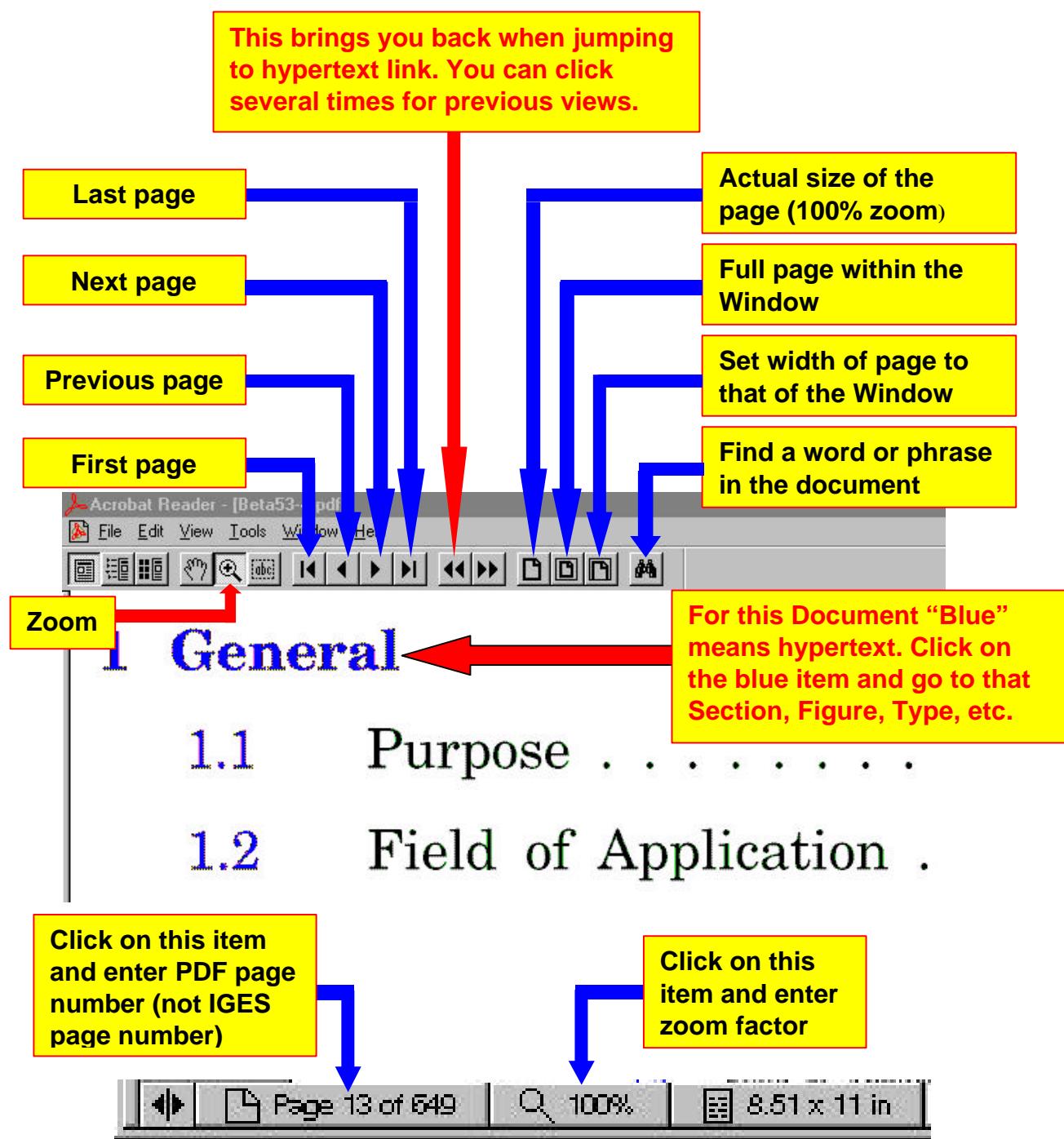




***The U.S. Product Data Association (US PRO) was a nonprofit membership organization established by industry in 1992 and active through December 31, 2006.*** US PRO provided the management functions for the IGES/PDES Organization (IPO) and its related activities. US PRO's primary activities included hosting the ISO/IPO standards development meetings held in the U.S. each year, support for the U.S. TAG as required to maintain accreditation with ANSI, publication and distribution of the standards documents developed and approved as a result of these activities, as well as education and training, marketing, and communications efforts. These activities served both U.S. industry and government agencies by providing a national forum for participation from all interested parties from industry, government, and academia.

The US PRO association helped remove barriers that inhibited the exchange of product data and its flow across the supply chain linked to product design, manufacture, and support. Advancement of Product Data Exchange technology is dramatically improving U.S. and global competitiveness. Participation in US PRO ensured that critical requirements for member businesses and industries were addressed in order to meet their Product Data Exchange needs.

# Quick PDF Toolbar Tutorial



# IGES

*Formerly ANS US PRO/IPO-100-1996*

## Initial Graphics Exchange Specification IGES 5.3



IGES/PDES Organization

---

**US PRO**

---

This standard was developed under procedures accredited as meeting the criteria for American National Standards. The committee that approved the standard was balanced to assure that individuals from competent and concerned interests have had an opportunity to participate. The proposed standard was made available for public review and comment which provided an opportunity for additional public input from industry, academia, regulatory agencies and the public-at-large.

*US PRO makes no warranty of any kind with regard to this document or the procedures or standards specified herein, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose.* While every precaution has been taken in preparation, publication and distribution of this document, US PRO shall not be liable for errors or omissions contained herein or for direct, indirect, special, incidental or consequential damages in connection with the furnishing or use of this document or the procedures or standards described herein.

US PRO does not *approve, rate, or endorse* any item, construction, proprietary device, or activity.

US PRO does not take any position with respect to the validity of any patent rights asserted in connection with any items mentioned in this document, and does not undertake to insure anyone utilizing a standard against liability for infringement of any applicable Letters Patent, nor assume any such liability. Users of a standard are expressly advised that determination of the validity of any such patent rights, and the risk of infringement of such rights, is entirely their own responsibility.

Participation by federal agency representative(s) or person(s) affiliated with industry is not to be interpreted as government or industry endorsement of this standard.

**Published by**  
**U.S. Product Data Association**  
**Trident Research Center, Suite 204, 5300 International Blvd., N. Charleston, SC 29418**

Copyright 1996 by the IGES/PDES Organization  
All Rights Reserved  
Printed in the United States of America

---

---

**US PRO**

---

U.S. Product Data Association  
Trident Research Center, Suite 204  
5300 International Blvd.  
N. Charleston, SC 29418

---

---

---

**An American National Standard from 1996 through 2006**

---

---

# **Initial Graphics Exchange Specification**

## **Formerly ANS US PRO/IPO-100-1996**

---

---

**IGES Technical Editor**

**P. R. Kennicott**  
Sandia National Laboratories  
Albuquerque, NM

**IGES Project Manager**

**Gregory Morea**  
General Dynamics  
Electric Boat Division  
Groton, CT

**IGES Deputy Project Manager**

**Ed Reid**  
Caterpillar Inc.  
Peoria, IL

**IGES Change Control Secretary**

**Curtis Parks**  
National Institute of Standards and Technology  
Gaithersburg, MD

**IGES Configuration Manager**

**Gaylen Rinaudot**  
National Institute of Standards and Technology  
Gaithersburg, MD

**IGES Figure Editor**

**Dennette A. Harrod Jr.**  
WizWorx  
Concord, MA

**Chairman,  
IGES/PDES Organization**

**William B. Gruttke**  
National Institute of Standards and Technology  
Gaithersburg, MD

**U.S. Product Data Association**  
Trident Research Center, Suite 204  
N. Charleston, SC 29418  
September 23, 1996

**Officers and Committees of the IGES/PDES Organization**  
February 8, 1995

**Officers**

Chair	Bill Gruttke
Deputy Chair	Mary Mitchell
IGES Project Manager	Greg Morea
Deputy IGES Project Manager	Ed Reid
PDES Project Manager	Haidee Halvorson
Testing Project Manager	Gary Conkol
Deputy Testing Project Manager	Alan Peltzman
U.S. TAG to ISO/TC184/SC4	Dick Justice

**Associated Staff**

Executive Assistant	Ellen Trager
Secretary	Cremona Randall
Administrative Coordinator, NCGA	Nancy Flower
IGES Technical Editor	Philip Kennicott
IGES Change Control Secretary	Curt Parks
IGES Configuration Manager	Gaylen Rinaudot
IGES Ballot Coordinator	Ellen Trager
IPO Communications Program	Dave Mattei
IPO Education Program	Dave Sanford
IPO Editor	Joan Wellington

**Special Interest Groups**

CALS/IGES	Lisa Deeds
CALS/PDES	Ben Kassel, deputy
Configuration Management	Wey Chang
Process Plant	Haidee Halvorson
Software Products	Mark Palmer
	Robert M. Wessely (acting)

**Steering Committee**

Chair	Frank Tidaback
Vice Chair	Dick Wandmacher
Secretary	Dick Justice
Treasurer	S. Greg Hugh

## **Technical Committee Chairs**

Application Protocol Validation Methodology	Joel Peterson
Architecture, Engineering & Construction	Ben Kassell, deputy
Composites	Burt Gischner, deputy
Conformance & Verification Testing Methodologies	Glen Ziolk
Drafting	Tom Phelps
Electrical Applications	Linas Polikaitis (STEP APs)
Finite Element Analysis	Ed Reid, deputy (IGES)
Geometry	Curt Parks, deputy (IGES) (acting)
Implementation Specifications	Keith Hunten
Implementations	Ed Clapp
Integration	Noel Christensen, deputy
Interoperability Accept. Testing Methodology	Dave Price (acting)
Manufacturing Technology	Bill Turcotte co-chair
Materials	George Baker co-chair
Mechanical Product Definition	Yuhwei Yang
PDES Development Methods	George Elwood
Product Structure and Life Cycle Support	Gary Conkol, deputy
Qualification & Validation	Greg Paul
Sheet Metal	Larry Parker, deputy
Standard Parts	Joe Carpenter
Technical Publications	Bill Cain
	Phil Kennicott co-chair
	Rick Bsharah co-chair
	Chuck Amaral co-chair
	Shirley Goodman, deputy
	Pete Lazo
	Mike Strub, deputy
	Patrick Rourke, deputy
	Yuri Rubinsky (acting)

## **Members of the IGES/PDES Organization**

### **Members of the IGES/PDES Organization**

Amaral, Chuck	Rockwell International Corp
Anderson, Bill D	SCRA
Anderson, John R	US Army Research Laboratory
Baker, George W	International TechneGroup Inc
Barnard Feeney, Allison	NIST
Basham, Dave	Boeing Co
Benigni, Daniel R	NIST CSL
Borchert, Cliff	Eastman Kodak Co
Bradley, Jeff	CACI Inc
Bringuel, Martin	Hewlett Packard Co
Brown, David	Auto-trol Technology Corp
Brubaker, Craig	Newport News Shipbuilding
Bsharah, Rick	Rockwell Aerospace NAAD
Cain, William D	Martin Marietta Energy Systems
Carpenter, Joseph A	NIST
Cederman, Charles	General Motors
Chang, Wey Tyng	Accurate Information Systems
Christensen, Noel C	Allied Signal Inc
Clapp, Edward	Autodesk Inc
Clark, Stephen	NIST
Collins, Michael F	Control Data Systems Inc
Conkol, Gary K	CAMP ECRC
Conroy, Bill	NIST/Electronic Data Systems
Cook, Jeff	Texas Instruments Inc
Corley, Jack	SCRA
Craig, Diane L	PDIT
Crusey, Jesse L	NIST
Danner, William F	NIST
Deeds, Lisa V	David Taylor Research Center
Demasek, Frank W	General Motors
DePauw, J Spencer	Caterpillar Inc
Durnin, Marc W	Lockheed Aeronautical Systems Co
Elwood, George	Air Force CALS Testbed
Fleig, Gottlieb	NIST/NIPDE
Fletcher, Rob	RLF Enterprises
Freeman Jr, William B	SCRA
Freund, Kevin	D Appleton Co Inc
Ganus, Floyd O	Vought Aircraft Co
Gilbert, Mitch	Grumman Data Systems
Goodman, Shirley A	Naval Surface Warfare Center
Goosen, Ted	Lockheed Fort Worth Co / PDES Inc
Graves, Gerald R	SCRA
Grout, J S (Steve)	Sematech
Gruttke, William B	Northrop Grumman Corp

## **Members of the IGES/PDES Organization**

Halvorson, Haidee	Auto-trol Technology Corp
Hardwick, Martin	RPI/STEP Tools Inc
Harrod, Dennette	Wiz Worx
Hazzard, Lon R	Grumman Data Systems
Heiser, John E	JEH Consulting
Heller, Mitchell H	Raytheon Co
Hodges, John	International TechneGroup Inc
Home, Lance	NIST
Humphrey, Chris	NIST
Hunten, Keith	Lockheed Fort Worth Co
Johnson, Thomas G	UTC/Pratt & Whitney
Jones, Alan K	Boeing Co
Justice, Dick	AIAG
Karns, Larry	Arthur D Little Inc
Kassel, Ben	CD NSWC/NIDDESC
Kennicott, Philip R	Sandia National Laboratories
Kiggans, Robert	SCRA
Kipp Jr, Thomas E	MacNeal-Schwendler Corp
Koopman, Michael	Concurrent Technologies Corp
Kramer, Thomas R	NIST
Lampkin, Mark	Watervliet Arsenal
Laurance, Neal	Ford Motor Co
Laze, Peter L	Newport News Shipbuilding
Lim, Brenda	Modern Engineering
Marians, Carol	Rosetta Technology Inc
Mathew, Abraham	Intergraph Corp
Mattei, David	International TechneGroup Inc
Mawhirter, Jeff	International TechneGroup Inc
McKee, Larry	IBM/SCRA
Mitchell, Mary J	NIST
Moor, Paul	CAMAX Manufacturing Technologies Inc
Morea, Greg	General Dynamics
Nell, James G	NIST
Nelson, Paul A	Hughes Aircraft Co/GM
Nicholson, Martha B	Arthur D Little Inc
Noel, R Hank	NIPDE/NCMS
OConnell, Larry	Sandia National Laboratories
Orogo, Constantine D	Concurrent Technologies Corp
Palmer, Mark	NIST
Parker, Lawrence O	GM Hughes Electronics
Parks, Curtis H	NIST
Parks, Robert E	Sandia National Laboratories
Paul, Greg A	Lockheed Fort Worth Co
Peltzman, Alan	DISA Center for Standards
Petersen, Joel S	Autodesk Inc
Peterson, Brian C	CAMAX Manufacturing Technologies Inc
Phelps, Thomas A	Industrial Technology Institute
Phillips, Lisa	NIST
Polikaitis, Linas	Northrop Grumman Corp
Preston, Joseph	Dimensions International
Price, David M	IBM

## **Members of the IGES/PDES Organization**

Quintero, Andrew	Aerospace Corp
Rando, Tom	General Dynamics
Rathman, Andy	Auto-trol Technology Corp
Reed, Kent A	NIST
Reid, Ed	Caterpillar Inc
Rinaudot, Gaylen R	NIST
Robinson, Virgil L	Boeing Co
Rourke, Patrick W	Newport News Shipbuilding
Rubinsky, Yuri	SoftQuad Inc
Ruys, Ronald	NIST
Ryan, Steven A	GE Aircraft Engines
Sanford, David T	Boeing Computer Services
Sauder, Dave	NIST
Shab, Ted	NIST/NIPDE
Shaw, Nigel K	ProSTEP Productdatentechnologie GmbH
Shih, Chia-Hui	SDRC
Sieveking, Terry W	McDonnell Douglas Corp
Silvernale, Gerard J	International TechneGroup Inc
Sinha, Ashwini	McDonnell Douglas
Skeels, Jack A	Product Data Integration Technologies
Slovensky, Len	Northrop Grumman Corp
Smith, Bradford	NIST
Stark, Chuck	SCRA
Stratton, Keith	IBM
Strub, Michael C	General Motors
Szmrecsanyi, Emery	Chrysler Corp
Tanguay, Ken	Accurate Information Systems
Thiel, Bruce	International TechneGroup Inc
Thurman, Thomas R	Rockwell Defense Electronics
Tidaback, Frank W	Caterpillar Inc
Tocco, Mark A	Ford Motor Co
Trager, Ellen	NIST
Tucker, Hugh	Documents Aps
Turcotte, William	IGES Data Analysis Inc
Washington, Deborah	Watervliet Arsenal
Waterbury, Stephen C	NASA/GSFC
Weaver, Earl P	US Army Research Laboratory
Wellington, Joan	NIST
Wessely, Robert M	SciSo Inc
Wilson, Alan	Computervision Inc
Wilson, Peter	NIST
Wooley, Dan	Newport News Shipbuilding
Yang, Yuhwei	Product Data Integration Technologies
Zimmerman, John	Allied Signal Inc
Ziolko, Glen A	SCRA

## **Foreword**

ECO630

This version of the Initial Graphics Exchange Specification (IGES) continues the practice of publishing IGES as a fully accredited American National Standards Institute (ANSI) Standard. This is done through the auspices of the US Product Data Association (US PRO). As with Version 5.2, this version is copyrighted by the IGES PDES Organization (IPO).

Version 5.3 appears approximately two years after the publication of IGES 5.2. It is still considered a point version since the amount of new material does not constitute the release of a new, full number, version. This version is only being published as a complete document. Changes are tracked via numbered Edit Change Orders (ECOS); each changed area has a margin note "ECOnnn" for identification.

This version brings a new look, and several new capabilities to the IGES user. The introductory material, Chapters 1 and 2, has been completely re-written to make it more readable and succinct. Gray page material, formerly found in [Appendix G](#), is now integrated into the main body of the document. Untested entities now have an "UNTESTED" banner (‡) to denote them.

Regarding new capabilities, a formal link is now available between external documents, such as military specifications and application protocols, and the IGES Global Section. Also, BREP solids may now be used as CSG primitives, infinite lines may now be exchanged, and several new properties have been added. As with previous versions, numerous clarifications and editorial corrections are also made.

Please note that the version number assigned by the IPO will differ from the official ANSI designation. The IPO number can still be used for informal tracking, although the ANSI number should be used for official matters.

Those Edit Change Orders included in this version of the Specification are:

<b>ECO</b>	<b>Section</b>	<b>Title</b>
ECO622	4.60	Remove Font Code 2001 from Untested Status
ECO625	C	Clarification to <a href="#">Appendix C</a> (Conic Arc)
ECO626	4.60	Clarify General Note slant angle
ECO627	4.147	Add an Open form to the Shell entity
ECO628	4.62	Arrowhead illustration
ECO629	4.60	Rotate Internal text clarification
ECO630	multiple	Version 5.3 master for editorial corrections
ECO631	4.128	Create Drawing Sheet Approval property
ECO632	4.129	Create Drawing Sheet ID property
ECO634	4.66	Remove Radius Dimension, Form 1 from Untested Status
ECO635	multiple	Clarify Dimension Planarity
ECO636	4.96	Proper scale factor usage in the View entity
ECO637	3.6	Clarify subfigure processing in Sect. 3.6
ECO638	2.2.4.3	Global dates beyond the year 1999
ECO639	4.17	Ruled Surface Parameterization
ECO640	4.19	TabCyl: parameterization and constituents
ECO642	4.4	Use of a Composite Curve as a Logical Connection
ECO643	2.2.4.3	Add a Mil-Spec Code to the Globals
ECO644	multiple	BREP Objects as CSG Primitives
ECO645	4.66	Add Attributes for Geometry of Cataloged Parts
ECO646	4.13	Add Unbounded Lines
ECO647	multiple	Add Functional Underscoring/OverScoring
ECO648	4.132	Create Closure Property ( <a href="#">Type 406, Form 36</a> )
ECO649	multiple	Change Electrical Terminology to LEP
ECO650	multiple	Revise Parameter Data Indices
ECO651	4.66	Add Electronic Attribute Values
ECO652	multiple	Boundary and Bounded Surface Clarification
ECO653	2	New Chapter 2
ECO655	4.120	Extension to LEP Layer Map Property ( <a href="#">Type 406, Form 24</a> )
ECO656	4.92	Change to Flow Associativity Entity ( <a href="#">Type 402, Form 18</a> )
ECO658	1	New Chapter 1
ECO659	multiple	Move untested entity descriptions to body of specification

## **Contents**

<b>Members of the IGES/PDES Organization</b>	<b>iv</b>
<b>Foreword</b>	<b>vii</b>
<b>1 General</b>	<b>1</b>
1.1 Purpose . . . . .	1
1.2 Field of Application . . . . .	1
1.3 Concepts of Product Definition . . . . .	1
1.4 Conformance to the Specification . . . . .	3
1.4.1 Background . . . . .	3
1.4.2 Documentation requirements. . . . .	3
1.4.3 Conformance rules . . . . .	3
1.4.4 Conformance rules for exchange files. . . . .	4
1.4.4.1 Unprocessable entities . . . . .	4
1.4.5 Conformance rules for preprocessors. . . . .	4
1.4.6 Conformance rules for postprocessors. . . . .	5
1.4.7 Conformance rules for editor, analyzer or viewer tools. . . . .	5
1.4.7.1 Functional requirements for editors and analyzers . . . . .	5
1.4.7.2 Functional requirements for browsers. . . . .	6
1.4.7.3 Functional requirements for viewers. . . . .	6
1.5 Concepts of the File Structure . . . . .	6
1.6 Concepts of Information Structures for Product Models . . . . .	7
1.6.1 Property Entity . . . . .	7
1.6.2 Associativity Entity . . . . .	7
1.6.3 View Entity . . . . .	7
1.6.4 Drawing Entity . . . . .	7
1.6.5 Transformation Matrix Entity . . . . .	8
1.6.6 Implementor-defined Entities . . . . .	8

## CONTENTS

1.7	Appendices . . . . .	8
1.8	Illustrations . . . . .	8
1.9	Untested Entities . . . . .	8
<b>2</b>	<b>Data Form</b>	<b>9</b>
2.1	General . . . . .	9
2.2	ASCII File Formats . . . . .	9
2.2.1	Field Categories and Defaulting. . . . .	10
2.2.2	Data types . . . . .	12
2.2.2.1	Integer data type. . . . .	12
2.2.2.2	Real data type. . . . .	13
2.2.2.3	String data type. . . . .	13
2.2.2.4	Pointer data type. . . . .	14
2.2.2.5	Language Statement data type. . . . .	14
2.2.2.6	Logical data type. . . . .	14
2.2.3	Rules for Forming and Interpreting Free Formatted Data . . . . .	14
2.2.3.1	Parameter and Record Delimiter Combinations. . . . .	15
2.2.4	File Structure . . . . .	15
2.2.4.1	Flag Section . . . . .	16
2.2.4.2	Start Section . . . . .	16
2.2.4.3	Global Section . . . . .	16
2.2.4.4	Directory Entry Section. . . . .	23
2.2.4.5	Parameter Data Section. . . . .	32
2.2.4.6	Terminate Section . . . . .	34
2.3	Compressed Format . . . . .	35
2.3.1	File Structure . . . . .	35
<b>3</b>	<b>Classes of Entities</b>	<b>37</b>
3.1	General . . . . .	37
3.2	Curve and Surface Geometry Entities . . . . .	37
3.2.1	Entity Types . . . . .	37
3.2.2	Coordinate Systems . . . . .	37
3.2.3	Multiple Transformation Entities. . . . .	39
3.2.4	Directionality . . . . .	39
3.2.5	Continuity and Non-degeneracy. . . . .	41

<b>3.3</b>	<b>Constructive Solid Geometry Entities . . . . .</b>	<b>41</b>
<b>3.3.1</b>	<b>Entity Types . . . . .</b>	<b>41</b>
<b>3.3.2</b>	<b>Constructive Solid Geometry Models. . . . .</b>	<b>42</b>
<b>3.4</b>	<b>Boundary Representation Solid Entities . . . . .</b>	<b>43</b>
<b>3.4.1</b>	<b>Entity Types . . . . .</b>	<b>43</b>
<b>3.4.2</b>	<b>Topology for B-Rep Solid Models. . . . .</b>	<b>44</b>
<b>3.4.3</b>	<b>Analytical Surfaces for B-Rep Solid Models. . . . .</b>	<b>45</b>
<b>3.4.3.1</b>	<b>Entity Types . . . . .</b>	<b>45</b>
<b>3.4.3.2</b>	<b>Parameterization of Analytical Surfaces. . . . .</b>	<b>45</b>
<b>3.5</b>	<b>Annotation Entities . . . . .</b>	<b>46</b>
<b>3.5.1</b>	<b>Entity Types . . . . .</b>	<b>46</b>
<b>3.5.2</b>	<b>Construction . . . . .</b>	<b>46</b>
<b>3.5.3</b>	<b>Definition Space . . . . .</b>	<b>46</b>
<b>3.5.4</b>	<b>Dimension Attributes . . . . .</b>	<b>47</b>
<b>3.5.4.1</b>	<b>General . . . . .</b>	<b>47</b>
<b>3.5.4.2</b>	<b>Usage Rules . . . . .</b>	<b>48</b>
<b>3.6</b>	<b>Structure Entities . . . . .</b>	<b>50</b>
<b>3.6.1</b>	<b>Entity Types . . . . .</b>	<b>50</b>
<b>3.6.2</b>	<b>Subfigures . . . . .</b>	<b>50</b>
<b>3.6.3</b>	<b>Connectivity . . . . .</b>	<b>51</b>
<b>3.6.3.1</b>	<b>Connectivity Entities . . . . .</b>	<b>53</b>
<b>3.6.3.2</b>	<b>Entity Relationships . . . . .</b>	<b>53</b>
<b>3.6.3.3</b>	<b>Information Display . . . . .</b>	<b>53</b>
<b>3.6.3.4</b>	<b>Additional Considerations. . . . .</b>	<b>53</b>
<b>3.6.4</b>	<b>External Reference Linkage . . . . .</b>	<b>53</b>
<b>3.6.5</b>	<b>Drawings and Views . . . . .</b>	<b>55</b>
<b>3.6.6</b>	<b>Finite-Element Modeling . . . . .</b>	<b>57</b>
<b>3.6.7</b>	<b>Attribute Tables . . . . .</b>	<b>60</b>
<b>4 Entity Types</b>		<b>61</b>
<b>4.1</b>	<b>General . . . . .</b>	<b>61</b>
<b>4.2</b>	<b>Null Entity (Type 0) . . . . .</b>	<b>63</b>
<b>4.3</b>	<b>Circular Arc Entity (Type 100) . . . . .</b>	<b>64</b>
<b>4.4</b>	<b>Composite Curve Entity (Type 102). . . . .</b>	<b>67</b>
<b>4.5</b>	<b>Conic Arc Entity (Type 104) . . . . .</b>	<b>71</b>

## CONTENTS

<a href="#">4.6</a>	Copious Data Entity (Type 106, Forms 1-3) . . . . .	75
<a href="#">4.7</a>	Linear Path Entity (Type 106, Forms 11-13) . . . . .	77
<a href="#">4.8</a>	Centerline Entity (Type 106, Forms 20-21) . . . . .	79
<a href="#">4.9</a>	Section Entity (Type 106, Forms 31-38). . . . .	81
<a href="#">4.10</a>	Witness Line Entity (Type 106, Form 40) . . . . .	84
<a href="#">4.11</a>	Simple Closed Planar Curve Entity (Type 106, Form 63)	86
<a href="#">4.12</a>	Plane Entity (Type 108) . . . . .	87
<a href="#">4.13</a>	Line Entity (Type 110, Form 0). . . . .	90
<a href="#">4.14</a>	Parametric Spline Curve Entity (Type 112) . . . . .	94
<a href="#">4.15</a>	Parametric Spline Surface Entity (Type 114) . . . . .	98
<a href="#">4.16</a>	Point Entity (Type 116) . . . . .	102
<a href="#">4.17</a>	Ruled Surface Entity (Type 118) . . . . .	104
<a href="#">4.18</a>	Surface of Revolution Entity (Type 120). . . . .	107
<a href="#">4.19</a>	Tabulated Cylinder Entity (Type 122).. . . . .	110
<a href="#">4.20</a>	Direction Entity (Type 123) ‡ . . . . .	112
<a href="#">4.21</a>	Transformation Matrix Entity (Type 124) . . . . .	113
<a href="#">4.22</a>	Flash Entity (Type 125) . . . . .	120
<a href="#">4.23</a>	Rational B-Spline Curve Entity (Type 126) . . . . .	123
<a href="#">4.24</a>	Rational B-Spline Surface Entity (Type 128) . . . . .	126
<a href="#">4.25</a>	Offset Curve Entity (Type 130) . . . . .	129
<a href="#">4.26</a>	Connect Point Entity (Type 132) . . . . .	131
<a href="#">4.27</a>	Node Entity (Type 134) . . . . .	134
<a href="#">4.28</a>	Finite Element Entity (Type 136) . . . . .	137
<a href="#">4.29</a>	Nodal Displacement and Rotation Entity (Type 138) . . . . .	153
<a href="#">4.30</a>	Offset Surface Entity (Type 140)..... . . . . .	155
<a href="#">4.31</a>	Boundary Entity (Type 141) . . . . .	157
<a href="#">4.32</a>	Curve on a Parametric Surface Entity (Type 142) . . . . .	162
<a href="#">4.33</a>	Bounded Surface Entity (Type 143) . . . . .	165
<a href="#">4.34</a>	Trimmed (Parametric) Surface Entity (Type 144) . . . . .	166
<a href="#">4.35</a>	Nodal Results Entity (Type 146) ‡. . . . .	168
<a href="#">4.36</a>	Element Results Entity (Type 148) ‡. . . . .	171
<a href="#">4.37</a>	Block Entity (Type 150) . . . . .	174
<a href="#">4.38</a>	Right Angular Wedge Entity (Type 152) . . . . .	176
<a href="#">4.39</a>	Right Circular Cylinder Entity (Type 154) . . . . .	178

<b>4.40</b>	Right Circular Cone Frustum Entity (Type 156) . . . . .	180
<b>4.41</b>	Sphere Entity (Type 158) . . . . .	182
<b>4.42</b>	Torus Entity (Type 160) . . . . .	184
<b>4.43</b>	Solid of Revolution Entity (Type 162) . . . . .	186
<b>4.44</b>	Solid of Linear Extrusion Entity (Type 164) . . . . .	188
<b>4.45</b>	Ellipsoid Entity (Type 168) . . . . .	190
<b>4.46</b>	Boolean Tree Entity (Type 180) . . . . .	192
<b>4.47</b>	Selected Component Entity (Type 182) ‡ . . . . .	195
<b>4.48</b>	Solid Assembly Entity (Type 184) . . . . .	196
<b>4.49</b>	Manifold Solid B-Rep Object Entity (Type 186) ‡ . . . . .	197
<b>4.50</b>	Plane Surface Entity (Type 190) ‡ . . . . .	203
<b>4.51</b>	Right Circular Cylindrical Surface Entity (Type 192) ‡ . . . . .	206
<b>4.52</b>	Right Circular Conical Surface Entity (Type 194) ‡ . . . . .	209
<b>4.53</b>	Spherical Surface Entity (Type 196) ‡ . . . . .	212
<b>4.54</b>	Toroidal Surface Entity (Type 198) ‡ . . . . .	215
<b>4.55</b>	Angular Dimension Entity (Type 202) . . . . .	218
<b>4.56</b>	Curve Dimension Entity (Type 204) ‡ . . . . .	220
<b>4.57</b>	Diameter Dimension Entity (Type 206) . . . . .	222
<b>4.58</b>	Flag Note Entity (Type 208) . . . . .	224
<b>4.59</b>	General Label Entity (Type 210) . . . . .	227
<b>4.60</b>	General Note Entity (Type 212) . . . . .	229
<b>4.61</b>	New General Note Entity (Type 213) ‡ . . . . .	246
<b>4.61.1</b>	Parameter Field Descriptions . . . . .	246
<b>4.61.2</b>	Control Codes . . . . .	248
<b>4.61.2.1</b>	Control Codes Which Cannot Be Nested . . . . .	248
<b>4.61.2.2</b>	Control Codes Which Can Be Nested . . . . .	249
<b>4.62</b>	Leader (Arrow) Entity (Type 214) . . . . .	257
<b>4.63</b>	Linear Dimension Entity (Type 216) . . . . .	262
<b>4.64</b>	Ordinate Dimension Entity (Type 218) . . . . .	264
<b>4.65</b>	Point Dimension Entity (Type 220) . . . . .	267
<b>4.66</b>	Radius Dimension Entity (Type 222) . . . . .	269
<b>4.67</b>	General Symbol Entity (Type 228) . . . . .	272
<b>4.68</b>	Sectioned Area Entity (Type 230) . . . . .	275
<b>4.69</b>	Associativity Definition Entity (Type 302) . . . . .	289

## CONTENTS

<b>4.70</b>	Line Font Definition Entity (Type 304) . . . . .	292
<b>4.71</b>	MACRO Definition Entity (Type 306) ‡ . . . . .	298
<b>4.71.1</b>	General . . . . .	298
<b>4.71.2</b>	MACRO Syntax . . . . .	298
<b>4.71.2.1</b>	Constants. . . . .	298
<b>4.71.2.2</b>	Variables. . . . .	298
<b>4.71.2.3</b>	Functions. . . . .	299
<b>4.71.2.4</b>	Expressions. . . . .	301
<b>4.71.3</b>	Language Statements. . . . .	302
<b>4.71.3.1</b>	LET Statement . . . . .	302
<b>4.71.3.2</b>	SET Statement . . . . .	305
<b>4.71.3.3</b>	REPEAT Statement . . . . .	306
<b>4.71.3.4</b>	CONTINUE Statement . . . . .	307
<b>4.71.3.5</b>	BREAK Statement . . . . .	307
<b>4.71.3.6</b>	IF Statement . . . . .	307
<b>4.71.3.7</b>	LABEL Statement . . . . .	308
<b>4.71.3.8</b>	GOTO Statement . . . . .	308
<b>4.71.3.9</b>	MACRO Statement . . . . .	308
<b>4.71.3.10</b>	ENDM Statement . . . . .	310
<b>4.71.4</b>	The MACRO Definition Entity . . . . .	310
<b>4.72</b>	MACRO Instance Entity ‡ . . . . .	312
<b>4.72.1</b>	Example 1: Isosceles Triangle . . . . .	313
<b>4.72.2</b>	Example 2: Repeated parallelograms . . . . .	315
<b>4.72.3</b>	Example 3: Concentric circles . . . . .	317
<b>4.72.4</b>	Example 4: Electrical ground symbol . . . . .	319
<b>4.72.5</b>	Example 5: Useful features . . . . .	321
<b>4.73</b>	Subfigure Definition Entity (Type 308) . . . . .	322
<b>4.74</b>	Text Font Definition Entity (Type 310) . . . . .	323
<b>4.75</b>	Text Display Template Entity (Type 312) . . . . .	327
<b>4.76</b>	Color Definition Entity (Type 314) . . . . .	329
<b>4.77</b>	Units Data Entity (Type 316) ‡ . . . . .	330
<b>4.78</b>	Network Subfigure Definition Entity (Type 320) . . . . .	333
<b>4.79</b>	Attribute Table Definition Entity (Type 322) . . . . .	335
<b>4.80</b>	Associativity Instance Entity (Type 402) . . . . .	374

<b>4.80.1</b>	<b>Predefined Associativities . . . . .</b>	<b>374</b>
<b>4.81</b>	<b>Group Associativity (Type 402, Form 1) . . . . .</b>	<b>376</b>
<b>4.82</b>	<b>Views Visible Associativity (Type 402, Form 3) . . . . .</b>	<b>377</b>
<b>4.83</b>	<b>Views Visible, Color, Line Weight Associativity (Form 4) . . . . .</b>	<b>379</b>
<b>4.84</b>	<b>Entity Label Display Associativity (Type 402, Form 5) . . . . .</b>	<b>381</b>
<b>4.85</b>	<b>Group Without Back Pointers Associativity (Form 7) . . . . .</b>	<b>383</b>
<b>4.86</b>	<b>Single Parent Associativity (Type 402, Form 9) . . . . .</b>	<b>384</b>
<b>4.87</b>	<b>External Reference File Index Associativity (Type 402, Form 12) . . . . .</b>	<b>385</b>
<b>4.88</b>	<b>Diminished Geometry Associativity (Type 402, Form 13) . . . . .</b>	<b>386</b>
<b>4.89</b>	<b>Ordered Group with Back Pointers Associativity (Type 402, Form 14) . . . . .</b>	<b>389</b>
<b>4.90</b>	<b>Ordered Group, no Back Pointers Associativity (Type 402, Form 15) . . . . .</b>	<b>390</b>
<b>4.91</b>	<b>Planar Associativity (Type 402, Form 16) . . . . .</b>	<b>391</b>
<b>4.92</b>	<b>Flow Associativity (Type 402, Form 18) . . . . .</b>	<b>393</b>
<b>4.93</b>	<b>Segmented Views Visible Associativity (Type 402, Form 19) ‡ . . . . .</b>	<b>397</b>
<b>4.94</b>	<b>Piping Flow Associativity (Type 402, Form 20) ‡ . . . . .</b>	<b>399</b>
<b>4.95</b>	<b>Diminished Geometry Associativity (Type 402, Form 21) ‡ . . . . .</b>	<b>403</b>
<b>4.96</b>	<b>Drawing Entity (Type 404) . . . . .</b>	<b>409</b>
<b>4.97</b>	<b>Property Entity (Type 406) . . . . .</b>	<b>415</b>
<b>4.98</b>	<b>Definition Levels Property (Form 1) . . . . .</b>	<b>416</b>
<b>4.99</b>	<b>Region Restriction Property (Form 2) . . . . .</b>	<b>417</b>
<b>4.100</b>	<b>Level Function Property (Form 3) . . . . .</b>	<b>419</b>
<b>4.101</b>	<b>Line Widening Property (Form 5) . . . . .</b>	<b>420</b>
<b>4.102</b>	<b>Drilled Hole Property (Form 6) . . . . .</b>	<b>422</b>
<b>4.103</b>	<b>Reference Designator Property (Form 7) . . . . .</b>	<b>423</b>
<b>4.104</b>	<b>Pin Number Property (Form 8) . . . . .</b>	<b>424</b>
<b>4.105</b>	<b>Part Number Property (Form 9) . . . . .</b>	<b>425</b>
<b>4.106</b>	<b>Hierarchy Property (Form 10) . . . . .</b>	<b>426</b>
<b>4.107</b>	<b>Tabular Data Property (Form 11) . . . . .</b>	<b>427</b>
<b>4.108</b>	<b>External Reference File List Property (Form 12) . . . . .</b>	<b>445</b>
<b>4.109</b>	<b>Nominal Size Property (Form 13) . . . . .</b>	<b>446</b>
<b>4.110</b>	<b>Flow Line Specification Property (Form 14) . . . . .</b>	<b>447</b>
<b>4.111</b>	<b>Name Property (Form 15) . . . . .</b>	<b>448</b>
<b>4.112</b>	<b>Drawing Size Property (Form 16) . . . . .</b>	<b>449</b>
<b>4.113</b>	<b>Drawing Units Property (Form 17) . . . . .</b>	<b>450</b>

4.114	Intercharacter Spacing Property (Form 18) ‡ . . . . .	451
4.115	Line Font Property (Form 19) ‡. . . . .	452
4.116	Highlight Property (Form 20) ‡ . . . . .	457
4.117	Pick Property (Form 21) ‡ . . . . .	458
4.118	Uniform Rectangular Grid Property (Form 22) ‡ . . . . .	459
4.119	Associativity Group Type Property (Form 23) ‡ . . . . .	460
4.120	Level to LEP Layer Map Property (Form 24) ‡ . . . . .	462
4.121	LEP Artwork Stackup Property (Form 25) ‡ . . . . .	465
4.122	LEP Drilled Hole Property (Form 26) ‡ . . . . .	466
4.123	Generic Data Property (Form 27) ‡ . . . . .	468
4.124	Dimension Units Property (Form 28) ‡ . . . . .	470
4.125	Dimension Tolerance Property (Form 29) ‡ . . . . .	472
4.126	Dimension Display Data Property (Form 30) ‡ . . . . .	475
4.127	Basic Dimension Property (Form 31) ‡ . . . . .	478
4.128	Drawing Sheet Approval Property (Type 406, Form 32) ‡ . . . . .	480
4.129	Drawing Sheet ID Property (Type 406, Form 33) ‡ . . . . .	481
4.130	Underscore Property (Type 406, Form 34) ‡ . . . . .	482
4.131	Overscore Property (Type 406, Form 35) ‡ . . . . .	483
4.132	Closure Property (Type 406, Form 36) ‡. . . . .	485
4.133	Singular Subfigure Instance Entity (Type 408) . . . . .	488
4.134	View Entity (Type 410) . . . . .	490
4.135	Perspective View Entity (Type 410, Form 1) ‡. . . . .	496
4.136	Rectangular Array Subfigure Instance Entity (Type 412) . . . . .	499
4.137	Circular Array Subfigure Instance Entity (Type 414) . . . . .	501
4.138	External Reference Entity (Type 416) . . . . .	503
4.139	Nodal Load/Constraint Entity (Type 418) . . . . .	506
4.140	Network Subfigure Instance Entity (Type 420) . . . . .	508
4.141	Attribute Table Instance Entity (Type 422) . . . . .	510
4.141.1	Attribute Table Instance (Form 0). . . . .	510
4.141.2	Attribute Table Instance (Form 1). . . . .	511
4.142	Solid Instance Entity (Type 430). . . . .	512
4.143	Vertex Entity (Type 502) ‡. . . . .	514
4.143.1	Vertex List Entity (Type 502, Form 1) . . . . .	514
4.144	Edge Entity (Type 504) ‡ . . . . .	516

## CONTENTS

<b>4.144.1</b> Edge List Entity (Type 504, Form 1) . . . . .	<b>516</b>
<b>4.145</b> Loop Entity (Type 508) ‡ . . . . .	<b>518</b>
<b>4.146</b> Face Entity (Type 510) ‡. . . . .	<b>520</b>
<b>4.147</b> Shell Entity (Type 514) ‡ . . . . .	<b>522</b>
 <b>A Part File Examples</b>	 <b>524</b>
<b>A.1</b> Electrical Part Example . . . . .	<b>526</b>
<b>A.2</b> Mechanical Part Example . . . . .	<b>529</b>
<b>A.3</b> Drawing and View Example . . . . .	<b>537</b>
 <b>B Spline Curves and Surfaces</b>	 <b>545</b>
<b>B.1</b> Introduction . . . . .	<b>545</b>
<b>B.2</b> Spline Functions . . . . .	<b>545</b>
<b>B.3</b> Spline Curves . . . . .	<b>546</b>
<b>B.4</b> Rational B-Spline Curves . . . . .	<b>547</b>
<b>B.5</b> Spline Surfaces . . . . .	<b>548</b>
<b>B.6</b> Rational B-spline Surfaces . . . . .	<b>549</b>
 <b>C Conic Arcs</b>	 <b>551</b>
 <b>D Color-Space Mappings</b>	 <b>554</b>
 <b>E ASCII Form Conversion Utility</b>	 <b>555</b>
 <b>F Obsolete Entities</b>	 <b>569</b>
<b>F.1</b> General . . . . .	<b>569</b>
<b>F.2</b> Obsolete General Note FC 0	<b>569</b>
<b>F.3</b> Obsolete Use of Single Parent Associativity	<b>571</b>
<b>F.4</b> External Logical Reference File Index (Type 402, Form 2) .	<b>572</b>
<b>F.5</b> View List Associativity (Type 402, Form 6) . . . . .	<b>573</b>
<b>F.6</b> Signal String Associativity (Type 402, Form 8) . . . . .	<b>574</b>
<b>F.7</b> Text Node Associativity (Type 402, Form 10) . . . . .	<b>576</b>
<b>F.8</b> Connect Node Associativity (Type 402, Form 11) . . . . .	<b>578</b>
<b>F.9</b> Region Fill Property (Type 406, Form 4) . . . . .	<b>580</b>
 <b>G Parallel Projections from Perspective Views</b>	 <b>581</b>

## CONTENTS

<b>H Deprecated Binary Form</b>	<b>583</b>
H.1    Constants . . . . .	583
H.1.1    Integer Numbers . . . . .	583
H.1.2    Real Numbers. . . . .	585
H.1.3    String Constants . . . . .	585
H.1.4    Pointers. . . . .	585
H.1.5    Language Constants. . . . .	585
H.2    File Structure . . . . .	587
H.2.1    Binary Flag Section. . . . .	588
H.2.2    Start Section. . . . .	590
H.2.3    Global Section. . . . .	591
H.2.4    Directory Entry Section. . . . .	591
H.2.5    Parameter Data Section. . . . .	593
H.2.6    Terminate Section. . . . .	594
<b>I Manifold Solid B-Rep Objects</b>	<b>596</b>
<b>J List of References</b>	<b>600</b>
<b>K Glossary</b>	<b>603</b>
<b>L Index of Entities</b>	<b>618</b>

## LIST OF FIGURES

As an aid to testing postprocessor implementations, some of the data files contain the actual entities that they illustrate. The corresponding illustrations can be identified by the fact that the data file name is embedded in the figure caption. For example, [Figure 105](#) on page 281 is an example of the 20 fill pattern codes defined for the Sectioned Area Entity ([Type 230](#)), and the data file F230.IGS actually contains 20 instances of this entity.

