$, $|Y|$, $|Z| \leq 1000$.)
21	String	Name of author
22	String	Organization

2.2.3.2.1 Delimiter Character. This parameter indicates which character is to be used to separate parameter values in the Global and Parameter Data sections. Each occurrence of this character denotes the end of the current parameter and the start of the next parameter. Two exceptions exist: (1) string constants in which the delimiter character may be part of the hollerith string; (2) language statements in which the delimiter character may be a part of the language syntax. The default value is a comma. See 2.2.2

2.2.3.2.2 End of Record Delimiter. This parameter indicates which character is to be used to denote the end of a list of parameters in the Global section and each Parameter Data section entry. Each occurrence of this character denotes the end of the current parameter list. Two exceptions exist: (1) string constants in which the delimiter character may be part of the hollerith string; (2) language statements in which the delimiter character may be a part of the language syntax. The default value is a semicolon. See 2.2.2.

2.2.3.2.3 Product Identification From Sender. This is the name of another identifier which is used by the sender to reference this product.

2.2.3.2.4 File Name. This is the name of the IGES file.

2.2.3.2.5 System ID. This parameter is an identification code which should uniquely identify the system which generated this file. It includes both the name of the system and the version of software on that system.

2.2.3.2.6 Translator Version. This parameter identifies the version of the translation software which created this file.

2.2.3.2.7 Number of Bits for Integer Representation. This parameter indicates how many bits are present in the integer representation of the sending system. This parameter sets limits on the range of values for integer parameters in the file.

2.2.3.2.8 Number of Bits in a Single Precision Floating Point Exponent. This parameter indicates how many bits are present in the exponent portion of the floating point number representation on the sending system. This parameter sets limits on the magnitude of floating point values in the file.

2.2.3.2.9 Number of Bits in a Single Precision Floating Point Mantissa. This parameter indicates how many bits are present in the fractional part of the floating point number representation on the sending system. The value of this parameter sets a limit on the precision of single precision floating point values in the file.

2.2.3.2.10 Number of Bits in a Double Precision Floating Point Exponent. This parameter indicates the number of bits in the exponent portion of the double precision floating point number representation on the sending system. The value sets limits on the magnitude of double precision floating point values in the file.

2.2.3.2.11 Number of Bits in a Double Precision Mantissa. This parameter indicates the number of bits in the fractional portion of a double precision floating point number representation on the sending system. This value sets a limit on the precision of double precision floating point values in the file.

2.2.3.2.12 Product Identification for the Receiver. This is the name or identifier which is intended to be used by the receiver to reference this product.

2.2.3.2.13 Model Space Scale. The ratio of model space to real world space.

2.2.3.2.14 Unit Flag. An integer value denoting the measuring system used in the file. The values in the file are assumed to be:

Unit flag = 1 (Inches)
= 2 (Millimeters)
= 3 (See Parameter 15 for name of units)

This is the controlling definition of units. A value of '3' should only be used when it is intended to transfer data to a system using the same units, in which case parameter 15 may provide additional information as to those units.

2.2.3.2.15 Units. A text string naming the unit of measure in the system. (e.g. 4HINCH)

2.2.3.2.16 Maximum Number of Line Weight Gradations. This is the number of equal subdivisions of line thickness.

2.2.3.2.17 Size of Maximum Line Width in Units. This is the actual width of the thickest line possible in the (scaled) file.

2.2.3.2.18 Date and Time of File Generation. This is a time stamp of when the file was created. (See Table 2-1.)

2.2.3.2.19 Minimum User-Intended Resolution. This parameter indicates the smallest distance in model space units that the system should consider as discernable. Coordinate locations in the file which are less than this distance apart should be considered to be coincident.

2.2.3.2.20 Approximate Maximum Coordinate Value. This is an upper bound on the value of coordinate data. The absolute magnitude of all coordinates is less than or equal to this value.

2.2.3.2.21 Name of Author. The name of the person responsible for the generation of the data contained in this file.

2.2.3.2.22 Organization. The organization or group with whom the author is associated.

2.2.3.3 Directory Entry Section

The directory entry section has one directory entry for each entity in the file. The directory entry for each entity is fixed in size and contains twenty fields of eight characters each spread across two consecutive eighty character records. Data are right justified in each field.

The purposes of the directory entry section are to provide an index for the file and to contain attribute information for each entity. The order of the directory entries within the directory entry section is arbitrary with the exception that a definition entity must precede all of its instances.

Some of the fields in the directory entry can contain either an attribute value directly, or a pointer to a set of such values. In these fields, a negative number indicates a pointer, while a positive number indicates an attribute value. Table 2-2 and the following paragraphs describe each directory entry field. For those fields accommodating either an attribute value or a pointer, there are two descriptions given. Figure 2-2 gives an abbreviated listing of the fields making up the directory entry for each entity.

TABLE 2-2 DIRECTORY ENTRY FIELD DESCRIPTION

NO.	FIELD NAME	MEANING AND NOTES
1	Entity Type Number	Identifies the entity type.
2	Parameter Data Pointer	Pointer to the first record of the parameter data for the entity. The letter P is not included.
3	Version Number	The version number indicates how to interpret the parameter data for this entity. This value will be 1 for all entities in the initial release of this Specification.
	Version = Pointer	Pointer to the first record of directory entry of the definition entity that specifies this entity's meaning. The letter D is not included.

4 Line Font Pattern Number

Selection of a system line font.

1 = Solid
2 = Dashed
3 = Phantom
4 = Centerline

Line Font Pattern Pointer

Pointer to the directory entry of a line font definition entity.

5 Level Number

Entity is defined on this level.

Level = Pointer

Pointer to the directory entry of a property entity (Form 1) which contains a list of levels on which the entity is defined.

6 View Pointer

Pointer to the directory entry of a view entity (410) or to views visible associativity entities. (402, Forms 3 or 4)

7 Defining Matrix Pointer

Pointer to the directory entry of a transformation (entity type number 124) matrix used in defining this entity; zero implies the identity transformation matrix will be used.

8 Label Display Associativity Pointer

Pointer to the directory entry of a label display associativity (Form 5).

9 Status Number

Provides four two-digit status values.

1-2 Blank Status

00 Visible
01 Blanked

3-4 Subordinate Entity Switch

00 Independent
01 Physically Dependent
02 Logically Dependent
03 Both (01) and (02)

5-6 Entity Use Flag

- 00 Geometry
- 01 Annotation
- 02 Definition
- 03 Other
- 04 Logical

7-8 Hierarchy

- 00 Global top down
- 01 Global defer
- 02 Use hierarchy property

Example: If an entity A has 00 in its DE status digits 7 and 8, all entities subordinate to A will have the attributes assigned to A. Consequently, the attributes assigned to all entities subordinate to A are ignored.

If an entity A has 01 in its DE status digits 7 and 8, the entities immediately subordinate to A will retain their own status. Consequently, the attributes assigned to A are ignored.

If an entity A has 02 in its DE status digits 7 and 8, the status of each attribute is determined by the hierarchy property entity which is pointed to by a pointer.

10	Section Code & Sequence Number	Physical count of this record from the beginning of the directory entry section, preceded by the letter D (odd number).
11	Entity Type Number	(Same as Field 1.)

12	Line Weight Number	System display thickness; given as a gradation value in the range of 0 to the maximum (parameter 16 of the global section). Thus, display thickness is: $\text{(Line Weight Number)} * \frac{\text{(Global parameter 17)}}{\text{(Global parameter 16)}}$. If 0 is specified, the receiving system's default line weight is to be used.
13	Pen Number	Pen or color number.
14	Parameter Record Count Number	Number of records in the parameter data for this entity.
15	Form Number	Certain entities have different interpretations. These interpretations are uniquely identified by a form number. Possible form numbers are listed within each entity description.
16-17	Reserved for future use	
18	Entity Label	Up to eight alphanumeric characters (right justified).
19	Entity Subscript Number	1 to 8 digit unsigned number associated with the label.
20	Section Code and Sequence Number	Same meaning as field 10 (even number).

DIRECTORY ENTRY (DE) SECTION

RECORD 1	ENTITY TYPE NO.	PARA-METER DATA	VERSION	LINE FONT PATTERN	LEVEL	VIEW	DEFINING MATRIX	LABEL DISPLAY	STATUS	SEQ #
	# 1	► 2	#,► 3	#,► 4	#,► 5	► 6	► 7	► 8	#'S 9	D------ 10
RECORD 2	ENTITY TYPE NO.	LINE WEIGHT	PEN NUMBER	PARA-METER RECORD COUNT	FORM NUMBER	RESERVED	RESERVED	ENTITY LABEL	ENTITY SUB-SCRIPT	SEQ #
	# 11	# 12	# 13	# 14	# 15	16	17	18	# 19	D------ 20

#	– NUMBER
►	– POINTER
#, ►	– NUMBER OR POINTER (POINTER HAS NEG SIGN)

FIG. 2-2 DIRECTORY ENTRY (DE) SECTION

2.2.3.3.1 Entity Type Number. An integer number indicating the type of entity.

2.2.3.3.2 Parameter Data Pointer. This is the sequence number of the first parameter data record for this entity. The letter P is not included.

2.2.3.3.3 Version Number. For a positive value, this indicates the IGES version to which the entity conforms. For a negative value, the absolute value of this field is interpreted as a pointer to the structure definition entity which specifies the schema for this entity type number.

2.2.3.3.4 Line Font Pattern Number. This indicates a display pattern to be used to display a geometric entity. A positive value indicates that the receiving system's corresponding version of the solid, dashed, phantom and centerline fonts should be used. A negative value indicates that the absolute value should be interpreted as a pointer to a line font definition entity (entity number 304) which provides the information specifying the display pattern.

2.2.3.3.5 Level Number. This value specifies a graphic display level or levels to be associated with this entity. A positive value indicates the graphic level this entity exists on. A negative value indicates the absolute value should be interpreted as a pointer to a property entity (entity number 406, form number 1) which contains a list of levels to be associated with the entity. This feature allows an entity to exist on multiple graphic levels.

2.2.3.3.6 View Pointer. This value is a pointer to the directory entry of a view entity (entity number 410) or a Views Visible Associativity (entity number 402, form 3 or 4) which defines the viewing perspective to be used to display the entity. The Views Visible Associativity allows the specification of view dependent characteristics for entities associated with this view.

2.2.3.3.7 Defining Matrix Pointer. This value is a pointer to the directory entry of a transformation matrix entity (entity number 124). This entity provides four form numbers to indicate the interpretation of the matrix. Form 0 indicates that the matrix defines a local coordinate system in which the entity coordinate data is defined. Form 10, 11 and 12 define specific cartesian, cylindrical and spherical coordinate systems respectively. A pointer value of zero indicates that the transformation matrix is the identity matrix.

2.2.3.3.8 Label Display Associativity Pointer. This is a pointer to the directory entry of a label display associativity (entity number 402, form 5) which defines how the entity's label and subscript are to be displayed in different views.

2.2.3.3.9 Status Number. This value contains four pieces of information which are concatenated together into a single integer number. The individual values are described in the following paragraphs.

2.2.3.3.9.1 Blank Status. This value defines whether the entity is meant to be visible on the output device of the receiving system. A value of 00 implies the entity is to be displayed and a value of 01 implies the entity is not to be displayed.

2.2.3.3.9.2 Subordinate Entity Switch. This value indicates whether or not the entity is referenced by other entities in the file. This implies whether or not the receiving system's processor needs to "remember" this entity because it is involved in the processing of other entities. The values are defined as follows:

00: The entity is an independent entity not referenced (i.e. pointed to) by any other entities in the file. It is not involved in the processing of other entities.

01: The entity is an element of a geometric or annotative entity structure and is not intended to exist independently outside the context of that geometric or annotative entity structure. Processing of this entity should be deferred until the processing of the geometric or annotative entity structure.

02: The entity is a member of (i.e. is pointed to by) a logical relationship structure such as an associativity or a subfigure. The entity is not dependent on the processing of the logical relationship structure for its existence.

03: Both conditions 01 and 02 above apply to this entity.

2.2.3.3.9.3 Entity Use Flag. This indicates the intent of the entity. It classifies the entity as intending to serve in the following manners:

00: The entity is used to define the geometry of the structure of the product.

01: The entity is used to add annotation or description to the file. This includes geometric entities used to form annotation or description.

02: The entity is used in definition structures of the file. It is not intended to be valid outside of the other entities which reference the definition structure. An example is the entities in a subfigure definition. They are intended to be valid in the subfigure instances that reference the subfigure definition. This class includes all entities in the 300 entity type number range.

03 (other):

The entity is being used for other purposes such as defining structural features in the file. This category corresponds roughly to the 400 range, but there are exceptions. For example, a subfigure instance (408) could define geometry, thus having an entity use flag = 0 or it could define a drawing format, thus having an entity use flag = 01. An associativity instance would ordinarily have the value 03. Exceptions include connect nodes and text nodes (value = 00) and associativities concerned with display where it would have the value 01. The view and drawing entities have value 01 (annotation). Transformation depends on its use: If used only for annotation (e.g., defining a view) the value is 01; if used for defining geometry or for defining geometry and annotation, value is 00.

04 (logical):

The entity defines virtual geometry present because of the instancing of a structural entity such as a text node and connect node.

Default:

Entity use is geometry.

2.2.3.3.9.4 Hierarchy. This value indicates the relationship between entities in a hierarchical structure and is used to determine which entity's directory entry attributes should be applied. Three values are provided:

00: The directory entry attributes will apply to entities subordinate to this entity.

01: The directory entry attributes of this entity will not apply to subordinate entities.

02: The hierarchy property determines which attributes from this entity are to be applied to subordinate entities.

2.2.3.3.10 Sequence Number. A number which specifies the position of the DE record in the directory entry section. The first record is record number 1, the second 2, etc. Successive records each increment this value by 1 (no gaps allowed) with each directory entry consisting of exactly 2 records; thus the sequence number of the first DE record for any entity is always odd and the sequence number of the second record is always even.

2.2.3.3.11 Entity Type Number. This is the same as Field 1.

2.2.3.3.12 Line Weight Number. This value denotes the thickness (or width) with which an entity should be displayed. A specific series of possible thicknesses are specified by global parameters 16 and 17. The largest thickness possible is that specified in global parameter 17 and is denoted by setting this value equal to the value in global parameter 16. The smallest thickness possible is equal to the result of dividing global parameter 17 by global parameter 16 and is denoted by setting this value equal to 1. Thicknesses between the smallest and largest thickness are available in increments equal to the smallest possible thickness and are denoted by setting this value equal to the integer number of (adjacent) increments required.

A value of 0 indicates that the default line weight display of the receiving system is to be used.

2.2.3.3.13 Pen Number. This value indicates a pen selection that the entity is to be displayed with on a plotting device. It may be interpreted to provide color selection for multi-color display devices.

2.2.3.3.14 Parameter Record Count Number. This is the number of records in the parameter data section which contain the parameter data for this entity.

2.2.3.3.15 Form Number. This value indicates an individual interpretation of the entity to be used when processing the parameter data for this entity. Some entity types allow multiple interpretations of their parameter data. This parameter along with the entity type number uniquely identify the interpretation of the parameter data.

2.2.3.3.16 Reserved Field. This field is reserved for future use and should be left blank.

2.2.3.3.17 Reserved Field. Same as Field 16.

2.2.3.3.18 Entity Label. This is an alphanumeric identifier or name for this entity. It is used in conjunction with the entity subscript number (Field 19) to provide an alphanumeric identifier for the entity.

2.2.3.3.19 Entity Subscript Number. This is a numeric qualifier for the entity label (Field 18).

2.2.3.3.20 Sequence Number. Same as 2.2.3.3.10

2.2.3.4 Parameter Data Section

The parameter data section of the file contains the parameter data associated with each entity. The following information is true for all parameter data.

2.2.3.4.1 Parameter data are placed in free format (see 2.2.2) with the first field always containing the entity type number. Therefore, the entity type number and a field delimiter (default is cornma) precede parameter one of each entity. The free field part of the parameter record ends in column 64. Columns 66 through 72 on all parameter records contain the sequence number of the first record in the directory entry of the entity for which parameter data is being presented. Column 73 of all records in the parameter section shall contain the letter P and columns 74 through 80 shall contain the sequence number (See 2.2.1.4)

2.2.3.4.1.1 With the exception of text strings, all parameter values are restricted from crossing record boundaries. Thus, numeric values must start and end on the same record and the terminating delimiter must be on the same record. When a text field does cross record boundaries, column 64 on the current record is considered to be next to column 1 on the next record. Parameters to be defaulted are indicated by two field delimiters (default is comma) with zero or more intervening blank characters. A record delimiter (default is semicolon) indicates that the parameter list is complete and any remaining parameters should receive default values. A record delimiter should always be the last character of a parameter set even if all parameters were explicitly specified.

2.2.3.4.2 Note that two groups of parameters appear at the end of nearly all entities. The first of these general parameters, the set of back pointers/text pointers, serves two purposes. There is an option in the associativity definition to specify that back pointers are required. If this option is chosen, an entity which is pointed to by the associativity must have a pointer in its own parameter list pointing back to the associativity, i.e., a back pointer. This back pointer will appear in the first group of parameters.

A second purpose for the first group of general parameters is to allow an entity to point to text which is associated with that particular entity. This is done by inserting a pointer to a general note in the first group of general parameters. Thus, there are only two types of entities which can be pointed to by the first group of general parameters: an associativity instance which points to the entity and a general note.

The second group of general parameters is used to point to property entities which may be associated with the particular entity.

2.2.3.4.3 Any desired comment may be added after the record delimiter. Note that additional records may be used for this purpose by keeping the directory entry pointer in columns 65-72 constant. Figure 2-3 shows a parameter section.

PARAMETER DATA SECTION

		65	73	74
RECORD 1	ENTITY TYPE NO. (PARAMETERS SEPARATED BY COMMAS)	DE PTR	P	SEQUENCE NUMBER
RECORD 2	(PARAMETERS SEPARATED BY COMMAS)	DE PTR	P	SEQUENCE NUMBER
RECORD 3	• • • •			
RECORD N	(PARAMETERS SEPARATED BY COMMAS);	DE PTR	P	SEQUENCE NUMBER

DE PTR POINTS TO THE DIRECTORY ENTRY FOR THIS ENTITY
SEQUENCE NUMBER BEGINS WITH THE LETTER P AND IS SEQUENTIALLY NUMBERED

FIG. 2-3 PARAMETER DATA SECTION

2.2.3.5 Terminate Section

There is only one record in the terminate section of the file. It is divided into ten fields of eight columns each. The terminate section must be the last record of the file. It has a "T" in column 73 and columns 74 through 80 contain the sequence number with a value of one (1).

The fields on the terminate record contain the character representing the section type and the last sequence number used in each of the previous sections. The fields are defined below and shown in Figure 2-4.

FIELD	COLUMNS	SECTION
1	1-8	Start Section
2	9-16	Global Section
3	17-24	Directory Entry Section
4	25-32	Parameter Section
5-9	33-72	(not used)
10	73-80	Terminate Section

TERMINATE SECTION

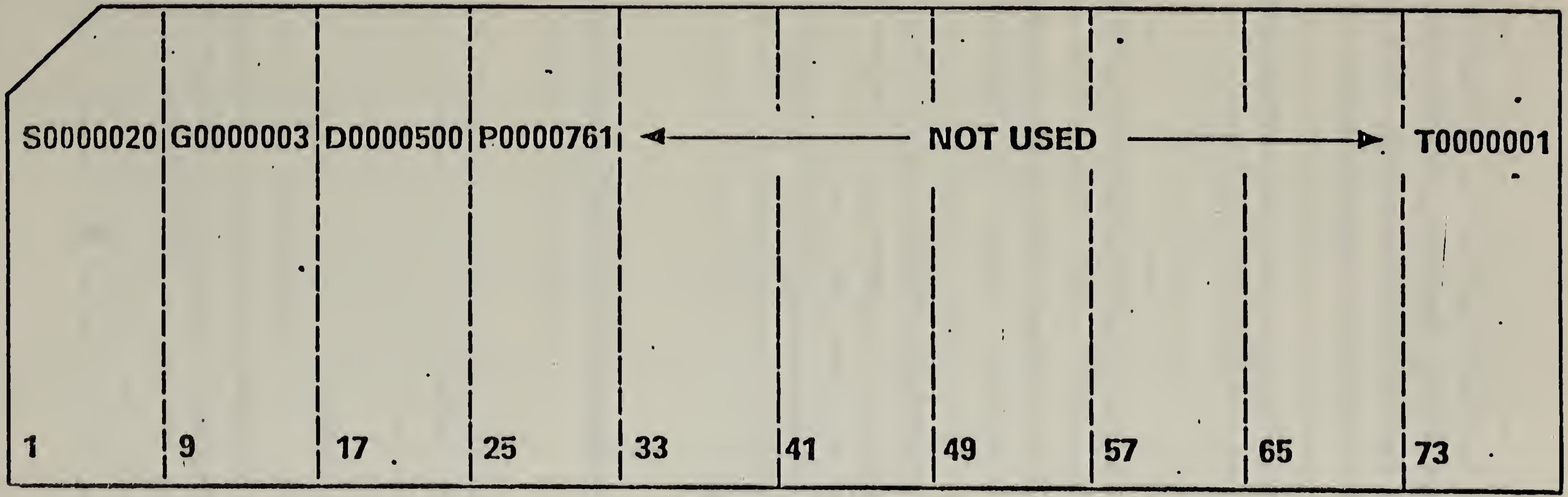


FIG. 2-4 TERMINATE SECTION

2.3 Binary Format

The format defined in Section 2.2, refered to as ASCII IGES, has 80 character fixed length records. This format, although usable by a wide community, creates files that can be quite large. In an attempt to minimize required file size and speed numerical conversion, this section describes a binary representation of the IGES data which may be used as an alternative format to ASCII IGES. All entity parameterizations and data organization are otherwise identical to the ASCII form.

The described format is intended to be treated as a continous bit stream by the sending and receiving systems. Thus the data may be blocked or grouped in any way acceptable to the sending and receiving systems. The data is transportable by all current communication protocols with the data treated as "transparent" or bit stream data. Each data set may be stored as a bit stream on other media, such as tape or disk in any format supported by the system which preserves the integrety of the bit streams. Data blocking is considered to be the province of the operating systems on the various support systems. For this reason data blocking was not considered as a part of the ASCII IGES Specification. Therefore, data blocking will not be considered a part of the Binary IGES Specification.

2.3.1 Constants. The following constants need to be represented in Binary IGES

- integer numbers
- floating point numbers
- text strings
- pointers
- language constants

A control byte will precede each value or set of values of the same type, unless otherwise specified. The control byte will specify the format of the following value or set of values, the quantity of subsequent values with that format, and whether values other than the initial value following the control byte are present. If the control byte indicates that values subsequent to the initial value of the set are absent, all subsequent values, up to the quantity indicated are assumed to have the same value as the initial value following the control byte.

The repetition portion of the control byte is unsigned and biased by 1 so that the true quantity of numbers to which the repetition field applies is one more than the unsigned value of the field.

The format of the control byte is as shown in Figure 2-5.

2.3.1.1 Integer Numbers. Integer numbers will conform to a specified structural standard.

The structure of an integer number shall be a sign bit followed by a two's complement integer of length i-1 as shown in Figure 2-6.

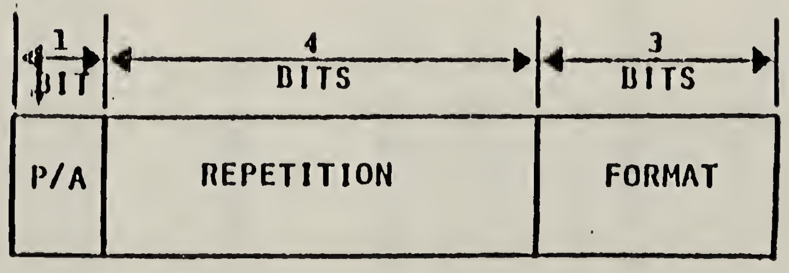
The generator of IGES data can select two lengths, i, of integer data.

The length of single length data is i_s and the length of double length data is i_d.

2.3.1.2 Floating Point Numbers. Floating point numbers will conform to a specified structural standard.

The structure of a floating point number shall be a sign bit followed by a biased exponent value of NX bits which is a power of 2, and a normalized binary fraction of NF bits. The fraction lies between 0.5 (inclusive) and 1.0 (exclusive). The value of the number is the sign applied to the fractional part multiplied by two raised to the power specified by the exponent part. The sign field consists of one bit. A sign of 0 indicates a positive number and a sign of 1 denotes a negative number. The exponent field consists of NX bits and is interpreted as an unsigned integer, BX, often referred to as the biased exponent. The value of the exponent is its unbiased value X which is obtained by deducting the bias B=2**(NX-1). A special interpretation is given to a biased exponent of zero, as discussed later.

CONTROL BYTE FORMAT:



P/A: = 0 IF ONLY FIRST OF A SET OF REPEATED VALUES IS PHYSICALLY PRESENT
= 1 IF ALL EXPECTED VALUES ARE PHYSICALLY PRESENT

REPETITION: (NUMBER OF FOLLOWING VALUES + 1) TO WHICH THIS CONTROL BYTE APPLIES

FORMAT: = 0 IF DEFAULT VALUE IS TO BE USED
= 1 IF SINGLE LENGTH INTEGER
= 2 IF DOUBLE LENGTH INTEGER
= 3 IF SINGLE PRECISION FLOATING POINT
= 4 IF DOUBLE PRECISION FLOATING POINT
= 5 IF POINTER
= 6 IF TEXT STRING

FIG. 2-5 FORMAT OF CONTROL BYTE

THE PAD OF ZEROES IS INCLUDED ONLY IF THE LENGTH I IS NOT A MULTIPLE OF 8 BITS

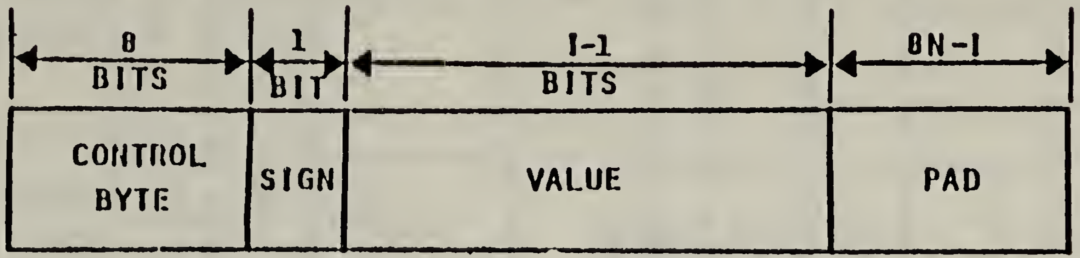


FIG. 2-6 INTEGER PRIMITIVE FORMAT

The fraction field consists of NF bits interpreted as the low order bits of a normalized (NF+1)-bit fraction part, F. Since the most significant bit of a normalized fraction is always 1 it is not explicitly represented.

Numbers with a non-zero biased exponent have a value given by:

$$(-1)^{SIGN} * 2^{(BX-B)} * F$$

When the biased exponent is zero, the number is interpreted as follows:

1. SIGN = 0. The number represented is zero, regardless of the contents of the fraction field.

2. SIGN = 1. The number represented is a reserved operand.

The structure of a floating point number is shown in Figure 2-7.

The generator of IGES data can select two lengths of floating point data by specifying the length of each exponent (NX) and the length of each fractional portion (NF).

2.3.1.3 Text Strings. Text Strings will conform to a specified structural standard.

Following the control byte will be a character count with a length of i_s. Where the character count exceeds the capability of an i_s length integer, the text string is broken up into substrings. In order to indicate that another substring follows the current string, a negative character count is used. The number of characters in the substring is the absolute value of the character count. A positive character count indicates the last substring.

The structure of the text string is shown in Figure 2-8.

THE PAD OF ZEROES IS INCLUDED ONLY IF THE LENGTH OF THE FLOATING POINT NUMBER (1+NX+NF) IS IIOT A MULTIPLE OF 8 BITS

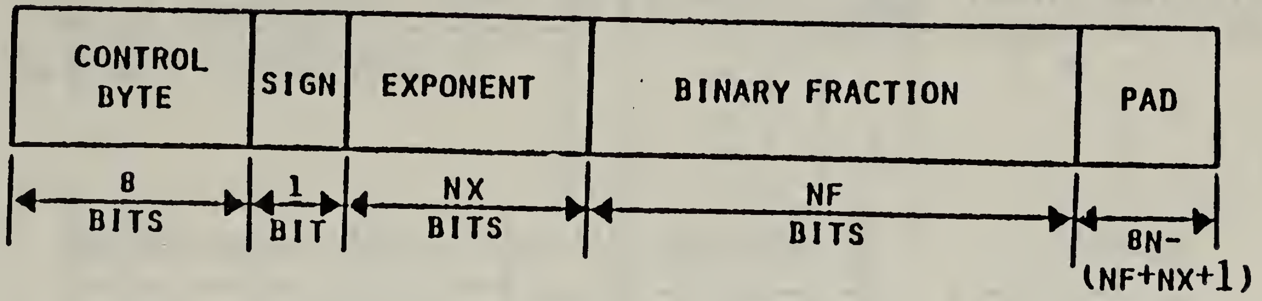


FIG. 2-7 FLOATING POINT PRIMITIVE FORMAT

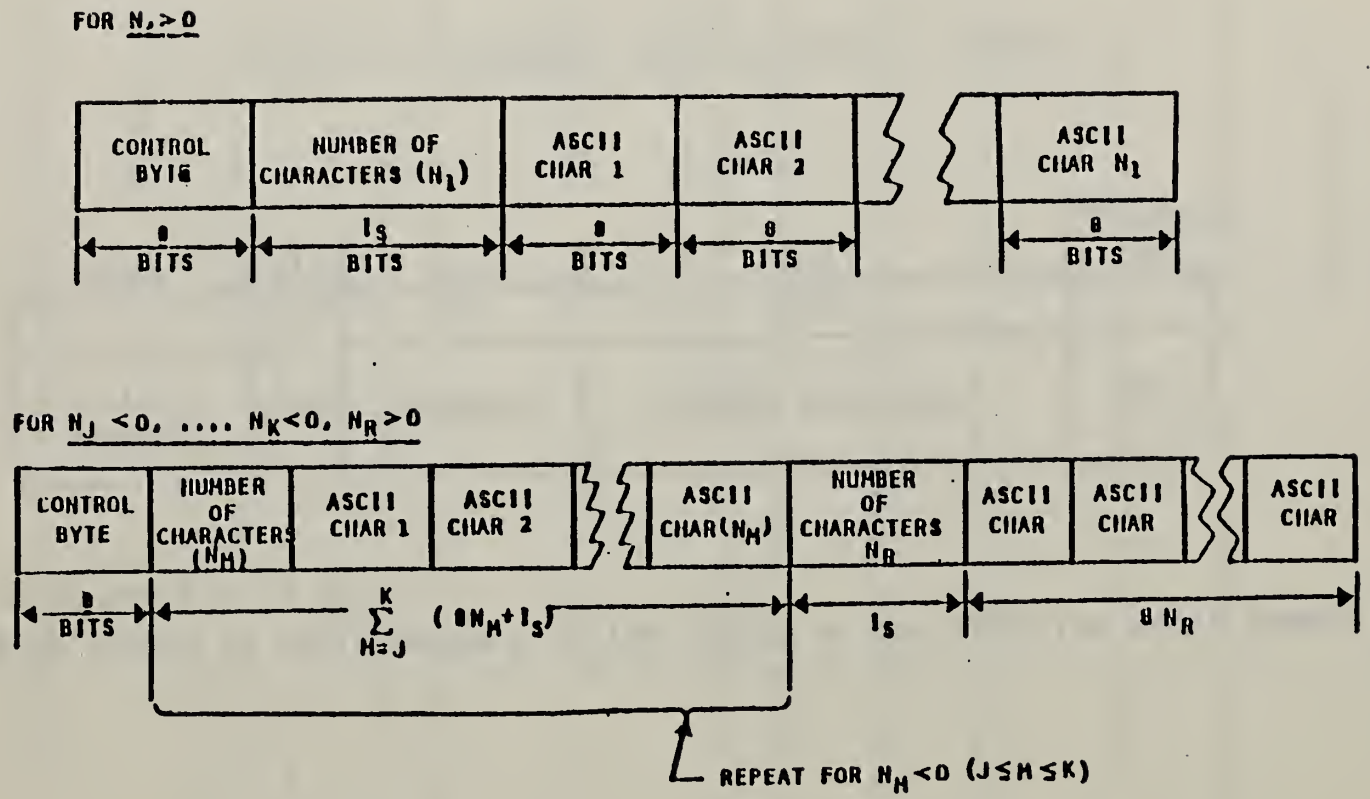


FIG. 2-8 TEXT STRING PRIMITIVE FORMAT

2.3.1.4 Pointers. Pointers will conform to a specified structural standard.

The structure of a pointer shall be a 32 bit integer. The pointer shall contain the relative byte position of the entity byte count of the DE or PD entity to which it is pointing. A pointer to the first DE entity will have a value of 1. A pointer to the second DE entity will have a value equal to the number of bytes of the first DE entity plus one. A pointer to the first PD entity will have a value of 1. Pointers with values of zero or negative are not actual pointers but may have a default meaning depending upon the IGES interpretation. For example, a defining matrix pointer of zero would imply that the identity matrix is to be used. This case might also be handled by using the control byte, instead, to indicate a default value.

2.3.1.5 Language Primitives. Language primitives are the text strings of the MACRO definition entity which, in ASCII IGES, are not preceded by nH and are terminated with a record delimiter. In Binary IGES the format of language primitives will be identical to text strings. Each language primitive (MACRO statement) will be an individual text string.

2.3.2 File Structure. The general file structure is as shown in Figure 2-9 and is comprised of the following six sections:

- Binary information section
- Start section
- Global section
- Directory entry section
- Parameter Data section
- Terminate section

Following each section is zero, one or many 8-bit null padding characters, represented by the ASCII letter N. These characters do not belong to the section and have no meaning. They are provided to assist the creator of an IGES file with physical system limitations such as word or sector boundaries.

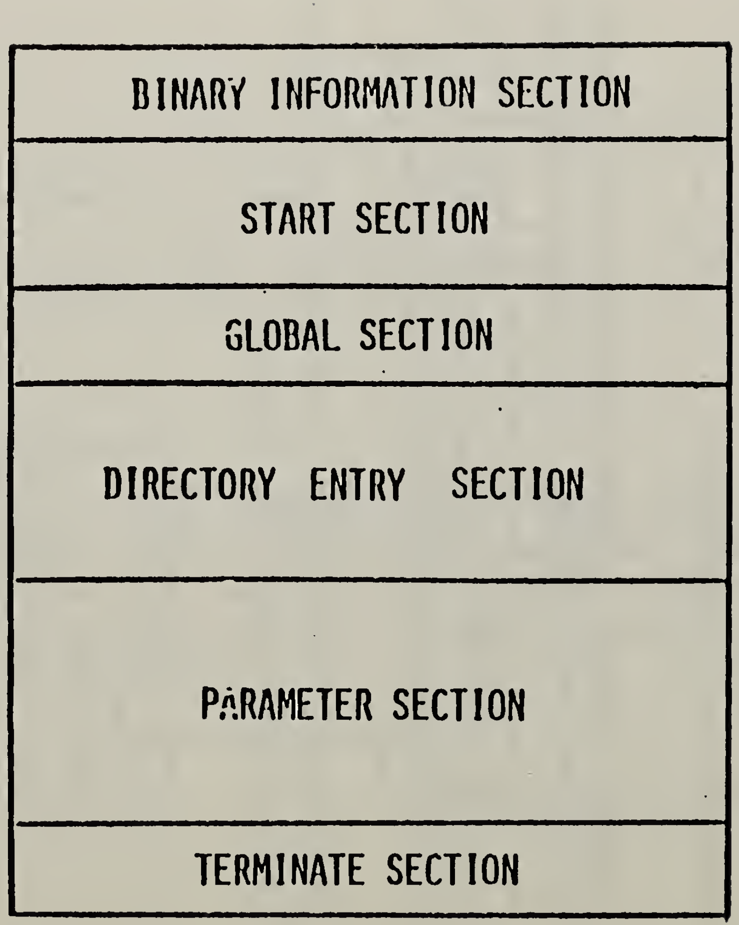
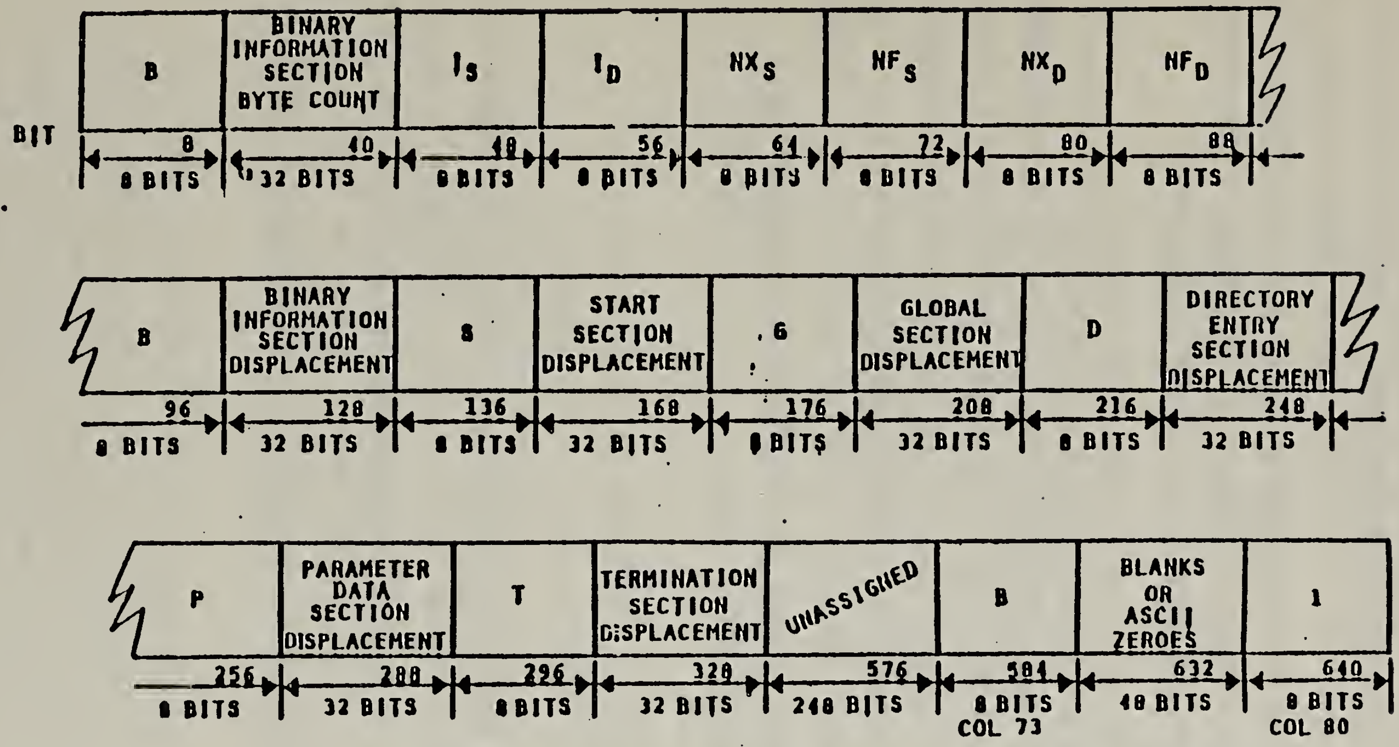


FIG. 2-9 BINARY IGES GENERAL FILE STRUCTURE

Several files can be concatenated by following the terminate section of the first file with the binary information section of the second file. Following the terminate section of the last file shall be zero, one, or many null padding characters followed by an 8-bit end of information designator, the ASCII letter E. Any information following the letter E shall be ignored.

2.3.2.1 Binary Information Section. The format of the binary information section is as shown in Figure 2-10. It is comprised of the following data items, all of which are integers unless otherwise specified.

- Binary identification section identifier consisting of the ASCII letter B.
- Binary identification section byte count. This byte count excludes the 5 bytes required for the section identifier and section byte count. This byte count also excludes any null padding characters. The value of this byte count will be 75.
- Length i_s of single length integer primitives.
- Length i_d of double length integer primitives.
- Length NX_s of exponent of single precision floating point primitives.
- Length NF_s of binary fraction of single precision floating point primitives.
- Length NX_d of exponent of double precision floating point primitives.
- Length NF_d of binary fraction of double precision floating point primitives.
- ASCII letter B.
- Binary information section displacement. This is the byte count of the total length of the binary information section including all null padding characters. This length is the actual length from the initial B of the binary information section up to but not including the S of the start section.
- ASCII letter S.
- Start section displacement. This is the byte count of the total length of the start section including all control bytes and null padding characters. This length is the actual length from the initial S of the start section up to but not including the G of the global section.
- ASCII letter G.
- Global section displacement. This is the byte count of the total length of the global section including all control bytes and null padding characters. This length is the actual length from the initial G of the global section up to but not including the D of the directory entry section.



* NO FIELDS IN THE BINARY INFORMATION SECTIONS HAVE CONTROL BYTES.

FIG. 2-10 FORMAT OF BINARY INFORMATION SECTION

- Directory entry section displacement. This is the byte count of the total length of the descriptive entity section including all control bytes and null padding characters. The length is the actual length from the initial D of the directory entry section up to but not including the P of the parameter data section.
- ASCII letter P.
- Parameter data section displacement. This is the byte count of the total length of the parameter data section including all control bytes and null padding characters. This length is the actual length from the initial P of the parameter data section up to but not including the T of the terminate section.
- ASCII letter T.
- Terminate section displacement. This is the byte count of the total length of the terminate section including all null padding characters. This length is the actual length from the initial T of the terminate section up to but not including either the letter B of binary information section of the next binary IGES file, the initial character of the start section of an ASCII IGES file, or the letter E of the end of information designator.
- 31 unassigned bytes.
- ASCII letter B.
- 6 ASCII blanks or zeroes.
- ASCII character 1.

No control bytes are applied to this section. Thus the characters in the equivalent of columns 73 through 80 of the binary information section are similar in format to the section identification of ASCII IGES and can be used to determine if a file is ASCII or binary. If the file contains an S in column 73 of its first 80 bytes, it is ASCII. If it contains a B, it is binary.

2.3.2.2 Start Section. The format of the start section is as shown in Figure 2-11. It is comprised of the following data items:

- A start section identifier consisting of the ASCII letter S
- Byte count for the start section. The byte count excludes the 5 bytes required for the start section identifier and section byte count. This byte count also excludes any null padding characters.
- One or more language or text primitives which are logically equivalent to columns 1 through 72 of ASCII IGES. There is no required physical correspondence between ASCII IGES card images and language/text primitives in that one language/text primitive may contain the equivalent of several complete or partial ASCII IGES card images. Carriage return characters may be embedded in the language/text primitives. Control bytes only apply to the language and text primitives. No control bytes precede the section identifier and byte count.

2.3.2.3 Global Section. The format of the global section is as shown in Figure 2-12. The global section is comprised of the following data items:

- Global section identifier consisting of the ASCII letter G
- Global section byte count. This byte count excludes the 5 bytes required for the global section identifier and the section byte count. This byte count also excludes any null padding characters.
- 22 global parameters.

Control bytes apply only to the 22 global parameters.

The global parameters have the same sequence and meaning as the ASCII IGES global parameters with the exception that global parameters 1 (delimiter character), 2 (end of parameter delimiter), 7 (number of bits for integer representation), 8 (number of bits in a single precision floating point exponent), 9 (number of bits in a single precision floating point

THE FORMAT OF THE LANGUAGE/TEXT PRIMITIVES IS AS SHOWN IN FIGURE 2-3

* THESE FIELDS DO NOT HAVE CONTROL BYTES

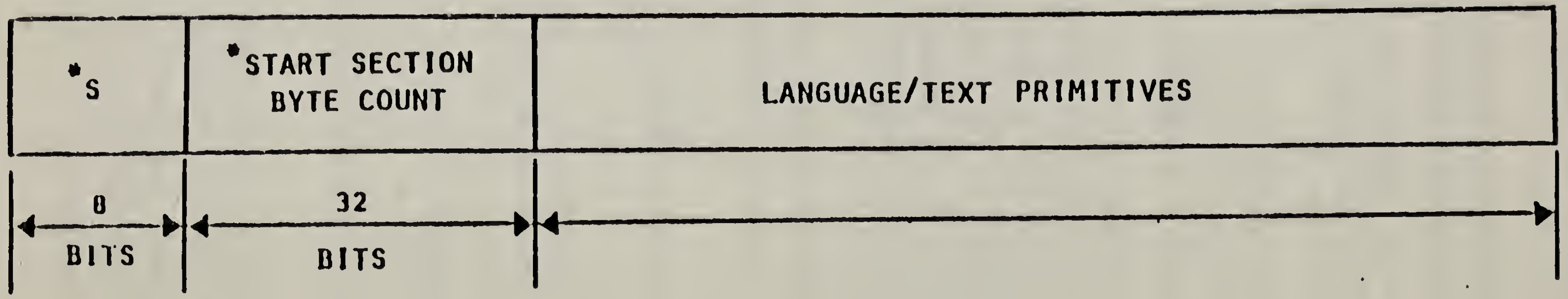


FIG. 2-11 FORMAT OF START SECTION

8 BITS	32 BITS				
G*	GLOBAL SECTION BYTE COUNT*	GLOBAL PARAMETER 1	GLOBAL PARAMETER 2		GLOBAL PARAMETER 22

* THESE FIELDS DO NOT HAVE CONTROL BYTES

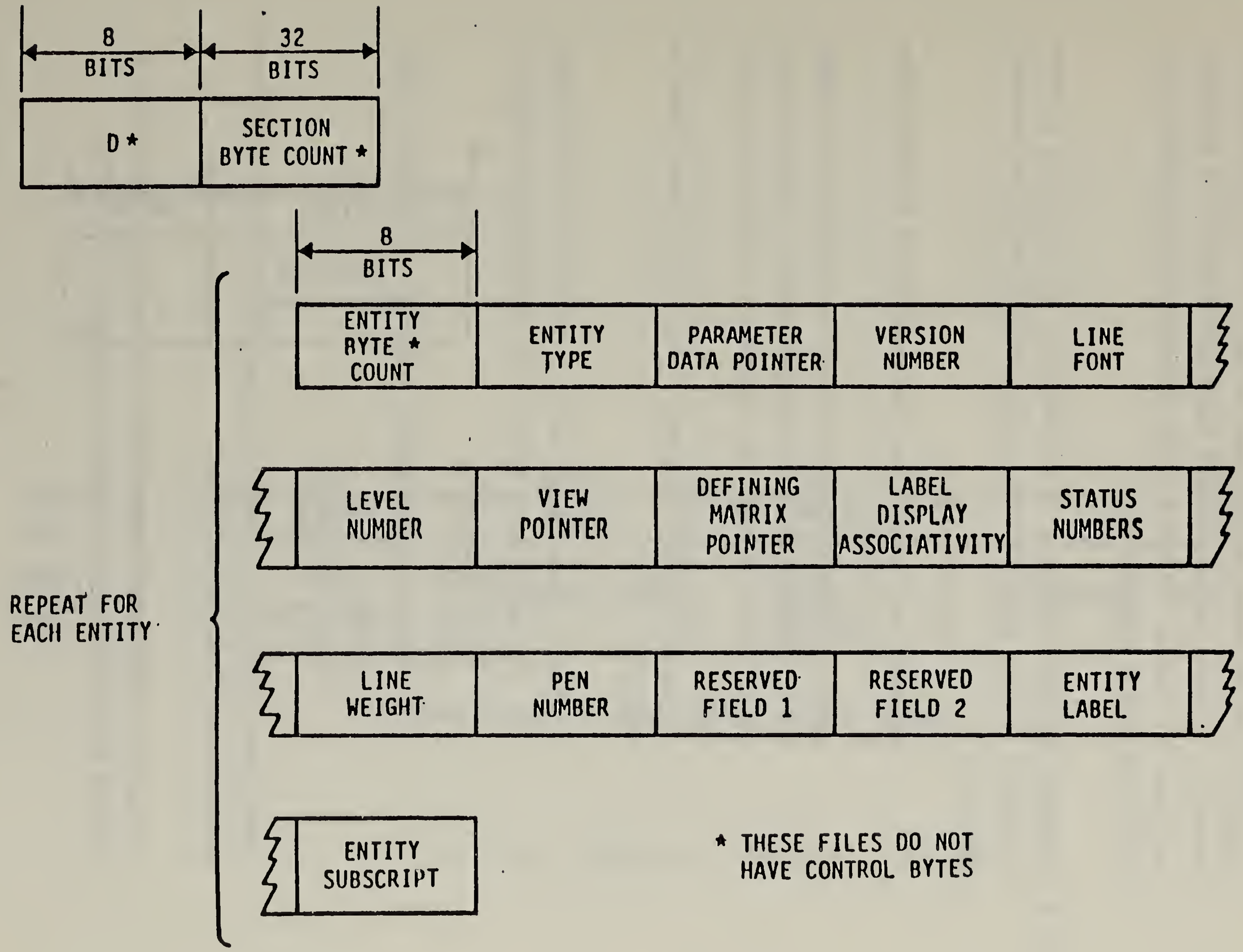
FIG. 2-12 GLOBAL SECTION FORMAT

mantissa), 10 (number of bits in a double precision exponent), and 11 (number of bits in a double precision mantissa) shall be ignored in binary IGES. The binary information section shall supersede these global parameters.

2.3.2.4 Directory Entry Section. The format of the directory entry section is as shown in Figure 2-13. The directory entry section is comprised of the following data items:

- Directory entry section identifier consisting of the ASCII letter D
- Directory entry section byte count. This byte count excludes the 5 bytes required for the section identifier and section byte count. This byte count also excludes any null padding characters.
- For each directory entry, the following 17 data fields are present:

  - entity byte count, which is composed of the lengths, including control bytes, of the subsequent 16 data fields.
  - entity type
  - parameter data pointer (relative to the parameter data section)
  - version number
  - line font
  - level number
  - view pointer (relative to the directory entry data section)
  - defining matrix pointer (relative to directory entry data section)
  - label display associativity
  - status number
  - line weight
  - pen number
  - form number
  - reserved field 1
  - reserved field 2
  - entity label
  - entity subscript



* THESE FILES DO NOT HAVE CONTROL BYTES

FIG. 2-13 FORMAT OF DE SUBRECORD

Control bytes apply only to the last 16 data fields.

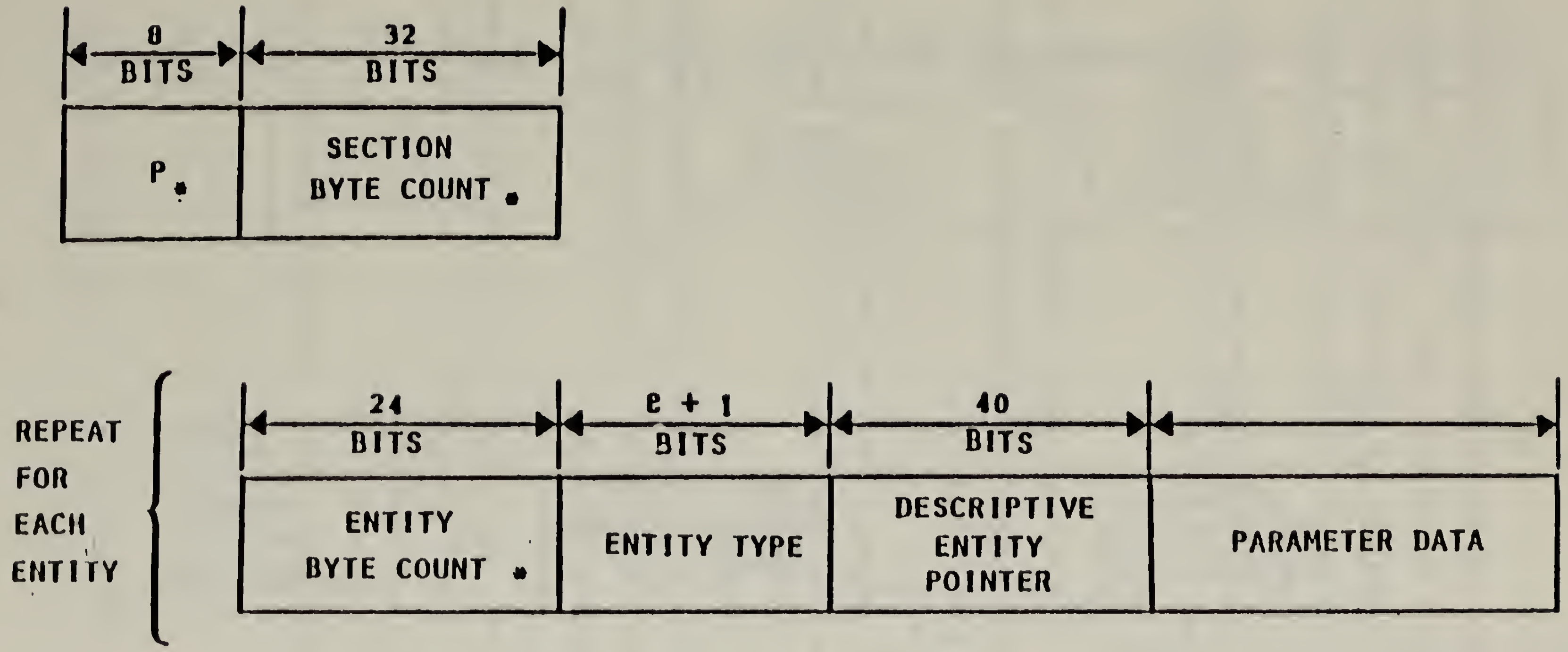
The directory entry data fields, except for the entity byte count, are identical to and have the same sequence as the ASCII IGES fields. Within a single IGES file, the length of DE record for each entity (in bytes) shall be consistent. If in the future additional fields are required, it is preferable to increase the number of fields for each directory entry and add any new fields subsequent to existing fields.

2.3.2.5 Parameter Section. The format of the parameter data section is as shown in Figure 2-14. The parameter data section is comprised of the following data items:

- Parameter data section identifier consisting of the ASCII letter P
- Parameter data section byte count. This byte count excludes the 5 bytes required for the section identifier and section byte count. This byte count also excludes any null padding characters.
- For each parameter data entity, the following data fields are required:
  - entity byte count, which is composed of the lengths, including control bytes, of all subsequent data fields for this entity.
  - entity type
  - directory entry pointer (relative to directory entry section)
  - parameter data

Control bytes apply only to the entity type, directory entry pointer and parameter data fields.

The parameter data entity fields, except for the entity byte count, are identical to and have the same sequence as the ASCII IGES fields.