<a href="#">1</a>	Categories of Product Definition.. . . . .	2
<a href="#">2</a>	General file structure of the Fixed Format. . . . .	16
<a href="#">3</a>	Format of the Start section in the Fixed Format . . . . .	17
<a href="#">4</a>	Format of the Directory Entry (DE) Section in the Fixed Format .. . . . .	24
<a href="#">5</a>	Format of the Parameter Data (PD) Section in the Fixed Format . . . . .	33
<a href="#">6</a>	Format of the Terminate section in the Fixed Format . . . . .	34
<a href="#">7</a>	General file structure in the Compressed Format . . . . .	36
<a href="#">8</a>	Multiple Transformation Cases.. . . . .	40
<a href="#">9</a>	Interpretation of ZT Displacement (Depth) for Annotation Entities . . . . .	47
<a href="#">10</a>	Entity Usage According to System Category. . . . .	49
<a href="#">11</a>	Subfigure Structures . . . . .	52
<a href="#">12</a>	General Connectivity Pointer Diagram . . . . .	54
<a href="#">13</a>	External Linkages . . . . .	56
<a href="#">14</a>	Finite Element Modeling File Structure . . . . .	58
<a href="#">15</a>	Finite Element Modeling Logical Structure . . . . .	59
<a href="#">16</a>	F100X.IGS Examples Defined Using the Circular Arc Entity . . . . .	66
<a href="#">17</a>	Parameterization of the Composite Curve . . . . .	69
<a href="#">18</a>	Example Defined Using the Composite Curve Entity . . . . .	70
<a href="#">19</a>	F104X.IGS Examples Defined Using the Conic Arc Entity . . . . .	74
<a href="#">20</a>	F10620X.IGS Examples Defined Using the Centerline Entity . . . . .	80
<a href="#">21</a>	Definition of Patterns for the Section Entity . . . . .	83
<a href="#">22</a>	F10640X.IGS Examples Defined Using the Witness Line entity . . . . .	85
<a href="#">23</a>	Examples Defined Using the Plane Entity . . . . .	89
<a href="#">24</a>	F110X.IGS Examples Defined Using the Line Entity . . . . .	91
<a href="#">25</a>	F112PX.IGS Parameters of the Parametric Spline Curve Entity . . . . .	96
<a href="#">26</a>	F112X.IGS Examples Defined Using the Parametric Spline Curve Entity . . . . .	97
<a href="#">27</a>	Parameters of the Parametric Spline Surface Entity . . . . .	101
<a href="#">28</a>	Examples Defined Using the Point Entity . . . . .	103
<a href="#">29</a>	Examples Defined Using the Ruled Surface Entity . . . . .	106
<a href="#">30</a>	Parameters of the Ruled Surface Entity . . . . .	106
<a href="#">31</a>	Examples Defined Using the Surface of Revolution Entity . . . . .	108

## LIST OF FIGURES

32	Parameters of the Surface of Revolution Entity . . . . .	109
33	Parameters of the Tabulated Cylinder Entity . . . . .	111
34	Example of the Transformation Matrix Coordinate Systems . . . . .	118
35	Notation for FEM-specific Forms of the Transformation Matrix Entity . . . . .	119
36	Definition of Shapes for the Flash Entity (continues on next page) . . . . .	121
37	F126X.IGS Example of Rational B-Spline Curve Entity . . . . .	125
38	Nodal Displacement Coordinate Systems . . . . .	136
39	Finite Element Topology Set. . . . .	140
40	Finite Element Topology Set (continued) . . . . .	142
41	Finite Element Topology Set (continued) . . . . .	144
42	Finite Element Topology Set (continued) . . . . .	146
43	Finite Element Topology Set (continued) . . . . .	148
44	Finite Element Topology Set (continued) . . . . .	150
45	Finite Element Topology Set (continued) . . . . .	152
46	Offset Surface in 3-D Euclidean Space . . . . .	156
47	Examples of the Boundary Entity . . . . .	161
48	Parameters of the CSG Block Entity . . . . .	175
49	Parameters of the CSG Right Angular Wedge Entity . . . . .	177
50	Parameters of the CSG Right Circular Cylinder Entity . . . . .	179
51	Parameters of the CSG Right Circular Cone Frustum Entity . . . . .	181
52	Parameters of the CSG Sphere Entity . . . . .	183
53	Parameters of the CSG Torus Entity . . . . .	185
54	Parameters of the CSG Solid of Revolution Entity . . . . .	187
55	Parameters of the CSG Solid of Linear Extrusion Entity . . . . .	189
56	Parameters of the CSG Ellipsoid Entity . . . . .	191
57	Example of a Boolean Tree . . . . .	194
58	Hierarchical nature of the MSBO . . . . .	201
59	Construction of the MSBO . . . . .	202
60	Defining data for un-parameterized plane surface (Form Number = 0) . . . . .	205
61	Defining data for parameterized plane surface (Form Number = 1) . . . . .	205
62	Defining data for un-parameterized right circular cylindrical surface (Form Number = 0) . . . . .	208
63	Defining data for parameterized right circular cylindrical surface (Form Number = 1) . . . . .	208

## LIST OF FIGURES

64	Defining data for un-parameterized right circular conical surface (Form Number = 0) . . . . .	211
65	Defining data for parameterized right circular conical surface (Form Number = 1) . . . . .	211
66	Defining data for un-parameterized spherical surface (Form Number = 0) . . . . .	214
67	Defining data for parameterized spherical surface (Form Number = 1) . . . . .	214
68	Defining data for un-parameterized toroidal surface (Form Number = 0) . . . . .	217
69	Defining data for parameterized toroidal surface (Form Number = 1) . . . . .	217
70	Construction of Leaders for the Angular Dimension Entity . . . . .	219
71	F202X.IGS Examples Defined Using the Angular Dimension Entity . . . . .	219
72	Examples Defined Using the Curve Dimension Entity . . . . .	221
73	F206X.IGS Examples Defined Using the Diameter Dimension Entity . . . . .	223
74	Parameters of the Flag Note Entity . . . . .	226
75	Examples Defined Using the Flag Note Entity . . . . .	226
76	F210X.IGS Examples Defined Using the General Label Entity . . . . .	228
77	F212X.IGS Examples Defined Using the General Note Entity . . . . .	235
78	General Note Text Construction. . . . .	235
79	F212BX.IGS General Note Example of Text Operations . . . . .	236
80	Examples of Drafting Symbols That Exceed Text Box Height . . . . .	236
81	General Note Font (OCR-B) Specified by FC 19 . . . . .	237
82	General Note Font Specified by FC 1001 . . . . .	238
83	General Note Font Specified by FC 1002. . . . .	239
84	General Note Font Specified by FC 1003 . . . . .	240
85	UNTESTED General Note Font Specified by FC 3001 . . . . .	241
86	Text Containment Area . . . . .	253
87	Character Height, Inter-line Spacing . . . . .	253
88	Character Width, Inter-space, Box Width . . . . .	254
89	Examples of Fixed Width Character Inter-space . . . . .	254
90	Rotation, Slant and Character Angle . . . . .	255
91	Text Containment Area . . . . .	255
92	Character Height, Width, Inter-space, Box Width . . . . .	256
93	Character Height, Width, Inter-space, Box Width . . . . .	256
94	Examples Defined Using the Leader Entity . . . . .	260
95	Structure of Leaders Internal to a Dimension . . . . .	260
96	F214X.IGS Definition of Arrowhead Types for the Leader (Arrow) Entity . . . . .	261
97	F216X.IGS Examples Defined Using Form 0 of the Linear Dimension Entity . . . . .	263

## LIST OF FIGURES

98	F21601X.IGS Examples of Linear Dimension Forms ‡ . . . . .	263
99	F218X.IGS Examples Defined Using the Ordinate Dimension Entity . . . . .	266
100	F21801X.IGS Example Defined Using Form 1 of the Ordinate Dimension Entity ‡ .	266
101	Examples Defined Using the Point Dimension Entity . . . . .	268
102	F222X.IGS Examples Defined Using the Radius Dimension Entity . . . . .	271
103	F22201X.IGS Example Defined Using Form 1 of the Radius Dimension Entity .	271
104	Examples of Symbols Defined Using the General Symbol Entity . . . . .	274
105	F230X.IGS Predefined Fill Patterns for the Sectioned Area Entity . . . . .	281
106	F230_4X.IGS Examples of Nested Definition Curves . . . . .	285
107	Examples of Invalid Definition Curves . . . . .	285
108	Example of an Invalid Relationship for Definition Curves . . . . .	286
109	Example of Two Ways to Define an Area . . . . .	286
110	F23000X.IGS Examples of Standard and Inverted Crosshatching ‡ . . . . .	287
111	Relationships Between Entities in an Associativity . . . . .	291
112	Line Font Definition Using Form Number 1 (Template Subfigure) . . . . .	296
113	Line Font Definition Using Form Number 2 (Visible-Blank Pattern) . . . . .	296
114	F30402X.IGS Examples of Standard Line Font Patterns . . . . .	297
115	Parameters of the Isosceles Triangle Macro in Example 1 in Text . . . . .	314
116	Repeated Parallelograms Created by Macro Example 2 in Text . . . . .	316
117	Concentric Circles Created by Macro Example 3 in Text . . . . .	318
118	Ground Symbol Created by Macro Example 4 in Text . . . . .	320
119	Example of a Character Definition . . . . .	326
120	Example of a Character Definition Including Descenders . . . . .	326
121	Dimensioned Geometry Associativity . . . . .	388
122	Use of DOF with Angular Dimensions . . . . .	407
123	Use of DOF with Linear and Ordinate Dimensions. . . . .	407
124	Use of DLF . . . . .	408
125	Using Clipping Planes with a View in a Drawing . . . . .	413
126	Parameters of the Drawing Entity . . . . .	414
127	Measurement of the Line Widening Property Values . . . . .	421
128	Relationship Between Properties Used to Represent a Composite Material . .	431
129	Use of the Vector $\bar{D}$ to Define the Element Material Coordinate System . . .	436
130	Internal Load and Strain Sign Convention . . . . .	436
131	Illustrations of Line Font Patterns for Different Values of LFPC . . . . .	455

## LIST OF FIGURES

132	Examples of tolerance formats (UTOL = 0.01, LTOL = -0.02) . . . . .	474
133	Placement of Text Using TP and TL. . . . .	477
134	F40631X.IGS Example of Basic Dimension Property . . . . .	479
135	F40635X.IGS Examples defined using the underscore and overscore properties. .	484
136	Use of the Closure Property . . . . .	487
137	Relationship Between Subfigure Definition and Subfigure Instance . . . . .	489
138	F408X.IGS Examples of Subfigure Instances at Various Scales and Orientations. .	489
139	Orthographic Parallel Projection of AB on a View Plane . . . . .	494
140	View Coordinate System . . . . .	494
141	Planes Defining the View Volume . . . . .	495
142	Definition of a Perspective View . . . . .	498
143	Relationship Between the Nodal Load/Constraint Entity and Tabular Data Properties	507
A1	Electrical Part Example . . . . .	525
A2	Mechanical Part Example . . . . .	528
A3	Drawing and View Example . . . . .	536
C1	Case 1: Hyperbola oriented (aligned) along the X-axis . . . . .	552
C2	Case 2: Hyperbola oriented (aligned) along the Y-axis . . . . .	553
F1	Obsolete General Note Font specified by FC 0 . . . . .	570
H1	Format of the Control Byte Used in the Binary Form . . . . .	584
H2	Format of an Integer Number in the Binary Form . . . . .	584
H3	Format of a Real Number in the Binary Form . . . . .	586
H4	Structure of a String Constant in the Binary Form . . . . .	586
H5	General File Structure in the Binary Form . . . . .	587
H6	Format of the Binary Flag Section in the Binary Form . . . . .	589
H7	Format of the Start Section in the Binary Form . . . . .	590
H8	Format of the Global Section in the Binary Form . . . . .	591
H9	Format of the Directory Entry (DE) Section in the Binary Form . . . . .	592
H10	Format of the Parameter Data (PD) Section in the Binary Form . . . . .	593
H11	Format of the Terminate Section in the Binary Form . . . . .	595
I1	One possible MSBO representation and the Euler formula of a cylinder with capping planar surfaces . . . . .	597
I2	One possible MSBO representation and the Euler formula of a sphere. . . . .	598
I3	Euler formula of a Torus . . . . .	599

## **List of Tables**

1	Parameters in the Global Section. . . . .	18
2	Directory Entry (DE) Section . . . . .	25
3	Curve and Surface Entities . . . . .	38
4	Examples of Physical Parent-Child Relationships . . . . .	41
5	Untested Entities . . . . .	61
6	Finite Element Topology Set. . . . .	138
7	Finite Element Topology . . . . .	139
8	Description of TYPE Numbers for the Nodal and Element Results Entities . . . . .	170
9	Character Names for the Symbol and Drafting Fonts . . . . .	242
10	Predefined Fill Patterns for the Sectioned Area Entity . . . . .	276
11	Electrical Attribute List (ALT=2) . . . . .	339
12	AEC Attribute List (ALT=3) . . . . .	360
13	Process Plant Attribute List (ALT=4). . . . .	362
14	Electrical and LEP Manufacturing Attribute List (ALT=5) . . . . .	367
15	Line Font Property Patterns . . . . .	453

### 1.1 Purpose

This Specification establishes information structures for the digital representation and exchange of product definition data. It supports exchanging this data among Computer- Aided Design and Computer Aided Manufacturing (CAD/CAM) Systems.

### 1.2 Field of Application

This Specification defines file structure and language formats to represent geometric, topological, and non-geometric product definition data. These formats are independent of the modeling method used, and they support data exchange using physical media or electronic communication protocols (defined in other standards).

Chapter 1 defines the overall purpose and objectives of this Specification. Chapter 2 defines each section of the exchange file's structure. Chapter 3 classifies the entities that contain the product definition data. Chapter 4 defines each entity and how it is used to represent the geometry, annotation, definition, and organization components of a complete product definition.

### 1.3 Concepts of Product Definition

This Specification provides the framework for communicating the essential engineering characteristics of physical objects called *products*. Because these characteristics describe a product in terms of its shape, dimensions, and features, they can be used to design, manufacture, market, and maintain products.

Traditionally, engineering drawings and related information have defined products, but in today's CAD/CAM environments, most drawings exist in computerized form. Because contemporary computer technology ranges from two-dimensional drafting systems to sophisticated solid modelers, data exists in a variety of incompatible formats. A common data communication format facilitates concurrent product and process development among users of different systems, as well as the eventual communication to computerized machines that manufacture and inspect the product.

[Figure 1](#) categorizes product definition data by its principal roles in describing a product. This Specification provides for communicating a portion of this data consistent with the capabilities of basic and advanced CAD/CAM systems.

### **1.3 CONCEPTS OF PRODUCT DEFINITION**

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

**Figure 1. Categories of Product Definition**

## 1.4 Conformance to the Specification

ECO658

**1.4.1 Background.** This Specification's diverse functionality complicates assessing implementation conformance because it can be used in so many ways. Applications having basically different functionality (*e.g.*, mechanical CAD and electrical design) are likely to use different combinations of the entities defined in this Specification. Furthermore, even applications having basically similar functionality (*e.g.*, two CAD products) may use different combinations of entities either because the systems have dissimilar approaches to the same task, or because the designers simply decided to use different entities to represent similar native information. Application protocols have been created to help resolve the diversity issue by specifying exactly how entities should be used for particular purposes. Application protocols include their own conformance requirements which supplement the conformance requirements in this section.

When conformance evaluations are based on solely objective criteria, they can determine only whether files contain the documented combinations of entities, and whether these entities are both syntactically and structurally correct. An implementation conforming to all of the objective criteria is not necessarily interoperable with other implementations. Thus, conformance is a prerequisite for successful interoperability, but it does not guarantee it. Although interoperability is *not* a conformance criterion, it is clear that effective interoperability is a primary goal of exchanging files as defined by this Specification.

The availability of good documentation improves testing effectiveness and can assist in assessment of interoperability between potential exchange partners. Refer to "Interoperability Acceptance Testing Methodology Guidelines [IPO93]" for more information.

**1.4.2 Documentation requirements.** All implementations claiming conformance to this version of the Specification shall adhere to all of the requirements in this section and to all of the specific rules for all individual entities they claim to support.

All implementations claiming conformance to this Specification shall have user documentation which accurately indicates the implementation's support of entities defined in this Specification. Preprocessors and postprocessors shall also document entity mapping. Without such documentation, assessing conformance is costly, difficult, and totally subjective.

The documentation shall specify expected processing results for all entities defined in the version of this Specification to which the implementation claims conformance (*i.e.*, the mapping information shall be comprehensive). This does not imply that an implementation must *support* all possible entity data to conform, since support is claimed and evaluated for individual entities, or for related entity combinations, rather than for the implementation as a whole. Furthermore, since few implementations are comprehensive enough to support *everything* defined in this Specification or in their native system, the documentation shall identify the category of support (full, partial, or none) by entity type, form number, or element (*e.g.*, many implementations would state "partial support" for the General Note Entity (Type 212) since they don't support the entity element specifying KANJI text). Exhaustive documentation of mathematical limitations is not required; however, failures due to such limitations are non-conforming.

**1.4.3 Conformance rules.** It is intended that conforming implementations shall be capable of processing input files according to their documentation, without halting or aborting, regardless of bad data. Any other behavior is a bug. Developers are responsible for bug repair, and users are responsible for determining if bugs are unacceptable. When a specific validation test suite is used to evaluate claimed conformance, any failure is non-conforming.

## 1.4 CONFORMANCE TO THE SPECIFICATION

Conformance rules are based on these principles:

1. Conformance is defined in terms of a complying exchange file and the implementation's mapping table documentation.
2. Conformance is defined for a single processor in isolation (*i.e.*, not in terms of interoperability).
3. Conformance is defined separately for these implementation categories: preprocessors, post-processors (including format converters), and tools (including editors, analyzers, browsers, and viewers).
4. Conformance is based on factual information, not a value judgment; it is categorized as "conforming," or "non-conforming."
5. An implementation is considered "conforming" if all of its documented support claims for individual entities are met.

**1.4.4 Conformance rules for exchange files.** All sections of a complying exchange file shall be syntactically and structurally correct as defined by the version of the Specification specified in the file's Global Section.

**1.4.4.1 Unprocessable entities.** For the purpose of evaluating conformance, *unprocessable entities* are defined as 1) obsolete entities listed in [Appendix F.2](#)) entity types or forms defined in a newer version of the Specification than the implementation supports according to the user documentation, or 3) entities specified as "not supported" in the user documentation. If a file contains an unprocessable entity within a multi-entity structure (e.g., a composite curve), an implementation can ignore the entity or can ignore the entire structure; either behavior is considered conforming providing it is specified in the user documentation.

For information concerning entities having UNTESTED status in this Specification, [see section 1.9](#).

**1.4.5 Conformance rules for preprocessors.** A preprocessor is an implementation designed to translate native CAD system data, other graphics system data, or data in another standard exchange format, into the exchange file format defined by this Specification.

A conforming preprocessor shall create complying exchange files. File content shall represent the native entities according to the user documentation. The preprocessor shall translate all supported native entities, shall report all unsupported native entities, and shall report all processing errors. It is sufficient to report the first occurrence of each kind of error condition and to summarize errors.

Preprocessor conformance is claimed for native entities and their mapping to the exchange file format (*i.e.*, a preprocessor does not claim conformance for the Arc Entity ([Type 100](#)); it claims conformance for its native entity named "circle" and maps it to the Arc Entity.). If conformance testing substantiates the mapping, the preprocessor is conforming. Users need to review both the mapping and the conformance test results to determine if the implementation meets their requirements.

Conforming example:

The native database contains an entity called "line" defined by its start and end points. The documentation states that the native entity is mapped as two instances of the Point Entity ([Type 116](#)). Evaluation of the exchange file indicates the implementation meets its conformance claim for "line" because the output file contains two instances of the Point Entity with the same coordinates as the "line" start and end points.

## 1.4 CONFORMANCE TO THE SPECIFICATION

Non-conforming example:

The native database contains an entity called “line” defined by its start and end points. The documentation states that the “line” is mapped to the Line Entity ([Type 110](#)). Evaluation of the output file indicates the implementation fails to meet its conformance claim for “line” because the output file contains two instances of the Point Entity ([Type 116](#)).

**1.4.6 Conformance rules for postprocessors.** A postprocessor is an implementation designed to translate data from the exchange file format defined by this Specification into native CAD system data, other graphics system data, or into another standard exchange format.

A conforming postprocessor shall be capable of reading any complying exchange file without halting or aborting, including exchange files containing unprocessable entities. All unprocessable entities shall not be translated. Incorrect translation of any entity defined in this Specification due to insufficient entity type or form validation is non-conforming. The postprocessor shall translate all supported entities, shall report all unprocessable entities, and shall report all processing errors. It is sufficient to report the first occurrence of each kind of error condition and to summarize errors. Postprocessors which include viewing capability shall comply with the conformance rules for viewers (see [Section 1.4.7](#)).

Postprocessor conformance is claimed for exchange file entities and how they are mapped to native format. All translated entities shall be mapped into native entities which preserve the functionality and match the attributes and relationships of the entities in the exchange file according to the user documentation. Any entity that is processed differently than documented is non-conforming. If conformance testing substantiates the mapping, the postprocessor is conforming. Users need to review both the mapping and the conformance test results to determine if the implementation meets their requirements.

**1.4.7 Conformance rules for editor, analyzer or viewer tools.** For this purpose, *editor*, *analyzer*, or *viewer tool* refers to a special-purpose implementation for intelligent editing, checking or viewing of exchange files in the format defined by this Specification. General-purpose text editors are excluded.

A conforming tool shall be capable of reading and processing any complying exchange file without halting or aborting, including files that contain unprocessable entities.

A conforming tool shall issue an error message and exit if an exchange file cannot be processed because it has incorrect record structure or does not contain data as defined in this Specification (e.g., native format files). Tools shall report all file processing errors. It is sufficient to report the first occurrence of each kind of error and to summarize errors.

Any tool with viewing capability shall also conform to the functional requirements for viewers; see [section 1.4.7.3](#).

**1.4.7.1 Functional requirements for editors and analyzers** Since file analysis and repair are primary uses for these tools, a conforming tool with edit or analysis capability shall also correctly read and process *non-complying* exchange files having incorrect data within correctly structured records without halting or aborting.

Following any user-initiated editing (assuming no user errors), a conforming editor shall correctly update any automatically maintained values ( e.g., the Parameter Data Line Count in the DE section) prior to producing a complying exchange file.

A conforming editor shall not affect entities that the user did not edit (except for pointers, line numbers, and other “housekeeping” values such as entity counts); defaulted values shall remain defaulted (*i.e.*, it is *not* conforming to export the field’s defined default value). This requirement is intended to prevent introducing problems because the editor assigns an incorrect default value. A conforming editor may export numeric fields with different appearance if the values evaluate identically according to this Specification (*e.g.*, replacing leading spaces with leading zeros in an integer field is conforming).

**1.4.7.2 Functional requirements for browsers.** A conforming browser shall display field values for each entity in the file, including unprocessable or user-defined entities, because doing so does not require knowledge of a field’s functional purpose. Field description labeling is an optional feature; its presence or absence is conforming according to the implementation’s documentation.

**1.4.7.3 Functional requirements for viewers.** For each displayable entity claimed as “supported” in its documentation, a viewer shall create a visual appearance equivalent to the examples appearing in this Specification that depict the entity’s functional intent. Error reporting by ‘view only’ implementations is an optional feature; its presence or absence is conforming according to the implementation’s documentation.

### 1.5 Concepts of the File Structure

This Specification treats product definition data as an organized collection of entities in a format that is independent of the application. The entities include forms common to current and emerging technologies; therefore, mapping to each system’s native representations is simplified.

A file consists of five or six sequentially numbered sections in the following order:

**Flag** Optional section used only when remainder of file is in compressed ASCII or binary form. The binary form is deprecated ([see Appendix H](#)).

**Start** Sender comments

**Global** General file characteristics

**Directory Entry** Entity index and common attributes

**Parameter Data** Entity data

**Terminate** Control totals

The Flag, Directory Entry, and Terminate Sections contain data in fixed-length fields. The Global and Parameter Data Sections contain delimited, variable-length fields. The Start Section is free-form.

Within the file, the fundamental unit of data is the *entity*. Entities are categorized as geometry and non-geometry and may be used in any quantity as required to represent the product definition data.

- Geometry entities define the physical shape of a product and include points, curves, surfaces, solids, and relations that are collections of similarly structured entities.
- Non-geometry entities specify annotation, definition, and structure. They provide a viewing mechanism for composing a planar drawing. They also specify attributes of entities such as color and status, associations among entities, and a flexible grouping structure that allows instancing of entity groups contained either within the file or in an external definition file.

## 1.6 CONCEPTS OF INFORMATION STRUCTURES FOR PRODUCT MODELS

Each entity format includes an entity type and form numbers. Although all are not presently assigned, type numbers 0000-0599 and 0700-5000 are allocated for Specification-defined entities and type numbers 0600-0699 and 10000-99999 are reserved for implementor-defined (*i.e.*, macro) entities. (See Section 1.6.6.)

Within each type, the default form number is zero; some entities have form numbers greater than zero to classify additional functionality. Each entity format also includes a structure for an arbitrary number of pointers to associativity and property entities that also support Specification-defined and implementor-defined types and forms.

### 1.6 Concepts of Information Structures for Product Models

The geometric model of a product is created using the entity set defined in Chapter 4. Since geometry entities generally are defined independently of one another, property and associativity entities are used to augment and define their relationships.

**1.6.1 Property Entity.** The Property Entity (Type 406) allows non-geometric numeric or text information to be related to

- one or more entities that reference it, or
- when the Property Entity is un-referenced, all entities sharing the Property Entity's level number. (This capability allows assigning an application's function to a level. )

Because the Directory Entry Level Number may point to a Definition Levels Property Entity (Type 406, Form 1), a property may be applied to multiple levels. Property values also may be displayed as text if an additional pointer of the property points to a Text Display Template Entity (Type 312). (See Section 2.2.4.5.2.)

**1.6.2 Associativity Entity.** The Associativity Entities (Types 302 and 402) allow several entities to be related to one another. The Specification includes predefine associativities that may be instanced as required. (See Section 4.80.1.) Implementor-defined associativities may be instanced after the Associativity Definition Entity (Type 302) has been used to define the structure of the Associativity Instance Entity (Type 402).

**1.6.3 View Entity.** A view depicts a geometric model of a product. It is a two-dimensional projection of a selected subset of the model, and may include non-geometric information such as text.

The View Entity (Type 410) and Views Visible Associativity Entities (Type 402, Forms 3,4, and 19) control the orientation, scaling, clipping, hidden line removal, and other characteristics associated with individual views. It is essential to understand that a view defines only the *rules and parameters* for depicting a geometric model. Product definition data is *not* duplicated in various views, eliminating the risk of conflicting or ambiguous information.

**1.6.4 Drawing Entity.** The Drawing Entity (Type 404) views and annotation for human presentation. Each file may contain one or more drawings.

**1.6.5 Transformation Matrix Entity.** The Transformation Matrix Entity ([Type 124](#)) applies translation and rotation as needed to any entity in the geometric model. It aids construction of the model itself and supports the development of views and drawings.

**1.6.6 Implementor-defined Entities.** This Specification allows the implementor to define entities to support archiving of data forms unique to a particular system.

### 1.7 Appendices

As an aid to the implementor or user, a series of appendices is included with this Specification. (See the [Table of Contents](#).)

### 1.8 Illustrations

The technical illustrations in this Specification were created on a variety of CAD/CAM systems before conversion to data files in the format defined by this Specification. Because of limitations in the software used to publish this Specification, some of the data files were edited by various tools to create flat, two-dimensional representations. Finally, the files were processed through filtering software to remove identification of the creating system.

As an aid to testing postprocessor implementations, some of the data files contain the actual entities they illustrate; in this case, the data file name is embedded in the figure caption. For example, [Figure 105](#) shows the twenty fill-pattern codes defined for the Sectioned Area Entity ([Type 230](#)), and the data file F230.IGS actually contains 20 instances of this entity.

All of these data files are available from the IGES/PDES Organization's administrative office.

### 1.9 Untested Entities

The IGES/PDES Organization recommends that special consideration be given when implementing certain untested entities or entity forms labeled “‡.” (For a list of entities in this category, see [Section 4](#).) The organization policy is to note those entities or entity forms which are not known to have been implemented. Implementors are cautioned that the entities may not work and may be significantly changed based on implementation experience. The IGES/PDES Organization will remove the untested status when these extensions are known to be useful, complete, and correct. Procedures to accomplish this are documented in [IGES95]. Please communicate any implementation results to the IGES/PDES Organization administrative office.

## 2.1 General

This Specification supports data exchange via an ASCII [ANSI68, ANSI77] file either in Fixed or in Compressed Format. (A Binary Format (which shall not be used to create new files) is described in [Appendix H](#).)

## 2.2 ASCII File Formats

**Fixed Format** Beginning with its first character, the file consists of 80-column lines. Lines are grouped into sections. Each line contains section-specific data field(s) in columns 1-72, an identifying letter code in column 73, and an ascending sequence number in columns 74-80. Within each section, the sequence number begins at 1 and is incremented by 1 for each line. Sequence numbers are right-justified in their field with leading space or leading zero fill.

Sections in the Fixed Format shall appear in the following order:

Section name	Col. 73 Letter Code
Start	S
Global	G
Directory Entry	D
Parameter Data	P
Terminate	T

[See Section 2.2.4](#) for more details concerning purpose and data content of file sections.

Within a section, each entity's set of data fields (appearing on one or more lines) is called a *record*.

Unsequenced lines (*i.e.*, completely blank lines) shall not appear prior to the Terminate Section, nor shall any sequenced lines appear after it. Unsequenced lines may appear after the Terminate Section when the sending system's file structure has blocks larger than 80 bytes and the quantity of records in the file is not a multiple of the block size. Postprocessors shall ignore all lines appearing after the Terminate Section.

**Compressed Format** Compressed Format files shall begin with a Flag Section consisting of one line having spaces in columns 1-72, the section identifier letter code "C" in column 73, and the sequence number 1 right-justified in columns 74-80. The Start, Global, and Terminate Sections are the same as in the Fixed Format. The Directory Entry and Parameter Data sections are combined into a variable-line-length Data Section which saves space by omitting fields having the same value as the previous entity.

Sections in the **Compressed Format** shall appear in the following order:

<b>Section name</b>	<b>Col. 73 Letter Code</b>
Flag	C
Start	S
Global	G
Data	none
Terminate	T

See [Section 2.2.4](#) for more details concerning purpose and data content of file sections.

The Compressed Format has not been widely implemented. Commercial file compression software can reduce the size of Fixed Format files. For details, [see Section 2.3](#).

ECO653

**2.2.1 Field Categories and Defaulting.** All data fields in files conforming to this Specification fit into one of the following categories. When a field's description does not specify its category, the correct category is determined by using the identification criteria and examples. Most fields are designated "required" because their presence (even when defaulted) is mandatory to enable correct parsing of the remainder of the record.

#### **Required, fixed value**

- the field shall appear, and it shall contain the fixed value defined in this Specification.
- postprocessors shall use the value defined in this Specification.

Identification: the field allows one, explicitly defined value.

Examples: Entity use flag for the Drawing Entity ([Type 404](#)), count of parameter fields for the Name Property ([Type 406, Form 15](#)).

#### **Required, default**

- the field shall appear, and a value may be supplied; supplying of a value does not imply the native system user entered it, and no additional information is implied when a field value equals its default value.
- postprocessors shall use the supplied value or shall assign the default value if the field is empty; additional information shall not be inferred from the presence of any value, whether or not it is the same as the field's default value.

Identification: the field has an explicitly defined default value, or has an implicit default value because it is not identifiable as another category.

Examples: the field delimiter character in field 1 of the Global Section, the entity form number in the Directory Entry section, or the count of associated entity pointers appearing after every entity's defined Parameter Data fields.

#### **Required, no default**

- the field shall appear, and a value shall be supplied.
- postprocessors shall use the supplied value.

**Identification:** the field inexplicitly defined “may not be defaulted”, the field can contain a pointer and no meaning is specified for a zero value, or a preceding integer field specifies a non-zero count of required fields.

**Examples:** Directory Entry Section pointer to Parameter Data record, Terminate Section counts, pointers to the constituent entities of a Composite Curve Entity ([Type 102](#)).

### Optional, no default

ECO653

- the field may appear; if it does, its value may be supplied or it may be empty.
- postprocessors shall use the supplied value, but may assign a system-dependent value if the field is empty.

**Identification:** the field is explicitly defined as “optional” in the Directory Entry Section. Optional fields do not occur in free-formatted sections to avoid parsing problems; although trailing fields may function as if they were optional, they are categorized as “required, default,” and the implicit default value is interpreted as meaning “unspecified.”

**Examples:** the entity label and subscript in the Directory Entry Section.

### Ignored

- the field may appear, and if it does, its value may be supplied; any value shall be represented using the defined data type for the field (e.g., even though the field’s value is ignored, a preprocessor shall not put a string data value into an integer field).
- processors shall ignore any supplied value.

**Identification:** the field is explicitly defined as “ignored” or “not applicable (n.a.).”

**Examples:** color of an Associativity Entity ([Type 402](#)), all data other than entity type number of the Null Entity ([Type 0](#)).

### Reserved

- an empty field shall appear; using reserved fields for any exchange purpose prior to their definition by this Specification is prohibited because it will cause compatibility problems.
- postprocessors shall ignore any supplied value.

**Identification:** field is explicitly defined as “reserved.”

**Examples:** fields 16 and 17 of the Directory Entry Section.

In fixed-length-field sections, a default (*i.e.*, empty) value shall be specified by filling the field with space characters. In delimited, variable-length-field sections, a default value shall be specified by the occurrence of two consecutive field delimiters, or by a field delimiter followed by a record delimiter.

Field values shall *not* be defaulted when there is no implicit or explicit default value defined in this Specification.

**NOTE:** Neither a numeric field containing zero (*i.e.*, a “zero” field), nor a space-filled Hollerith string (*i.e.*, a “blank” string such as “4H ”) is a “defaulted” field.

**2.2.2 Data types.** This Specification defines six data types for field values:

1. integer (fixed point)
2. real (floating point)
3. string
4. pointer
5. language statement
6. logical

Regardless of whether data fields are fixed or variable length, the following rules apply to data types:

- Blanks are values only within string fields and in language statements. For all other data types, an entirely blank (*i.e.*, empty) field indicates a “defaulted” field.
- Postprocessors shall ignore leading blanks in numeric fields. Numeric fields shall not contain either embedded or trailing blanks.
- A numeric data type may be either signed or unsigned. If signed, the leading plus or minus determines the sense of the number; if unsigned, the sense is non-negative.
- Numeric data types shall not contain embedded commas even if Global Section field 1 changes the field delimiter to another character. This rule also applies when files originate in countries where “comma” is used instead of “period” as the decimal point in real numbers.
- A string field or language statement may cross line boundaries; this is allowed because their length can exceed the number of usable columns available in one line. When a string field crosses a line boundary, its character count and Hollerith delimiter (“H”) shall appear consecutively on the first line. The string or language statement value continues to the last usable column on the current line (*i.e.*, to column 64 in the Parameter Data Section, and to column 72 in all other sections). The field continues with column 1 on following line(s), until the total quantity of characters is processed.

**2.2.2.1 Integer data type.** An integer (*i.e.*, a fixed point value) always represents an integer value exactly. It may have a positive, negative, or zero value. The absolute magnitude of an integer data type shall not exceed the value  $2^{(N-1)} - 1$ , where N is the number of bits used to represent integer values (Global Parameter 7).

The implicit default for an integer field is zero.

An integer has an optional sign followed by a non-empty string of digits representing a decimal number.

The following are examples of valid integers (assuming the value of Global Parameter 7 is 32):

1	150	2147483647	+3451
0	-10	-2147483647	

**2.2.2.2 Real data type.** A real data type (*i.e.*, a floating point value) is a system-dependent approximation of the value of a real number. It may have a positive, negative, or zero value. The absolute magnitude and precision of a real data type shall not exceed that indicated by Global Parameters 8-9 (for single precision) and 10-11 (for double precision).

The implicit default for a real field is zero.

The following rules and examples apply to real data types, either as parameter data or as processed for text display:

- A *basic real value* contains (in this order) an optional sign, an integer part, a decimal point, a fractional part and an exponent. Both the integer part and the fractional part are sequences of the digits 0-9; either may be omitted, but not both. Either the decimal point or the exponent may be omitted, but not both. A basic real value is interpreted as a decimal number.
- Neither leading zeros in the integer part nor trailing zeros in the fractional part shall be ECO653 interpreted as altering accuracy or implying tolerances of real values.
- An exponent is either of the letters "E" or "D" followed by an optionally signed integer representing the power of ten by which the preceding basic real value is multiplied. An "E" specifies single-precision (corresponding to Global Section parameter 9) and "D" specifies double-precision (corresponding to Global Section parameter 10). If unsigned, the sense of an exponent is non-negative.

The following are examples of valid real values:

256.091	0 .	-0.58	+4.21
1.36E1	-1.3E-02	0.1E-3	1.E+4
145.98763D4	-2145.980001D-5	0.123456789D+09	-.43E2

**2.2.2.3 String data type.** Strings are represented in the Hollerith form as specified in Appendix C of the FORTRAN Standard [ANSI78]. A string is an unlimited-length sequence of ASCII characters. Blanks, parameter delimiters, and record delimiters are treated as ordinary characters within strings.

The string data type consists of a nonzero, unsigned integer value (character count), followed by the Hollerith delimiter ("H"), followed by the quantity of contiguous characters specified by the character count. A string shall not contain any ASCII control characters (*i.e.*, hexadecimal 00 through 1F and hexadecimal 7F).

The implicit default for a string field is NULL (see NULL STRING in [Appendix K](#)).

The following are examples of valid strings:

3H123	10HABC .. ; ABCD
8H0.457E03	12H HELLO THERE

**2.2.2.4 Pointer data type.** A pointer data type is represented by an integer value in the range -9999999 through 9999999; either leading-zero or leading-space fill may be used in fixed-length fields.

The implicit default for a pointer field is zero; a pointer having a zero value (explicitly or due to default assignment) also may be called a *null pointer*. Default assignment rules for pointers differ from the rules for other data types; for a pointer field, defaulting shall occur only when the meaning is defined in this Specification. A typical field description permitting pointer defaulting is “Pointer to the DE of the <referenced item> or zero (default).”

The absolute value of a pointer represents either the Directory Entry or Parameter Data sequence number. This Specification uses the term *reference* to mean “points to.”

Negated pointer values occur in fields which define a different meaning for a zero or positive value. For example, in the color field of the Directory Entry section, a negated pointer references a Color Definition Entity ([Type 314](#)), and non-negative values specify entity colors.

A negated or zero pointer value is valid only where it is explicitly defined in this Specification.

**2.2.2.5 Language Statement data type.** The Language Statement data type is an arbitrary character string containing alphanumeric, punctuation, and space characters from the ASCII character set. A language statement shall not contain any ASCII control characters (*i.e.*, hexadecimal 00 through 1F and hexadecimal 7F). ECO653

Language statement syntax prohibits implicit default values in the language statement itself; however, normal implicit defaults apply to other data types which can be referenced by language statements.

Unlike the string data type, the language statement shall *not* contain a character count and Hollerith delimiter ("H") before its text. [Section 4.71.3](#) defines the syntax of the language statement as used for the Macro Entity. The length of the language statement is determined by means of the Parameter Data line count in the Directory Entry record for the entity ([see Directory Entry Parameter 14](#)).

**2.2.2.6 Logical data type.** A logical data type has only two values: "TRUE" and "FALSE"; The unsigned integer 0 denotes FALSE and the unsigned integer 1 denotes TRUE.

The implicit default for a logical field is FALSE.

**2.2.3 Rules for Forming and Interpreting Free Formatted Data.** The data in several file sections appears in “free format” within specified ranges of columns. When free format is used, the following rules apply (in addition to those in [Section 2.2.2](#)):

- The parameter delimiter (Global Parameter 1) separates parameters.
- The record delimiter (Global Parameter 2) ends the record (*i.e.*, it terminates a list of parameters).
- If two parameter delimiters, or a parameter delimiter followed by a record delimiter, appear consecutively or are separated only by blanks, the field they delimit is “empty” (*i.e.*, “defaulted”). Postprocessors shall assign the explicit or implicit default values according to data type.
- When a record delimiter appears before the end of the parameter list, all remaining “required, fixed value” fields shall be assigned their defined values, and all remaining “required, default” fields shall be assigned their explicit or implicit default values according to the data type.

Parameter Data Section records may be terminated with the record delimiter character prior to the two groups of additional parameters (see Section 2.2.4.5.2). This is valid because the ECO653 pointer counts in the two “required, default” numeric fields preceding the unused “required, no default” pointer fields have been defaulted. The postprocessor shall assign the implicit default of zero, so it does not expect the unused pointer fields.

- The last data column on a free-formatted line (*i.e.*, Column 72 in the Global Section, and Column 64 in the Parameter Data Section) does not substitute for either a parameter delimiter or a record delimiter.

A numeric field shall end at least one column prior to the last data column so its end-of-field delimiter character is on the same line.

- The parameter delimiter and record delimiter characters are treated as text (not as delimiters) when they appear within a string field.

**2.2.3.1 Parameter and Record Delimiter Combinations.** The following ASCII characters are prohibited from being used as either Global Parameter 1 (Parameter Delimiter) or Global Parameter 2 (Record Delimiter) because they will cause parsing difficulties for postprocessors.

Name	Hexadecimal Range
The Control Symbols	0-1F, 7F
The Space Character	20
The Digits 0 through 9	30-39
The Characters + - .	2B, 2D, 2E
The Letters D E H	44, 45, 48

Only four combinations are allowed for the Parameter Delimiter and Record Delimiter in the Global section. They are (where  $\alpha$  and  $\beta$  represent ASCII characters):

Form	Interpretation
	Parameter Delimiter Record Delimiter
	Character Character
1. „	,
2. 1H $\alpha$ $\alpha$ 1H $\beta$ $\alpha$	$\alpha$ $\beta$
3. 1H $\alpha$ $\alpha$ $\alpha$	$\alpha$ ;
4. ,1H $\beta$ ,	, $\beta$

**2.2.4 File Structure.** The file contains six subsections which shall appear contiguously in the file, with no intervening blank lines, in the following order:

- a. Flag Section (Binary or Compressed Format files only)
- b. Start Section
- c. Global Section
- d. Directory Entry Section
- e. Parameter Data Section
- f. Terminate Section

Directory Entry and Parameter Data Section information is combined in the Data Section of Compressed Format files ([see Section 2.3](#)).

[Figure 2](#) illustrates the Fixed Format, which does not include the Flag Section.

1	8 9	16 17	24 25	32 33	40 41	48 49	56 57	64 65	72	73	80
<b>Start Section</b> – a human readable prologue to the file.											S0000001 S0000002 S0000003 ⋮ S000000N
It contains one or more lines											G0000001 G0000002 G0000003 ⋮ G000000N
<b>Global Section</b> – sending system and file information.											D0000001 D0000002
It contains the number of lines needed to hold the parameter fields, separated by											⋮
parameter delimiters, and terminated by one record delimiter, in columns 1–72.											G000000N
<b>Directory Entry Section</b> - contains one pair of lines for each entity											DE back
Directory entry' fields 1-9 in nine 8-column-wide fields											Pointer
Directory entry fields 10-18 in nine 8-column-wide fields											P0000001 P0000002
S0000020 G0000003D0000500 P0000261	<b>Parameter Data Section</b> – values and parameter delimiters terminated by one record delimiter, in columns 1–64; column 65 is unused										T0000001
<b>Terminate Section</b> – record counts for preceding sections; columns 33–72 unused											

Figure 2. General file structure of the Fixed Format

**2.2.4.1 Flag Section.** The optional Flag Section indicates the file is in the Binary Format ([see Appendix H](#)) or in the Compressed Format ([see Section 2.3](#)).

**2.2.4.2 Start Section.** The required Start Section provides a human-readable prologue to the file.

- Start Section lines are identified with the letter code “S” in column 73 and are sequenced in columns 74–80.
- Start Section lines have one data field in columns 1–72. The field may have any content desired by the sender, except that it shall not contain any ASCII control characters (*i.e.*, hexadecimal 00 through 1F and hexadecimal 7F).
- At least one Start Section line shall appear in the file, even if it is blank except for the sequence field.

An example of a Start Section is shown in [Figure 3](#).

**2.2.4.3 Global Section.** The required Global Section contains information describing the pre-processor and information needed by postprocessors to handle the file.

- Global Section records are identified with the letter code “G” in column 73 and are sequenced in columns 74–80 ([see Section 2.2.1](#))

1	72   73    80
This section is a human readable prologue to the file. It contains one or more lines : using ASCII characters in columns 1- 72.	S0000001 S0000002 S0000003 : S000000N

Figure 3. Format of the Start section in the Fixed Format

The first two global parameters define the parameter delimiter and record delimiter characters if the default values (“comma” and “semicolon,” respectively) are not used.

The parameters for the Global Section are written as delimited, variable-length field values described in [Section 2.2.3](#). As stated in [Section 2.2.3](#), Global Section parameter values end at the record delimiter. If the Global Section specifies new delimiter characters, they take effect immediately and are used in the remainder of the Global Section as well as in the rest of the file. The parameters in the Global Section are defined in [Table 1](#) and in the following paragraphs.

Table 1. Parameters in the Global Section

**2.2.4.3.1 Parameter Delimiter Character.** This “required, default” field indicates which character is used to separate parameter values in the Global and Parameter Data sections. The default value is “comma.” Each occurrence of this character denotes the end of the current parameter and the start of the next parameter, except: (1) strings in which the delimiter character may be part of the string, and (2) language statements in which the delimiter character may be a part of the language syntax. [See Section 2.2.3.](#)

**2.2.4.3.2 Record Delimiter.** This “required, default” field indicates which character denotes the end of parameters in the Global Section and in each Parameter Data Section entry. The default value is “semicolon.” Each occurrence of this character denotes the end of the current parameter as well as the end of the parameter list. Two exceptions exist: (1) strings in which the delimiter character may be part of the string; (2) language statements in which the delimiter character may be a part of the language syntax. [See Section 2.2.3.](#)

**2.2.4.3.3 Product Identification From Sender.** This “required, no default” field contains the name or identifier which is used by the sender reference this product.

**2.2.4.3.4 File Name.** This “required, no default” field contains the name of the exchange file.

**2.2.4.3.5 Native System ID.** This “required, no default” field uniquely identifies the native system software which created the native format file used to generate this exchange file (*i.e.*, it does *not* refer to the preprocessor version, which is specified in the next parameter). It shall include the complete vendor’s name, the name by which the system is marketed, and the product ID, version number, or release date of the native system software.

**2.2.4.3.6 Preprocessor Version.** This “required, no default” field uniquely identifies the version or release date of the preprocessor which created this file (*i.e.*, it does *not* refer to the version of the Specification supported by the preprocessor, which is specified by parameter 23.). If the native system software contains the preprocessor (*i.e.*, they are a single executable), this value may be the same as the Native System ID field, or it may be different depending on the release naming convention used by the vendor.

**2.2.4.3.7 Number of Binary Bits for Integer Representation.** This “required, no default” field indicates how many bits are present in the integer representation of the sending system, thereby limiting the range of valid values for integer parameters in the file.

**2.2.4.3.8 Single-Precision Magnitude.** This “required, no default” field indicates the maximum power of ten which can be represented as a single-precision floating-point number on the sending system.

**2.2.4.3.9 Single-Precision Significance.** This “required, no default” field indicates the number of decimal digits of significance which can be represented accurately in the single-precision floating point representation on the sending system.

**2.2.4.3.10 Double-Precision Magnitude.** This “required, no default” field indicates the maximum power of ten which can be represented as a double-precision floating-point number on the sending system.

**2.2.4.3.11 Double-Precision Significance.** This “required, no default” field indicates the number of decimal digits of significance which can be represented accurately in the double-precision floating-point representation on the sending system.

Example: For an IEEE floating point representation (see [IEEE85]) with 32 bits, the magnitude and significance parameters have the values 38 and 6, respectively; for a representation with 64 bits, the values are 308 and 15, respectively.

**2.2.4.3.12 Product Identification for the Receiver.** This “required, default” field contains the name or identifier which shall be used by the receiving system’s software to reference this product. ECO653 The default value is the value specified in parameter 3.

**2.2.4.3.13 Model Space Scale.** This “required, default” field contains the ratio of model space to real-world space (*e.g.*, 0.125 indicates that 1 model space unit equals 8 real-world units). The default value is 1.0. ECO653

**2.2.4.3.14 Units Flag.** This “required, default” field contains an integer value denoting the model units used in the file according to the following table. Postprocessors shall use this field’s value to control the units unless the value is 3. (Field 15 is redundant when the value is not 3, but is convenient for human readability.). The default value is 1. ECO653

Value	Model Units
1	Inches (default)
2	Millimeters
3	(See Parameter 15 for name of units)
4	Feet
5	Miles
6	Meters
7	Kilometers
8	Mils ( <i>i.e.</i> , 0.001 inch)
9	Microns
10	Centimeters
11	Microinches

**2.2.4.3.15 Units Name.** This “required, default” field contains a string naming the model units in the system; the value shall specify the same units as field 14 unless field 14 is 3. The default value is 1. Postprocessors shall ignore this field if it is inconsistent with field 14. ECO653

Value	Model Units
2HIN or 4HINCH	Inches (default)
2HMM	Millimeters
2HFT	Feet
2HMI	Miles
1HM	Meters
2HKM	Kilometers
3HMIL	Mils
2HUM	Microns
2HCM	Centimeters
3HUIN	Microinches

When field 14 is 3, the string naming the desired unit shall conform to [MIL12] or [IEEE260].

**2.2.4.3.16 Maximum Number of Line Weight Gradations.** This “required, default” field is the number of equal subdivisions of line thickness. The value shall be greater than zero. The ECO653 default value is 1.

**2.2.4.3.17 Width of Maximum Line Weight in Units.** This “required, no default” field contains the actual width in model units of the thickest line possible in the file. ECO653

**2.2.4.3.18 Date and Time of Exchange File Generation.** This “required, no default” field is a time stamp indicating when this exchange file was created. Its format is either

15HYYYYMMDD.HHNNSS

or

13HYYMMDD.HHNNSS.

ECO638

If the two-digit year format is used, YY is assumed to be prefixed by "19". The four-digit year format is necessary for years occurring beyond 1999 (or before 1900).

This date format applies to *all* date fields in both Global and Parameter Data Sections in this Specification.

**2.2.4.3.19 Minimum User-Intended Resolution.** This “required, no default” field specifies the smallest distance between coordinates, in model-space units, that the receiving system shall consider as discernible (*e.g.*, if the value is .0001, postprocessors shall consider as “coincident” any coordinate locations in the file which are less than .0001 model-space units apart.).

**2.2.4.3.20 Approximate Maximum Coordinate Value.** This “required, default” field contains the upper bound on the absolute values of all *coordinate* data actually occurring in this model after transformation (*e.g.*, 1000.0 means for all coordinates, |X|, |Y|, |Z| <= 1000.0). It specifies a cubic volume, centered on the origin, which can enclose the entire model. The enclosed volume may be larger than the model’s actual volume depending upon its shape and its distance from the origin (*e.g.*, a model whose coordinates range from (999,999,0) to (1000,1000,0) has a volume of 1 cubic unit for the model, but in this case, the field’s value is 1000, which specifies a volume of 1,000,000,000 cubic units. ECO653

This field shall be defaulted or shall contain 0.0 if its value cannot be determined accurately (*i.e.*, large values having no relationship to actual entity values in the file shall *not* be specified.). The default value is 0.0, which is interpreted as “the maximum coordinate value is unspecified.”

**2.2.4.3.21 Name of Author.** This “required, default” field contains the name of the person who created this exchange file. The default value is NULL, which is interpreted as “unspecified.” ECO653

**2.2.4.3.22 Author’s Organization.** This “required, default” field contains the name of the organization or group with whom the author is associated. The default value is NULL, which is interpreted as “unspecified.” ECO653

**2.2.4.3.23 Version Flag.** This “required, default” field contains an integer value corresponding to the version of the Specification to which the data in this file complies. The default value is 3. Postprocessors finding an unrecognized value less than 1 shall assign 3; postprocessors finding an unrecognized value greater than 11 shall assign 11.

The values in the table below are valid for this Specification version, and will be incremented for each successive version or ANSI Specification. [Appendix J](#) includes a full citation for each version.

Value	Version	Reference
1	1.0	[NBS80]
2	ANSI Y14.26M -1981	[ANSI81]
3	2.0	[NBS83] (default)
4	3.0	[NBS86]
5	ASME/ANSI Y14.26M -1987	[ASME87]
6	4.0	[NBS88]
7	ASME Y14.26M -1989	[ASME89]
8	5.0	[NIST90]
9	5.1	[USPRO91]
10	USPRO/IPO100 IGES5.2	[USPRO93]
11	5.3	This document

**2.2.4.3.24 Drafting Standard Flag.** This “required, default” field contains an integer value corresponding to the drafting standard to which this file complies, if any. The default value is 0.

Code	Drafting Standard	
0	None	No standard specified (default)
1	ISO	International organization for Standardization
2	AFNOR	French Association for Standardization
3	ANSI	American National Standards Institute
4	BSI	British Standards Institute
5	CSA	Canadian Standards Association
6	DIN	German Institute for Standardization
7	JIS	Japanese Institute for Standardization

**2.2.4.3.25 Date and Time Model was Created or Modified.** This “required, default” field is a time stamp indicating when the native system model was created or last modified. The field shall be defaulted if its value is unavailable; *i.e.*, neither the value of field 18 nor an arbitrary value

shall be specified in lieu of defaulting the field. The default value is NULL, which is interpreted as “unspecified.”

**2.2.4.3.26 Application Protocol/Subset Identifier.** This “required, default” field specifies that the file content conforms to an application protocol, subset, or user-defined protocol. The ECO643 default value is NULL, which is interpreted as “unspecified.” Protocols define rules for using this Specification in a uniform way to improve information exchange for a particular purpose. When not defaulted, this field’s value is defined in the application protocol or subset to which the file content conforms.

**2.2.4.4 Directory Entry Section.** The Directory Entry Section has one Directory Entry record for each entity in the file. The Directory Entry record for each entity is fixed in size and contains 20 fields of eight characters each, in two consecutive 80-character lines. Values are right-justified in each field. With the exception of the fields numbered 10, 16, 17, 18, and 20, all fields in this section shall be either integer or pointer data types. In this section, the word “number” is sometimes used in place of the word “integer.”

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 Entry records within the Directory Entry Section is arbitrary.

ECO653

Within the Directory Entry Section, an entirely blank (*i.e.*, empty) field is defaulted; postprocessors shall assign the default values defined in this Specification (values vary by entity type). Fields 1, 2, 10, 11, 14, and 20 shall *not* be defaulted except in Compressed Format files.

Some of the fields in the Directory Entry may contain either an attribute value or a pointer to an entity containing one or more such values. In these fields, a positive value corresponds to an attribute; a negated value indicates that its absolute value is a pointer to the Directory Entry of an entity containing one or more attribute values.

Since valid files have sequence numbers increasing from one, zero is a valid pointer value only when a specific interpretation for a zero value has been defined for that field in this Specification. In such cases, an empty field or a blank field is equivalent to the zero field. See Section 2.2.4.4.7 for one such instance (*i.e.*, the defaulted field is used instead of a pointer to a Transformation Matrix entity (Type 124) which contains an identity matrix.). Figure 4 shows the format of the fields making up the Directory Entry for each entity. Table 2 and the following paragraphs describe each Directory Entry field.

Elsewhere in this Specification, figures similar to Figure 4 are used with individual entity definitions. The same nomenclature is used, with the following additions and exceptions:

- If the field is blank, it is defaulted, and the postprocessor shall assign 0. (Exception: fields 16 and 17, which are *undefined*, and field 18, which is treated as an empty text string. )
- Explicit values in fields are the only allowed values, *e.g.*, the Entity Type Number and the Form Number.
- The symbol <*n.a.*> indicates that the field has no meaning for this entity. The field shall be empty or shall contain zero. Postprocessors shall ignore the value.
- In the Status Number Field, the following symbols are used:
  - . The symbol (\*\*) has the same meaning as <*n.a.*> as defined in Section 2.2.1. Preprocessors shall supply 00 in the field and postprocessors shall ignore the value. Note: when

## 2.2 ASCII FILE FORMATS

the field is identified as \*\*, the table may contain 00 for clarity. Since \*\* means the field is ignored by postprocessors, 00 is functionally equivalent to \*\* (*i.e.*, \*\*?01\*\* and 00??0100 are functionally equivalent.).

- The symbol (?) means that an appropriate value from the defined range for this field shall appear.
- An explicit numeric value (*e.g.*, 00 or 02) is the only value that shall be supplied in the field.
- Footnotes are used to indicate that the values of some fields shall be ignored under certain conditions.

1	8 9	16 17	24 25	32 33	40 41	48 49	56 57	64 65	72 73	80
(1) Entity Type Number #	(2) Para- meter Data ⇒	(3) Structure #,⇒	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Transfor- mation Matrix 0,⇒	(8) Label Display Assoc. 0,⇒	(9) Status Number #	(10) Sequence Number D #	
(11) Entity Type Number #	(12) Line Weight Number #	(13) Color Number #,⇒	(14) Para- meter Line Count #	(15) Form Number #	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript Number #	(20) Sequence Number D # + 1	

Nomenclature:

- (n) - Field number n
- # - Integer
- ⇒ - Pointer
- #,⇒ - Integer or pointer (pointer is negated)
- 0,⇒ - Zero or pointer

Figure 4. Format of the Directory Entry (DE) Section in the Fixed Format

Table 2. Directory Entry (DE) Section

<u>No.</u>	<u>Field Name</u>	<u>Meaning and Notes</u>
1	Entity Type Number	Identifies the entity type.
2	Parameter Data	Pointer to the first line of the parameter data record for the entity. The letter P is not included.
3	Structure	Negated pointer to the directory entry of the definition entity that specifies this entity's meaning, or zero (default). The letter D is not included.
4	Line font pattern	Line font pattern number, or negated pointer to the Directory Entry of a Line Font Definition Entity ( <a href="#">Type 304</a> ), or zero (default).
5	Level	Number of the level upon which the entity resides, or a negated pointer to the Directory Entry of a Definition Levels Property Entity ( <a href="#">Type 406, Form 1</a> ), or zero (default).
6	View	Pointer to the Directory Entry of a View Entity ( <a href="#">Type 410</a> ), pointer to a Views Visible Associativity Instance ( <a href="#">Type 402, Form 3,4, or 19</a> ), or zero (default).
7	Transformation Matrix	Pointer to the Directory Entry of a Transformation Matrix Entity ( <a href="#">Type 124</a> ) used in defining this entity or zero (default).
8	Label Display Associativity	Pointer to the Directory Entry of a label Display Associativity ( <a href="#">Type 402, Form 5</a> ), or zero (default).
9	Status Number	Comprises four two-digit values which are concatenated in the order listed into a single 8-digit number which fills the field; no space characters are allowed. 1-2 Blank Status 00 Visible 01 Blanked
3-4	Subordinate Entity Switch	3-4 Subordinate Entity Switch 00 Independent 01 Physically Dependent 02 Logically Dependent 03 Both (01) and (02)
5-6	Entity Use Flag	5-6 Entity Use Flag 00 Geometry 01 Annotation 02 Definition 03 Other 04 Logical/Positional 05 2D Parametric 06 Construction geometry
7-8	Hierarchy	7-8 Hierarchy 00 Global top down 01 Global defer 02 Use hierarchy property
10	Section Code and Sequence Number	Physical count of this line from the beginning of the Directory Entry Section, preceded by the letter D (odd number).
11	Entity Type Number	(Same value 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).
13	Color Number	Color number or negated pointer to the Directory Entry of a Color Definition Entity ( <a href="#">Type 314</a> ), or zero (default).
14	Parameter Line Count Number	Quantity of lines in the parameter data record for this entity.
15	Form Number	Form number for entities having more than one interpretation of their parameter values, or zero (default). Entity form numbers are included within each entity's description.
16	Reserved for future use	
17	Reserved for future use	
18	Entity Label	Up to eight alphanumeric characters (right justified), or NULL (default).
19	Entity Subscript Number	1 to 8 digit unsigned number associated with the entity label.
20	Section Code and Sequence Number	Same meaning as Field 10 (even number).

**2.2.4.4.1 Entity Type Number.** Integer number specifying entity type. This number shall be the same as the entity type number in the Parameter Data for this Directory Entry record.

**2.2.4.4.2 Parameter Data Pointer.** Sequence number of the first parameter data record for this entity. The letter P is not included. The number shall be greater than zero and less than or equal to the value of Field 4 in the Terminate Section ([see Section 2.2.4.6](#)).

**2.2.4.4.3 Structure.** For a negated value, the absolute value of this field references the structure definition entity which specifies the schema for this entity type number. This field has meaning only for the Macro Instance Entity (UNTESTED), the Implementor-Defined Associativity Instance Entity ([Type 402](#), [Forms 5001-9999](#)) and the Attribute Table Instance Entity ([Type 422](#), [Forms 0](#) and 1). Non-negative integer values are permitted in this field, but postprocessors shall ignore them. (In versions prior to Version 3.0, non-negative integers were used in this field to designate version numbers.)

**2.2.4.4.4 Line Font Pattern.** Integer number corresponding to the line font (*i.e.*, display pattern) used to display an entity. A positive value indicates that the receiving system's corresponding version of the indicated font shall be used. A negated value indicates that its absolute value references a Line Font Definition Entity ([Type 304](#)) which specifies the display pattern.

Value	Pattern
0	No pattern specified (default)
1	Solid
2	Dashed
3	Phantom
4	Centerline
5	Dotted

Additional line font patterns maybe assigned by using the Line Font Property Entity ([Type 406, Form 19](#)) (see [Section 4.115](#)).

**2.2.4.4.5 Level.** This value specifies one or more levels to be associated with this entity. A positive value specifies the single level number which is associated with this entity. A negated value indicates its absolute value references a Definition Levels Property Entity ([Type 406, Form 1](#)) containing a list of levels to be associated with this entity, thereby allowing the entity to appear on more than one level.

**2.2.4.4.6 View.** Three options exist:

- When the entity is visible in all views, and its display characteristics are the same in all views, the value shall be zero (default).
- When the entity is visible in only one view, the value shall reference a View Entity ([Type 410](#)).
- Otherwise, the value shall reference a Views Visible Associativity Entity ([Type 402, Form 3, 4, or 19](#)). [Type 402, Forms 4 or 19](#) shall be used when the display characteristics of the entity are not the same in all views.

**2.2.4.4.7 Transformation Matrix.** This value references a Transformation Matrix Entity ([Type 124](#)) or is zero (default). Zero implies the identity rotation matrix and a zero translation vector. Transformation Matrix Entity form numbers specify transformation matrix characteristics. See [Section 4.21](#).

**2.2.4.4.8 Label Display Associativity.** This value references a Label Display Associativity Entity ([Type 402, Form 5](#)) which defines how the entity's label and subscript are to be displayed in different views, or is zero (default).

ECO653

**2.2.4.4.9 Status Number.** This value contains four pieces of information which are concatenated into a single integer number that is right-justified in the field; no space characters are allowed. The four two-digit values are concatenated from left to right in the order of the following subsections.

**2.2.4.4.9.1 Blank Status.** This value specifies entity visibility on the receiving system display. A value of 00 specifies the entity is displayed and a value of 01 specifies the entity is not displayed.

**2.2.4.4.9.2 Subordinate Entity Switch.** This value indicates whether or not the entity is referenced by other entities in the file; and if so, what type of relationship exists. An entity can be independent, physically dependent, logically dependent, or both physically and logically dependent. The values are defined as follows:

- 00: Independent.** The entity is not referenced (*i.e.*, pointed to) by any other entities in the file. It can exist alone in the native database.
- 01: Physically Dependent.** This entity (the child) is referenced by another entity (the parent) in the file. The child cannot exist unless the parent exists. The matrix referenced by the entity

(as a child) shall be applied to the entity's definition in order to determine its location in the parent's definition space ([see Section 3.2.3](#)).

Entity A is subordinate to Entity B if, and only if, the parameter data entry of Entity B references Entity A. The additional pointers as defined in [Section 2.2.4.5.2](#) are ignored for the purposes of this definition. This means that entities are NOT subordinate to the View (or Views Visible Associativity) Entity defining the view within which the entity is displayed.

The structure formed by a parent entity and its physically subordinate components is indivisible and may therefore be considered as a single entity. The following are examples of physically subordinate entities:

- A Leader Line Entity referenced by a Linear Dimension Entity.
- A Circular Arc Entity referenced by a Plane Entity.
- A Circular Arc Entity referenced by a Composite Curve Entity.
- A Composite Curve Entity referenced by a Subfigure Definition Entity (note that the Subfigure Definition does NOT reference the constituent entities of the composite curve).

Multiple entity example:

- Entity A is physically subordinate to Entity
- Entity A references a Transformation Matrix M1.
- Transformation Matrix M1 references a Transformation Matrix M2.
- Entity B is subordinate to a Subfigure Definition Entity C.
- Entity B references a Transformation Matrix M3.
- Entity C is instanced in a Subfigure Instance D.
- The parameter data of entity D specifies its scale factor as Sd and position as (Xd,Yd,Zd).
- Entity D references a Transformation Matrix M4.
- Entity D references a View Entity E.
- The view scale factor defined in the parameter data of entity E is Se.
- Entity E occurs within a drawing F at drawing coordinates (Fx,Fy).
- Entity E references a Transformation Matrix M5.

In order to obtain the drawing space coordinates of entity A, the following operations are performed:

1. The coordinates of entity A are transformed by M1.
2. The coordinates resulting from the preceding step are transformed by M2.
3. The coordinates resulting from the preceding step are transformed by M3.
4. The coordinates resulting from the preceding step are scaled by Sd.
5. The coordinates resulting from the preceding step are transformed by M4.
6. The coordinates resulting from the preceding step are translated by the vector (Xd,Yd,Zd). The coordinates resulting from this step are the model space coordinates of entity A.
7. The coordinates resulting from the preceding step are transformed by M5.
8. The coordinates resulting from the preceding step are scaled by the scale factor Se.
9. The coordinates resulting from the preceding step are translated by the vector (Fx,Fy).

- 02: Logically Dependent.** This entity (the child) can exist alone in the native database, but is referenced by one or more grouping entities (the parent(s)) such as the Group Associativity Entity ([Type 402](#), [Form 1](#), [7](#), [14](#), or [15](#)). The matrix referenced by any parent entity has no effect on the location of the child.

An example of a logically subordinate entity is a Line Entity ([Type 110](#)) referenced by a Group Associativity Entity.

- 03: Both Physically and Logically Dependent.** This entity (the child) is physically dependent upon one entity (the physical parent) which references it and is subject to the physical dependency rules. This entity also is referenced by one or more logical grouping entities (the logical parent(s)) and also is subject to the logical dependency rules described. Additionally, an entity shall *not* be physically and logically dependent upon the same parent entity. When positioning the child, the matrix referenced by the physical parent shall be used.

An example of a logically and physically subordinate entity is a line which is part of a group of lines in a subfigure. The Line Entity is referenced by the Subfigure Definition Entity and also is referenced by a Group Associativity Entity.

#### 2.2.4.4.9.3 Entity Use Flag.

This value indicates the entity's classification as follows:

- 00: Geometry.** The entity is used to define the geometry of the structure of the product.
- 01: Annotation.** The entity is used to add annotation or description to the file. This includes geometric entities used to form annotation or description.
- 02: Definition.** 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 which 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 ([Type 408](#)) could define geometry, thus having Entity Use Flag 00, or it could define a drawing format, thus having an Entity Use Flag 01. An Associativity Instance ordinarily would have the Entity Use Flag 03. Exceptions include Associativities concerned with display where they would have the Entity Use Flag 01. The View and Drawing Entities have Entity Use Flag 01 (annotation). Transformation Matrix Entities ([Type 124](#)) are classified according to their use: If used only for annotation (*e.g.*, defining a view), assign Entity Use Flag 01; if used for defining geometry or for defining geometry and annotation, assign Entity Use Flag 00.
- 04: Logical/Positional.** The entity is used as a logical or positional reference by other entities. This usage does not prevent the entity from referencing other entities or having its own attributes. Some entities which may be instanced in this way are Node, Connect Point, and Point when their primary use is as a reference.
- Composite curves consisting of only two connect points used as logical connectors shall have their entity use flag set to 04.
- 05: 2-D Parametric.** This entity is positioned in two-dimensional XY parameter space, considered as a subset of three-dimensional XYZ space, by ignoring the Z coordinate. The transformation matrix from definition space to parameter space shall be two-dimensional (*i.e.*, in Entity 124, [Section 4.21](#),  $T_3 = R_{13} = R_{31} = R_{32} = R_{23} = 0.0$  and  $R_{33} = 1.0$ ). In addition,

the coordinates do not have units of length (*i.e.*, the model space scale and units conversion do not apply). This is intended for use in defining curves on surfaces.

**06: Construction Geometry.** The entity is used only for convenience in preparing the model or drawing, NOT for defining the geometry of the structure of the product. An example is the two lines intersected to find the center of a rectangle.

When an entity having Entity Use Flag 06 is a PARENT entity, then all CHILD entities also shall have Entity Use Flag 06 unless the CHILD has Entity Use Flag 02 (Definition). Entity Use Flag 06 entities may be grouped with Entity Use Flag 00 (Geometry) entities.

**2.2.4.4.9.4 Hierarchy.** This value indicates the relationship between entities in a hierarchical structure and determines which entity's Directory Entry attributes shall control line font, view, entity level, blank status, line weight, and color number. Three values are provided:

- 00:** All of the above Directory Entry attributes shall apply to entities physically subordinate to this entity.
- 01:** None of the above Directory Entry attributes of this entity shall apply to physically subordinate entities. Any physically subordinate entities shall use their own Directory Entry attributes.
- 02:** Individual setting of each of the above directory entry attributes is allowed. A Hierarchy Property Entity ([Type 406, Form 10](#)) (see [Section 4.106](#)) shall specify whether 00 or 01 is applied for each Directory Entry attribute to physically subordinate entities.

Example: If an entity A has 00 in its DE status digits 7 and 8, entities immediately subordinate to A shall use A's attributes; their own attributes are not used. Conversely, if an entity A has 01 in its DE status digits 7 and 8, entities immediately subordinate to A shall use their own attributes; A's attributes are not used.

**2.2.4.4.10 Sequence Number.** A number which specifies the sequence number of the DE line in the Directory Entry Section. The sequence number of the first DE line for any entity is always odd and the sequence number of the second line is always even.

**2.2.4.4.11 Entity Type Number.** This is the same as Field 1.

**2.2.4.4.12 Line Weight Number.** This value specifies the thickness (or width) to use for displaying an entity. Global Parameters 16 and 17 specify a uniform series of possible thicknesses. The largest thickness possible is that specified in Global Parameter 17 and is denoted by setting the Line Weight Number 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 the Line Weight Number equal to 1. Thicknesses between the smallest and largest thickness are increments of the smallest possible thickness and are denoted by setting the Line Weight Number equal to the integer number of (adjacent) increments required.

Thus, display thickness is:

$$\text{Line Weight Number} * (\text{Global Parameter 17}/\text{Global Parameter 16}).$$

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

Thickness is a display attribute which is the same for all occurrences of an entity, regardless of scale factors applied to the entity when it is seen in multiple views or in multiple subfigure instances.

**2.2.4.4.13 Color Number.** Field 13 specifies entity display color. A non-negative color number represents “standard” colors and shall be specified when the precise shade is unimportant; a negated value shall be specified when the precise shade is important; its absolute value references a Color Definition Entity ([Type 314](#)).

Postprocessors shall use the receiving system’s display color which best corresponds to the following descriptive names:

Color No.	Color
0	No color assigned (default)
1	Black
2	Red
3	Green
4	Blue
5	Yellow
6	Magenta
7	Cyan
8	White

Note: Since this Specification includes no mechanism for specifying background color, exchange partners need to realize that it is possible for entities to have the same color as the display background; this makes them appear “invisible” even though they are present. ECO653

**2.2.4.4.14 Parameter Line Count Number.** This is the quantity of lines in the Parameter Data Section which contain the parameter data record for this entity, including any comment lines which follow the line containing the record delimiter character. This value shall be greater than zero, except for the Null Entity ([Type 0](#)), which may specify zero parameter data records.

**2.2.4.4.15 Form Number.** This value indicates an individual interpretation of the entity to be used when processing the parameter data for this entity for those entity types having multiple interpretations of their parameter data, or zero (default). The form number and entity type number uniquely specify parameter data interpretation.

**2.2.4.4.16 Reserved Field.** This field is reserved for future use and shall be empty.

**2.2.4.4.17 Reserved Field.** This field is reserved for future use and shall be empty.

**2.2.4.4.18 Entity Label.** This is the application-specified alphanumeric identifier or name for this entity. It is used in conjunction with the entity subscript number (Field 19) to provide the application-specified alphanumeric identifier for the entity. The entity label is right-justified within the field with leading space fill.

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

**2.2.4.4.20 Sequence Number.** See Section 2.2.4.4.10.

**2.2.4.5 Parameter Data Section.** This file section contains the parameter data associated with each entity. The following information is true for all parameter data.

ECO653

**2.2.4.5.1** Parameter data is free-formatted (see Section 2.2.3) with the first field always containing the entity type number. Therefore, even though the Parameter Data Section tables do not show it, the entity type number and a parameter delimiter precede Index 1 of each entity in the exchange file. The free-formatted part of a parameter line ends in Column 64. Column 65 shall contain a space character. Columns 66 through 72 on all parameter lines shall contain the sequence number of the first line in the Directory Entry of this entity. Column 73 of all lines in the Parameter Data Section shall contain the letter P and Columns 74 through 80 shall contain the sequence number. See Section 2.2.1.

**2.2.4.5.2** Two groups of parameters are defined at the end of the specified parameters for each entity.

The first group of parameters may contain pointers to any combination of one or more of the following entities: Associativity Instance Entity (Type 402), General Note Entity (Type 212), Text Template Entity (Type 312).

- Pointers to associativity instances are called “back pointers” because they point back to the Associativity Instance Entity (Type 402) which references them; back pointers are used only when they are required by the associativity’s definition.
- If an entity references associated text, a pointer to a General Note Entity (Type 212) may be included in the first group of pointers. The referenced note specifies the string and its display parameters.
- If an entity itself contains a string to be displayed, a pointer to a Text Template Entity (Type 312) may be included in the first group of pointers. In this way, Text Template Entities provide display parameters for the first information item in the entity referencing them (see Section 4.75).

The second group of parameters may contain pointers to one or more properties or attribute tables. Either group of parameters, or both, may be defaulted (*i.e.*, empty).

When present, the pointers comprising these parameters are added after all the other specified (or defaulted) parameters, but ahead of the record delimiter as follows:

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
:	:	:	:
Let NV = last parameter number			
NV+1	NA	Integer	Number of pointers to the DEs of Associativity Instances/Text Entities
NV+2	DE(1)	Pointer	Pointer to the DE of the first Associativity Instance/Text Entity
:	:	:	:
NV+NA+1	DE(NA)	Pointer	Pointer to the DE of the last Associativity Instance/Text Entity
NV+NA+2	NP	Integer	Number of pointers to the DEs of Property or Attribute Table Entities
NV+NA+3	DE(1)	Pointer	Pointer to the DE of the the first Property or Attribute Table Entity
:	:	:	:
NV+NA+NP+2	DE(NP)	Pointer	Pointer to the DE of the the last Property or Attribute Table Entity

**2.2.4.5.3** Any desired comment may be added after the record delimiter. Additional comment lines may be used by keeping the same Directory Entry pointer in Columns 65-72 and including the comment lines in the entity's parameter line count (DE Field 14).

[Figure 5](#) shows the format of the Parameter Data Section.

1	64	66	72	73	80
Entity type number followed by parameter delimiter followed by parameters separated by parameter delimiters		DE Pointer	P0000001		
Parameters separated by parameter delimiters followed by record delimiter		Pointer	P0000002		
:		:	:	:	:

Note: The DE pointer is the sequence number of the first Directory Entry line for this entity

[Figure 5.](#) Format of the Parameter Data (PD) Section in the Fixed Format

**2.2.4.6 Terminate Section.** There is only one line in the Terminate Section of the file. It is divided into ten fields of eight columns each. The Terminate Section shall be the last sequenced line of the file.

Unsequenced lines (*i.e.*, completely blank lines) shall not appear prior to the Terminate Section, nor shall any sequenced lines appear after it. Unsequenced lines may appear *after* the Terminate Section when the sending system's file structure has blocks larger than 80 bytes and the quantity of records in the file is not a multiple of the block size. Postprocessors shall ignore all lines appearing after the Terminate Section.

The Terminate Section has a "T" in Column 73 and Columns 74 through 80 contain the sequence number with a value of one (1).

Each field in the Terminate Section record contains a section identifier, left-justified in the field, and the last sequence number used in that section, right-justified in the field. Each field is defined in the table below and is shown in [Figure 6](#). Leading zeroes are not required in sequence numbers.