* THESE FIELDS DO NOT HAVE CONTROL BYTES

FIG. 2-14 FORMAT OF PARAMETER SECTION

2.3.2.6 Terminate Section. The format of the terminate section is as shown in Figure 2-15. The terminate section is comprised of the following data items:

- Terminate section identifier consisting of the ASCII letter T
- Terminate section byte count. This byte count excludes the 5 bytes required for the section identifier and section byte count. This byte count also excludes any null padding characters.
- ASCII letter B
- Binary identification section byte count, including the section identifier, and section byte count, but excluding any null padding characters.
- ASCII letter S
- Start section byte count, including the section identifier, section byte count, and all control bytes but excluding any null padding characters.
- ASCII letter G
- Global section byte count, including the section identifier, section byte count, and all control bytes but excluding any null padding characters.
- ASCII letter D
- Directory entry section byte count, including the section identifier, section byte count, and all control bytes but excluding any null padding characters.
- ASCII letter P
- Parameter data section byte count, including the section identifier, section byte count, and all control bytes but excluding any null padding characters.

The terminate section has no control bytes applied to any of its data.

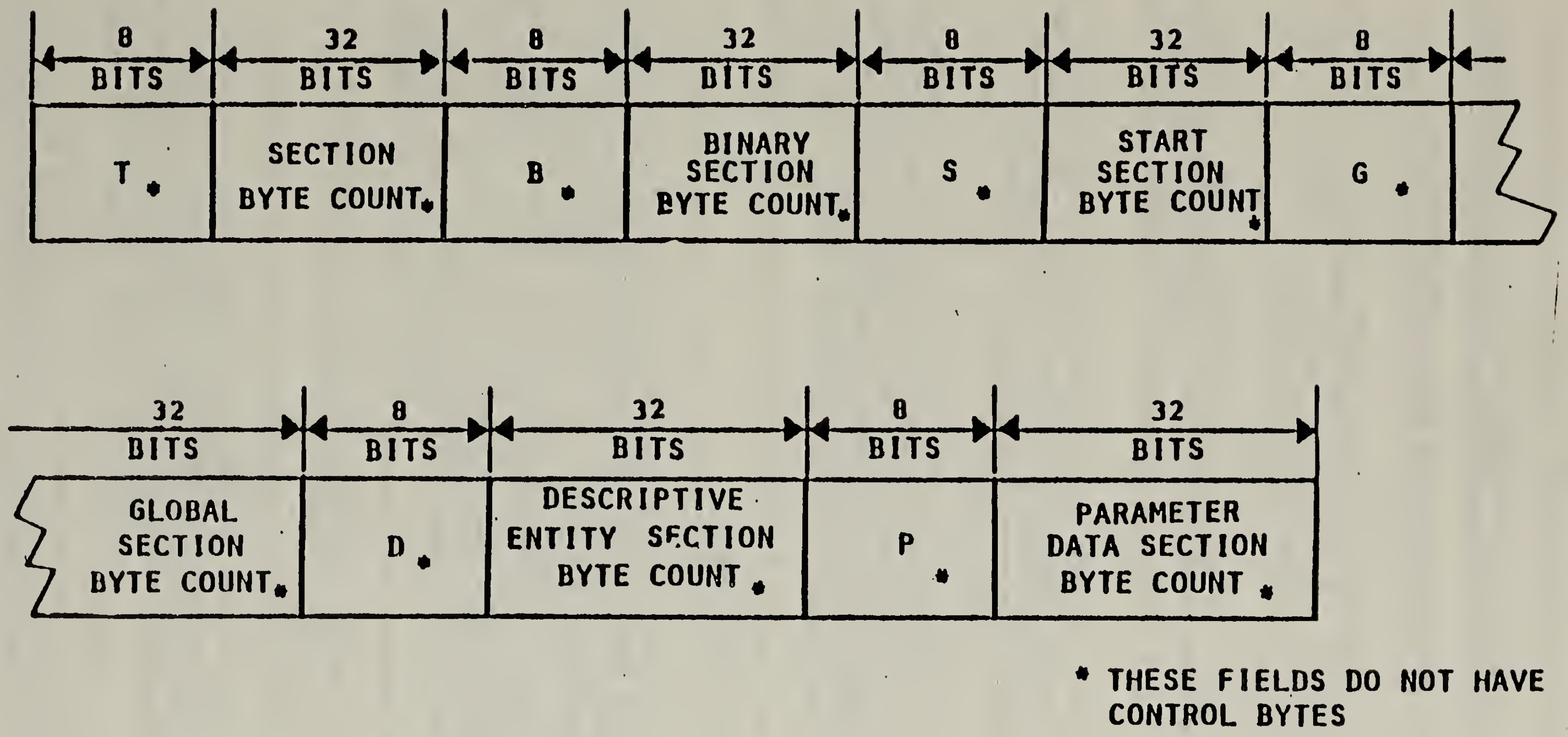

8 BITS
T *
32 BITS
SECTION BYTE COUNT *
8 BITS
B *
32 BITS
BINARY SECTION BYTE COUNT *
8 BITS
S *
32 BITS
START SECTION BYTE COUNT *
8 BITS
G *
32 BITS
GLOBAL SECTION BYTE COUNT *
8 BITS
D *
32 BITS
DESCRIPTIVE ENTITY SECTION BYTE COUNT *
8 BITS
P *
32 BITS
PARAMETER DATA SECTION BYTE COUNT *
* THESE FIELDS DO NOT HAVE CONTROL BYTES

FIG. 2-15 FORMAT OF TERMINATE SECTION

3 GEOMETRY

3.1 General

This section gives information concerning the geometry entity types available to be used in the entity-based product definition file.

3.1.1 Descriptions of the various directory entry fields were given in 2.4. These fields remain the same across all entities.

3.1.2 In this section, those entities making extended use of field 15 in the directory entry (Form Number) are indicated, and the various options are listed.

3.1.3 The parameter data entry for each entity is also described in this section. The fields for this entry vary from entity to entity.

3.1.4 This section introduces a model space concept and a definition space concept. Model space is three-dimensional Euclidean space, the space in which the "model" (or product) being represented resides. The model space X, Y, Z coordinate system is a right-handed Cartesian coordinate system. It is fixed relative to the model.

3.1.5 Definition space is also three-dimensional Euclidean space, but has its own right-handed Cartesian XT, YT, ZT coordinate system. In contrast to model space where a single fixed coordinate system exists, the definition space coordinate system may vary from entity to entity. The origin of a definition space coordinate system may be any point in model space, and the orientation may be arbitrary with respect to model space. It is assumed that the unit of length is always the same in both the model space and the definition space coordinate systems.

3.1.6 The definition space concept allows the use of a temporary coordinate system in positioning certain geometric entities into model space. This concept plays a simplifying role that is most apparent in connection with those entities which can be contained within a single plane. Use of definition space entails initially describing an entity in definition space, and then converting this to a model space description. Thus, a rotation matrix and a translation vector, if needed, are used to generate model space coordinates from definition space coordinates. The rotation matrix and the translation vector are both treated within the transformation matrix entity.

3.1.7 There are two equivalent points of view that can be taken concerning how the geometric entity is related to model space from its definition space description. In order to examine these with minimum involvement, the translation vector is assumed to be the zero vector. This implies that the origin of the definition space coordinate system coincides with the origin in the model space coordinate system.

3.1.8 The first point of view imagines that the two coordinate systems are initially coincident (that is, X axis to XT axis, etc.) but that the XT, YT, ZT coordinate frame is free to rotate relative to the X, Y, Z frame. The geometry entity is then considered to be defined relative to the XT, YT, ZT frame, and the rotation matrix then rotates this frame, geometry included, so that the geometry entity is positioned as desired relative to the X, Y, Z frame.

3.1.9 The second point of view imagines that the XT, YT, ZT frame is initially situated so that the geometry entity within definition space is positioned in the desired manner relative to model space. The rotation matrix then leaves the geometry entity fixed, but rotates the XT, YT, ZT frame. At the completion of the rotation, the XT, YT, ZT frame becomes the X, Y, Z frame. The result is that the geometry entity is then positioned as desired relative to the X, Y, Z frame.

3.1.10 It is to be emphasized that the discussion here pertains to a single rotation matrix whose action in transforming coordinates can be viewed intuitively in two ways. Each point of view stresses the temporary nature of the XT, YT, ZT system, insofar as what is ultimately of interest is the relationship of the geometry entity to the X, Y, Z frame.

3.1.11 From what has been said, it can be seen that the rotation matrix is always an orthogonal matrix with determinant equal to one.

3.1.12 In a case when the geometry entity to be located within model space can be contained within a single plane, it can likewise be seen that the definition space concept can be used in such a way that the geometry entity as initially described in definition space can be considered to lie in the XT, YT-plane (i.e., the plane ZT=0). From this, it is then convenient to also allow entities to be situated in definition space in any plane parallel to the XT, YT plane (i.e., ZT=arbitrary constant).

3.1.13 As indicated in 1.5.5, each entity in this section is acted upon by a transformation matrix. This implies that each entity makes use of the definition space concept, i.e., is defined initially in definition space, and then transformed into model space. Thus the complete definition of a geometry entity, with respect to model space, involves the transformation matrix. However, in some instances, it may very well be that the transformation matrix will leave all coordinates unchanged. This will be the case exactly when the rotation matrix is the identity matrix and the translation vectory is the zero vector. (In this situation, a convention can be used to prevent unnecessary processing. See the explanation given in 2.2.3.3 for Field 7 of the directory entry.)

3.1.14 Within model space, circular arcs, conic arcs, straight lines, and parametric splines arising from the circular arc entity, the conic arc entity, the line entity, or the parametric spline entity, respectively, are directed curves i.e., have an associated start point and terminate point. (An "end point" of a curve may be either a start point or a terminate point.) Any curve resulting from an instance of the composite curve entity is also a directed curve. For each of these entity types, the manner of assigning one of the two possible directions is discussed within the description of each individual entity.

3.1.15 Within the entity descriptions that follow, some refer to a "counterclockwise direction" with respect to a sense of rotation in the XT, YT plane. Since the XT, YT plane is located within three dimensional XT, YT, ZT space, this phrase is ambiguous unless a viewing direction is specified from which to view the rotation within the plane. The viewing direction is taken to be from the positive ZT axis looking "down" upon the XT, YT plane. Then, if a clock were imagined to be lying "face up" in the XT, YT plane, i.e., so as to be readable from the chosen viewing direction along the ZT axis - the phrase "counterclockwise direction" refers to the sense of rotation which is opposite the sense of rotation of the hands of the clock. This same notion of the meaning of counterclockwise carries over to any plane that is parallel to the XT, YT plane.

3.1.16 Entity type numbers from 100 through 199 are reserved for geometry entities. The following entity type numbers have been assigned:

Entity Type	Entity Type Number
Circular Arc Entity	100
Composite Curve Entity	102
Conic Arc Entity	104
Copious Data Entity	106
Plane Entity	108
Line Entity	110
Parametric Spline Curve Entity	112
Parametric Spline Surface Entity	114
Point Entity	116
Ruled Surface Entity	118
Surface of Revolution Entity	120
Tabulated Cylinder Entity	122
Transformation Matrix Entity	124
Linear Path Entity	106
Simple Closed Area Entity	106
Flash Entity	125
Rational B-Spline Curve Entity	126
Rational B-Spline Surface Entity	128
Node	134
Finite Element	136

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 3.1.15 for information relating to use of the term counterclockwise.

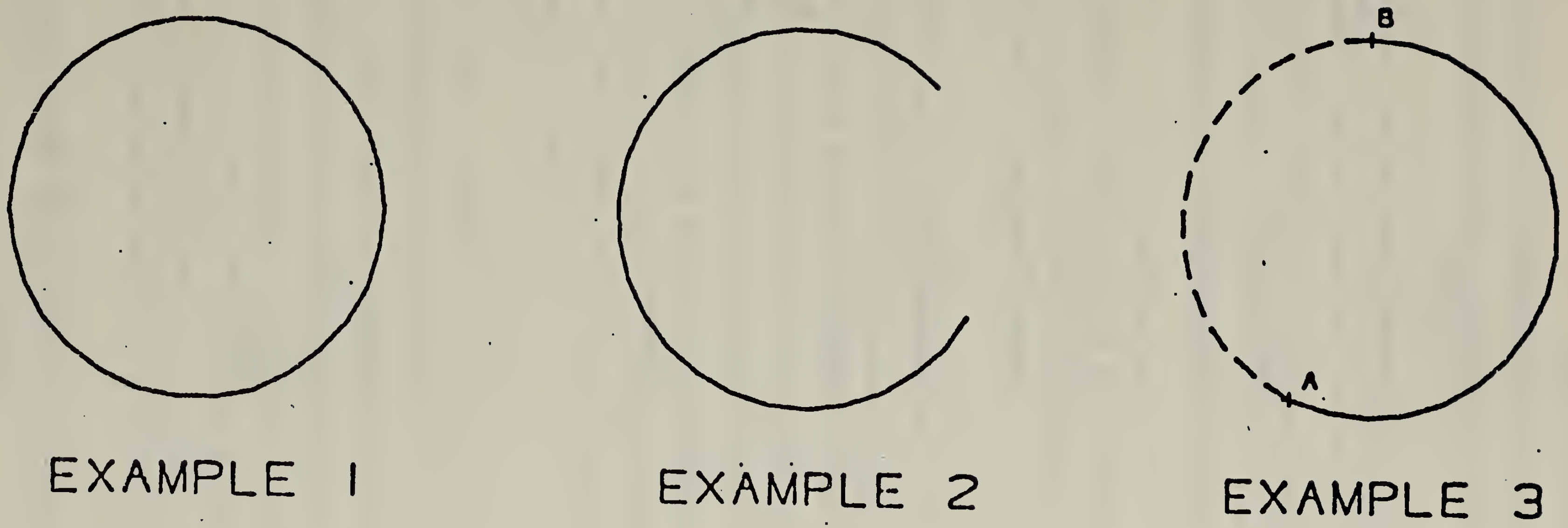
3.2.2 The direction of the arc with respect to model space is determined by the original counterclockwise direction of the arc within definition space, in conjunction with the action of the transformation matrix on the arc.

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

$$C(t) = \text{Center} + (R*\cos t, R*\sin t)$$

$$t_o \leq t \leq t_1$$
$$0 \leq t_o \leq 2\pi$$
$$t_o \leq t_1 \leq t_o + 2\pi$$

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.



EXAMPLE 1

EXAMPLE 2

EXAMPLE 3

FIG. 3-1 EXAMPLES OF THE CIRCULAR ARC ENTITY

3.2.5 Directory Data

ENTITY TYPE NUMBER : 100

3.2.6 Parameter Data

Parameter	Value	Format	Comment
1	ZT	Floating Point	Parallel ZT displacement of arc from XT, YT plane
2	X1	Floating Point	Arc center abscissa
3	Y1	Floating Point	Arc center ordinate
4	X2	Floating Point	Start point abscissa
5	Y2	Floating Point	Start point ordinate
6	X3	Floating Point	Terminate point abscissa
7	Y3	Floating Point	Terminate point ordinate
8	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
9	DE	Pointer	
.	.	.	Pointers to associativities or general notes
.	.	.	
.	.	.	
8+N	DE	Pointer	
9+N	M	Integer	Number of properties
10+N	DE	Pointer	
.	.	.	
.	.	.	Pointers to properties
.	.	.	
9+N+M	DE	Pointer	

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. 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 exists as an independent entity, and thus has its own transformation matrix and display attributes. Each constituent entity may have text or properties associated with it.

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 entity is included as an allowable entity type because of a specific functional capability deemed desirable. The desirable functional capability is to be able to attach data to either the start point or the terminate point of any of the constituent curve entities in the defining list. (Intuitively, this capability allows data to be attached to any of the "corners" of the composite curve or to either of its end points.) When used in a certain well-defined way, the point entity can provide this functional capability. Accordingly, there are certain restrictions regarding the use of the point entity in this entity. They are:

a. Two point entities cannot appear consecutively in the defining list.

b. If a point entity and a curve entity are adjacent in the defining list, then the coordinates of the point entity must agree with the coordinates of the terminate point of the curve entity whenever the curve entity precedes the point entity, and must agree with the coordinates of the start point of the curve entity whenever the curve entity follows the point entity.

c. A composite curve cannot consist of a point entity alone.

3.3.5 An example of a composite curve entity is shown in Figure 3-2 .

3.3.6 Directory Data

ENTITY TYPE NUMBER : 102

3.3.7 Parameter Data

Parameter	Value	Format	Comment
1	N	Integer	Number of entities
2	DE	Pointer	
.	.	.	Pointers to directory entries for the constituent entities
.	.	.	
.	.	.	
N+1	DE	Pointer	
N+2	NA	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
N+3	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
N+2+NA	DE	Pointer	
N+3+NA	M	Integer	Number of properties

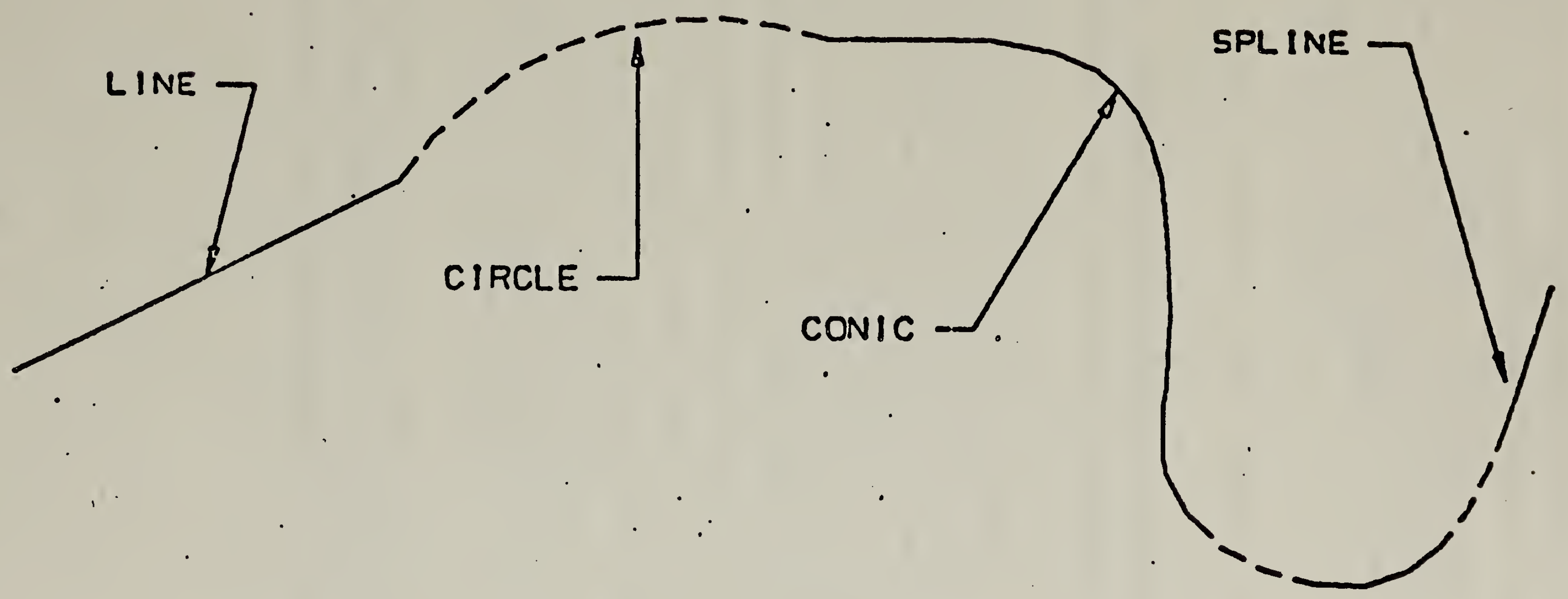


FIG. 3-2 EXAMPLE OF THE COMPOSITE CURVE ENTITY

Parameter	Value	Format	Comment
N+4+NA	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
N+3+NA+M	DE	Pointer	

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 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 first, 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 distinquished from its complementary elliptical arc, the direction of the desired elliptical arc must be counterclockwise with respect to its major and minor axes. 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 3.1.15 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 In the event that a parameterization is required but not given, the default parameterization is:

C(t) = Center + (rotation) * A(t)

where: parabola $A(t) = (t^2/a, t)$

$$-\infty < t_0 \leq t \leq t_1 < \infty$$

ellipse A(t) = (a*cos t, b*sin t)

$$t_0 \leq t \leq t_1$$
$$0 \leq t_0 \leq 2\pi$$
$$t_0 \leq t_1 \leq t_0 + 2\pi$$

hyperbola A(t) = (a*sec t, b*tan t)

$$-\pi/2 < t_0 \leq t \leq t_1 < \pi/2$$

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

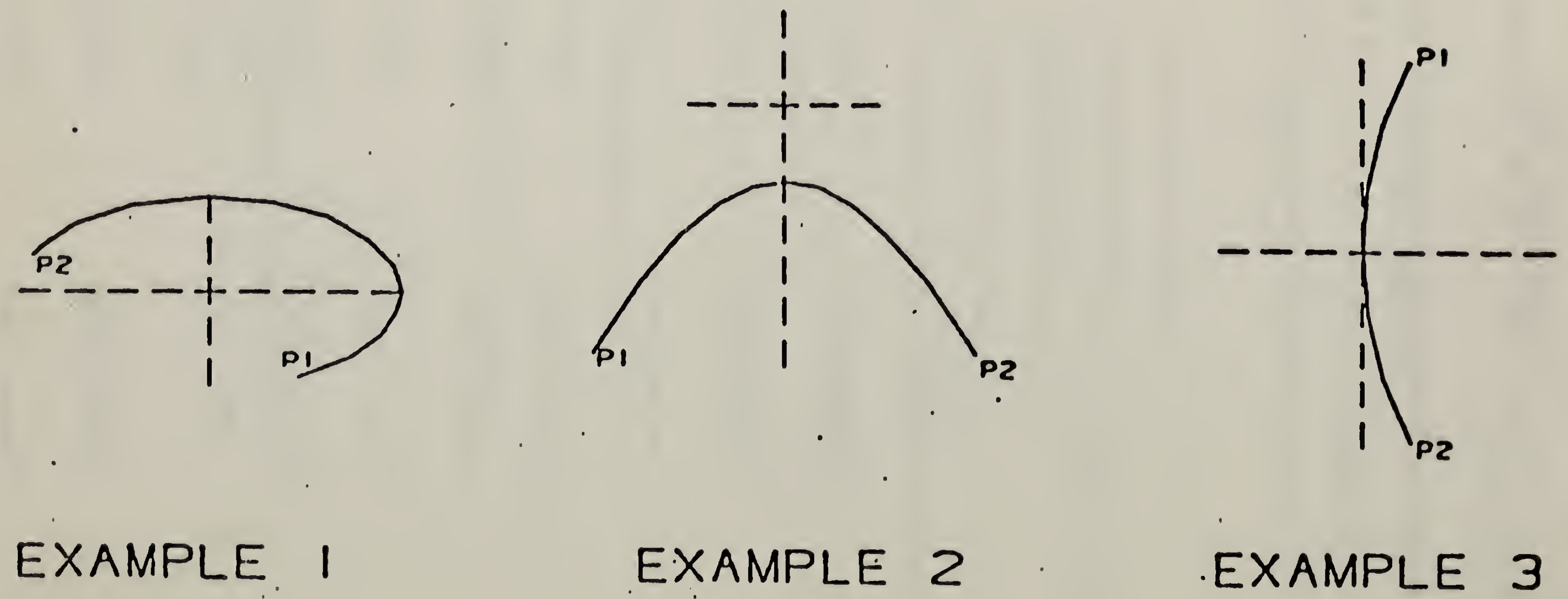
FORM	Meaning
0	Form of parent conic curve must be determined from conic equation.
1	Parent conic curve is an ellipse (See example 1, Figure 3-3).
2	Parent conic curve is a hyperbola (See example 2, Figure 3-3).
3	Parent conic curve is a parabola (See example 3, Figure 3-3).

3.4.8 Directory Data

ENTITY TYPE NUMBER : 104

3.4.9 Parameter Data

Parameter	Value	Format	Comment
1	A	Floating Point	Conic Coefficient
2	B	Floating Point	Conic Coefficient
3	C	Floating Point	Conic Coefficient
4	D	Floating Point	Conic Coefficient
5	E	Floating Point	Conic Coefficient
6	F	Floating Point	Conic Coefficient
7	ZT	Floating Point	Parallel ZT Displacement of Conic Arc from XT, YT plane
8	X1	Floating Point	Start Point Abscissa
9	Y1	Floating Point	Start Point Ordinate
10	X2	Floating Point	Terminate Point Abscissa



EXAMPLE 1

EXAMPLE 2

EXAMPLE 3

FIG. 3-3 EXAMPLES OF THE CONIC ARC ENTITY

11	Y2	Floating Point	Terminate Point Ordinate
12	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
13	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
12+N	DE	Pointer	
13+N	M	Integer	Number of properties
14+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
13+N+M	DE	Pointer	

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 IF.

3.5.1 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.)

3.5.2 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. (IF=1)
2	Data points in the form of coordinate triples. (IF=2)
3	Data points in the form of sextuples. (IF=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. (IF=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). (IF=2)
13	Data points in the form of sextuples which represent the vertices of a piecewise linear curve (piecewise linear string is sometimes used) (IF=3)
20	Centerline through points (IF=1)

21	Centerline through circle centers (IF=1)
31	Section Form 31 (IF=1)
32	Section Form 32 (IF=1)
33	Section Form 33 (IF=1)
34	Section Form 34 (IF=1)
35	Section Form 35 (IF=1)
36	Section Form 36 (IF=1)
37	Section Form 37 (IF=1)
38	Section Form 38 (IF=1)
40	Witness Line (IF=1)
63	Simple Closed Area (IF=1)

3.5.3 Refer to the centerline, section, and witness line entities in Section 4 of this Specification for examples of the Form Numbers in the range 20-40. Each of these annotation entities contains a description of how the associated copious data are to be interpreted.

3.5.4 Directory Data

ENTITY TYPE NUMBER : 106

3.5.5 Parameter Data

Parameter	Value	Format	Comment
1	IF	Integer	Interpretation flag IF=1; x, y pairs, common z IF=2; x, y, z coordinates IF=3; x, y, z coordinates and i, j, k vector
2	N	Integer	Number of 2 - tuples, 3 - tuples, or 6 - tuples
3	Data Points	Floating Point	If IF=1, K=3+2N. (In this case, this third parameter is a ZT displacement.) If IF=2, K=2+3N. If IF=3, K=2+6N.
.			
.			
.			
K			
K+1	NA	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
K+2	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.		
K+1+NA	DE	Pointer	
K+2+NA	M	Integer	Number of properties
K+3+NA	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
K+2+NA+M	DE	Pointer	

3.6 Plane Entity

The plane entity can be used to represent an unbounded plane, as well as a bounded portion of a plane. No preferred positioning with respect to definition space is assumed. In either of the above cases, the plane is defined within definition space by means of the coefficients A, B, C, D, where

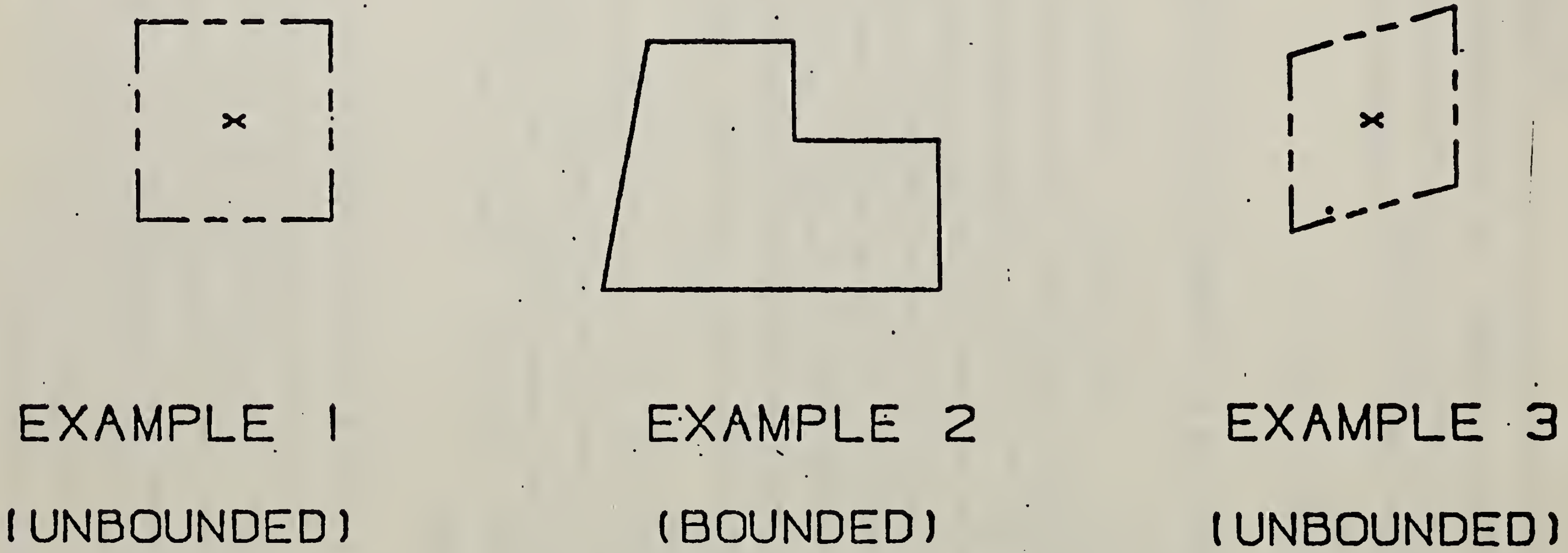
A*XT + B*YT + C*ZT = D

for each point lying in the plane, and having definition space coordinates (XT, YT, ZT).

3.6.1 The definition space coordinates of a point, as well as a size parameter, can be specified in order to assist in defining a system-dependent display symbol. These values are parameter data entries six through nine, respectively. This information, together with the four coefficients defining the plane, provides sufficient information relative to definition space in order to be able to position the display symbol. (In Examples 1 and 3 of Figure 3-4, the dashed curve and the crosshair together constitute the display symbol.) Setting the size parameter to zero indicates that a display symbol is not intended.

3.6.2 The case of a bounded portion of a fixed plane is indicated by the existence of a pointer to a closed curve lying in the plane. This is parameter five. The only self-coincident points for this curve are the start point and the terminate point. Setting this pointer to zero indicates the case of an unbounded plane.

3.6.3 The case of a bounded portion of a fixed plane minus some portion(s) of that plane, such as those shown in Figure 3-5, may be expressed through the use of the Single Parent Associativity (Type 402, Form 9) where the outer closed curve defines the parent bounded plane and each internal closed curve defines some child bounded plane to be subtracted from the parent. Each of these planes (parent and child) is a separate plane entity in the IGES file and has a backpointer to the associativity structure. The child plane entity will have a subordinate entity switch class of 01.



EXAMPLE 1

(UNBOUNDED)

EXAMPLE 2

(BOUNDED)

EXAMPLE 3

(UNBOUNDED)

FIG. 3-4 EXAMPLES OF THE PLANE ENTITY

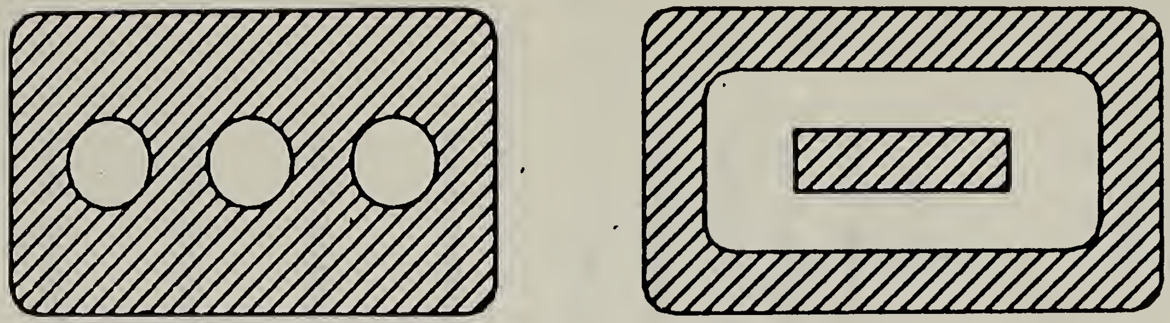


FIG. 3-5 SINGLE PARENT ASSOCIATIVITY AS USED WITH A COLLECTION OF BOUNDED PLANES

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

FORM	Meaning
+1	Bounded planar portion is considered positive.
-1	Bounded planar portion is considered negative (hole).

3.6.5 Directory Entry Data

ENTITY TYPE NUMBER : 108

3.6.6 Parameter Data

Parameter	Value	Format	Comment
1	A	Floating Point	Coefficients of Plane
2	B	Floating Point	
3	C	Floating Point	
4	D	Floating Point	
5	PTR	Pointer	Pointer to directory entry of closed curve entity or 0
6	X	Floating Point	XT coordinate of location point for display symbol
7	Y	Floating Point	YT coordinate of location point for display symbol
8	Z	Floating Point	ZT coordinate of location point for display symbol
9	SIZE	Floating Point	Size parameter for display symbol
10	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
11	DE	Pointer	Pointers to associativities or general notes

.	.	.	
.	.	.	
.	.	.	
10+N	DE	Pointer	
11+N	M	Integer	Number of properties
12+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
11+N+M	DE	Pointer	

3.7 Line Entity

A line is a bounded, connected portion of a parent straight line which consists of more than one point. No preferred positioning with respect to definition space is assumed.

A line is defined by its end points. Each end point is specified relative to definition space by a triple of coordinates. With respect to definition space, a direction is associated with the line by considering the start point to be listed first and the terminate point second.

The direction of the line with respect to model space is determined by the original direction of the line within definition space, in conjunction with the action of the transformation matrix on the line. Examples of the line entity are shown in Figure 3-6.

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

$$C(t) = P_1 + t(P_2-P_1) \quad 0 \leq t \leq 1$$

3.7.2 Directory Entry Data

ENTITY TYPE NUMBER : 110

3.7.3 Parameter Data

Parameter	Value	Format	Comment
1	X1	Floating Point	Start Point P1
2	Y1	Floating Point	
3	Z1	Floating Point	
4	X2	Floating Point	Terminate Point P2
5	Y2	Floating Point	
6	Z2	Floating Point	
7	N	Integer	Number of back pointers (to associativity entities/text pointers (to general note entities)

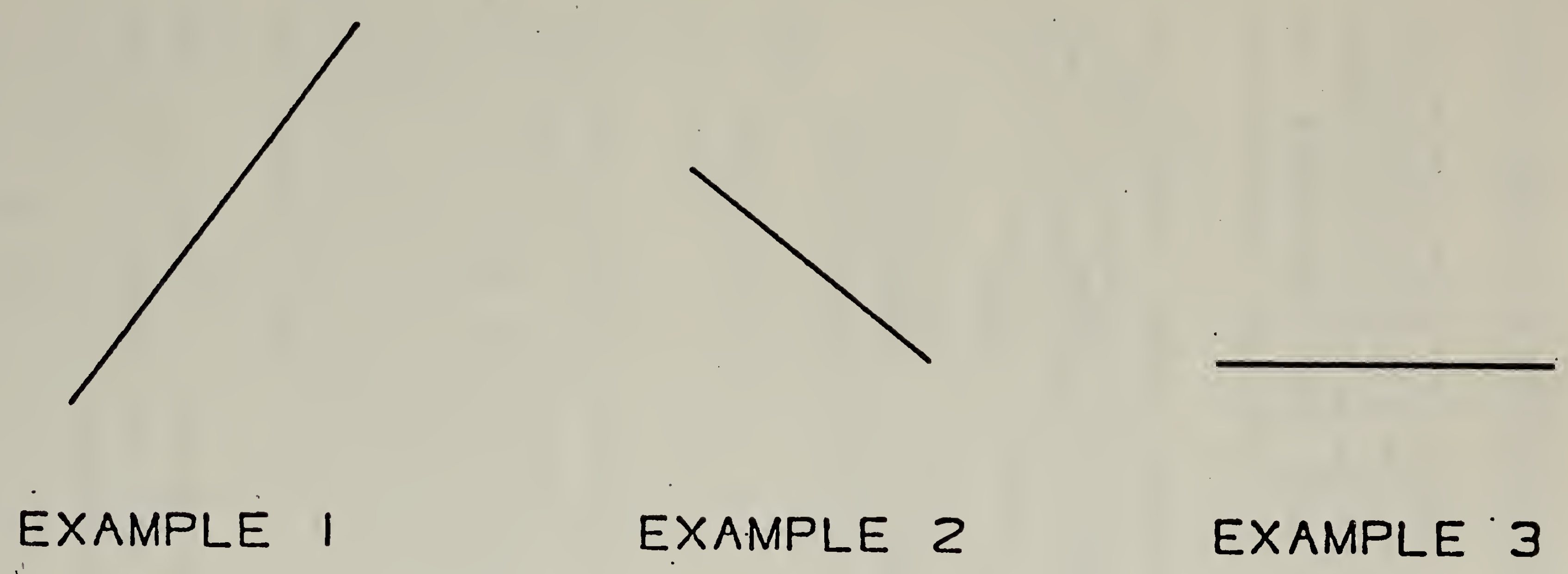


FIG. 3-6 EXAMPLES OF THE LINE ENTITY

8	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
7+N	DE	Pointer	
8+N	M	Integer	Number of properties
9+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
8+N+M	DE	Pointer	

3.8 Parametric Spline Curve Entity

(Consult Appendix A for additional mathematical details)

The parametric spline curve is a sequence of parametric polynomial segments of degree 1, 2, or 3 in at least one of the defining equations. This entity also represents the various splines used in present day systems (linear, quadratic, cubic, Wilson-Fowler, modified Wilson-Fowler and B-splines). The CTYPE value in Parameter 1 indicates the type of curve under consideration.

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

$t(i)\leq u\leq t(i+1)$, $i=1,\ldots,$

$s=u-t(i)$

3.8.2 If the spline is planar, it should be parametrized in terms of the X and Y polynomials only. The Z polynomial will then be zero except for the AZ(i) term which indicates the Z-depth in definition space. To enable determination of the terminate point and derivatives without computing the polynomials, a dummy N+1st polynomial segment is included in the entity. The parameter t(N+2) is not provided for this segment since the terminate point of the dummy segment and the derivatives at that point are implied by the N+1st segment coefficients.

3.8.3 There is a parameter H which specifies the degree of continuity at the breakpoints with respect to arc length. H=0 means that the curve is continuous but is not necessarily differentiable with respect to arc length at a breakpoint. H=1 means that the curve is differentiable with respect to arc length at each breakpoint but may have an undefined curvature at a breakpoint. H = 2 means that the curve has well-defined curvature at all breakpoints with respect to arc length parameterization. Parameterization with respect to arc length is chosen to express H because arc length moves along the curve at unit speed and is therefore able to detect jaggedness in the curve.

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

3.8.5 Directory Data

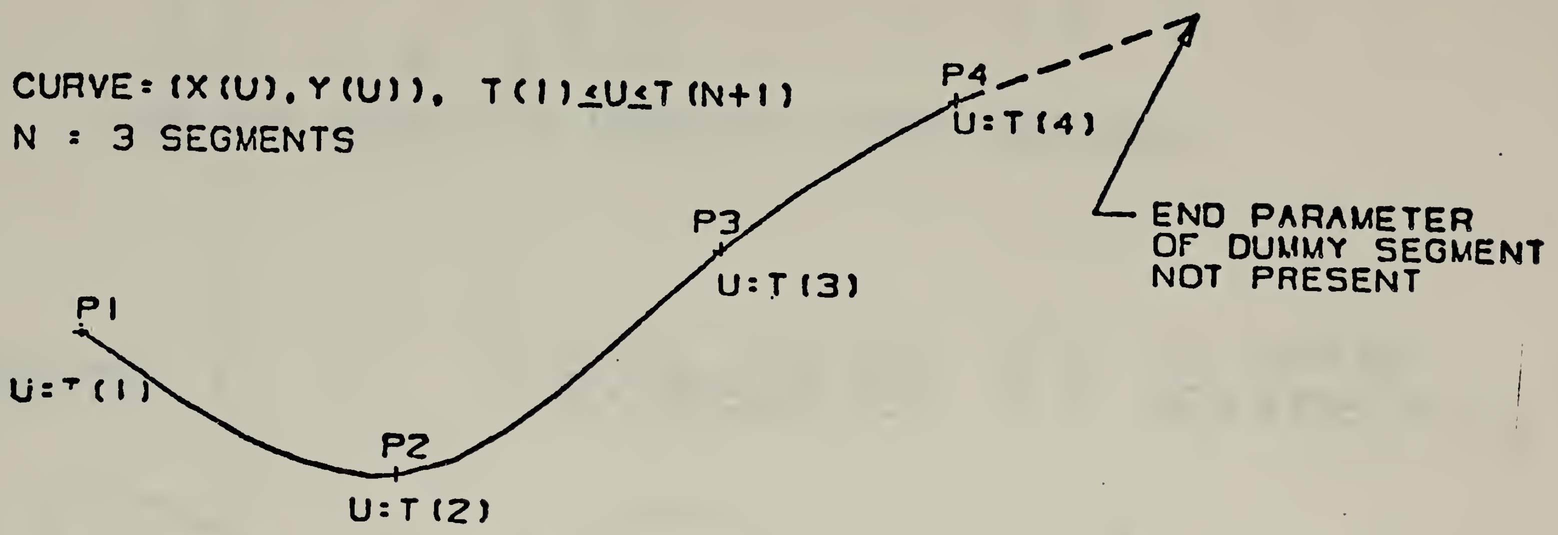
ENTITY TYPE NUMBER : 112

3.8.6 Parameter Data

Parameter	Value	Format	Comment
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

CURVE: $(X(U), Y(U))$, $T(1) \le U \le T(N+1)$

N : 3 SEGMENTS



FOR SEGMENT NUMBER 2:

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

P1: (AX(1), AY(1))

P2: (AX(2), AY(2))

P3: (AX(3), AY(3))

P4: (AX(4), AY(4))

DERIVATIVES AT P4: (BX(4), BY(4))

FIG. 3-7 EXAMPLE OF THE PARAMETRIC SPLINE (2D) CURVE ENTITY

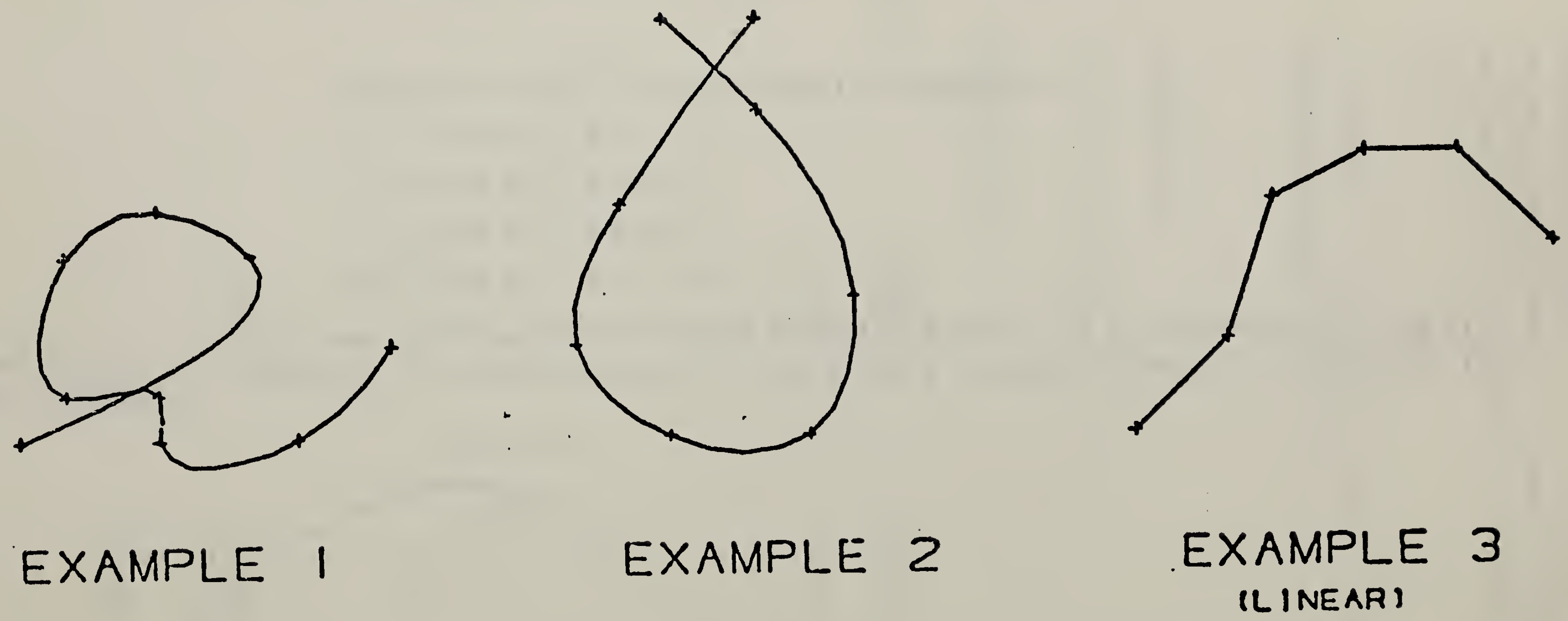


FIG. 3-8 EXAMPLES OF PARAMETRIC SPLINE CURVE ENTITY

Parameter	Value	Format	Comment
5	T(1)	Floating Point	Break points of piecewise polynomial
.	.		
.	.		
.	.		
5+N	T(N+1)		
6+N	AX(1)	Floating Point	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
	.		
	.		

(A dummy segment included only to indicate the value of the spline and its derivatives at the end point)

			Last polynomial
6+13*N	AX(N+1)	Floating Point	X value at endpoint
	BX(N+1)		X first derivative
	CX(N+1)		X second derivative/2!

	DX(N+1)		X third derivative/3!
	AY(N+1)		Y polynomial
	BY(N+1)		
	CY(N+1)		
	DY(N+1)		
	AZ(N+1)		Z polynomial
	BZ(N+1)		
	CZ(N+1)		
	DZ(N+1)		

TOTAL ENTRIES (TE) = 5+N+12*(N+1)

TE+1	NA	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
TE+2	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
TE+1+NA	DE	Pointer	
TE+2+NA	M	Integer	Number of properties
TE+3+NA	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
TE+2+NA+M	DE	Pointer	

3.9 Parametric Spline Surface Entity

(Consult Appendix A for additional mathematical details)

The parametric spline surface is a grid of parametric polynomial patches. Because of its generality, this entity also represents the various surfaces used in present day systems (Coons, Bezier, B-spline, Ferguson, Cartesian product surfaces). PTYPE in the Parameter Data Section indicates the type of patch under consideration.

3.9.1 The MxN grid of patches is defined by the u breakpoints tu(1), ... , tu(M+1) and the v breakpoints tv(1), ... , tv(N+1). The coordinates of the points in each of the patches are given by the general bicubic polynomials (given here for the (i, j) Patch).

$$\begin{aligned} X(u,v) \quad = \quad & AX(i,j)+BX(i,j)*s+CX(i,j)*s^2+DX(i,j)*s^3 \\ + \quad & EX(i,j)*t+FX(i,j)*t*s+GX(i,j)*t*s^2+HX(i,j)*t*s^3 \\ + \quad & KX(i,j)*t^2+LX(i,j)*t^2*s+MX(i,j)*t^2*s^2+NX(i,j)*t^2*s^3 \\ + \quad & PX(i,j)*t^3+QX(i,j)*t^3*s+RX(i,j)*t^3*s^2+SX(i,j)*t^3*s^3 \end{aligned}$$

$$Y(u,v) \quad = \quad \ldots$$

$$Z(u,v) \quad = \quad \ldots$$

where

$tu(j) \leq u \leq tu(j+1), \qquad j=1, \ldots , M$

$s=u-tu(j)$

and

$tv(i) \leq v \leq tv(i+1), \qquad i=1, \ldots , N$

$t=v-tv(i).$

3.9.2 To provide edge values and derivatives without evaluating polynomials, an additional dummy row and column of patches is included in the entity.

3.9.3 An example of the bicubic surface is shown in Figure 3-9.

3.9.4 Directory Data

ENTITY TYPE NUMBER : 114

3.9.5 Parameter Data

Parameter	Value	Format	Comment
1	STYPE	Integer	Spline Type (1=Linear 2=Quadratic 3=Cubic 4=Wilson-Fowler 5=Modified Wilson-Fowler 6 = B spline)
2	PTYPE	Integer	Patch Type (1=Cartesian Product 0=Unspecified)
3	M	Integer	Number of u segments
4	N	Integer	Number of v segments
5	TU(1)	Floating Point	Breakpoints in u (u values of grid lines)
.	.		
.	.		
.	.		
5+M	TU(M+1)		

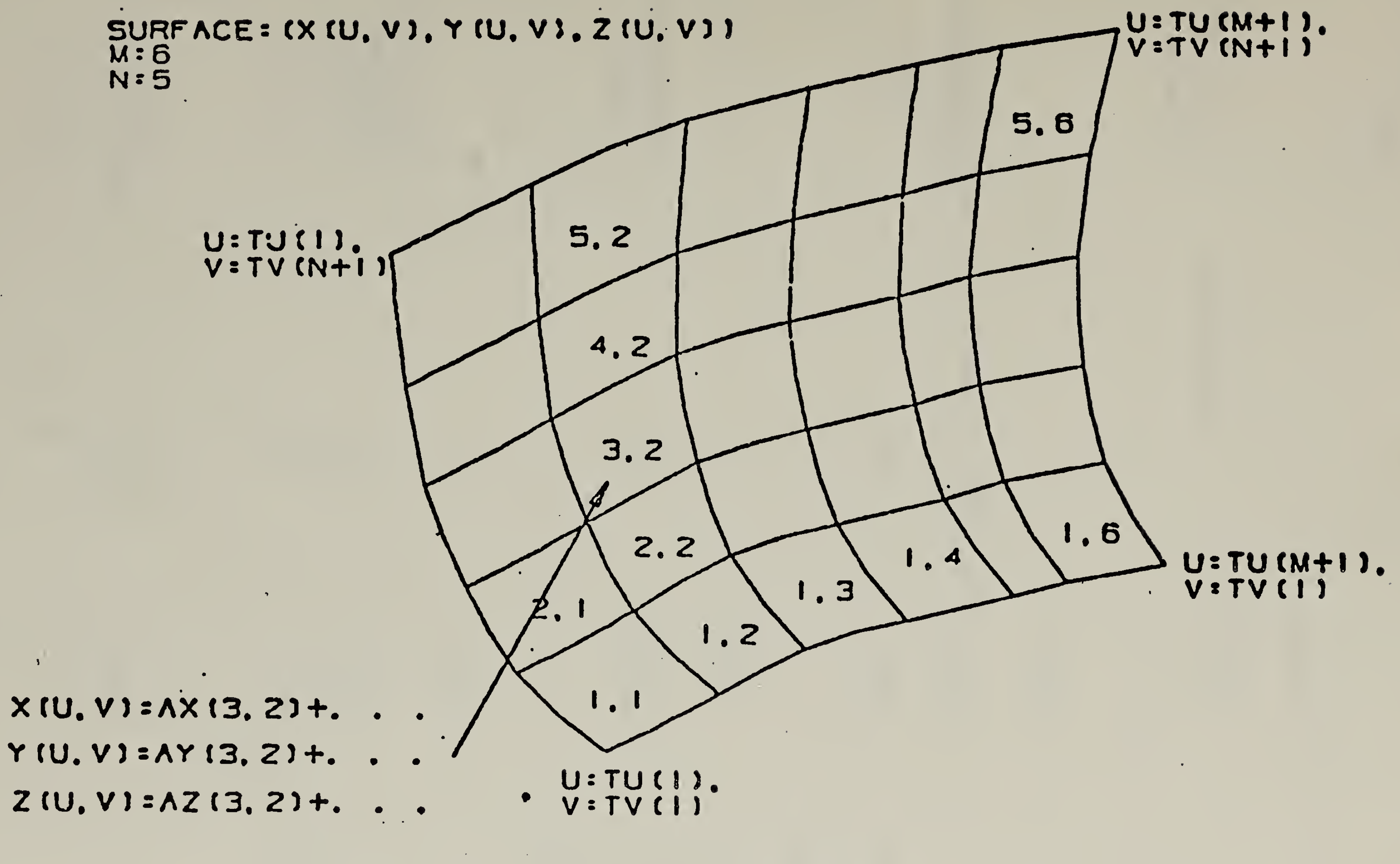


FIG. 3-9 EXAMPLE OF THE PARAMETRIC SPLINE SURFACE ENTITY

Parameter	Value	Format	Comment
6+M	TV(1)	Floating Point	Breakpoints in v (v values of grid lines)
	.		
	.		
	.		
6+M+N	TV(N+1)		
7+M+N	AX(1,1)	Floating Point	X Coefficients of (1,1) Patch
.	.		
.	.		
.	.		
22+M+N	SX(1,1)		
23+M+N	AY(1,1)		Y Coefficients of (1,1) Patch
.	.		
.	.		
.	.		
38+M+N	SY(1,1)		
39+M+N	AZ(1,1)		Z Coefficients of (1,1) Patch
.	.		
.	.		
.	.		
54+M+N	SZ(1,1)		
.	.		Coefficients of (1,2) Patch
.	.		
.	.		
			.
			.
			.
			(1,N+1)
			(2,1) Patch
			.
			.
			.

Parameter	Value	Format	Comment
			(2, N+1)
			.
			.
			.
			(M+1,1) Patch
			.
			.
			.
			(M+1,N+1)
TOTAL ENTRIES = 6+M+N+48*(M+1)*(N+1)=TE			
TE+1	NA	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
TE+2	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
TE+1+NA	DE	Pointer	
TE+2+NA	MA	Integer	Number of properties
TE+3+NA	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
TE+2+NA+MA	DE	Pointer	

3.10 Point Entity

A point is defined by its coordinates in definition space. Examples of the point entity are shown in Figure 3-10.