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

1   8   9	16   17	24   25	32   33	40   41	48   49	56   57	64   65	72   73	80
S0000020	G0000003	D0000500	P0000261		Not Used			T0000001	

Figure 6. Format of the Terminate section in the Fixed Format

### 2.3 Compressed Format

The format described here is an alternative to using the Fixed Format for large files. The Compressed Format can be converted to the Fixed Format and vice-versa. An example of conversion software is shown [Appendix E](#).

**2.3.1 File Structure.** A single Flag Section record shall precede the Start Section and shall contain the character “C” in character position 73 to identify the file as being in the compressed Format. The Start, Global and Terminate Sections are the same as those for the Fixed Format, while the Directory Entry Section and the Parameter Data Section are combined into a single Data Section.

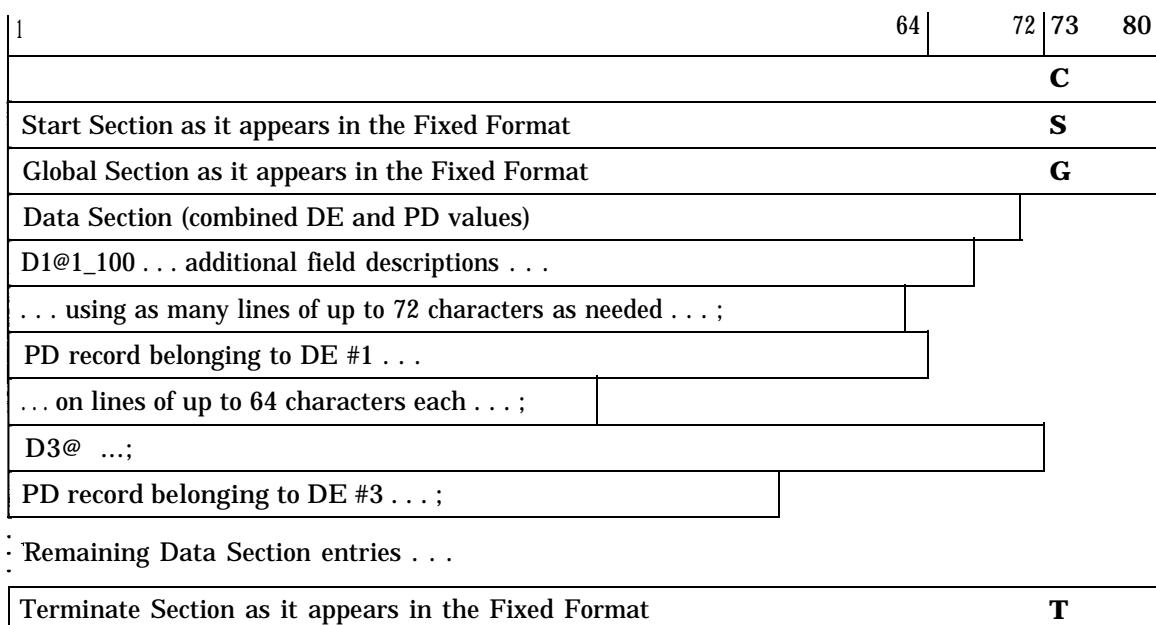
A record in the Data Section contains the data from the entity’s Directory Entry record followed immediately by the data from its Parameter Data record. The first line of the Data record begins with the letter “D” followed without intervening blanks by an unsigned integer whose value is that of the sequence number of the corresponding Directory Entry record ([see Figure 7](#)).

The “D<sequence number>” group of characters is followed by zero or more Directory Entry field specifiers. The field specifier consists of the symbol “@” (commercial at) followed by an unsigned integer identifying the field being specified. The “@<field number>” group is followed by the character “-” (underscore) which is in turn followed by the value of the field (“@<field number>-<value>”). No delimiter is used between the Directory Entry field specifiers, but the collection of field specifications is terminated by a record delimiter character (default: “;”).

The Directory Entry field numbers are the same as those used to identify the Directory Entry fields in the Fixed Format. Fields 2, 10, 11, and 20 are not specified because they are either redundant or meaningless in the Compressed Format. When several Directory Entry fields are being specified, additional lines may be used. The sequence of field specifiers may be broken only between complete specifications, thus assuring that new lines will begin with the character “@”.

The Directory Entry field values need be specified only when they change. Thus, a field retains its value from entity to entity unless a new value is explicitly stated. Only the first entity in a file is assured of containing a complete set of field specifications.

The Directory Entry portion of the Data Section record is followed immediately by the Parameter Data portion. The data from the Parameter Data record begins on a new line and is the same in the Compressed Format as it is in the Fixed Format. Each line is of variable length, and terminates before character position 65, thus assuring that character position 65, if it existed (*i.e.*, if the line were read into a fixed-length, 80-character buffer), would always contain a blank character.



Note: Default record delimiter assumed

Figure 7. General file structure in the Compressed Format

### **3. Classes of Entities**

#### **3.1 General**

This Chapter contains information about the classes of entities and their structures in the product data exchange file. The five classes of entities defined in this Specification are curve and surface geometry entities, constructive solid geometry entities, boundary representation solid entities, annotation entities, and structure entities. Entity type numbers from 100 through 199 are generally reserved for geometry entities.

ECO630

#### **3.2 Curve and Surface Geometry Entities**

**3.2.1 Entity Types.** [Table 3](#) shows curve and surface geometry entities defined in this Specification.

**3.2.2 Coordinate Systems.** 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.

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.

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, an orthogonal matrix and a translation vector are used to generate model space coordinates from definition space coordinates. The orthogonal matrix used for this purpose is called the defining matrix; both it and the translation vector are treated in the description of the Transformation Matrix Entity ([see Section 4.21](#)).

ECO630

The value of the determinant of an orthogonal matrix is always plus or minus one. In the case that the determinant is one, 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 simplify the discussion that follows, 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.

### 3.2 CURVE AND SURFACE GEOMETRY ENTITIES

- The first point of view imagines that the two coordinate systems are initially coincident (*i.e.*, 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 considered to be defined relative to the XT, YT, ZT frame, and the defining matrix then rotates this frame, geometry included, so that the geometry entity is positioned as desired relative to the X, Y, Z frame. ECO630
- 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 defining 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 positioned as desired relative to the X, Y, Z frame. ECO630

It is to be emphasized that the discussion here pertains to a single defining matrix whose action in transforming coordinates can be viewed intuitively in two ways. Each point of view stresses

Table 3. Curve and Surface Entities

<b>Entity Type Number</b>	<b>Entity Type</b>
100	Circular Arc
102	Composite Curve
104	Conic Arc
106	Copious Data
106/11	2D Linear Path
106/12	3D Linear Path
106/63	Simple Closed Planar Curve
108	Plane
110	Line
112	Parametric Spline Curve
114	Parametric Spline Surface
116	Point
118	Ruled Surface
120	Surface of Revolution
122	Tabulated Cylinder
124	Transformation Matrix
125	Flash
126	Rational B-Spline Curve
128	Rational B-Spline Surface
130	Offset Curve
140	Offset Surface
141	Boundary
142	Curve on a Parametric Surface
143	Bounded Surface
144	Trimmed Parametric Surface
190	Plane Surface
192	Right circular cylindrical Surface
194	Right Circular Conical Surface
196	Spherical Surface
198	Toroidal Surface

## 3.2 CURVE AND SURFACE GEOMETRY ENTITIES

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.

In a case when the geometry entity to be located within model space can be contained within a single plane, it can 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).

Each entity is acted upon by a transformation matrix. This implies that each entity makes use ECO630 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 Entity. 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 defining matrix is the identity rotation matrix and the translation vector is the zero vector. (In this situation, a convention is provided to prevent unnecessary processing. See the explanation given in [Section 2.2.4.4.7](#) for Field 7 of the directory entry.)

**3.2.3 Multiple Transformation Entities.** There are only two cases in which entities can be operated on by multiple transformation entities. The first is the explicit case in which an entity points to a transformation entity through its Directory Entry Field 7, and that transformation entity, in turn, points to an additional transformation entity through its Directory Entry Field 7. This structure is illustrated in [Figure 8\(a\)](#).

In the case illustrated by [Figure 8\(a\)](#), the points represented by entity XXX are first operated on by matrix 1. The transformed points resulting from application of matrix 1 are then operated on by matrix 2.

The other case is an implicit one in which two entities are in a parent/child relationship, and each points to a transformation entity through its respective Directory Entry Field 7. A parent/child relationship occurs when one entity (the parent) is pointing to another entity (the child). This structure is illustrated in [Figure 8\(b\)](#). In the case illustrated by [Figure 8\(b\)](#) the points represented by entity XXX are operated upon by matrix 2 and from that point on are transformed like the points in entity YYY, using matrix 1.

ECO630

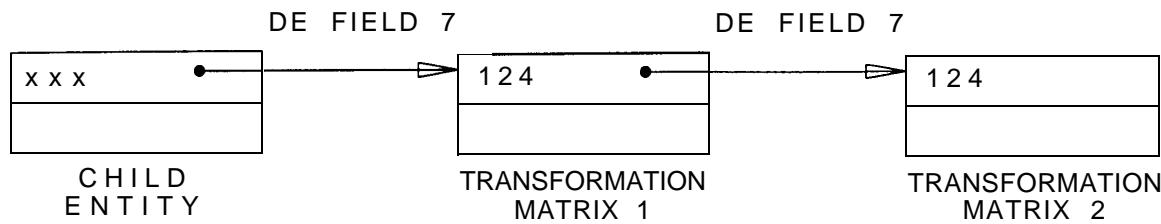
When the specific parent/child relationships shown in [Table 4](#) occur, the implicit relation rule shall apply. Each of the relationships in [Table 4](#) ordinarily results in the subordinate entity switch of the child entity being set to 01 (physically dependent). The exception is the case in which a preprocessor wishes to actually instance the child entity. In this case the child's subordinate entity switch is set to 02 (logically dependent), and the matrix pointed to by the parent has no effect on the location of the child ([see Section 2.2.4.4.9.2](#)).

**3.2.4 Directionality.** Within model space, all curves are directed. Such curves have associated end points; *i.e.*, start point and terminate point. The manner of assigning direction is discussed within the description of each individual entity.

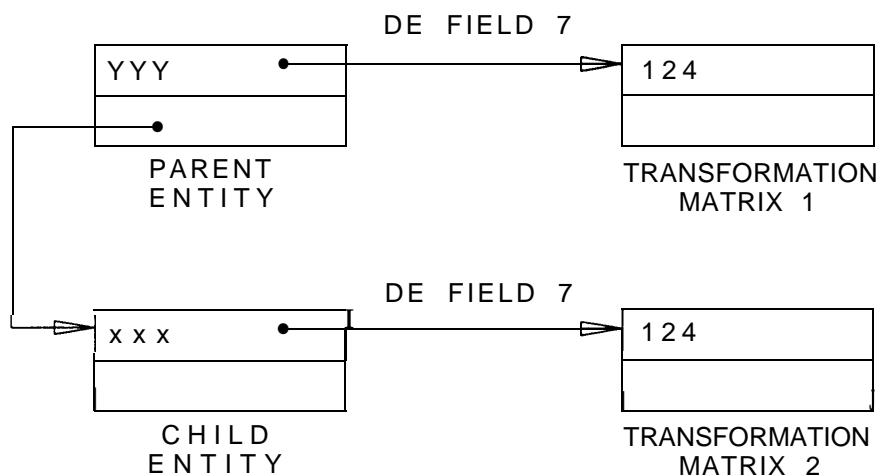
ECO630

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

ECO630



### a) EXPLICIT CASE



### b) PARENT-CHILD IMPLICIT CASE

Figure 8. Multiple Transformation Cases

### 3.3 CONSTRUCTIVE SOLID GEOMETRY ENTITIES

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.

#### 3.2.5 Continuity and Non-degeneracy.

- All model space curves and surfaces shall be at least  $C^0$  (positionally) continuous. ECO630
- All curves shall have non-zero arc length.
- All surfaces shall have non-zero area.
- All solids shall have non-zero volume.

### 3.3 Constructive Solid Geometry Entities

**3.3.1 Entity Types** The Constructive Solid Geometry (CSG) primitive entities are defined ECO630

Table 4. Examples of Physical Parent-Child Relationships

<b>Parent</b>	<b>Child</b>
Composite Curve	all constituents
Plane	bounding curve
Point	display symbol
Ruled Surface	rail curves
Flash	defining entity
Surface of Revolution	axis, generatrix
Tabulated Cylinder	directrix
Offset Curve	base curve
Offset Surface	surface
Trimmed Surface	surface
Angular Dimension	all subordinate entities
Diameter Dimension	all subordinate entities
Flag Note	all subordinate entities
General Label	all subordinate entities
Linear Dimension	all subordinate entities
Ordinate Dimension	all subordinate entities
Point Dimension	all subordinate entities
Radius Dimension	all subordinate entities
General Symbol	all subordinate entities
Sectioned Area	all boundary curves
Entity Label Display	all leaders
Connect Point	display symbol, Text Display Templates
Drawing	all annotation entities
Subfigure Definition	all associated entities
Network Subfigure Definition	all associated entities, Text Display Templates and Connect Points
Nodal Display and Rotation	all General Notes and Nodes
Any entity with Entity Use Flag = 00 or 01	all General Notes in text pointer field

### 3.3 CONSTRUCTIVE SOLID GEOMETRY ENTITIES

set of solid modeling primitive constructs to be used in all solid modelers--either directly in CSG modelers or in other types of modelers after conversion.

CSG primitive entities include the following:

Entity Type Number	Entity Type
150	Block
152	Right Angular Wedge
154	Right Circular Cylinder
156	Right Circular Cone Frustum
158	Sphere
160	Torus
162	Solid of Revolution
164	Solid of Linear Extrusion
168	Ellipsoid

These primitive entities and manifold solid B-Rep object entities can be combined into more complex CSG solids using the following entities: ECO644

Entity Type Number	Entity Type
180	Boolean Tree
182	Selected Component
184	Solid Assembly
430	Solid Instance

**3.3.2 Constructive Solid Geometry Models.** The Constructive Solid Geometry (CSG) entities support a standard format for one of the two mostly widely used solid model representations—CSG.

The CSG entities in this section can be thought of as being one of two types—geometric or structural. The geometry entities are volumetric primitives. The model information for a primitive contains dimensions that define the shape of the primitive, point and vector coordinates that define the local coordinate system of the primitive, and an optional directory entry pointer to a transformation matrix which may be used to further position the primitive. If the point and vector coordinates defining the local coordinate system are not given values, the local coordinate system defaults to the global coordinate system. For the Solid of Revolution and Solid of Linear Extrusion Entities, the shape is partly defined indirectly, via a pointer to a planar boundary curve. ECO630

The structural entities are the Boolean Tree, Solid Instance, and Solid Assembly Entities. The Boolean Tree Entity contains pointers to the elements of the tree and operations such as union, difference, and intersection to be performed on these elements. Elements may be primitives, other boolean trees, solid instances, or manifold solid B-Rep object entities. There may also be a directory entry pointer to a transformation matrix to relocate the entire boolean resultant. ECO630  
ECO644

The Solid Instance Entity contains a pointer to an entity representing a solid and a directory entry pointer to a transformation matrix by which the entity is to be transformed. It is a copy of the solid entity relocated in global space. The solid entity may be a primitive, boolean tree, another solid instance, or an assembly, or a manifold solid B-Rep object entity. ECO630  
ECO644

A solid assembly is a collection of items that share a fixed geometric relationship. The relationship is a logical one and is not to be confused with a boolean union. If the faces of different items in ECO644

### 3.4 BOUNDARY REPRESENTATION SOLID ENTITIES

an assembly touch, they are not removed, as they would be in a boolean union. The items of an assembly may include primitives, boolean trees, other assemblies, solid instances, and manifold solid B-Rep object entities. Corresponding to each item pointed to by the assembly is an optional pointer to a transformation matrix to be applied to that item. Thus, each item of the assembly can be moved independently. There is also an optional directory entry pointer to a global transformation matrix to be applied to the entire assembly of items. This global transformation matrix is applied after each of the individual transformation matrices are applied.

The description of a solid model is an acyclic directed graph. The nodes in the graph are the various geometric and structural entities. This type of graph is like a tree structure, except that the branches of this graph may reconvene as a move is made down the graph, where down is the general direction from root to terminal node. There may be any number of root nodes, which represent the actual solid models. A root may even be within the branches of another root's graph.

The terminal nodes are the primitives and manifold solid B-Rep object entities—the geometric entities. All the other nodes are structural entities. The structural entities are all able to point to each of the other structural entities as well as to primitives or manifold solid B-Rep object entities, with one exception. The boolean tree cannot point to an assembly. ECO644

A CSG solid model is represented by appropriately combining geometric entities with structural entities to create a graph structure.

#### 3.4 Boundary Representation Solid Entities

**3.4.1 Entity Types** The boundary representation (B-Rep) solid model entities consist of a set ECO630 of topological entities, a set of surface entities, and a set of curve entities.

The following topological entities for B-Rep solid models are defined in this Specification:

Entity Type Number	Entity Type
186	#Manifold Solid B-Rep Object
502	#Vertex
504	#Edge
508	#Loop
510	#Face
514	#Shell

### 3.4 BOUNDARY REPRESENTATION SOLID ENTITIES

Only the following surface entities may be used in the construction of B-Rep solid models:

<b>Entity Type Number</b>	<b>Entity Type</b>
114	Parametric Spline Surface
118/1	Ruled Surface
120	Surface of Revolution
122	Tabulated Cylinder
128	Rational B-Spline Surface
140	Offset Surface
190	‡Plane Surface
192	‡Right Circular Cylindrical Surface
194	‡Right Circular Conical Surface
196	‡Spherical Surface
198	‡Toroidal Surface

Only the following curve entities may be used in the construction of B-Rep solid models:

<b>Entity Type Number</b>	<b>Entity Type</b>
100	Circular Arc
102	Composite Curve
104	Conic Arc
106/ 11	2D Path
106/12	3D Path
106/63	Simple Closed Planar Curve
110	Line
112	Parametric Spline Curve
126	Rational B-Spline Curve
130	Offset Curve

**3.4.2 Topology for B-Rep Solid Models.** In mechanical CAD systems the role of topology ECO630 has been traditionally limited to its use in defining B-Rep solid models.

Constraints have been placed on each topological entity with the intention that they be used in the specific application domain of B-Rep solid models. Should another application domain (e.g., AEC or FEM) require different constraints, new form numbers of these entities should be created that limit the context or the utility of the entities. ECO630

Each entity has its own set of constraints. A higher-level entity (e.g., a loop) may impose constraints ECO630 on a lower-level entity (e.g., an edge). At the higher level, the constraints on the lower-level entity are the sum of the constraints imposed by each entity in the chain between the higher- and lower-level entities.

Several topological entities use an Orientation Flag (OF) to indicate whether the direction of a referenced entity agrees with, or is opposed to, the direction of the referencing entity. If the OF is .TRUE., the direction of the referenced entity is correct; if the OF is .FALSE., the direction of the referenced entity should be (conceptually) reversed. It can happen that there are several Orientation Flags in the chain of entities from the high-level referencing entity to the low-level referenced entity. ECO630

**3.4.3 Analytical Surfaces for B-Rep Solid Models.** The entities defined in this set encompass those commonly used for describing the surface geometry of B-Rep solid models. The surfaces specified here are defined in terms of point, vector, and scalar quantities. In general, a point is used to provide positional information and a vector to provide directional information. One or more scalars provide dimensional data.

The symbol convention used in the definition of these entities is shown in the following table:

Symbols used for analytical surfaces	
Symbol	Definition
$a$	Scalar quantity
$\mathbf{A}$	Vector quantity
$\langle \rangle$	Vector normalization
$\mathbf{a}$	Normalized vector (e.g., $\mathbf{a} = \langle \mathbf{A} \rangle = \mathbf{A}/ \mathbf{A} $ )
$\times$	Vector (cross) product
$\cdot$	Scalar (dot) product
$S(x, y, z)$	Analytic surface
$\sigma(u, v)$	Parametric surface
$s_x$	Partial derivative of $S$ with respect to $x$

**3.4.3.1 Entity Types** The following analytical surface entities for B-Rep Solid models are defined in this Specification:

Entity Type Number	Entity Type
123	‡Direction
190	‡Plane Surface
192	‡Right Circular Cylindrical Surface
194	‡Right Circular Conical Surface
196	‡Spherical Surface
198	‡Toroidal Surface

Note that the Plane Surface Entity (Type 190) shall not be used as a clipping plane for a view, and several of these surfaces (plane, cylinder, and cone) are unbounded; *i.e.*, they are infinite surfaces. With the exception of the Plane Surface Entity, these surfaces shall *only* be used in conjunction with B-Rep solid models.

**3.4.3.2 Parameterization of Analytical Surfaces.** For those systems that use parameterized surfaces, a parameterization is defined for each surface. All the surfaces defined here include a point that forms the origin of a Local Coordinate System (LCS). Two direction vectors are used to complete the definition of the LCS. One is the local Z axis direction, and the other is an approximation to the local X axis direction. Let  $\mathbf{z}$  be the local Z axis direction and  $\mathbf{a}$  be the approximate local X axis direction. The method for calculating the local X and Y axis directions is to project the vector  $\mathbf{a}$  onto the plane defined by the origin point  $\mathbf{P}$  and the vector  $\mathbf{z}$ . The local axes are given by:

$$\mathbf{x} = \langle \mathbf{a} - (\mathbf{a} \cdot \mathbf{z})\mathbf{z} \rangle$$

and

$$\mathbf{y} = \langle \mathbf{z} \times \mathbf{x} \rangle.$$

### 3.5 Annotation Entities

**3.5.1 Entity Types** The following annotation entities are defined in this Specification:

Entity Type Number	Entity Type
106	Copious Data Centerline Section Witness Line
202	Angular Dimension
204	‡Curve Dimension
206	Diameter Dimension
208	Flag Note
210	General Label
212	General Note
213	‡New General Note
214	Leader (Arrow)
216	Linear Dimension
218	Ordinate Dimension
220	Point Dimension
222	Radius Dimension
228	General Symbol
230	Sectioned Area

**3.5.2 Construction.** Many annotation entities are constructed by using other entities. For example, the dimension entities may have 0, 1, or 2 pointers to Witness Line Entities (a form of Copious Data), 0, 1, or 2 pointers to Leader (Arrow) Entities and a pointer to a General Note Entity.

For some annotation entities, a witness line or leader, although allowed, may not exist. For these cases the Parameter Data field pointer value can be set zero. If any constructive entity exists, but its display is suppressed, it can be set to blank status or, if allowed, the pointer value can be set to zero.

**3.5.3 Definition Space.** An annotation entity may be defined in XT, YT, ZT definition space (see the discussion in [Section 3.2.2](#)) or in a two-dimensional space associated with a Drawing Entity ([Type 404](#)). In the case of XT, YT, ZT definition space, a transformation matrix is 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 9](#).

While the option of having dimensions occupy different planes exists, it is expected that only a single ECO630 plane will be used. The reason for its existence is due to the structure of annotation entities. As each dimension may comprise several subordinate entities, each subordinate entity, by its definition, has the ability to stand alone and may require its own ZT displacement. It is likely, though not necessary, that each ZT displacement is identical.

In the case where a dimension entity, excluding the curve dimension, has subordinate entities, the ECO635

entities subordinate to the dimension entity must be either coplanar or in parallel planes. All of the children of a particular dimension entity must have the same value in directory entry field 7 (Matrix Pointer). Either the children's or the parent's Matrix Pointer may be non-null, but not both.

### 3.5.4 Dimension Attributes

**3.5.4.1 General.** Most of the dimension entities defined by this specification provide only enough data for the receiving system to restore a visually equivalent representation of the original; additional information (*e.g.*, the geometry being dimensioned) is lost. Dimension attributes enable exchanging this added data to maximize the potential of functionally equivalent entity transfer between systems which support them. Receiving systems lacking CAD entities to contain all attribute data may find some portions useful, or they may ignore the attributes without losing the visual data.

CAD system dimensioning capabilities can be grouped into one of three categories:

ECO630

**MANUAL:** Dimensions are constructed using lines, arcs, and text.

**GENERATIVE:** Dimensions are generated automatically from selected geometry, but the association with the geometry is not maintained after creation.

**ASSOCIATIVE:** Dimensions are generated automatically from selected geometry, and the association is maintained so that a subsequent change to the geometry will cause a corresponding change in the dimension value. Some associative systems with parametric design capabilities also can alter geometry if the dimension value is changed.

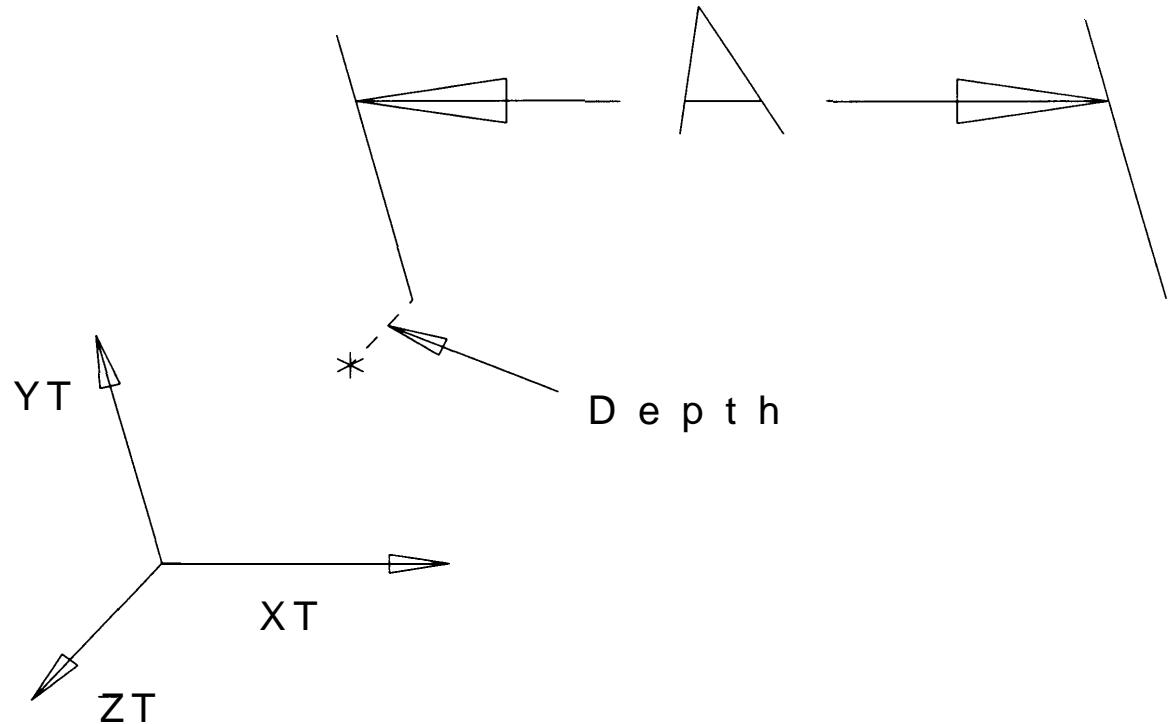


Figure 9. Interpretation of ZT Displacement (Depth) for Annotation Entities

Usage of dimension attribute entities will directly correspond to the CAD system's category. Category 1 systems will be unable to send any attributes, and will probably ignore them in received files. Category 2 systems will be able to send and receive the dimension properties: Dimension Units Property Entity ([Type 406, Form 28](#)), Dimension Tolerance Property Entity ([Type 406, Form 29](#)), Dimension Display Data Property Entity ([Type 406, Form 30](#)), and Basic Dimension Property Entity ([Type 406, Form 31](#)). Category 3 systems will be able to send and receive the Dimensioned Geometry Associativity Entity ([Type 402, Form 21](#)); this entity groups the dimensioned geometry with the necessary dimension properties. [Figure 10](#) illustrates category usage for a diameter dimension.

**3.5.4.2 Usage Rules.** Dimension properties may not be independent; they shall be logically ECO630 subordinate to at least one dimension entity. In some cases (e. g., the Dimension Units Property Entity), more than one dimension can reference one property instance. Properties may be used in any combination which is consistent with dimension entity data; thus, the same dimension will never point to both the Dimension Tolerance and Basic Dimension Property Entities because *basic* dimensions are not tolerance. Property data shall correspond to the data stored in the dimension(s) which reference the property.

If the Dimensional Geometry Associativity Entity is used, the dimension entity and geometry will be ECO630 logically subordinate to it, and any dimension properties will have logically subordinate status. The Dimensioned Geometry Associativity Entity will always have only physically subordinate status; it will always be referenced only by one dimension entity's back pointer. Refer to [Figure 10](#), Category 3.

Some systems maintain additional information about dimensions that is of a global nature and some that is specific to a particular instance of a dimension. Some systems are able to associate ECO630 a dimension with geometry in such a way that if the geometry is changed, the dimension value is automatically updated to reflect the new values. To support the variety of functionality available for dimensions, several Form Numbers of the Property Entity ([Type 406](#)) and a Dimensioned Geometry Associativity Entity ([Type 402, Form 21](#)) are provided.

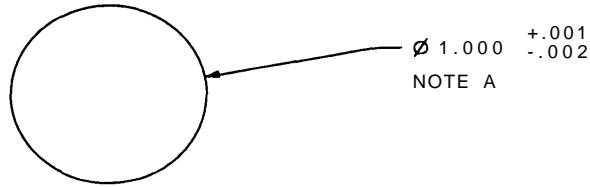
All of these properties are optional, but none may exist independently in a file; each instance must ECO630 be referenced by at least one dimension entity as described in [Section 2.2.4.5.2](#). For example, in the case of the Dimension Units Property Entity ([Type 406, Form 28](#)), it is possible that one instance of the property is sufficient for all of the dimensions in the drawing, or all Angular Dimension Entities ([Type 202](#)) may reference one instance while all Linear Dimension Entities ([Type 216](#)) reference another instance. A similar situation exists for the Dimension Tolerance Property Entity ([Type 406, Form 29](#)).

Some of the properties shall be referenced by only one entity. For example, the Basic Dimension ECO630 Property Entity ([Type 406, Form 31](#)) contains the coordinates of the corners of a box to be drawn around the dimension text, so an instance of this property can be referenced by only one dimension.

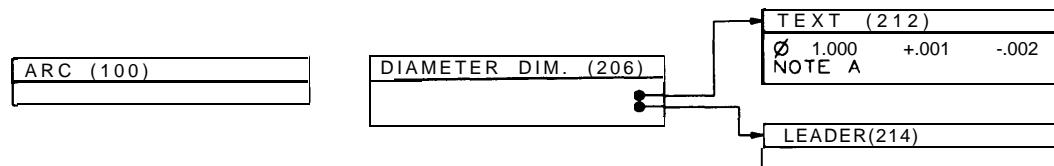
There is no restriction on the order in which these properties are referenced; any or all of them may be present in any combination. If present, some contain numeric values that are intended to replace the text string(s) in the General Note Entity ([Types 212 and 213](#)) that is referenced by the dimension in its PD section, or they may provide information for the interpretation of the text string(s).

Several Form Numbers of the General Note Entity ([Type 212](#)) have been provided to indicate dimension types. Specifically, Form Numbers 1, 2, 3, 4, and 5 communicate information about text placement for dual and tolerance dimensions. The dimension attribute properties and the Form Numbers of the General Note should be used in a logically consistent, non-conflicting manner.

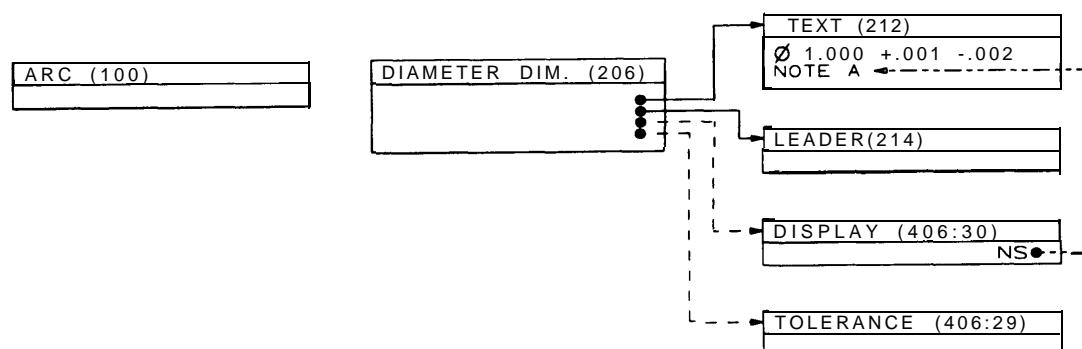
#### C A D S y s t e m E n t i t i e s :



#### C a t e g o r y 1 -- M A N U A L :



#### C a t e g o r y 2 -- G E N E R A T I V E :



#### C a t e g o r y 3 -- A S S O C I A T I V E :

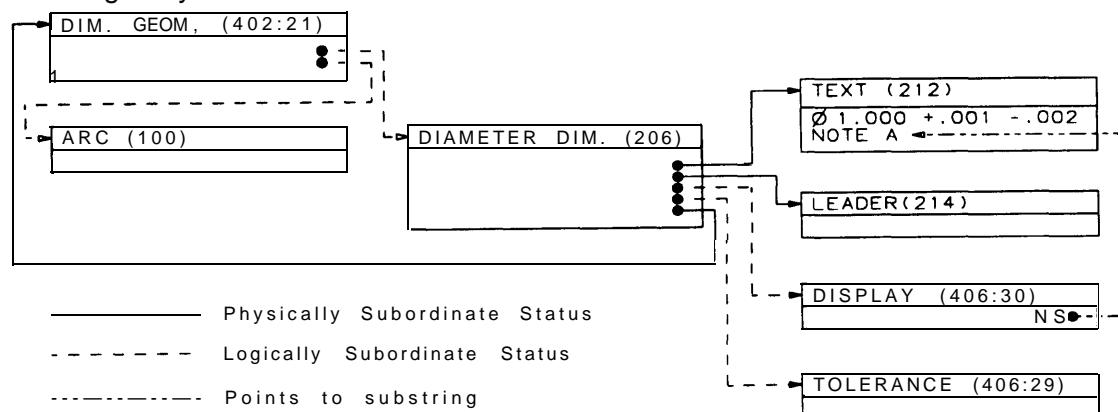


Figure 10. Entity Usage According to System Category

### 3.6 Structure Entities

**3.6.1 Entity Types** The following structure entities are defined in this Specification:

Entity Type Number	Entity Type
0	Null
132	Connect Point
134	Node
136	Finite Element
138	Nodal Displacement and Rotation
146	‡Nodal Results
148	‡Element Results
302	Associativity Definition
304	Line Font Definition
306	MACRO Definition
308	Subfigure Definition
310	Text Font Definition
312	Text Display Template
314	Color Definition
316	‡Units Data
320	Network Subfigure Definition
322	Attribute Table Definition
402	Associativity Instance
404	Drawing
406	Property
408	Singular Subfigure Instance
410	View
412	Rectangular Array Subfigure Instance
414	Circular Array Subfigure Instance
416	External Reference
418	Nodal Load/Constraint
420	Network Subfigure Instance
422	Attribute Table Instance
600-699	Implementor specified MACRO Instance
10000-99999	Implementor specified MACRO Instance

The following sections describe some of the uses of the structure entities.

**3.6.2 Subfigures.** Subfigures have been provided to enable the use of a collection of entities many times within the model at various locations, orientations, and Scales. In some cases, the collection itself is specified by a Subfigure Definition Entity (Type 308), and each placement of the collection is specified by a Singular Subfigure Instance Entity (Type 408). The Network Subfigure Definition (Type 320) and Instance (Type 420) Entity pair is similar in concept but has some special features to accommodate the notion of connect points in a network. (Section 3.6.3 provides additional information about network subfigure.) In other cases, a Rectangular Array (Type 412) or a Circular Array (Type 414) Subfigure Instance Entity specifies a base entity to be copied according to one of these two overall patterns.

Subfigure may be nested. For example, a Subfigure Definition Entity may include a Singular ECO630 Subfigure Instance Entity as one entity in its collection. Figure 11 illustrates subfigure nesting. A

similar interpretation of Depth applies also to the Network Subfigure Definition and Instance Entity pair. In these cases, the X,Y,Z location and the scale factor(s) in the Subfigure Instance Entity help locate the Subfigure Definition Entity in the definition space of the referring Subfigure Definition Entity instead of in model space.

Thus, the processing sequence in these cases is as follows: Each entity in the subfigure definition is operated upon by its defining matrix and translation vector. Each entity is now located within the definition space of the Subfigure Definition Entity. Then, the defining matrix and translation vector of the Subfigure Definition Entity are applied. The entity collection of the Subfigure Definition Entity is now located in the definition space of the Subfigure Instance Entity. Next, the scale factor(s) located in the parameter data of the Subfigure Instance Entity is (are) applied. This results in a scaling about the origin of the definition space of the Subfigure Instance Entity. Next, the defining matrix and translation vector of the Subfigure Instance Entity are applied. This locates the scaled entities either in model space or in the definition space of another Subfigure Definition Entity. Finally, the X,Y,Z translation data located in the parameter data of the Subfigure Instance Entity is applied. Note that this translation data can be relative either to model space or to the definition space of a Subfigure Definition Entity. The latter case occurs when the Subfigure Instance Entity is referenced by another entity. ECO630

The above processing sequence requires that the Transformation Matrix Entity ([Type 124](#)) referenced by the instancing entity shall not be applied to: ECO637

- the X, Y, Z translation data for the Singular Subfigure Instance Entity ([Type 408](#)),
- the X, Y, Z translation data for the Network Subfigure Instance Entity ([Type 420](#)),
- the X, Y, Z coordinate data for the Rectangular Array Subfigure Instance Entity ([Type 412](#))
- the X, Y, Z coordinate data for the Circular Array Subfigure Instance Entity ([Type 414](#)).

**3.6.3 Connectivity.** The following file structure shall be used to define logical (and the location for physical) connections between objects. ECO630

A formed connection between two or more objects requires the data to represent the following:

1. the exact location of each connection point;
2. the flow path formed and its identification (if any);
3. the physical connection between the objects (if any).

These objects may include electrical or mechanical components such as transistors, pipes and valves, or air conditioning ductwork. Each connection formed defines a flow path between the objects, allowing a fluid (electricity, water, or air) to flow from one object to another. The Network Subfigure Definition and Instance Entities are used to represent the objects to be connected. The Connect Point Entity ([Type 132](#)) is used to represent the exact location of connection. The term “link” will refer to the logical representation of the flow path (signal) formed, and “flow-name” will refer to the flow path identifier. The term “join” will refer to the file entity or entities which represent the physical connection (geometries between the items). ECO630

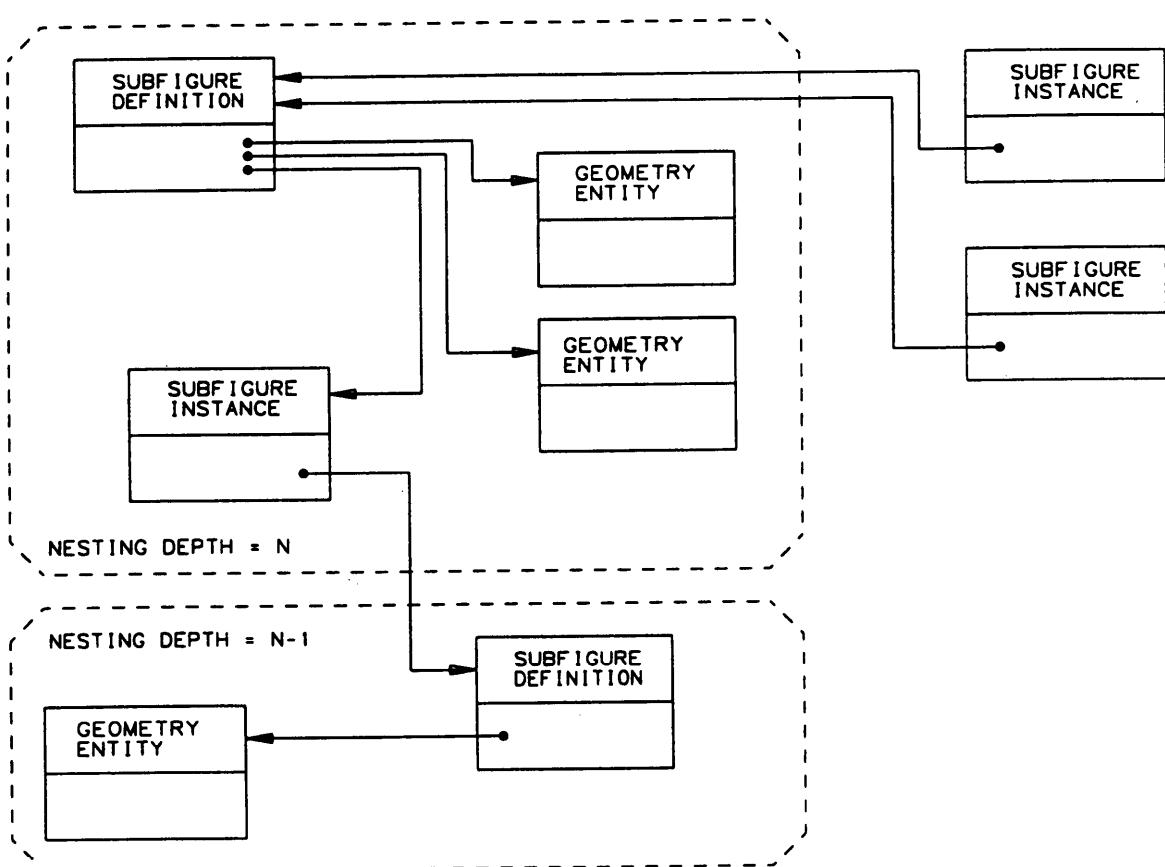


Figure 11. Subfigure Structures

**3.6.3.1 Connectivity Entities.** The entities used to implement connectivity include the Network Subfigure Definition ([Type 320](#)) and Network Subfigure Instance ([Type 420](#)) Entities, the Flow Associativity Entity ([Type 402, Form 18](#)), the Piping Flow Associativity Entity ([Type 402, Form 20](#)), the Connect Point Entity ([Type 132](#)), and the Text Display Template Entity ([Type 312, Form 0, Form 1](#)).