3.10.1 Directory Data

ENTITY TYPE NUMBER : 116

3.10.2 Parameter Data

Parameter	Value	Format	Comment
1	X	Floating Point	Coordinates
2	Y	Floating Point	of point
3	Z	Floating Point	
4	PTR	Pointer	Pointer to directory entry of subfigure instance specifying the display symbol. If 0, no display symbol specified.
5	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
6	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
5+N	DE	Pointer	
6+N	M	Integer	Number of properties
7+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Pointer	

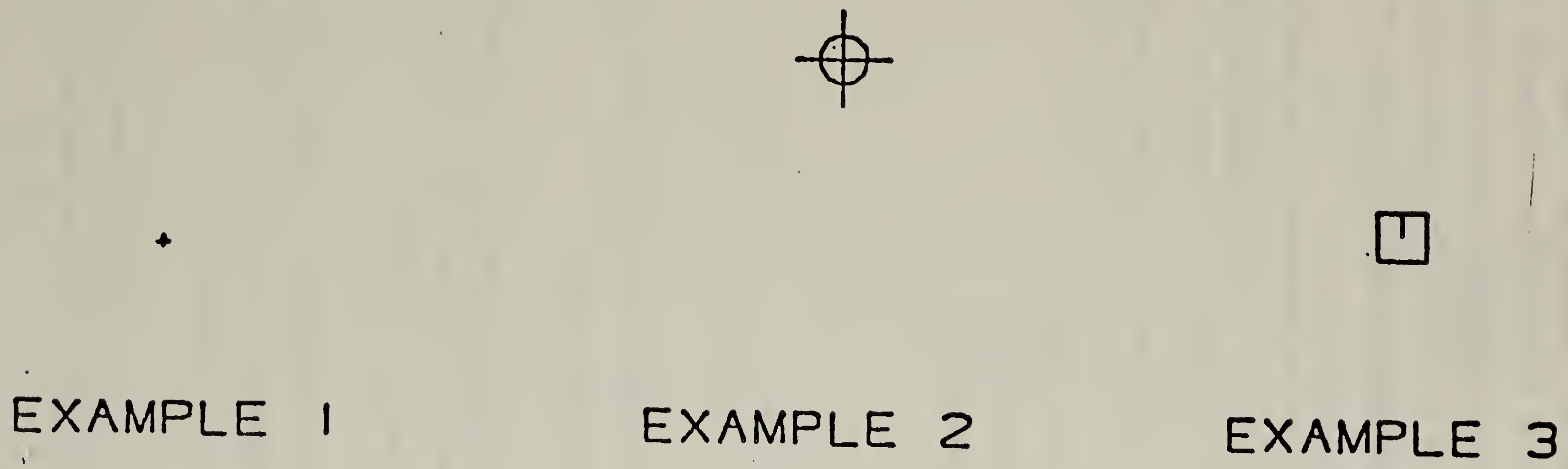


FIG. 3-10 EXAMPLES OF THE POINT ENTITY

3.11 Ruled Surface Entity

A ruled surface is formed by moving a line connecting points of equal relative arc length (Form 0) or equal relative parametric value (Form 1) on two parametric curves from a start point to a terminate point on the curves. The parametric curves may be points, lines, circles, conics, parametric splines, or any parametric curves defined in IGES (both planar and non-planar).

3.11.1 Assume the two curves are expressed as the parametric functions ($C1_X(t)$, $C1_Y(t)$, $C1_Z(t)$) and ($C2_X(s)$, $C2_Y(s)$, $C2_Z(s)$), with some range $a \leq t \leq b$ and $c \leq s \leq d$, then the coordinates of the points on the ruled surface can be written as

$$X(u,v) = (1-v)*C1_X(t)+v*C2_X(s)$$
$$Y(u,v) = (1-v)*C1_Y(t)+v*C2_Y(s)$$
$$Z(u,v) = (1-v)*C1_Z(t)+v*C2_Z(s)$$

where

$0 \leq u \leq 1,$

$0 \leq v \leq 1$

$t = a+u*(b-a)$

$s = c+u*(d-c)$

C1(t) and C2(s) are said to be of equal relative parametric value if t and s are evaluated at the same u value.

3.11.2 The above set of equations corresponds to the case DIRFLG=0. In this case, the first point of curve 1 is joined to the first point of curve 2 and the last point of curve 1 to last point of curve 2.

If DIRFLG=1, then the first point of curve 1 is joined to the last point of curve 2, the last point of curve 1 to the first point of curve 2, and the parameter S is given by $S = (1-u)*b$.

3.11.3 If DEVFLG=1, then the surface is a developable surface; if DEVFLG=0, the surface may or may not be a developable surface.

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

FORM	MEANING
0	Equal relative arc length
1	Equal relative parametric values.

The default is FORM 0.

3.11.5 An example of the Ruled Surface Entity is shown in Figure 3-11. Additional examples are shown in Figure 3-12.

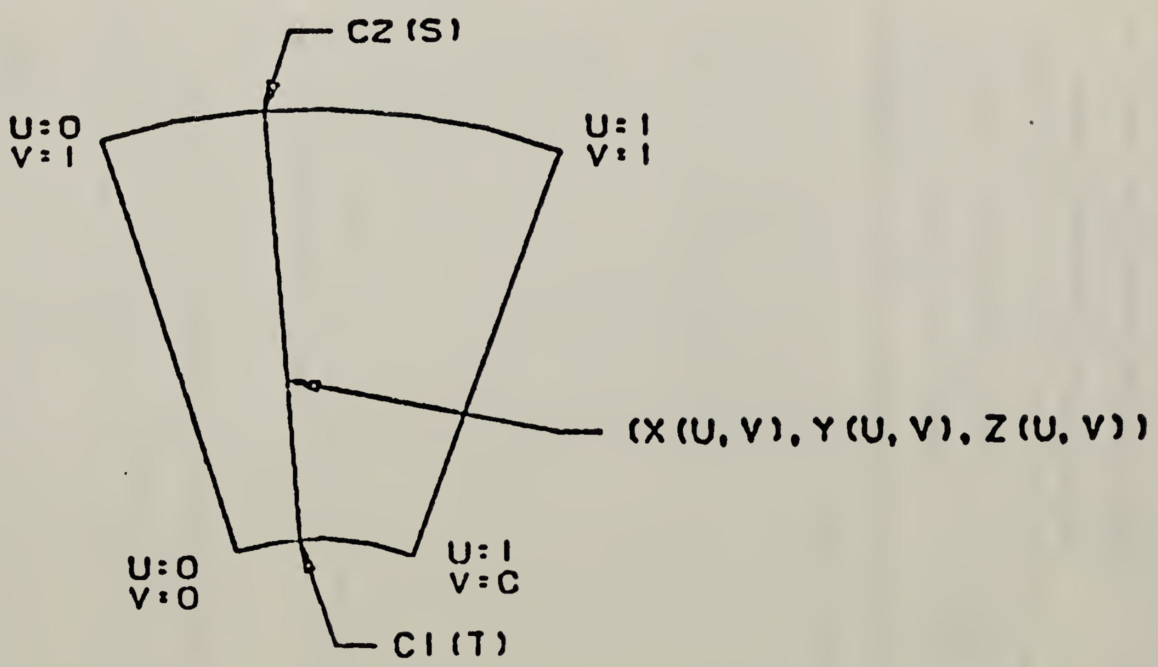


FIG. 3-11 EXAMPLE OF THE RULED SURFACE ENTITY

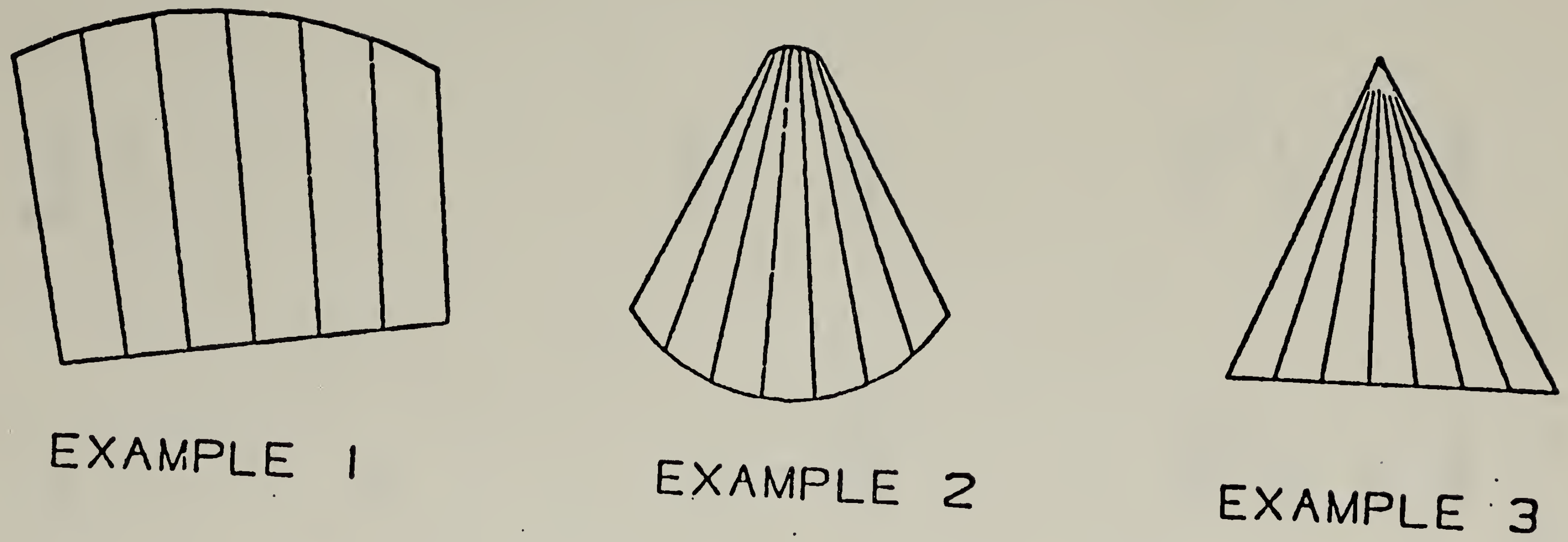


FIG. 3-12 EXAMPLES OF THE RULED SURFACE ENTITY

3.11.6 Directory Data

ENTITY TYPE NUMBER : 118

3.11.7 Parameter Data

Parameter	Value	Format	Comment
1	DE1	Pointer	Pointer to first curve
2	DE2	Pointer	Pointer to second curve
3	DIRFLG	Integer	Direction flag (0=join first to first, last to last 1=join first to last, last to first)
4	DEVFLG	Integer	Developable surface flag (1=Developable, 0=Possibly not)
5	N	Integer	Number of back pointers (to associativity entities)/ text pointers (to general note entities)
6	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
5+N	DE	Pointer	

Parameter	Value	Format	Comment
6+N	M	Integer	Number of properties
7+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Pointer	

3.12 Surface of Revolution Entity

A surface of revolution is defined by an axis of rotation (which must be a line entity), a generatrix, and start and terminate rotation angles. The surface is created by rotating the generatrix about the axis of rotation through the start and terminating angles. Since the axis of rotation is a line entity, it contains in its Parameter Data section the coordinates of its start point first, followed by the coordinates of its terminate point. This gives the axis of rotation an implicit direction. Using this direction, we can place the eye at the terminate point and look toward the face of a clock centered at the start point and lying in a plane perpendicular to the axis of rotation. This enables one to talk about clockwise and counterclockwise rotations, and this is the method used to measure the angles of rotation (counterclockwise is in the positive direction). The generatrix may be a conic arc, line, circular arc, parametric spline curve, or composite curve.

3.12.1 Examples of surface of revolution entities are shown in Figure 3-13.

3.12.2 The start and terminate angles of the surface can be explained by geometric construction. Refer to Figure 3-14 and the following:

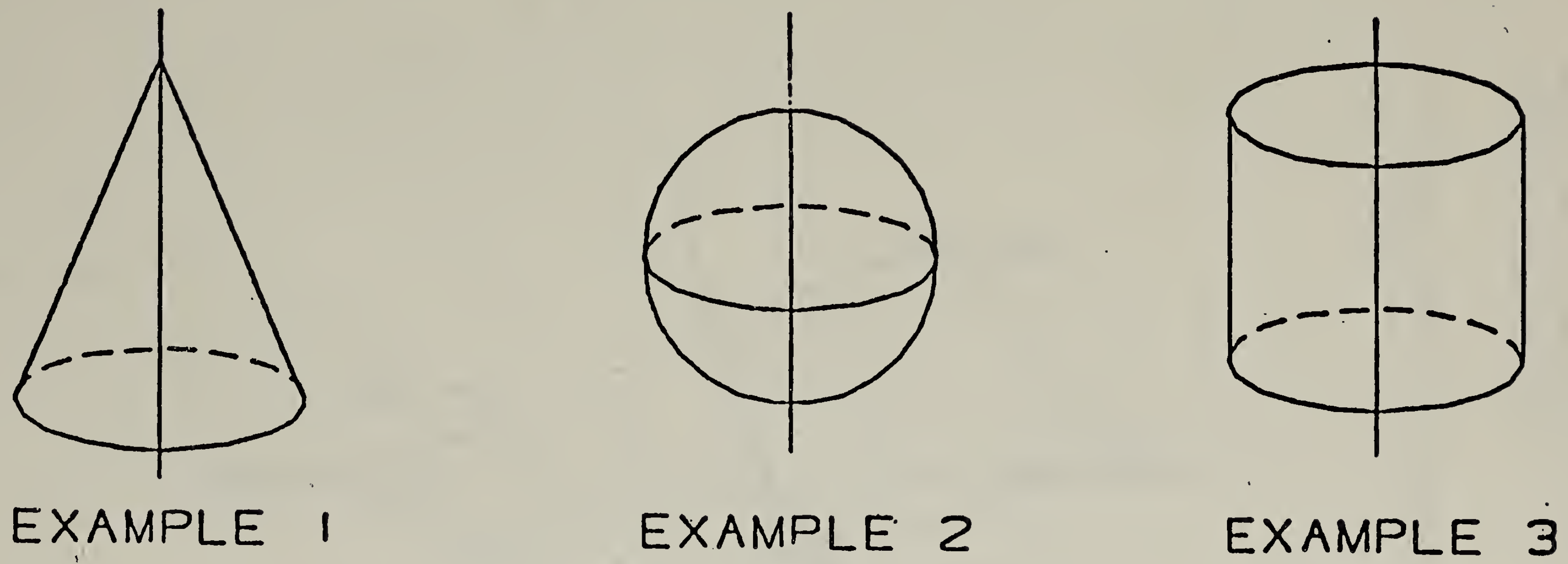
a. Select a point on the generatrix which does not lie on the axis of rotation; label the point P1.

b. Construct a line through P1 such that it is perpendicular to the axis of rotation extended; label this line L1.

c. Construct a plane PN1 containing L1 and perpendicular to the axis of rotation.

d. All rotations in the plane PN1 about the axis of rotation are applied counterclockwise according to the method described in 3.12.

e. Rotate counterclockwise the line L1 and the point selected from the generatrix the number of radians indicated in the start angle resulting in $L1_{SA}$. The location is labeled LOC1.



EXAMPLE 1

EXAMPLE 2

EXAMPLE 3

FIG. 3-13 EXAMPLES OF THE SURFACE OF REVOLUTION ENTITY

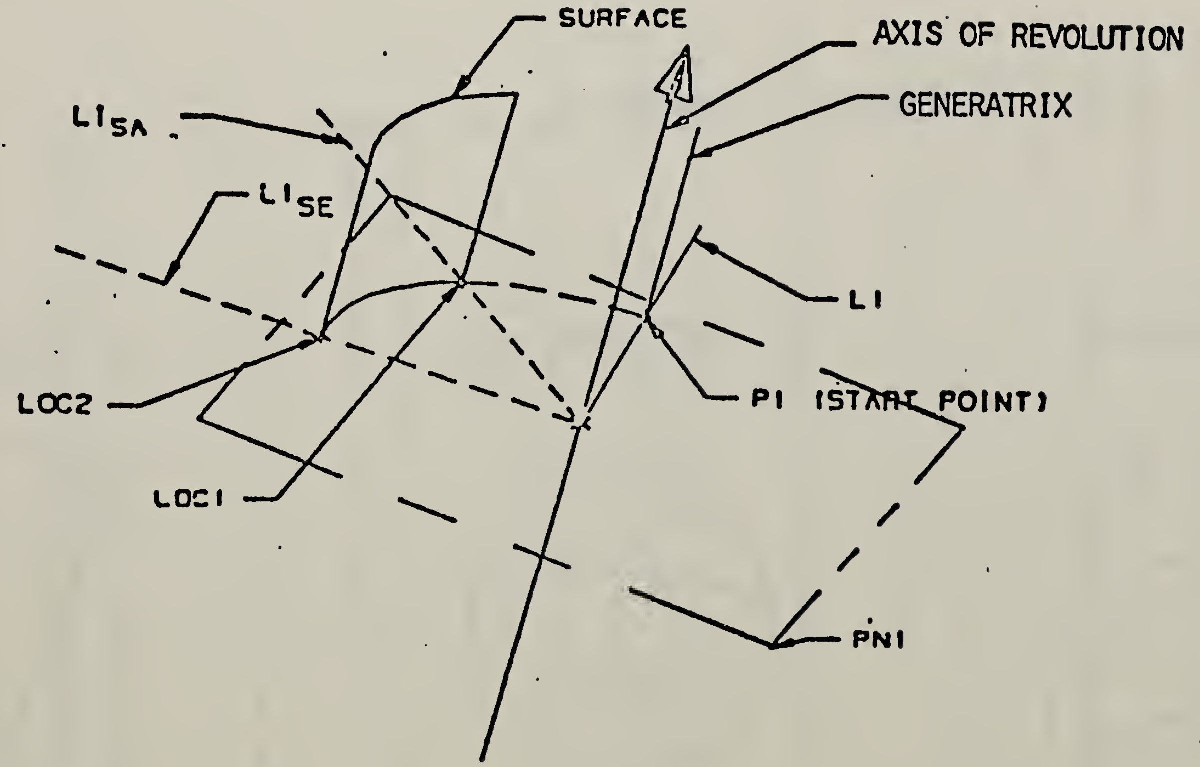


FIG. 3-14 SURFACE OF REVOLUTION START AND TERMINATING ANGLES

f. Rotate counterclockwise the line L1 and the point selected from the generatrix an additional number of radians given by the terminate angle minus the start angle resulting in $L1_{SE}$. The second location of the point is labeled LOC2.

g. The resulting surface is that generated by rotating the generatrix from LOC1 to LOC2.

3.12.3 Directory Data

ENTITY TYPE NUMBER : 120

3.12.4 Parameter Data

Parameter	Value	Format	Comment
1	DE1	Pointer	Pointer to a line (axis of revolution)
2	DE2	Pointer	Pointer to generatrix
3	SA	Floating point	Start angle in radians
4	TA	Floating point	Terminate angle in radians
5	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
6	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
5+N	DE	Pointer	
6+N	M	Integer	Number of properties
7+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Pointer	

3.13 Tabulated Cylinder Entity

A tabulated cylinder is a surface formed by moving a line segment called the generatrix parallel to itself along a curve called the directrix. This curve may be a line, circular arc, conic arc, parametric spline curve, or composite curve.

3.13.1 It must be pointed out that different parameterizations of the generating curves will produce different parameterized surfaces, but the underlying point set surface will still be the same. Assuming a parameterization u on the directrix and v on the generatrix, both of which run from 0 to 1, we can express the points on the surface by

$$X(u,v) = CX(u)+v*(LX-CX(0))$$
$$Y(u,v) = CY(u)+v*(LY-CY(0))$$
$$Z(u,v) = CZ(u)+v*(LZ-CZ(0))$$

where $0 \leq u \leq 1$, $0 \leq v \leq 1$

and CX, CY, CZ represent the X, Y, Z components, respectively, along the directrix curve, while (CX(0), CY(0), CZ(0)) and (LX, LY, LZ) represent the coordinates of the start and terminate points, respectively, of the generatrix.

3.13.2 An example of the tabulated cylinder is shown in Figure 3-15.

3.13.3 Directory Data

ENTITY TYPE NUMBER : 122

3.13.4 Parameter Data

Parameter	Value	Format	Comment
1	DE1	Integer	Pointer to directrix curve
2	LX	Floating Point	Coordinates of the terminate point of the generatrix. The start point of the generatrix is identical with the start point of the directrix.

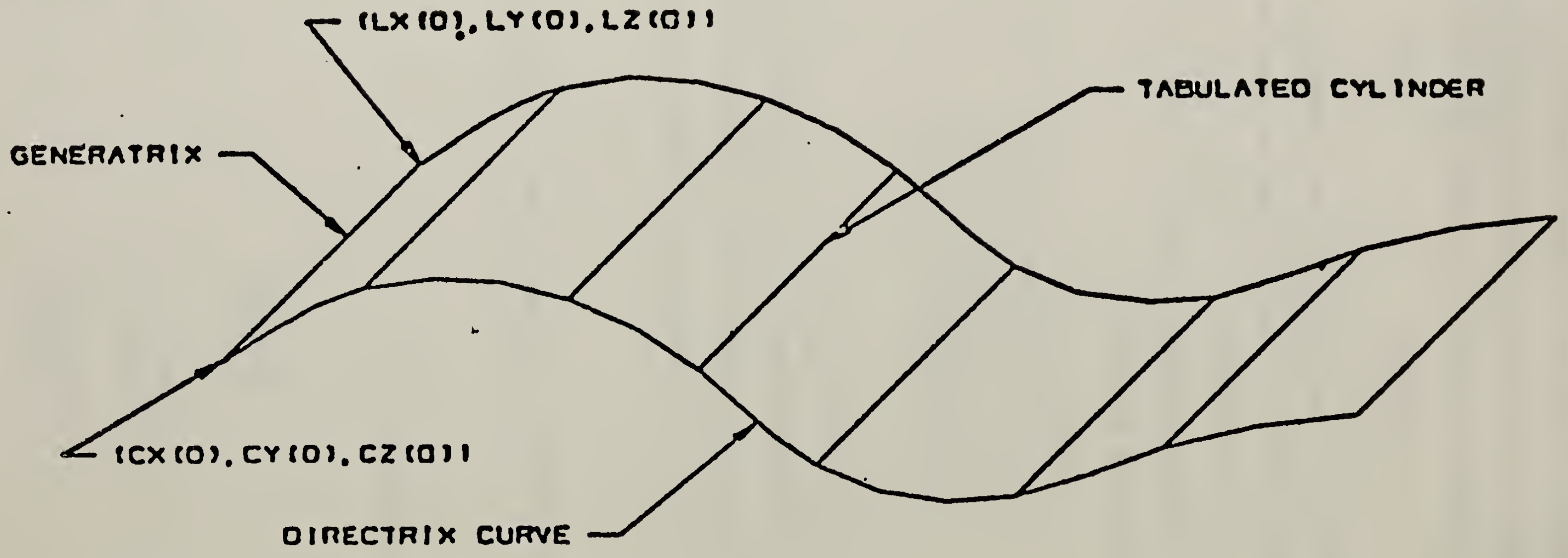


FIG. 3-15 EXAMPLE OF THE TABULATED CYLINDER ENTITY

Parameter	Value	Format	Comment
3	LY	Floating Point	
4	LZ	Floating Point	
5	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
6	DE	Pointer	Pointers to associativities or general notes.
.	.	.	
.	.	.	
.	.	.	
5+N	DE	Pointer	
6+N	M	Integer	Number cf properties
7+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Pointer	

3.14 Transformation Matrix Entity

The Transformation Matrix is an entity which contains a three by three rotation matrix R and a translation

$$\text{vector } T = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

The referencing entity will have a pointer value in its directory entry. The pointer will be to the directory entry of the transformation matrix. A zero pointer in the directory entry implies the identity rotation matrix with no translation vector as explained in 2.4.7.

The notation for the matrix operation is:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \cdot \begin{bmatrix} XT \\ YT \\ ZT \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

3.14.1 Field 15 of the directory entry accommodates a form number. For this entity, the options are as follows:

Form	Meaning
0	(Default) The transformation matrix moves the associated entity from definition space to model space by first applying the rotation R, then the translation T. The XT, YT, ZT coordinates refer to the definition space of the referencing entity. The matrix represents a right hand coordinate system. The column vectors (R_{1i},R_{2i},R_{3i}), i = 1, 2, 3 form an orthonormal system. (R_{11}, R_{21}, R_{31}) is the unit vector in the direction of the definition space X-axis. (R_{12}, R_{22}, R_{32}) is the unit vector in the direction of the definition space Y-axis. (R_{13}, R_{23}, R_{33}) is the unit vector in the direction of the definition space Z-axis.

10 The transformation matrix defines a cartesian coordinate system.

11 The transformation matrix defines a cylindrical coordinate system.

12 The transformation matrix defines a spherical coordinate system.

3.14.2 Form numbers 10, 11 and 12 are used for entities that require a reference to a specific coordinate system. When the transformation matrix is used to define specific coordinate systems, the R and T parameters are used as follows:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} Xn \\ Yn \\ Zn \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} Xm \\ Ym \\ Zm \end{bmatrix}$$

where the columns of the matrix are the unit vectorial directions of the Xn, Yn, and +Zn axes respectively of the local coordinate system. This relationship is defined below.

$$+Xn = R_{11}*i \quad + \quad R_{21}*j \quad + \quad R_{31}*k$$

$$+Yn = R_{12}i \quad + \quad R_{22}j \quad + \quad R_{32}k$$

$$+Zn = R_{13}i \quad + \quad R_{23}j \quad + \quad R_{33}k$$

The R and T values are defined in coordinate system m, the reference coordinate system. Xn, Yn and Zn are the coordinates defined in local coordinate system n while Xm, Ym, and Zm are the corresponding coordinates in reference coordinate system m.

3.14.3 Directory Data

ENTITY TYPE NUMBER : 124

3.14.4 Parameter Data

Parameter	Value	Format	Comment
1	R11	Floating Point	Top Row
2	R12	Floating Point	
3	R13	Floating Point	
4	T1	Floating Point	
5	R21	Floating Point	Second Row
6	R22	Floating Point	
7	R23	Floating Point	
8	T2	Floating Point	
9	R31	Floating Point	Third Row
10	R32	Floating Point	
11	R33	Floating Point	
12	T3	Floating Point	
13	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
14	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
13+N	DE	Pointer	
14+N	M	Integer	Number of properties
15+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
14+N+M	DE	Pointer	

3.15 Linear Path Entity

The linear path entity 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 collinear. 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.

3.15.1 Directory Data

ENTITY TYPE NUMBER: 106

3.15.2 Parameter Data

Parameter data for this entity is described in section 3.5.

3.16 Simple Closed Area Entity

A simple closed area entity is a bounded region of XY coordinate space represented by a set of points that form a series of connected linear segments. These segments must form a closed loop, i.e., the first point 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.

3.16.1 Directory Data

ENTITY TYPE NUMBER: 106

3.16.2 Parameter Data

Parameter data for this entity is described in section 3.5.

3.17 Flash Entity

A flash entity is a point in the ZT=0 plane that locates a specific instance of a particular closed area. That closed area can be defined in one of two ways. First, it can be an arbitrary closed area defined by any entity capable of defining a closed area. The points of this entity must all lie in the ZT=0 plane. Second, it can be a member of a predefined set of flash shapes. This definition is determined by the form number as follows:

Form:	0	defined by attached entity
	1	circular
	2	rectangle
	3	donut
	4	canoe

In the latter case, parameters 3 thru 5 of the flash entity control the final size of the flash. Figure 3-16 indicates the usage of those parameters for the specific flash forms. Those parameters are ignored in form zero.

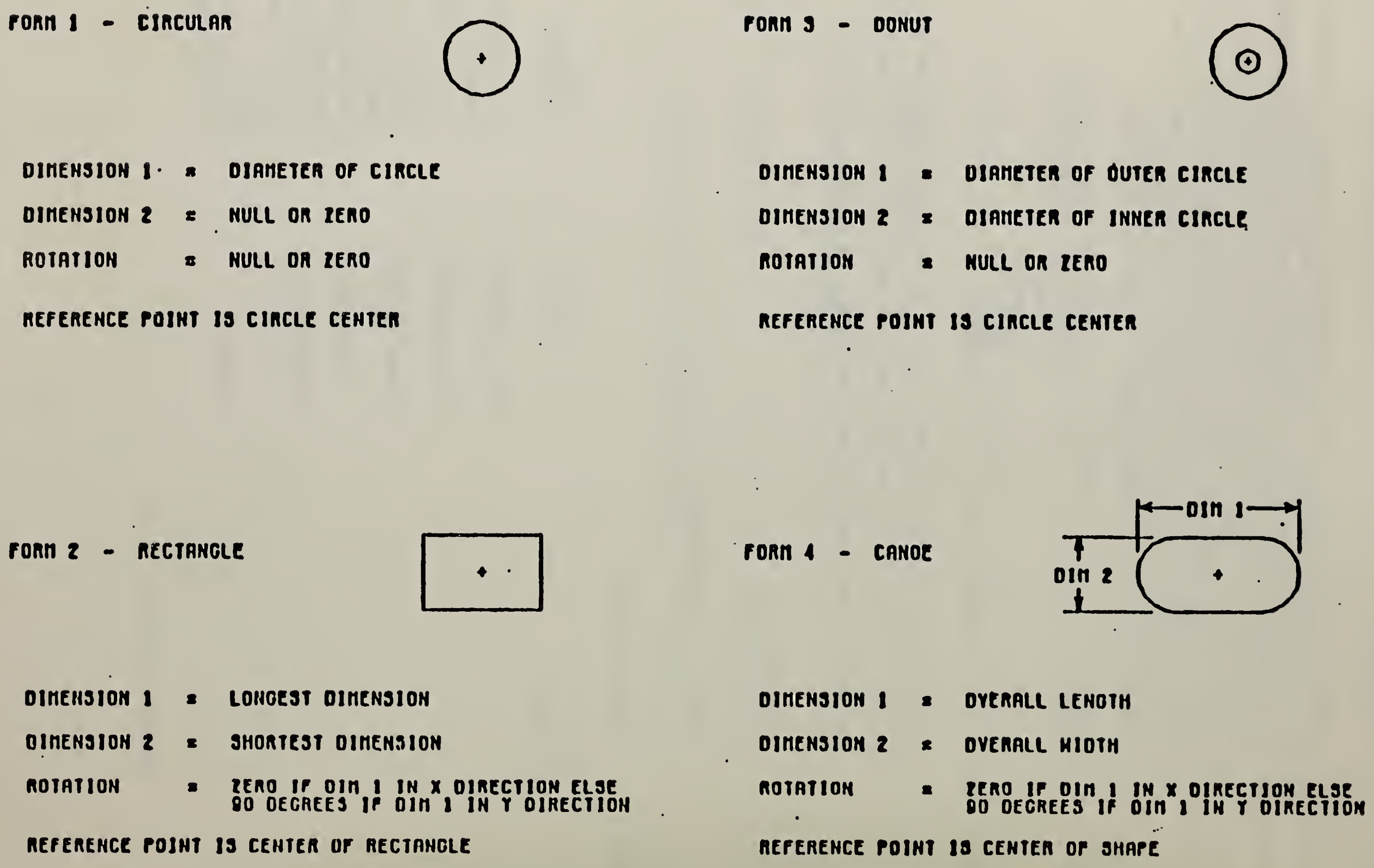
3.17.1 Directory Data

ENTITY TYPE NUMBER: 125

3.17.2 Parameter Data

Parameter	Value	Format	Comment
1	X	Floating Point	X reference of flash
2	Y	Floating Point	Y reference of flash
3	P1	Floating Point	First flash sizing parameter
4	P2	Floating Point	Second flash sizing parameter
5	R	Floating Point	Rotation of flash about reference point
6	DE	Integer	DE of defining entity (or 0)

Number of entries = K = 6



FORM 1 - CIRCULAR

DIMENSION 1	=	DIAMETER OF CIRCLE
DIMENSION 2	=	NULL OR ZERO
ROTATION	=	NULL OR ZERO

REFERENCE POINT IS CIRCLE CENTER

FORM 3 - DONUT

DIMENSION 1	=	DIAMETER OF OUTER CIRCLE
DIMENSION 2	=	DIAMETER OF INNER CIRCLE
ROTATION	=	NULL OR ZERO

REFERENCE POINT IS CIRCLE CENTER

FORM 2 - RECTANGLE

DIMENSION 1	=	LONGEST DIMENSION
DIMENSION 2	=	SHORTEST DIMENSION
ROTATION	=	ZERO IF DIM 1 IN X DIRECTION ELSE 90 DEGREES IF DIM 1 IN Y DIRECTION

REFERENCE POINT IS CENTER OF RECTANGLE

FORM 4 - CANOE

DIMENSION 1	=	OVERALL LENGTH
DIMENSION 2	=	OVERALL WIDTH
ROTATION	=	ZERO IF DIM 1 IN X DIRECTION ELSE 90 DEGREES IF DIM 1 IN Y DIRECTION

REFERENCE POINT IS CENTER OF SHAPE

FIG. 3-16 FLASH ENTITIES

]

K+1	NA	Integer	Number of associated entities
.			
.			
.			
K+NA+1	DE	Pointer	
K+NA+2	NP	Integer	Number of associated properties
.			
.			
.			
K+NA+NP+2	DE	Pointer	

See Figure 3-16.

3.18 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. A tutorial on the mathematical details of rational B-splines is given in a separate publication available from the National Bureau of Standards.

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.

Field 15 of the directory entry accommodates a form number. For this entity, the options are as follows:

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

If the curve lies entirely within a unique plane, the planar flag (PROP1) is set to 0, otherwise it is set to 1. If it is set to 0, the plane normal (parameters 15+A+4K through 17+A+4K) contain a unit vector normal to the plane containing the curve.

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 non-rational and PROP3 is set to 1.

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

3.18.1 Directory Data

ENTITY TYPE NUMBER: 126

3.18.2 Parameter Data

Parameter	Value	Format	Comment
1	K	Integer	Upper index of sum
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 - non-rational
6	PROP4	Integer	=0 - non-periodic =1 - periodic

(Let N = K - M + 1, A = N + 2M)

7	T(-M)	Floating Point	Knot Sequence
.	.		
.	.		
.	.		
7+A	T(N+M)		
8+A	W(0)	Floating Point	Weights
.	.		
.	.		
.	.		
9+A+K	W(K)		
10+A+K	X-coord. of P(0)	Floating Point	Control Points
11+A+K	Y-coord. of P(0)		
12+A+K	Z-coord. of P(0)		
.	.		
.	.		
.	.		
10+A+4K	X-coord. of P(K)		
11+A+4K	Y-coord. of P(K)		
12+A+4K	Z-coord. of P(K)		
13+A+4K	V(0)	Floating Point	Starting parameter value
14+A+4K	V(1)	Floating Point	Ending parameter value
15+A+4K	X-coord. of NORM	Floating Point	Unit Normal (if curve is planar)
16+A+4K	Y-coord. of NORM		
17+A+4K	Z-coord. of NORM		
(Let D = 18+A+4K)			
D	E	Integer	Number of backpointers to associativity entities plus number of text pointers to general note entities

D+1	DE	Pointer	Pointers to associativities or general notes
D+E	DE	Pointer	
D+E+1	F	Integer	Number of properties
D+E+2	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
D+E+F+1	DE	Pointer	Pointers to properties

3.19 Rational B-Spline Surface Entity

A tutorial on the mathematical details of rational B-splines is given in a separate publication available from the National Bureau of Standards.

3.19.1 The rational B-spline surface may represent various analytical surfaces of general interest. This knowledge is important to both the generating and receiving system. The directory entry Form Number parameter is provided to communicate such information.

3.19.2 If the rational B-spline surface represents a preferred surface type, the form number corresponds to the most preferred type. The preference order is from 1 through 9 followed by 0. For example, if the surface is a right circular cylinder, the form number is set to 2. If the surface is a surface of revolution and also a torus, the form number is set to 5. If the surface is not one of the preferred types, the form number is set to 0.

3.19.3 Field 15 of the directory entry accommodates a form number. For this entity the options are as follows:

Form	Meaning
0	Form of the surface must be determined from the rational B-spline parameters
1	Plane
2	Right circular cylinder
3	Cone
4	Sphere
5	Torus
6	Surface of revolution
7	Tabulated cylinder
8	Ruled surface
9	General quadric surface

3.19.4 If, for each fixed value of the second parameter the resulting curves which are functions of the first parameter are closed, set PROP1 to 1, otherwise set PROP1 to 0. Similarly, if for each fixed value of the first parameter the resulting curves which are functions of the second parameter are closed, set PROP2 to 1, otherwise set PROP2 to 0.

3.19.5 If the surface is rational (does not have all weights equal) set PROP3 to 0. If all weights are equal to each other, the curve is non-rational and PROP3 is set to 1.

3.19.6 If the surface is periodic with respect to the first parameter, set PROP4 to 1, otherwise set PROP4 to 0. If the surface is periodic with respect to the second parameter, set PROP5 to 1, otherwise set PROP5 to 0.

3.19.7 Directory Data

ENTITY TYPE NUMBER: 128

3.19.8 Parameter Data

Parameter	Value	Format	Comment
1	K1	Integer	Upper index of first sum
2	K2	Integer	Upper index of second sum
3	M1	Integer	Degree of first set of basis functions
4	M2	Integer	Degree of second set of basis functions
5	PROP1	Integer	=0 Open in first parametric direction =1 Closed
6	PROP2	Integer	=0 Open in second parametric direction =1 Closed
7	PROP3	Integer	=0 Rational =1 Non-rational
8	PROP4	Integer	=0 Non-periodic in first parametric direction =1 Periodic in first parametric direction
9	PROP5	Integer	=0 Non-periodic in second parametric direction =1 Periodic in second parametric direction

Let $N1 = K1 - M1 + 1$, $N2 = K2 - M2 + 1$,

$A = N1 + 2M1$, $B = N2 + 2M2$, $C = (K1 + 1)(K2 + 1)$

10	S(-M1)	Floating Point	First knot sequence
.	.		
.	.		
.	.		
10+A	S(N1 + M1)		
11+A	T(-M2)	Floating Point	Second knot sequence
.	.		
.	.		
.	.		
11+A+B	T(N2 + M2)		
12+A+B	W(0,0)	Floating Point	Weights
13+A+B	W(1,0)		
.	.		
.	.		
.	.		
13+A+B+C	W(K1,K2)		
14+A+B+C	X-coord. of P(0,0)	Floating Point	Control Points
15+A+B+C	Y-coord. of P(0,0)		
16+A+B+C	Z-coord. of P(0,0)		
17+A+B+C	X-coord. of P(1,0)		
18+A+B+C	Y-coord. of P(1,0)		
19+A+B+C	Z-coord. of P(1,0)		
.	.		
.	.		
.	.		
11+A+B+4C	X-coord. of P(K1, K2)		
12+A+B+4C	Y-coord. of P(K1, K2)		
13+A+B+4C	Z-coord. of P(K1,K2)		
14+A+B+4C	U(0)	Floating Point	Starting parameter value for first basis
15+A+B+4C	U(1)	Floating Point	Ending parameter value for first basis
16+A+B+4C	V(0)	Floating Point	Starting parameter value for second basis
17+A+B+4C	V(1)	Floating Point	Ending parameter value for second basis

3.20 Node Entity

The node entity is a geometric point used in the definition of a finite element. Directory entry field 7 points to a labeled definition coordinate system transformation matrix. Coordinate angles for the cylindrical and spherical coordinate systems are specified in degrees.

3.20.1 Every node has a nodal displacement coordinate system associated with it. This is form 10, 11, or 12 of the transformation matrix entity which provides translational and rotational components for load, restraint and displacement results.

The origin of the nodal displacement coordinate system is always the location of the node. However, the orientation of the nodal displacement axes depends on the location of the node and the type of displacement coordinate system being referenced. Cartesian (Rectangular), cylindrical and spherical are the three possible types.

If the displacement coordinate system is cartesian, then the nodal displacement axes are parallel to the respective referenced coordinate system. This is illustrated in Figure 3-17(a) Cartesian.

For the cylindrical type, the orientation of the nodal displacement axes depends on the coordinate value of the node as defined in the referenced displacement coordinate system. The nodal displacement axes are respectively in the radial, tangetial and axial directions as illustrated in Figure 3-17(b) Cylindrical.

Finally, for spherical, the orientation of the nodal displacement axes depend on both the θ and ϕ coordinates of the node as defined in the referenced displacement coordinate system. The nodal displacement axes are respectively in the radial, meridional and aximuthal directions as indicated in Figure 3-17(c) Spherical.

If a node lies on the polar axes of either the cylindrical or spherical coordinate system, the nodal displacement axes are defined parallel to the referenced displacement coordinate system axes. For cylindrical the first

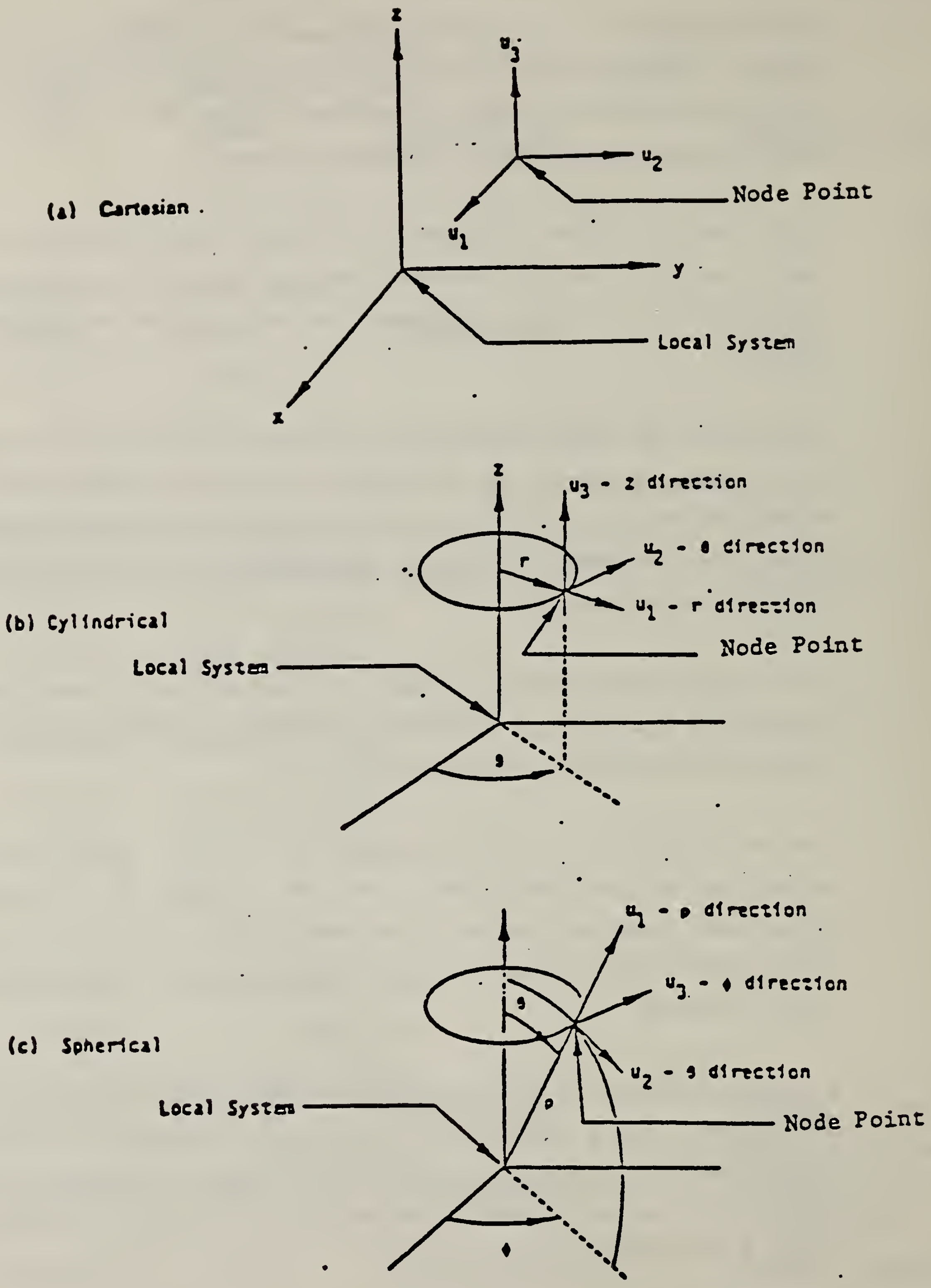


FIG. 3-17 DISPLACEMENT COMPONENTS

axis is the (0=0) axis and the third axis is the polar axis. For spherical the first axis is the (0=0) axis while the third axis is the polar axis. The second axis of both systems is defined by the appropriate cross product of the previously defined axes.

3.20.2 Directory Data

ENTITY TYPE NUMBER: 134

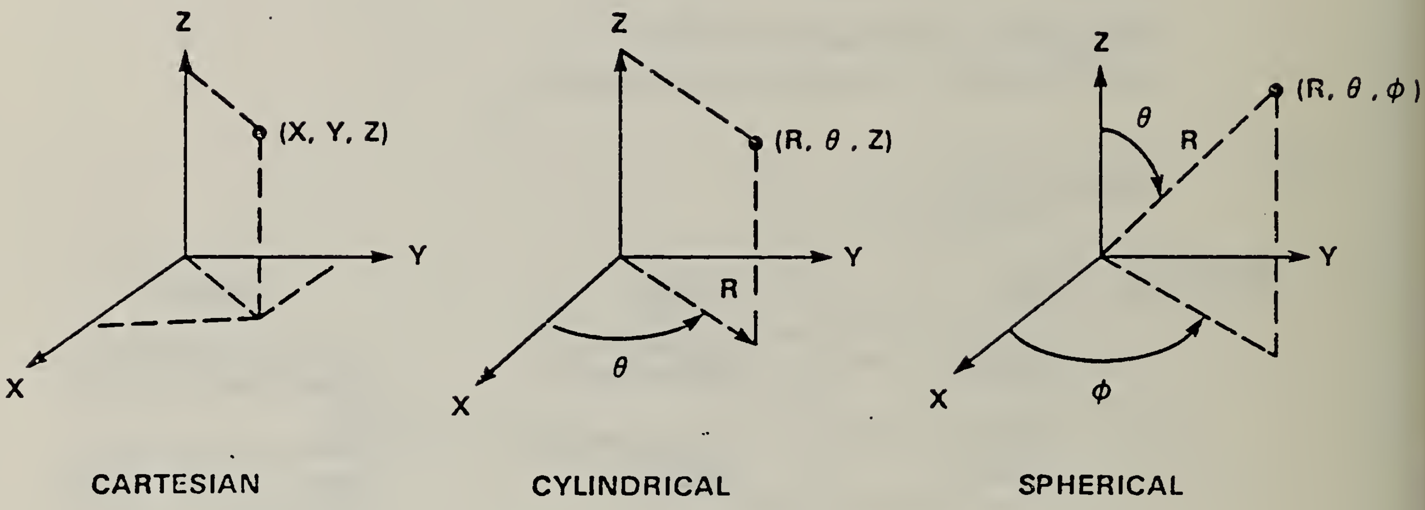
Entity Label: Node Label (Optional)

Entity Subscript: Node Number (Required)

3.20.3 Parameter Data

Parameter	Value	Format	Comment
1	X/R/R	Floating Point	First nodal coordinate
2	Y/ϕ/ϕ	Floating Point	Second nodal coordinate
3	Z/Z/ϕ	Floating Point	Third nodal coordinate
4	NDPCS	Pointer	Pointer to the Nodal Displacement Coordinate System entity. Default (zero) is Global Cartesian Coordinate System.
5	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
6	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
5+N	DE	Pointer	
6+N	M	Integer	Number of properties
7+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Pointer	

3.20.4 Figure 3-18 illustrates the definition of a node in the three coordinate systems.



CARTESIAN

CYLINDRICAL

SPHERICAL

FIG. 3-18 NODE DEFINITION IN EACH COORDINATE SYSTEM

3.21 Finite Element Entity

A finite element is defined by an element topology (i.e., node connectivity) along with physical and material properties.

3.21.1 Table 3-1 lists the data to define the element topology. Figure 3-19 illustrates the node connectivity for each element topology.

3.21.2 In Table 3-1 the IGES element name is an English abbreviation or acronym describing the element. The element topology type is an integer number which will appear as the first parameter of the parameter data. The order is an integer identifying the order of an edge where 0=not applicable, 1=linear, 2=parabolic and 3=cubic. The number of nodes from table N will appear as the second parameter of the finite element parameter data. A missing node in the connectivity sequence will have its corresponding pointer value equal to zero.

3.21.3 Directory Data

ENTITY TYPE NUMBER: 136

Entity Label: Element Label (Optional)

Entity Subscript: Element Number (Required)

3.21.4 Parameter Data

Parameter	Value	Format	Comment
1	ITOP	Integer	Topology Type. See attachment.
2	N1	Integer	Number of Nodes defining Element. See note.
3	DE	Pointer	Pointer to first node defining element. See note.
. . .	. . .	. . .	
N1+2	DE	Pointer	Pointer to last node defining element
N1+3	ETYP	String	Element type name

N1+4	N	Integer	Number of back pointers (to associativity entities)/test pointers (to general note entities)
N1+N+3	DE	Pointer	Pointers to associativities or general notes
	.	.	
	.	.	
	.	.	
N1+N+4	DE	Pointer	
N1+N+5	M	Integer	Number of properties
N1+N+6	DE	Pointer	Pointers to properties
	.	.	
	.	.	
	.	.	
N1+N+M+5	DE	Pointer	

TABLE 3-1 IGES/FINITE ELEMENT TOPOLOGY

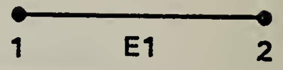
ELEMENT DATA CHART

Element Data

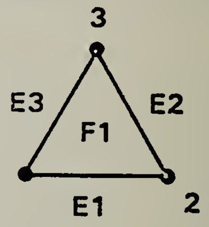
IGES Element Name	Element Topology Type	Order	Number of Nodes	Number of Edges	Number of Faces
BEAM	1	1	2	1	0
LTRIA	2	1	3	3	1
PTRIA	3	2	6	3	1
CTRIA	4	3	9	3	1
LQUAD	5	1	4	4	1
PQUAD	6	2	8	4	1
CQUAD	7	3	12	4	1
PTSW	8	2	12	9	5
CTSW	9	3	18	9	5
PTS	10	2	16	12	6
CTS	11	3	24	12	6
LSOT	12	1	4	6	4
PSOT	13	2	10	6	4
LSOW	14	1	6	9	5
PSOW	15	2	15	9	5
CSOW	16	3	24	9	5
LSO	17	1	8	12	6
PSO	18	2	20	12	6
CSO	19	3	32	12	6
ALLIN	20	1	2	1	0
APLIN	21	2	3	1	0
ACLIN	22	3	4	1	0
ALTRIA	23	1	3	3	0
APTRIA	24	2	6	3	0
ALQUAD	25	1	4	4	0
APQUAD	26	2	8	4	0
SPR	27	0	2	0	0
GSPR	28	0	1	0	0
DAMP	29	0	2	0	0
GDAMP	30	0	1	0	0
MASS	31	0	1	0	0
RBDY	32	0	2	0	0

FIG. 3-19 IGES FINITE ELEMENT TOPOLOGY SET

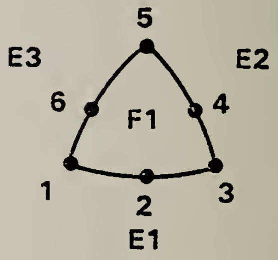
1. BEAM
   E1=1,2



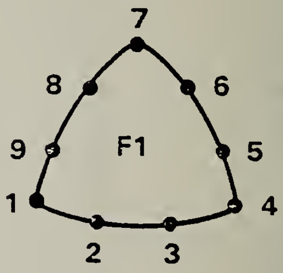
2. LTRIA - Linear Triangle
   E1=1,2 F1=1,2,3,
   E2=2,3
   E3=3,1



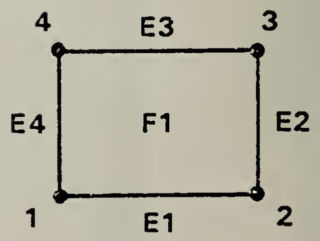
3. PTRIA - Parabolic Triangle
   E1=1,2,3 F1=1,2,3,4,5,6,
   E2=3,4,5
   E3=5,6,1



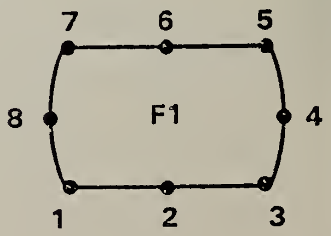
4. CTRIA - Cubic Triangle
   E1=1,2,3,4 F1=1,2,3,4,5,6,7,8,9,
   E2=4,5,6,7
   E3=7,8,9,1



5. LQUAD - Linear Quadrilateral
   E1=1,2 F1=1,2,3,4,
   E2=2,3
   E3=3,4
   E4=4,1



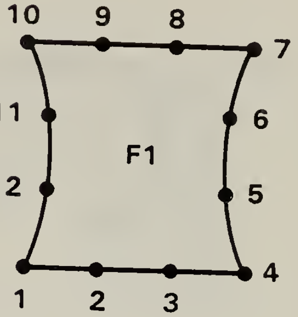
6. PQUAD - Parabolic Quadrilaterial
   E1=1,2,3 F1=1,2,3,4,5,6,7,8,
   E2=3,4,5
   E3=5,6,7
   E4=7,8,1



7. CQUAD - Cubic Quadrilateral

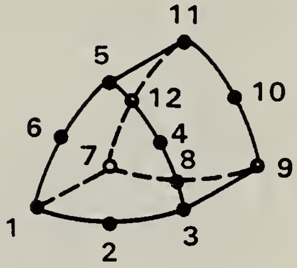
E1=1,2,3,4
E2=4,5,6,7
E3=7,8,9,10
E4=10,11,12,1

F1=1,2,3,4,5,6,7,8,9,10,11,12,



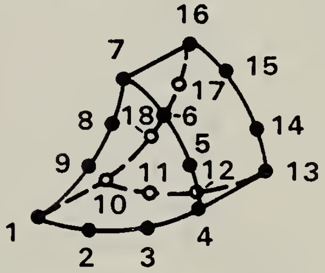
8. PTSW - Parabolic Thick Shell Wedge

E1=1,2,3
E2=3,4,5
E3=5,6,1
E4=7.8.9
E5=9,10,11
E6=11,12,7
E7=1,7
E8=3,9
E9=5,11
F1=1,2,3,4,5,6,
F2=7,8,9,10,11,12,
F3=1,2,3,9,8,7
F4=3,4,5,11,10,9
F5=5,6,1,5,6,1,7,12,11



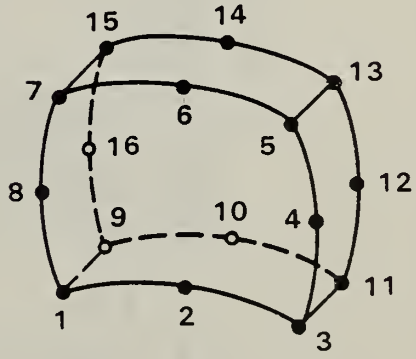
9. CTSW - Cubic Thick Shell Wedge

E1=1,2,3,4
E2=4,5,6,7
E3=7,8,9,1
E4=10,11,12,13
E5=13,14,15,16
E6=16,17,18,10
E7=1,10
E8=4,13
E9=7,16
F1=1,2,3,4,5,6,7,8,9,
F2=10,11,12,13,14,15,16,17,18
F3=1,2,3,4,13,12,11,10
F4=4,5,6,7,16,15,14,13
F5=7,8,9,1,10,18,17,16



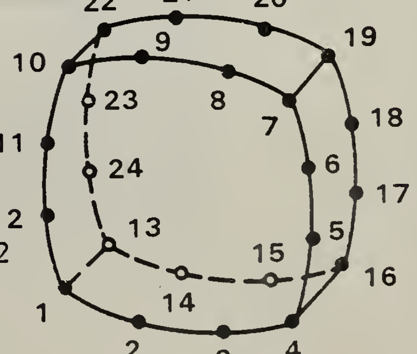
10. PTS - Parabolic Thick Shell

E1=1,2,3
E2=3,4,5
E3=5,6,7
E4-7,8,1
E5=9,10,11
E6=11,12,13
E7=13,14,15
E8=15,16,9
E9=1,9
E10=3,11
E11=5,13
E12=7,15
F1=1,2,3,4,5,6,7,8,
F2=9,10,11,12,13,14,15,16
F3=1,2,3,11,10,9
F4=3,4,5,13,12,11,
F5=5,6,7,5,15,14,13
F6=7,8,1,9,16,15



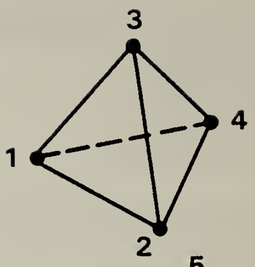
11. CTS - Cubic Thick Shell

E1=1,2,3,4
E2=4,5,6,7
E3=7,8,9,10
E4=10,11,12,1
E5=13,14,15,16
E6=16,17,18,19
E7=19,20,21,22
E8=22,23,24,13
E9=1,13
E10=4,16
E11=7,19
E12=10,22
F1=1,2,3,4,5,6,7,8,9,10,11,12,
F2=13,14,15,16,17,18,19,20,21,22,23,24
F3=1,2,3,4,16,15,14,13
F4=4,5,6,7,,19,18,17,16
F6=10,11,12,1,1,13,24,23,22
F5=7,8,9,10,22,21,20,19
F6=10,22,13,1,1,13,24,23,22



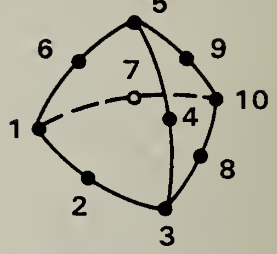
12. LSOT - Linear Solid Tetrahedron

E1=1,2
E2=2,3
E3=3,1
E4=1,4
E5=2,4
E6=3,4
F1=1,2,3,
F2=1,2,4,
F3=2,3,4,
F4=3,1,4



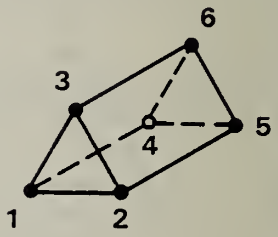
13. PSOT - Parabolic Solid Tetrahedron

E1=1,2,3,
E2=3,4,5
E3=5,6,1
E4=1,7,10
E5=3,8,10
E6=5,9,10
F1=1,2,3,4,5,6,
F2=1,2,3,8,10,7
F3=3,4,5,9,10,8
F4=5,6,1,7,10,9



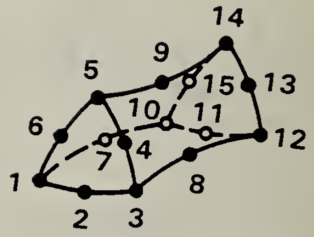
14. LSOW - Linear Solid Wedge

E1=1,2
E2=2,3
E3=3,1
E4=4,5
E5=5,6
E6=6,4
E7=1,4
E8=2,5
E9=3,6
F1=1,2,3,
F2=4,5,6
F3=1,2,5,4
F4=2,3,6,5
F5=3,1,4,6



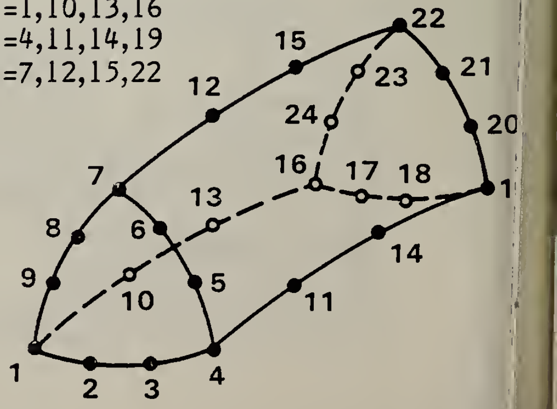
15. PSOW - Parabolic Solid Wedge

E1=1,2,3
E2=3,4,5
E3=5,6,1
E4=10,11,12
E5=12,13,14
E6=14,15,10
E7=1,7,10
E8=3,8,12
E9=5,9,14
F1=1,2,3,4,5,6,
F2=10,11,12,13,14,15
F3=1,2,3,8,12,11,10,7
F4=3,4,5,9,14,13,12,8
F5=5,6,1,7,10,15,14,9



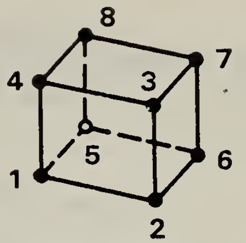
16. CSOW - Cubic Solid Wedge

E1=1,2,3,4
E2=4,5,6,7
E3=7,8,9,1
E4=16,17,18,19
E5=19,20,21,22
E6=22,23,24,16
E7=1,10,13,16
E8=4,11,14,19
E9=7,12,15,22
F1=1,2,3,4,5,6,7,8,9,
F2=16,17,18,19,19,20,21,22,23,24
F3=1,2,3,4,11,14,19,18,17,16,13,10
F4=4,5,6,7,12,15,22,21,20,19,14,11
F5=7,8,9,1,10,13,16,24,23,22,15,12



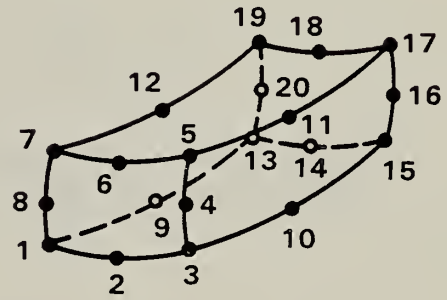
17. LSO - Linear Solid

E1=1,2 E5=5,6 E9=1,5
E2=2,3 E6=6,7 E10=2,6
E3=3,4 E7=7,8 E11=3,7
E4=4,1 FR=R,5 E12=4,8
F1=1,2,3,4,
F2=5,6,7,8
F3=1,2,6,5
F4=2,3,7,6
F5=3,4,4,8
F6=4,1,5,8



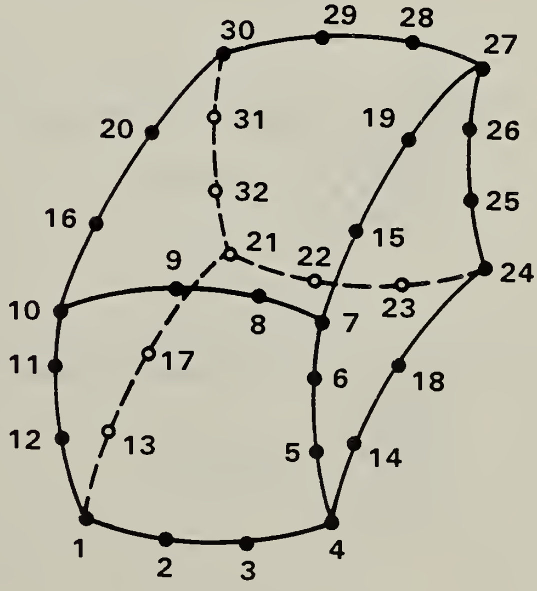
18. PSO - Parabolic Solid

E1=1,2,3 E7=17,18,19
E2=3,4,5 E8=19,20,13
E3=5,6,7 E9=1,9,13
E4=7,8,1 E10=3,10,15
E5=13,14,15 E11=5,11,17
E6=15,16,17 E12=7,12,19
F1=1,2,3,4,5,6,7,8
F2=13,14,15,16,17,18,19,20
F3=1,2,3,10,15,14,13,9 F5=5,6,7,12,19,18,17,11
F4=3,4,5,11,17,16,15,10 F6=7,8,1,9,13,20,19,12

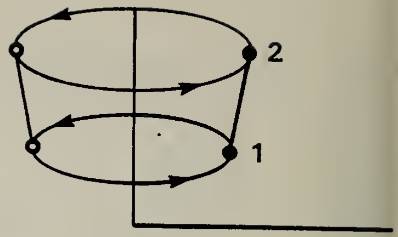


19. CSO - Cubic Solid

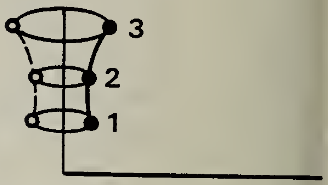
E1=1,2,3,4
E2=4,5,6,7
E3=7,8,9,10
E4=10,11,12,1
E5=21,22,23,24
E6=24,25,26,27
E7=27,28,29,30
E8=30,31,32,21
E9=1,13,17,21
E10=4,14,18,24
E11-7,15, 19, 27
E12=10,16,20,30
F1=1,2,3,4,5,6,7,8,9,10,11,2
F2=21,22,23,24,25,26,27,28,29,30,31,32
F3=1,2,3,4,14,18,24,23,22,21,17,13
F4=4,5,6,7,15,19,27,26,25,24,18,14
F5=7,8,9,10,16,20,30,29,28,27,19,15
F6=10,11,12,1,13,17,21,32,31,30,20,16



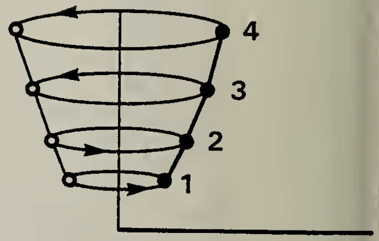
20. ALLIN - Axisymmetric Linear Line
E1=1,2 No Faces



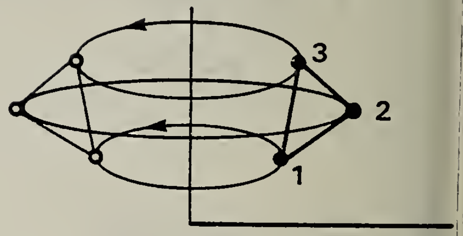
21. APLIN - Axisymmetric Parabolic Line
E1=1,2,3 No Faces



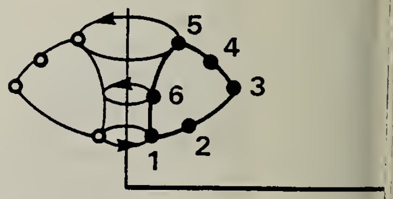
22. ACLIN - Axisymmetric Cubic Line
E1=1,2,3,4 No Faces



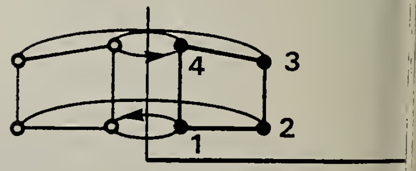
23. ALTRIA - Axisymmetric Linear Triangle
E1=1,2
E2=2,3 No Faces
E3=3,1



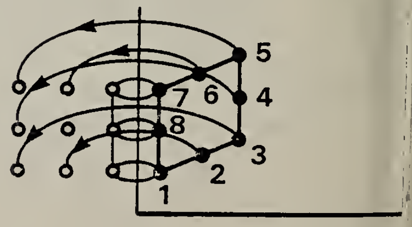
24. APTRIA - Axisymmetric Parabolic Triangle
E1=1,2,3
E2=3,4,5 No Faces
E3=5,6,1



25. ALQUAD - Axisymmetric Linear Quadrilateral
E1=1,2
E2=2,3
E3=3,4 No Faces
E4=4,1

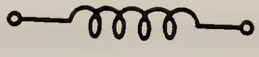


26. APQUAD - Axisymmetric Parabolic Quadrilateral
E1=1,2,3
E2=3,4,5
E3=5,6,7 No Faces
E4=7,8,1

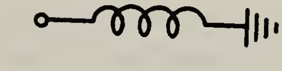


27. SPR - Spring

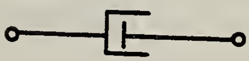
No edges or faces



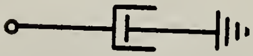
28. GSPR - Grounded Spring



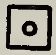
29. DAMP - Damper



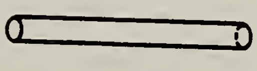
30. GDAMP - Grounded damper



31. MASS - Mass



32. RBDY - Rigid Body



4 NON-GEOMETRY

4.1 GENERAL

This section contains capabilities for representing non-geometry and includes:

- Annotation Entities

- Structure Entities

Entity numbers from 200 through 499 are reserved for this Section. In addition, some non-geometric entities make use of entity type number 106 (copious data).

4.2 Annotation Entities

4.2.1 Construction. Many annotation entities are constructed by using other entities. For example, the linear dimension entity will contain a pointer to two witness line entities (a form of copious data), two pointers to leader (arrow) entities, and one pointer to a general note entity.

4.2.2 Definition Space. An annotation entity may be defined in XT, YT, ZT definition space (see the discussion in Section 3.1) or in a two-dimensional space associated with a drawing entity (Entity type number 404).

In the case of XT, YT, ZT definition space, a transformation matrix may be applied to locate the annotation entity within model space.

Within the XT, YT, ZT definition space, subordinate entities to an annotation entity may have different ZT displacements. For example, within the linear dimension, a different ZT value may be found in each of: general note, leader, and witness lines (which are pointed to in the linear dimension parameter data). An example showing the use of ZT displacement (DEPTH) is shown in Figure 4-1.

While the option of having dimensions occupy different planes exists, it is expected that only a single plane will be used. The reason for its existence is due to the structure of annotation entities. As each dimension may be comprised of several subordinate entities, each subordinate entity by its definition has the ability to stand alone and may require its own ZT displacement. When used in conjunction with other entities as a subordinate to a primary entity, it is likely, though not necessary, that each ZT is identical.

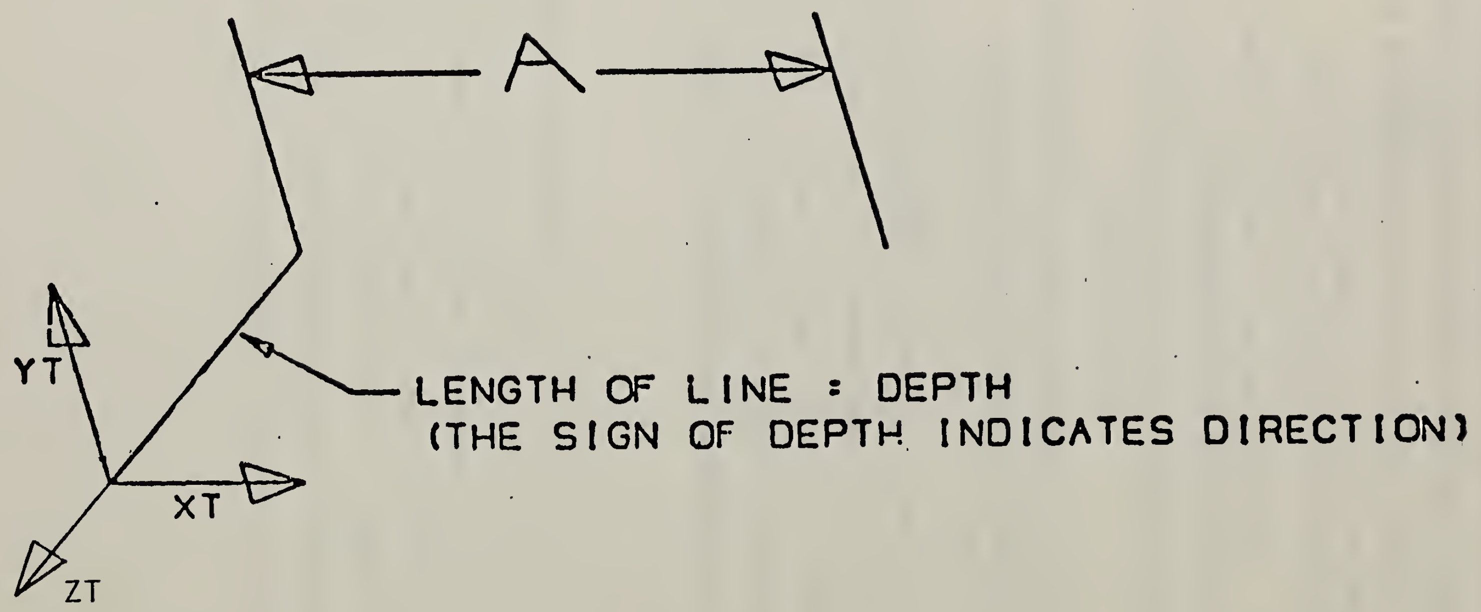


FIG. 4-1 CONSTRUCTION OF ZT DEPTH OF ANNOTATION ENTITIES

4.2.3 Entity Type/Type Number.

The following entities are defined in this section:

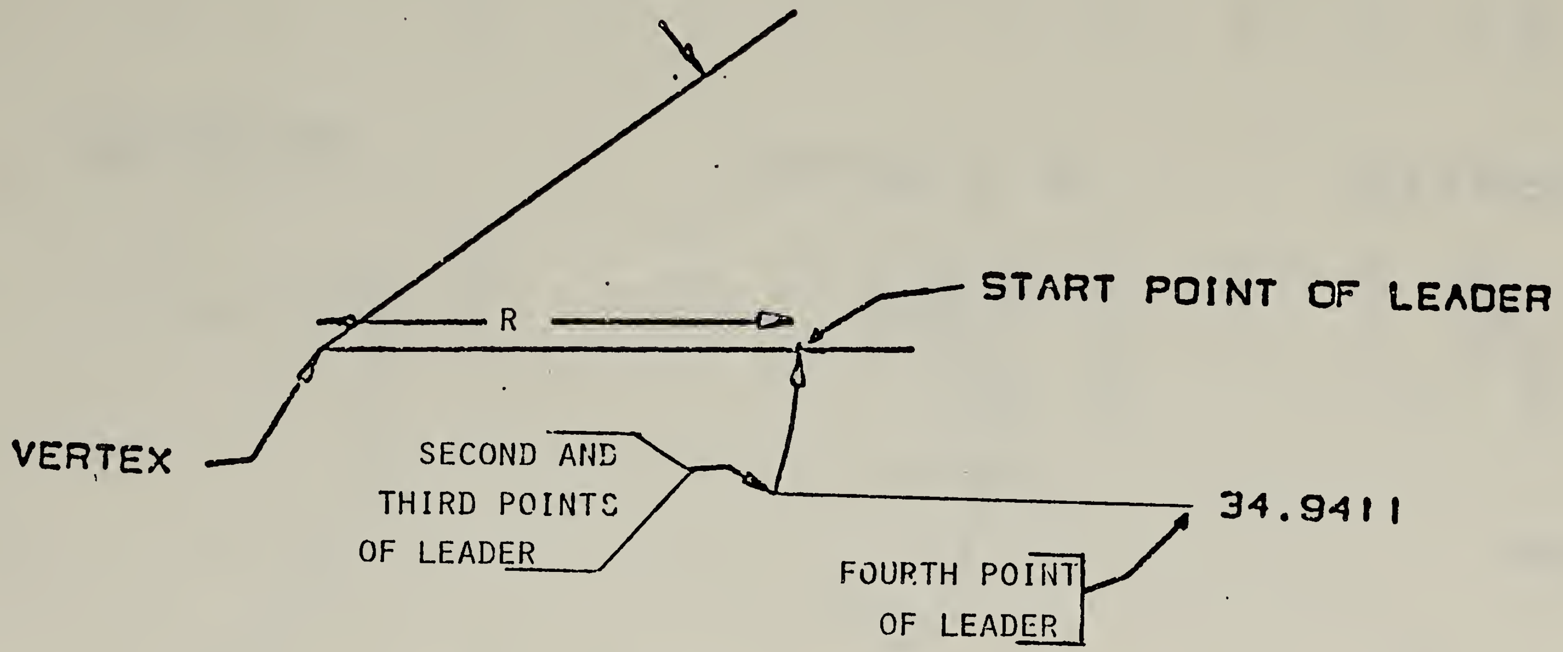
Entity Type	Entity Type Number
Angular Dimension Entity	202
Centerline Entity	106
Diameter Dimension Entity	206
Flag Note Entity	208
General Label Entity	210
General Note Entity	212
Leader (Arrow) Entity	214
Linear Dimension Entity	216
Ordinate Dimension Entity	218
Point Dimension Entity	220
Radius Dimension Entity	222
Section Entity	106
Witness Line Entity	106

4.2.4 Angular Dimension Entity. An angular dimension entity consists of a general note, zero, one or two witness lines, two leaders, and an angle vertex point. Refer to Figure 4-2 for examples of angular dimensions. If two witness lines are used, each is contained in its own copious data entity.

Each leader consists of at least one circular arc segment with an arrowhead at one end. The leader pointers are ordered such that the first circular arc segment of the first leader is defined in a counterclockwise manner from arrowhead to terminate point, and the first circular arc segment of the second leader is defined in a clockwise manner. (Refer to 3.1.15 for information relating to the use of the term counterclockwise).

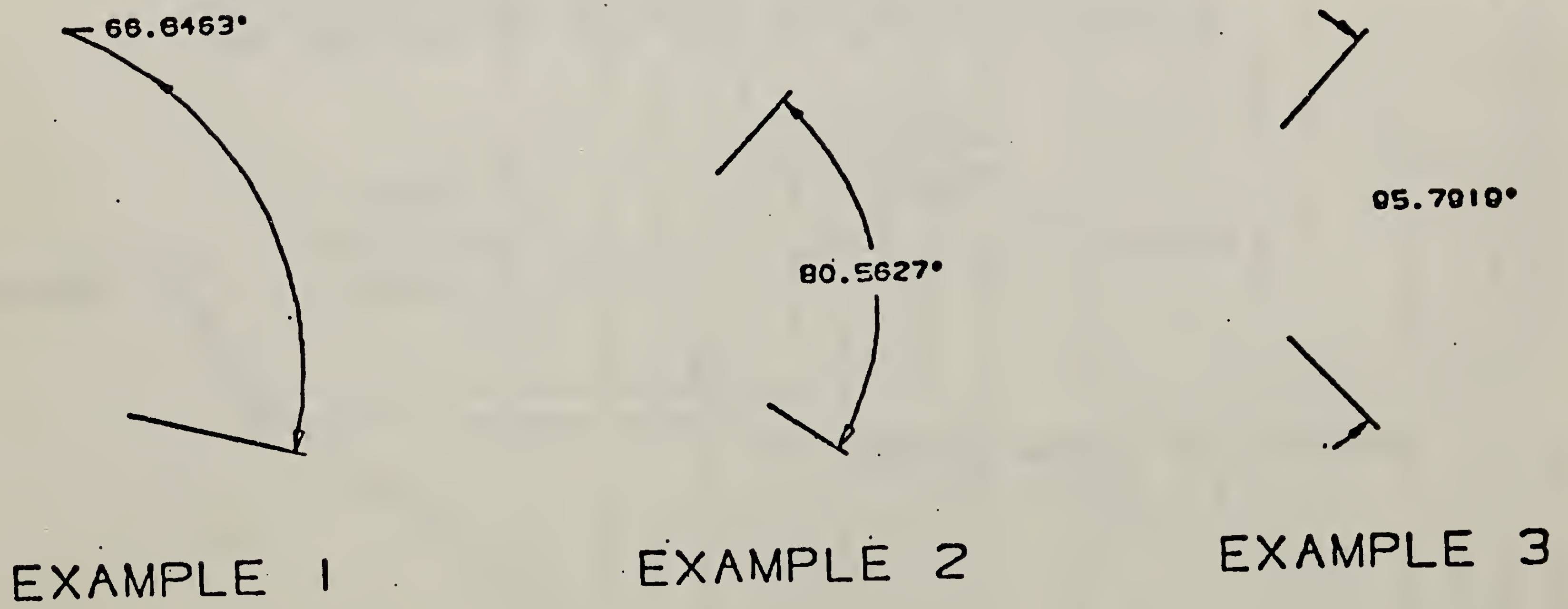
4.2.4.1 Section 4.2.10 contains a discussion of multi-segment leaders. For those leaders in angular dimension entities consisting of more than one segment, the first two segments are circular arcs with a center at the vertex point. The second circular arc segment is defined in the opposite direction from the first circular arc segment. Remaining segments, if any, are straight lines. Any leader segment in which the start point is the same as the terminate point is to be ignored. This convention arises to facilitate the definition of the second circular arc segment in the bottom leader in Fig. 4-2. Example 1 in Fig 4-3 illustrates a leader with three segments.

Refer to Figure 4-3 for examples of angular dimensions.



THE RADIUS OF THE ARC IN THE LEADER MUST BE CALCULATED BETWEEN THE VERTEX POINT AND THE START POINT OF THE LEADER

FIG. 4-2 ANGULAR DIMENSION: CONSTRUCTION OF ARCS IN THE ASSOCIATED LEADERS



EXAMPLE 1

EXAMPLE 2

EXAMPLE 3

FIG. 4-3 EXAMPLES OF THE ANGULAR DIMENSION ENTITY

4.2.4.2 Directory Data

ENTITY TYPE NUMBER : 202

4.2.4.3 Parameter Data

Parameter	Value	Format	Comment
1	GN	Pointer	Pointer to general note directory entry
2	W1	Pointer	Pointer to first witness line directory entry or 0
3	W2	Pointer	Pointer to second witness line directory entry or 0
4	XT	Floating Point	Coordinates of vertex point
5	YT	Floating Point	
6	R	Floating Point	Radius of leader arcs
7	A1	Pointer	Pointer to 1st leader directory entry or 0
8	A2	Pointer	Pointer to 2nd leader directory entry or 0
9	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
10	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
9+N	DE	Pointer	

10+N	M	Integer	Number of properties
11+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
10+N+M	DE	Pointer	

4.2.5 Centerline Entity

The centerline entity takes one of two forms. The first, as illustrated in Figure 4-4, Examples 1 and 2, appears as crosshairs and is normally used in conjunction with circles. The second type (Example 3) is a construction between 2 positions.

The Centerline entities are stored as a form of copious data. The associated matrix transforms the XT-YT plane of the centerline into model space. The coordinates of the centerline points describe the centerline display symbol. The display symbol is described by line segments where each line is from

(X_n, Y_n, Z_n), to $(X_{n+1}, Y_{n+1}, Z_{n+1})$ where $n = 1,3,5,\ldots,N-1$

4.2.5.1 Examples of the centerline entity are shown in Figure 4-4.

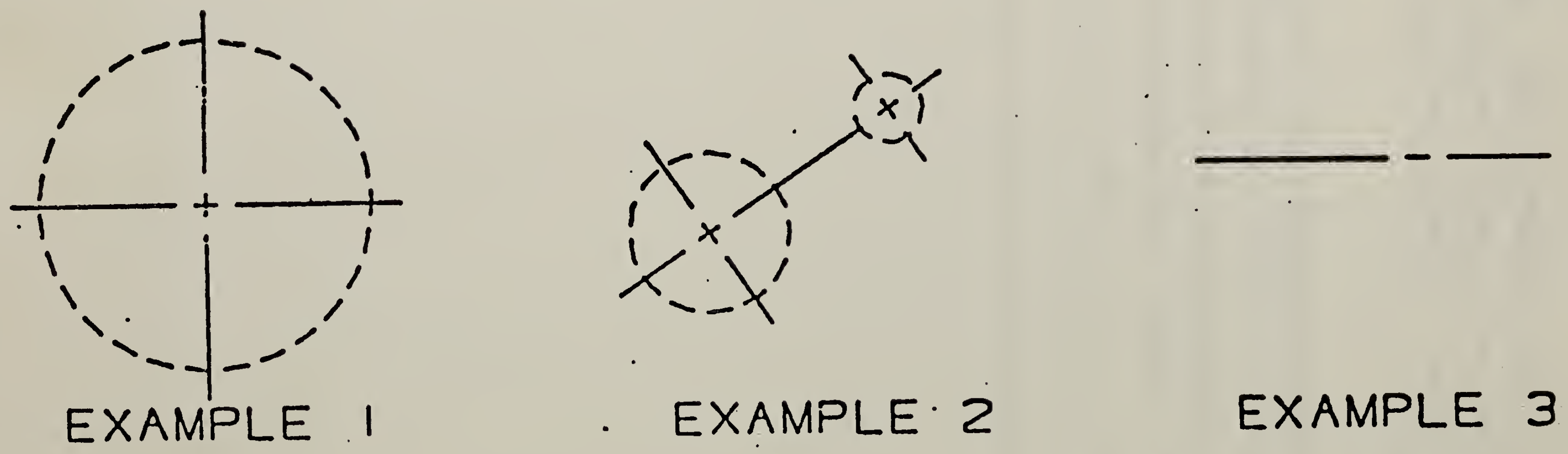


FIG. 4-4 EXAMPLES OF THE CENTERLINE ENTITY

4.2.6 Diameter Dimension Entity

A diameter dimension consists of a general note, one or two leaders, and an arc center point. If a second leader does not exist, its pointer value will be 0. Refer to Figure 4-5 for examples of the diameter dimension entity. The arc center coordinates are used for positioning the diameter dimension line relative to the arc being dimensioned.

4.2.6.1 Directory Data

ENTITY TYPE NUMBER : 206

4.2.6.2 Parameter Data

Parameter	Value	Format	Comment
1	NE	Pointer	Pointer to general note directory entry
2	A1	Pointer	Pointer to first leader directory entry
3	A2	Pointer	Pointer to second leader directory entry
4	XT	Floating Point	Arc center coordinates
5	YT	Floating Point	
6	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)

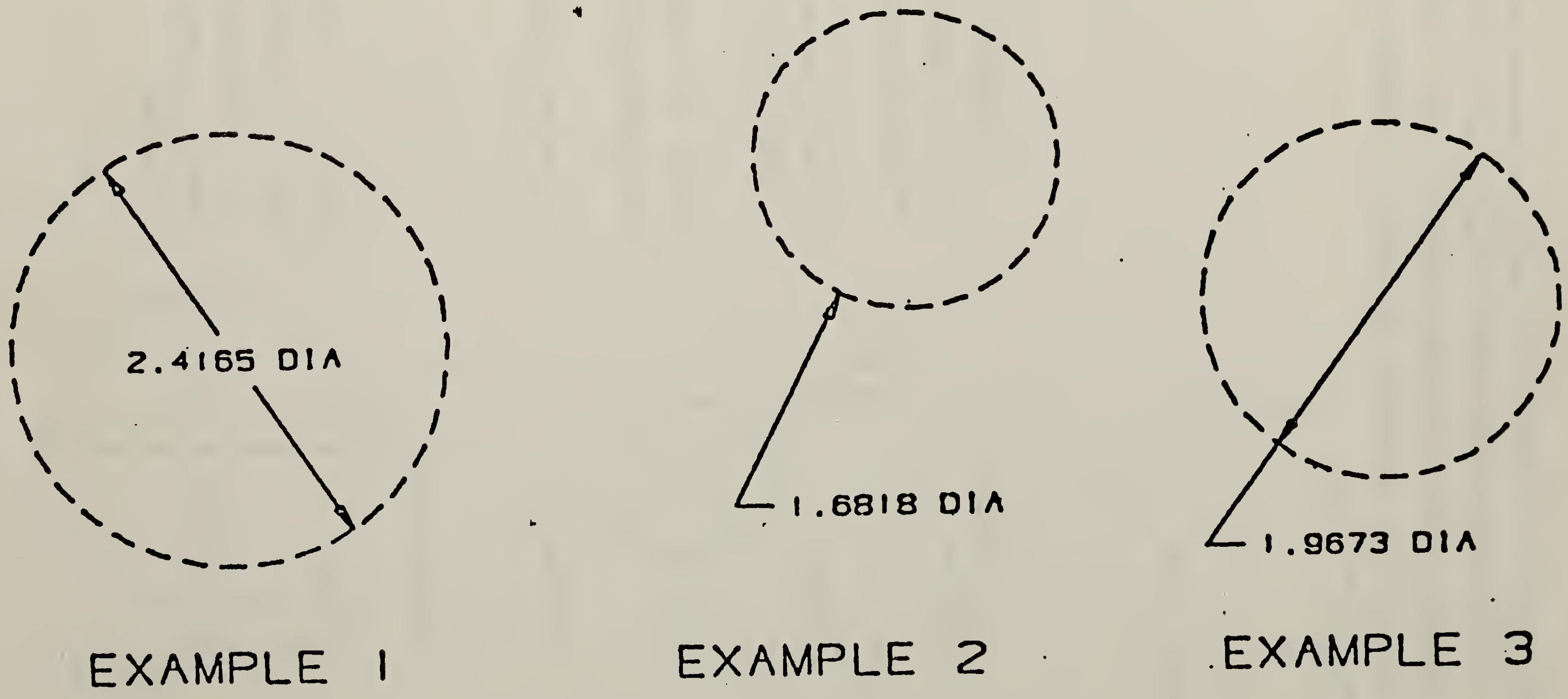


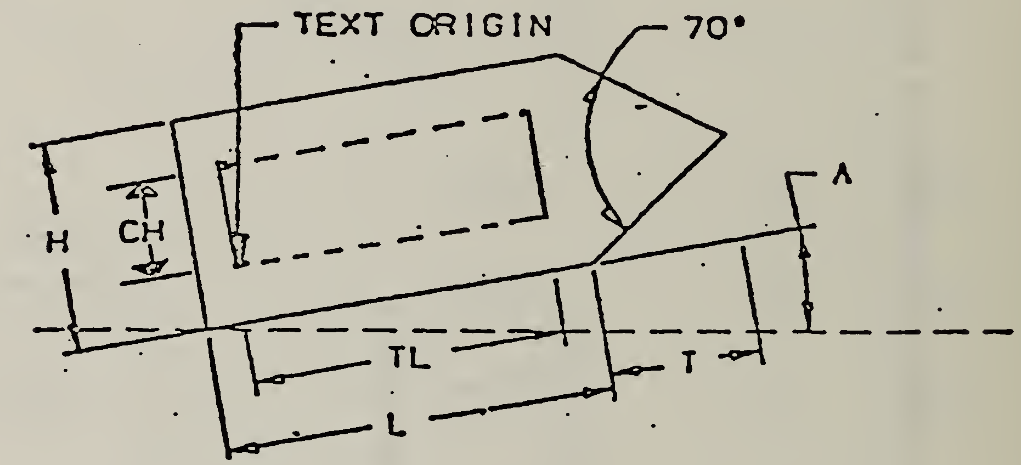
FIG. 4-5 EXAMPLES OF THE DIAMETER DIMENSION ENTITY

Parameter	Value	Format	Comment
7	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
6+N	DE	Pointer	
7+N	M	Integer	Number of properties
8+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
7+N+M	DE	Pointer	

4.2.7 Flag Note Entity

A flag note entity is label information formatted as shown in Figure 4-6. The rotation angle overrides the general note rotation angle and placement. Additional examples of the flag note entity are shown in Figure 4-7.

The flag note entity may be defined with or without associated leaders.



NOTE: Box outlined within flag illustrates bounds of text and should not be interpreted as sub-symbol.

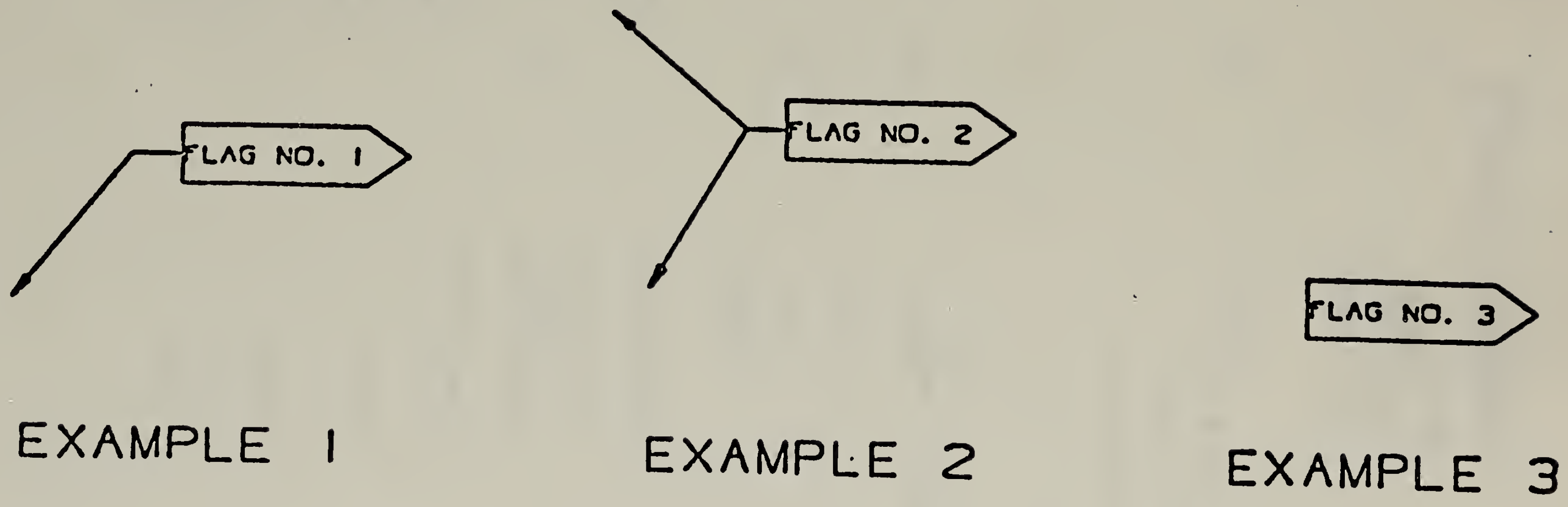
Fig. 4-6 FLAG NOTE

4.2.7.1 The flag note is constructed from information defined in the general note entity. This data is the character height and number of characters. For this reason, no geometric definition is explicit within the definition of the flag note entity.

The following specifications apply to Figure 4-6.

Variables:

H	=	Height	CH	=	Character height
L	=	Length	NC	=	Number of characters
TL	=	Text Length			(in general note)
T	=	Tip Length	A	=	Rotation angle in radians



EXAMPLE 1

EXAMPLE 2

EXAMPLE 3

FIG. 4-7 EXAMPLES OF THE FLAG NOTE ENTITY.

Formulas:

$TL = (.8)(CH)(NC)+(.4)(CH)(NC-1)$

$H = (2)(CH)$

$L = (TL)+(.4)(CH)$

$T = (.5)(H)/TAN35^{\circ}$

Restrictions:

H shall never be less than .3 in.

L shall never be less than .6 in.

T shall never be less than .214 in.

4.2.7.2 Directory Data

ENTITY TYPE NUMBER : 208

4.2.7.3 Parameter Data

Parameter	Value	Format	Comment
1	XT	Floating Point	Lower left corner coordinate
2	YT	Floating Point	
3	ZT	Floating Point	
4	A	Floating Point	Rotation angle in radians
5	DENOTE	Pointer	Pointer to general note directory entry
6	N	Integer	Number of arrows (leaders)
7	DE1	Pointer	Pointers to associated leaders
.			
.			
.			
6+N			

Parameter	Value	Format	Comment
7+N	NA	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
8+N	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
7+N+NA	DE	Pointer	
8+N+NA	M	Integer	Number of properties
9+N+NA	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
8+N+NA+M	DE	Pointer	

4.2.8 General Label Entity. A general label entity consists of a general note with one or more associated leaders.

Examples of general label entities are shown in Figure 4-8

4.2.8.1 Directory Data

ENTITY TYPE NUMBER : 210

4.2.8.2 Parameter Data

Parameter	Value	Format	Comment
1	DENOTE	Pointer	Pointer to associated general note
2	N	Integer	Number of leaders
3	DE	Pointer	Pointers to associated leaders
.	.	.	
.	.	.	
.	.	.	
N+2			
N+3	NA	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
N+4	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
N+3+NA	DE	Pointer	

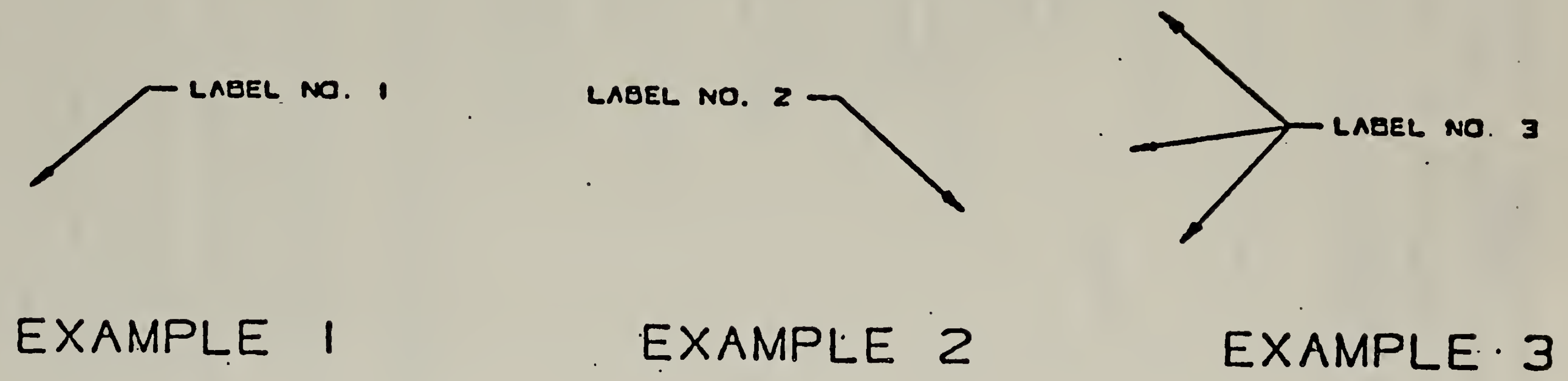


FIG. 4-8 EXAMPLES OF THE GENERAL LABEL ENTITY

Parameter	Value	Format	Comment
N+4+NA	M	Integer	Number of properties
N+5+NA	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
N+4+NA+M	DE	Pointer	

4.2.9 General Note Entity. A general note entity consists of 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:

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

0. Symbol Fonts
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.

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.

The parameters for the text block are applied in the following order:
(See Figure 4-14)

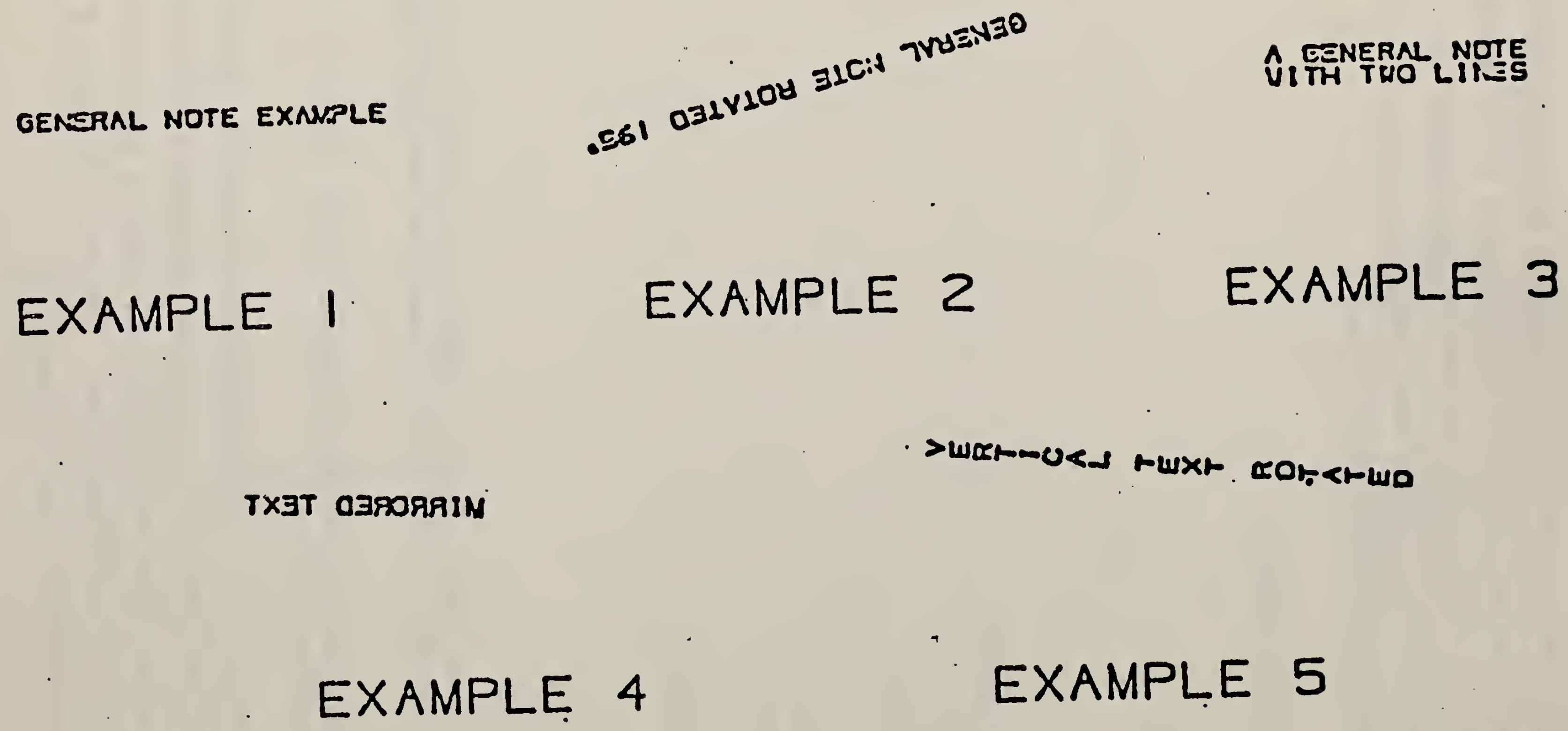


FIG. 4-9 EXAMPLES OF THE GENERAL NOTE ENTITY

Column 1 Displayable Font
Column 2 ASCII Character

Disp. Font	ASCII Char.
SP	
!	!
"	"
#	#
$	$
%	%
&	&
'	'
(	(
)	)
*	+
+	'
,	,
-	-
.	.
/	/
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
:	:
;	;
<	<
=	=
>	>
?	?
@	@
A	A
B	B
C	C
D	D
E	E
F	F
G	G
H	H
I	I
J	J
K	K
L	L
M	M
N	N
O	O
P	P
Q	Q
R	R
S	S
T	T
U	U
V	V
W	W
X	X
Y	Y
Z	Z
[	[
\	\
]	]
^	^
-	-
'	'
∠	a
⎅	b
▱	c
⌓	d
○	e
//	f
b/	g
↗	h
÷	i
⌖	j
⌒	k
⊥	l
Ⓜ	m
ø	n
o	o
Ⓟ	p
℄	q
◎	r
Ⓢ	s
⎕	t
⦶	u
△	v
◇	u
⍋	x
⧗	y
Y	z
{	{
|	|
}	}
~	~

FIG. 4-10 FONT 1001

Column 1 Displayable Font
Column 2 ASCII Character

Disp. Font	ASCII Char.	Disp. Font	ASCII Char.	Disp. Font	ASCII Char.	Disp. Font	ASCII Char.	Disp. Font	ASCII Char.
SP		3	3	F	F	Y	Y	∨	l
!	!	4	4	G	G	Z	Z	∧	m
"	"	5	5	H	H	[	[	≈	n
±	#	6	6	I	I	\	\	Σ	o
○	$	7	7	J	J	]	]	↑	p
S	%	8	8	K	K	∧	∧	↓	q
&	&	9	9	L	L	_	_	→	r
'	'	:	:	M	M	'	'	←	s
(	(	;	;	N	N	⧗	a	φ	t
)	)	<	<	O	O	÷	b	θ	u
*	*	=	=	P	P	≤	c	T	v
÷	+	>	>	Q	Q	≥	d	ψ	u
,	,	?	?	R	R	Δ	e	ω	x
-	-	@	@	S	S	√	f	λ	y
.	.	A	A	T	T	X	g	α	z
/	/	B	B	U	U	I	h	δ	{
0	0	C	C	V	V	≠	i	μ	|
1	1	D	D	W	W	!	j	π	}
2	2	E	E	X	X	⊃	k	~	~

FIG. 4-11 FONT 1002

0	⧖	27	ω	56	.	105	E	134	\	163	Ⓢ
1	÷	30	λ	57	/	106	F	135	]	164	⊓
2	≤	31	∝	60	0	107	G	136	^	165	⏀
3	≥	32	δ	61	1	110	H	137	_	166	△
4	△	33	μ	62	2	111	I	140	`	167	◇
5	√‾	34	π	63	3	112	J	141	∠	170	△
6	X	35	‾	64	4	113	K	142	⌖	171	⧖
7	≡	36	±	65	5	114	L	143	▱	172	Y
10	≠	37	°	66	6	115	M	144	⌓	173	{
11	∫	40	SP	67	7	116	N	145	○	174	|
12	⊃	41	!	70	8	117	O	146	//	175	}
13	∨	42	"	71	9	120	P	147	⌭	176	~
14	∧	43	#	72	:	121	Q	150	↗	177	≠
15	≈	44	$	73	;	122	R	151	⌯		
16	Σ	45	%	74	<	123	S	152	⌖		
17	↑	46	&	75	=	124	T	153	⌒		
20	↓	47	'	76	>	125	U	154	⊥		
21	→	50	(	77	?	126	V	155	Ⓜ		
22	←	51	)	100	@	127	W	156	∅		
23	φ	52	*	101	A	130	X	157	○		
24	θ	53	+	102	B	131	Y	160	Ⓟ		
25	τ	54	,	103	C	132	Z	161	℄		
26	ψ	55	−	104	D	133	[	162	◎		

FONT CODE = 0

ASCII CODE TO
CHARACTER MA

FIG. 4-12 CHARACTER SET & OCTAL CODE FOR FONT CODE ZERO

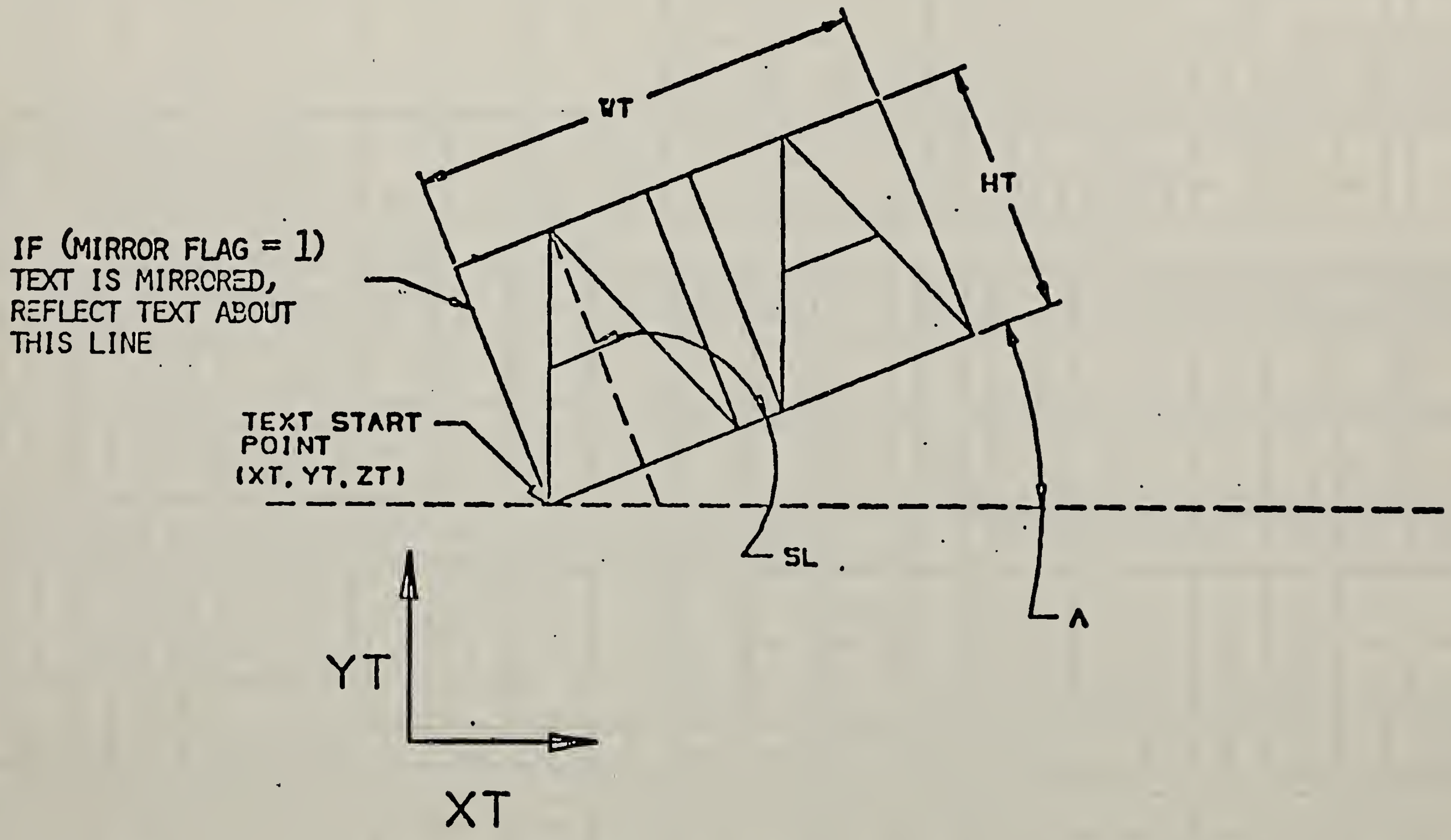


FIG. 4-13 GENERAL NOTE TEXT CONSTRUCTION

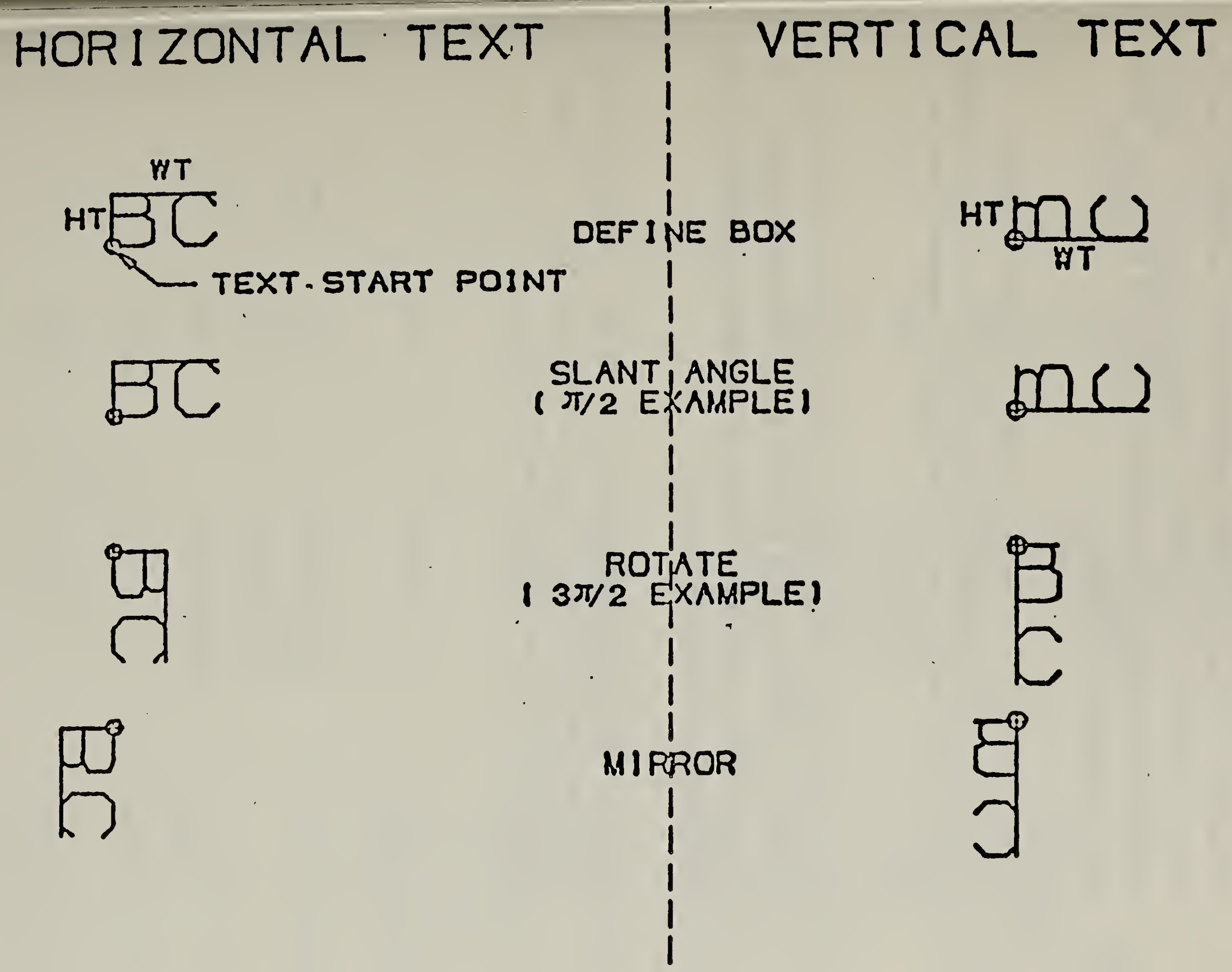


FIG. 4-14 GENERAL NOTE EXAMPLE OF TEXT OPERATIONS

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 ZT (where ZT is the value given for the text start point).