**3.6.3.2 Entity Relationships.** A flow path (signal) may be formed between items by a link which references the items' connect points (entities) to be related. This creates an Associativity among the connect points and thus the entities connected. The flow name may be used to uniquely identify the particular signal formed. The join may be used to provide a graphical representation of the flow path. In electrical applications, the join will be represented by geometry entities such as lines, arcs, subfigures, copious data, etc. In a piping application, an example of a join represented might be the section of pipe between a valve and a tank. The logical constructs (link and flow name) shall be implemented by the Flow Associativity Entity or by the Piping Flow Associativity Entity which in turn identifies (by pointer) the entities which form the join.

In electrical applications, for example, the items to be connected are components (*i.e.*, resistor, 16-pin dual in-line package, *etc.*), or integrated circuit cells, represented and instanced by network subfigures. Each pin (or signal port) is a potential connection point in a flow path, thus each network subfigure has a connect point for each pin (or port). When such a subfigure is instanced, its connect points must also be instanced. An instanced connect point, when added to a flow path, is different from its definition which shall not be a member of any flow path. See [Figure 12](#) for the basic entity relationships.

**3.6.3.3 Information Display.** The network subfigures, representing electrical components, for example, often contain text describing the component and its pins. The Text Display Template Entity ([Type 312](#)) allows text embedded in another entity to be displayed without redundant specification of the text string. The Text Display Template Entity may be used to display reference designators and pin numbers. The absolute form, within a network subfigure, is recommended for the reference designator text. Each instance of the subfigure need only supply the text string. The pin number can be represented in the incremental form. All the pin numbers on a given side of a package outline having the same X, Y, and Z offsets relative to the pin whose number is to be displayed may use the same text display template definition.

**3.6.3.4 Additional Considerations.** The situation is exactly the same for both logical and physical product representations. The only differences arise in the subfigure and join entities used. One file may contain both schematic and physical representations of a product. The Flow Associativity Entity ([Type 402, Form 18](#)) contains a Type flag to indicate the connection type (logical or physical). In this case, one Flow Associativity Entity would represent the logical connection and a second the physical connection. The two associativities would be related by the pointers provided in the Flow Associativity Entity.

**3.6.4 External Reference Linkage.** Linkages between entities can occur not only within a file, but also between entities in different files. Two entities are used in a referencing file to establish this linkage: the External Reference Entity ([Type 416](#)) which provides the actual linkage to the referenced file, and the External Reference File List Property Entity ([Type 406, Form 12](#)) which provides a list of the names of all the files referenced. Further, only directly referenced files shall be in this property's parameter list. Each file name listed in the parameter data of this property shall match the name in the fourth global parameter of a referenced file.

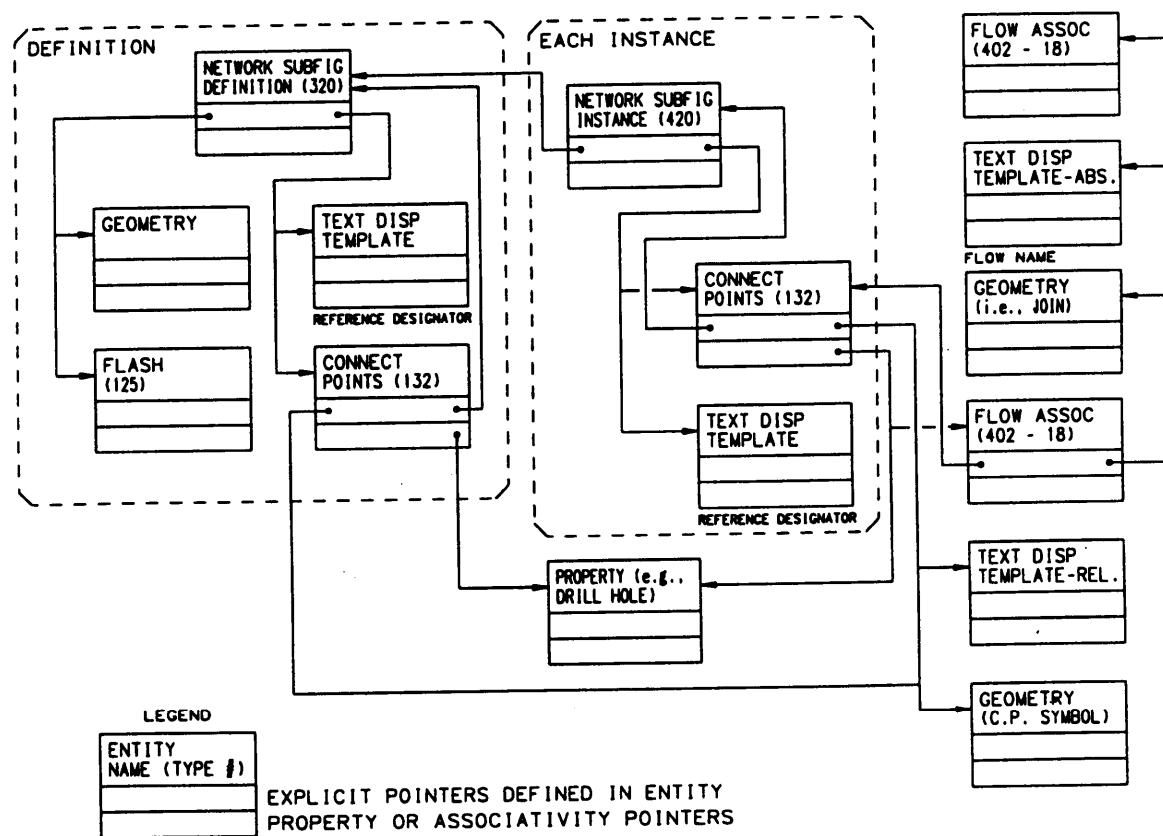


Figure 12. General Connectivity Pointer Diagram

An External Reference File Index Associativity Entity ([Type 402, Form 12](#)) is required in the ECO630 referenced file when the [Type 416, Form 0 or 2](#) is used (*i.e.*, more than one referenced entity in the referenced file). This Associativity provides a directory to the referenced entities within its file, and both relate a symbolic name to the directory entry of an entity within the file ([see Figure 13](#)). All symbolic names used within a set of files linked by references shall be unique. Definitions may be nested, and a symbolic name used need be unique only on the nesting level on which it is used.

Because of the intricacy of the linkages, an example follows (refer to [Figure 13](#)). Consider a file ECO630 containing a Subfigure Instance Entity ([Type 408](#)). The first item in its parameter data record is a pointer to the subfigure definition entry in the Directory Entry Section of the file. In the case that the Subfigure Definition Entity ([Type 308](#)) is to be contained in a library file, this first parameter is a pointer to an External Reference Entity ([Type 416](#)). That External Reference Entity will have in its parameter data record the name of the file which is to contain the definition and the symbolic name of the definition itself. The file name is the fourth global parameter in the referenced file. The symbolic name is a string which identifies the appropriate referenced definition.

In the case of a library file which contains several definitions, each of which are expected to be ECO630 referenced by other files, the External Reference File Index Associativity Entity ([Type 402, Form 12](#)) provides a “table of contents” of the available definitions in the file. The parameter data record of this Associativity contains pairs of data: the symbolic name associated with the definition (the same one used in the [Type 416](#) entity’s parameter data record), and a pointer to the directory entry record which contains the desired definition.

In the case that the entire external file is to be included (*i.e.*, a super-subfigure), Form 1 of the ECO630 [Type 416](#) entity is used which does not contain a symbolic name in the parameter data record. In a similar manner, the referenced file does not contain an associativity [Type 402, Form 12](#) entity; it is unneeded, since the entire file is to be used.

In either case, the External Reference File List Property Entity ([Type 406, Form 12](#)) will be found ECO630 in the referencing file. The parameter data record contains a simple list of the file names of the various external files referenced by this file. Once again, the file name used is that in the fourth global parameter of the referenced file. Note that this list contains only those file names that are directly referenced; it gives no information about files which may be referenced in turn by those files used by this file.

A limitation of external referencing is that the back pointers (in the “back pointers to associativities” ECO630 addition to an entity’s parameters) cannot be used. If a pointer is required in each direction, separate external reference mechanisms must exist in each file (*e.g.*, the double linkage between files A and B in [Figure 13](#)).

A preprocessor implementor should use the external reference mechanism with care because of the burden placed on the postprocessor.

**3.6.5 Drawings and Views.** This Specification provides a mechanism for associating models and drawings so that there is consistency between them. The mechanism is based on the existing practices of some CAD/CAM graphic systems to define the views of a part on a drawing in terms of a single three-dimensional (3-D) model.

The Drawing Entity ([Type 404](#)) specifies a drawing of a given size within a special drawing space ECO630 coordinate system. This entity can refer to one or more View Entities ([Type 410](#)) which will specify the projection from 3-D model space to the two-dimensional drawing space. Annotation entities such as dimensioning can be defined directly in the drawing coordinate system or can be defined in the 3-D model space and then be included in individual views. More than one drawing entity may be included in a file.

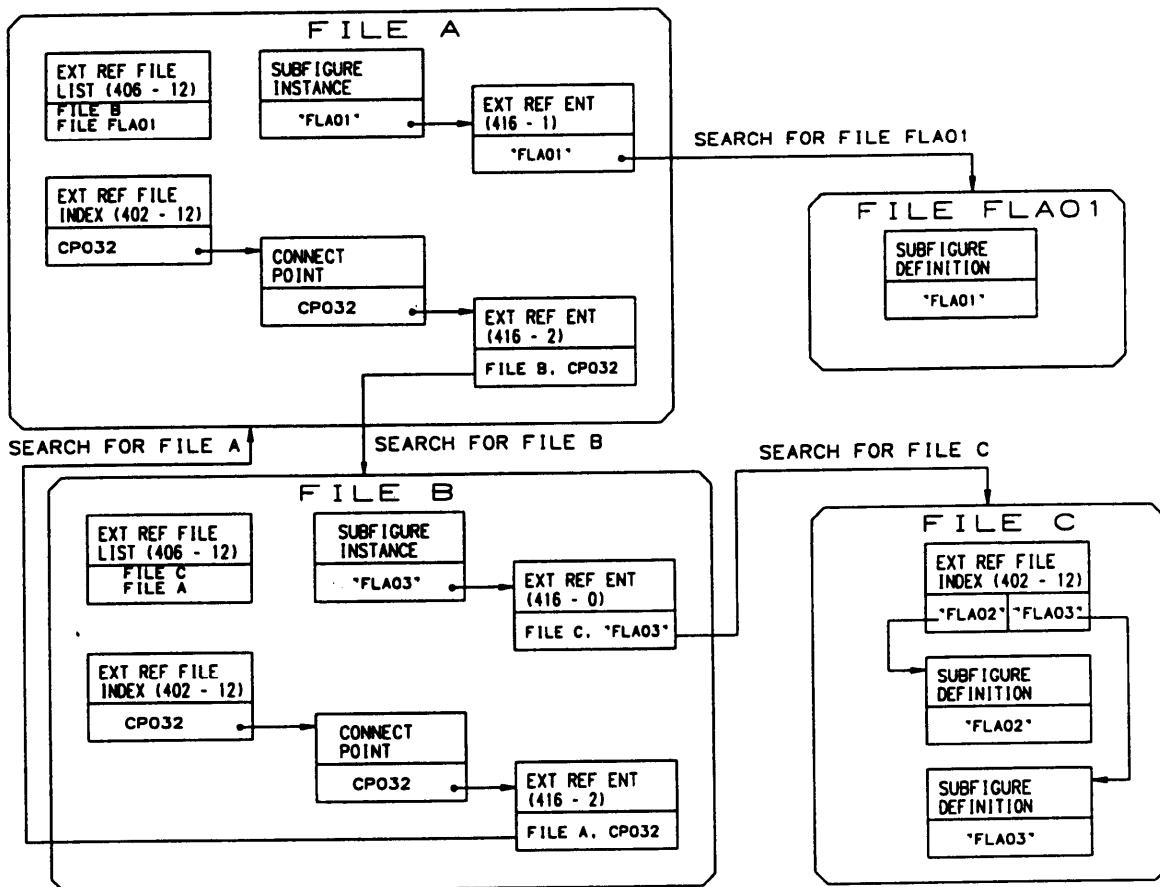


Figure 13. External Linkages

In addition to being used in conjunction with the Drawing Entity, the view-specific display of parts of the model can be used to communicate hidden lines, phantom lines, etc.

Graphic systems which do not have the ability to define drawings and views of models in this manner ECO630 are not required to preprocess this construct into a file, but all systems with postprocessors must be able to process the Drawing and View Entities in received files.

To represent that a defined view is not displayed, the preprocessor shall set the Blank Status Flag for the view to 01 (blanked).

**3.6.6 Finite-Element Modeling.** This section defines the entities and their relationships (*i.e.*, ECO630 pointers) required to support the finite-element modeling (FEM) application and to display results of analysis on those systems which support finite element analysis postprocessing.

The entities available for exchanging FEM data are illustrated in [Figures 14](#) and [15](#). The left side of [Figure 14](#) illustrates the relationships between the entities that define the model's parametric attributes. The right side illustrates the addition of the analysis results. [Figure 15](#) illustrates the FEM entities used to define an example beam structure with accompanying material properties, a load, and a constraint. The entities defined in support of such analysis are the Element Entity ([Type 136](#)), Node Entity ([Type 134](#)), Nodal Load/Constraint Entity ([Type 418](#)), Tabular Data Property Entity ([Type 406](#), [Form 11](#)), Nodal Results Entity ([Type 146](#)) and Element Results Entity ([Type 148](#)).

The Element Entity ([Type 136](#)) defines a finite element to be used in the finite-element model. ECO630 Several finite elements are defined in this Specification. Examples of an element are: BEAM, CTRIA, and DAMP. Specifically, the Element Entity specifies the topology type, number of nodes, and the element-type name. Pointers locate the defining nodes and the material properties of the element. The connectivity of the nodes is implied in the order of the contained pointers and topology type.

The Node Entity ([Type 134](#)) defines the grid points or nodes of the element. It contains the spatial values that define the node and a pointer to the coordinate system upon which it is defined. ECO630

The Nodal Load/Constraint Entity ([Type 418](#)) is an entity that points to a node. It defines either a load or a constraint as applied to that node. It also contains a pointer to General Note Entities ([Type 212](#)) that define the load case. Property pointers reference the Tabular Data Property Entity ([Type 406](#), [Form 11](#)) that contains the values of the load or constraint vector. ECO630

The Tabular Data Property Entity ([Type 406](#), [Form 11](#)) contains the material property data of the elements and the load and constraint data as required. ECO630

The Nodal Results Entity ([Type 146](#)) is used to communicate nodal finite-element analysis results data. It contains analysis results at FEM nodes that are independent of the FEM elements that are attached to them. (The Element Results Entity ([Type 148](#)) should be used if the analysis results data are dependent on FEM elements.) The Nodal Results Entity is intended to supercede the old Nodal Displacement and Rotation Entity ([Type 138](#)), as it permits far greater flexibility in the transfer of nodal results. ECO630

The Element Results Entity ([Type 148](#)) is used to communicate FEM element results that vary within a FEM element. The data communicated may be results at various layers within the FEM element: at the FEM elements and nodes, at the FEM centroid, at the FEM element Gauss points, or at any combination of these locations. ECO630

For example, consider the extrapolated stress values at the nodes of several quadratic, plane-stress FEM elements. There is no guarantee that the nodal values of stress will be identical for adjacent FEM elements at common nodes. There are at least as many possible FEM element result values

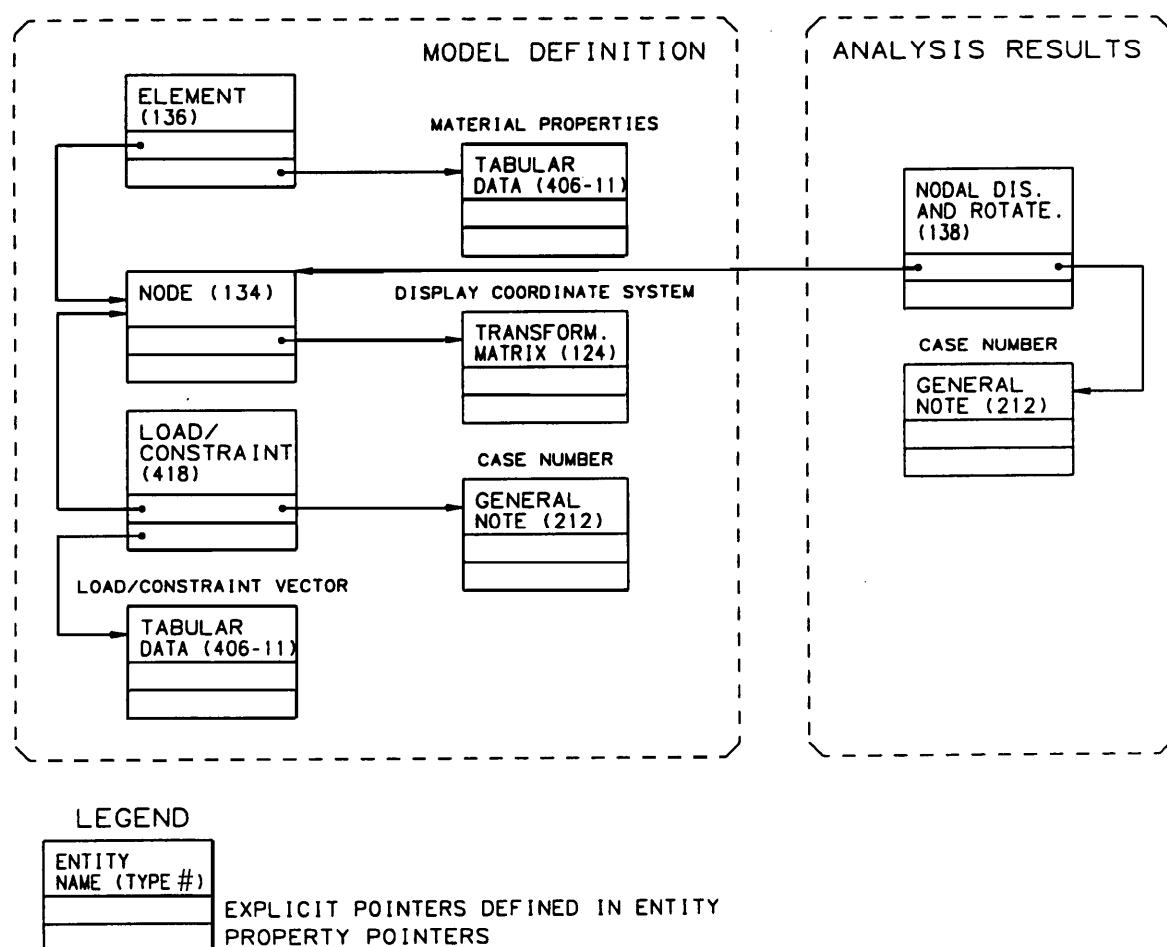


Figure 14. Finite Element Modeling File Structure

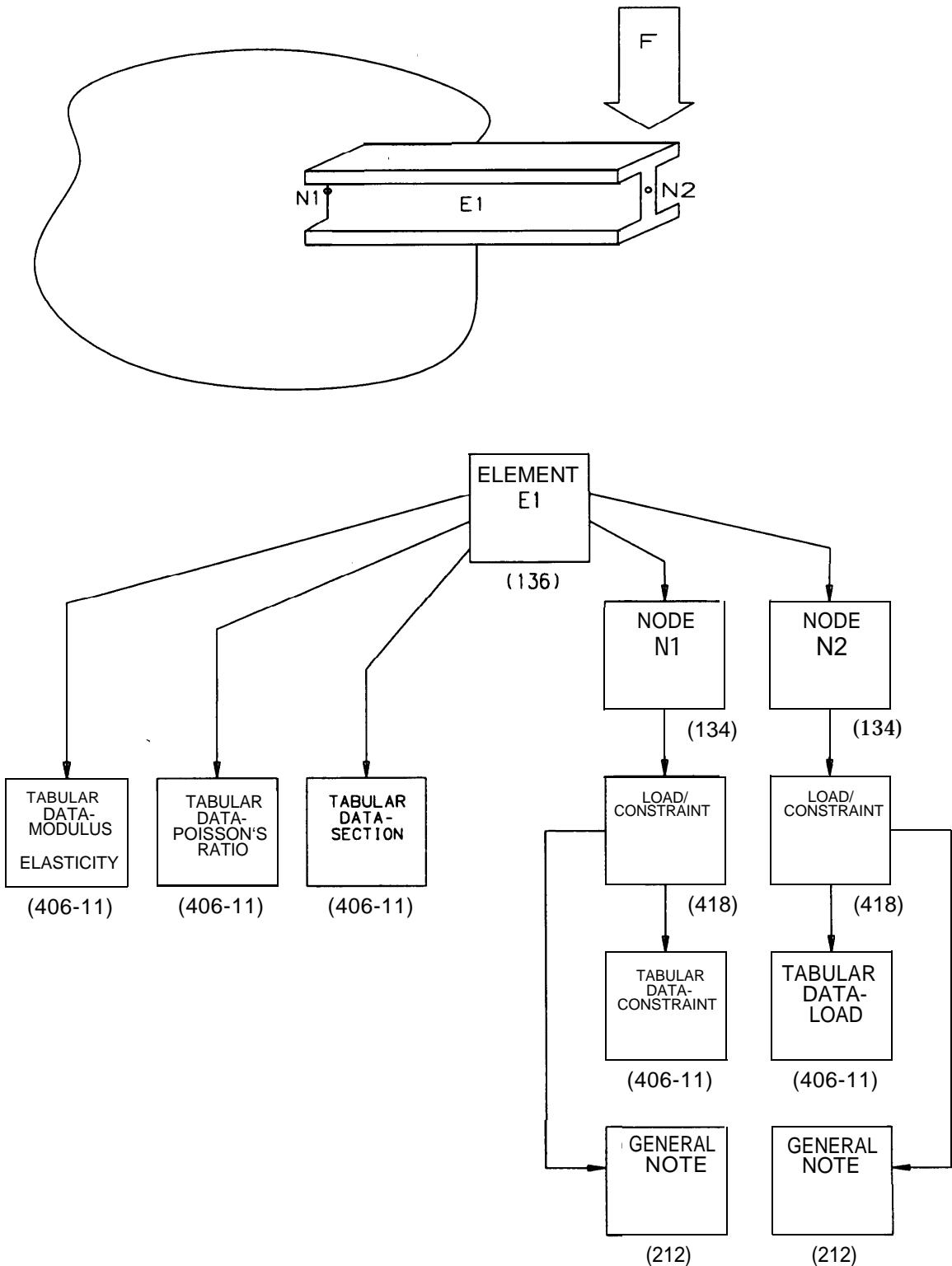


Figure 15. Finite Element Modeling Logical Structure

as there are finite elements that contain common nodes in their topologies. These data are different from the results data expressed at the same node in the Nodal Results Entity.

**3.6.7 Attribute Tables.** An attribute table (see Sections 4.79 and 4.141) is a collection of attribute definitions and values in the form of a single row or table. The structure consists of an Attribute Table Definition Entity ([Type 322](#)), where each attribute is defined by a name, a data-type, and a count. The attribute values are either supplied as part of the attribute definition, or instanced using the Attribute Table Instance Entity ([Type 422](#)). One or more Attribute Table Instance Entities may point to the Attribute Table Definition Entity using the third field of their Directory Entry.

Three types of Attribute Table Definition Entities and two types of Attribute Table Instance Entities ECO630 are defined. The Attribute Table Definition Entity can have: (1) attribute definitions only, (2) attribute definitions followed immediately by the attribute values, or (3) attribute definitions followed by attribute values with each value followed by a pointer to a Text Display Template Entity ([Type 312](#)). The Attribute Table Instance Entity can store: (1) a single row of attribute values, or (2) a table of rows of attribute values, stored in row-major order.

## 4. Entity Types

### 4.1 General

This Chapter defines the entity types available to be used in the entity-based product definition ECO630 file. Descriptions of the various directory entry fields were given in [Section 2.2.4.4](#). The meanings of these fields remain the same across all entities. In this Chapter, those entities making extended use of Field 15 in the directory entry (Form Number) are indicated, and the various options are listed. The parameter data record for each entity is also described in this Chapter. The fields for this record vary from entity to entity.

Beginning with Version 5.3 of this Specification, those entities whose testing is not yet complete are ECO630 marked with the label “‡” and a reference to [Section 1.9](#). Table 5 lists the untested entities.

Table 5. Untested Entities

Entity Type Number	Form	Entity Type
123		Direction
136		Finite Element (additional topologies)
141		Boundary
143		Bounded Surface
146	0-34	Nodal Results
148	0-34	Element Results
182		Selected Component
186		Manifold Solid B-Rep Object
190		Plane Surface
192		Right Circular Cylindrical Surface
194		Right Circular Conical Surface
196		Spherical Surface
198		Toroidal Surface
204		Curve Dimension
212	All	Additional General Note Fonts: OCR-B Text Font Kanji Text Font
213		New General Note
216	0-2	Linear Dimension (Form Numbers)
218	1	Ordinate Dimension (Form Number)
222	1	Radius Dimension (Multiple Leader)
228	1-3	General Symbol (Form Numbers)
(continued)		

Table 5 Untested Entities (continued)

Entity Type Number	Form	Entity Type
230	0	Sectioned Area (Pattern Hatches)
230	1	Sectioned Area (Form Number)
306		MACRO
316		Units Data
402	19	Segmented Views Visible Associativity
402	20	Piping Flow Associativity
402	21	Dimensioned Geometry Associativity
404	1	Drawing with Rotated Views
406	18	Intercharacter Spacing Property
406	19	Line Font Property
406	20	Highlight Property
406	21	Pick Property
406	22	Uniform Rectangular Grid Property
406	23	Associativity Group Type Property
406	24	Level to PWB Layer Map Property
406	25	PWB Artwork Stackup Property
406	26	PWB Drilled Hole Property
406	27	Generic Data Property
406	28	Dimensioned Units Property
406	29	Dimension Tolerance Property
406	30	Dimension Display Data Property
406	31	Basic Dimension Property
406	32	Drawing Sheet Approval Property
406	33	Drawing Sheet ID Property
406	34	Underscore Property
406	35	Overscore Property
406	36	Closure Property
410	1	View (Perspective)
416	3	External Reference (Form Number)
416	4	External Reference (Form Number)
502		Vertex
504		Edge
508		Loop
510		Face
514		Shell

Potential implementors are warned that significant changes may occur to UNTESTED entities as they are tested and validated. Please communicate any test results or problems to the IGES/PDES Organization's Administrative Office.

## **4.2 Null Entity (Type 0)**

The Null Entity (Type 0) is intended to be ignored by a processor. It may contain an arbitrary amount of data in its PD data. When encountered by a processor, this entity shall be skipped over and not processed. Any value is permitted in a DE field labeled < n.a. > and may be ignored by a postprocessor.

This entity is useful when editing a file. By changing the entity type number of an entity in a file to 0, one ensures that the entity will not be processed. Thus, the replacement of an entity in a file can easily be done by adding the replacement entity to the end of the DE and PD Sections and changing the replaced entity type number to 0.

When editing a file to create a Null Entity, care should be taken to change both Entity Type Number Fields in the DE Section, as well as the first field of the first PD line.

### **Directory Entry**

(1) Entity Type Number 0	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number *****: *****:	(10) Sequence Number D #
(11) Entity Type Number 0	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number < n.a. >	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

## 4.3 CIRCULAR ARC ENTITY (TYPE 100)

### 4.3 Circular Arc Entity (Type 100)

A circular arc is a connected portion of a circle which has distinct start and terminate points. The ECO630 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.

A circular arc determines unique arc endpoints 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 direction. (See Section 3.2.4.) This convention serves to distinguish the desired circular arc from its complementary arc (complementary with respect to the parent circle).

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.

If required, the default parameterization is:

ECO630

$$C(t) = (X_1 + R \cdot \cos t, Y_1 + R \cdot \sin t, Z_T), \quad t_2 \leq t \leq t_3,$$

where  $Z_T$  is the coordinate of a point along the  $Z_T$  axis, for  $i = 2$  and  $3$ ,

$$R = \sqrt{(X_i - X_1)^2 + (Y_i - Y_1)^2},$$

$t_i$  is such that

$$(R \cos t_i, R \sin t_i) = (X_i - X_1, Y_i - Y_1),$$

and

$$\begin{aligned} 0 &\leq t_2 &< 2\pi \\ 0 &\leq t_3 - t_2 &\leq 2\pi. \end{aligned}$$

Examples of the Circular Arc Entity are shown in Figure 16. In Example 1 of Figure 16 the solid ECO630 arc is a full circle, and the start and terminate points are coincident. In Example 2 of Figure 16, 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 start point listed in the parameter data entry would be B, and the terminate point would be A.

### 4.3 CIRCULAR ARC ENTITY (TYPE 100)

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
100	$\Rightarrow$	< n.a. >	#, $\Rightarrow$	#, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	??????**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16)	Reserved	Reserved	(18) Entity Label	(19) Entity Subscript
100	#	#, $\Rightarrow$	#	0				#	D # + 1

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>	
1	ZT	Real	Parallel ZT displacement of arc from XT, YT plane	EC0630
2	X1	Real	Arc center abscissa	
3	Y1	Real	Arc center ordinate	
4	X2	Real	Start point abscissa	
5	Y2	Real	Start point ordinate	
6	X3	Real	Terminate point abscissa	
7	Y3	Real	Terminate point ordinate	

Additional pointers as required (see Section 2.2.4.5.2).

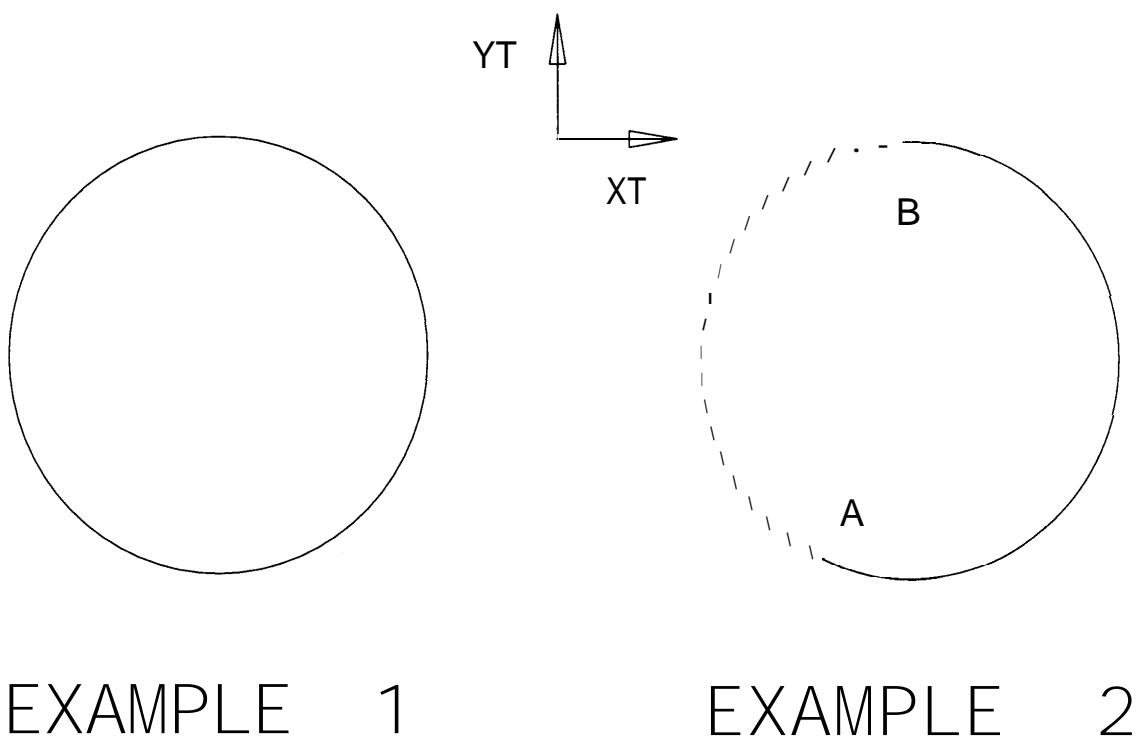


Figure 16. F100X.IGS Examples Defined Using the Circular Arc Entity

#### 4.4 Composite Curve Entity (Type 102)

A composite curve is a continuous curve that results from the grouping of certain individual constituent entities into a logical unit.

A composite curve is defined as an ordered list of entities consisting of point, connect point, and parameterized curve entities (excluding the Composite Curve Entity). 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 the same as the order of the listing of these pointers.

Each constituent entity has its own transformation matrix and display attributes. Each constituent entity may have text or properties associated with it. Because the constituent entities are subordinate to the composite entity, the Subordinate Entity Switch (digits 3–4 in Directory Entry Field 9) of each constituent entity shall indicate a physical dependency.

A composite curve is a directed curve, having a start point and a terminate point. The direction of the composite curve is determined 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.

The Point and Connect Point Entities are included as allowable entity types so that properties or general notes can be attached to either the start point or the terminate point of any constituent curve entities in the defining list.

A logical connection relationship can be indicated by having two composite curves or a composite curve and a network subfigure reference the Connect Point Entity. For the special case of the logical connection of a connect point on one subfigure instance to a connect point on another subfigure instance, a composite curve is allowed whose list contains only two Connect Point Entities with no intervening curve entity. In this case, the instance of the Composite Curve Entity is not a curve in the normal sense; it is not continuous and has no arc length. This usage is permitted in certain applications (*e.g.*, FEM and AEC). There are certain restrictions regarding the use of the point entity in a composite entity. They are:

1. Two Point or Connect Point Entities cannot appear consecutively in the defining list unless they are the only entities in the composite curve. Such composite curves used as logical connectors shall have an Entity Use Flag value = 04 (logical/positional). ECO642
2. If a Point or Connect Point Entity and a curve entity are adjacent in the defining list, then the coordinates of the Point or Connect Point Entity must agree with the coordinates of the terminate point of the curve entity whenever the curve entity precedes the Point or Connect Point Entity, and must agree with the coordinates of the start point of the curve entity whenever the curve entity follows the Point or Connect Point Entity.
3. A composite curve cannot consist of a single Point Entity or a single Connect Point Entity. ECO630

If required, the default parameterization of the composite curve is obtained from the parameterization of the constituent curves as defined below. As point and connect point entities do not contribute to the parameterization of a composite curve, they are not considered in this definition.

#### 4.4 COMPOSITE CURVE ENTITY (TYPE 102)

Let

- $C$  be the composite curve;
- $N$  be the number of constituent curves ( $N \geq 1$ );
- $CC(i)$  be the  $i$ -th constituent curve, for each  $i$  such that  $1 \leq i \leq N$ ;
- $PS(i)$  be the parametric value of the start of  $CC(i)$ ;
- $PE(i)$  be the parametric value of the end of  $CC(i)$ ;
- $T(0)$  be 0.0;
- $T(i)$  be  $\sum_{j=1}^i (PE(j) - PS(j))$ ,  
for each  $i$  such that  $1 \leq i < N$ ;

then

1. The parametric values of  $C$  range from  $T(0)$  to  $T(N)$ ; and
2.  $C(u) = CC(i) (u - T(i-1) + PS(i))$ , where  $u$  is a parametric value such that  $T(i-1) \leq u \leq T(i)$ .

A composite curve consisting solely of Point and/or Connect Point Entities is not given a parameterization. ECO630

As an example of a parameterization of a Composite Curve Entity, let  $N = 3$ , and for each  $i$  such that  $1 \leq i \leq 3$ , let  $CC(i)$  be the  $i$ -th constituent curve of the composite curve  $C$ . Assume the parametric values of the start and end points of each  $CC(i)$  are given by the table:

$i$	$PS(i)$	$PE(i)$
1	0.0	0.4
2	3.3	3.5
3	0.0	0.3

Then  $T(0) = 0.0$ ,  $T(1) = 0.4$ ,  $T(2) = 0.6$ ,  $T(3) = 0.9$ , and the composite curve  $C$  is defined from 0.0 to 0.9. This situation is illustrated in [Figure 17](#).

The curve combining  $CC(1)$ ,  $CC(2)$ , and  $CC(3)$  represents the composite curve  $C$ .

An example of a composite curve and its parameterization is shown in [Figure 18](#).

**Directory Entry**

(1) Entity Type Number 102	(2) <b>Parameter Data</b> ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ?????????	(10) Sequence Number D #
(11) Entity Type Number 102	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

Note: When the Hierarchy is set to Global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, Color Number, Level, View, and Blank Status.