4) The mirror operation is performed last. The value 1 indicates the mirror axis is the (rotated) YT axis through the text start point. The value 2 indicates the mirror axis is the rotated XT axis.

4.2.9.1 Directory Data

ENTITY TYPE NUMBER : 212

4.2.9.2 Parameter Data

Parameter	Value	Format	Comment
1	NS	Integer	Number of text strings in general note
2	NC1	Integer	Number of characters in first string (TEXT1)
3	WT1	Floating Point	Box width
4	HT1	Floating Point	Box height
5	FC1	Integer	Font characteristic
6	SL1	Floating Point	Slant angle of TEXT1 in radians ($\pi/2$ is the value for no slant angle and is the default value)
7	A1	Floating Point	Rotation angle in radians for TEXT1
8	M1	Integer	Mirror flag (0-no mirror, 1-YT mirror axis, 2-XT mirror axis)
9	VH1	Integer	Rotate internal text flag (0-text horizontal, 1-text vertical)
10	XT1	Floating Point	First text start point
11	YT1	Floating Point	
12	ZT1	Floating Point	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	Character	Last text string
2+NS*12	N	Integer	Number of back pointers (to associativity entities) /text pointers (to general note entities)
3+NS*12	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
2+NS*12+N	DE	Pointer	
3+NS*12+N	M	Integer	Number of properties
4+NS*12+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
3+NS*12+N+M	DE	Pointer	

4.2.10 Leader (Arrow) Entity. A leader consists of one or more line segments except when the leader is part of an angular dimension (see 4.2.4). The first segment begins with an arrowhead. There will be ten different arrowheads and the selection is made by assigning values to FORM (see 4.2.10.1). Remaining segments successively link to a presumed text item. The leader entity includes parameters to define the size and shape of the arrowhead and the end points of each segment of the leader. An individual segment is assumed to extend from the end point of its predecessor in the segment list to its defined end point. Examples of leaders are shown in Figure 4-15.

In the use of angular, diameter, and linear dimension, there are instances where the text is exterior to the line or arc lying between the two arrows. In these situations, it remains the case that the appearance of two arrows implies the use of two leaders. These are formed by dividing the line or arc lying between the two arrows into two non-overlapping segments. Refer to Figure 4-16.

Some leaders (for example, the leader involved with the radius dimension in Figure 4-16) give the appearance of locating an arrow interior to a segment. This is not the case. However, there are two overlapping segments. The first segment begins at the arrow and, ·in the radius dimension example, ends at the center of the arc or circle being dimensioned. The second segment then overlaps the first, retraces the first in the opposite direction, and extends it. Leaders of this type for other types of dimensions are constructed similarly. For cases involving angular dimension, the first two segments are arcs.

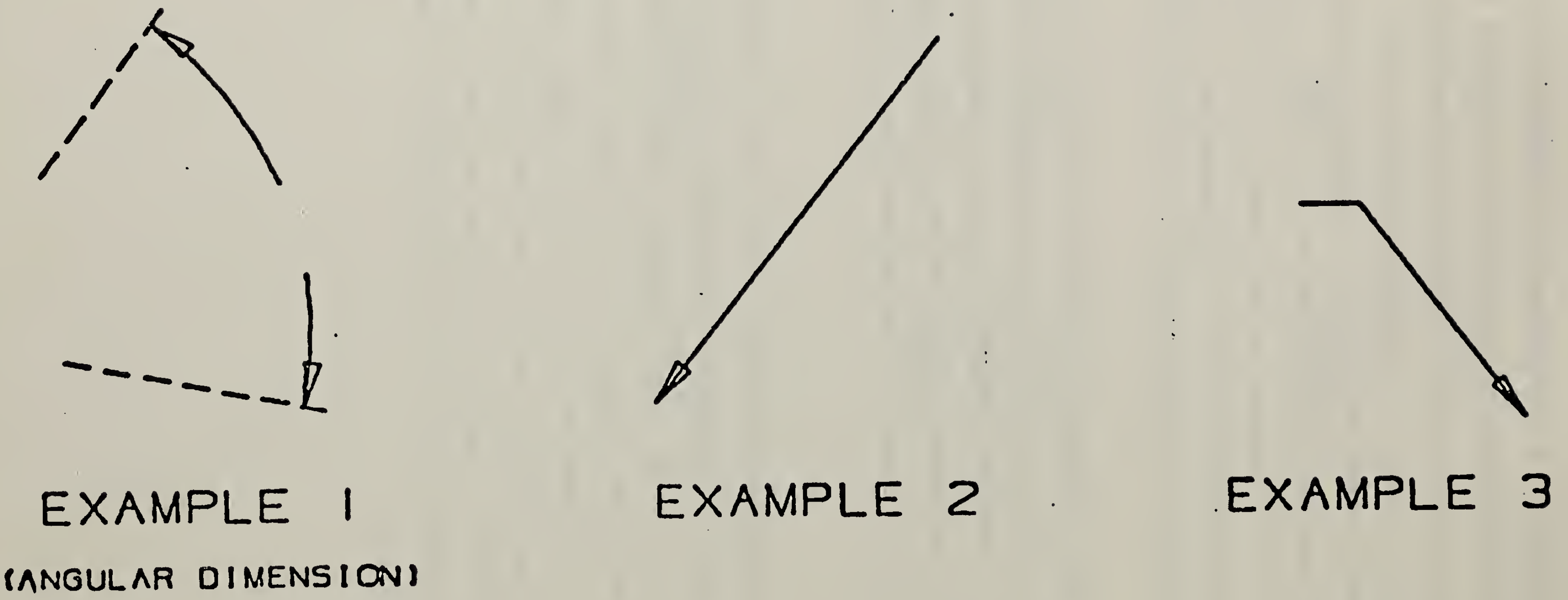
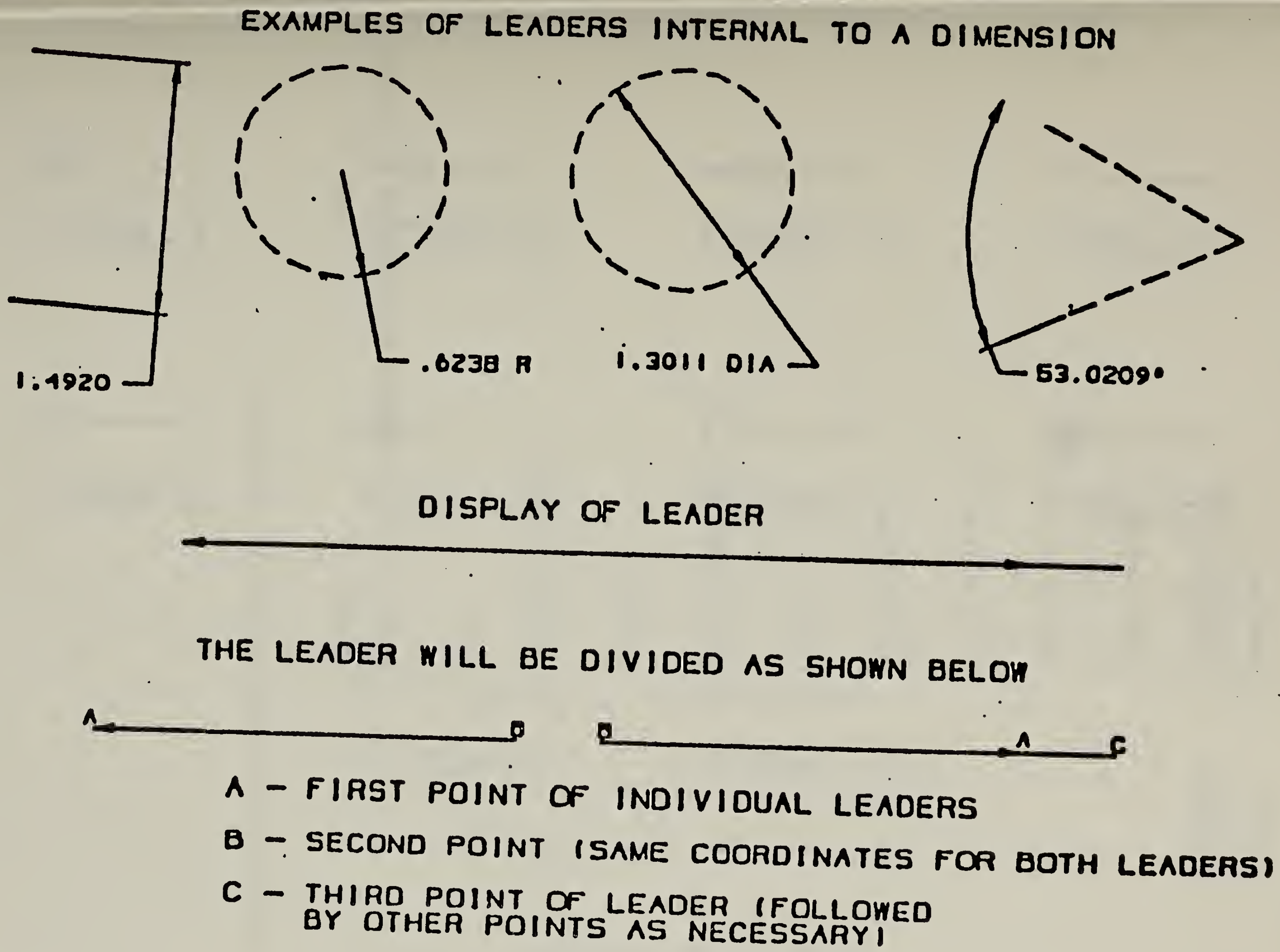


FIG. 4-15 EXAMPLES OF THE LEADER ENTITY


EXAMPLES OF LEADERS INTERNAL TO A DIMENSION
1.4920
.6238 R
1.3011 DIA
53.0209°
DISPLAY OF LEADER
THE LEADER WILL BE DIVIDED AS SHOWN BELOW
A
B
B
A
C

A – FIRST POINT OF INDIVIDUAL LEADERS

B – SECOND POINT (SAME COORDINATES FOR BOTH LEADERS)

C – THIRD POINT OF LEADER (FOLLOWED BY OTHER POINTS AS NECESSARY)

FIG. 4-16 · STRUCTURE OF LEADERS INTERNAL TO A DIMENSION

4.2.10.1 FORM Definition

FORM NUMBER	Meaning Arrowhead Type
1	Open Triangle
2	Triangle
3	Triangle (filled in)
4	No arrowhead
5	Circle
6	Circle (filled in)
7	Rectangle
8	Rectangle (filled in)
9	Slash
10	Integral sign centered at the end point of the first segment

4.2.10.2 Examples of each FORM are shown in Figure 4-17.

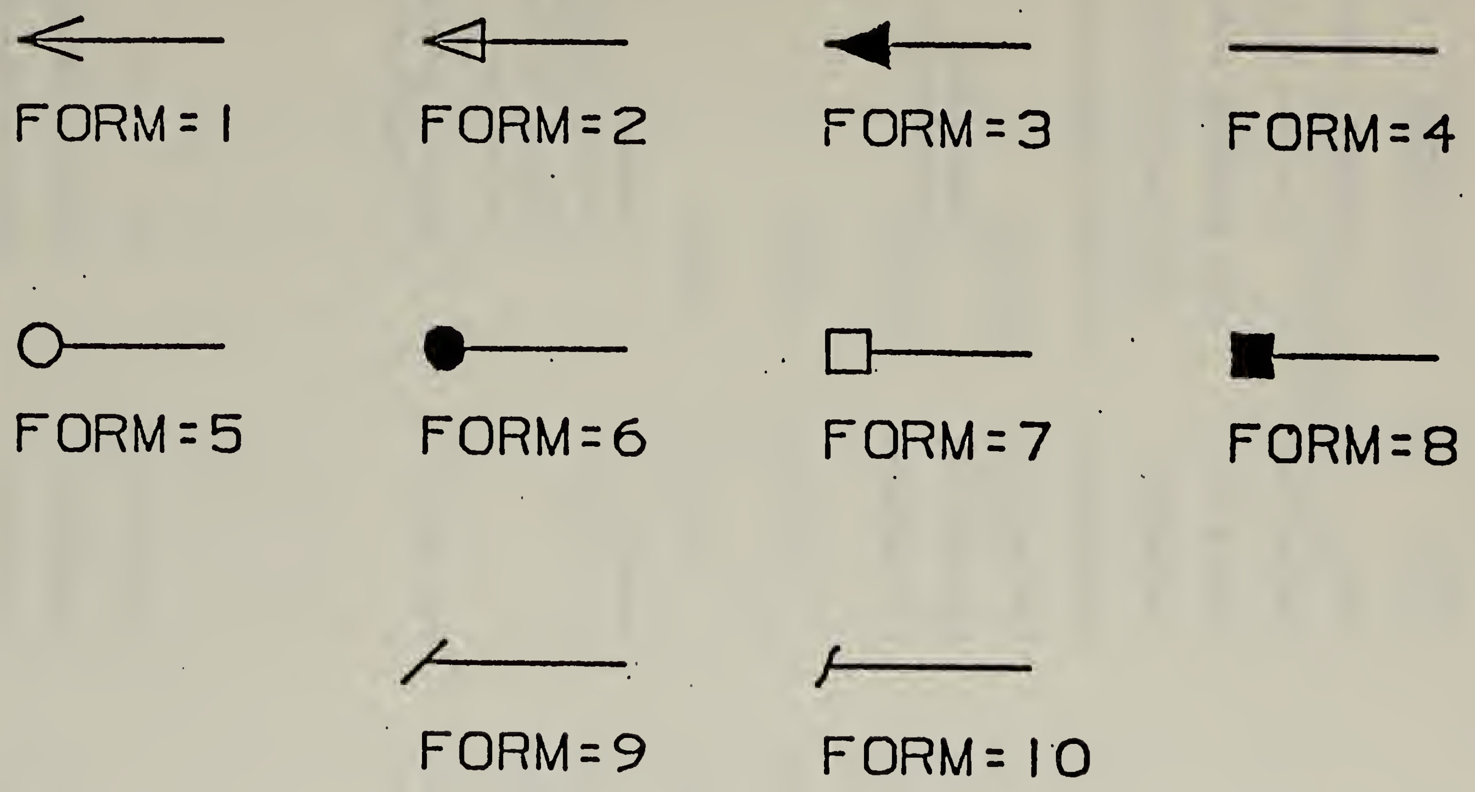


FIG. 4-17 ARROWHEAD DEFINITIONS

4.2.10.3 Directory Data

ENTITY TYPE NUMBER: 214

4.2.10.4 Parameter Data

Parameter	Value	Format	Comment
1	N	Integer	Number of segments
2	AD1	Floating Point	Arrowhead height
3	AD2	Floating Point	Arrowhead width
4	ZT	Floating Point	Z depth
5	XH	Floating Point	Arrowhead coordinates
6	YH	Floating Point	
7	X	Floating Point	Segment tail coordinate pairs
.	Y	Floating Point	
.			
.			
6+2N			
7+2N	NA	Integer	Number of back pointers (to associativity entries)/text pointers (to general note entities)
8+2N	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
7+2N+NA	DE	Pointer	
8+2N+NA	M	Integer	Number of properties
9+2N+NA	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
8+2N+NA+M	DE	Pointer	

4.2.11 Linear Dimension Entity. A linear dimension consists of a general note, two leaders, and zero to two witness lines. Refer to Figure 4-18 for examples of linear dimensions.

4.2.11.1 Directory Data

ENTITY TYPE NUMBER : 216

4.2.11.2 Parameter Data

Parameter	Value	Format	Comment
1	DENOTE	Pointer	Pointer to general note directory entry
2	DEARRW1	Pointer	Pointer to first leader directory entry
3	DEARRW2	Pointer	Pointer to second leader directory entry
4	DEWIT1	Pointer	Pointer to witness line directory entry, 0 if not defined
5	DEWIT2	Pointer	Pointer to witness line directory entry or 0
6	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
7	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
6+N	DE	Pointer	
7+N	M	Integer	Number of properties
8+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
7+N+M	DE	Pointer	

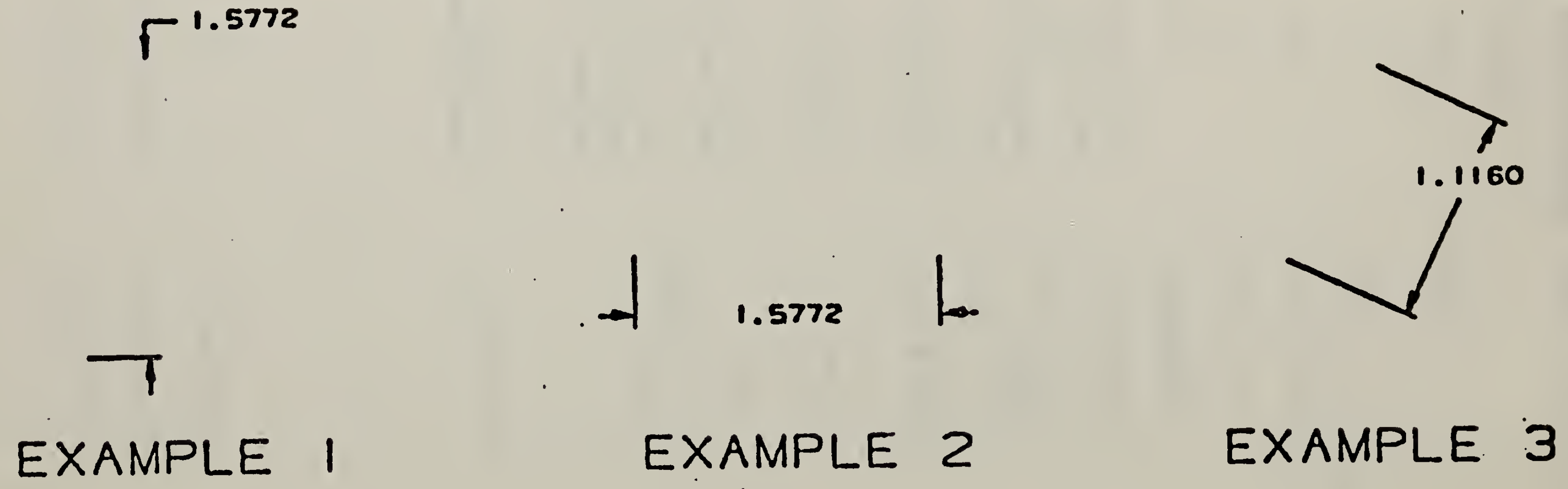


FIG. 4-18 EXAMPLES OF THE LINEAR DIMENSION ENTITY

4.2.12 Ordinate Dimension Entity. The ordinate dimension entity is used to indicate dimensions from a common base line. Dimensioning is only permitted along the XT or YT axis.

4.2.12.1 An ordinate dimension consists of a general note and a witness line or leader. The values stored are pointers to the directory entry for the associated note and witness line. Examples of ordinate dimensions are shown in Figure 4-19.

4.2.12.2 Directory Data

ENTITY TYPE NUMBER : 218

4.2.12.3 Parameter Data

Parameter	Value	Format	Comment
1	DENOTE	Pointer	Pointer to general note directory entry
2	DEWIT	Pointer	Pointer to witness line directory entry or leader directory entry
3	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
4	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
3+N	DE	Pointer	
4+N	M	Integer	Number of properties
5+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
4+N+M	DE	Pointer	

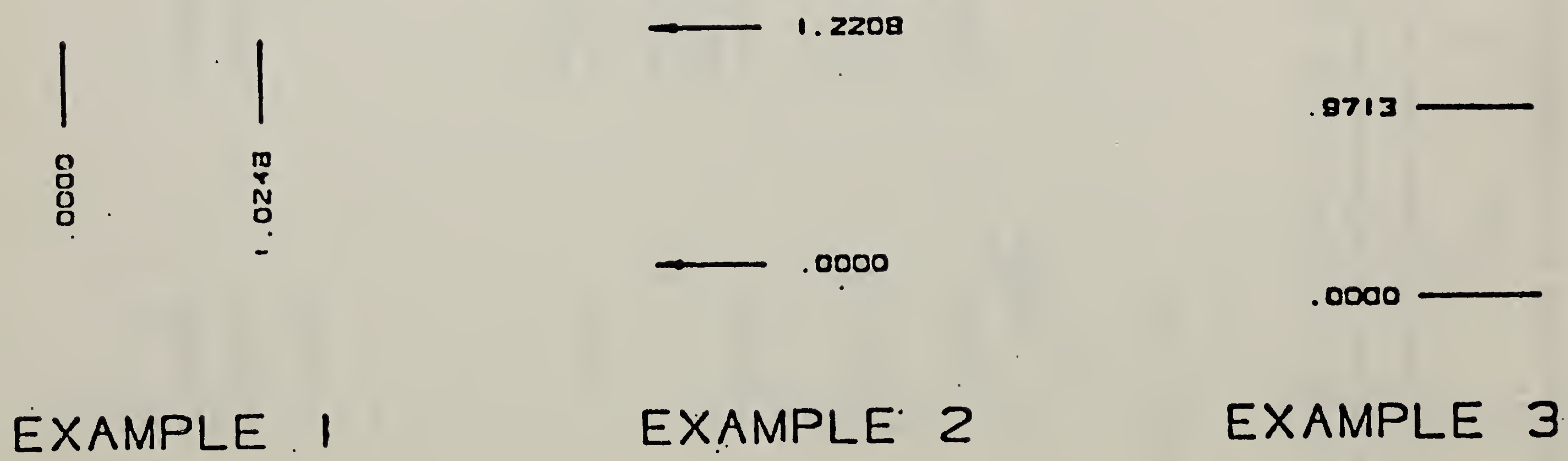


FIG. 4-19 EXAMPLES OF THE ORDINATE DIMENSION ENTITY

4.2.13 Point Dimension Entity. A point dimension consists of a leader, text, and an optional circle or hexagon enclosing the text.

4.2.13.1 The leader will always contain three segments, and its first and last segments are always horizontal or vertical. If a hexagon encloses the text, it will be described by a composite entity. If a circle or hexagon does not enclose the text, the last segment of the leader will be horizontal and it will underline the text.

4.2.13.2 Examples are shown in Figure 4-20.

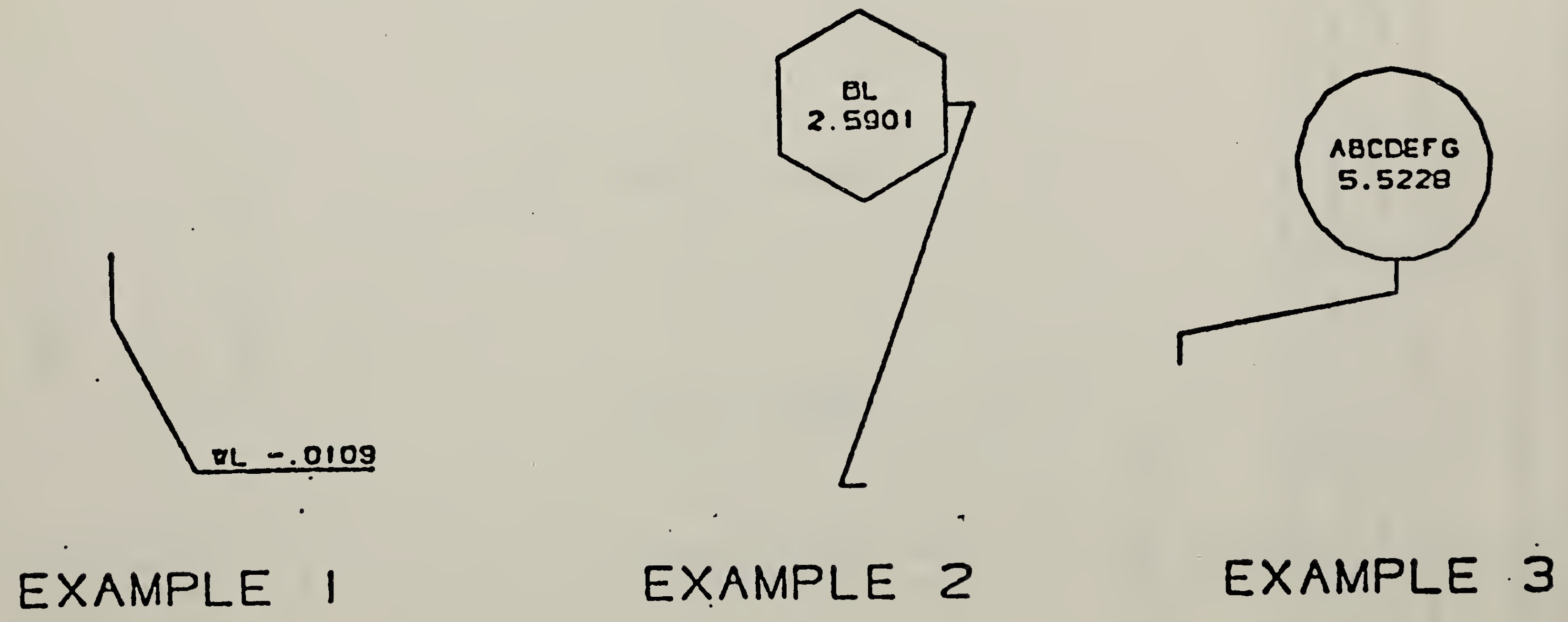


FIG. 4-20 EXAMPLES OF THE POINT DIMENSION ENTITY

4.2.13.3 Directory Data

ENTITY TYPE NUMBER : 220

4.2.13.4 Parameter Data

Parameter	Value	Format	Comment
1	DE1	Pointer	Pointer to general note directory entry
2	DE2	Pointer	Pointer to leader directory entry
3	DE3	Pointer	Pointer to circular arc, composite, or 0.
4	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
5	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
4+N	DE	Pointer	
5+N	M	Integer	Number of properties
6+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
5+N+M	DE	Pointer	

4.2.14 Radius Dimension Entity. A radius dimension consists of a general note, a leader, and an arc center point, (XT, YT). Refer to Figure 4-21 for examples of radius dimensions.

4.2.14.1 The arc center coordinates are used for positioning the radius dimension line relative to the arc being dimensioned.

4.2.14.2 Directory Data

ENTITY TYPE NUMBER: 222

4.2.14.3 Parameter Data

Parameter	Value	Format	Comment
1	DEN	Pointer	Pointer to general note directory entry
2	DEP	Pointer	Pointer to leader directory entry
3	XT	Floating Point	Arc center coordinates
4	YT	Floating Point	
5	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
6	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
5+N	DE	Pointer	
6+N	M	Integer	Number of properties

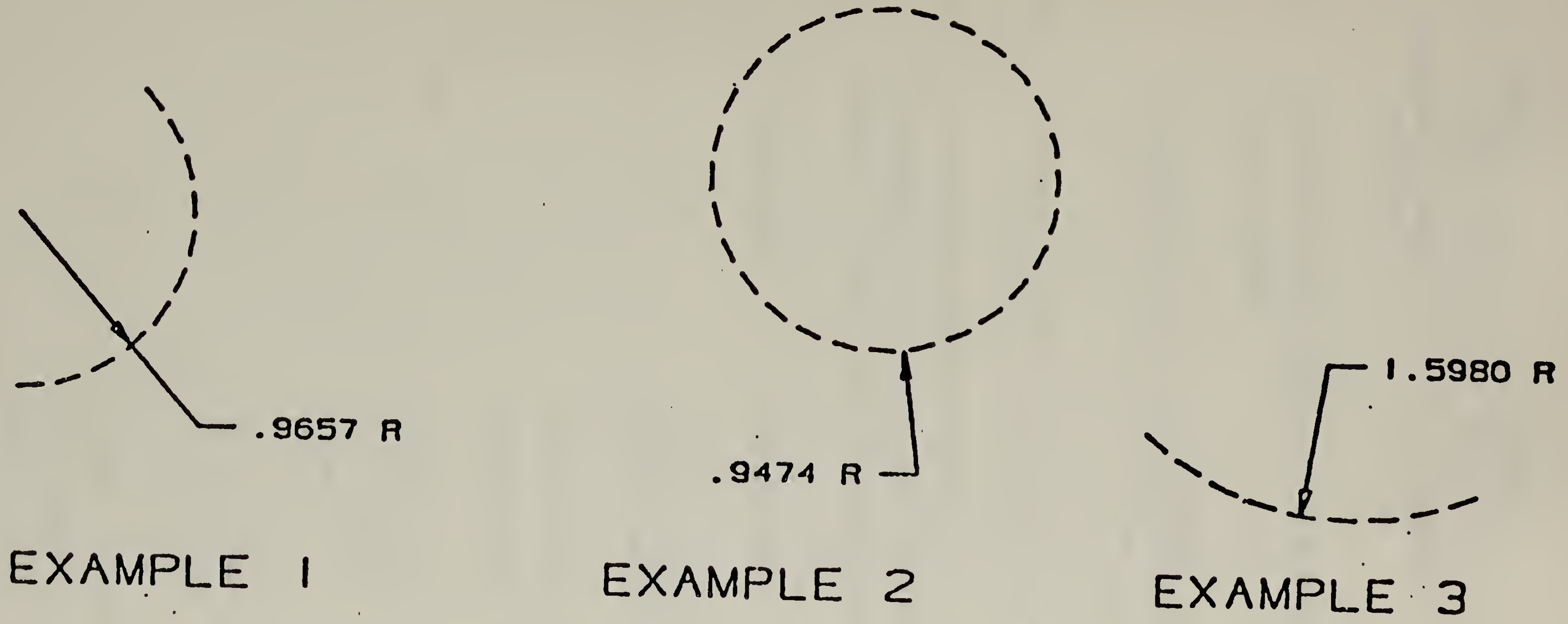


FIG. 4-21 EXAMPLES OF THE RADIUS DIMENSION ENTITY

Parameter	Value	Format	Comment
7+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Pointer	

4.2.15 Section Entity. A section entity is a copious data entity (see 3.5) of form numbers 31 to 38. The form number describes how the data are to be interpreted.

4.2.15.1 The point data contains a list of points (Xn, Yn) n=1, 2, . . . , N-1. (The Z value is constant.

The display symbol is described by line segments where each line is from (Xn, Yn, Zn) to (Xn+1, Yn+1, Zn+1) where n=1, 3, 5, ... , N-1.

4.2.15.2 These points represent line segments of the section display symbol.

4.2.15.3 The table below describes the use of the form number, and Figure 4-22 shows examples of each possible form number. (Reference: American National Standard, ANSI Y14.2M-1979, Line Conventions and Lettering)

Form number	Display action
31	Display all line segments as solid lines. (iron, brick, stone masonry)
32	Do not display every third line. All other lines display as solid. (steel)
33	Display odd numbered lines as solid. Display even numbered lines dashed. (bronze, brass, copper)
34	Do not display every fifth line. All other lines display as solid. (plastic, rubber)
35	Display odd numbered lines as solid. Display even numbered lines with a centerline line font. (fire brick, refractory material)
36	Display all lines with a centerline line font. (marble, slate, glass)

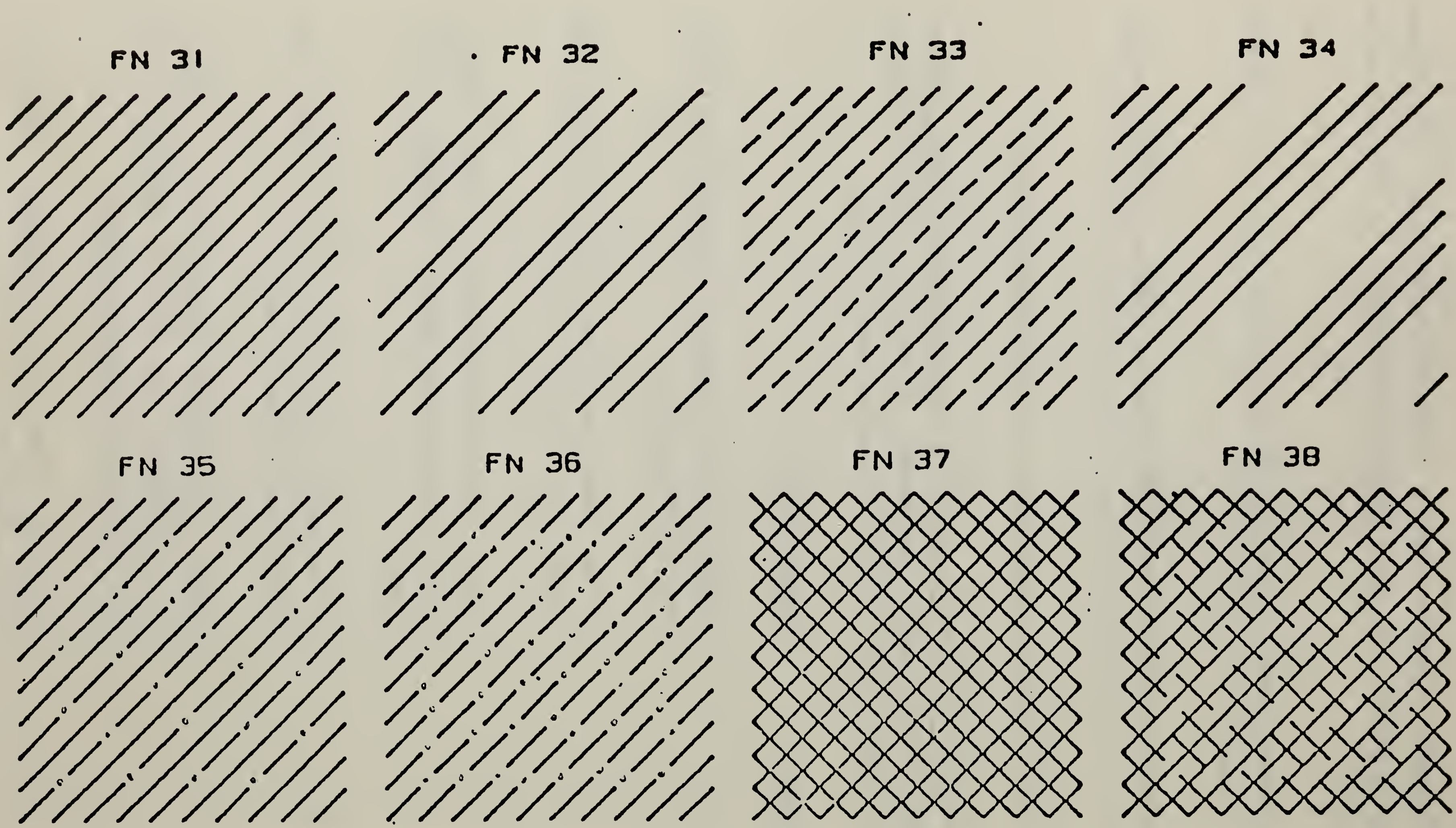


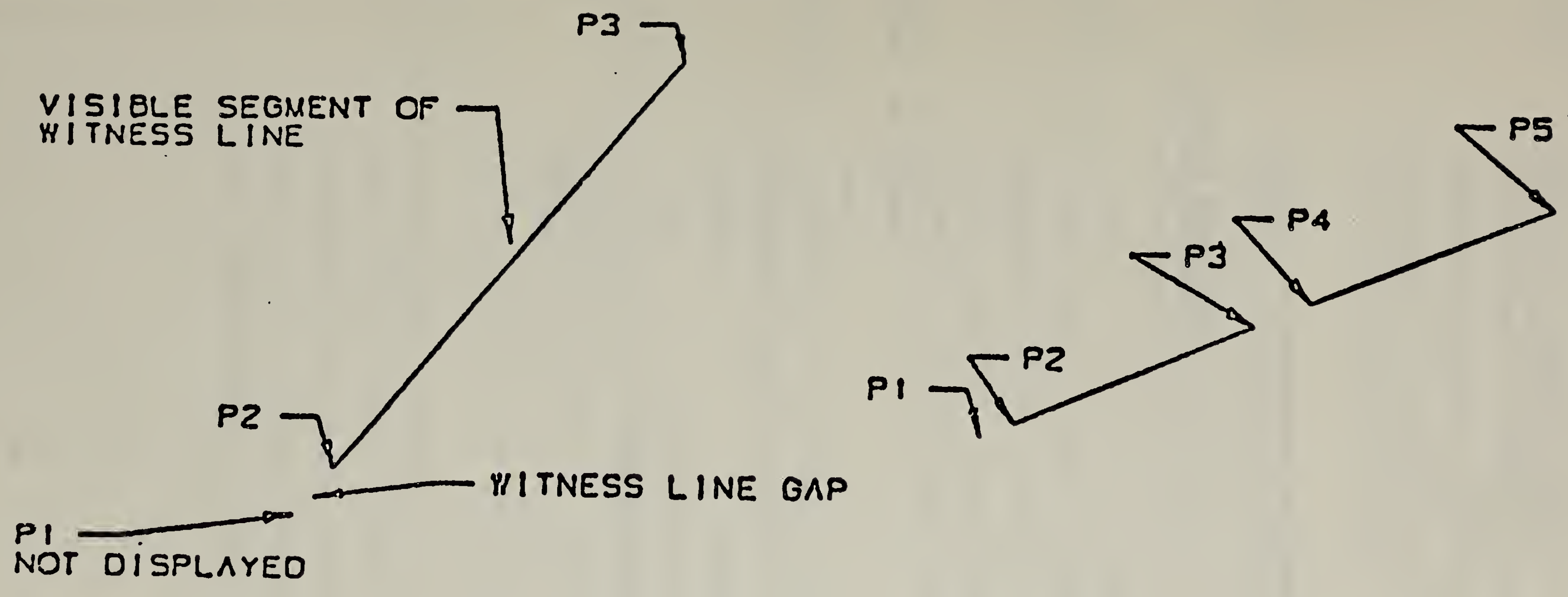
FIG. 4-22 EXAMPLES OF THE SECTION ENTITY
(FN = FORM NUMBER)

Form number	Display action
37	Display all line segments as solid lines. (lead, zinc, magnesium, sound or heat insulation, electrical insulation)
38	Display the first half of the lines with a solid line font. Display the last half of the lines with a dashed line font. (aluminum)

4.2.16 Witness Line Entity. A witness line is a form number 40 of a copious data block that contains one or more straight line segments associated with drafting entities of various types. Each line segment may be visible or invisible. Refer to Figure 4-23 for examples.

4.2.16.1 If the witness line is suppressed, this is indicated by a 0 in the pointer field of the drafting entity pointing to a witness line, or by setting the blank status of the directory entry of the copious data entity for the witness line.

4.2.16.2 Within the copious data, there will be the location from which the witness line gap must be maintained. This point is indicated in Figure 4-23 as P1. The location will be the first point in the copious data. P1 will be coincident with the geometry being dimensioned or equal to P2 when the location of the geometry is unknown. Note: for those annotation methods that do not allow drafting entities to be displaced from the plane of annotation, coincident with the geometry indicates that a line normal to the plane of annotation connects P1 and the point on the geometry being dimensioned. Note that all points must be colinear, and that the number of points will be odd and at least 3 (3, 5, 7, . . .), with alternating blank and displayed segments. The examples in Figure 4-20 show the blanking of segments and the order of points stored in the copious data.



EXAMPLE 1

EXAMPLE 2

FIG. 4-23 EXAMPLES OF THE WITNESS LINE ENTITY

4.3 Structure Entities

4.3.1 Entity Type/Type Number

The following entities are defined in this section:

Entity Type	Entity Type Number
Associativity Definition Entity	302
Associativity Instance Entity	402
Drawing Entity	404
Line Font Definition Entity	304
MACRO Definition Entity	306
MACRO Instance Entity	600-699 as specified by user
Property Entity	406
Subfigure Definition Entity	308
Singular Subfigure Instance Entity	408
Rectangular Array Subfigure Instance Entity	412
Circular Array Subfigure Instance Entity	414
Text Font Definition Entity	310
View Entity	410

4.3.2 Associativity Definition Entity. The associativity definition entity permits the preprocessor to define an associativity schema. That is, by using the associativity definition, the preprocessor defines the type of relationship. It is important to note that this mechanism specifies the syntax of such a relationship and not the semantics.

4.3.2.1 Schema. The definition schema allows the specification of multiple groups of data which are called classes. A class is considered to be a separate list, and the existence of several classes implies an association among the classes as well as among the contents of each class.

4.3.2.1.1 For each class, the schema has provision to specify whether or not back pointers are required. A back pointer being required implies that an entity which is a member of this associativity (when it is instanced) has a pointer to the directory entry of the associativity instance in its back pointer parameter section.

4.3.2.1.2 The provision in the schema to specify whether or not a class is ordered is used to state whether the order of appearance of entries in the class is significant.

4.3.2.1.3 In the schema, "ENTRIES" are the members of the class. However, each entry could be composed of several items. If multiple items are required, they will be ordered. For example, if the entries were locations, each entry might have three items to specify X, Y, and Z values.

4.3.2.1.4 The associativity definition will fix the number of classes for an associativity and the number of items per entry in a particular class. Each associativity instance will have a variable number of entries per class. In order to help decode instances of the definition, each item is specified as a pointer (to an entity directory entry) or a data value.

4.3.2.2 Form. Two kinds of associativity are permitted within the file. Pre-defined associativities will have form numbers in the range of 1 to 5000 and appear in 4.3.3.3. The second kind of associativity is defined in the file by a preprocessor (Form numbers 5001-9999). These definitions appear once in the file for each form of associativity defined, and allow the preprocessor to fill in the definition according to a schema which defines the details of the associativity.

4.3.2.2.1 The definition includes the associativity form, the number of class definitions, the number and type of items in each entry, and whether back pointers (from the entity to the associativity) are required. Each set of values (BP, Order, N, and Item type) are considered a class. The pre-defined associativity definitions are located in 4.3.3.3. See 4.3.3.3.1 for a complete example of associativity.

4.3.2.2.2 Directory Data

ENTITY NUMBER : 302

4.3.2.2.3 Parameter Data

Parameter	Value	Format	Comment
1	K	Integer	Number of class definitions .
2	BP1	Integer	1-back pointers required 2-no back pointers
3	OR1	Integer	1-ordered class 2-unordered class
4	N1	Integer	Number of items per entry
5	IT1(1)	Integer	1-pointer to a directory entry 2-value
.			
.			
4+N1			

The items in parameters 2 through 4+N1 are repeated for each of the K classes.

4.3.3. Associativity Instance Entity. Each time an associativity relation is needed in the file an associativity instance entity is used.

The form number of the associativity instance will identify the meaning of the entity. If the form number is between 1 and 5000, the definition is specified as described in 4.3.2.2 and the version number of the associativity instance will be 1. If the form number is between 5001 and 9999, an associativity definition will occur in the file and the version number field of the instance directory entry will contain a pointer to the directory entry of the associativity definition.

Each entity that is a member of an associativity instance can contain a back pointer (in the back pointer portion of its parameter data) to the associativity instance.

The parameters K and N1, N2, ...NK are specified in the associativity definition. (see 4.3.2.2.3)

4.3.3.1 Directory Entry

ENTITY TYPE NUMBER : 402

4.3.3.2 Parameter Data

Parameter	Value	Format	Comment
1	NE1	Integer	Number of entries for class one
2	NE2	Integer	Number of entries for class two
. . .			
K	NEK	Integer	Number of entries for class K

For K classes with (NE1,...,NEK) entries
with (N1, . . . , NK) items per entry

Parameter	Value	Format	Comment
K + 1	Class 1	Variable	Entry 1, Item 1
		.	Item 2
		.	. .
		.	. .
		.	. .
		Variable	Item N1
		Variable	Entry 2, Item 1
		.	.
		.	.
		.	.
		Variable	Item N1
		.	.
		.	.
		.	.
		Variable	Entry NE1, Item 1
		.	.
		.	.
		.	.
		.	Item N1
	Class 2	Variable	Entry 1, Item 1
		.	.
		.	.
		.	.
		Variable	Item N2
		Variable	Entry 2, Item 1
		.	.
		.	.
		.	.
		.	Item N2
		.	.
		.	.
		.	.
		Variable	Entry NE2, Item N2
	.	.	
	.	.	
	.	.	
	Class K	Variable	Entry 1, Item 1
X	Class K	Variable	Entry NEK, Item NK

Parameter	Value	Format	Comment
X+1	M	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
X+2	DE	Pointer	Pointers to associativities or general notes
.			
.			
.			
X+M+1	DE	Pointer	
X+M+2	N	Integer	Number of properties
X+M+3	DE	Pointer	Pointers to properties
.			
.			
.			
X+M+2+N	DE	Pointer	

4.3.3.3 Pre-defined Associativities. As defined in 4.3.2.2, the associativity definition entity will only occur for Form Numbers 5001 through 9999. The following paragraphs contain the definitions of the pre-defined associativities as they would appear if they were defined by a user. Also included in this Section are the descriptions of each associativity's parameters in a manner similar to other entities in this Specification.

4.3.3.3.1 Form Number: 1 Group

The Group Associativity allows a collection of a set of entities to be maintained as a single, logical entity. Figure 4-24 is an example.

The user should note Form Number 7 differs from Form Number 1 only in that back pointers are not required in Form Number 7.

DEFINITION

Parameter	Set Value	Meaning
1	1	One class
2	1	Back pointers required
3	2	Unordered
4	1	One item per entry
5	1	The item is a pointer

DESCRIPTION

Directory Data

ENTITY TYPE NUMBER: 402
FORM NUMBER: 1

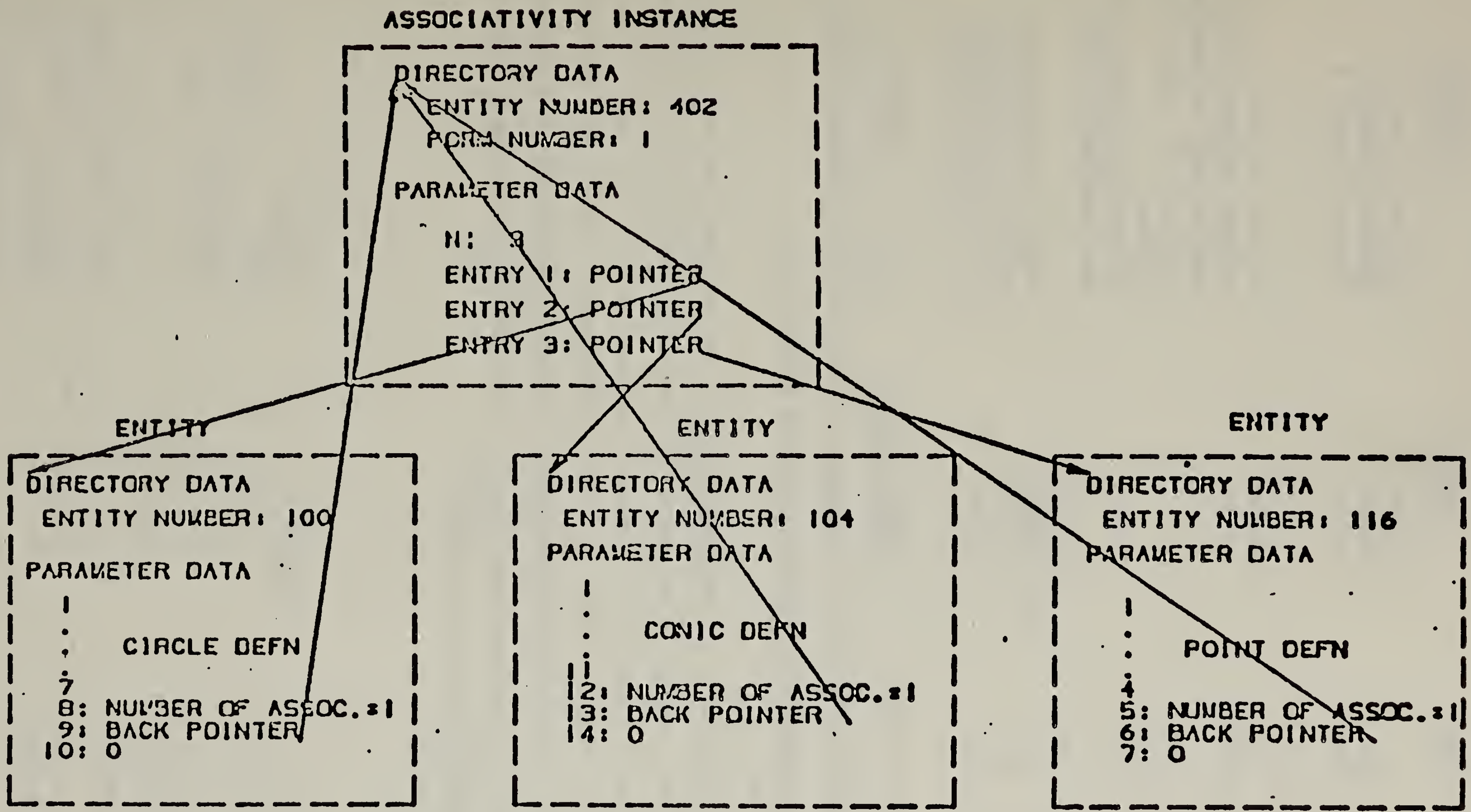


FIG. 4-24 ASSOCIATIVITY INSTANCE AND RELATED ENTITIES

Parameter Data

Parameter	Value	Format	Comment
1	N	Integer	Number of entries
2	DE	Pointer	Pointer to entity 1
3	DE	Pointer	Pointer to entity 2
.	.	.	
.	.	.	
N+1	DE	Pointer	Pointer to entity N
N+2	N1	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
N+3	DE	Pointer	Pointers to associativities or general notes
N+N1+2	DE	Pointer	
N+N1+3	M	Integer	Number of properties
N+N1+4	DE	Pointer	Pointers to properties
N+N1+M+3	DE	Pointer	

4.3.3.3.2 FORM NUMBER : 3 Views Visible

When an entity is to be displayed in a single view, a pointer to that view entity is entered in parameter 6 of the entity's DE.

If an entity is to be displayed in more than one view, parameter 6 of its DE contains a pointer to an instance of a form 3 associativity. This form of the associativity contains 2 classes of information. The first class contains the number of views visible followed by pointers to each of the view entities visible in the specific associativity instance. The second class contains the number of entities whose display is specified by this instance, followed by pointers to each of the entities.

DEFINITION

Parameter	Set Value	Meaning
1	2	Two classes
	(Class 1)	
2	1	Back pointers required
3	2	Unordered
4	1	One item per entry

5	1	Item is a pointer (to view entity)
	(Class 2)	
6	2	No back pointers
7	2	Unordered
8	1	One item per entry
9	1	Item is a pointer (to other entity)

DESCRIPTION

Directory Data

ENTITY TYPE NUMBER: 402
FORM NUMBER: 3

Parameter	Value	Format	Comment
1	N1	Integer	Number of views visible
2	N2	Integer	Number of entities displayed in these views
3	DEV1	Pointer	Pointer to view entity
.	.	.	
.	.	.	
N1+2	DEVN1	Pointer	
N1+3	DE1	Pointer	Pointer to entity whose display is being specified by this associativity instance
.	.	.	
.	.	.	
N1+N2+2	DEN2	Pointer	Pointer to entity N2
N1+N2+3	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
N1+N2+4	DE	Pointer	Pointers to associativities or general notes
.	.	.	

N1+N2+N+3	DE	Pointer	
N1+N2+N+4	M	Integer	Number of properties
N1+N2+N+5	DE	Pointer	Pointers to properties
.	.	.	
N1+N2+N+M+4	DE	Pointer	

4.3.3.3.3 FORM NUMBER: 4 Views Visible, Pen, Line Weight

This associativity is an extension of Form Number 3. For those entities that are visible in multiple views, but must have a different line font, pen number, or line weight in each view, there will be an occurrence of the associativity instance Form Number 4.

In the parameter data portion of the associativity instance, the parameter N1 will indicate the number of blocks containing the view visible, line font, pen number, and line weight specifications. Each block will contain a pointer to the view entity, a line font value or 0, a pointer to a line font directory entry if the line font value was 0, a pen value, and a line weight value. Parameter N2 will contain the number of entities which are members of this associativity (i.e., entities which have this particular display characteristic).

Note that N2 may often be 1. If more than one entity appears in Class 2 the complete set of display characteristics in Class 1 apply to each entity in Class 2.

Parameter	Set Value	Meaning
1	2	Two classes
	(Class 1 (View))	
2	1	Back pointers required
3	2	Unordered
4	5	Five items per entry
	(Entry template)	
5	1	Pointer to view directory entry

6	2	Line font value
7	1	Pointer to line font directory entry
8	2	Pen number (value)
9	2	Line weight (value)
	Class 2 (entity)	
10	2	No back pointers required
11	2	Unordered
12	1	One item per entry
13	1	Item is a pointer (to entity)

DESCRIPTION

Directory Data

ENTITY TYPE NUMBER: 402
FORM NUMBER: 4

PARAMETER DATA

Parameter	Value	Format	Comment
1	N1	Integer	Number of blocks containing the view visible, line font, pen number, and line weight information
2	N2	Integer	Number of entities which have this particular set of display characteristics
3	DEV1	Pointer	Pointer to view entity 1
4	LF	Integer	Line font value or 0
5	DEF1	Pointer	If parameter 4 = 0, pointer to a line font definition. Otherwise = 0.
6	PN1	Integer	Pen number value 1

7	LW1	Integer	Line weight value 1
8	DEV2	Pointer	Pointer to view entity 2
.	.	.	
.	.	.	
5*N1+2	LWN1	Integer	Line weight value N1
5*N1+3	DE1	Pointer	Pointer to entity 1
.	.	.	
.	.	.	
5*N1+N2+2	DE	Pointer	Pointer to entity N
5*N1+N2+3	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
5*N1+N2+4	DE	Pointer	Pointers to associativities or general note entities
.	.	.	
5*N1+N2+N+3	DE	Pointer	
5*N1+N2+N+4	M	Integer	Number of properties
5*N1+N2+N+5	DE	Pointer	Pointers to properties
5*N1+N2+N+M+4	DE	Pointer	

4.3.3.3.4 FORM NUMBER: 5 Entity Label Display

Some entities may have one or more possible displays for their entity labels, depending on the view in which they are being displayed. For those entities, there will be an occurrence of the associativity instance Form Number 5.

In the parameter data portion of the associativity instance, the parameter N1 will indicate the number of blocks containing label placement information. Each block will contain a pointer to a view entity which specifies the view of visibility. The remaining information (text location, leader, and level number) applies to the label for that view.

DEFINITION

Parameter	Set Value	Meaning
1	1	One class
2	2	No back pointers
3	1	Ordered
4	7	Seven items per entry
5	1	Pointer to view directory entry
6	2	XT of text location
7	2	YT of text location
8	2	ZT of text location
9	1	Pointer to leader directory entry
10	2	Entity label level number
11	1	Pointer to entity

DESCRIPTION

Directory Data

ENTITY TYPE NUMBER: 402
FORM NUMBER: 5

PARAMETER DATA

Parameter	Value	Format	Comment
1	N1	Integer	Single entry
2	DEV	Pointer	Pointer to a view entity
3	XT	Floating Point	XT coordinate of text location
4	YT	Floating Point	YT coordinate of text location
5	ZT	Floating Point	ZT coordinate of text location
6	DEL	Pointer	Pointer to leader
7	LLN	Integer	Entity label level number
8	DE1	Pointer	Pointer to entity which is being displayed
.	.	.	
.	.	.	
7*N1+1	DEN1	Pointer	
7*N1+2	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
7*N1+3	DE	Pointer	Pointers to associativities or general notes
.	.	.	
7*N1+N+2	DE	Pointer	
7*N1+N+3	M	Integer	Number of properties
7*N1+N+4	DE	Pointer	Pointers to properties
.	.	.	
7*N1+N+M+3	DE	Pointer	

4.3.3.3.5 FORM NUMBER: 6 View List

This associativity has two classes. The first class has only one entry which is a pointer to the directory entry of a specific view. The second class is a list of entities (pointers to their respective directory entities) which are visible in the view referenced in Class 1. Back pointers are required in both classes; the view as well as all entities visible in the view must have pointers to this associativity instance.

DEFINITION

Parameter	Set Value	Meaning
1	2	Two classes
	Class 1 (view)	
2	1	Back pointers required
3	2	Unordered
4	1	One item per entry
5	1	Pointer to view directory entry
	Class 2 (Entities)	
6	1	Back pointers required
7	2	Unordered
8	1	One item per entry
9	1	Pointer to directory entry of entity visible in view referred to in parameter 5

DESCRIPTION

Directory Data

ENTITY TYPE NUMBER: 402
FORM NUMBER: 6

PARAMETER DATA

Parameter	Value	Format	Comment
1	1	Integer	Single entry in first class
2	N1	Integer	Number of entities in second class
3	DEV	Pointer	Pointer to view entity
4	DE1	Pointer	Pointers to entities visible in view specified in parameter 3
N1+3	DEN1	Pointer	
N1+4	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
N1+5	DE	Pointer	Pointers to associativities or general notes
.	.	.	
N1+N+4	DE	Pointer	
N1+N+5	M	Integer	Number of properties
N1+N+6	DE	Pointer	Pointers to properties
.	.	.	
N1+N+M+5	DE	Pointer	

4.3.3.3.6 FORM NUMBER: 7 Group Without Back Pointers

DEFINITION

Parameter	Set Value	Meaning
1	1	One class
2	2	Back pointers not required
3	2	Unordered
4	1	One item per entry
5	1	The item is a pointer

DESCRIPTION

Directory Data

ENTITY TYPE NUMBER: 402
FORM NUMBER: 7

PARAMETER DATA

Parameter	Value	Format	Comment
1	N	Integer	Number of entries
2	DE	Pointer	Pointer to entity 1
.	.	.	
.	.	.	
N+1	DE	Pointer	Pointer to entity N
N+2	N1	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
N+3	DE	Pointer	Pointers to associativities or general notes
.	.	.	
N+N1+2	DE	Pointer	
N+N1+3	M	Integer	Number of properties
N+N1+4	DE	Pointer	Pointers to properties
N+N1+M+3	DE	Pointer	

4.3.3.3.7 FORM NUMBER: 8 Signal String

This associativity has four classes and is intended to represent a single signal string. Class one provides all names of the signal in an order that should be preserved. Class two collects together a set of connection nodes in the string and thus can be considered as specifying the connections for the signal. Class three relates the signal string to set of geometric entities on a schematic drawing, while class four accomplishes the same thing with respect to the implemented board or chip.

The geometric entities which may be members of classes 2 and 3 include composite, copious (forms 11 or 12), or any of the entities which may be members of composite.

DEFINITION

Parameter	Set Value	Meaning
1	4	Four classes
	Class One -- Signal Names	
2	2	No back pointers
3	1	Ordered
4	1	One item per entry
5	2	Item is value
	Class Two -- Connections	
6	1	Back pointers
7	2	Unordered
8	1	One item per entry
9	1	Pointer to Connect Node
	Class Three -- Schematic	
10	1	Back pointers
11	1	Ordered
12	1	One item per entry
13	1	Pointer to geometry
	Class Four -- Board Implementation	
14	1	Back pointers
15	1	Ordered
16	1	One item per entry
17	1	Pointer to geometry

DESCRIPTION

Directory Data

ENTITY TYPE NUMBER: 402

FORM NUMBER: 8

Parameter	Value	Format	Comment
1	NS	Integer	Number of signal names
2	N1	Integer	Number of Connection Nodes
3	N2	Integer	Number of Entities in Schematic signal string
4	N3	Integer	Number of Entities in PWB signal string
5	SIG1	String	Signal name
•	•		
NS+4	SIGNS		
•	•	•	
NS+5	PTR	Pointer	Pointer to Connect Nodes
NS+N1+4	PTR1	Pointer	
NS+N1+5	PTR2	Pointer	Pointer to Entity in Schematic Signal String
NS+N2+N1+4	PTR2	Pointer	
NS+N2+N1+5	PTR3	Pointer	Pointer to entity in PWB signal string
NS+N3+N2+N1+4	PTR3	Pointer	
k (number of entries) = NS+N3+N2+N1+4			
k+1	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general notes)
k+2	DE	Pointer	Pointer to either an associativity or to a general note
•	•	•	
k+N+1	DE	Pointer	
k+N+2	M	Integer	Number of properties
k+N+3	DE	Pointer	Pointer to property
•	•	•	
k+N+M+2		Pointer	

4.3.3.3.8 FORM NUMBER: 9 Single Parent Associativity

This associativity defines a logical structure of one independent (parent) entity and one or more subordinate (children) entities.

Both parent and child entities require back pointers to this instance. Any necessary display parameters are governed by the parent entity.

DEFINITION:

Parameter	Set Value	Meaning
1	1	One Class
2	1	Back pointers required
3	1	Ordered
5	1	The item is a pointer

DESCRIPTION

Directory Data

ENTITY TYPE NUMBER: 402
FORM NUMBER: 9

PARAMETER DATA

Parameter	Value	Format	Comment
1	1	Integer	Single Parent Entity Pointer to Parent Entity
2	NC	Integer	Number of Children
3	DE	Pointer	Pointer to Parent Entity
4	DE1	Pointer	Pointer to Child Entity 1
. . .	. . .	. . .	
2+NC	DENC	Pointer	Pointer to Child Entity NC
3+NC	N	Integer	Number of Back Pointers (to associativity entities)/text pointers (to general note entities)

Parameter	Value	Format	Comment
4+NC	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
3+NC+N	DE	Pointer	
4+NC+N	M	Integer	Number of Properties
5+NC+N	DE	Pointer	Pointers to Properties
.	.	.	
.	.	.	
.	.	.	
4+NC+N+M	DE	Pointer	

4.3.3.3.9 FORM NUMBER: 10 Text Node

The purpose of the text node is to act as a template for future addition of text. It is defined as an associativity to allow it to refer to multiple instances of itself in those cases in which it is instanced as part of a subfigure definition.

In accordance with the general rule of multiple-instanced entities, digits 5-6 of directory entry field 9 have the value 04, and class 1 consists of a pointer to a point representing its original location followed by pointers to multiple instances, if these exists.

Class 2 consists of those parameters of the general note which are pertinent to the definition of a text template, as opposed to text itself. In general, these consist of all parameters but the text string. The location is omitted because it is included in class 1 as a pointer to a point representing the geometric location of the text node.

An instance of a text node consists of this associativity, a point indicating the position of the instance, and 1 or more general notes atached to the node through the text pointers of the geometric entities. If parameters in the general notes are null, the value of the same parameter in class 2 of the associativity is taken as the default; non-null parameters over-ride the defaults. In the cases of multiple instances from a subfigure, the general notes representing text will be attached to the instance point (pointers 2, 3, . . . in class 1).

As a text-type entity, the text node can be pointed to by the back pointer/text pointer field in each IGES entity.

Note that the associativity definition has an unusual value for parameter 11 (font characteristic). The value 3 implies either a pointer or a data item. A positive value implies a data item; a negative value implies the absolute value is to be taken as a pointer.

DEFINITION

Parameter	Set Value	Meaning
1	2	Two classes
	(Geometry Pointers)	
2	1	Back pointers required
3	1	Ordered class
4	1	One item per entry
5	1	Item is pointer (to point entity)
	(Text Description)	
6	2	Back pointers not required
7	1	Ordered class
8	7	Seven items/entry
9	2	Box length
10	2	Box height
11	3	Font characteristic
12	2	Slant angle
13	2	Rotation angle
14	2	Mirror flag
15	2	Rotate internal flag

DESCRIPTION

Directory Data

ENTITY TYPE NUMBER: 402
FORM NUMBER: 10

Parameter	Value	Format	Comment
1	NP	Integer	Number of geometry pointers
2	1	Integer	One item in class 2
3	GP1	Pointer	Pointer to point (original location)
4	GP2	Pointer	Pointer to instance point (first instance)
.	.	.	
.	.	.	
NP+2	GPNP	Pointer	Pointer to instance point (NP-1 instance)
NP+3	WT	Floating point	Box length
NP+4	HT	Floating point	Box height
NP+5	FC	Integer or pointer	Font characteristic
NP+6	SL	Floating point	Slant angle of text in radians. /2 is the value for no slant angle and is the default value.

NP+7	A	Floating point	Rotation angle in radians for text.
NP+8	M	Integer	Mirror flag (0=no mirror, 1=YT mirror axis, 2=XT mirror axis.)
NP+9	VH	Integer	Rotate internal text flag (0=text horizontal, 1=text vertical)

k (number of entries) =NP+9

k=1	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general notes)
k=2	DE	Pointer	Pointer to either an associativity or to a general note
.	.	.	
k+N+1	DE	Pointer	
k+N+2	M	Integer	Number of properties
k+N+3	DE	Pointer	Pointer to property
.	.	.	
k+N+M+2	DE	Pointer	

4.3.3.3.10 FORM NUMBER: 11 Connect Node

The purpose of the connect node is to imply a logical connection betwen one or more entities. In the case of an electrical application, this logical connectivity would mean an electrical connection, but the connect node has applicability in other applications such as piping.

The connect node is defined as a 2-class associativity with the second class undefined.

In accordance with the general rule of multiple-instanced entities, digits 5-6 of directory entry field 9 have the value 04, and class 1 consists of a pointer to the geometry representing the original location of the connect node, followed by pointers to multiple instances, if these exist. Each of the geometry entities is the point entity. In the case of a singly-instanced connect node, the point represents the position of the connect node. In the case of a multiply-instanced connect node (i.e., a connect node in a subfigure definition), the first point in the class represents the defining location (in the subfigure definition), while the remaining points represent instance locations of the connect node.

The second class is intended to describe the properties of the connect node such as physical connection constraints. Its definition will be developed in the future when these requirements become more clear.

The name of a connect node is found in its entity label. If the name is longer than 8 characters, the entity label is blank, and the name is found in a name property attached to the entity. In the case of multiply-instanced connect nodes, separate names can be attached to the instance points by the same means.

DEFINITION

Parameter	Set Value	Meaning
1	2 (Geometry pointers)	Two classes
2	1	Back pointers required
3	1	Ordered class
4	1	One item per entry
5	1	Item is pointer (to point entity)
6	2	Back pointers not required
7	2	Unordered class
8	1	One item per entry
9	2	Item is data

DESCRIPTION

Directory Data

ENTITY NUMBER: 402
FORM NUMBER: 11

Parameter	Value	Format	Comment
1	NC	Integer	Number of pointers (to points)
2	NP	Integer	Number of entries in second class

3	PT1	Pointer	Pointer to defining point (original location)
4	PT2	Pointer	Pointer to instance point (first instance)
.	.	.	
.	.	.	
NC+2	PTNC	Pointer	Pointer to last instance point (NC-1 instance)
NC+3	DT1	Data	First data entry
.	.	.	
.	.	.	
NC+NP+2	DTNP	Data	Last data entry
k (number of entries) = NC + NP + 2			
k+1	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general notes)
k+N+1	DE	Pointer	
k+N+2	M	Integer	Number of properties
k+N+3	DE	Pointer	Pointer to property
.	.	.	
k+N+M+2	DE	Pointer	

4.3.4 Drawing Entity. The drawing entity defines a collection of annotation entities and views of geometrical entities which, together, constitute a single representation of the part, in the sense that a real drawing constitutes a single representation of a part in standard drafting practice. If desired, multiple drawings can be included in a single file, referring to the same model space.

Annotation entities can appear either in the drawing entity using parameters 3N+2,..., or in individual views.

The (XD, YD) coordinates define a drawing space which is different from either model space or defintion space. The matrix associated with the view entity transforms entities from model space into drawing space. Note that drawing space is 2-dimensional, rather than 3-dimensional, as is the case with model or definition space.

Refer to Figures 4-25 and 4-26 for an example of the use of the drawing entity.

4.3.4.1 Directory Data

ENTITY TYPE NUMBER : 404

4.3.4.2 Parameter Data

Parameter	Value	Format	Comment
1	N	Integer	Number of view pointers
2	VPTR1	Pointer	Pointer to directory entry of first view entity
3	XD1	Floating Point	Location in drawing coordinates of origin of
4	YD1	Floating Point	transformed view
5	VPTR2	Pointer	Pointer to directory entry of second view entity

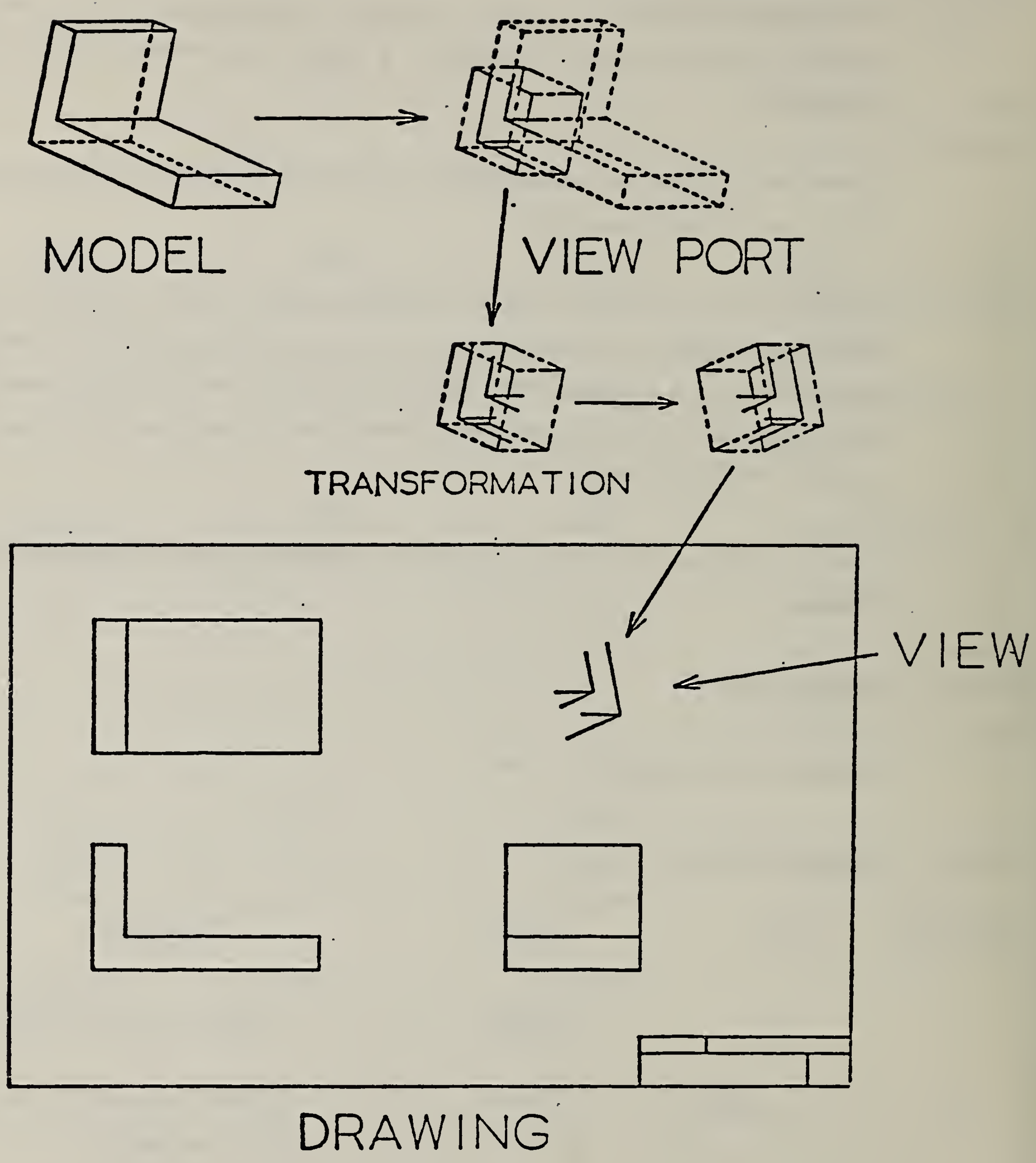


FIG. 4-25 DRAWING ENTITY EXAMPLE 1

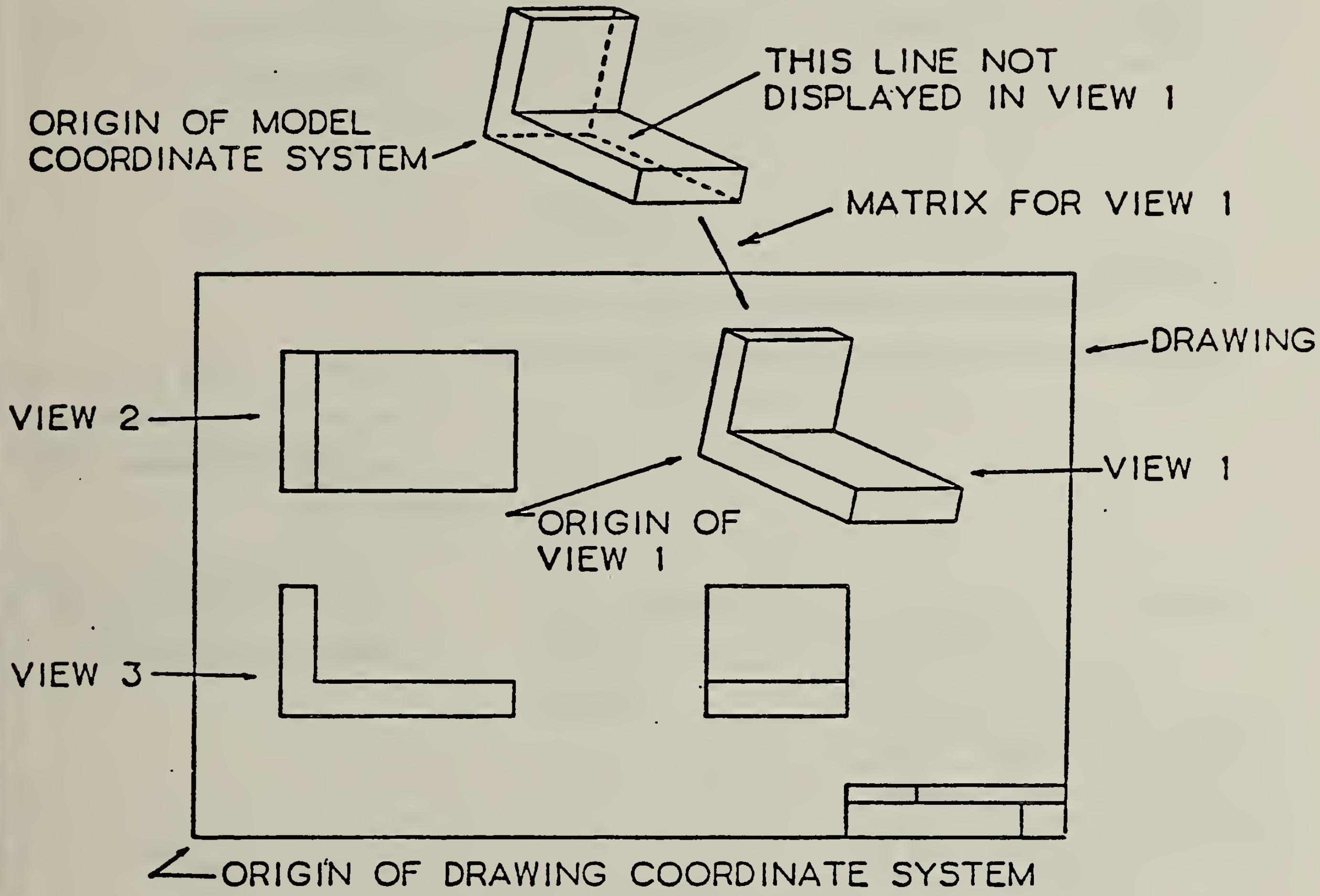

THIS LINE NOT DISPLAYED IN VIEW 1
ORIGIN OF MODEL COORDINATE SYSTEM
MATRIX FOR VIEW 1
DRAWING
VIEW 2
VIEW 1
ORIGIN OF VIEW 1
VIEW 3
ORIGIN OF DRAWING COORDINATE SYSTEM

FIG. 4-26 DRAWING ENTITY EXAMPLE 2

Parameter	Value	Format	Comment
.	.		
.	.		
3N+2	M	Integer	Number of annotation entities
3N+3	DPTR1	Pointer	Pointer to first annotation entity in this drawing
.	.		
.	.		
.	.		
TE=3N+M+2			
TE+1	NA	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
TE+2	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
TE+1+NA	DE	Pointer	
TE+2+NA	MA	Integer	Number of properties
TE+3+NA	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
TE+2+NA+MA	DE	Pointer	

4.3.5 Line Font Definition Entity.

This entity is used only to generate line fonts. A repeating pattern is specified with on or off segments. A repeating subfigure pattern is specified when Form=1. If Form=1, parameter 3 indicates the length of the first segment of the subfigure pointed to in parameter 2, and parameter 4 is the scale factor to be applied to each instance of the subfigure. If Form=1, parameters labeled 5 through M+2 will not occur, and the parameters labeled M+3, M+4, are actually 5, 6. Figure 4-27 demonstrates the use of the line font pattern. Figure 4-28 demonstrates the use of a subfigure (Form=1).

Form	Meaning
1	Parameter data contains pointer to subfigure.
2	Parameter data contains repeating structure description

4.3.5.1 Directory Data

ENTITY TYPE NUMBER: 304

4.3.5.2 Parameter Data

Parameter	Value	Format	Comments
1	M	Integer	FORM=1 (Subfigure) M equals zero (0) is used to specify that each subfigure instance is oriented with its axes in the same orientation as the axes of the definition space, respectively. M equals one (1) is used to specify that for each subfigure instance, the subfigure is oriented with its X-axis tangent to the referencing entity at the instance origin and in the direction of the entity, and the Y-axis is defined by the cross product of the Z-axis of the definition space with the X-axis tangent vector. The referencing entity must lie in a plane parallel to the XT, YT plane.

			FORM=2 (Repeating structure) Number of segments in the minimum repeating structure of the line font (M≤16).
2	L1	Floating Point (or Pointer)	Length of the first segment. If FORM=1, this is a pointer to a subfigure to be used as the repeat pattern.
3	L2	Floating Point	Length of the second segment. If FORM=1, L2 is the length of the repeating pattern.
4	L3	Floating Point	If FORM=1, L3 is a scale factor to be applied to the subfigure. Otherwise L3 is the length of the third segment.
.			
.			
.			
M+1	LM	Floating Point	Length of the Mth segment.
M+2	B	String	Up to 4 hexadecimal digits representing whether or not a segment is blank or not blank. Ex: '5' would indicate that segments 1 and 3 were not blank. Bits are right justified.
M+3	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
M+4	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
M+N+3	DE	Pointer	
M+N+4	MP	Integer	Number of properties
M+N+5	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
M+N+MP+4	DE	Pointer	

M+2	B	String	Up to 4 hexadecimal digits representing whether or not a segment is blank or not blank. Ex: '5' would indicate that segments 1 and 3 were not blank. Bits are right justified.
M+3	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
M+4	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
M+N+3	DE	Pointer	
M+N+4	MP	Integer	Number of properties
M+N+5	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
M+N+MP+4	DE	Pointer	

M = 8

L1 THRU L8 = 1

CHAR = A1

BIT PATTERN = 10100001

RESULTING FONT:

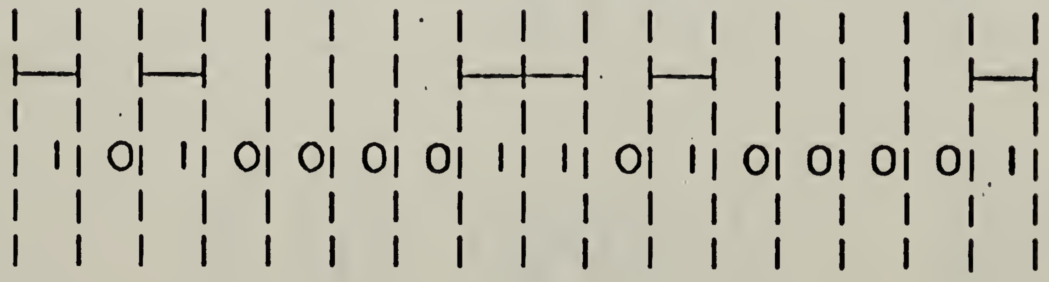


FIG. 4-27 CONSTRUCTION OF LINE FONTS

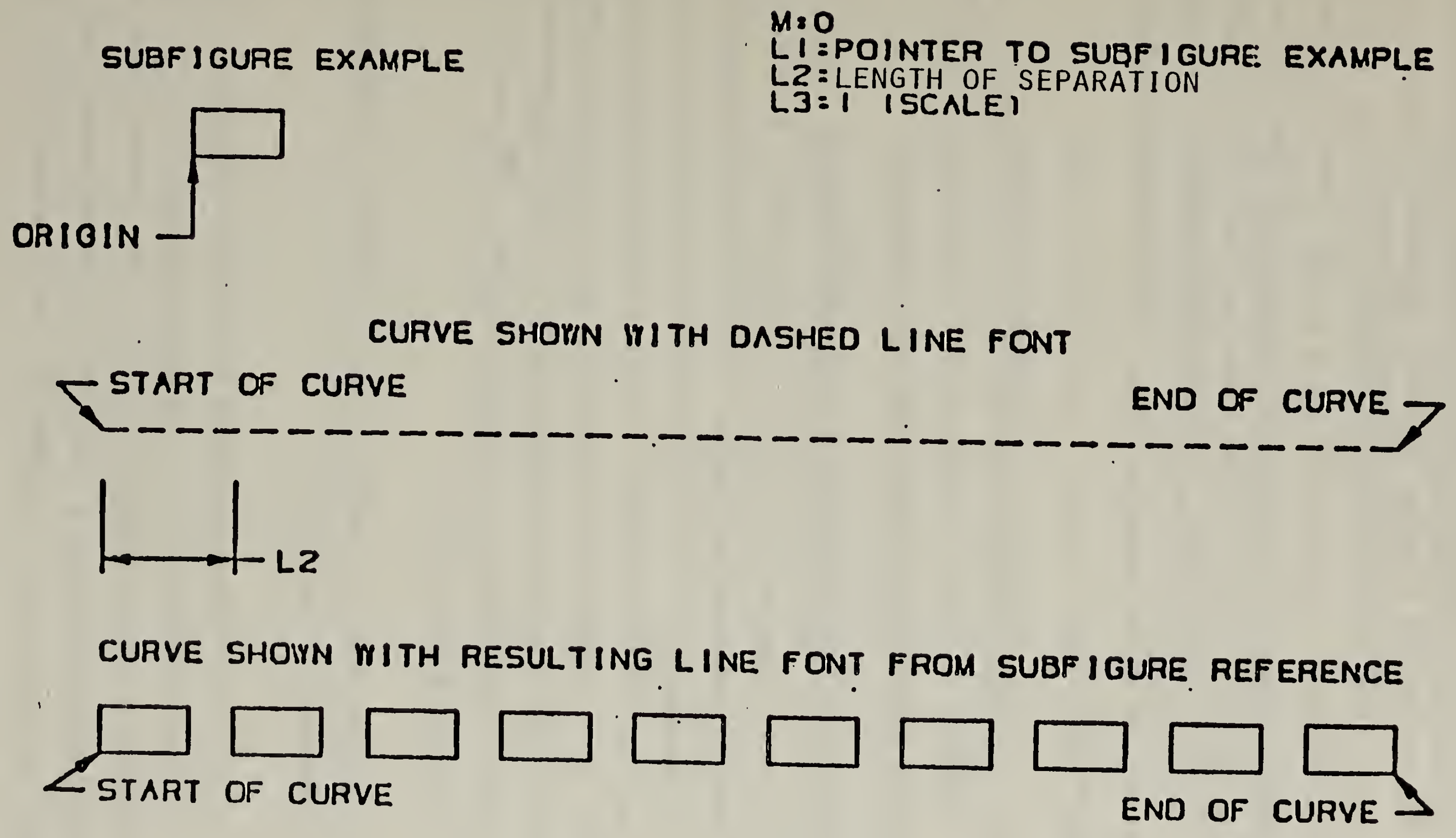

SUBFIGURE EXAMPLE
ORIGIN
M:O
L1:POINTER TO SUBFIGURE EXAMPLE
L2:LENGTH OF SEPARATION
L3:1 (SCALE)
CURVE SHOWN WITH DASHED LINE FONT
START OF CURVE
END OF CURVE
L2
CURVE SHOWN WITH RESULTING LINE FONT FROM SUBFIGURE REFERENCE
START OF CURVE
END OF CURVE

FIG. 4-28 USE OF SUBFIGURE WITH LINE FONTS

4.3.6 MACRO Capability

4.3.6.1 General. This Specification provides a means for communicating 3-dimensional and 2-dimensional geometric models and drawings. The Specification, however, does not provide a format for every geometric or drafting entity available on all currently used CAD/CAM systems, and is thus a common subset of such entities. To allow exchange of a larger subset of entities - a subset containing some of the entities not defined in this Specification but which can be defined in terms of the basic entities, the MACRO capability is provided. This capability allows the use of the Specification to be extended beyond the common entity subset, utilizing a formal mechanism which is a part of the Specification.

The MACRO capability provides for the definition of a "new" entity in terms of other entities. The "new" entity schema is provided in a MACRO definition which occurs once for every "new" entity in the file.

A MACRO definition is written using the MACRO definition entity. The parameter section of the entity contains the MACRO body. In the MACRO body, six types of statements are usable. The statements permissable are the assignment statement (LET), the entity definition statement (SET), and the REPEAT and CONTINUE statements. The repeat statement defines a portion of the body to be repeated and is terminated by a CONTINUE statement. References to other MACRO definitions appear on a MREF statement. The MACRO body is terminated by an ENDM statement. The details of the MACRO syntax are given in 4.3.6.5.

Each of the statements in a MACRO definition entity is terminated by a record delimiter (default: semicolon).

In order to use a "new" entity defined by the MACRO definition, a MACRO instance is placed in the file. The directory entry portion of an instance specifies the new entity type number in field 1 of the directory entry record and refers to the definition by a pointer in the version number field. The parameters to the instance are placed in the parameter section of the instance.

The directory entry section of a MACRO definition has the standard form. The attributes 4 through 9, 12 through 15, 18, and 19 have no significance. The default values for these attributes are taken from the directory section of the MACRO instance (described in 4.3.6.3).

The parameter data section of a MACRO definition consists of MACRO language statements in the ASCII character set. The statements are not in hollerith form, i.e., they have no preceding "H" specification. The statements are free format and may branch over record boundaries. Every statement is terminated by a record delimiter.

A processor for the MACRO language in the Specification has been developed and is available from the National Bureau of Standards.

4.3.6.2 MACRO Definition Entity. The MACRO definition entity specifies the action of a specific MACRO. After having been specified in a definition entity, the MACRO can be used as often as necessary by means of the MACRO instance entity.

The MACRO definition entity differs from other entity structures in this Specification by consisting only of text strings in the parameter data. The text strings constituting the statements in the MACRO definition are not set off by means of the nH structure of other text strings in this Specification but rather consist only of the actual string terminated by a record delimiter.

4.3.6.2.1 Directory Data

ENTITY TYPE NUMBER: 306

4.3.6.2.2 Parameter Data

Parameter	Value	Format	Comments
1	TEXT	Language statement	First statement
2	TEXT	Language statement	Second Statement
.	.	.	.
.	.	.	.
.	.	.	.
N	TEXT	Language statement	Last statement

4.3.6.3 MACRO Instance Entity. A MACRO instance entity is used to invoke a MACRO. The parameter data records of the instance contain values for the arguments to the MACRO. This is similar to a standard entity entry.

The directory entry for a MACRO instance entity contains the attribute values that are to be used as the default values during the expansion of the MACRO. The only special field is the VERSION field, which contains a pointer to the directory entry of the MACRO definition. The pointer is preceded by a minus sign.

4.3.6.3.1 Directory Data

ENTITY TYPE NUMBER: As defined for each MACRO in the range 600 to 699.

Version field: Pointer to directory entry of MACRO definition, preceded by a minus sign.

Other attributes: Default values to be used during expansion of the MACRO. Attributes listed as defaulting to /HDR obtain their values from here. (See page 254.)

4.3.6.3.2 Parameter Data

The parameter data section for an instance has the following form:

Parameter	Value	Format	Comment
1	n	Integer	Entity type number of the MACRO
2,...K	As appropriate for the particular MACRO		The values for the arguments to the MACRO. They must agree in format and number with the arguments in the MACRO statement of the definition.

Parameter	Value	Format	Comment
K+1	N	Integer	Number of back pointers (to associativity entities)/ text pointers (to general note entities)
K+2	DE	Pointer	Pointers to associativities or general notes
K+N+1	DE	Pointer	
K+N+2	M	Integer	Number of properties
K+N+3	DE	Pointer	Pointers to properties
K+N+M+2	DE	Pointer	

4.3.6.4 Examples of MACRO usage:

The following MACRO creates an isosceles triangle, given a vertex point, a base width, and a height.

Directory Data

ENTITY TYPE NUMBER: 306

Parameter Data

```
MACRO, 621, X1, Y1, A1, A2, K;

LET Z = 0;

SET #Line1      =      110, X1, Y1, Z, K*(X1+A1), (Y1+A2/2.), Z, 0, 0;

SET #Line2      =      110, K*(X1+A1), (Y1+A2/2.), Z,
                       K*(X1+A1), (Y1-A2/2.), Z, 0, 0;

SET #Line3      =      110, K*(X1+A1), (Y1-A2/2.), Z
                       X1, Y1, Z, 0, 0;

ENDM;
```

The MACRO can be used to create a triangle by creating an instance which supplies the needed values for X1, Y1, A1, A2, and K.

The parameter data section for the MACRO would have the following format:

Parameter	Value	Format	Comment
1	621	Integer	Entity type number of MACRO
2	X	Floating point	X coordinate of vertex
3	Y	Floating point	Y coordinate of vertex
4	A1	Floating point	Height of triangle
5	A2	Floating point	Base of triangle
6	K	Integer	Scaling factor

In particular, to create a triangle with a base of 5. and a height of 17., with a vertex at (0,0), and scale factor 1, the following instance could be placed into the file:

Directory data: Entity type number 621

Version: -nnn, where "nnn" is the sequence number of the directory entry of the definition.

Other attributes: As desired for default values during MACRO expansion.

Parameter data:

621, 0., 0., 17., 5., 1;

(See Figure 4-29)

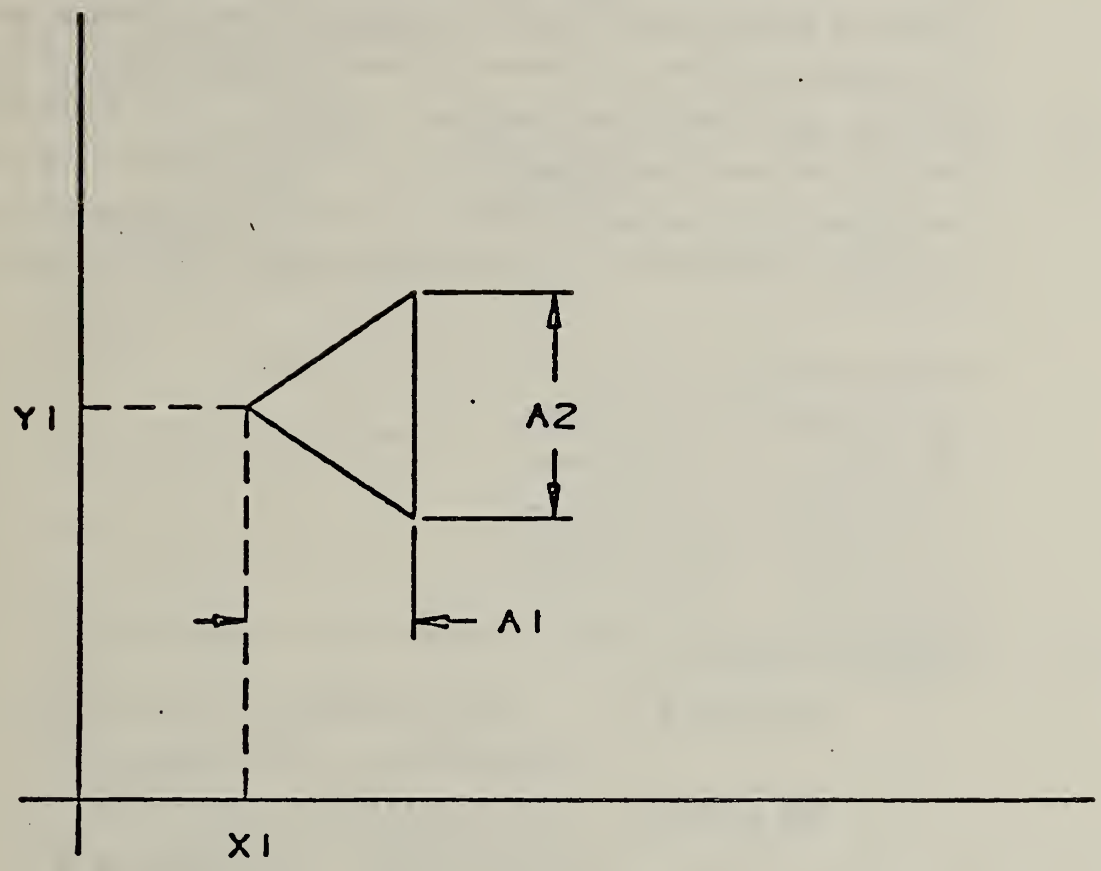


FIG. 4-29 EXAMPLE OF TRIANGLE MACRO

These examples are meant to illustrate some of the capabilities of a MACRO.

Example 1: Repeated parallelograms

The following MACRO takes the coordinates of three points and a repetition number as arguments and creates a pattern of repeated parallelograms. The three points represent the corners of a parallelogram. The parallelogram will be filled with similar parallelograms inside it. The argument NTANG controls how many parallelograms will be drawn inside; NTANG represents the number of parallelograms along any one side. For simplicity, the points have been constrained to all lie in a plane parallel to the X-Y plane.

```
MACRO, 600, X1, Y1, X2, Y2, X3, Y3, Z, NTANG;
LET YHDEL = (Y3-Y1)/NTANG;
LET YVDEL = (Y2-Y1)/NTANG;
LET XHDEL = (X3-X1)/NTANG;
LET XVDEL = (X2-X1)/NTANG;
LET K = 0;
REPEAT NTANG +1;
        SET #VLINE =    110, X1=K*XVDEL, Y1=K*YVDEL, Z,
                        X3=K*XVDEL, Y3=K*YVDEL, Z, 0, 0;
        SET #HLINE =    110, X1=K*XHDEL, Y1=K*YHDEL, Z,
                        X2=K*XHDEL, Y2=K*YHDEL, Z, 0, 0;
        LET K  =  K + 1;
CONTINUE;
ENDM;
```

An instance for this MACRO looks like this:

```
600, 2., 4., 6., 5., 1., 3;
```

Example 2:

The following MACRO takes a point, a radius, and a number and creates concentric circles out to the radius. A point is put into the center.

```
MACRO, 601, XC, YC, ZC, R, NCIRC;
LET DELTR = R/NCIRC;
REPEAT NCIRC;
          SET #CIR =          100, ZC, XC, YX, X, Y+R, X, Y+R, X, Y+R, 0, 0;
          LET R =             R - DELTR;
CONTINUE;
SET #PT = 116, XC, YC, ZC, 0, 0, 0;
ENDM;
```

An example of an instance:

```
601, 0., 0., 0., 20., 4;
```

This would create four concentric circles around the origin out to a radius of 20.

Example 3:

This MACRO takes a point and a base length and constructs a ground symbol (horizontally) at that point.

```
MACRO, 602, X, Y, Z, B;
LET DELY = B/6;
LET DELX = DELY;
SET #LINE1 =  110, X, Y, Z, X+B, Y, Z, 0, 0;
SET #LINE2 =  110, X+DELX, Y-DELY, Z, X+B-DELX, Y-DELY, Z, 0, 0;
SET #LINE3 =  110, X+2*DELX, Y-2*DELY, Z, X+B-2*DELX, Y-2*DELY, Z,
          0, 0;
ENDM;
```

Variables. Variable names may be from one to six characters in length. The first character must be one of the characters listed below. This character determines the variable type. It is not possible to override the conventions. The six character limitation includes the first character. Upper and lower case letters are recognized as distinct, i.e., X is different from x. Variable names longer than six characters may be used; however, only the first six characters will be significant. Variable names may contain imbedded blanks. These blanks are NOT taken as part of the name; therefore "A B" is equivalent to "AB." Except for the first character, as outlined below, all characters must be alphabetic (A-Z or a-z), or numeric (0-9).

Variable type	First character
Integer	I-N, i-n
Real	A-H, O-Z a-h, o-z
Double precision	!
String	$
Pointer	#

Examples of valid variable names are:

Integer:	IJK	ICOUNT	K101	NTIMES	max
Real:	XYZ	X1	y2	QrsTu1	
Double:	!h	!xi	!Y2	!12341	
String:	$str	$TITLE	$label		
Pointer:	#line	#note	#REF	#XYZ1	

Some invalid variable names:

$$$$	($ not permitted after first character)
1X43B	(1 may not be first character)
A.BC	(. is illegal)

Note that there are no "reserved" words. Thus a variable name such as "MACRO", which is identical to a statement keyword (described below), will not confuse the interpreter, although it may confuse the user of such a MACRO. It is suggested that these words be avoided.

4.3.6.5 MACRO Syntax

Constants. Constants may be integer, real or double precision. Integer constants are distinguished by the lack of a decimal point and exponent. Reals are distinguished by the presence of a decimal point or the presence of an E exponent. The decimal point is optional ONLY when the E exponent is present. Double precision constants are distinguished by the presence of a D exponent, which is mandatory; a decimal point is optional. Any constant may be preceded by a "+" or "-". A "+" is assumed if neither is present. Exponents may also contain a "+" or "-". Constants may not contain a comma but may contain imbedded blanks. Examples of valid constants follow:

Integer 3 4123 13152 +0 -0
(+0, 0 and -0 are equivalent)

Real: -1. 3.14159 6.62E-34 1E10 3.1E+3

Double: 1D0 0D0 3.1415926535897D3 -11562.18D-10

Examples of invalid constants include:

1,000	(commas not allowed)
E10	(need mantissa -- use 1E10 instead)
3.1-06	(E or D cannot be implied)

The limitations on magnitude and accuracy are inherently machine dependent and are specified in the global section of the file.

An instance:

```
602, 1., 6., 2., 1.3;
```

This last example demonstrates the use of various MACRO features. It is not meant as an example of a "useful" MACRO.

```
MACRO, 613, NROW, NCOL, VDIST, HDIST,!ANGLE;
LET /LABEL = 6HPOINTS;
LET !SIN =SDIN(!ANGLE); LET !COS = !COS = DCOS (!ANGLE);
LET YHD = HDIST * !SIN;
LET XHD =HDIST * !COS;
LET YVD =VDIST * !COS;
LET XVD = VDIST * (-!SIN);
LET IRC = 0; LET ICC = 0;
REPEAT NROW;
          LET XCOL = IRC * XVD;
          LET YCOL = IRC * YVD;
          REPEAT NCOL;
                 LET X = XCOL + ICC*XHD;
                 LET Y = YCOL + ICC*YHD;
                 SET #PT = 116, X, Y, 0., 0, !, #LINE, 0;
                 LET ICC = ICC + 1;
          CONTINUE;
          LET IRC = ICC + 1;
CONTINUE;
LET $NPTS = STRING(NROW*NCOL, I7);
LET /LABEL = $NPTS;
SET #LINE = 110, 0., 0., 0., 10., 0., 0., 1, #CIRC, 0;
SET #CIRC = 100, 0., 0., 0., 10., 0., 10., 0., 1, #LINE, 0;
MREF, 22, 601, 0., 0., 0., 10., 5;
ENDM;
```

An instance:

```
613, 4, 5, 0. 2, 0.1, 7.85398D-01;
```

Functions. Functions similar to FORTRAN library functions are provided. The rules for mixed mode, however, have been relaxed, so that it is not necessary to use, for example, SQRT(2.) instead of SQRT(2). While this assists the preprocessor writer in preparing MACROs, it places a responsibility on the writer of a processor for the MACRO language in handling the mixed mode. While the arguments can be mixed mode, note, however, that functions do have a specific type of value that they return, i.e., integer, real, or double precision. The functions are described here by the type of value returned. The type of argument usually used is also noted; however, this is primarily for aid in clearly documenting MACROs. For example, either IDINT(!d) or INT(!d) will work equally well, although the meaning might be a little clearer with IDINT(!d). Functions are only recognized in one case (UPPER).

Functions returning integer values:

IABS(i)
Returns the absolute value of i.
ISIGN(i)
Returns 1 if i is positive, 0 if it is zero, or -1 if it is negative.
IFIX(x) or INT(x)
Returns the integer part of x.
IDINT(!d)
Returns the integer part of !d.
Functions returning real values:
FLOAT(i)
Returns a real (floated) value for i, e.g., FLOAT(2) returns "2."
COS(x)
Returns cosine of angle x; angle in radians.
SIN(x)
Returns sine of angle x; angle in radians.
TAN(x)
Returns tangent of angle x; angle in radians.

ATAN(x)

Returns arctangent of x; angle returned in radians.

EXP(x)

Returns natural anti-logarithm of x ("e to the x").

ALOG(x)

Returns natural logarithm of x.

ALOG10(x)

Returns common (base 10) logarithm of x.

ABS(x)

Returns absolute value of x.

SQRT(x)

Returns square root of x.

AINT(x)

Returns integer part of x, just like INT, but returns value in floating-point form.

SIGN(x)

Returns 1 if x is greater than 0, 0 if x equals 0, and -1 if x is less than 0.

SNGL(!d)

Returns single (real) value of double precision variable !d. As many significant digits of !d as possible are given to the returned value.

Functions returning double precision values:

DBLE(x)

Returns "double precision"ed value of x. Note that this is merely a conversion, not an extension. Thus, DBL(.333333333) will return .333333333D0, but not .3333333333333333333333333D0. Thus, DBLE(1./3.) is not necessarily equal to 1D0/3D0.

DCOS(!d)

Returns cosine of angle !d; angle in radians.

DSIN(!d)

Returns sine of angle !d; angle in radians.

DTAN(!d)

Returns tangent of angle !d; angle in radians.

DATAN(!d)

Returns arctangent of !d; value returned in radians.

DEXP(!d)

Returns natural anti-logarithm of !d("e to the !d").

DLOG(!d)

Returns natural logarithm of !d.

DLOG10(!d)

Returns common (base 10) logarithm of !d.

DABS(!d)

Returns absolute value of !d.

DSQRT(!d)

Returns square root of !d.

DSIGN(!d)

Returns 1D0 if !d positive, 0D0 if zero, -1D0 if negative.

Expressions. Expressions may be formed using the above functions, variables and constants, and the following operators:

Function	Symbol
addition	+
subtraction	-
multiplication	*
division	/
exponentiation	**

The operators are evaluated in normal algebraic order, e.g., first exponentiation, then unary negation, then multiplication or division, then addition or subtraction. Within any one hierarchy, operators evaluate left to right. Parentheses may be used to override the normal evaluation order, as in the expression "A*(B+C)," which is different from "A*B+C." Extra parentheses do not alter the value of the expression; it is a good idea to use them, even if not truly necessary. Examples of expressions include:

```
X + 1.0
-B+SQRT(B**2. - 4*A*C)
I + 1
3.14159/2.
-X
!DEL*(!ALPHA-!BETA)
```

Except for the ** operator, it is never permissable to have two operators next to each other, i.e., not 2*-2, but -2*2 or 2*(-2). Multiplication may not be implied by parentheses, e.g., (A+B)(C+D) is illegal, and AB does not imply A*B, but rather the separate variable AB.

Mode of expression evaluation. Unlike FORTRAN, mixed mode (integer mixed with real, etc.) is permitted. Whenever two different types are to be operated upon, the calculation is performed in the "highest" type. Integer is the lowest type, Real is next, and Double precision is the highest. Note, however, that this decision is made for each operation, not once for the entire expression. Thus 1/3 + 1.0 evaluates to 1, because the "1/3" is done first, and it is done in integer mode. Integer mode truncates fractions, and does not round. Therefore, the expression "2/3+2/3+2/3" has a value of zero.

Statements. There are seven basic statements that can be used. They are:

LET
SET
REPEAT
CONTINUE
MACRO
ENDM
MREF

These "keywords" are recognized only in upper case, and every statement must begin with one of these keywords. Statements are free format; blanks and tabs are ignored except within strings. Statements may extend over several records, or more than one statement may be present on a card. All statements are terminated by a record delimiter which must be present.

LET statement (Arithmetic)

This is the basic assignment statement and is equivalent to the LET statement of BASIC. The format of a LET statement is:

```
LET variable = expression;
```

The expression and the variable may be integer, real, or double precision; they need not be of the same type. Note that this is an assignment statement and not an algebraic equality. All of the variables on the right hand side of the expression must have been previously defined; it cannot be assumed that variables will default to zero if they are undefined. Some examples of legal LET statements:

```
LET HYPOT = SQRT(A**2+B**2);
LET X = X + 1;
LET ROOT1 = -B + SQRT(B*B - 4*A*C);
LETI = i;
LET !XYZ = I * 2;
LET START = 0;
```

LET Statement (String)

String variables allow characters to be manipulated. String variables may be used in statements almost anywhere that any other variable type may be used; exceptions are noted below.

String variables may be used in LET statements. Note that they shall not be mixed with any other type of variable in a LET statement. Also note that operations (i.e., +,-, *, etc.) are not possible with string variables. Two forms of LET statement for string variables are possible:

LET $str = 23Hstring of 23 characters;

or

LET $stri = $str2;

In the first case, the 23 characters following the H are assigned to the string variable $str. In the second case, the string "$str2" is copied into "stri." Examples of these statements include:

LET $title = 3HBox;
LET $label = 6HBottom;
LET $x = $label;

Note that if a string variable appears on the right hand side of the statement, it must have been previously defined. Spaces are not ignored within a string constant; they become part of the string. Any ASCII character may be part of a string.

There is one other form for setting up a string. It involves the STRING function. The STRING function may only appear in this form. Specifically, it shall not appear in SET statement argument lists, MACRO statements, or MREF statements. However, string constants, such as "6Hstring," and variables, such as "$x" , may appear in SET statements and MACRO statements.

The form of a STRING function statement is:

LET $str = STRING(expression, format);

where "expression" is any normal integer, real or double precision expression, "$str" is a string variable name, and "format" is a format field similar to the format fields usable in a FORTRAN FORMAT statement. The allowable format fields are:

Iw
Fw.d
Ew.d
Dw.d

The effect of this statement is to convert the numeric value of the expression into characters, i.e., the statement:

```
LET $PI = STRING(3.14159,F7.5);
```
will result in the same thing as
```
LET $PI = 7H3.14159;
```

Of course, the usefulness of the STRING function is that expressions can be converted, rather than constants. Thus:

```
LET x = 1;
LET y = 2;
LET $xyz = STRING(x+y+1,F5.0);
```
will result in the same thing as
```
LET $xyz = 5H   4.;
```

The rules for the format fields follow the standard FORTRAN convention. "Iw" causes integer conversion, resulting in "w" characters. "Fw.d" causes real conversion, resulting in "w" characters, with "d" characters after the decimal point. "Ew.d" results in real conversion, but using an exponent form "Dw.d" is the same as "E" but for double values. Note that this is one place where mixed mode is not allowed. The type of format field and the type of the result of the expression must be identical.

SET statement

The SET statement establishes directory and parameter data entries for the specified entity. The form is:

```
SET #ptr = entity type number, argument list;
```

"#ptr" is a pointer variable, such as "#XYZ"; "entity type number" is an IGES entity type number, such as "110"; and "argument list" is a group of variables which are to be written in the parameter data section of the entity.

Examples of this type of SET statement:

```
SET #LINE = 110,X1,Y1,Z,X2,Y2,Z,0,0;
SET #ABC = 228,Z,A+B/C,Y1,X2,Y2+1,0,0;
SET #qwe = 264,15Hstrings allowed, X,Y,$this2;
```

The argument list may contain expressions and may spread over more than one record. At least one argument must be present, i.e., the argument list may not be null. The entity type number may not be an expression. It must be an integer constant. The pointer variable will be assigned a value corresponding to the sequence number of the directory entry of the entity created.

The format of the argument list written out in the parameter data section depends on the type of argument in the list. Integer arguments will be written in integer format, reals as reals, and doubles as doubles. Thus, functions such as FLOAT, INT, DBLE, etc., might be of use in order to force an expression that, for example, would normally evaluate to an integer value to be written as a real number.