**Parameter Data**

ECO650

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1		Integer	Number of entities
2	DE(1)	Pointer	Pointer to the DE of the first constituent entity
:	:	:	
1+N	DE(N)	Pointer	Pointer to the DE of the last constituent entity

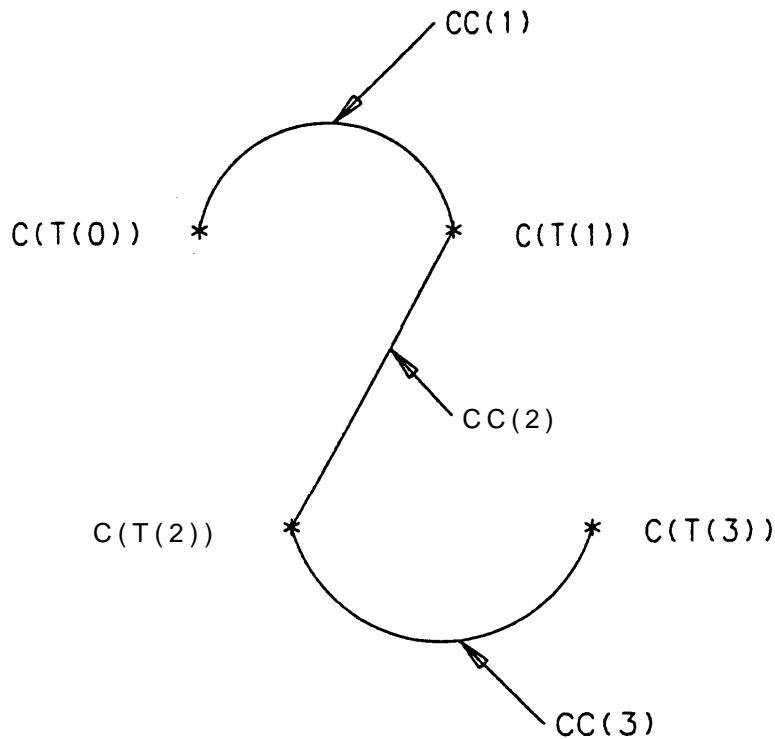


Figure 17. Parameterization of the Composite Curve

#### **4.4 COMPOSITE CURVE ENTITY (TYPE 102)**

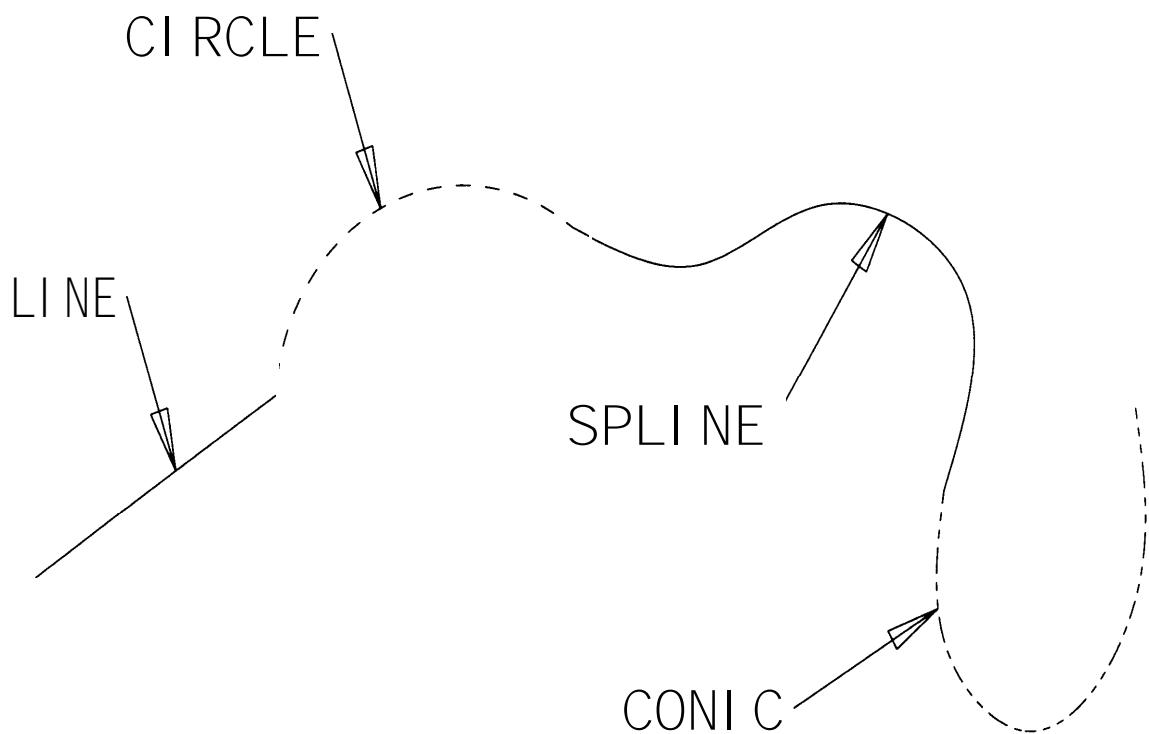


Figure 18. Example Defined Using the Composite Curve Entity

#### 4.5 Conic Arc Entity (Type 104)

A conic arc is a bounded connected portion of a conic curve which has distinct start and terminate ECO630 points. The parent conic curve is either an ellipse, a parabola, or a hyperbola. The definition space coordinate system is always chosen so that the conic arc lies in a plane either coincident with or parallel to the  $XT$ ,  $YT$  plane. Within such a plane, a conic is defined by the six coefficients in the following equation, where  $X_T, Y_T$  are the coordinates of a point in the  $XT$ ,  $YT$  plane:

$$AX_T^2 + BX_TY_T + CY_T^2 + DX_T + EY_T + F = 0$$

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

A conic arc determines unique arc endpoints. A conic arc is defined within definition space by the six ECO630 coefficients above and the two endpoints. By considering the conic arc endpoints to be enumerated and listed in an ordered manner, start point followed by terminate point, a direction with respect to definition space can be associated with the arc. In order for the desired elliptical arc to be distinguished from its complementary elliptical arc, the direction of the desired elliptical arc shall be counterclockwise. (See Section 3.2.4) 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.

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.

The definitions of the terms ellipse, parabola, and hyperbola are given in terms of the quantities  $Q_1$ ,  $Q_2$ , and  $Q_3$ . These quantities are:

$$\begin{aligned} Q_1 &= \begin{vmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{vmatrix} \\ Q_2 &= \begin{vmatrix} A & B/2 \\ B/2 & C \end{vmatrix} \\ Q_3 &= A + C \end{aligned}$$

A parent conic curve is:

- An ellipse if  $Q_2 > 0$  and  $Q_2Q_3 < 0$ .
- A hyperbola if  $Q_2 < 0$  and  $Q_2 \neq 0$ .
- A parabola if  $Q_2 = 0$  and  $Q_1 \neq 0$ .

An example of each type of conic arc is shown in [Figure 19](#).

Those entities which can be represented as various degenerate forms of a conic equation (e.g., point and line) shall not be put into the Entity [Type 104](#); more appropriate entity types exist for these forms.

Because of the numerical sensitivity of the implicit form of the conic description, conies shall be put into a standard position in definition space. A Conic Arc Entity is said to be in a standard position in definition space provided each of its axes is parallel to either the  $XT$  axis or  $YT$  axis and provided it is centered about the  $ZT$  axis. For a parabola, the origin is the vertex. The conic is

moved from this position in definition space to the desired position in space with a Transformation Matrix Entity ([Type 124](#)).

The form number shall be regarded as purely informational by a postprocessor. Further details may be found in [Appendix C](#).

If required, the default parameterization is: ( $Z_t$  is the coordinate of a point along the ZT axis.)

### **Parabola**

If  $A$  and  $E \neq 0.0$  and  $X_1 < X_2$ ,

$$C(t) = (t, -(A/E)t^2, Z_T), \quad t_1 \leq t \leq t_2,$$

where, for  $i = 1$  and  $2$ ,  $t_i = X_i$ . If  $X_2 < X_1$ ,

$$C(t) = (-t, -(A/E)t^2, Z_T), \quad t_1 \leq t \leq t_2,$$

where, for  $i = 1$  and  $2$ ,  $t_i = -X_i$ .

If  $C$  and  $D \neq 0.0$  and  $Y_1 < Y_2$ ,

$$C(t) = (-(C/D)t^2, t, Z_T), \quad t_1 \leq t \leq t_2,$$

for  $i = 1$  and  $2$ ,  $t_i = Y_i$ . If  $Y_2 < Y_1$  then

$$C(t) = (-(C/D)t^2, -t, Z_T), \quad t_1 \leq t \leq t_2,$$

where, for  $i = 1$  and  $2$ ,  $t_i = -Y_i$ .

### **Ellipse**

For the ellipse,

$$t_1 \leq t \leq t_2,$$

where  $a = \sqrt{-F/A}$ ,  $b = \sqrt{-F/C}$ , and, for  $i = 1$  and  $2$ ,  $t_i$  is such that

$$\begin{aligned} (a \cos t_i, b \sin t_i, Z_T) &= (X_i, Y_i, Z_T) \\ 0 \leq t_1 &\leq 2\pi \\ 0 \leq t_2 - t_1 &\leq 2\pi. \end{aligned}$$

### **Hyperbola**

If  $F \cdot A < 0.0$  and  $F \cdot C > 0.0$ , let  $a = \sqrt{-F/A}$  and  $b = \sqrt{F/C}$ . For  $i = 1$  and  $2$ ,  $t_i$  is such that

$$\begin{aligned} (a \sec t_i, b \tan t_i, Z_T) &= (X_i, Y_i, Z_T) \\ -\pi/2 < t_1, t_2 &< \pi/2. \end{aligned}$$

If  $t_1 < t_2$ ,

$$C(t) = (a \sec t, b \tan t, Z_T), \quad t_1 \leq t \leq t_2;$$

if  $t_2 < t_1$ ,

$$C(t) = (a \sec(-t), b \tan(-t), Z_T), \quad -t_1 \leq t \leq -t_2.$$

If  $F \cdot A > 0.0$  and  $F \cdot C < 0.0$ , let  $a = \sqrt{F/A}$  and  $b = \sqrt{-F/C}$ . For  $i = 1$  and  $2$ ,  $t_i$  is such that

$$\begin{aligned} (a \tan t_i, b \sec t_i, Z_T) &= (X_i, Y_i, Z_T); \\ -\pi/2 < t_1, t_2 &< \pi/2. \end{aligned}$$

If  $t_1 < t_2$ ,

$$C(t) = (a \tan t, b \sec t, Z_v), \quad t_1 \leq t \leq t_2;$$

if  $t_2 < t_1$ ,

$$C(t) = (a \tan(-t), b \sec(-t), Z_v), \quad -t_1 \leq t \leq -t_2.$$

For the Conic Arc Entity, the form numbers are:

Form	Meaning
1	Parent conic curve is an ellipse (See Figure 19)
2	Parent conic curve is a hyperbola (See Figure 19)
3	Parent conic curve is a parabola (See Figure 19)

Note: Previous versions of this Specification permitted a form number of 0. This is now deprecated.

### Directory Entry

(1) Entity Type Number 104	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 104	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 1-3	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** Valid values of the Form Number are 1–3.

### Parameter Data

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

Additional pointers as required (see Section 2.2.4.5.2).

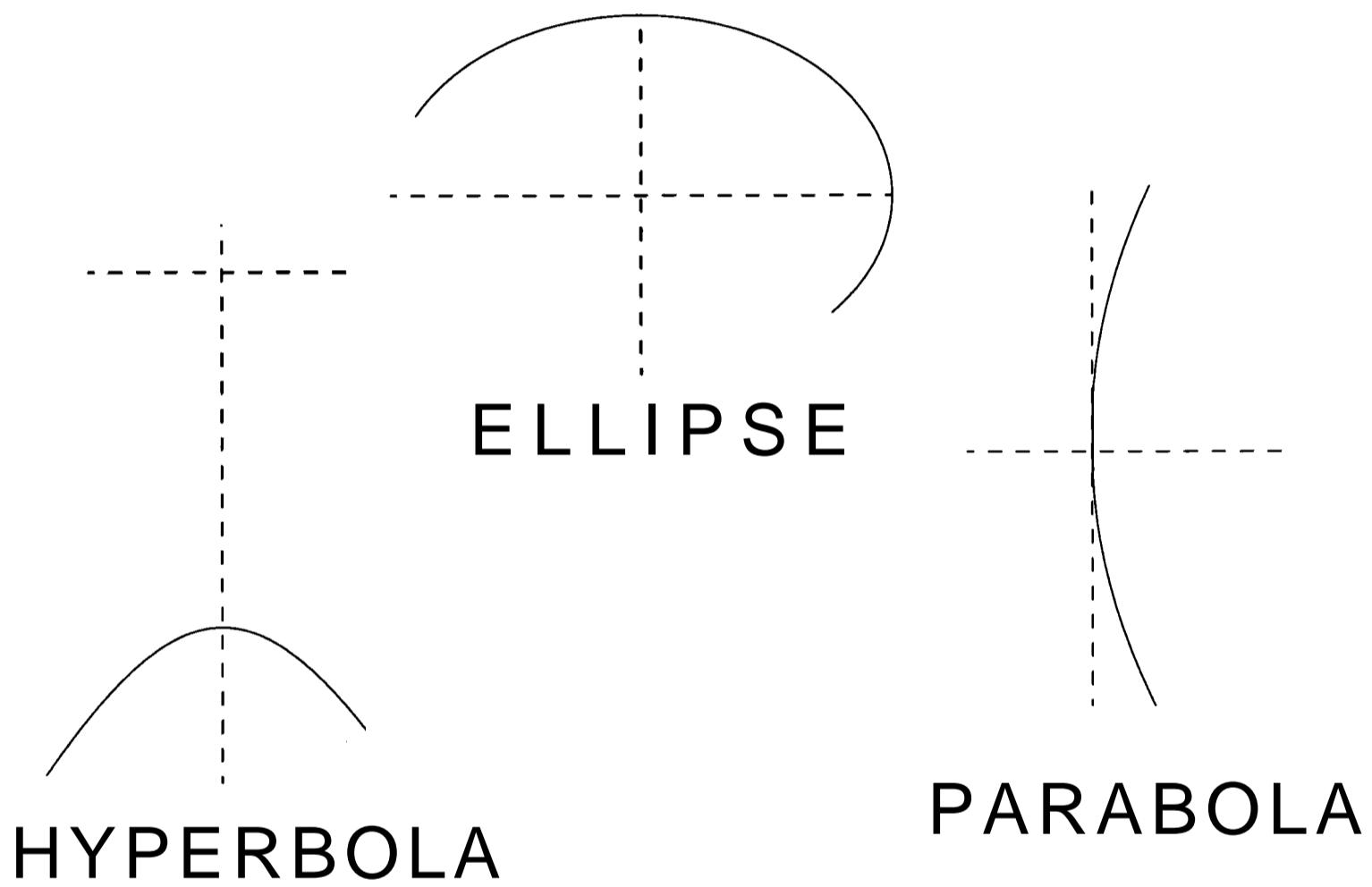


Figure 19. F104X.IGS Examples Defined Using the Conic Arc Entity

**4.6 Copious Data Entity (Type 106, Forms 1-3)**

ECO630

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 the first parameter data entry. The interpretation flag is abbreviated below by the letters IP.

Data points within definition space which lie within a single plane are specified in the form of XT, YT coordinate pairs. In this case, the common ZT value is also needed. Data points arbitrarily located within definition space are specified in the form of XT, YT, ZT coordinate triples. Data points within definition space which have an associated vector are specified in the form of sextuples; the XT, YT, ZT coordinates are specified first, followed by the i, j, k coordinates of the vector associated with the point. (Note that, for an associated vector, no special meaning is implicit.)

The Form numbers of a Copious Data Entity are as follows:

<b>Form</b>	<b>Meaning</b>
1	Data points in the form of coordinate pairs. All data points lie in a plane ZT= constant. (IP=1)
2	Data points in the form of coordinate triples (IP=2)
3	Data points in the form of sextuples (IP=3)
11	Data points in the form of coordinate pairs which represent the vertices of a planar, piecewise linear curve (piecewise linear string is sometimes used). All data points lie in a plane ZT=constant. (IP= 1)
12	Data points in the form of coordinate triples which represent the vertices of a piecewise linear curve (piecewise linear string is sometimes used) (IP=2)
13	Data points in the form of sextuples. The first triple of each sextuple represents the vertices of a piecewise linear curve (piecewise linear string is sometimes used). The second triple is an associated vector. (IP=3)
20	centerline Entity through points (IP=1)
21	Centerline Entity through circle centers (IP=1)
31	Section Entity Form 31 (IP=1)
32	Section Entity Form 32 (IP=1)
33	Section Entity Form 33 (IP=1)
34	Section Entity Form 34 (IP=1)
35	Section Entity Form 35 (IP=1)
36	Section Entity Form 36 (IP=1)
37	Section Entity Form 37 (IP=1)
38	Section Entity Form 38 (IP=1)
40	Witness Line Entity (IP=1)
63	Simple Closed Planar Curve Entity (IP=1)

Refer to the appropriate entity descriptions for descriptions of Forms other than 1, 2, or 3.

## 4.6 COPIOUS DATA ENTITY (TYPE 106, FORMS 1-3)

### Directory Entry

(1) Entity Type Number 106	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt;n.a.&gt;</i>	(5) Level #,⇒	(6) View 0,⇒	(7) Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 106	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 1-3	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	IP	Integer	Interpretation Flag 1 =x,y pairs, common z 2 =x,y,z coordinates 3 =x,y,z coordinates and i,j,k vectors
2	N	Integer	Number of n-tuples

For IP=1 (x,y pairs, common z), *i.e.*, for Form 1:

3	ZT	Real	Common z displacement
4	X(1)	Real	First data point abscissa
5	Y(1)	Real	First data point ordinate
:	:	:	
3+2*N	Y(N)	Real	Last data point ordinate

For IP=2 (x,y,z triples), *i.e.*, for Form 2:

3	X(1)	Real	First data point x value
4	Y(1)	Real	First data point y value
5	Z(1)	Real	First data point z value
:	:	:	
2+3*N	Z(N)	Real	Last data point z value

For IP=3 (x,y,z,i,j,k sextuples), *i.e.*, for Form 3:

3	X(1)	Real	First data point x value
4	Y(1)	Real	First data point y value
5	Z(1)	Real	First data point z value
6	I(1)	Real	First data point i value
7	J(1)	Real	First data point j value
8	K(1)	Real	First data point k value
:	:	:	
2+6*N	K(N)	Real	Last data point k value

Additional pointers as required (see [Section 2.2.4.5.2](#)).

**4.7 Linear Path Entity (Type 106, Forms 11-13)**

ECO630

The linear path is an ordered set of points in either 2- or 3-dimensional space. These points define a series of linear segments along the consecutive points of the path. The segments may cross, or be coincident with, each other. Paths may close; i.e., the first path point may be coincident with the last.

The linear path is implemented as three forms of the Copious Data Entity (Type 106). Form 11 is for 2-dimensional paths, [Form 12](#) is for 3-dimensional paths, and [Form 63](#) is for 2-dimensional closed paths. This entity is closely associated with properties indicating functionality and fabrication parameters, such as Line Widening.

If required, the default parameterization is as defined below. It is consistent with the 0-1 parameterization of the Line Entity ([Type 110](#)) in that it results in local 0-1 parameterizations for each of the line segments of the path.

Let

$C$       be the composite curve;

$P(i)$       be the  $i$ -th point in the definition of the path;

$N$       be the number of points in the definition of the path.

Then

1. The parametric values,  $u$ , of  $C$  range from 0 to  $N - 1$ ; and

2.  $C(u) = P(i + 1) + s(P(i + 2) - P(i + 1))$

where

$$i \leq u \leq i + 1$$

$$0 \leq i \leq N - 1$$

$$s = u - i.$$

## 4.7 LINEAR PATH ENTITY (TYPE 106, FORMS 11-13)

### Directory Entry

(1) Entity Type Number 106	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 106	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 11-13	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	IP	Integer	Interpretation Flag 1 = x,y pairs, common z 2 = x,y,z coordinates 3 = x,y,z coordinates and i,j,k vectors
2	N	Integer	Number of n-tuples; N >= 2

For IP=1 (x,y pairs, common z), i.e., for Forms 11:

3	ZT	Real	Common z displacement
4	X(1)	Real	First data point abscissa
5	Y(1)	Real	First data point ordinate
:	:	:	
3+2*N	Y(N)	Real	Last data point ordinate

For IP=2 (x,y,z triples), i.e., for Form 12:

3	X(1)	Real	First data point x value
4	Y(1)	Real	First data point y value
5	Z(1)	Real	First data point z value
:	:	:	
2+3*N	Z(N)	Real	Last data point z value

For IP=3 (x,y,z,i,j,k sextuples), i.e., for Form 13:

3	X(1)	Real	First data point x value
4	Y(1)	Real	First data point y value
5	Z(1)	Real	First data point z value
6	I(1)	Real	First data point i value
7	J(1)	Real	First data point j value
8	K(1)	Real	First data point k value
:	:	:	
2+6*N	K(N)	Real	Last data point k value

Additional pointers as required (see Section 2.2.4.5.2).

## 4.8 CENTERLINE ENTITY (TYPE 106, FORMS 20-21)

### 4.8 Centerline Entity (Type 106, Forms 20-21)

The Centerline Entity takes one of two forms. The first, as illustrated in Example 1 of [Figure 20](#) appears as crosshairs and is normally used in conjunction with circles. The second type (Example 2) is a construction between 2 positions.

The Centerline entities are defined as [Form 20](#) or [21](#) of the Copious Data Entity. 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

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

See [Section 4.6](#) for more information about the Copious Data Entity ([Type 106](#)).

#### Directory Entry

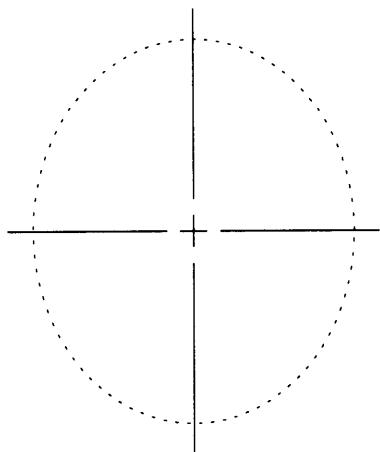
(1) Entity Type Number 106	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern 1	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ='	(9) Status Number ????01**	(10) Sequence Number D #
(11) Entity Type Number 106	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 20-21	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

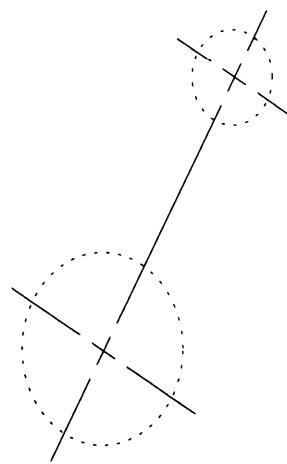
EC0650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	IP	Integer	Interpretation Flag: IP = 1
2	N	Integer	Number of data points: N is even
3	ZT	Real	Common z displacement
4	X(1)	Real	First data point abscissa
5	Y(1)	Real	First data point ordinate
:	:	:	
3+2*N	Y(N)	Real	Last data point ordinate

Additional pointers as required ([see Section 2.2.4.5.2](#)).



**EXAMPLE 1**



**EXAMPLE 2**

Figure 20. F10620X.IGS Examples Defined Using the Centerline Entity

**4.9 Section Entity (Type 106, Forms 31-38)**

A Section Entity is defined as a Copious Data Entity ([Type 106, Forms 31 to 38](#)). The form number describes how the data are to be interpreted. These descriptions are included for compatibility with previous versions of the Specification. The Sectioned Area Entity ([Type 230](#)) provides a more compact method for transferring this information.

The point data contains a list of points  $(X_n, Y_n)$ ,  $n = 1, 2, \dots, N$ , (The Z value is constant and N is an even integer.)

The display of the lines consists of solid line segments between the points  $(X_n, Y_n, Z)$  and  $(X_{n+1}, Y_{n+1}, Z)$  where  $n = 1, 3, 5, \dots, N-1$ .

A portion of collinear line segments which appear to be a dashed line shall consist of point pairs for each dash.

The defined line patterns are described below and illustrated in [Figure 21](#).

<b>Form</b>	<b>Description (see [ANSI79])</b>
31	Parallel line segments from section edge to edge (Cast or malleable iron and general use for all materials)
32	Parallel line segments in pairs with a gap between pairs (Steel)
33	Alternating pattern of a solid line and a set of collinear dash segments (Bronze, brass, copper, and compositions)
34	Parallel lines in quadruples with a gap between groups (Rubber, plastic, and electrical insulation)
35	Triples of parallel lines consisting of two solid lines and a set of collinear dash segments between them with a gap between triples (Titanium and refractory material)
36	Parallel sets of collinear dash segments (Marble, slate, glass, porcelain)
37	Two perpendicular sets of parallel lines (White metal, zinc, lead, babbitt, and alloys)
38	Two perpendicular sets of lines with the principal set solid from edge to edge and the second set consisting of collinear dash segments alternating on the solid lines (Magnesium, aluminum, and aluminum alloys)

See [Section 4.6](#) for more information about the Copious Data Entity.

## 4.9 SECTION ENTITY (TYPE 106, FORMS 31-38)

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure <i>&lt;n.a.&gt;</i>	(4) Line Font Pattern	(5) Level <i>#,⇒</i>	(6) View <i>0,⇒</i>	(7) Xformation Matrix <i>0,⇒</i>	(8) Label Display <i>0,⇒</i>	(9) Status Number <i>????01**</i>	(10) Sequence Number <i>D #</i>
(11) Entity Type Number	(12) Line Weight	(13) Color Number <i>#,⇒</i>	(14) Par ameter Line Count <i>#</i>	(15) Form Number <i>31-38</i>	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript <i>#</i>	(20) Sequence Number <i>D # + 1</i>

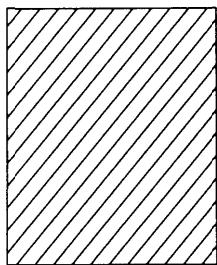
### Parameter Data

ECO650

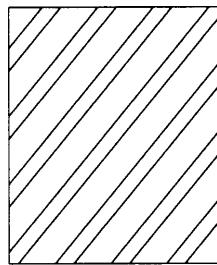
<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	IP	Integer	Interpretation Flag: IP = 1
2	N	Integer	Number of data points: N is even
3	ZT	Real	Common z displacement
4	X(1)	Real	First data point abscissa
5	Y(1)	Real	First data point ordinate
:	:	:	
3+2*N	Y(N)	Real	Last data point ordinate

Additional pointers as required ([see Section 2.2.4.5.2](#)).

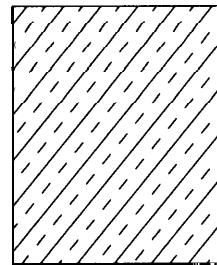
**4.9 SECTION ENTITY (TYPE, FORMS 31-38)**



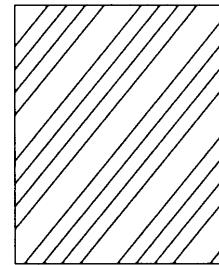
FORM 31



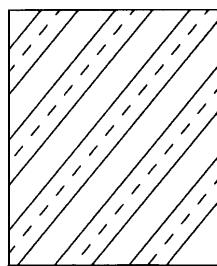
FORM 32



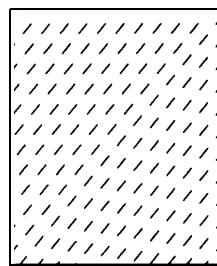
FORM 33



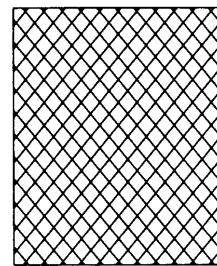
FORM 34



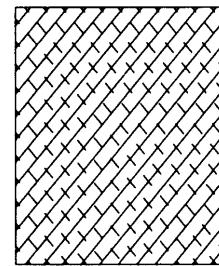
FORM 35



FORM 36



FORM 37



FORM 38

Figure 21. Definition of Patterns for the Section Entity

## 4.10 WITNESS LINE ENTITY (TYPE 106, FORM 40)

### 4.10 Witness Line Entity (Type 106, Form 40)

A Witness Line Entity is a Form Number 40 of a Copious Data Entity 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 22](#) for examples.

Within the copious data, there will be the location from which the witness line gap must be maintained. This point is indicated in the figure as PI. 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 collinear, and that the number of points will be odd and at least 3 (*i.e.*, 3, 5, 7, . . . ), with alternating blank and displayed segments. The examples in [Figure 22](#) show the blanking of segments and the order of points stored in the copious data.)

See [Section 4.6](#) for more information about the Copious Data Entity (Type 106).

#### Directory Entry

(1) Entity Type Number 106	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern 1	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????01**	(10) Sequence Number D #
(11) Entity Type Number 106	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 40	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	IP	Integer	Interpretation Flag: IP = 1
2	N	Integer	Number of data points: N >= 3 and odd
3	ZT	Real	Common z displacement
4	X(1)	Real	First data point abscissa
5	Y(1)	Real	First data point ordinate
⋮	⋮	⋮	⋮
3+2*N	Y(N)	Real	Last data point ordinate

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.10 WITNESS LINE ENTITY (TYPE 106, FORM 40)**

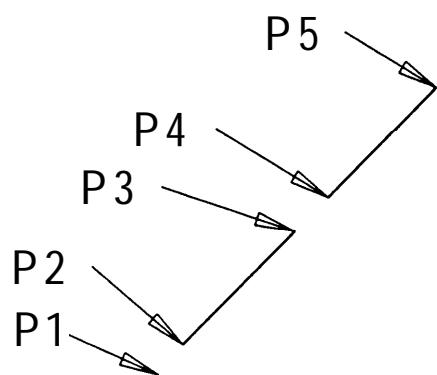
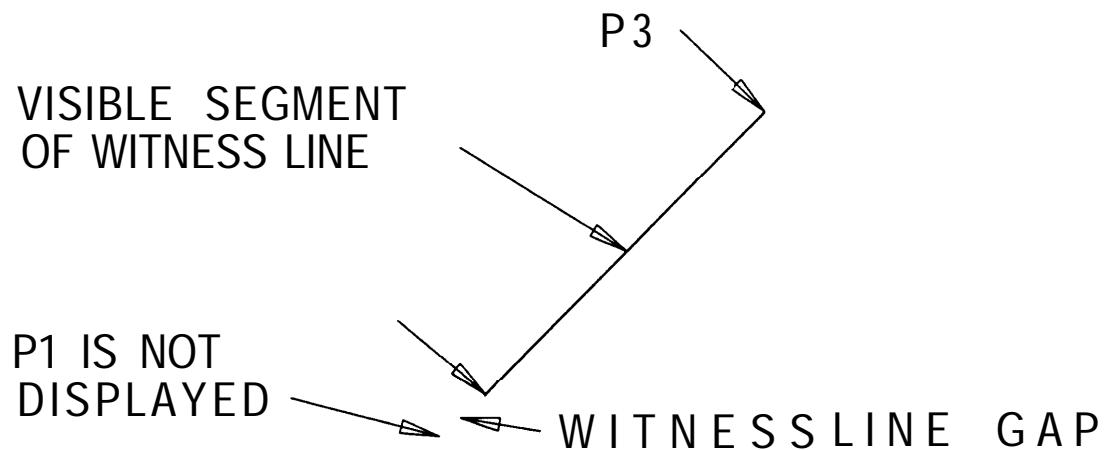


Figure 22. F10640X.IGS Examples Defined Using the Witness Line entity

## 4.11 SIMPLE CLOSED PLANAR CURVE ENTITY (TYPE 106, FORM 63)

### 4.11 Simple Closed Planar Curve Entity (Type 106, Form 63)

ECO630

A simple closed planar curve (Form 63) defines the boundary of a region in XY coordinate space. This entity must meet the constraints of a simple closed curve (see Appendix K) that lies in a plane ZT = constant. The default parameterization is the same as defined for the planar linear path (Form 11). The Simple Closed Planar Curve is closely related to entities that require the functionality of a closed region.

#### Directory Entry

(1) Entity Type Number 106	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 106	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 63	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	IP	Integer	Interpretation Flag 1 = x,y pairs, common z 2 = x,y,z coordinates 3 = x,y,z coordinates and i,j,k vectors
2	N	Integer	Number of n-tuples; N >=2
3	ZT	Real	Common z displacement
4	X(1)	Real	First data point abscissa
5	Y(1)	Real	First data point ordinate
:	:	:	
3+2*N	Y(N)	Real	Last data point ordinate

Additional pointers as required (see Section 2.2.4.5.2).

## 4.12 PLANE ENTITY (TYPE 108)

### 4.12 Plane Entity (Type 108)

The plane entity can be used to represent an unbounded plane, as well as a bounded portion of a plane. In either of the above cases, the plane is defined within definition space by means of the coefficients A, B, C, D, where at least one of A, B, and C is nonzero and ECO630

$$A \cdot X_r + B \cdot Y_r + C \cdot Z_r = D$$

for each point lying in the plane, and having definition space coordinates ( $X_r, Y_r, Z_r$ ). ECO630

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 the unbounded plane example of [Figure 23](#), the curves and the crosshair together constitute the display symbol.) Defaulting, or setting the size parameter to zero, indicates that a display symbol is not intended. ECO630

The case of a bounded portion of a fixed plane requires the existence of a pointer to a simple closed curve lying in the plane. This is parameter five. The only allowed coincident points for this curve are the start point and the terminate point. The case of an unbounded plane requires this pointer to be zero. ECO630

Versions of the Specification prior to 5.0 used the obsolete Single Parent Associativity ([Type 402](#), [Form 9](#)) to represent a bounded plane surface with holes (see [Appendix F](#)). This functionality shall now be implemented using the Bounded Surface Entity ([Type 143](#)) or the Trimmed (Parametric) Surface Entity ([Type 144](#)). ECO630

For the Plane Entity, the Form Numbers are as follows: ECO630

Form	Meaning
1	Bounded planar portion is considered positive. PTR shall not be zero.
0	Plane is unbounded. PTR shall be zero.
-1	Bounded planar portion is considered negative (hole). PTR shall not be zero.

**Directory Entry**

(1) Entity Type Number 108	(2) Parameter Data ⇒	(3) Structure <n.a.>	(4) Line Font Pattern < n.a. >	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 108	(12) Line Weight <n.a.>	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number #	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** When used as a view clipping plane, Entity Use Flag shall be Annotation (01).

ECO630

**Unbounded Plane Entity (Type 108, Form 0)**

ECO630

**Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	A	Real	Coefficients of Plane
2	B	Real	Coefficients of Plane
3	C	Real	Coefficients of Plane
4	D	Real	Coefficients of Plane
5	PTR	Pointer	Zero
6	X	Real	XT coordinate of location point for display symbol
7	Y	Real	YT coordinate of location point for display symbol
8	Z	Real	ZT coordinate of location point for display symbol
9	SIZE	Real	Size parameter for display symbol

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**Bounded Plane Entity (Type 108, Forms 1 and -1)**

**Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	A	Real	Coefficients of Plane
2	B	Real	Coefficients of Plane
3	C	Real	Coefficients of Plane
4	D	Real	Coefficients of Plane
5	PTR	Pointer	Pointer to the DE of the closed curve entity
6	X	Real	XT coordinate of location point for display symbol
7	Y	Real	YT coordinate of location point for display symbol
8	Z	Real	ZT coordinate of location point for display symbol
9	SIZE	Real	Size parameter for display symbol

Additional pointers as required ([see Section 2.2.4.5.2](#)).

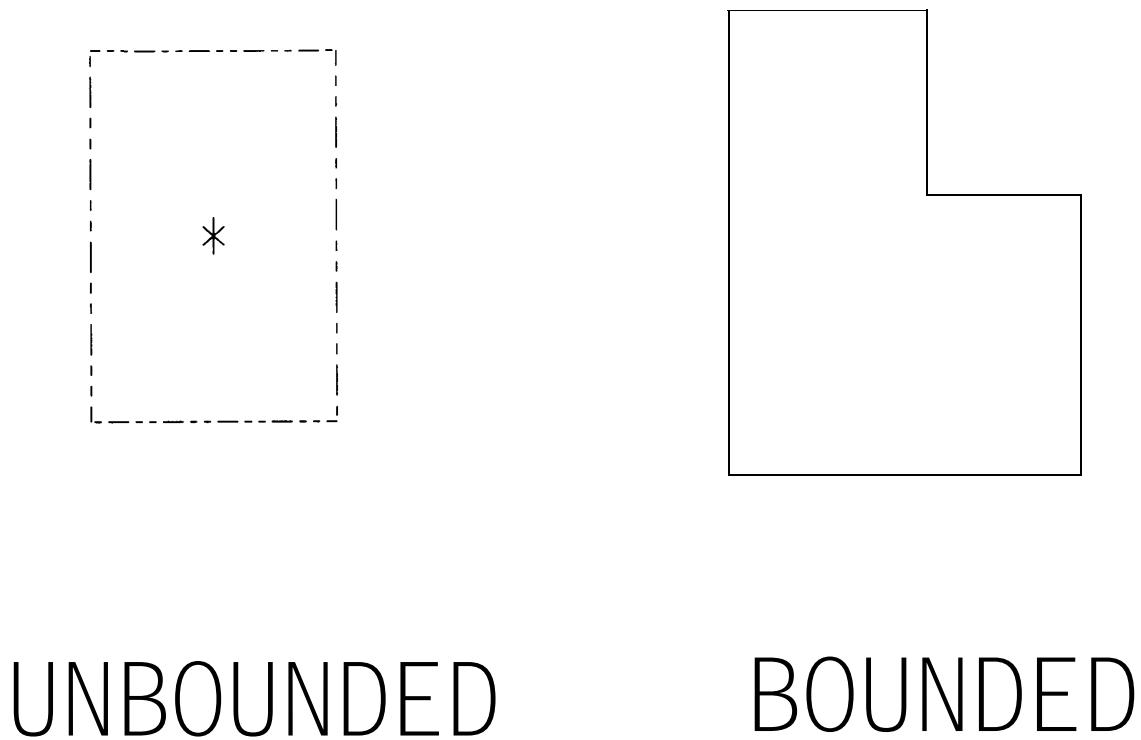


Figure 23. Examples Defined Using the Plane Entity

## 4.13 Line Entity (Type 110, Form 0)

ECO646

A line is a bounded, connected portion of a straight line which has distinct start and terminate ECO630 points.

A line is defined by its end points. Each end point is specified relative to definition space by triple 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 24](#).

If required, the default parameterization is:

ECO630

$$C(t) = P_1 + t(P_2 - P_1) \text{ for } 0 \leq t \leq 1$$

**Directory Entry**

(1) Entity Type Number 110	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 110	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	X1	Real	Start Point $P_1$
2	Y1	Real	
3	Z1	Real	
4	X2	Real	Terminate Point $P_2$
5	Y2	Real	
6	Z2	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.13 LINE ENTITY (TYPE 110, FORM 0)**

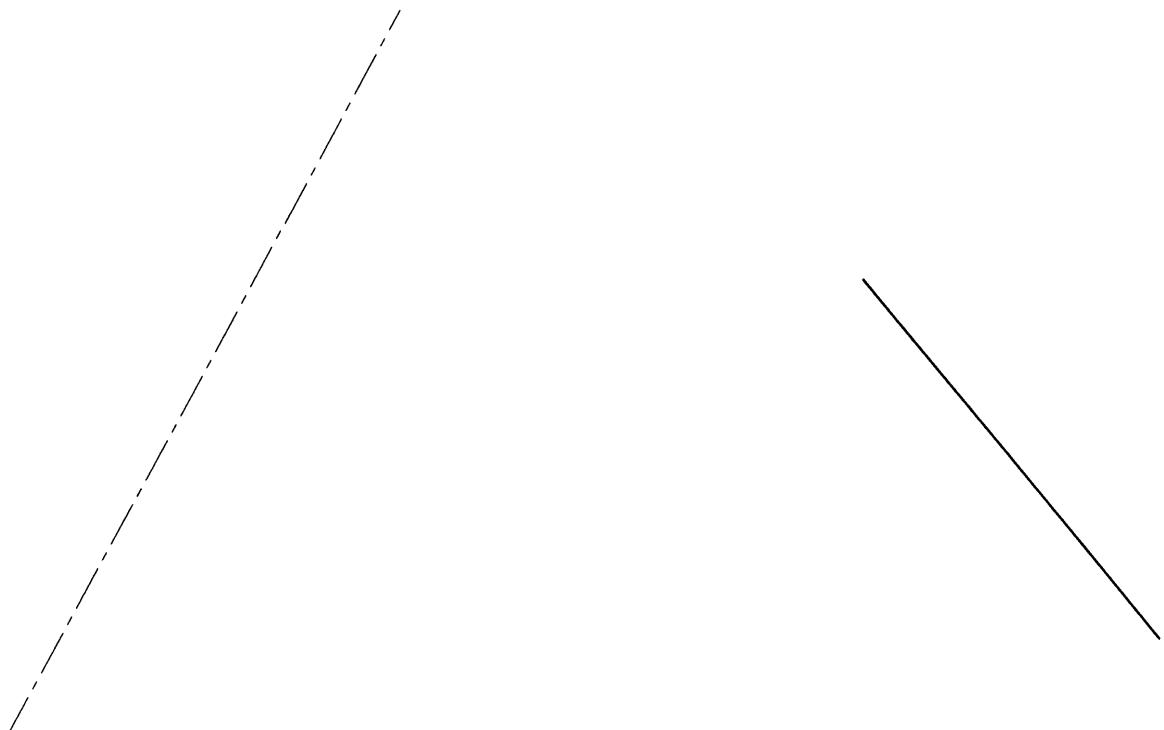


Figure 24. F110X.IGS Examples Defined Using the Line Entity

**Line Entity (Type 110, Forms 1-2)‡**

ECO646

‡These forms of the Line Entity have not been tested. See Section 1.9.

**Form 1:** A semi-bounded line is a line bounded on one end and unbounded on the other end. It is defined by a start point (P1) and an arbitrary point (P2) through which the line passes and continues without bound.

**Form 2:** An unbounded line is an infinite line. It is defined by two points (P1 and P2) through which the line passes and continues without bound in both directions.

The arbitrary points shall be chosen to be within the extent of their definition space (*i.e.*, drawing or model space). Points P1 and P2 shall be used (*i.e.*, not infinity) when determining Approximate Maximum Coordinate Value (Global field 20).

Form	Description	Default parameterization
1	Semi-Bounded Line	$C(t) = P_1 + t(P_2 - P_1)$ for $0 \leq t < \infty$
2	Unbounded Line	$C(t) = P_1 + t(P_2 - P_1)$ for $-\infty < t < \infty$

**Requirements:** Forms 1 and 2 shall specify 06 for the Entity Use Flag since semi-bounded and unbounded lines are construction geometry. Line font patterns start at P1 and continue toward P2. For form 2, the line font pattern shall repeat in the opposite direction from P1 by placing the end of the pattern at P1, with no perceptible break.

## 4.13 LINE ENTITY (TYPE 110, FORMS 1-2)‡

### Directory Entry

(1) Entity Type Number #	(2) Parameter Data ⇒	(3) Structure < <i>n.a.</i> >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????06**	(10) Sequence Number D #
(11) Entity Type Number #	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 1-2	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Semi-bounded Line Entity (Type 110, Form 1)

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	X1	Real	Start point <i>P1X</i>
2	Y1	Real	Start point <i>P1Y</i>
3	Z1	Real	Start point <i>P1Z</i>
4	X2	Real	Arbitrary point <i>P2X</i>
5	Y2	Real	Arbitrary point <i>P2Y</i>
6	Z2	Real	Arbitrary point <i>P2Z</i>

Additional pointers as required ([see Section 2.2.4.5.2](#)).