"Forward referencing" of pointers is legal in the argument list of a SET statement. A pointer may appear in the argument list of a SET statement that comes before the SET statement defining the pointer. The only restriction is that any pointer so referenced must appear on the left hand side of one SET statement.

Pointers which appear on the left hand side of more than one SET statement or those which are located inside of REPEAT loops, should not be forward referenced.

Note that the STRING function is not allowable in a SET statement -- use a separate LET statement with a string variable instead.

REPEAT statement

The REPEAT statement causes a group of statements terminated by a CONTINUE statement to be repeated a specified number of times. The form of a REPEAT statement is:

```
REPEAT expression;
```

The expression is evaluated, and the resulting value is the number of times the statements will be repeated. The expression may be of integer, real or double type; in the case of real or double expressions, the result is truncated to determine the repeat count. If the repeat count is zero or negative, the group of statements is still executed one time.

Examples of REPEAT statements:

```
REPEAT 3;
REPEAT N+1;
REPEAT 0;             (will still go through one time)
REPEAT X+Y;
```

There is a limit of ten REPEAT statements that may be nested inside one another.

After a REPEAT statement such as REPEAT N it is legal to alter the value of N. This does not affect the repeat count. Also note that REPEAT is unlike a FORTRAN "DO" because there is no variable being incremented on every pass.

CONTINUE statement

The CONTINUE statement marks the end of a REPEAT group. The form of a CONTINUE statement is:

```
CONTINUE;
```

When a CONTINUE statement is encountered, the repeat count is decremented by one and checked to see if it is greater than zero. If it is, the interpreter goes back to the first statement after the most recent REPEAT.

If not, then the next statement is processed. The number of REPEAT's and CONTINUE's in a MACRO should be the same. CONTINUE is not implied by ENDM.

MACRO statement

The MACRO statement is used to signify the start of a MACRO definition. The format of a MACRO statement is:

```
MACRO, entity type number, argument list;
```

where "entity type number" is the entity type number of the MACRO, and "argument list" is a list of parameters that are to be assigned values at execution time. The argument list may not be null. The first statement in every MACRO definition must be a MACRO statement. Note that the argument list may not contain expressions, only symbolic variable names of type integer, real, double precision, string, or pointer.

The MACRO statement must be the first statement of the MACRO and there may be no other MACRO statements inside a MACRO. Use the MREF statement to reference other MACROs but defining a MACRO inside of another MACRO is meaningless.

Examples of a MACRO statement:

```
MACRO,610,X1,Y1,X2,Y2,$ABC,#PR;
MACRO,600,A,B,C;
MACRO,600,IX,IY,I;
```

ENDM statement

ENDM signifies the end of a MACRO. The form of an ENDM statement is:

```
ENDM;
```

All MACROs must have an ENDM statement as their last statement. ENDM is not implied by the end of the parameter data section.

MREF statement

The MREF statement is used to reference another MACRO from inside a MACRO definition. The format of a MREF statement is:

MREF, ptr, entity type number, argument list;

where "ptr" may be either a pointer variable or an integer expression. The value refers to the directory entry block of the definition of the MACRO being referenced. "Entity type number" is the entity type number of the MACRO being referenced. "Argument list" is an argument list exactly like that of a SET statement. The effect of the argument list is to replace the symbolic names found in the MACRO definition with the values of the expressions contained in the MREF statement, so that execution of the referenced MACRO will start with the appropriate values. Note that MREF does not start expansion of the referenced MACRO. Rather it creates an entity entry which may later be expanded. It is thus not possible for a MACRO being referenced to have access to any of those values except for those in the argument list. Even then, it is not possible for the occurrence of a MREF statement to alter any of those values.

Examples of MREF statements:

```
MREF,#mac1,600,X1,Y1,Z1,X2,Y2,3.1;
MREF,33,621,A,B,3-X/W-1.6*W,3.,0,6Hstring,$x;
```

When a MREF statement is encountered during a MACRO expansion, pertinent information will be printed out to enable the referenced MACRO to be expanded in another pass. Note that it is not strictly necessary for the values in a MREF statement to be of the same type as the values in the definition MACRO, within certain limitations. Integer, real, and double values may be freely mixed, although it might be considered a good idea not to do so. String values may only appear where string variables appear in the definition.

Attributes. Attributes may be set using the LET statement. The format for doing so is:

```
LET /attribute name = expression;
or
LET /attribute name = /HDR;
```

The first form allows an attribute to be set to any constant value, including numeric expressions. Note, however, that the use of expressions is discouraged. Attributes may also be set to string constants or string variables, but not to the result of a STRING function.

Examples:

```
LET /LEV = 1;
LET /VIEW = 3;
LET /LABEL = 6HBottom;
LET /LABEL = $X;
```

The second form allows restoring an attribute to its default value.

Examples:

```
LET /LEV = /HDR;
LET /LABEL = /HDR;
```

The word "/HDR" is the only non-constant that is allowed on the right side of an attribute assignment statement. The effect is to restore the value of the attribute to what it was in the directory entry for the instance or, in some cases, to a specified default value. The defaults are described below.

Attributes may not be mixed with any other variable type nor may they appear anywhere but in the above two forms of LET statements.

The allowable attribute names and their defaults are given here. A default of /HDR indicates that the attribute defaults to the value in the directory entry of the instance.

/LFP	/HDR
/LEV	/HDR
/VIEW	/HDR
/MTX	/HDR
/CE	0
/BS	/HDR
/SE	subordinate
/ET	/HDR
/LW	/HDR
/PN	/HDR
/FORM	0
/LABEL	(blank)
/SUB	0

4.3.7 Property Entity. The property entity contains numerical or textual data. It also has a form number to indicate its meaning. Certain generic Property form numbers are described in the following sections and are expected to be augmented by others in future versions of this Specification. Form numbers in the range 5001 - 9999 are left undefined for users.

Note that properties can also point to other properties, as well as participate in associativities or have attached text in the form of a general note.

4.3.7.1 Directory Data

ENTITY TYPE NUMBER : 406

4.3.7.2 Parameter Data

Parameter	Value	Format	Comment
1	N	Integer	Number of properties
2		Variable	Property values
. . .			
1+N			
2+N	NA	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
3+N	DE	Pointer	Pointers to associativities or general notes
. . .			
2+N+NA	DE	Pointer	
3+N+NA	M	Integer	Number of properties
4+N+NA	DE	Pointer	Pointers to properties
. . .			
3+N+NA+M	DE	Pointer	

4.3.7.3 Defined Properties

4.3.7.3.1 FORM NUMBER: 1 Definition levels

Parameter	Value	Format	Comment
1	N1	Integer	Number of multiple levels
2	L	Integer	Level number
.	.	.	
.	.	.	
.	.	.	
1+N1	.	.	

For each entity in the file that is defined on multiple levels, there will be an occurrence of the property instance (Form 1). In the parameter data portion of the property instance, the parameter N1 will contain the number of multiple levels followed by a list of those levels. Each entity that is defined on this set of levels will contain a pointer (in the level field of the directory entry) to this property instance. A different set of multiple levels will result in a different property instance.

4.3.7.3.2 FORM NUMBER: 2 Region Restriction

DESCRIPTION

This property allows entities that can define regions to set an applications restriction over that region. The restrictions will indicate whether a given applications item must lie completely within regions with this property or completely outside such regions. The restriction applies to all points of entities used to represent the applications item and to all points within the effect of the item when all properties, such as line widening, are applied.

Each of the parameters in this property will have one of the following three values indicating the region restriction relevant to the applications item:

K = 0	No restriction
K = 1	Item must be inside region
K = 2	Item must be outside region

Parameter	Value	Format	Comment
1	3	Integer	Three parameters
2	K=0, 1 or 2	Integer	Electrical vias
3	K=0, 1 or 2	Integer	Electrical components
4	K=0, 1 or 2	Integer	Electrical circuitry

4.3.7.3.3 FORM NUMBER: 3 Level Function

DESCRIPTION

This property provides a code which identifies the applications function of the level. It also provides a value that can preserve a level-like value relevant to the source system on a system to system transfer. Entities with the same level value as the level value of this property are all associated with the applications function the property describes. Transfer of applications data between systems that use a level-like parameter to classify functions will be quite common. This property will serve to guide that transfer. The property can be applied to multiple levels thru use of an associativity of levels as indicated in the DE definition and IGES-defined associativity number 2, Definition Levels.

Parameter	Value	Format	Comment
1	2	Integer	Two parameters
2	FC	Integer	Function description code
3	SL	Integer	Source level

4.3.7.3.4 FORM NUMBER: 4 Region Fill Property

DESCRIPTION

This property helps define the functional value of any closed region. It classifies the region as to its "filled" status. It will be used most often to identify which region-defining entities are defining a functional region (or a gap in that region) and which have other purposes. The actual function of the region will likely be determined in conjunction with level or subfigure membership.

Parameter	Value	Format	Comment
1	2	Integer	Two parameters
2		Integer	Fill code: 0 solid fill 1 unfill (i.e., a gap in solid fill) 2 meshed fill
3	DES	Integer	DE of a section entity defining linear segments of meshed fill (Note: Usage of the Line Widening property with the section entity determines the actual width of the mesh segments.)

4.3.7.3.5 FORM NUMBER: 5 Line Widening

DESCRIPTION

This property defines the characteristics of entities when they are used to define the location of items such as strips of metalization on printed wiring boards.

The justification flag terminology is interpreted as follows: right justified means that a defining line segment forms the right edge of the widened line in the direction from first defining point to second. Left justified is the opposite while center justified indicates that the defining line segment splits the widening exactly in half.

Figure 4-30 indicates the measurement of the property values.

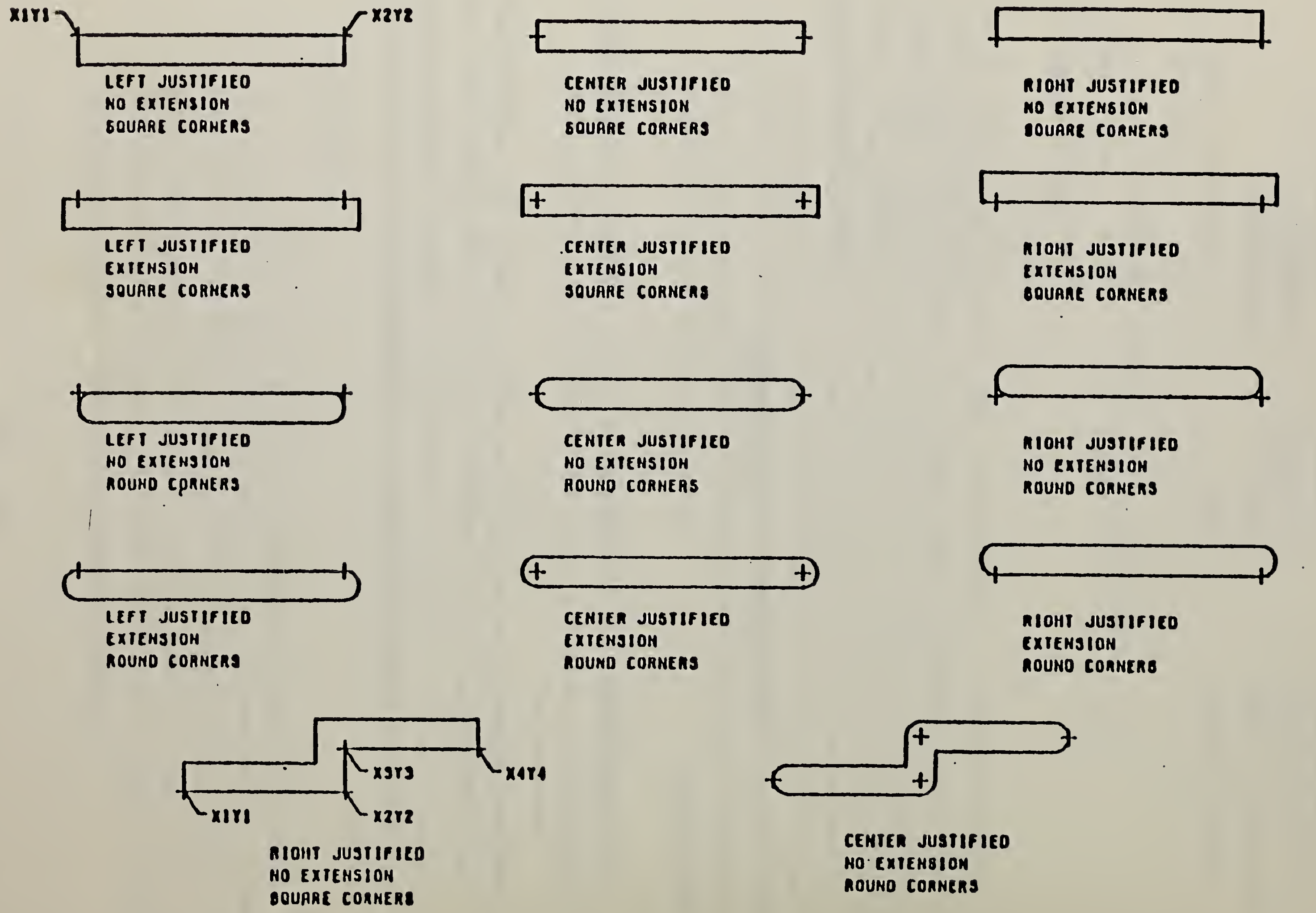


FIG. 4-30 LINE WIDENING EXAMPLES

Parameters	Value	Format	Comment
1	5	Integer	Five data items
2		Floating Point	Width of metalization
3		Integer	Cornering code: 0 rounded 1 squared
4		Integer	Extension flag with values: 0 No extension 1 One-half width extension 2 Extension set by parameter
5		Integer	Justification flag with values: 0 center justified 1 left justified 2 right justified
6		Floating Point	Extension value if parameter 4 = 2 (Note: This value may be negative)

4.3.7.3.6 FORM NUMBER: 6 Drilled Hole

DESCRIPTION

The Drilled Hole property identifies an entity representing a drilled hole through a printed circuit board. The parameters of the property define the characteristics of the hole necessary for actual machining. The layer range indicated by parameters 5 and 6 refers to physical layers of the assembled printed circuit board.

Parameters	Value	Format	Comment
1	5	Integer	Five parameters
2		Floating Point	Drill diameter size
3		Floating Point	Finish diameter size
4		Integer	Plating indication (0 =no, 1 = yes)
5		Integer	Lower numbered layer
6		Integer	Higher numbered layer

4.3.7.3.7 FORM NUMBER: 7 Reference Designator

DESCRIPTION

The Reference Designator property attaches a text string containing the value of a component reference designator to an entity being used to represent an electrical component.

Parameter	Value	Format	Comment
1	1	Integer	One parameter
2		String	Reference designator text

4.3.7.3.8 FORM NUMBER: 8 Pin Number

DESCRIPTION

The Pin Number property attaches a text string representing a component pin number to an entity being used to represent an electrical component's pin.

Parameter	Value	Format	Comment
1	1	Integer	One parameter
2		String	Pin Number Value

4.3.7.3.9 FORM NUMBER: 9 Part Number

DESCRIPTION

The Part Number property attaches a set of text strings that define the common electrical part numbers to an entity being used to represent an electrical component. Null text values in any parameter will imply that the missing value is not relevant to the transferred data.

Parameter	Value	Format	Comment
1	4	Integer	Three parameters
2		String	Generic part number or name
3		String	MIL-STD part number
4		String	Vendor part number or name
5		String	Internal part number

4.3.7.3.10 FORM NUMBER: 10 Hierarchy

DESCRIPTION

The hierarchy property provides the ability to control the hierarchy of each directory entry attribute. This property is referenced when the directory entry status digits 7 and 8 are 02.

Parameter	Value	Format	Set Value	Comment
1	NP	Integer	7	Seven parameters
2	LF	Integer		Line font
3	VU	Integer		View
4	LAB	Integer		Entity level
5	BL	Integer		Blank status
6	LW	Integer		Line weight
7	PEN	Integer		Pen number

8	N	Integer	Number of back pointers/ text pointers
NP+N+1	DE	Pointer	Pointer to associativities or general notes
NP+N+2	DE	Pointer	
.		.	
NP+M+3	M	Integer	Number of properties
NP+N+M+1	DE	Pointer	

Acceptable values for parameters 2 through 7 are 0 and 1.

0 = use directory entry attribute value

1 = use the directory entry attribute of the subordinate entity

4.3.8 Subfigure Definition Entity. The subfigure definition entity is designed to support the concept of a subpicture (if one equates drawing creation with graphics picture processing). This entity permits a single definition of a detail to be utilized in multiple instances in the creation of the whole picture. The contents of the subfigure include a set of pointers to any combination of entities and other subfigures. DEPTH indicates the actual nesting of the subfigures. If DEPTH=0, the subfigure has no references to any subfigure instances. A subfigure cannot reference a subfigure instance that has equal or greater depth. A DEPTH=N indicates there is a reference to a subfigure instance with DEPTH N-1.

4.3.8.1 Directory Data

ENTITY TYPE NUMBER : 308

4.3.8.2 Parameter Data

Parameter	Value	Format	Comment
1	DEPTH	Integer	Depth of subfigure (indicating the amount of nesting)
2	NAME	String	Subfigure name
3	N	Integer	Number of entities in the subfigure
4	DE	Pointer	Pointers to the directory entries for the associated entities
⋮			
N+3		Pointer	
N+4	NA	Integer	Number of back pointers (to associativities)/text pointers (to general notes)
N+5	DE	Pointer	Pointers to associativities or general notes
	⋮	⋮	
NA+N+4	DE	Pointer	
NA+N+5	M	Integer	Number of properties
NA+N+6	DE	Pointer	Pointers to properties
	⋮	⋮	
M+N+NA+5	DE	Pointer	

4.3.9 Subfigure Instance Entities. Each occurrence of a subfigure is defined by a subfigure instance entity. This may exist as a single instance or as a two dimensional array of the same subfigure.

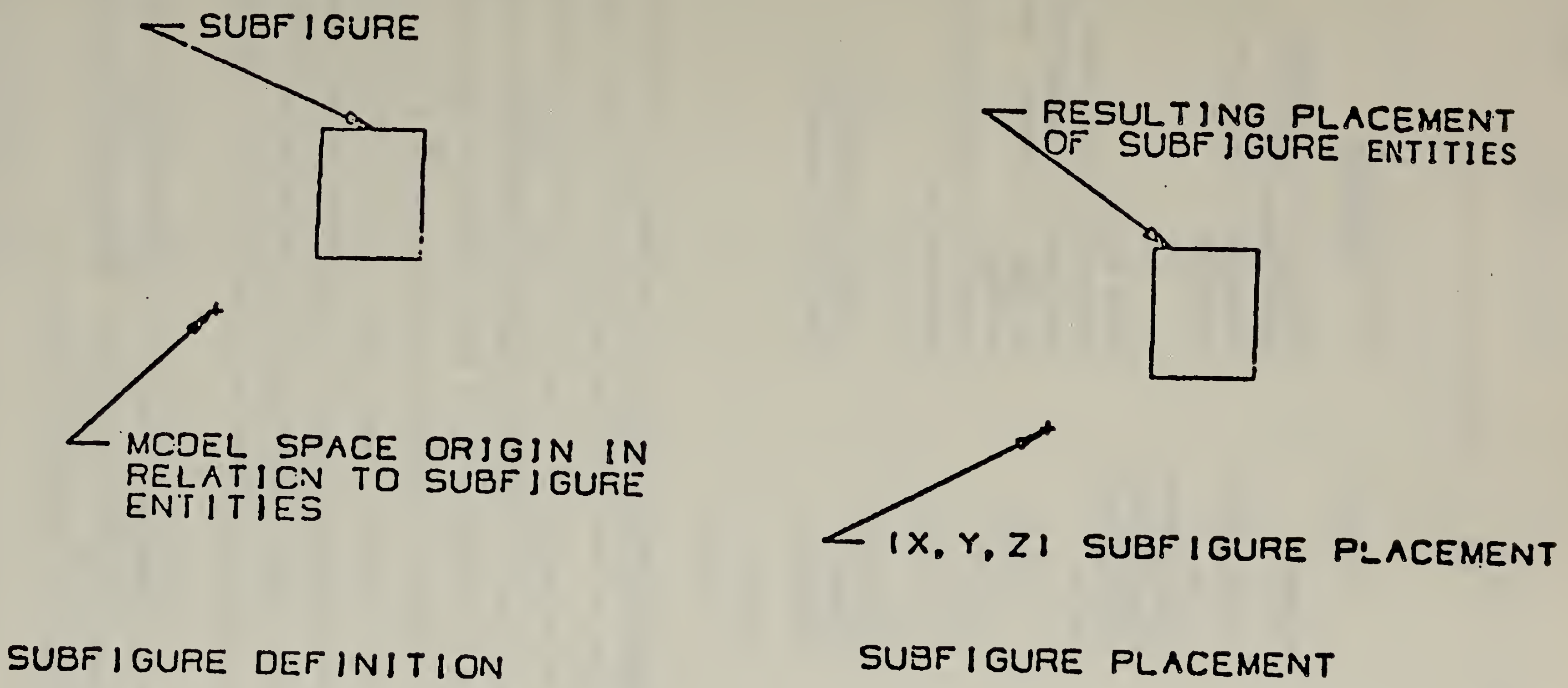
Before placement by the subfigure instance, each entity is operated upon by any defining matrix which may be associated with the individual entity. All the entities are then scaled about the origin of the defined subfigure by multiplying their model space coordinates by the scale factor in parameter 5. If a matrix reference is specified by the subfigure instance, it is then applied to the entities in the subfigure. The model space placement of the subfigure instance is then used to translate the subfigure into the model space of the file. See Figure 4-31 for an example of the placement of a subfigure.

In some applications of interactive graphics, entities exist which act as both subordinate entities in the sense that they are part of a subfigure definition and as independent entities in the sense that each subfigure instance results in an entity, some properties of which differ from those of the same entity in other instances. These entities are referred to as multiply-instanced. Examples are Text Node and Connect Node.

The entity use flag (digits 5-6 of field 9 in the directory entry) will be set to a value of 04 to designate such an entity. This value implies that the entity is being used as a logical construct rather than an actual entity.

Each multiple-instancing entity is defined as an associativity. Class 1 of the associativity contains pointers to any geometry associated with the entity. The first pointer in class 1 always points to the original geometry. Additional pointers point to additional instances of the geometry.

4.3.9.1 Singular Subfigure Instance Entity. This entity defines the occurrence of a single instance of the defined subfigure.


SUBFIGURE
MCDEL SPACE ORIGIN IN RELATICN TO SUBFIGURE ENTITIES
RESULTING PLACEMENT OF SUBFIGURE ENTITIES
(X, Y, Z) SUBFIGURE PLACEMENT

SUBFIGURE DEFINITION

SUBFIGURE PLACEMENT

FIG. 4-31 SUBFIGURE ORIGIN

4.3.9.1.1 Directory Data

ENTITY TYPE NUMBER : 408

4.3.9.1.2 Parameter Data

Parameter	Value	Format	Comment
1	DE	Pointer	Pointer to subfigure definition entry
2	X	Floating Point	Model space placement of subfigure
3	Y	Floating Point	
4	Z	Floating Point	
5	S	Floating Point	Scale factor
6	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
7	DE	Pointer	Pointers to associativities or general notes
.	.	.	
6+N	DE	Pointer	
7+N	M	Integer	Number of properties
8+N	DE	Pointer	Pointers to properties
.	.	.	
7+N+M	DE	Pointer	

4.3.9.2 Rectangular Array Subfigure Instance Entity. The rectangular array produces copies of an object called the base entity, arranging them in equally spaced rows and columns. The following type of base entity can be selected: group, subfigure instance, point, line, circular arc, conic arc, rectangular or circular array. The number of columns and rows of the rectangular array, together with their respective horizontal and vertical displacements are given. Also, the coordinates of the lower left hand corner for the entire array is indicated. This is where the first entity in the reproduction process is placed and is called position No. 1.The successive positions are counted vertically up the first column, then vertically up the second column to the right, and so on.

The entire array can be tilted with respect to the screen by an angle of rotation of the horizontal rows about the origin. This angle is measured in degrees counterclockwise from the positive X-axis.

A DO-DON'T flag enables one to display only a portion of the array. If the DO flag is chosen, half or fewer of the elements of the rectangular array are to be defined. If the DON'T flag is chosen, half or more of the elements of the rectangular array are to be defined.

4.3.9.2.1 Directory Data

ENTITY TYPE NUMBER: 412

4.3.9.2.2 Parameter Data

Parameter	Value	Format	Comment
1	DE	Pointer	Pointer to base entity
2	SC	Floating	Scale factor
3	X	Floating	Coordinates of point
4	Y	Floating	to be used as lower
5	Z	Floating	left hand corner of array
6	NC	Integer	Number of columns
7	NR	Integer	Number of rows
8	DX	Floating	Horizontal distance between columns
9	DY	Floating	Vertical distance between columns
10	AX	Floating	Rotation angle in degrees
11	LC	Integer	DO-DON'T list count =L. (L=0 indicates all to be displayed
12	IF	Integer	DO-DON'T flag (DO=0; DON'T=1
13	N1	Integer	Position number of entity to be processed (DO), or not to be processed (DON'T)
10+LC	NK	Integer	
K(number of entries) = LC+10			
K+1	N	Integer	Number of backpointers (to associativity entities)/text pointers (to general note entities)
K+2	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
K+N+1	DE	Pointer	
K+N+2	M	Integer	Number of properties
K+N+3	DE	Pointer	Pointer to property
.	.	.	.
.	.	.	.
.	.	.	.
K+N+(M)+2	DE	Pointer	

4.3.9.3 Circular Array Subfigure Instance Entity

The circular array produces copies of an object called the base entity, arranging them around the edge of an imaginary circle whose center and radius are specified. The following type of base entity can be selected: group, point, line circular arc, conic arc, rectangular or circular array. The number of times that the base entity is replicated is given, together with the angle the first replicated entity makes with the positive X-axis running through the center of the imaginary circle. This angle is called the start angle, and the location of this replicated entity is called position No. 1. The successive positions follow a counterclockwise direction around the imaginary circle and are distributed according to a given delta angle.

A DO-DON'T flag enables one to display only a portion of the array. If the DO-flag is chosen, half or fewer of the elements of the circular array are to be defined. If the DON'T-flag is chosen, half or more of the elements of the circular array are to be defined.

4.3.9.3.1 Directory Data

ENTITY TYPE NUMBER: 414

4.3.9.3.2 Parameter Data

Parameter	Value	Format	Comment
1	DE	Pointer	Pointer to base entity
2	NE	Integer	Total number of replicated entities
3	X	Floating Point	Coordinates of center
4	Y	Floating Point	of imaginary circle
5	X	Floating Point	
6	R	Floating Point	Radius of imaginary circle
7	AS	Floating Point	Start angle in degrees
8	AD	Floating Point	Delta angle in degrees
9	LC	Integer	DO-DON'T list count = L. (L=0 indicates all replicated entities to be displayed)
10	IF	Integer	DO-DON'T Flag (DO=0; DON'T=1)

11	N1		Position number of entity to be processed (DO), or not to be processed (DON'T)
.	.		
.	.		
.	.		
10+LC	NK		

K(number of entries) = LC+10

K+1	N	Integer	Number of backpointers (to associativity entities)/text pointers (to general notes)
K+2	DE	Pointer	Pointer to either an associativity or to a general note
K+N+1	DE	Pointer	
K-N+2	M	Integer	Number of properties
K+N+3	DE	Pointer	Pointer to property
.	.	.	
.	.	.	
.	.	.	
K+N+M+2	DE	Pointer	

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. When this entity is a modification to another font, the Supercedes Font value (Field 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., Field 3 references another font, the definitions in this entity supercede 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. The pen is assumed to be down at the start of the stroking. 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 Entry

ENTITY TYPE NUMBER : 310

4.3.10.2 Parameter Data

Parameter	Value	Format	Comment
1	FC	Integer	Font Number
2	FNAME	String	Font Name
3	SF	Integer	Number of the font which this definition supercedes
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
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	
.	.	.	
.	.	.	
.	.	.	

Parameter	Value	Format	Comment
9+NM1*3	AC2	Integer	ASCII code for second character
10+NM1*3	NM2	Integer	Number of pen motions for second character
•	•	•	
•	•	•	
•	•	•	
$6+4N+\sum_{i=1}^{N} 3*NMi$	NA	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
$7+4N+\sum_{i=1}^{N} 3*NMi$	DE	Pointer	Pointers to associativities or general notes
•	•	•	
•	•	•	
•	•	•	
$NA+7+4N+\sum_{i=1}^{N} 3*NMi$	M	Integer	Number of properties
$NA+8+4N+\sum_{i=1}^{N} 3*NMi$	DE	Pointer	Pointers to properties
•	•	•	
•	•	•	
•	•	•	
$M+NA+*+4N+\sum_{i=1}^{N} 3*NMi$	DE	Pointer	

An example of this entity using the character in Figure 4-32 is

FC	1
FNAME	8H STANDARD
SF	
SCALE	8
N	60
AC1	65
NX1	11
NY1	0
NM1	4
PF1	0
X1	8
Y1	4
PF2	0
X2	8
Y2	0
PF3	1
X3	2
Y3	2
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,,8,4,,8,0,1,2,4,,6,4....
```

Figure 4-33 provides another example.

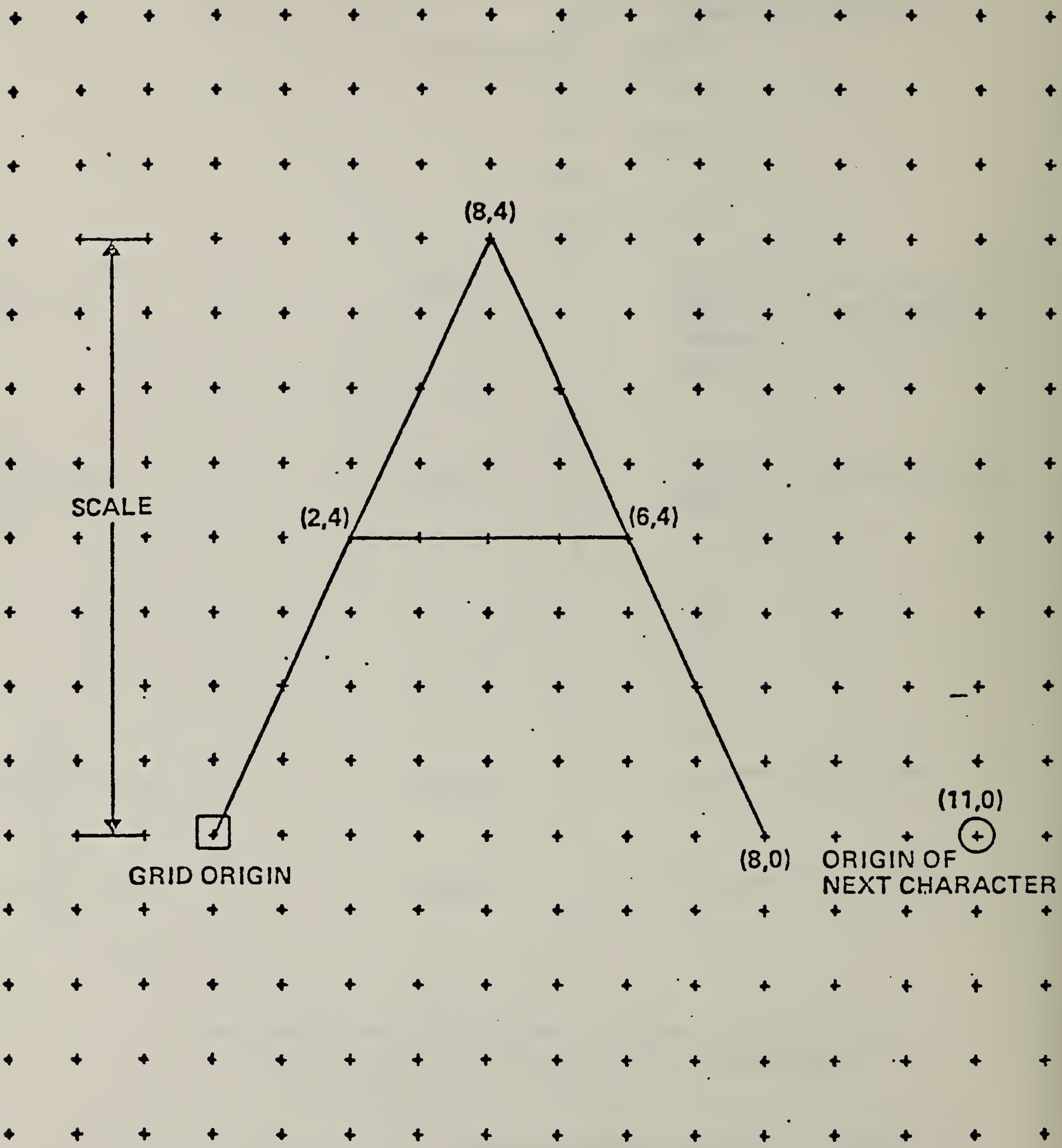


FIG. 4-32 EXAMPLE OF A CHARACTER DEFINITION

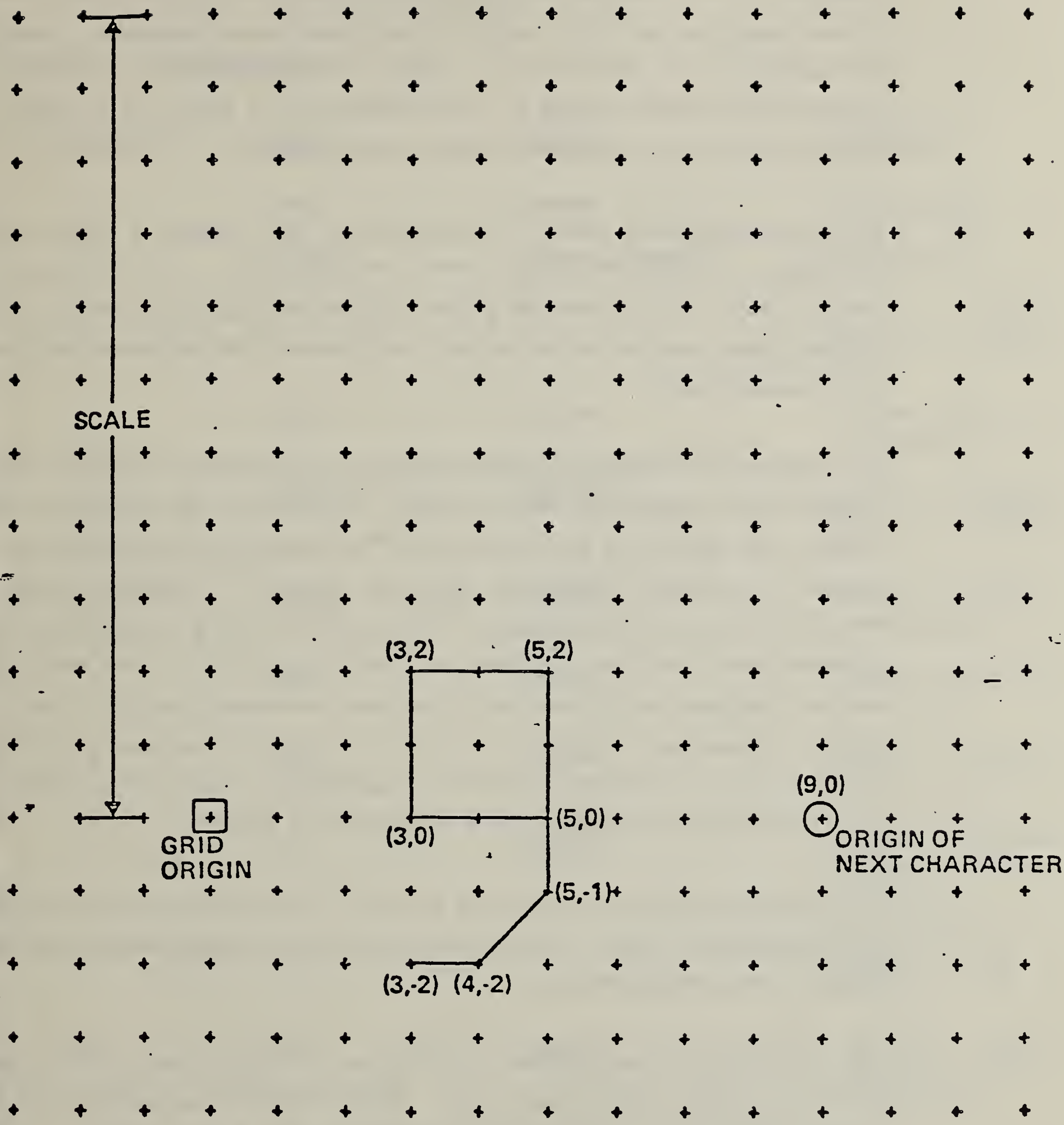


FIG. 4-33 SECOND CHARACTER DEFINITION EXAMPLE

4.3.11 View Entity. The view entity defines a specific "look" of the model. Since the "look" is taken to be axonometric, only a viewing direction need be specified. This is specified by means of a defining matrix of the form found in the transformation matrix entity. The default viewing direction is along the positive Z axis toward the X, Y plane in model space. In its simplest form the view entity consists of a view number and a pointer to a defining matrix (in field seven of the view entity directory entry.)

In more complicated cases, the view entity also defines a scale and optionally, a clipping box which control the projection of the view onto a drawing plane. Following clipping and rotation, the scale parameter multiplies all model coordinates before they are translated and projected onto the X, Y drawing plane.

If a clipping box is specified within model space, each entity is clipped at the surface of the clipping box before scaling. This allows only portions of the model to be shown in a particular view. The viewing box is specified by 4 pointers to entities defining the sides and possibly 2 additional entities defining the front and back planes. (The entities defining the sides will be plane entities .) The front and back are plane entities. If the front and back are not specified, the clipping box is assumed to extend from plus to minus infinity. Depending on the form of the view entity, the parameter list can have 1, 2, 6, or 8 members, exclusive of associativity and property pointers. Parameters two through eight that do not apply will be zero.

The clipping planes, when they are provided, must be consistent with the viewing definition matrix. That is, the front and back clipping planes must be normal to the viewing direction.

The view entity makes possible the selection of display characteristics for each entity (font, weight, pen, etc.). These attributes are specified in a views visible (form 4) associativity associated with a given entity. If a display attribute is not specified, its default is 1. If no views visible associativity exists for an entity, the entity is displayed with default attributes in all views.

4.3.11.1 Directory Data

ENTITY TYPE NUMBER: 410

4.3.11.2 Parameter Data

Parameter	Value	Format	Comments
1	VNO	Integer	View number
2	SCALE	Floating Point	Scale factor
3	FAC1	Pointer	Pointer to left side of viewing box
4	FAC2	Pointer	Pointer to top of viewing box
5	FAC3	Pointer	Pointer to right side of viewing box
6	FAC4	Pointer	Pointer to bottom of viewing box
7	BAK	Pointer	Pointer to back of viewing box
8	FRO	Pointer	Pointer to front of viewing box
9	N	Integer	Number of back pointers (to associativity entities)/text pointers (to general note entities)
10	DE	Pointer	Pointers to associativities or general notes
.	.	.	
.	.	.	
.	.	.	
9+N	DE	Pointer	

Parameter	Value	Format	Comments
10+N	M	Integer	Number of properties
11+N	DE	Pointer	Pointers to properties
.	.	.	
.	.	.	
.	.	.	
10+N+M	DE	Pointer	

APPENDIX A
SPLINE REPRESENTATIONS

A1 INTRODUCTION

Section 3 of the Specification includes four different types of spline representations:

a. A Parametric Piecewise Cubic Polynomial (for curves)

b. A Rational B-Spline Curve

c. A Grid of General Bicubic Patches (for surfaces).

d. A Rational B-Spline Surface

Most of the spline types used in CAD/CAM systems can be mapped into these representations without change in shape. Spline types supported in Section 3 of this Specification include parametric cubics, piecewise linear, Wilson-Fowler, modified Wilson-Fowler, rational and non-rational B-splines, rational and non-rational cartesian product B-spline surfaces, and Coons' patches. Spline types not supported include splines under tension and extended Coons' patches.

A2 SPLINE FUNCTIONS

In Section 3.8 of this Specification, spline curves are represented by a number of cubic spline functions, one for each of the X,Y,Z coordinates. Each cubic spline function S(u) is defined by

a. N: The number of segments,

b. t(1),...,t(N+1): The endpoints and the breakpoints separating the cubic polynomial segments,

c. a(i),b(i),c(i),d(i), i=1,...,N+1: The coefficients of the polynomials representing the spline in each of the N segments (the N+1st segment is not required to define the spline, but is included to make the endpoint value and derivative available without evaluating the polynomial),

d. CTYPE: The spline type. (1=linear, 2=quadratic, 3=cubic, 4=Wilson-Fowler, 5=Modified Wilson-Fowler, 6 = B-spline).

e. H: Degree of continuity.

To evaluate the spline at a point "u", first determine the segment containing "u", i.e., the segment "i" such that t(i) u t(i+1), then evaluate the cubic polynomial in that segment, i.e., compute

S(u) = A(i) + B(i)*(uu) + C(i)*(uu)**2 + D(i)*(uu)**3

where uu = u-t(i).

The polynomial is written in terms of the relative displacement uu (rather than u) so that the values of the spline at the breakpoints can be read directly out of the representation (i.e., S(t(i)) = A(i), i=1, ..., N+1). Computations using the relative displacement also have less floating-point roundoff error.

This particular "piecewise polynomial" form is only one of many used to represent the spline segments in CAD/CAM systems. Other representations employed include:

a. End points E1,E2 and end slopes S1,S2: The spline can be evaluated using the "Hermite" basis (see Ref. 2, page 59).

b. Values at four points: The spline value can be computed from the Lagrange or Newton interpolation formulae (see Ref. 2).

c. End points and "control" points: There are a number of schemes for computing splines from control points which will not be described here.

Reference 2 gives techniques for conversion between these representations.

Splines can also be represented as a linear combination of the B-spline basis functions. In CAD/CAM systems, B-splines have been used directly in curve fitting (e.g., the B-spline Bezier polygon (Ref. 4) and indirectly in various spline calculations (e.g., computing a cubic spline interpolate). For every set of breakpoints t(1),...,t(N+1) and degree of continuity H, a set of B-spline functions B(1,u),B(2,u),...,B(n',u) can be constructed (see Ref. 2). Then, for any piecewise polynomial S(u) with these breakpoints and continuity there is a set of B-spline coefficients a(1),..., a(n') such that S(u) can be represented as a linear combination of these B-splines

S(u) = a(1)*B(1,u) + a(2)*B(2,u) + ... + a(n')*B(n',u)

where n' = (N-1)*(3-H)+4.

B-splines can be computed from piecewise polynomials and vice versa (see p.116 of Ref. 2 and subroutine BSPLPP in Ref. 2).

Several other types of spline representations (e.g., cardinal bases) have been employed, but they are much less common and do not appear to present a problem for this Standard.

A3 SPLINE CURVES

Since curves in CAD/CAM problems are frequently many-valued, spline functions cannot represent such curves adequately. The most common approach to curve fitting is to parameterize the curves, i.e., to represent each curve as either two or three spline functions (one for each coordinate)

X(u) = Sx(u),
Y(u) = Sy(u), and
Z(u) = Sz(u)

which sketch out the curve as the parameter u varies from t(1) to t(N+1). All of the spline function representations of the previous section can be generalized to parametric curves and the algorithms for converting spline curves from one representation to the other follow easily from multiple applications of the corresponding function conversion algorithms.

Wilson-Fowler Curves: In the early sixties, the Wilson-Fowler spline (a special case of parametric cubics) was developed for curve fitting (see Ref. 1)). It is still used in many turnkey drafting systems. In the Wilson-Fowler representation, each spline segment is defined in a separate coordinate system whose X-axis begins at one endpoint of the segment and passes through the other. Each spline segment is then defined by a cubic spline function Swf(x) and the coordinates of the two endpoints. These Wilson-Fowler splines can be converted to splines defined in Section 3.8 by rotating the parametric spline (u,Swf(u)) back into the current coordinate system; however, most types of splines defined in Section 3.8 cannot be converted to Wilson-Fowler splines.

A4 RATIONAL B-SPLINE CURVES

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 <u>B-spline basis functions.</u> These are defined as soon as we specify their <u>degree</u>, M, and underlying <u>knot sequence</u>, T.

We do this 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 parameterized 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 $\left[T(i-M), T(i+1)\right]$. 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 $N(t|t_{i-m},...,t_{i+1})$ denote the B-spline basis function of degree m supported by the interval $\left[t_{i-m}, t_{i+1}\right]$.

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

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

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, we adopt the convention 0/0 = 0 in the above definition.

Rational Bezier curves (and surfaces), popular in the European CAD/CAM community, can be expressed exactly as rational B-spline curves (and surfaces). (See Ref. 7)

A5 SPLINE SURFACES

The spline surface defined in Section 3.9 is the analog of the spline curve, i.e., it is also pieced together out of other primitive functions. The surface is a grid of parametric bicubic patches defined by:

a. M: The number of grid lines in u,

b. tu(1),...,tu(M+1): The grid lines in u,

c. N: The number of grid lines in v,

d. tv(1),...,tv(N+1): The grid lines in v,

e. Ax(i,j),Bx(i,j),...,Ay(i,j),...,Az(i,j),...,i=1,...,M+1;j=1,...,N+1: The (M+1)*(N+1) sets of 3*16 coefficients defining the bicubic polynomial for each of the three coordinates of the patch. As for the parametric curve, additional patches not strictly required to define the surface are included to make the edge values and derivatives available without explicitly evaluating the polynomial,

f. CTYPE: The spline type. (1=linear, 2=quadratic, 3=cubic, 4=Wilson-Fowler, 5=Modified Wilson-Fowler, 6 = B-spline),

g. PTYPE: The patch type. (1=Cartesian product, 0=unspecified), and

h. H: Degree of continuity.

To evaluate the spline at a point "u,v", first determine the patch containing the point "u,v" in the parameter grid, i.e., the patch "i,j" such that $tu(i) \leq u \leq tu(i+1)$ and $tv(j) \leq v \leq tv(j+1)$, then evaluate the bicubic polynomial in that patch, i.e., compute

```
X(u,v) = Ax(i,j) *vv**0 *uu**0  + Bx(i,j) *vv**0 *uu**1
             + Cx(i,j) *vv**0 *uu**2  + Dx(i,j) *vv**0 *uu**3

         + Ex(i,j) *vv**1 *uu**0  + Fx(i,j) *vv**1 *uu**1
             + Gx(i,j) *vv**1 *uu**2  + Hx(i,j) *vv**1 *uu**3

         + Kx(i,j) *vv**2 *uu**0  + Lx(i,j) *vv**2 *uu**1
             + Mx(i,j) *vv**2 *uu**2  + Nx(i,j) *vv**2 *uu**3

         + Px(i,j) *vv**3 *uu**0  + Qx(i,j) *vv**3 *uu**1
             + Rx(i,j) *vv**3 *uu**2  + Sx(i,j) *vv**3 *uu**3

Y(u,v) = Ay(i,j) . . .