### Unbounded Line Entity (Type 110, Form 2)

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	X1	Real	Arbitrary point <i>P1X</i>
2	Y1	Real	Arbitrary point <i>P1Y</i>
3	Z1	Real	Arbitrary point <i>P1Z</i>
4	X2	Real	Arbitrary point <i>P2X</i>
5	Y2	Real	Arbitrary point <i>P2Y</i>
6	Z2	Real	Arbitrary point <i>P2Z</i>

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.14 PARAMETRIC SPLINE CURVE ENTITY (TYPE 112)

### 4.14 Parametric Spline Curve Entity (Type 112)

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

The N polynomial segments are delimited by the breakpoints  $T(1), T(2), \dots, T(N + 1)$ . The coordinates of the points in the  $i$ -th segment of the curve are given by the following cubic polynomials:

$$\begin{aligned} X(u) &= A_X(i) + s \cdot B_X(i) + s^2 \cdot C_X(i) + s^3 \cdot D_X(i) \\ Y(u) &= A_Y(i) + s \cdot B_Y(i) + s^2 \cdot C_Y(i) + s^3 \cdot D_Y(i) \\ Z(u) &= A_Z(i) + s \cdot B_Z(i) + s^2 \cdot C_Z(i) + s^3 \cdot D_Z(i) \end{aligned}$$

where

$$\begin{aligned} T(i) &\leq u \leq T(i + 1), \quad i = 1, \dots, N \\ s &= u - T(i). \end{aligned}$$

(If the degree of a polynomial is 2 or 1, the coefficients  $D$ , or  $C$  and  $D$  shall be zero, respectively.)

In order to avoid degeneracy, for each  $i$  at least one of the following nine real coefficients shall be nonzero:  $B_X(i)$ ,  $C_X(i)$ ,  $D_X(i)$ ,  $B_Y(i)$ ,  $C_Y(i)$ ,  $D_Y(i)$ ,  $B_Z(i)$ ,  $C_Z(i)$ , and  $D_Z(i)$ .

If the spline is planar, it shall be parameterized in terms of the X and Y polynomials only. The coefficients of the Z polynomial shall be zero except, for each  $i$ , the  $A_Z(i)$  term which indicates the Z-depth in definition space.

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

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

Examples of a parametric spline are shown in [Figure 25](#) and [Figure 26](#); see [Appendix B](#) for additional mathematical details.

## 4.14 PARAMETRIC SPLINE CURVE ENTITY (TYPE 112)

### Directory Entry

(1) Entity Type Number 112	(2) Parameter Data $\Rightarrow$	(3) Structure $< n.a. >$	(4) Line Font Pattern $\#, \Rightarrow$	(5) Level $\#, \Rightarrow$	(6) View $0, \Rightarrow$	(7) Xformation Matrix $0, \Rightarrow$	(8) Label Display $0, \Rightarrow$	(9) Status Number ??????**	(10) Sequence Number $D \#$
(11) Entity Type Number 112	(12) Line Weight $\#$	(13) Color Number $\#, \Rightarrow$	(14) Parameter Line Count $\#$	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript $\#$	(20) Sequence Number $D \# + 1$

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
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	Number of dimensions: 2=planar 3=nonplanar
4	N	Integer	Number of segments
5	T(1)	Real	First break point of piecewise polynomial
:	:	:	
5+N	T(N+1)	Real	Last break point of piecewise polynomial
6+N	AX(1)	Real	X coordinate polynomial
7+N	BX(1)	Real	
8+N	CX(1)	Real	
9+N	DX(1)	Real	
10+N	AY(1)	Real	Y coordinate polynomial
11+N	BY(1)	Real	
12+N	CY(1)	Real	
13+N	DY(1)	Real	
14+N	AZ(1)	Real	Z coordinate polynomial
15+N	BZ(1)	Real	
16+N	CZ(1)	Real	
17+N	DZ(1)	Real	
:	:	:	Subsequent X, Y, Z polynomials concluding with the twelve coefficients of the Nth polynomial segment.

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

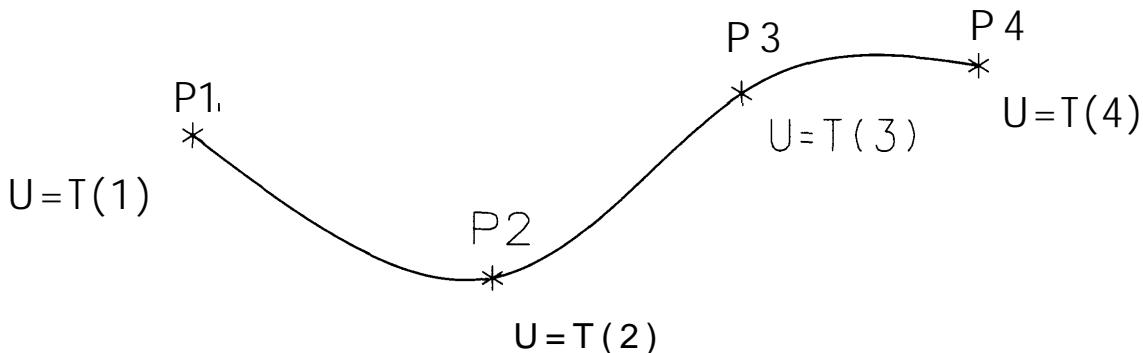
6+13\*N TPX0 Real X value  
7+13\*N TPX1 Real X first derivative

#### 4.14 PARAMETRIC SPLINE CURVE ENTITY (TYPE 112)

8+13*N	TPX2	Real	X second derivative/2!
9+13*N	TPX3	Real	X third derivative/3!
10+13*N	TPY0	Real	Y value
11+13*N	TPY1	Real	Y first derivative
12+13*N	TPY2	Real	Y second derivative/2!
13+13*N	TPY3	Real	Y third derivative/3!
14+13*N	TPZ0	Real	Z value
15+13*N	TPZ1	Real	Z first derivative
16+13*N	TPZ2	Real	Z second derivative/2!
17+13*N	TPZ3	Real	Z third derivative/3!

Additional pointers as required (see Section 2.2.4.5.2).

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



$P_1 = (AX(1), AY(1), AZ(1))$   
 $P_2 = (AX(2), AY(2), AZ(2))$   
 $P_3 = (AX(3), AY(3), AZ(3))$   
 $P_4 = TP_0 = (TPX_0, TPY_0, TPZ_0)$   
 FIRST DERIVATIVE AT  $P_4 = TP_1 = (TPX_1, TPY_1, TPZ_1)$

Figure 25. F112PX.IGS Parameters of the Parametric Spline Curve Entity

#### **4.14 PARAMETRIC SPLINE CURVE ENTITY (TYPE 112)**

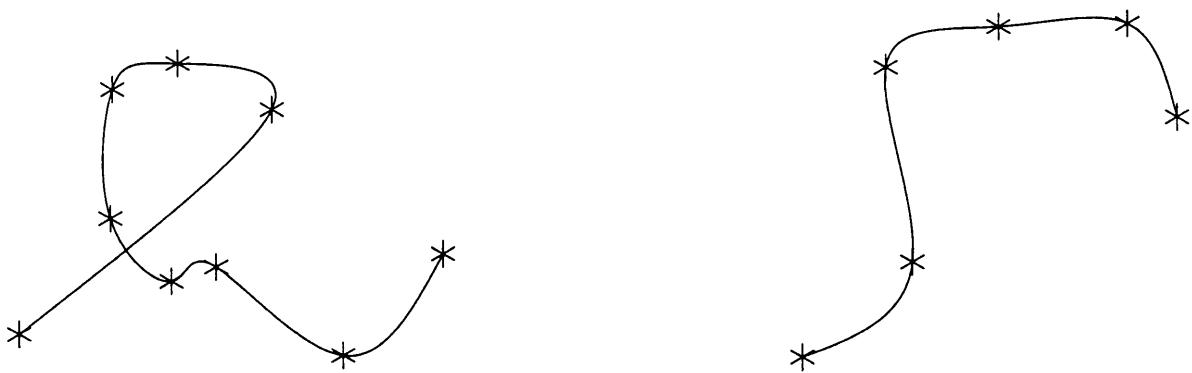


Figure 26. F112X.IGS Examples Defined Using the Parametric Spline Curve Entity

#### 4.15 Parametric Spline Surface Entity (Type 114)

The parametric spline surface is a grid of parametric polynomial patches. PTYPE in the Parameter Data Section indicates the type of patch under consideration.

The  $M \times N$  grid of patches is defined by the  $u$  breakpoints  $T_U(1), \dots, T_U(M + 1)$  and the  $v$  ECO630 breakpoints  $T_V(1), \dots, T_V(N + 1)$ . The coordinates of the points in each of the patches are given by the general bicubic polynomials (given here for the  $(i, j)$  patch).

$$\begin{aligned} X(u, v) &= A_X(i, j) + s \cdot B_X(i, j) + s^2 \cdot C_X(i, j) + s^3 \cdot D_X(i, j) \\ &\quad + t \cdot E_X(i, j) + t \cdot s \cdot F_X(i, j) + t \cdot s^2 \cdot G_X(i, j) + t \cdot s^3 \cdot H_X(i, j) \\ &\quad + t^2 \cdot K_X(i, j) + t^2 \cdot s \cdot L_X(i, j) + t^2 \cdot s^2 \cdot M_X(i, j) + t^2 \cdot s^3 \cdot N_X(i, j) \\ &\quad + t^3 \cdot P_X(i, j) + t^3 \cdot s \cdot Q_X(i, j) + t^3 \cdot s^2 \cdot R_X(i, j) + t^3 \cdot s^3 \cdot S_X(i, j) \\ Y(u, v) &= \dots \\ Z(u, v) &= \dots \end{aligned}$$

where

$$\begin{aligned} T_U(i) \leq u &\leq T_U(i + 1), & i = 1, \dots, M \\ s &= u - T_U(i) \end{aligned}$$

and

$$\begin{aligned} T_V(j) \leq v &\leq T_V(j + 1), & j = 1, \dots, N \\ t &= v - T_V(j) \end{aligned}$$

Postprocessors shall ignore parameters with the indices

$$7 + M + N + 48 \cdot (k \cdot N + (k - 1))$$

through

$$6 + M + N + 48 \cdot (k \cdot (N + 1)),$$

where

$$k = 1, 2, 3, \dots, M$$

(i.e., the  $(N + 1)$ -th row of patches) as well as

$$7 + M + N + 48 \cdot (M \cdot (N + 1))$$

through

$$6 + M + N + 48 \cdot (M + 1) \cdot (N + 1)$$

(i.e., the  $(M + 1)$ -th column of patches).

To maintain upward compatibility with previous versions of this Specification, the preprocessors ECO630 shall either enter a real number for each of these parameters or a series of parameter delimiters (see Section 2.2.3). These values act as placeholders in the parameter list. These parameters were intended to handle first, second, and third partial derivatives of the  $N$ -th row and  $M$ -th column of patches along the outer edge or boundary. However, these parameters can be computed by the receiving system, as needed, from the other parameter values contained in this entity, and therefore are not needed.

An example of the bicubic surface is shown in Figure 27; consult Appendix B for additional details.

## 4.15 PARAMETRIC SPLINE SURFACE ENTITY (TYPE 114)

### Directory Entry

(1) Entity Type Number 114	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ??????**	(11) Sequence Number D #
(11) Entity Type Number 114	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	CTYPE	Integer	Spline Boundary 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)	Real	First breakpoint in u (u values of grid lines)
⋮	⋮	⋮	⋮
5+M	TU(M+1)	Real	Last breakpoint in u
6+M	TV(1)	Real	First breakpoint in v (v values of grid lines)
⋮	⋮	⋮	⋮
6+M+N	TV(M+1)	Real	Last breakpoint in v
7+M+N	AX(1,1)	Real	First X coefficient of (1,1) Patch
⋮	⋮	⋮	⋮
22+M+N	SX(1,1)	Real	Last X Coefficient of (1,1) Patch
23+M+N	AY(1,1)	Real	First Y coefficient of (1,1) Patch
⋮	⋮	⋮	⋮
38+M+N	SY(1,1)	Real	Last Y Coefficient of (1,1) Patch
39+M+N	AZ(1,1)	Real	First Z coefficient of (1,1) Patch
⋮	⋮	⋮	⋮
54+M+N	SZ(1,1)	Real	Last Z Coefficient of (1,1) Patch
55+M+N	AX(1,2)	Real	First X Coefficient of (1,2) Patch
⋮	⋮	⋮	⋮
102+M+N	SZ(1,2)	Real	Last Z Coefficient of (1,2) Patch
⋮	⋮	⋮	⋮
7+M+N+48*(N-1)	AX(1,N)	Real	First X Coefficient of (1,N) Patch

## 4.15 PARAMETRIC SPLINE SURFACE ENTITY (TYPE 114)

$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$6+M+N+48^*N$	$SZ(1,N)$	Real	Last Z Coefficient of (1,N) Patch	
$7+M+N+48^*N$	$< n.a. >$	Real	Beginning of Arbitrary Values	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$6+M+N+48^*(N+1)$	$< n.a. >$	Real	End of Arbitrary Values	
$7+M+N+48^*(N+1)$	$AX(2,1)$	Real	First X Coefficient of (2,1) Patch	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$6+M+N+48^*(N+2)$	$SZ(2,1)$	Real	Last Z Coefficient of (2,1) Patch	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$7+M+N+48^*(2^*N)$	$AX(2,N)$	Real	First X Coefficient of (2,N) Patch	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$6+M+N+43^*(2^*N+1)$	$SZ(2,N)$	Real	Last Z Coefficient of (2,N) Patch	
$7+M+N+48^*(2^*N+1)$	$< n.a. >$	Real	Beginning of Arbitrary Values	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$6+M+N+48^*(2^*N+2)$	$< n.a. >$	Real	Arbitrary Value	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$7+M+N+48^*[(J-1)^*(N+1)+K-1]$	$AX(J,K)$	Real	First X Coefficient of (J,K) Patch	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$6+M+N+48^*[(J-1)^*(N+1)+K]$	$SZ(J,K)$	Real	Last Z Coefficient of (J,K) Patch	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$7+M+N+48^*[(M-1)^*(N+1)+N-1]$	$AX(M,N)$	Real	First X Coefficient of (M,N) Patch	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$6+M+N+48^*[(M-1)^*(N+1)+N]$	$SZ(M,N)$	Real	Last Z Coefficient of (M,N) Patch	
$7+M+N+48^*[(M-1)^*(N+1)+N]$	$< n.a. >$	Real	Beginning of Arbitrary Values	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$6+M+N+48^*[(M-l)^*(N+1)+(N+1)]$	$< n.a. >$	Real	Arbitrary Value	ECO630
$7+M+N+48^*[M^*(N+1)]$	$< n.a. >$	Real	Arbitrary Value	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$6+M+N+48^*[M^*(N+1)+(N+1)]$	$< n.a. >$	Real	End of Arbitrary Values	

Additional pointers as required (see Section 2.2.4.5.2).

#### 4.15 PARAMETRIC SPLINE SURFACE ENTITY (TYPE 114)

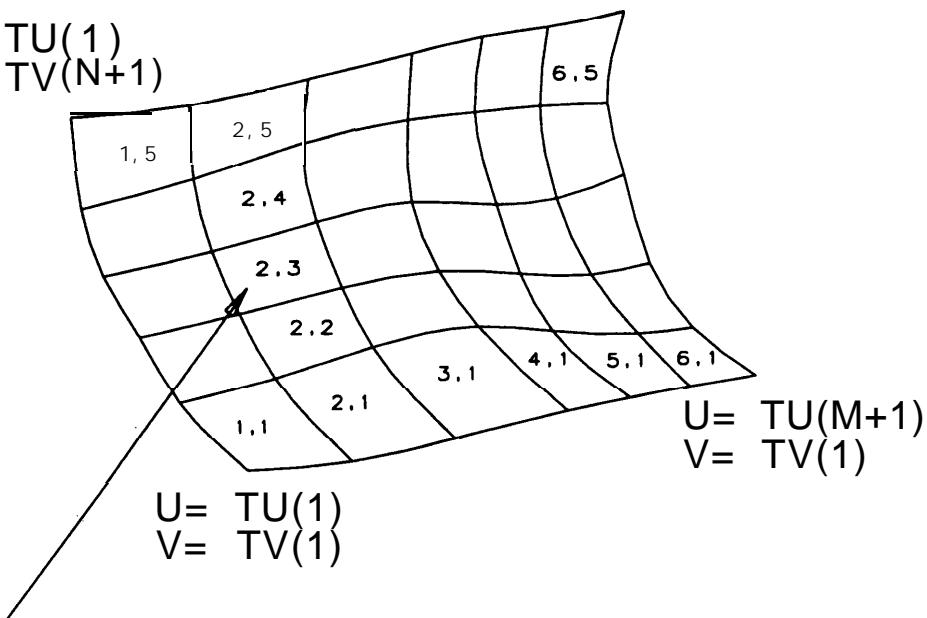
SURFACE =  $(X(U,V), Y(U,V), Z(U,V))$

M = 6

N = 5

$$U = \frac{TU(M+1)}{TV(N+1)}$$

$$U = \frac{TU(1)}{TV(N+1)}$$



$$U = \frac{TU(1)}{TV(1)}$$

$$X(U,V) = AX \quad (2,3) + \dots \quad .$$

$$Y(U,V) = AY \quad (2,3) + \dots \quad .$$

$$Z(U,V) = AZ \quad (2,3) + \dots \quad .$$

Figure 27. Parameters of the Parametric Spline Surface Entity

## 4.16 Point Entity (Type 116)

A point is defined by its coordinates in definition space. An optional pointer to a Subfigure Definition Entity ([Type 308](#)) references a display symbol. Examples of the Point Entity are shown in [Figure 28](#).

### Directory Entry

(1) Entity Type Number 116	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ?????????	(10) Sequence Number D #
(11) Entity Type Number 116	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

Note: If PD Index 4 (Pointer to Display Geometry) is 0 or defaulted, Line Font Pattern, Line Weight, and Hierarchy are ignored.

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	X	Real	Coordinates of point
2	Y	Real	
3	Z	Real	
4	PTR	Pointer	Pointer to the DE of the Subfigure Definition Entity specifying the display symbol or zero. If zero, no display symbol is specified.

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### **4.16 POINT ENTITY (TYPE 116)**



Figure 28. Examples Defined Using the Point Entity

### **4.17 Ruled Surface Entity (Type 118)**

A ruled surface is formed by moving a line connecting points of equal relative arc length (Form 0 or ECO630 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, conies, parametric splines, rational B-splines, composite curves, or any parametric curves defined in this Specification (both planar and non-planar). Examples of the Ruled Surface Entity are shown in [Figures 29 and 30](#).

If required, the default parameterization is:

ECO639

$$\begin{aligned} X(u, v) &= (1 - v) \cdot C1_x(t) + v \cdot C2_x(s) \\ Y(u, v) &= (1 - v) \cdot C1_y(t) + v \cdot C2_y(s) \\ Z(u, v) &= (1 - v) \cdot C1_z(t) + v \cdot C2_z(s), \end{aligned}$$

where the two curves are expressed parametrically by the functions ( $C1_x(t)$ ,  $C1_y(t)$ ,  $C1_z(t)$ ) and ( $C2_x(s)$ ,  $C2_y(s)$ ,  $C2_z(s)$ ),

$$\begin{aligned} a &\leq t \leq b, \\ c &\leq s \leq d, \\ 0 &\leq u \leq 1, \\ 0 &\leq v \leq 1, \\ t &= a + u \cdot (b - a), \\ s &= c + u \cdot (d - c), \quad DIRFLG = 0 \\ s &= d + u \cdot (c - d), \quad DIRFLG = 1. \end{aligned}$$

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

If  $DIRFLG=0$ , 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  $DIRFLG=1$ , the first point of curve 1 is joined to the last point of curve 2, and the last point of curve 1 to the first point of curve 2.

If  $DEVFLG=1$ , the surface is a developable surface (See [DOCA76] .); if  $DEVFLG=0$ , the surface may or may not be a developable surface.

For the Ruled Surface Entity, the Form Numbers are as follows:

ECO630

Form	Meaning
0	Equal relative arc length
1	Equal relative parametric values

**Form 0:** DE1 and DE2 specify the defining rail curves, but their given parameterizations are not the ones used to generate the ruled surface. Instead, their arc length reparameterizations,  $C1$  and  $C2$  (respectively), are used.

**Form 1:** DE1 and DE2 specify the defining rail curves,  $C1$  and  $C2$  (respectively). Moreover, their given parameterizations are the ones used to generate the ruled surface.

## 4.17 RULED SURFACE ENTITY (TYPE 118)

### Directory Entry      Parameter Data

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
118	$\Rightarrow$	< n.a. >	#, $\Rightarrow$	#, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	??????**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
118	#	#, $\Rightarrow$	#	0-1				#	D # + 1

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DE1	Pointer	Pointer to the DE of the first curve entity
2	DE2	Pointer	Pointer to the DE of the second curve entity
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

Additional pointers as required (see Section 2.2.4.5.2).

#### 4.17 RULED SURFACE ENTITY (TYPE 115)

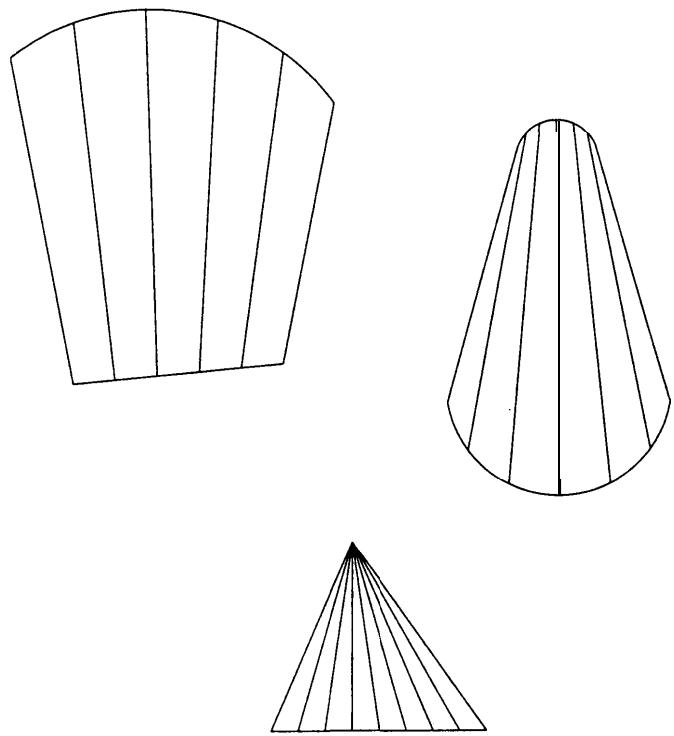


Figure 29. Examples Defined Using the Ruled Surface Entity

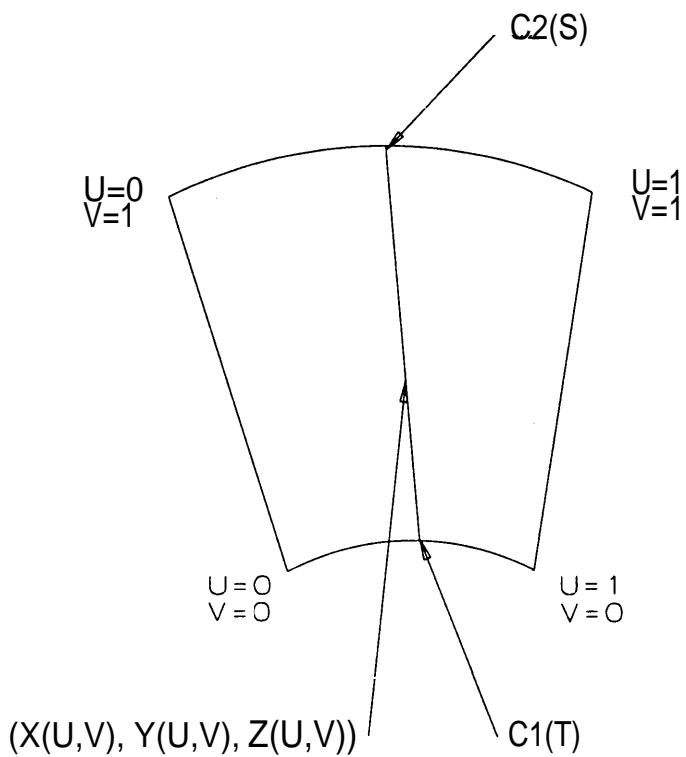


Figure 30. Parameters of the Ruled Surface Entity

## 4.18 SURFACE OF REVOLUTION ENTITY (TYPE 120)

### 4.18 Surface of Revolution Entity (Type 120)

A surface of revolution is defined by an axis of rotation (which shall 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 (Type 110), it contains in its parameter data section the coordinates of its start point first, followed by the coordinates of its terminate point. The angles of rotation are measured counterclockwise from the terminate point of the Line Entity defining the axis of revolution while looking in the direction of the start point of this line. The generatrix curve may be any curve entity to which a parameterization has been assigned. Examples of surfaces of revolution are given in [Figure 31](#). ECO630

The various parameters defining the Surface of Revolution Entity are illustrated in [Figure 32](#). The Line Entity L defines a unique straight line. This straight line defines the axis of revolution. The axis is given the same direction as the direction assigned to the Line Entity L. Let  $R_\theta$  be the unique rigid motion leaving each point of the axis of revolution fixed and rotating each point in three-dimensional Euclidean space  $\theta$  radians counterclockwise about the axis of revolution.  $R_\theta$  assigns to each element of three-dimensional Euclidean space another element of three-dimensional Euclidean space. ECO630

The curve C is the generatrix of the surface of revolution. For each real number in the parametric interval  $[a,b]$  that defines its domain, C assigns an element of three-dimensional Euclidean space. ECO630

SA and TA denote the start angle and terminate angle, measured in radians, of the surface of revolution to be defined. SA and TA are constrained so that  $0 < TA - SA \leq 2\pi$ .

The surface of revolution S defined by this entity is the surface that is swept by rotating the generatrix curve C from the angle SA to the angle TA, counterclockwise about the directed axis of revolution.

If required, the default parameterization for the surface of revolution S is given by

$$S(x, \theta) = R_\theta(C(x))$$

for each pair of real numbers  $(x, \theta)$  such that  $a \leq x \leq b$  and  $SA \leq \theta \leq TA$ .

## 4.18 SURFACE OF REVOLUTION ENTITY (TYPE 120)

### Directory Entry

(1) Entity Type Number 120	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 120	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	L	Pointer	Pointer to the DE of the Line Entity (axis of revolution)
2	C	Pointer	Pointer to the DE of the generatrix entity
3	SA	Real	Start angle in radians
4	TA	Real	Terminate angle in radians

Additional pointers as required (see Section 2.2.4.5.2).

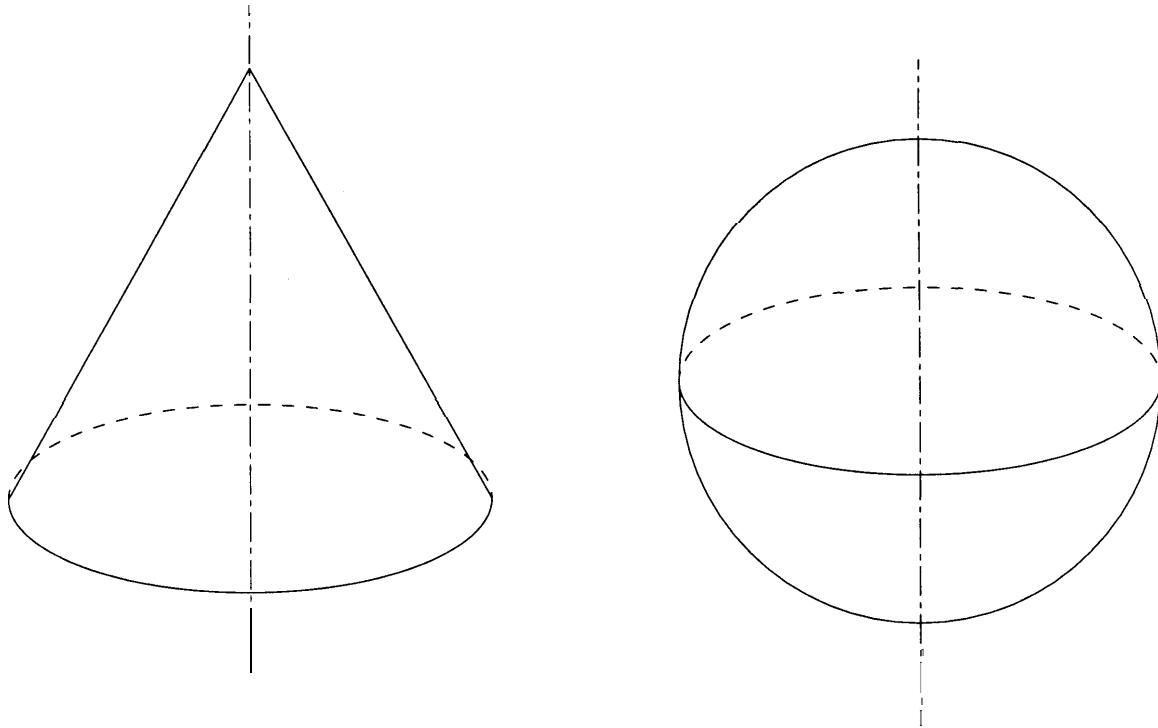


Figure 31. Examples Defined Using the Surface of Revolution Entity

#### 4.18 SURFACE OF REVOLUTION ENTITY (TYPE 120)

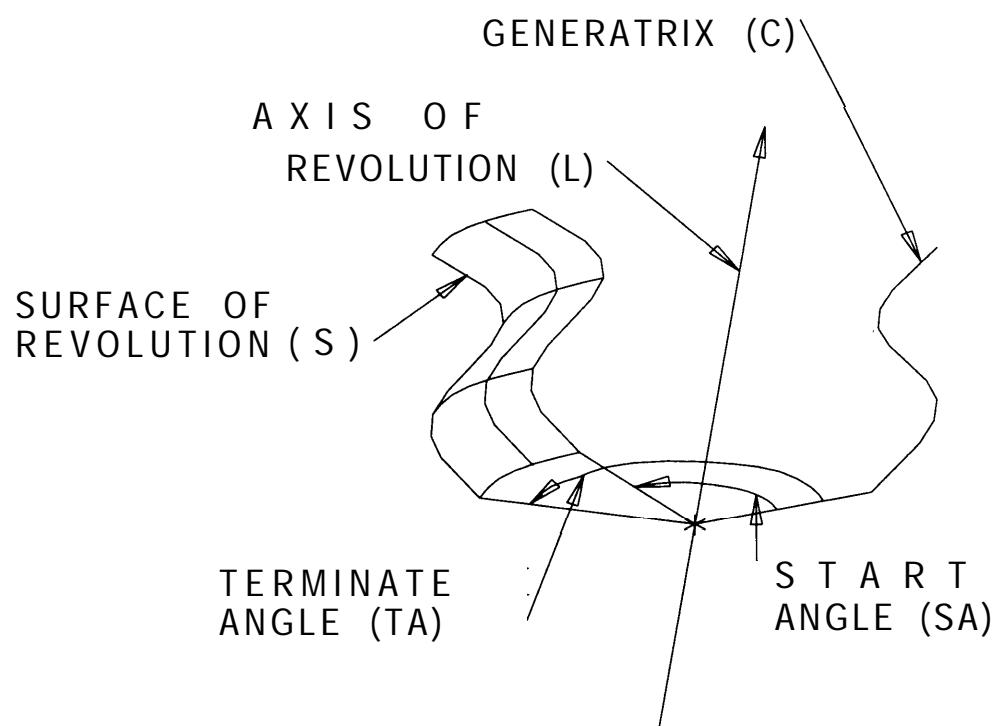


Figure 32. Parameters of the Surface of Revolution Entity

## 4.19 TABULATED CYLINDER ENTITY (TYPE 122)

### 4.19 Tabulated Cylinder Entity (Type 122)

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, rational B-spline curve, composite curve, or any parametric curve defined in this Specification (both planar and non-planar). The start point of the generatrix is identical with the start point of the directrix. An example of the tabulated cylinder is shown in [Figure 33](#). ECO630  
ECO640

Caution: different parameterizations of the generating curves will produce different parameterized surfaces, but the underlying point-set surface will still be the same. ECO630

If required, the default parameterization is: ECO640  
ECO630

$$\begin{aligned} X(u,v) &= CX(u) + v \cdot (LX - CX(0)) \\ Y(u,v) &= CY(u) + v \cdot (LY - CY(0)) \\ Z(u,v) &= CZ(u) + v \cdot (LZ - CZ(0)) \end{aligned}$$

where the curve is parameterized by  $(CX(t), CY(t), CZ(t))$ ,

$$\begin{aligned} a &\leq t \leq b, \\ 0 &\leq u \leq 1, \\ 0 &\leq v \leq 1, \\ t &= a + u \cdot (b-a), \end{aligned}$$

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

## 4.19 TABULATED CYLINDER ENTITY (TYPE 122)

### Directory Entry

(1) Entity Type Number 122	(2) Parameter Data $\Rightarrow$	(3) Structure $< n.a. >$	(4) Line Font Pattern $\#, \Rightarrow$	(5) Level $\#, \Rightarrow$	(6) View 0, $\Rightarrow$	(7) Xformation Matrix 0, $\Rightarrow$	(8) Label Display 0, $\Rightarrow$	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 122	(12) Line Weight $\#$	(13) Color Number $\#, \Rightarrow$	(14) Parameter Line Count $\#$	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript $\#$	(20) Sequence Number D # + 1

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DE	Pointer	Pointer to the DE of the directrix curve entity
2	LX	Real	Coordinates of the terminate point of the generatrix
3	LY	Real	
4	LZ	Real	

Additional pointers as required (see Section 2.2.4.5.2).

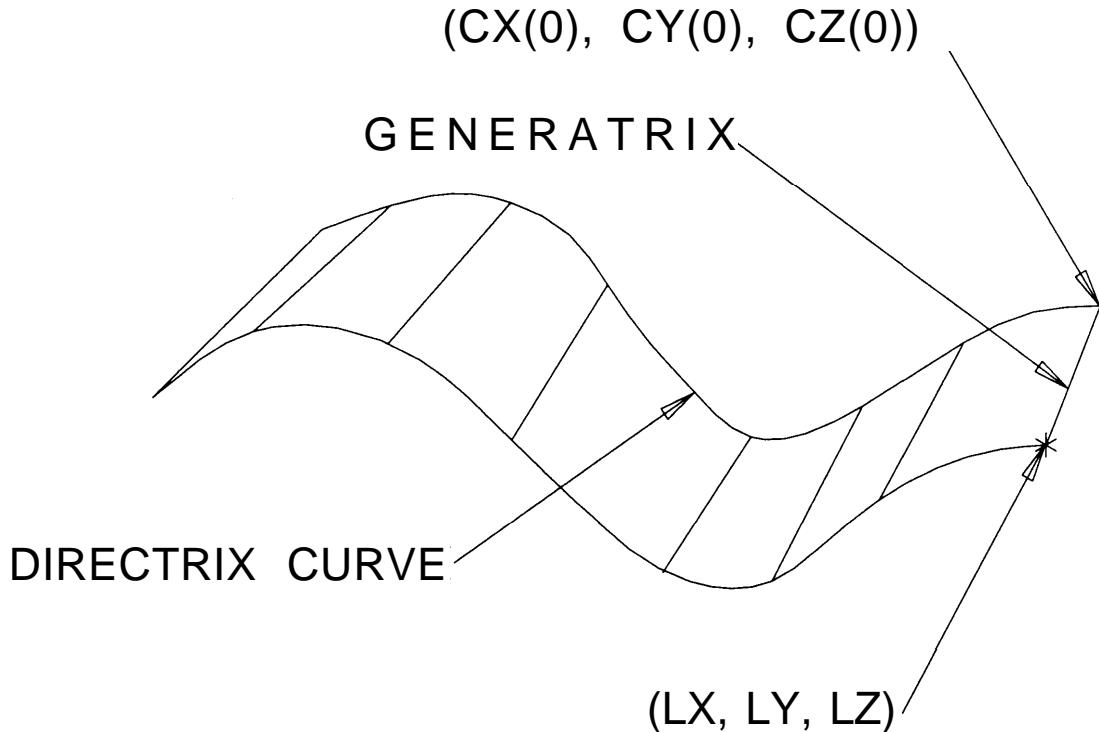


Figure 33. Parameters of the Tabulated Cylinder Entity

## 4.20 DIRECTION ENTITY (TYPE 123)‡

### 4.20 Direction Entity (Type 123)‡

‡The Direction Entity has not been tested. See Section 1.9.

A direction entity is a non-zero vector in Euclidean 3-space that is defined by its three components (direction ratios) with respect to the coordinate axes. If  $x, y, z$  are the direction ratios,

$$x^2 + y^2 + z^2 > 0.$$

The Subordinate Entity Switch shall always be set to Physically Dependent. The Transformation Matrix Entity (Type 124) shall not be referenced by this entity.

#### Directory Entry

ECO630

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) status Number	(10) Sequence Number
123	⇒	< n.a. >	< n.a. >	< n.a. >	< ma. >	< n.a. >	< n.a. >	**0102**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
123	< n.a. >	< n.a. >	#	0				#	D # + 1

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	X	Real	Direction ratio with respect to X axis
2	Y	Real	Direction ratio with respect to Y axis
3	Z	Real	Direction ratio with respect to Z axis

Additional pointers as required (see Section 2.2.4.5.2).

### 4.21 Transformation Matrix Entity (Type 124)

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

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

Here, column [XINPUT, YINPUT, ZINPUT] (*i.e.*, the column vector) is the vector being transformed, and column [XOUTPUT, YOUTPUT, ZOUTPUT] is the column vector resulting from this transformation.  $R = [R_{ij}]$  is a 3 row by 3 column matrix of real numbers, and  $T = \text{column } [T_1, T_2, T_3]$  is a three-row column vector of real numbers. Thus, 12 real numbers are required for a Transformation Matrix Entity. This entity can be considered to be an “operator” entity in that it starts with the input vector, operates on it as described above, and produces the output vector.

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

For example, for

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

the spatial relationship of the input and output coordinate systems is given in Figure 34.

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

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

The usual situation for this type of use of the Transformation Matrix Entity is when the input vector refers to the definition space coordinate system for a certain entity, and the output vector refers to the model space coordinate system ([See Section 3.2.2](#)). In this case, the matrix  $R$  is referred to as the defining matrix, and the Transformation Matrix Entity defining  $R$  and  $T$  is pointed to in field seven (transformation matrix field) of the directory entry of the entity ([See Section 2.2.4.4.7](#)). In this use of the Transformation Matrix Entity, the matrix  $R$  is subject to the restrictions given in Form 0 and Form 1 below.

A second situation is the case when the input vector refers to the model space coordinate system and the output vector refers to a viewing coordinate system. In this case, the matrix  $R$  is referred to as a view matrix, and is subject to the restrictions given in Form 0 below. Note that when a planar entity is viewed at true length (*i.e.*, The viewing plane is parallel to the plane containing the entity.), the rotation matrix pointed to by DE Field 7 of the Planar Entity will be the inverse (is equal to the matrix transpose) of the matrix pointed to by DE Field 7 of the View Entity ([See Section 4.134](#)).

A third situation involves finite element modeling applications. Here, it may be the case that an input coordinate system is related to an output coordinate system by a particular  $R$  and  $T$ , and, in

## 4.21 TRANSFORMATION MATRIX ENTITY (TYPE 124)

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

Whenever coordinate systems are related successively to each other as described above, a basic result ECO630 is that the combined effect of the individual coordinate system changes can be expressed in terms of a single matrix  $R$  and a single translation vector  $T$ . For example, if the coordinate system change involving the matrix  $R_2$  and the translation vector  $T_2$  is to be applied following the coordinate system change involving the matrix  $R_1$  and the translation vector  $T_1$ , then the matrix  $R$  and the translation vector  $T$  expressing the combined changes are  $R = R_2 \times R_1$  and  $T = R_2 \times T_1 + T_2$ .

Here,  $R_2 \times R_1$  denotes matrix multiplication of  $3 \times 3$  matrices, where multiplication order is important. The matrix  $R$  and the translation vector  $T$  are computed similarly whenever more than two coordinate system changes are to be applied successively.

Successive coordinate system changes are specified by allowing a Transformation Matrix Entity to reference another Transformation Matrix Entity through Field 7 of the directory entry. In the example above, the Transformation Matrix Entity containing  $R_1$  and  $T_1$  would contain in its directory entry field 7 a pointer to the Transformation Matrix Entity containing  $R_2$  and  $T_2$ . The general rule is that Transformation Matrix Entities applied earlier in a succession will reference Transformation Matrix Entities applied later. Note that the matrix product  $R_2 \times R_1$  in the example above does not appear explicitly in the data, but, if needed, can be computed according to the usual rules of matrix multiplication.