Z(u,v) = Az(i,j) . . .
where uu = u - tu(i) and vv = v - tv(j)
```

The patches in the spline surface are equivalent to the bicubic surface patch (or the Coons' patch, see p. 170, Ref. 5 for the conversion details). The parameters of the Coons' patch are given as the corner points, corner slopes, and twist vectors (similar in spirit to the point/slope representation for curves).

However, because the Specification spline is more general than splines found in many CAD/CAM systems (e.g., the APT Wilson-Fowler spline), shape-preserving transformations out of the Specification spline format may not be possible. Difficulties encountered include restrictions such as uniform breakpoint spacing and smooth second derivatives. In these cases, the conversion must be accomplished by an interpolation or smoothing process.

A6 RATIONAL B-SPLINE SURFACES

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

$$G(s,t) = \frac{\sum_{i=0}^{K1} \sum_{j=0}^{K2} W(i,j)P(i,j)b_i(s)b_j(t)}{\sum_{i=0}^{K1} \sum_{j=0}^{K2} W(i,j)b_i(s)b_j(t)}$$

where the notation is analogous to that used for rational B-spline curves.

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

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

The b_i are the <u>B-spline basis functions</u> of degree M1 determined by the <u>knot sequence</u> S(-M1),...,S(N1+M1). The b_j are the B-spline basis functions of degree M2 determined by the <u>knot sequence</u> T(-M2),...,T(N2+M2). Here, N1=K1-M1+1 and N2=K2-M2+1.

The surface itself is parameterized for $U(0) \leq s \leq U(1)$ and for $V(0) \leq t \leq V(1)$ where $S(0) \leq U(0) < U(1) \leq S(N1)$ and $T(0) \leq V(0) < V(1) \leq T(N2)$. (See Ref. 7)

A7 REFERENCES

(1) APT Computer System Manual. Volume 2 - Subroutine Library. IIT Research Institute, 1968.

(2) C. deBoor. A Practical Guide to Splines. Springer-Verlag, 1978.

(3) S. A. Coons. "Surfaces for Computer Aided Design of Space Forms." MIT Project MAC TR-41, June 1967.

(4) W. J. Gordon and R. F. Riesenfeld. "B-Spline Curves and Surfaces." in R.E. Barnhill and R. F. Riesenfeld, ed. Computer Aided Geometric Design. Academic Press, 1974.

(5) D. F. Rogers and J. A. Adams. Mathematical Elements for Computer Graphics. McGraw-Hill, 1976.

(6) I. D. Faux and M. J. Pratt. Computational Geometry for Design and Manufacture, John Wiley and Sons, 1979.

(7) Fuhr, Richard. A Tutorial on the Mathematical Details of Rational B-Splines. National Bureau of Standards, (to come).

APPENDIX B
ELECTRICAL EXAMPLE

It is the purpose of this Specification to transfer information from processor to processor in a computer-aided design and manufacturing system. As such, this Specification must be able to completely represent a design at any point in its development.

As a "thought experiment" to test the hypothesis that the Standard can, in fact, represent a product design throughout its lifetime, let us examine a simple electrical circuit shown in Figure B1. We will consider the signal string running from Pin R1 to Pin 6 of device D1. It should be pointed out that this scenario is only one of many electronic design scenarios possible. A more likely one in terms of today's capabilities would start with a finished printed wiring board design.

At the earliest stage in its design, the signal string is represented only in its logical sense. Figure B2 is a model of an associativity representing the logical signal string. Class 1 of the signal string associativity shows its logical structure. In Figure E2, this is the branch in the middle marked R1 and D1-6. The signal name itself, SR1, is represented in the associativity by means of a property.

The logical structure of the string is sufficient, for example, to represent the information necessary for simulating the circuit. That is, if all signal strings were represented as in class 1 of Figure B2, a logic simulator could be run which would simulate the circuit and allow the engineer to verify that the circuit did, in fact, accomplish the design goals. Similar information is necessary for generating test patterns to be applied to the completed circuit to verify that the components and wiring in the circuit are, in fact, correct and that the circuit functions as designed.

At the next stage in the design, our signal string and others like it are represented as a schematic diagram. It is a simple matter to extend the associativity in Figure B2 to pick up the details of the schematic diagram. In the case of SR1, this is represented by class 2 on the left-hand side marked "schematic goemetry."

Each of the pointers represented in Figure B2 as blanks would point to an element in the signal string, i.e., a specific line in the schematic. The complete design, up to this point, is represented by the schematic as shown in Figure B1 plus an associativity, such as the one shown in Figure B2 for SR1, representing each of the signal strings in the schematic.

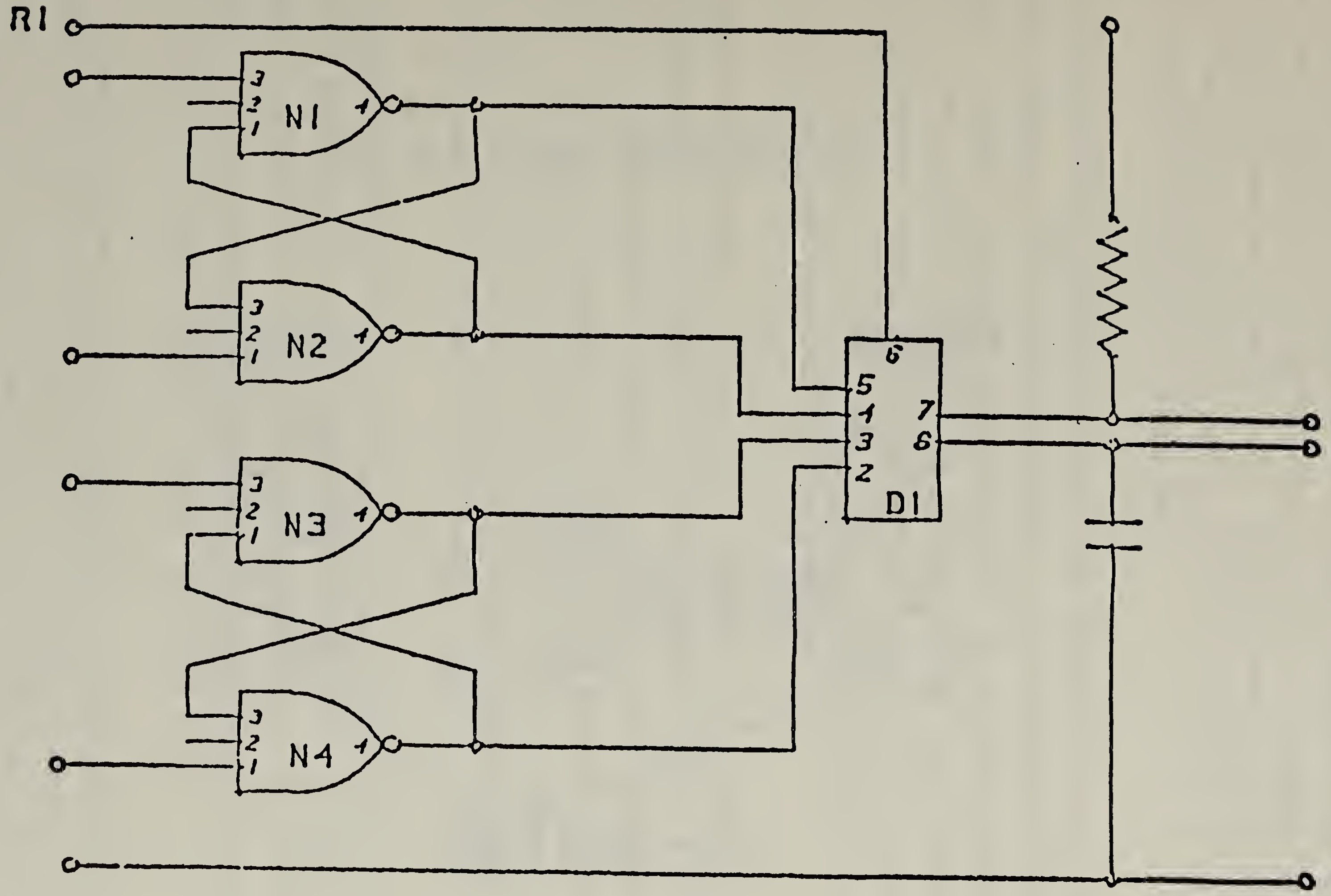


FIG. B-1 SCHEMATIC

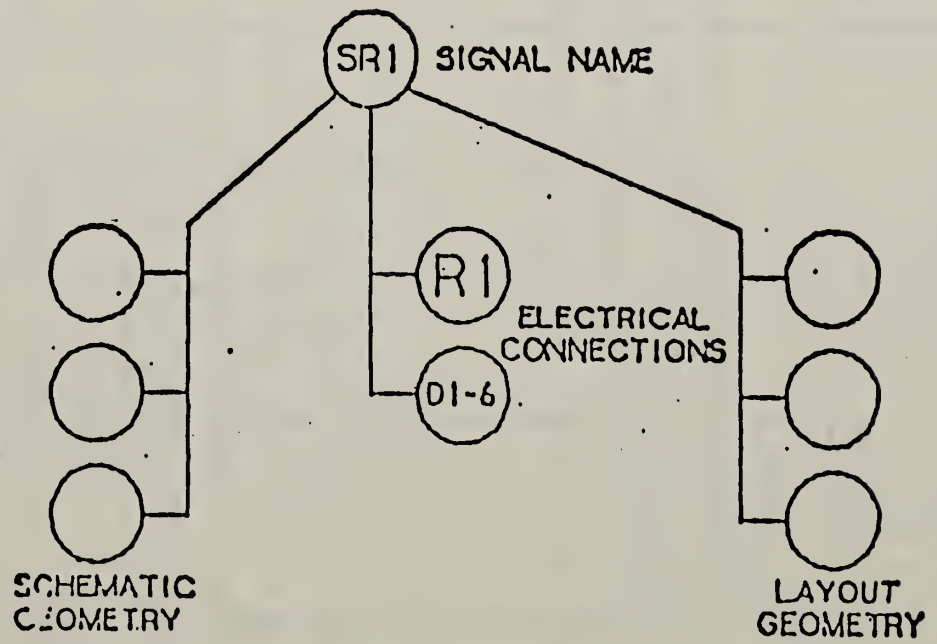


FIG. B-2 SIGNAL ASSOCIATIVITY

Up to this point, we have assumed that the information necessary for constructing the logical signal string existed before the schematic diagram was drawn. Our argument would be equally valid if the schematic were drawn first and the logical signal strings derived from it. The point is that both types of information--the logical signal strings and the schematic diagram--are essential to the design and must be represented both by the design database and by a transmission medium, such as the Standard, which is intended to transfer information in and out of this database.

If the associativities were produced first, they could come from such sources as a higher-level design language or a set of boolean equations. If the schematic is produced first, the information necessary to produce the associativities can be derived directly from the schematic automatically by the interactive graphics system. There remains the question of the signal names. In some cases these are meaningful to the user, and should be entered interactively. In other cases, it is perfectly permissible to enter them automatically using names generated by the design system.

The next step in the design of the circuit is partitioning the logical components (the devices represented in this schematic) to physical components which will eventually be placed on the finished printed wiring board. Such a procedure is called partitioning, and the results of the partitioning process are represented in an associativity such as shown in Figure B3. This relates the physical component U1 to the three logical components, N1, N2, N3. Note that gate N4 is a part of physical component U2, the remainder of U2 being unused. The pointers in the associativity represented in Figure E3 all point to subfigures.

The next step in a printed wiring board design is the placement and layout of the printed wiring board itself. The result of this step is shown in Figure B4. The signal string we are considering runs from Pin R1 to Pin 6 of physical device U3. This is shown dashed in Figure B4. The pointers represented on the right-hand branch (class 3) of the associativity shown in Figure B2 would each point to a segment of this wire run.

The entire board of Figure B4 would be represented in a similar fashion with the set of associativities built up in the construction of the schematic diagram pointing to the appropriate wire runs on the board using class 3. Thus, information is available for doing comparisons between the board and the schematic and thereby verifying that the implemented board agrees with the schematic and the set of logical signal strings.

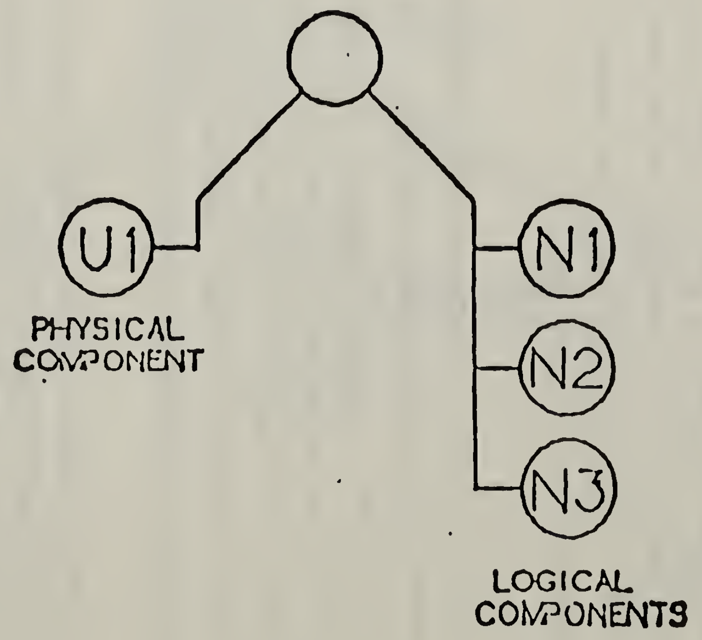


FIG. B-3 PARTITIONING ASSOCIATIVITY

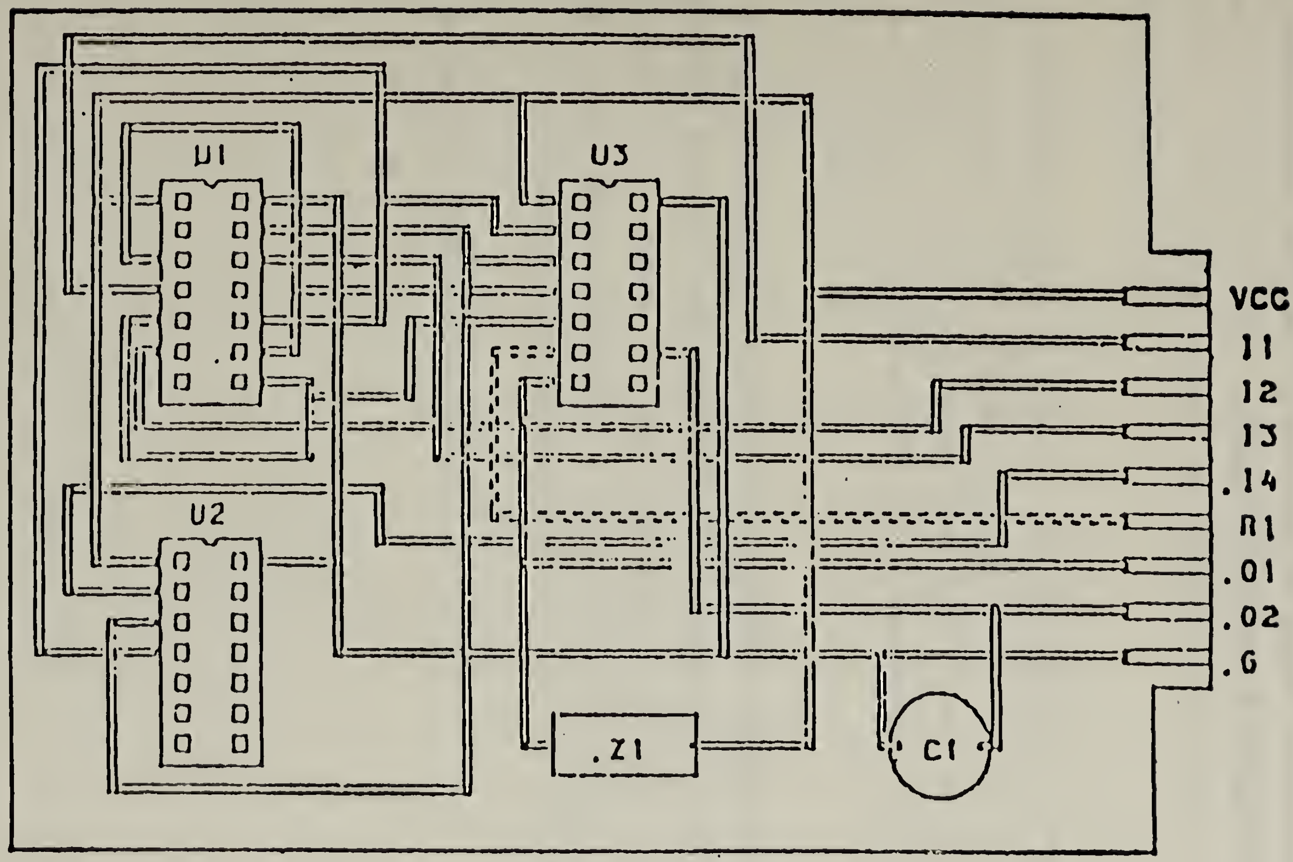


FIG. B-4 LAYOUT

All of the information described in this example can be placed in the single Standard file. The method of doing so is to present the information in Figure B1 as a single view, the information in Figure B4 as a single view, and then, if desired, present the appropriate view on each of two drawings. Such a mechanism serves to segregate the information into either schematic diagram or board layout, but at the same time maintains the information in the same data base so that the associativities which point to the two drawings can function appropriately.

APPENDIX C
PART FILE EXAMPLES

This appendix contains two sample parts encoded in the Specification format. The sample part which is shown in Figure C1 is a two-dimensional representation of a mechanical part comprised of lines, circles, a linear dimension, a radius dimension, and an angular dimension. The drafting entities, by definition, are made up of witness lines, general notes and leader entities.

The encoded file is shown in Table C1. On line P0000020 of the table, the degree symbol is shown as a part of the angular dimension entity. This symbol is represented by the octal constant 37 as dictated by the font code of zero; see Figure 4-11. Note that the entities with entity type number 500 are also used to define the geometry and to add face topology to the file.

The sample part shown in Figure C2 is an unknown application which has been included to demonstrate the use of associativities and properties. It consists of an outline (entities 3-18) together with several line segments and points in the interior. These interior line segments and points are described by properties 5001-5003, as well as by associativity 5001. Note that each of the segments is represented as a composite entity.

The encoded file is shown in Table C2.

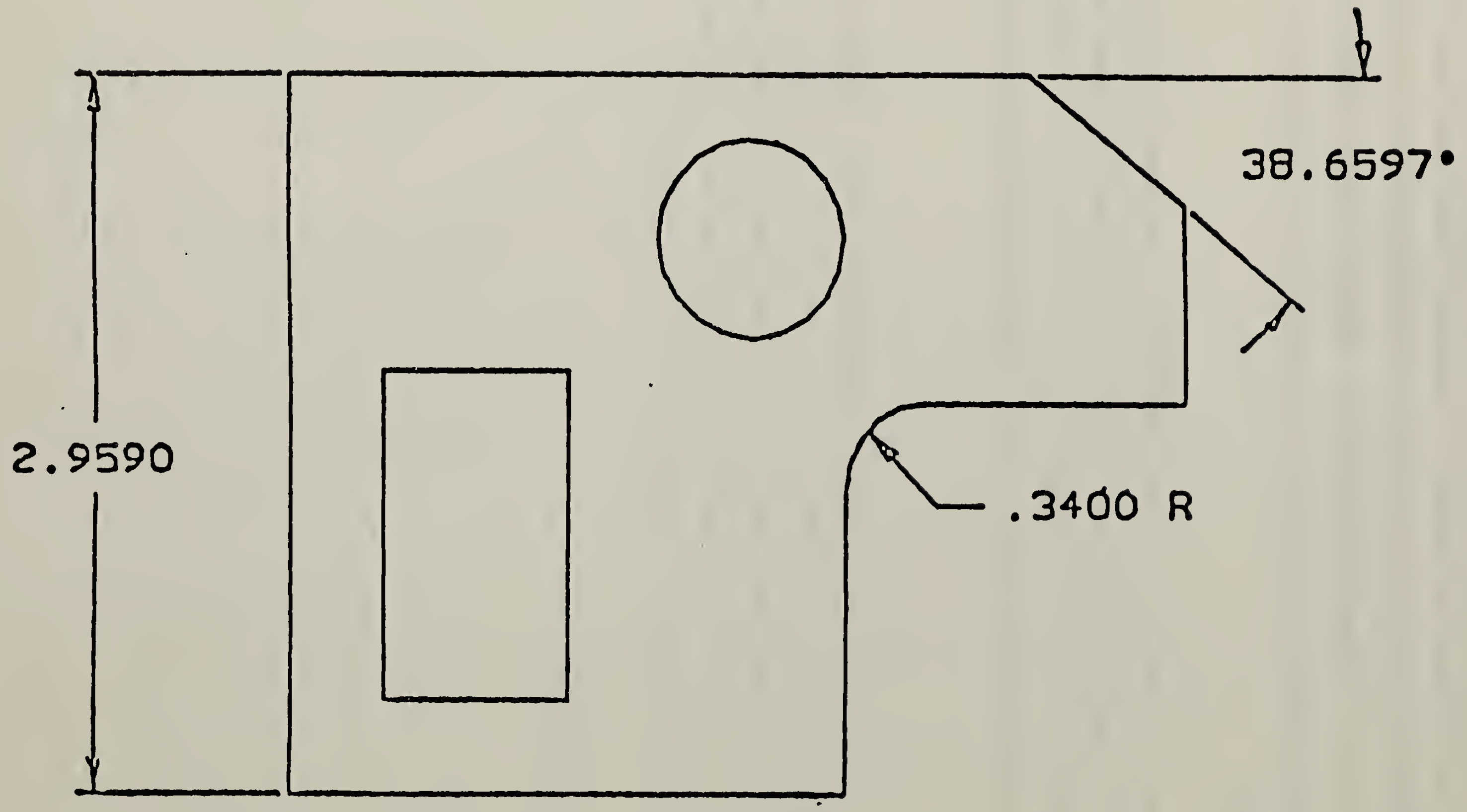


FIG. C-1 SAMPLE PART

TABLE C-1 ENCODED FILE

```
   SAMPLE PART                                                          S0000001
,,11H112C87901.5,11HIGES SAMPLE,6HME1.00,1H1,16,8,24,8,56,11H112C87901.5G0000001
,1.0,1,4HINCH,1,.01,13H800128.093243,.1345,800,5HJ.DOE,8HGRAPHICS;      G0000002
     12400000001       1                                        000000  D0000001
     124                       1                             MTX       1D0000002
     11000000002       1       1      10               0        000000  D0000003
     110       0       1       1                               L       1D0000004
     11000000003       1       1      10               0        000000  D0000005
     110       0       1       1                               L       2D0000006
     11000000004       1       1      10               0        000000  D0000007
     110       0       1       1                               L       3D0000008
     11000000005       1       1      10               0        000000  D0000009
     110       0       1       1                               L       4D0000010
     11000000006       1       1      10               0        000000  D0000011
     110       0       1       1                               L       5D0000012
     11000000007       1       1      10               0        000000  D0000013
     110       0       1       1                               L       6D0000014
     11000000008       1       1      10               0        000000  D0000015
     110       0       1       1                               L       7D0000016
     11000000009       1       1      10               0        000000  D0000017
     110       0       1       1                               L       8D0000018
     11000000010       1       1      10               0        000000  D0000019
     110       0       1       1                               L       9D0000020
     11000000011       1       1      10               0        000000  D0000021
     110       0       1       1                               L      10D0000022
     10000000012       1       1      10               0        000000  D0000023
     100       0       1       1                               C       1D0000024
     11000000013       1       1      10               0        000000  D0000025
     110       0       1       1                               L      11D0000026
     21200000014       1       1      1200000055       0        000101  D0000027
     212       0       1       1                                        D0000028
     21400000015       1       1      1200000055       0        000101  D0000029
     214       0       1       1       1                                D0000030
     21400000016       1       1      1200000055       0        000101  D0000031
     214       0       1       1       1                                D0000032
     10600000017       1       1      1200000055       0        000101  D0000033
     106       0       1       1      40                                D0000034
     21600000019       1       1      1200000055       0        000001  D0000035
     216       0       1       1                                        D0000036
     21200000020       1       1      1200000055       0        000101  D0000037
     212       0       1       1                                        D0000038
     21400000021       1       1      1200000055       0        000101  D0000039
     214       0       1       1       1                                D0000040
     21400000022       1       1      1200000055       0        000101  D0000041
     214       0       1       1       1                                D0000042
     10600000023       1       1      1200000055       0        000101  D0000043
     106       0       1       1      40                                D0000044
     20200000025       1       1      1200000055       0        000001  D0000045
     202       0       1       1                                        D0000046
     21200000026       1       1      1200000055       0        000101  D0000047
     212       0       1       1                                        D0000048
     21400000027       1       1      1200000055       0        000101  D0000049
     214       0       1       1       1                                D0000050
     22200000028       1       1      1200000055       0        000001  D0000051
     222       0       1       1                                        D0000052
     10000000029       1       1      10               0        000000  D0000053
     100       0       1       1                              C        2D0000054
     41000000030       1                               00000001 000001  D0000055
     410                       1                                        D0000056
     10600000018       1       1      1200000055       0        000101  D0000057
     106       0       1       1      40                                D0000058
     10600000024       1       1      1200000055       0        000101  D0000059
     106       0       1       1      40                                D0000060
124,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0;            00000001P0000001
110,2.0175,1.4795,0.0,4.4385,1.4795,0.0;                        00000003P0000002
110,4.4385,1.4795,0.0,4.4385,2.7535,0.0;                        00000005P0000003
110,4.7785,3.0935,0.0,5.9180,3.0935,0.0;                        00000007P0000004
110,5.918,3.0935,0.0,5.918,3.9005,0.0;                          00000009P0000005
110,5.2455,4.4385,0.0,2.0175,4.4385,0.0;                        00000011P0000006
110,2.0175,4.4385,0.0,2.0175,1.4795,0.0;                        00000013P0000007
110,2.421,1.883,0.0,3.228,1.883,0.0;                            00000015P0000008
110,3.228,1.883,0.0,3.228,3.228,0.0;                            00000017P0000009
110,3.228,3.228,0.0,2.421,3.228,0.0;                            00000019P0000010
110,2.421,3.228,0.0,2.421,1.883,0.0;                            00000021P0000011
100,0.0,4.7785,2.7535,4.7785,3.0935,4.4385,2.7535;              00000023P0000012
110,5.918,3.9005,0.0,5.2455,4.4385,0.0;                         00000025P0000013
212,1,6,.48,.1,0,1.570796,0.0,0,0,.807,2.8245,0.0;6H2.9590;     00000027P0000014
214,1,.1,.08,0.0,1.157,4.4385,1.157,3.0245;                     00000029P0000015
214,1,.1,.08,0.0,1.157,1.4795,1.157,2.7245;                     00000031P0000016
106,1,3,0.0,1.97844,4.4385,1.97844,4.4385,1.07888,4.4385;       00000033P0000017
106,1,3,0.0,1.97844,1.4795,1.97844,1.4795,1.07888,1.4795;       00000057P0000018
216,27,29,31,33,79;                                             00000035P0000019
212,1,8,.64,.1,0,1.570796,0.0,0,0,6.187,4.035,0.0,8H38.6597°;   00000037P0000020
214,1,.1,.08,0.0,6.38174,3.5295,6.17118,3.315;                  00000039P0000021
214,1,.1,.08,0.0,6.70059,4.4385,6.66967,4.7369;                 00000041P0000022
106,1,3,0.0,5.9485,3.8761,5.9485,3.8761,6.44274,3.48071;        00000043P0000023
106,1,3,0.0,5.28456,4.4385,5.28456,4.4385,6.77872,4.4385;       00000059P0000024
202,37,43,81,5.2455,4.4385,1.45509,39,41;                       00000045P0000025
212,1,7,.56,.1,0,1.570796,0.0,0,0,5.142,2.64,0.0,7H.3400 R;     00000047P0000026
214,2,.1,.08,0.0,4.53809,2.99392,4.842,2.69,5.042,2.69;         00000049P0000027
222,00000047,00000049,4.7785,2.7535;                            00000051P0000028
100,0.0,4.035,3.766,4.4385,3.766,4.4385,3.766;                  00000053P0000029
410,1;                                                          00000055P0000030
S0000001G0000002D0000060P0000030                                        T0000001
```

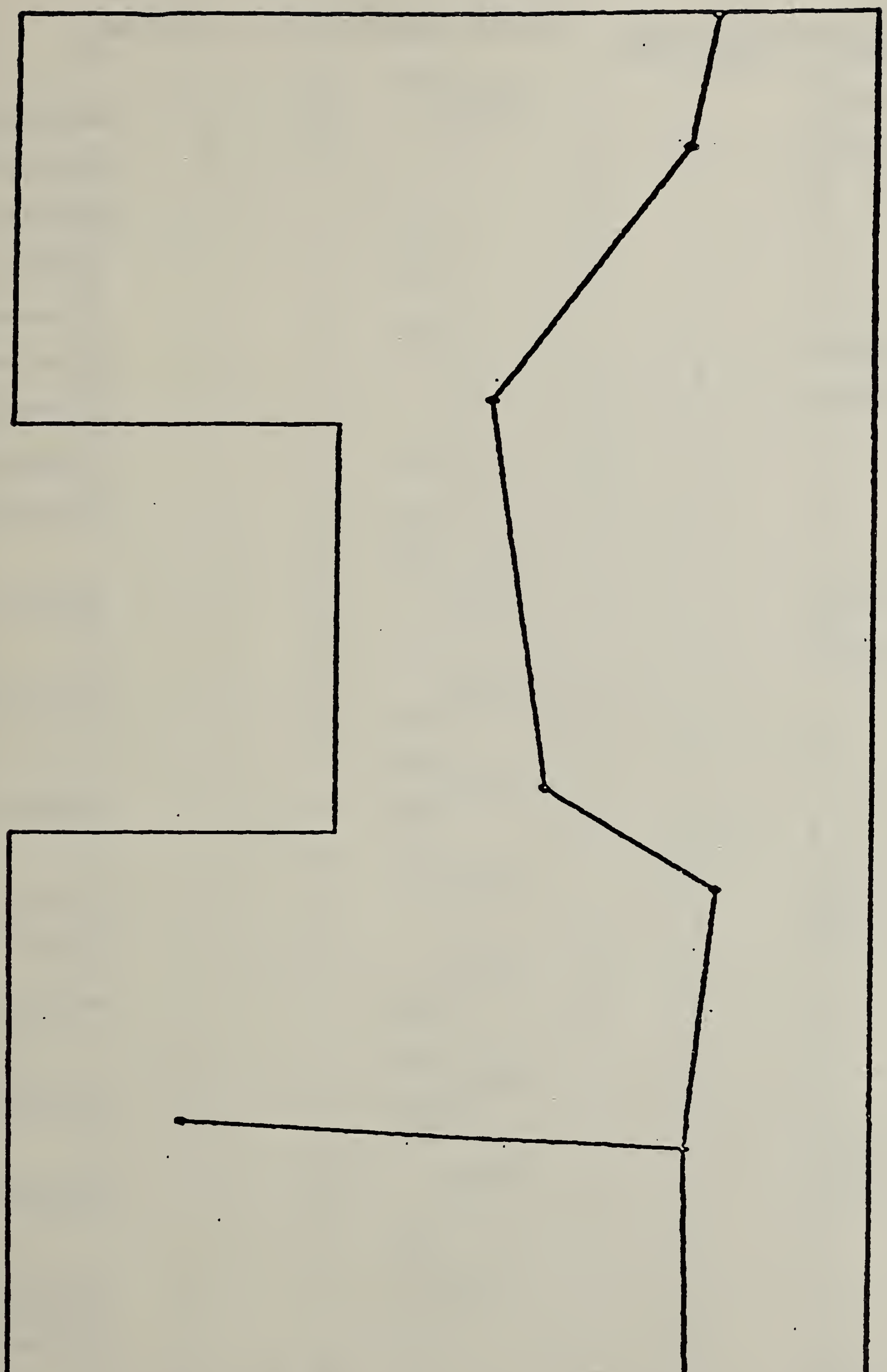


FIG. C-2 SAMPLE PART 2

TABLE C-2 ENCODED FILE

```
SAMPLE APPL                                                             SØØØØØØ1
,,8HMLD7.DAT,4HAPPL,2HAB,,16,,,,,4HAPPL,1,1,4HINCH,Ø,1.,13H1Ø278Ø,      GØØØØØØ1
13ØØØØ,.ØØØØØ1,1Ø.,3HPRK,3HCRD                                          GØØØØØØ2
     3Ø2       1       1                                                DØØØØØØ1
     3Ø2                       1    5ØØ1                                DØØØØØØ2
     11Ø       2       1       1       1                        ØØØØØØØØDØØØØØØ3
     11Ø       Ø       1       1                                        DØØØØØØ4
     11Ø       3       1       1       1                        ØØØØØØØØDØØØØØØ5
     11Ø       Ø       1       1                                        DØØØØØØ6
     11Ø       4       1       1       1                        ØØØØØØØØDØØØØØØ7
     11Ø       Ø       1       1                                        DØØØØØØ8
     11Ø       5       1       1       1                        ØØØØØØØØDØØØØØØ9
     11Ø       Ø       1       1                                        DØØØØØ1Ø
     11Ø       6       1       1       1                        ØØØØØØØØDØØØØØ11
     11Ø       Ø       1       1                                        DØØØØØ12
     11Ø       7       1       1       1                        ØØØØØØØØDØØØØØ13
     11Ø       Ø       1       1                                        DØØØØØ14
     11Ø       8       1       1       1                        ØØØØØØØØDØØØØØ15
     11Ø       Ø       1       1                                        DØØØØØ16
     11Ø       9       1       1       1                        ØØØØØØØØDØØØØØ17
     11Ø       Ø       1       1                                        DØØØØØ18
     116      1Ø       1       1       2                        ØØØØØØØØDØØØØØ19
     116       Ø       1       1                                        DØØØØØ2Ø
     4Ø6      11       1                                                DØØØØØ21
     4Ø6                       1    5ØØ1                                DØØØØØ22
     116      12       1       1       2                        ØØØØØØØØDØØØØØ25
     116       Ø       1       1                                        DØØØØØ26
     4Ø6      13       1                                                DØØØØØ27
     4Ø6                       1    5ØØ1                                DØØØØØ28
     116      14       1       1       2                        ØØØØØØØØDØØØØØ31
     116       Ø       1       1                                        DØØØØØ32
     4Ø6      15       1                                                DØØØØØ33
     4Ø6                       1    5ØØ1                                DØØØØØ34
     116      16       1       1       2                        ØØØØØØØØDØØØØØ37
     116       Ø       1       1                                        DØØØØØ38
     4Ø6      17       1                                                DØØØØØ39
     4Ø6                       1    5ØØ1                                DØØØØØ4Ø
     116      18       1       1       2                        ØØØØØØØØDØØØØØ43
     116       Ø       1       1                                        DØØØØØ44
     4Ø6      19       1                                                DØØØØØ45
     4Ø6                       1    5ØØ1                                DØØØØØ46
     116      2Ø       1       1       2                        ØØØØØØØØDØØØØØ49
     116       Ø       1       1                                        DØØØØØ5Ø
     4Ø6      21       1                                                DØØØØØ51
     4Ø6                       1    5ØØ1                                DØØØØØ52
     116      22       1       1       2                        ØØØØØØØØDØØØØØ55
     116       Ø       1       1                                        DØØØØØ56
     4Ø6      23       1                                                DØØØØØ57
     4Ø6                       1    5ØØ1                                DØØØØØ58
     116      24       1       1       2                        ØØØØØØØØDØØØØØ61
     116       Ø       1       1                                        DØØØØØ62
     4Ø6      25       1                                                DØØØØØ63
     4Ø6                       1    5ØØ1                                DØØØØØ64
     1Ø2      26       1       1       2                        ØØØØØØØØDØØØØØ67
     1Ø2       Ø       1       1                                        DØØØØØ68
     11Ø      27       1       1       2                        ØØØ1ØØØØDØØØØØ69
     11Ø       Ø       1       1                                        DØØØØØ7Ø
     4Ø6      28       1                                                DØØØØØ71
     4Ø6                       1    5ØØ1                                DØØØØØ72
     4Ø6      29       1                                                DØØØØØ73
```

```
406                       1     5001                                D0000074
406      30       1                                                 D0000077
406                       1     5002                                D0000078
102      31       1       1        2                        00000000D0000079
102       0       1       1                                         D0000080
110      32       1       1        2                        00010000D0000081
110       0       1       1                                         D0000082
406      33       1                                                 D0000083
406                       1     5001                                D0000084
406      34       1                                                 D0000085
406                       1     5001                                D0000086
406      35       1                                                 D0000089
406                       1     5002                                D0000090
102      36       1       1        2                        00000000D0000091
102       0       1       1                                         D0000092
110      37       1       1        2                        00010000D0000093
110       0       1       1                                         D0000094
406      38       1                                                 D0000095
406                       1     5001                                D0000096
406      39       1                                                 D0000097
406                       1     5001                                D0000098
406      40       1                                                 D0000101
406                       1     5002                                D0000102
102      41       1       1        2                        00000000D0000103
102       0       1       1                                         D0000104
110      42       1       1        2                        00010000D0000105
110       0       1       1                                         D0000106
406      43       1                                                 D0000107
406                       1     5001                                D0000108
406      44       1                                                 D0000109
406                       1     5001                                D0000110
402      45      -1                                                 D0000111
402                       2     5001                LINE=1          D0000112
406      47       1                                                 D0000113
406                       1     5002                                D0000114
102      48       1       1        2                        00000000D0000115
102       0       1       1                                         D0000116
110      49       1       1        2                        00010000D0000117
110       0       1       1                                         D0000118
406      50       1                                                 D0000119
406                       1     5001                                D0000120
406      51       1                                                 D0000121
406                       1     5001                                D0000122
406      52       1                                                 D0000125
406                       2     5002                                D0000126
102      54       1       1        2                        00000000D0000127
102       0       1       1                                         D0000128
110      55       1       1        2                        00010000D0000129
110       0       1       1                                         D0000130
406      56       1                                                 D0000131
406                       1     5001                                D0000132
406      57       1                                                 D0000133
406                       1     5001                                D0000134
406      58       1                                                 D0000137
406                       1     5002                                D0000138
102      59       1       1        2                        00000000D0000139
102       0       1       1                                         D0000140
110      60       1       1        2                        00010000D0000141
110       0       1       1                                         D0000142
406      61       1                                                 D0000143
```

```
     4Ø6                               1    5ØØ1                                DØØØØ144
     4Ø6      62       1                                                        DØØØØ145
     4Ø6                               1    5ØØ1                                DØØØØ146
     4Ø6      63       1                                                        DØØØØ149
     4Ø6                               1    5ØØ2                                DØØØØ15Ø
3Ø2,3,1,2,1,1,1,1,1,1,1,1,1,1;SIGNAL ASOCIATIVITY DEF.                 1P      1
11Ø,Ø,Ø,Ø,7,Ø,Ø;                                                       3P      2
11Ø,7,Ø,Ø,7,4,Ø;                                                       5P      3
11Ø,7,4,Ø,5,4,Ø;                                                       7P      4
11Ø,5,4,Ø,5,2.5,Ø;                                                     9P      5
11Ø,5,2.5,Ø,3,2.5,Ø;                                                  11P      6
11Ø,3,2.5,Ø,3,4,Ø;                                                    13P      7
11Ø,3,4,Ø,Ø,4,Ø;                                                      15P      8
11Ø,Ø,4,Ø,Ø,Ø,Ø;                                                      17P      9
116,6.989593,Ø.737364,Ø,,1,111,1,21;                                  19P     1Ø
4Ø6,3,1H2,4H9361,1HØ;AB2 PROPERTY                                     21P     11
116,5.116452,1.791873,Ø,,2,111,111,1,27;                              25P     12
4Ø6,3,1H2,4H9361,1HØ;AB2 PROPERTY                                     27P     13
116,2.716Ø55,.723489,Ø,,2,111,111,1,33;                               31P     14
4Ø6,3,1H2,4H9361,1HØ;AB2 PROPERTY                                     33P     15
116,-Ø.Ø45Ø94,Ø.834489,Ø,Ø,1,111,1,39;                                37P     16
4Ø6,3,1H2,4H9361,1HØ;AB2 PROPERTY                                     39P     17
116,6.351338,Ø.86224Ø,Ø,,2,111,111,1,45;                              43P     18
4Ø6,3,1H2,4H9361,1HØ;AB2 PROPERTY                                     45P     19
116,1.56442Ø,3.193261,Ø,,1,111,1,51;                                  49P     2Ø
4Ø6,3,1H2,4H9361,1HØ;AB2 PROPERTY                                     51P     21
116,1.425669,Ø.86224Ø,Ø,,3,111,111,111,1,57;                          55P     22
4Ø6,3,1H2,4H9361,1HØ;AB2 PROPERTY                                     57P     23
116,3.21556Ø,1.528246,Ø,,2,111,111,1,63;                              61P     24
4Ø6,3,1H2,4H9361,1HØ;AB2 PROPERTY                                     63P     25
1Ø2,1,69,1,111,3,71,73,77;                                            67P     26
11Ø,3.21556Ø,1.528246,Ø,5.116452,1.791873,Ø;                          69P     27
4Ø6,1,5H.4375;                                                        71P     28
4Ø6,4,1H1,5H16384,1HØ,3H125;AB1 PROPERTY                              73P     29
4Ø6,4,5H14344,1H3,6HCODE=4,9HDEPTH=1.4;APPL PROPERTY                  77P     3Ø
1Ø2,1,81,1,111,3,83,85,89;                                            79P     31
11Ø,2.716Ø55,Ø.723489,Ø,3.21556Ø,1.528246,Ø;                          81P     32
4Ø6,1,5H.4375;                                                        83P     33
4Ø6,4,1H1,5H16384,1HØ,3H125;AB1 PROPERTY                              85P     34
4Ø6,4,5H14344,1H3,6HCODE=4,9HDEPTH=1.3;APPL PROPERTY                  89P     35
1Ø2,1,93,1,111,3,95,97,1Ø1;                                           91P     36
11Ø,1.425669,Ø.86224Ø,Ø,2.716Ø55,Ø.723489,Ø;                          93P     37
4Ø6,1,5H.4375;                                                        95P     38
4Ø6,4,1H1,5H16384,1HØ,3H125;AB1 PROPERTY                              97P     39
4Ø6,4,5H14344,1H3,6HCODE=4,9HDEPTH=1.4;APPL PROPERTY                 1Ø1P     4Ø
1Ø2,1,1Ø5,1,111,3,1Ø7,1Ø9,113;                                       1Ø3P     41
11Ø,-Ø.Ø45Ø94,Ø.834489,Ø,1.425669,Ø.86224Ø,Ø;                        1Ø5P     42
4Ø6,1,5H.4375;                                                       1Ø7P     43
4Ø6,4,1H1,5H16384,1HØ,3H125;AB1 PROPERTY                             1Ø9P     44
4Ø2,Ø,15,Ø,37,1Ø3,55,115,49,31,91,61,79,25,67,43,127,19,139,         111P     45
Ø,Ø;                                                                 111P     46
4Ø6,3,5H14344,1H3,6HCODE=Ø;APPL PROPERTY                             113P     47
1Ø2,1,117,1,111,3,119,121,125;                                       115P     48
11Ø,1.425669,Ø.86224Ø,Ø,1.56442Ø,3.193261,Ø;                         117P     49
4Ø6,1,5H.4375;                                                       119P     5Ø
4Ø6,4,1H1,5H16384,1HØ,3H125;AB1 PROPERTY                             121P     51
4Ø6,7,5H14344,1H3,6HCODE=2,9HDEPTH=3.4,1ØHPITCH=Ø.33,13HEFLENGTH     125P     52
=1.34,15HWALLTHICK=Ø.125;APPL PROPERTY                               125P     53
1Ø2,1,129,1,111,3,131,133,137;                                       127P     54
11Ø,5.116452,1.791873,Ø,6.351338,Ø.86224Ø,Ø;                         129P     55
```

```
406,1,4H0.75;                                                       131P     56
406,4,1H1,5H16384,1H0,3H125;AB1 PROPERTY                            133P     57
406,4,5H14344,1H3,6HCODE=4,9HDEPTH=1.5;APPL PROPERTY                137P     58
102,1,141,1,111,3,143,145,149;                                      139P     59
110,6.351338,0.862240,0,6.989593,0.737364,0;                        141P     60
406,1,5H.4375;                                                      143P     61
406,4,1H1,5H16384,1H0,3H125;AB1 PROPERTY                            145P     62
406,3,5H14344,1H3,6HCODE=0;APPL PROPERTY                            149P     63
S0000001G0000002D0000150P     63                                      T0000001
```

APPENDIX D
IGES GLOSSARY

The spirit of this Glossary is to provide general, sometimes intuitive information pertaining to certain phrases and concepts either appearing in or alluded to by this document. The spirit is not to provide detailed mathematical definitions such as may be found within the document itself.

ANGULAR DIMENSION ENTITY

An annotation entity designating the measurement of the angle between two geometric lines.

ANNOTATION

Text or symbols, not part of the geometric model, which provide information.

ASSEMBLY (IEEE 200-1975)

A number of basic parts or subassemblies, or any combination thereof, joined together to perform a specific function.

ASSOCIATIVITY

A structure entity, which defines a logical link or relationship between different entities.

ASSOCIATIVITY DEFINITION ENTITY

A structure entity which designates the type (link structure) and generic meaning of a relationship. (See PRE-DEFINED ASSOCIATIVITIES)

ASSOCIATIVITY INSTANCE ENTITY

A structure entity formed by assigning specific values to the data items defining an associativity.

ATTRIBUTE

Information, provided in specific fields within the directory entry of an entity, which serves to qualify the entity definition.

AXONOMETRIC PROJECTION

A projection in which only one plane is used, the object being turned so that three faces show. The main axonometric positions are isometric, dimetric, and trimetric.

BACK POINTER

A pointer in the parameter data section of an entity pointing to an associativity instance of which it is a member.

BLANK STATUS FLAG

A portion of the status number field of the directory entry of an entity designating whether a data item is to be displayed on the output device.

BOUNDED PLANE

A finite region defined in a plane.

BREAKPOINT

A member of an increasing sequence of real numbers which is a subsequence of the knot sequence used to specify parametric spline curves.

B-SPLINE BASIS

A set of functions which form a basis for the set of splines of specified degree on a specified knot sequence. B-spline basis functions are characterized by being splines of minimal support. See appendix A4 for more details.

CENTERLINE ENTITY

An annotation entity for representing the axis of symmetry for all symmetric views or portions of views, such as the axis of a cylinder or a cone.

CIRCULAR ARC ENTITY

A geometric entity which is a connected portion of a circle or the entire circle.

CLASS

A group of data items pertinent to a common logical relationship in an associativity definition.

CLIP

To abbreviate or terminate the intended display of an entity along an intersecting curve or surface.

CLIPPING BOX

A bounding set of surfaces which abbreviate the intended display of data to that portion which lies within the box.

CLIPPING PLANE

A bounding plane surface which abbreviates the intended display of data to that portion which lies on one or the other side of the plane.

CLOSED CURVE

A curve with coincident start and terminate points.

COMPLEMENTARY ARC

Either of the two connected components of a closed connected curve which has been divided by two distinct points lying on the curve.

COMPONENT

Typically a synonym for part (e.g., resistor, capacitor, microcircuit, etc.), but also may refer to a subassembly being treated as a part. The IGES representation of a component may be a collection of entities, associativities, and properties.

COMPOSITE CURVE

A connected curve which is formed by concatenating two or more curve segments.

CONIC ARC ENTITY

A geometric entity which is a finite connected portion of an ellipse, a parabola, or a hyperbola.

CONNECTED CURVE

A curve such that if for any two points P1 and P2, one can travel from P1 to P2 without leaving the curve.

CONSTITUENT

A member of a set.

CONTROL POINT

A point in definition space which appears in the numerator of the expression for a rational B-spline curve or surface. If the weights are all positive, the resulting curve or surface lies within the convex hull of the control points. Its shape resembles that of the polygon or polyhedron whose vertices are the control points. A control point is sometimes referred to as a B-spline coefficient. See appendices A4 and A6 for more details.

COONS PATCH

A three dimensional surface.

COPIOUS DATA ENTITY

A geometric entity sometimes used as an annotation entity, containing arrays of types of real numbers to which a specific meaning has been assigned. One form number corresponds to one special meaning.

DEFINITION LEVEL (or DISPLAY LEVEL)

The graphics display level (or layer) on which one or more entities have been defined.

DEFINITION MATRIX

The matrix which transforms the coordinates represented in the definition space into the coordinates represented in the model space.

DEFINITION SPACE

A local Cartesian coordinate system chosen to represent a geometric entity for the purpose of mathematical simplicity.

DEFINITION SPACE SCALE

A scale factor applied within an entity definition space.

DEVELOPABLE SURFACE

A surface which can be unrolled onto a plane.

DIAMETER DIMENSION ENTITY

An annotation entity designating the measurement of a diameter of a circular arc.

DIRECTED CURVE

A curve with an associated direction derived from the start and terminate points.

DIRECTORY ENTRY SECTION

The section of an IGES file, consisting of fixed field data items for an index and attribute list of all entities in the file.

DIRECTRIX

The curve entity used in the definition of a tabulated cylinder entity.

DISPLAY SYMBOL

A method for graphically representing certain entities (plane, point, section) for identification purposes.

DRAWING ENTITY

A structure entity which specifies the projection(s) of a model onto a plane, with any required annotation and/or dimension.

EDGE VERTEX

A method of geometric modeling in which a two- or three-dimensional object is represented by curve segments (edges of the object) connected to points or vertices of the object. A higher level of topological information can be contained in such a model than is implied by a 'wire-frame'-terminology, but in the context of this specification the terms are used interchangeably.

ENTITY

The basic unit of information in a file. The term applies to single items which may be individual elements of geometry, collections of annotation to form dimensions, or collections of entities to form structured entities.

ENTITY LABEL

A one to eight character identifier for an entity. This term may implicitly include the entity subscript, providing for additional characters.

ENTITY SUBSCRIPT

A one to eight digit unsigned integer associated with the entity label. The label and subscript specify a unique instance of an entity within an array of entities.

ENTITY TYPE NUMBER

An integer used to specify the kind of the entity. For example, the circular arc entity has an entity type number of 100.

ENTITY USE FLAG

A portion of the status number field of the directory entry of an entity to designate whether the entity is used as geometry, annotation, structure, logical, or other. For example, a circle used as part of a point dimension would have an entity use flag which designates annotation.

FINITE ELEMENT

A small part of a structure defined by the connection of nodes, material, and physical properties.

FLAG NOTE ENTITY

An annotation entity which takes label information and formats it such that the text is circumscribed by a flag symbol.

FLEXIBLE PRINTED CIRCUIT

An arrangement of printed circuit and components utilizing flexible base materials with or without flexible cover layers.

FONT CHARACTERISTIC

An integer which is used to identify a text font. Font characteristic numbers may be positive which indicate an IGES-defined text font or may be negative which is interpreted as a text font definition entity.

FORM NUMBER

An integer which is used when needed to further define a specific entity. This becomes necessary when there are several interpretations of an entity type. For example, the form number of the conic arc entity indicates whether the curve is an ellipse, hyperbola, parabola, or unspecified. The form number is also used when necessary to supply sufficient information in the directory entry of an entity to allow the structure of the parameters in the parameter data entry to be decoded.

GENERAL LABEL ENTITY

An annotation entity consisting of a general note with one or more associated leaders.

GENERAL NOTE ENTITY

An annotation which consists of text which is to be displayed in some specific size and at some specific location and orientation.

GENERATRIX

The defining curve which is to be swept to generate a tabulated cylinder, or revolved to generate a surface of revolution.

GEOMETRIC

Having to do with the shape information (points, curves, surfaces, and volumes), necessary to represent some object.

GLOBAL SECTION

The section of an IGES file consisting of general information describing the file, the file generator (pre-processor), and information needed by the file reader (post-processor).

GRID

The set of (ui, vj) where ui and vj are the breakpoints on the u and v coordinates respectively used to specify a parametric spline or rational B-spline surface. The term grid is also applied to the projected image on the spline surface.

GROUND PLANE

A conductor layer, or portion of a conductor layer (usually a continuous sheet of metal with suitable clearances), used as a common reference point for circuit returns, shieldings, or heat sinking.

GROUP ASSOCIATIVITY

A predefined associativity for forming any collection of entities.

HIERARCHY

A tree structure consisting of a root and one or more dependents. In general, the root may have any number of dependents, each of which may have any number of lower-level dependents, and so on, to any number of levels.

INSTANCE

A particular occurrance of some item or relationship. Several instances may reference the same item.

KNOT SEQUENCE

A nondecreasing sequence of real numbers used to specify parametric spline curves.

LABEL DISPLAY ASSOCIATIVITY

A pre-defined associativity that is used by those entities that have one or more possible displays for their entity label. Entities requiring this associativity will have pointers in their directory entry to a label display associativity instance entity.

LEADER ENTITY

An annotation entity, also referred to as arrow, which consists of an arrowhead and one or more line segments. In the case of an angular dimension entity, the line segment is replaced by a circular arc segment. In general, these entities are used in connection with other annotation entities to link text with some location.

LEVEL

An entity attribute which defines a graphic display level to be associated with the entity.

LINE ENTITY

A geometric entity consisting of a straight segment connecting two points in space.

LINE FONT

A pattern for the appearance of a curve. The pattern is a repeating sequence of blanked and unblanked line segments, or of subfigure instances.

LINE FONT DEFINITION ENTITY

A structure entity which defines a line font.

LINE WEIGHT

An entity attribute which is used to determine the line display thickness for that entity.

LINEAR DIMENSION ENTITY

An annotation entity used to represent a distance between two locations.

MACRO BODY

The portion of a macro definition containing statements which define the action of the macro.

MACRO DEFINITION ENTITY

The structure entity, containing the macro body within its parameter data section, used to define a specific macro.

MACRO INSTANCE ENTITY

A structure entity which will invoke a macro which has been defined using a macro definition entity.

MIRROR

To reflect through an axis.

MODEL SPACE

A right-handed three-dimensional Cartesian coordinate space in which the product is represented.

NEGATIVE BOUNDED PLANAR PORTION

A hole.

NODE

A point in space used to define a finite element topology.

ORDINATE DIMENSION ENTITY

An annotation entity used to indicate dimensions from a common reference line in the direction of the XT or YT axis.

ORTHONORMAL

A term describing two vectors which are orthogonal and of unit length.

PARAMETER DATA SECTION

A section of an IGES file consisting of specific geometric or annotative information about the entities or pointers to related entities.

PARAMETRIC SPLINE CURVE ENTITY

A geometric entity consisting of polynomial segments subject to certain continuity conditions.

PARAMETRIC SPLINE SURFACE ENTITY

A geometric entity which is a smooth surface made from a grid of patches. The patches are regions between the component parametric curves.

PARENT CURVE

The full curve on which a segment curve lies.

PATCH

A surface represented by parametric functions of two parameters which blend four given boundary curves.

PLANE ENTITY

A geometric entity which is a surface with the property that the straight line passing through any two distinct points on the surface lies entirely on the surface.

PLATED-THROUGH HOLE (ANSI/IPC-T-50B)

A hole in which electrical connection is made between internal or external conductive patterns, or both, by the deposition of metal on the wall of the hole.

POINT ENTITY

A geometric entity which has no size but possesses a location in space.

POINT DIMENSION ENTITY

An annotation entity consisting of a leader, text, and an optional circle or hexagon enclosing the text.

POINTER

A number that indicates the location of an entity within an IGES file.

POSITIVE BOUNDED PLANAR PORTION

The top of a peg.

POST-PROCESSOR

A program which translates a file of product definition data from the form of this standard into the data base form of a specific CAD/CAM system.

PRE-DEFINED ASSOCIATIVITIES

Associativities which are defined within this standard.

PRE-PROCESSOR

A program which translates a file of product defintiion data from the data base form of a specific CAD/CAM system into the form of this standard.

PRINTED BOARD (ANSI/IPC-T-50B)

The general term for completely processed printed circuit or printed wiring configurations. It includes rigid or flexible, single, double, or multilayer boards.

PRINTED CIRCUIT (ANSI/IPC-T-50B)

A conductive pattern comprised of printed components, printed wiring, or a combination thereof, all formed in a predetermined design and intended to be attached to a common base. (In addition, this is a generic term used to describe a printed board produced by any of a number of techniques.)

PRINTED CIRCUIT BOARD (ANSI/IPC-T-50B)

A part manufactured from rigid base material upon which a completely processed printed circuit has been formed.

PRINTED WIRING (ANSI/IPC-T-50B)

The conductive pattern intended to be formed on a common base, to provide point-to-point connection of discrete components, but not to contain printed components.

PRODUCT DEFINITION

Data required to describe and communicate the characteristics of physical objects as manufactured products.

PROPERTY ENTITY

A structure entity which allows numeric or text information to be related to other entities.

RADIUS DIMENSION ENTITY

An annotation entity which is a measurement of the radius of a circular arc.

RATIONAL B-SPLINE CURVE

A parametric curve which is expressed as the ratio of two linear combinations of B-spline basis functions. Each basis function in the numerator is multiplied by a scalar weight and a vector B-spline coefficient. Each corresponding basis function in the denominator is just multiplied by the weight.

RATIONAL B-SPLINE SURFACE

A parametric surface which is expressed as the ratio of two linear combinations of products of pairs of B-spline basis functions. Each product of basis functions in the numerator is multiplied by a scalar weight and a vector B-spline coefficient. Each corresponding product of basis functions in the denominator is just multiplied by the corresponding weight.

REGION

The bounded area enclosed by a closed curve or a combination of curves.

RELATION

An aspect or quality that connects two or more things or parts as being or belonging or working together or as being of the same kind.

REPEATING PATTERN

An ordered sequence of items (elements) which, after a certain point, repeats itself.

RIGHT-HANDED CARTESIAN COORDINATE SYSTEM

A coordinate system in which the axes are mutually perpendicular and are positioned in such a way that, when viewed along the positive Z axis toward the origin, the positive X axis can be made to coincide with the positive Y axis by rotating the X axis 90 degrees in the counterclockwise direction.

RULED SURFACE ENTITY

A surface generated by connecting corresponding points on two space curves by a set of lines.

SECTION ENTITY

A pattern used to distinguish a closed region in a diagram. It is represented as a form of the copious data entity.

SECTION DISPLAY SYMBOL

An arrangement of fonted straight lines in a repetitive planar pattern at a specified spacing and angle.

SET (IEEE 200-1975)

A unit or units and necessary assemblies, subassemblies, and basic parts connected or associated together to perform an operational function.

SPLINE

A piecewise continuous polynomial interpolation function.

START SECTION

The section of an IGES file containing a man-readable file prolog.

SUBASSEMBLY (IEEE 200-1975)

Two or more basic parts which form a portion of an assembly or a unit, replaceable as a whole, but having a part or parts which are individually replaceable.

SUBFIGURE DEFINITION ENTITY

A structure entity which permits a single definition of a detail to be utilized in multiple instances.

SUBFIGURE INSTANCE ENTITY

A structure entity which specifies an occurrence of the subfigure definition.

SUBORDINATE ENTITY SWITCH

A portion of the status number field of the directory entry of an entity. An entity is subordinate if it is an element of a geometric or annotative entity structure or is a member of a logical relationship structure. The terms subordinate and dependent are equivalent within this document.

SURFACE OF REVOLUTION ENTITY

A geometric entity which is a surface generated by rotating a curve, called the generatrix, about an axis, called the axis of rotation.

SYSTEM (IEEE 200-1975)

A combination of two or more sets, generally physically separated when in operation, and other such units, assemblies, and basic parts necessary to perform an operational function or functions.

TABULATED CYLINDER ENTITY

A geometric entity which is a surface generated by moving a line parallel to itself along a space curve called the generatrix.

TERMINATE SECTION

The final section of an IGES file, indicating the sizes of each of the preceding file sections.

TEXT FONT

The specification of the appearance of the characters.

TEXT FONT DEFINITION ENTITY

The entity used to define the appearance of characters in a text font. A character is defined by pairing its character code with a sequence of display strokes and positional information.

TRANSFORMATION MATRIX ENTITY

An entity which allows translation and rotation to be applied to other entities. This is used to define alternate coordinate systems for the definition and viewing

TRANSLATION VECTOR

A three element vector which specifies the offsets (along the coordinate axes) required to move an entity linearly in space.

UNIT (IEEE 200-1975)

A major building block for a set or system, consisting of a combination of basic parts, subassemblies, and assemblies packaged together as a physically independent entity.

VERSION NUMBER

A means for uniquely designating one specification definition or translator implementation from a preceding or subsequent one.

VIA HOLE (ANSI/IPC-T-50B)

A plated-through hole used as a through connection, but in which there is no intention to insert a component lead or other reinforcing material.

VIEW ENTITY

A structure entity used to provide the definition of a human-readable representation of a two-dimensional projection of a selected subset of the model and/or non-geometry information.

VIEWING BOX

The clipping box used to define a view.

WEIGHT

A non-zero real number which appears in the numerator and denominator of the expression for a rational B-spline curve or surface. Increasing the weight associated with a particular control point will tend to draw the resulting curve or surface toward that control point. See appendices A4 and A6 for details.

WIRE-FRAME

A method of geometric modeling in which a two- or three-dimensional object is represented by curve segments which are edges of the object. In the context of this specification, 'wire-frame' and 'edge-vertex' models are considered as the same technique and the terms are used interchangeably.

WITNESS LINE

An annotation entity consisting of line segments and used in engineering drawings to indicate the beginning or the end of a measurement.

APPENDIX E
INDEX OF TOPICS

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO.	2. Performing Organ. Report No.	3. Publication Date

4. TITLE AND SUBTITLE

Initial Graphics Exchange Specification (IGES) Version 2.0

5. AUTHOR(S)

Bradford Smith, Kalman Brauner, Philip Kennicott, Michael Liewald, Joan Wellington, et al

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

FY1457-82-N5024

8. Type of Report & Period Covered

May 1980 - Nov 1982

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

Air Force ICAM Program, (Integrated Computer Aided Manufacturing),
Wright-Patterson AFB, Ohio 45433

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)*

This document contains Version 2.0 of the Initial Graphics Exchange Specification, a defined format for the creation of a file which enables data found in today's commercially available CAD/CAM systems to be exchanged or archived. IGES, Version 1.0, published as NBSIR 80-1978 (R) in January 1980, consisted of entity definitions for geometry, drafting and structural information. Definition entities were provied as a means of expanding the utility of IGES.

Version 2.0 of the Specification has been extended in the advanced geometry, electrical, and finite element modeling areas. In addition, the Specification has been reformatted and clarified to enable the user to reference the document more easily.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), design drawing, electrical information, exchange format, finite element modeling, geometrics, graphics.

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

15. Price