A second example of coordinate systems being related successively (concatenated or stacked), in addition to the finite element example mentioned above, involves one manner of locating into model space a conic arc that is in standard position in definition space. In this case,  $R_1$  and  $T_1$  move the conic arc from its standard position to an arbitrary location in any plane in definition space satisfying  $ZT=\text{constant}$ . (Therefore,  $R_{133} = 1.0$ ,  $R_{131} = R_{132} = R_{113} = R_{123} = 0.0$ .  $T_1$  can be an arbitrary translation vector.)  $R_2$  and  $T_2$  then position the relocated conic arc into model space. ( $R_2$  can be an arbitrary defining matrix and  $T_2$  can be an arbitrary translation vector.) Note that for  $R_1$  and  $T_1$ , both the input vector and the output vector refer to the same coordinate system, namely, the definition space for the conic arc.

A  $3 \times 3$  matrix  $R$  is called orthonormal provided its transpose,  $R'$ , yields a matrix inverse for  $R$  and its columns, considered as vectors, form an orthonormal collection of unit vectors. As  $(R')^t = R$ , the transpose of an orthonormal matrix is again an orthonormal matrix. The determinant of an orthonormal matrix is equal to either plus one or minus one. In the event  $R$  is an orthonormal matrix with determinant equal to positive one,  $R$  can be expressed as a rotation about an axis passing through the origin. In this event,  $R$  is referred to as a rotation matrix. In the event  $R$  is an orthonormal matrix with determinant equal to negative one,  $R$  can be expressed as a rotation about an axis passing through the origin followed by a reflection about a plane passing through the origin perpendicular to the axis of rotation.

## 4.21 TRANSFORMATION MATRIX ENTITY (TYPE 124)

For the Transformation Matrix Entity, the Form Numbers are:

ECO630

<b>Form</b>	<b>Use</b>
0 or 1	Defining matrix of an entity
10, 11, or 12	Special matrices representing Node Entity ( <a href="#">Type 134</a> )

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

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

A defining matrix associated with a View Entity ([Type 410](#)) shall not use Form 1.

**Form 10:** This form number conveys special information when used in conjunction with the Node Entity ([Type 134](#)) in Finite Element Applications.

Refer to [Figure 35\(a\)](#) for notation. The matrix  $R$  and the vector  $T$  are used to transform coordinate data from the  $(u_1, u_2, u_3)$  coordinate system to the  $(x, y, z)$  local system.

The  $(u_1, u_2, u_3)$  coordinate system has its origin at an arbitrary fixed point

$$\begin{bmatrix} XOFFSET \\ YOFFSET \\ ZOFFSET \end{bmatrix}$$

in the  $(x, y, z)$  coordinate system and is assumed to be displaced parallel to that reference coordinate system. Thus,

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

so that

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} XOFFSET \\ YOFFSET \\ ZOFFSET \end{bmatrix} = \begin{bmatrix} XLOCAL \\ YLOCAL \\ ZLOCAL \end{bmatrix}$$

Note that the orientation of the two coordinate systems can be described by saying that the  $(u_1, u_2, u_3)$  coordinate system is the system obtained by imposing orthonormal curvilinear coordinates onto the  $(x, y, z)$  space and then constructing unit tangent vectors to the three curvilinear coordinate curves at the given fixed point to serve as basis vectors. In this special case of parallel displacement, the curvilinear coordinates imposed are identical to the existing  $(x, y, z)$  coordinates.

**Form 11:** This form number conveys special information when used in conjunction with the Node Entity ([Type 134](#)) in Finite Element applications.

Refer to [Figure 35\(b\)](#) for notation. The matrix  $R$  and the vector  $T$  are used to transform coordinate data from the  $(u_1, u_2, u_3)$  (node point) coordinate system to the  $(x, y, z)$  (local system) coordinate system.

The  $(u_1, u_2, u_3)$  coordinate system has its origin at an arbitrary fixed point

$$\begin{aligned} XOFFSET &= r_0 \cos \theta_0 & r_0 > 0 \\ YOFFSET &= r_0 \sin \theta_0 & 0 \leq \theta_0 \leq 360^\circ \\ ZOFFSET &= z_0 & -\infty < z_0 < \infty \end{aligned}$$

## 4.21 TRANSFORMATION MATRIX ENTITY (TYPE 124)

in the  $(x, y, z)$  coordinate system. (for  $r_o=0$ , take  $\theta = 0^\circ$ ). The  $(u_1, u_2, u_3)$  system is the system obtained by imposing orthonormal curvilinear coordinates onto the  $(x, y, z)$  space which are the cylindrical coordinates  $(r, \theta, z)$  with

$$x = r \cos \theta$$

$$y = r \sin \theta$$

$$z = z,$$

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

Thus, the relationship between the  $(u_1, u_2, u_3)$  and the  $(x, y, z)$  local coordinate system is given by:

$$\begin{bmatrix} \cos \theta_0 & -\sin \theta_0 & 0 \\ \sin \theta_0 & \cos \theta_0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} XOFFSET \\ YOFFSET \\ ZOFFSET \end{bmatrix} = \begin{bmatrix} XLOCAL \\ YLOCAL \\ ZLOCAL \end{bmatrix}$$

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

Refer to Figure 35(c) for notation. The matrix  $R$  and the vector  $T$  are used to transform coordinate data from the  $(u_1, u_2, u_3)$  coordinate system to the  $(x, y, z)$  local system.

The  $(u_1, u_2, u_3)$  coordinate system has its origin at an arbitrary fixed point

$$\begin{aligned} XOFFSET &= r_o \sin \theta_o \sin \phi_o & r_o &\geq 0 \\ YOFFSET &= r_o \sin \theta_o \cos \phi_o & 0 &\leq \theta_o \leq 180^\circ \\ ZOFFSET &= r_o \cos \theta_o & 0 &\leq \phi_o \leq 360^\circ \end{aligned}$$

in the  $(x, y, z)$  coordinate system. (For  $r_o = 0$  take  $\theta_o = \phi_o = 0^\circ$ ; for  $\theta_o = 0^\circ$  or  $180^\circ$ , take  $\phi_o = 0^\circ$ ) The  $(u_1, u_2, u_3)$  system is the system obtained by imposing orthonormal curvilinear coordinates onto the  $(x, y, z)$  space which are the spherical coordinates  $(r, \theta, \phi)$  with

$$x = r \sin \theta \cos \phi$$

$$y = r \sin \theta \sin \phi$$

$$z = r \cos \theta$$

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

Thus, the relationship between the  $(u_1, u_2, u_3)$  and the  $(x, y, z)$  local coordinate systems is given by:

$$\begin{bmatrix} \sin \theta_0 * \cos \phi_0 & \cos \theta_0 * \cos \phi_0 & -\sin \phi_0 \\ \sin \theta_0 * \sin \phi_0 & \cos \theta_0 * \sin \phi_0 & \cos \phi_0 \\ \cos \theta_0 & -\sin \theta_0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} XOFFSET \\ YOFFSET \\ ZOFFSET \end{bmatrix} = \begin{bmatrix} XLOCAL \\ YLOCAL \\ ZLOCAL \end{bmatrix}$$

See Kaplan [KAPL52] or Hildebrand [HILD76] for a discussion of orthonormal curvilinear coordinate ECO630 systems.

## 4.21 TRANSFORMATION MATRIX ENTITY (TYPE 124)

### Directory Entry

### Parameter Data

(1) Entity Type Number 124	(2) Parameter Data $\Rightarrow$	(3) Structure $< n.a. >$	(4) Line Font Pattern $< n.a. >$	(5) Level $< n.a. >$	(6) View $< n.a. >$	(7) Xformation Matrix 0, $\Rightarrow$ <sub>I</sub>	(8) Label Display $< n.a. >$	(9) Status Number ****??**	(10) Sequence Number D #
(11) Entity Type Number 124	(12) Line Weight $< n.a. >$	(13) Color Number $< n.a. >$	(14) Parameter Line Count $\#$	(15) Form Number 0-1,10-12	(16) Reserved	(17) Reserved	(18) Entity Label $\#$	(19) Entity Subscript $\#$	(20) Sequence Number D # + 1

EC0630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	R 1 1	Real	Top Row
2	R 1 2	Real	.
3	R 1 3	Real	.
4	T 1	Real	.
5	R 2 1	Real	Second Row
6	R 2 2	Real	.
7	R 2 3	Real	.
8	T 2	Real	.
9	R 3 1	Real	Third Row
10	R 3 2	Real	.
11	R 3 3	Real	.
12	T 3	Real	.

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### **4.21 TRANSFORMATION MATRIX ENTITY (TYPE 124)**

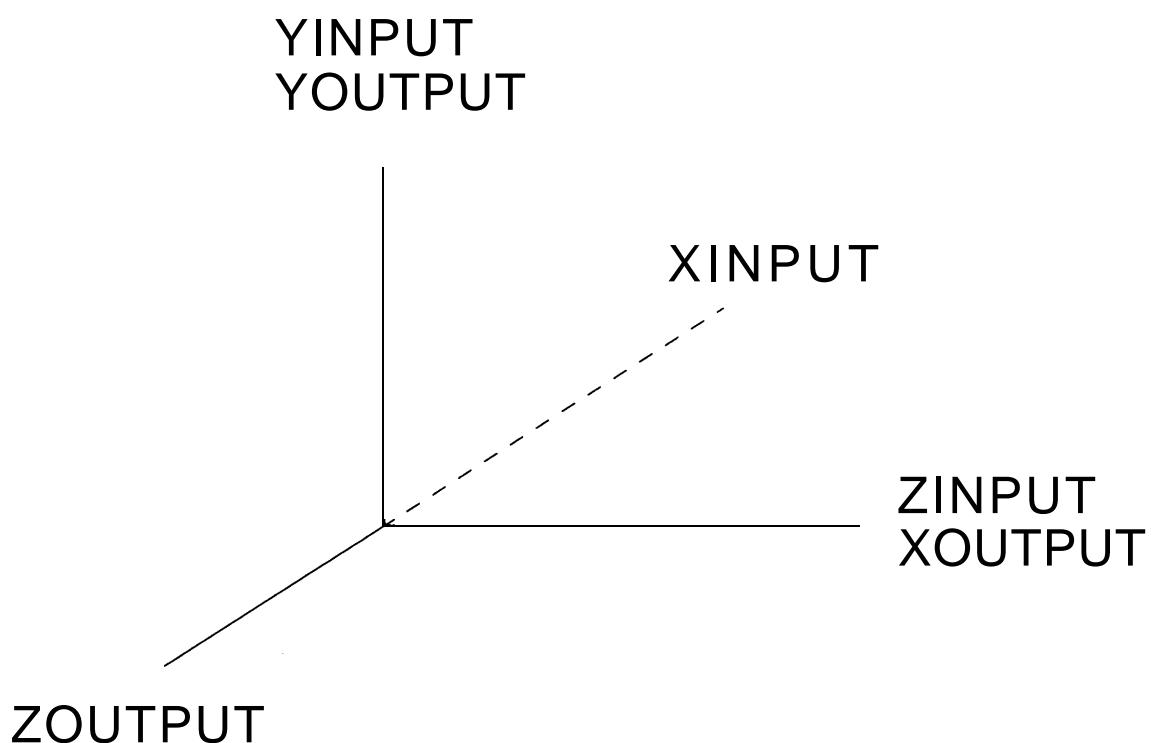


Figure 34. Example of the Transformation Matrix Coordinate Systems

#### 4.21 TRANSFORMATION MATRIX ENTITY (TYPE 124)

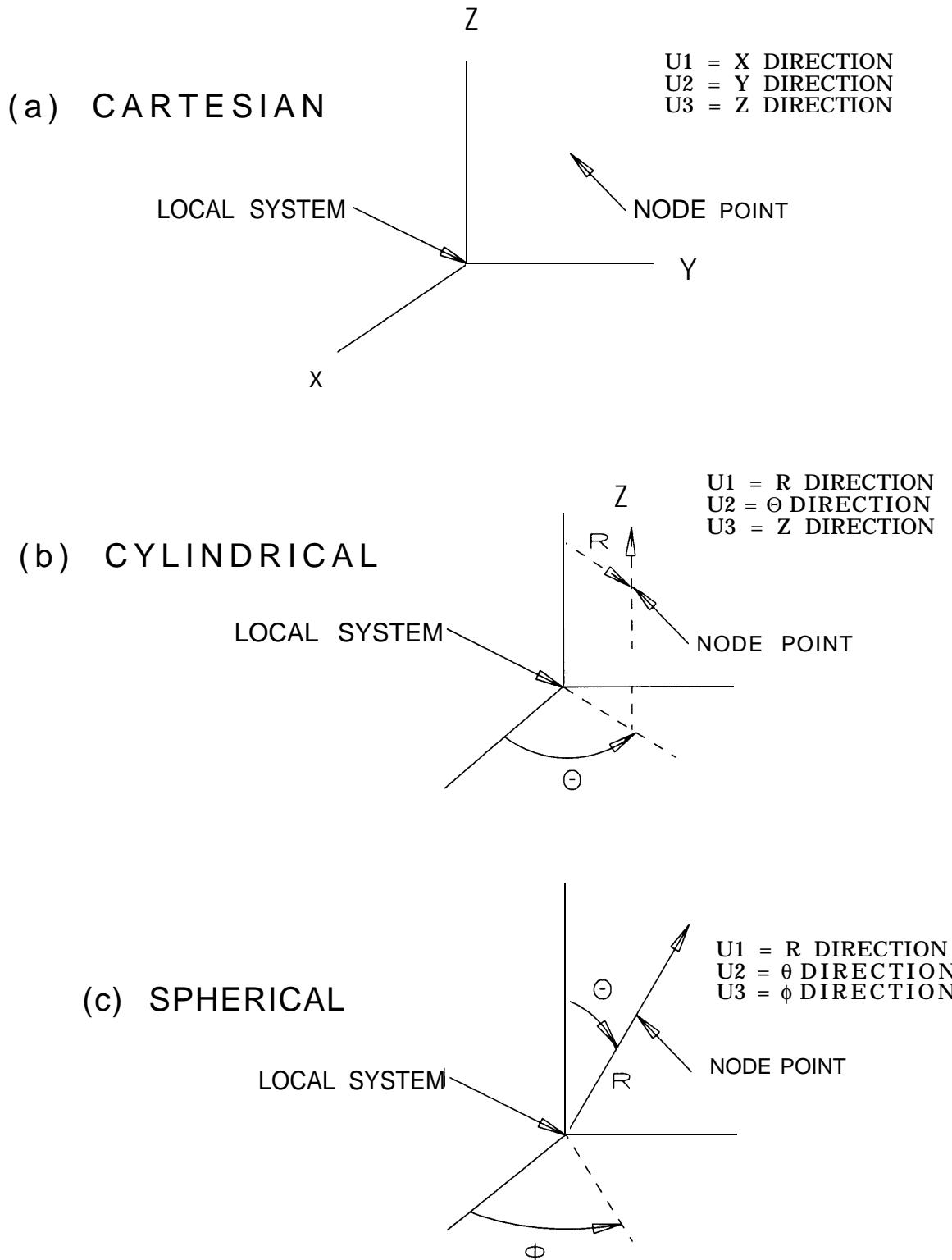


Figure 35. Notation for FEM-specific Forms of the Transformation Matrix Entity

## 4.22 Flash Entity (Type 125)

A Flash Entity is a point in the ZT=0 plane that defines the location of a specific instance of a ECO630 particular closed area. That closed area can be defined in one of two ways. In the case of Form zero, 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. For Forms one through four, the closed area can be a member of a pre-defined set of flash shapes. Refer to [Figure 36](#) for the definition of these shapes.

In the case of Forms one through four, Parameters 3 through 5 of the Flash Entity control the final ECO630 size of the flash. [Figure 36](#) indicates the definition and usage of those parameters for the specific flash forms. Parameters 3 through 5 are ignored for Form 0.

For the Flash Entity, the Form Numbers are as follows:

ECO630

Form	Meaning
0	Defined by referenced entity
1	Circular
2	Rectangle
3	Donut
4	Canoe

### Directory Entry

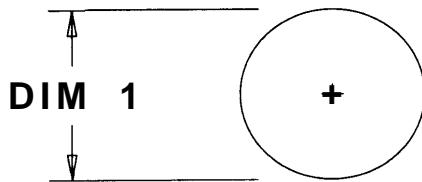
(1) Entity Type Number 125	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern 1	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ??????00	(10) Sequence Number D #
(11) Entity Type Number 125	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0-4	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

### Parameter Data

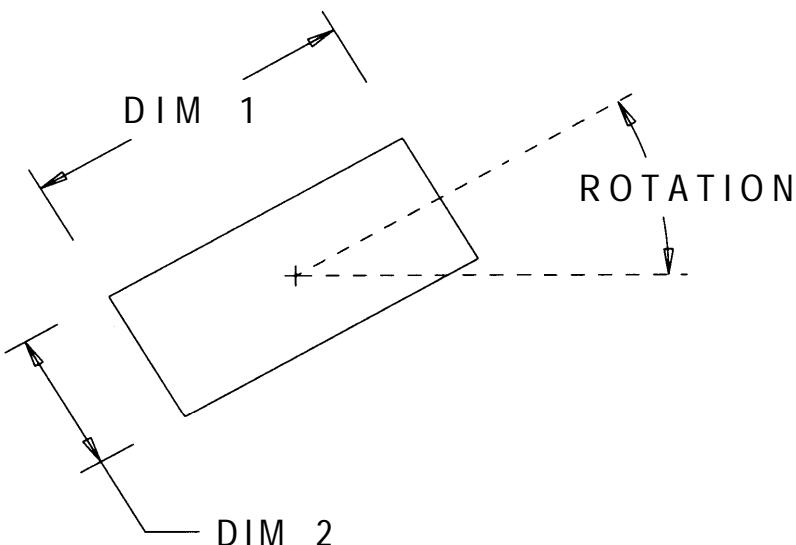
Index	Name	Type	Description
1	X	Real	X reference of flash
2	Y	Real	Y reference of flash
3	DIM1	Real	First flash sizing parameter
4	DIM2	Real	Second flash sizing parameter
5	ROT	Real	Rotation of flash about reference point in radians
6	DE	Pointer	Pointer to the DE of the referenced entity or zero

Additional pointers as required ([see Section 2.2.4.5.2](#)).



**FORM 1 - CIRCULAR**

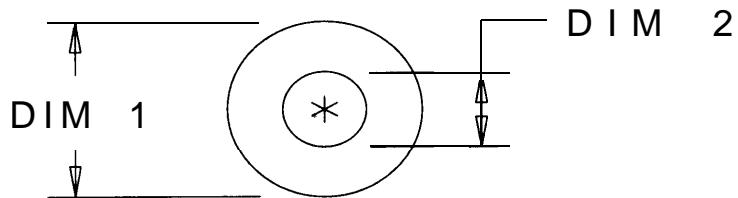
DIMENSION 1 = DIAMETER OF CIRCLE  
DIMENSION 2 = NULL OR ZERO  
ROTATION = NULL OR ZERO  
REFERENCE POINT IS CIRCLE CENTER



**FORM 2 - RECTANGLE**

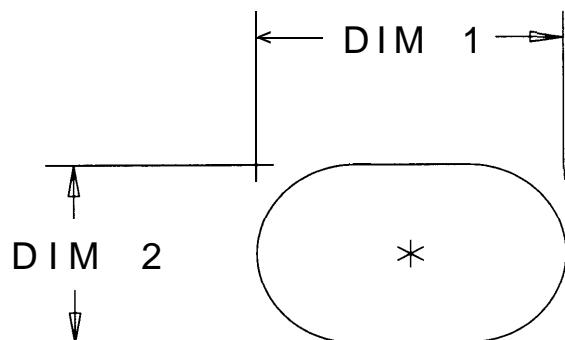
DIMENSION 1 = X AXIS LENGTH BEFORE ROTATION  
DIMENSION 2 = Y AXIS LENGTH BEFORE ROTATION  
ROTATION = ANGLE IN RADIANS COUNTERCLOCKWISE  
FROM X AXIS TO DIMENSION 1  
REFERENCE POINT IS CENTER OF RECTANGLE

Figure 36. Definition of Shapes for the Flash Entity (continues on next page)



**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 4 - CANOE**

DIMENSION 1 = OVERALL LENGTH  
DIMENSION 2 = OVERALL WIDTH  
ROTATION = ANGLE IN RADIANS CCW  
FROM X AXIS TO DIMENSION 1  
REFERENCE POINT IS CENTER OF CANOE

Figure 36. Definition of Shapes for the Flash Entity (continued)

#### 4.23 Rational B-Spline Curve Entity (Type 126)

The rational B-spline curve may represent analytic curves of general interest. This information is ECO630 important to both the sending and receiving systems. The Directory Entry Form Number Parameter is provided to communicate this information. For a brief description and a precise definition of rational B-spline curves, see [Appendix B](#). An example of the Rational B-spline Curve Entity is shown in [Figure 37](#).

If the rational B-spline curve represents a preferred curve type, the form number corresponds to the ECO630 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 shall be set to 2. If the curve is an ellipse with unequal major and minor axis lengths, the form number shall be set to 3. If the curve is not one of the preferred types, the form number shall be set to 0.

If the curve lies entirely within a unique plane, the planar flag (PROP1) shall be set to 1; otherwise it shall be set to 0 If it is set to 1, the plane normal (Parameters 14+A+4\*K through 16+A+4\*K) shall contain a unit vector normal to the plane containing the curve. These fields shall exist but are ignored if the curve is non-planar.

If the beginning and ending points on the curve, as defined by evaluating the curve at the starting ECO630 and ending parameter values (*i.e.*, V(0) and V(l)), are coincident, the curve is closed and PROP2 shall be set to 1. If they are not coincident, PROP2 shall be set to 0

If the curve is rational (does not have all weights equal), PROP3 shall be set to 0 If all weights are ECO630 equal to each other, the curve is polynomial and PROP3 shall be set to 1. The curve is polynomial since in this case all weights cancel and the denominator reduces to one. ([See Appendix B](#)) The weights shall be positive real numbers.

If the curve is periodic with respect to its parametric variable, PROP4 shall be set to 1; otherwise, ECO630 PROP4 shall be set to 0 The periodic flag is to be interpreted as purely informational; the curves which are flagged to be periodic are to be evaluated exactly the same as in the non-periodic case.

Note that the control points are in the definition space of the curve.

For the Rational B-Spline Curve Entity, the Form Numbers are as follows:

ECO630

<b>Form</b>	<b>Meaning</b>
0	Form of curve is determined from the rational B-spline parameters
1	Line
2	Circular arc
3	Elliptical arc
4	Parabolic arc
5	Hyperbolic arc

## 4.23 RATIONAL B-SPLINE CURVE ENTITY (TYPE 126)

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
126	$\Rightarrow$	$< n.a. >$	$\#, \Rightarrow$	$\#, \Rightarrow$	$0, \Rightarrow$	$0, \Rightarrow$	$0, \Rightarrow$	??????**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
126	#	$\#, \Rightarrow$	#	0-5				#	D # + 1

ECO630

ECO650

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	K	Integer	Upper index of sum. <a href="#">See Appendix B</a>
2	M	Integer	Degree of basis functions
3	PROP1	Integer	0 = nonplanar, 1 = planar
4	PROP2	Integer	0 = open curve, 1 = closed curve
5	PROP3	Integer	0 = rational, 1 = polynomial
6	PROP4	Integer	0 = nonperiodic, 1 = periodic

Let N = 1+K-M and A = N+2\*M

7	T(-M)	Real	First value of knot sequence
:	:		
7+A	T(N+M)	Real	Last value of knot sequence
8+A	W(0)	Real	First weight
:	:		
8+A+K	W(K)	Real	Last weight
9+A+K	X(0)	Real	First control point
10+A+K	Y(0)	Real	
11+A+K	Z(0)	Real	
:	:		
9+A+4*K	X(K)	Real	Last control point
10+A+4*K	Y(K)	Real	
11+A+4*K	Z(K)	Real	
12+A+4*K	V(0)	Real	Starting parameter value
13+A+4*K	V(1)	Real	Ending parameter value
14+A+4*K	XNORM	Real	Unit normal (if curve is planar)
15+A+4*K	YNORM	Real	
16+A+4*K	ZNORM	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.23 RATIONAL B-SPLINE CURVE ENTITY (TYPE 126)**

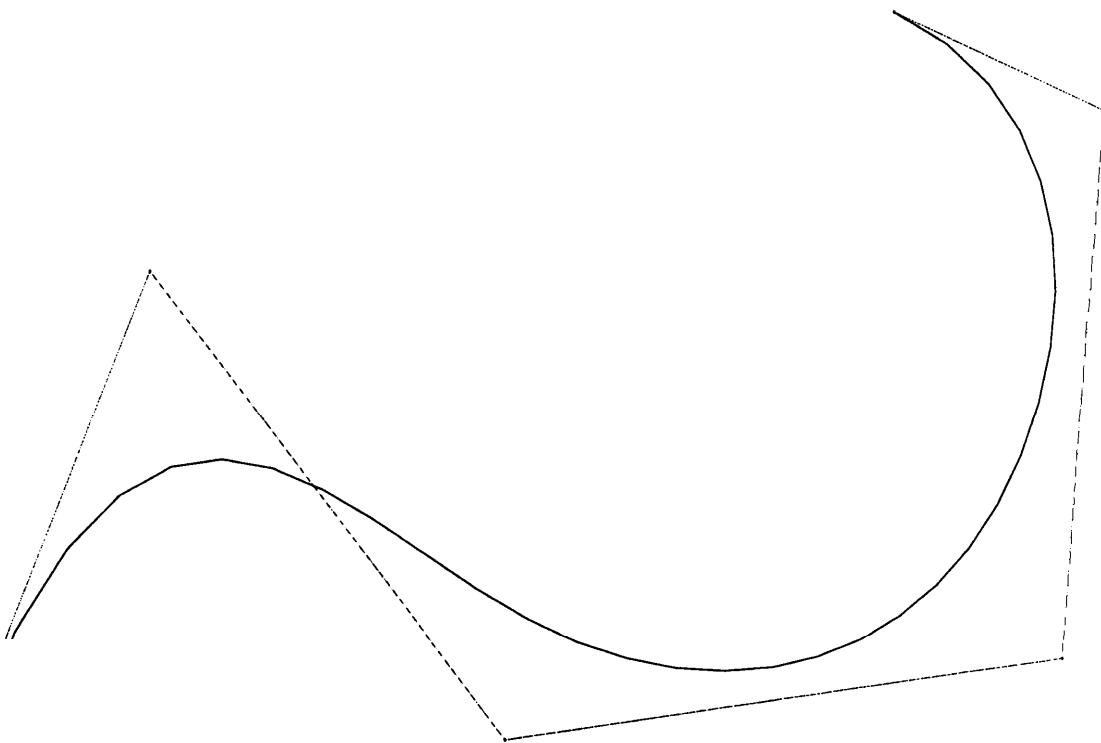


Figure 37. F126X.IGS Sample of Rational B-Spline Curve Entity

#### 4.24 Rational B-Spline Surface Entity (Type 128)

The rational B-spline surface represents various analytical surfaces of general interest. This information is important to both the generating and receiving systems. The Directory Entry Form Number Parameter is provided to communicate such information. For a brief description and a precise definition of rational B-spline surfaces, see [Appendix B](#). ECO630

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 shall be set to 2. If the surface is a surface of revolution and also a torus, the form number shall be set to 5. If the surface is not one of the preferred types, the form number shall be set to 0. ECO630

If, for each fixed value of the second parametric variable the resulting curves which are functions of the first parametric variable are closed, PROP1 shall be set to 1; otherwise, PROP1 shall be set to 0. Similarly, if for each fixed value of the first parametric variable the resulting curves which are functions of the second parametric variable are closed, PROP2 shall be set to 1; otherwise, PROP2 shall be set to 0. Mathematically, this is described as follows: ECO630

PROP1 shall be set to 1 if, and only if, for each value of  $V(0) \leq V \leq V(1)$ , the surface at  $(U(0), V)$  evaluates to the same point as it does for  $(U(1), V)$ . Correspondingly, PROP2 shall be set to 1 if, and only if, for each value of  $U(0) \leq U \leq U(1)$ , the surface at  $(U, V(0))$  evaluates to the same point as it does for  $(U, V(1))$ . ECO630

If the surface is rational (does not have all weights equal), PROP3 shall be set to 0. If all weights are equal to each other, the surface is polynomial and PROP3 shall be set to 1. The surface is polynomial since in this case all weights cancel and the denominator reduces to one (see [Appendix B](#)). The weights shall be positive real numbers. ECO630

If the surface is periodic with respect to the first parametric variable, PROP4 shall be set to 1; otherwise, PROP4 shall be set to 0. If the surface is periodic with respect to the second parametric variable, PROP5 shall be set to 1; otherwise, PROP5 shall be set to 0. The periodic flags are to be interpreted as purely informational. The surfaces which are flagged to be periodic are to be evaluated exactly the same as in the non-periodic case. ECO630

Note that the control points are in the definition space of the surface.

For the Rational B-Spline Surface Entity, the Form Numbers are as follows: ECO630

Form	Meaning
0	Form of surface is 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

## 4.24 RATIONAL B-SPLINE SURFACE ENTITY (TYPE 128)

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
128	$\Rightarrow$	$< n.a. >$	$\#, \Rightarrow$	$\#, \Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	??????**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
128	#	$\#, \Rightarrow$	#	0-9				#	D # + 1

EC0630

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	K1	Integer	Upper index of first sum. See Appendix B
2	K2	Integer	Upper index of second sum. See Appendix B
3	M1	Integer	Degree of first set of basis functions
4	M2	Integer	Degree of second set of basis functions
5	PROP1	Integer	1 = Closed in first parametric variable direction 0 = Not closed
6	PROP2	Integer	1 = Closed in second parametric variable direction 0 = Not closed
7	PROP3	Integer	0 = Rational 1 = Polynomial
8	PROP4	Integer	0 = Non-periodic in first parametric variable direction 1 = Periodic in first parametric variable direction
9	PROP5	Integer	0 = Non-periodic in second parametric variable direction 1 = Periodic in second parametric variable direction

```

Let N1 = 1+K1-M1,
N2 = 1+K2-M2,
A = N1+2*M1,
B = N2+2*M2,
C = (1+K1)*(I+K2)
10 S(-M1) Real First value of first knot sequence
:
10+A S(N1+M1) Real Last value of first knot sequence
11+A T(-M2) Real First value of second knot sequence
:
11+A+B T(N2+M2) Real Last value of second knot sequence
12+A+B W(0,0) Real First Weight
13+A+B W(1,0) Real
:
11+A+B+C W(K1,K2) Real Last Weight
12+A+B+C X(0,0) Real First Control Point
13+A+B+C Y(0,0) Real
14+A+B+C Z(0,0) Real
15+A+B+C X(1,0) Real

```

#### **4.24 RATIONAL B-SPLINE SURFACE ENTITY(TYPE128)**

16+A+B+C	Y(1,0)	Real
17+A+B+C	Z(1,0)	Real
:	:	:
9+A+B+4*C	X(K1,K2)	Real      Last Control Point
10+A+B+4*C	Y(K1,K2)	Real
11+A+B+4*C	Z(K1,K2)	Real
12+A+B+4*C	U(0)	Real      Starting value for first parametric direction
13+A+B+4*C	U(1)	Real      Ending value for first parametric direction
14+A+B+4*C	V(0)	Real      Starting value for second parametric direction
15+A+B+4*C	V(1)	Real      Ending value for second parametric direction

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### 4.25 Offset Curve Entity (Type 130)

The Offset Curve Entity defines the data necessary to determine the curve offset from a given base ECO630 curve C. This entity points to the base curve to be offset and contains the offset distance and additional pertinent information. Except as stated in the following paragraph, no restriction is placed on the entity types of curves; any parametric curve may be offset.

It is the intent of this Specification to limit the applicability of offsets to curves which are planar ECO630 and are slope-continuous. Let  $C$  denote a curve in definition space which is defined parametrically by  $r = r(t)$ , let  $T(t)$  denote the unit tangent at  $r(t)$  (See [FAUX79]), and let  $V$  be a unit vector normal to the plane which contains  $C$ . The offset curve lies in the plane which contains the base curve and is defined as follows:

$$O(t) = r(t) + f(s) \cdot (V \times T(t)); \quad TT1 \leq t \leq TT2$$

**FLAG = 1:** The offset distance is uniform;  $f(s) = D1$ .

**FLAG = 2:** The offset distance varies linearly;

$$f(s) = D1 + (D2 - D1) \cdot (s - TD1)/(TD2 - TD1)$$

with

**PTYPE = 1**

$s$  = arc length along  $r$  from  $r(TT1)$  to  $r(t)$ ,  
 $D1$  = the offset at arc length value  $TD1$ ;  
 $D2$  = the offset at arc length value  $TD2$ .

**PTYPE = 2**

$s = t$ ,  
 $D1$  = the offset at parametric value  $TD1$ ;  
 $D2$  = the offset at parametric value  $TD2$ .

**FLAG = 3:** The offset distance is defined by a function;  $f(s)$  is the NDIM-th coordinate function of the curve referenced by DE2, with

**PTYPE = 1:**

$s$  = arc length along  $r$  from  $r(TT1)$  to  $r(t)$ ;

**PTYPE = 2:**

$$s = t$$

Note that  $TT1$  and  $TT2$  shall be chosen to be in the domain of the base curve  $r(t)$ .

ECO630

**Directory Entry**

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
130	$\Rightarrow$	< n.a. >	#, $\Rightarrow$	#, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	??????**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
130	#	#, $\Rightarrow$	#	0				#	D # + 1

**Parameter Data**

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	DE1	Pointer	Pointer to the DE of the curve entity to be offset.
2	FLAG	Integer	Offset distance flag: 1 = Single value offset, uniform distance 2 = Offset distance varying linearly 3 = Offset distance as a specified function.
3	DE2	Pointer or 0	Pointer to the DE of the curve entity, one coordinate of which describes the offset as a function of its parameter. 0 unless FLAG = 3
4	NDIM	Integer	Pointer of particular coordinate of DE2 which describes offset as a function of its parameter. (only used if FLAG = 3)
5	PTYPE	Integer	Tapered offset type flag: 1 = Function of arc length 2 = Function of parameter (only used if FLAG=2 or 3)
6	D1	Real	First offset distance. (only used if FLAG=1 or 2)
7	TD1	Real	Arc length or parameter value, depending on PTYPE, of first offset distance. (only used if FLAG=2)
8	D2	Real	Second offset distance.
9	TD2	Real	Arc length or parameter value, depending on PTYPE, of second offset distance. (only used if FLAG=2)
10	VX	Real	X-component of unit vector normal to plane containing curve to be offset.
11	VY	Real	Y-component of unit vector normal to plane containing curve to be offset.
12	VZ	Real	Z-component of unit vector normal to plane containing curve to be offset.
13	TT1	Real	Offset curve starting parameter value.
14	TT2	Real	Offset curve ending parameter value.

Additional pointers as required (see Section 2.2.4.5.2).

Parameter data not required for a particular case shall be given zero values. For example, if the ECO630 value of Parameter 2 is not 3, Parameters 3 and 4 shall be given zero values.

#### 4.26 Connect Point Entity (Type 132)

A Connect Point Entity defines a point of connection for zero, one, or more entities. These entities include those required in piping diagrams, electrical and electronic schematics, and physical designs (e.g., printed wiring boards). The Connect Point Entity is referenced from either the Composite Curve (Type 102), Network Subfigure Definition (Type 320), Network Subfigure Instance (Type 420), or the Flow Associativity Instance (Type 402, Form 18). It may also appear in a file without being referenced by other entities. The connect point may be displayed by the receiving system using default display parameters or by symbols. [See Section 3.6.3.](#)

**TF.** The Type Flag (TF) is an enumerated list that specifies a particular type of connection:

TF Value	Meaning
0	Not Specified (default)
1	Nonspecific logical point of connection
2	Nonspecific physical point of connection
101	Logical component pin
102	Logical port connector
103	Logical offpage connector
104	Logical global signal connector
201	Physical PWA surface mount pin
202	Physical PWA blind pin
203	Physical PWA thru-pin
5001-9999	Implementor defined

#### 4.26 CONNECT POINT ENTITY (TYPE 132)

**FC.** The Function Code (FC) is an enumerated list that specifies a particular function for the connection:

FC Value	Meaning	FC Value	Meaning
0	Unspecified (default)	30	Reset
1	Input	31	Blanking
2	Output	32	Test
3	Input and Output	33	Address
4	Power (VCC)	34	Control
5	Ground	35	Carry
6	Anode	36	Sum
7	Cathode	37	Write
8	Emitter	38	Sense
9	Base	39	V+
10	Collector	40	Read
11	Source	41	Load
12	Gate	42	SYNC
13	Drain	43	Tri-State Output
14	Case	44	VDD
15	Shield	45	V-
16	Inverting Input	46	VEE
17	Regulated Input	47	Reference
18	Booster Input	48	Reference Bypass
19	Unregulated Input	49	Reference Supply
20	Inverting Output	98	Deferral
21	Regulated Output	99	No Connection
22	Booster Output	5001-9999	Implementor defined
23	Unregulated Output		
24	Sink		
25	Strobe		
26	Enable		
27	Data		
28	Clock		
29	Set		

(1) Entity Type Number 132	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????04??	(10) Sequence Number D #
(11) Entity Type Number 132	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** If PD Index 4 (Pointer to Display Geometry) is 0 or defaulted, Line Font Pattern, Line Weight, and Hierarchy are ignored.

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>	
1	X	Real	X coordinate of the connection point	
2	Y	Real	Y coordinate of the connection point	
3	Z	Real	Z coordinate of the connection point	
4	PTR	Pointer	Pointer to the DE of the display symbol geometry entity, or null. If null, no display symbol is specified.	ECO630
5	TF	Integer	Type flag	
6	FF	Integer	Function Flag: 0 = not specified 1 = electrical signal 2 = fluid flow path	
7	CID	String	Connect Point Function Identifier ( <i>e.g.</i> , Pin Number or Nozzle Label)	
8	PTTCID	Pointer	Pointer to the DE of the Text Display Template Entity for CID, or null. If null, no Text Display Template is specified.	ECO630
9	CFN	String	Connection Point Function Name	
10	PTTCFN	Pointer	Pointer to the DE of the Text Display Template Entity for CFN, or null. If null, no Text Display Template is specified.	ECO630
11	CPID	Integer	Unique Connect Point Identifier	
12	FC	Integer	Connect Point Function Code	
13	SF	Integer	Swap Flag 0 = Connect point may be swapped (default) 1 = Connect point may not be swapped	
14	PSFI	Pointer	Pointer to the DE of the “owner” Network Subfigure Instance Entity, Network Subfigure Definition Entity, or zero.	

Additional pointers as required (see Section 2.2.4.5.2).

#### 4.27 Node Entity (Type 134)

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. The form number of the Transformation Matrix indicates the definition coordinate system type. Coordinate angles for the cylindrical and spherical coordinate systems are specified in degrees.

Every node has an associated nodal displacement coordinate system. This is Form 10, 11, or 12 ECO630 of the Transformation Matrix Entity, which locates translational and rotational directions for load, restraint, and displacement results. Again, the form number of the Transformation Matrix indicates the coordinate system type.

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. [Figure 38](#) illustrates the definition of a node in the three coordinate systems.

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 38\(a\)](#) Cartesian.

For the cylindrical type displacement coordinate system, 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, tangential, and axial directions as illustrated in [Figure 38\(b\)](#) Cylindrical.

Finally, for spherical, the orientation of the nodal displacement axes depend on both the  $\theta$  and  $\phi$  coordinates of the node as defined in the referenced displacement coordinate system. The nodal displacement axes are respectively in the radial, meridional, and azimuthal directions as indicated in [Figure 38\(c\)](#) Spherical.

If a node lies on the polar axis of either the cylindrical or spherical coordinate system, the nodal displacement axes are defined parallel to the referenced displacement coordinate system axes. For a cylindrical system, the first axis is the  $\theta = 0$  axis and the third axis is the  $z$  axis. For a spherical system, the first axis is the  $\phi = 0$  axis while the third axis is the  $\theta = 0$  axis. The remaining axis of both systems is defined by the appropriate cross product of the previously defined axes.

**Directory Entry**

(1) Entity Type Number 134	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix ⇒	(8) Label Display < n.a. >	(9) Status Number ???04**	(10) Sequence Number D #
(11) Entity Type Number 134	(12) Line Weight < n.a. >	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Entity Subscript shall contain the Node Number. The Entity Label optionally may contain the Node Label.

**Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	X/R/R	Real	First nodal coordinate
2	Y/θ/θ	Real	Second nodal coordinate
3	Z/Z/φ	Real	Third nodal coordinate
4	NDCSP	Pointer	Pointer to the DE of the Transformation Matrix Entity Form 10, 11, or 12 which defines the Nodal Displacement Coordinate System Entity. Default (zero) is Global Cartesian Coordinate System.

Additional pointers as required (see Section 2.2.4.5.2).

**4.27 NODE ENTITY (TYPE 134)**

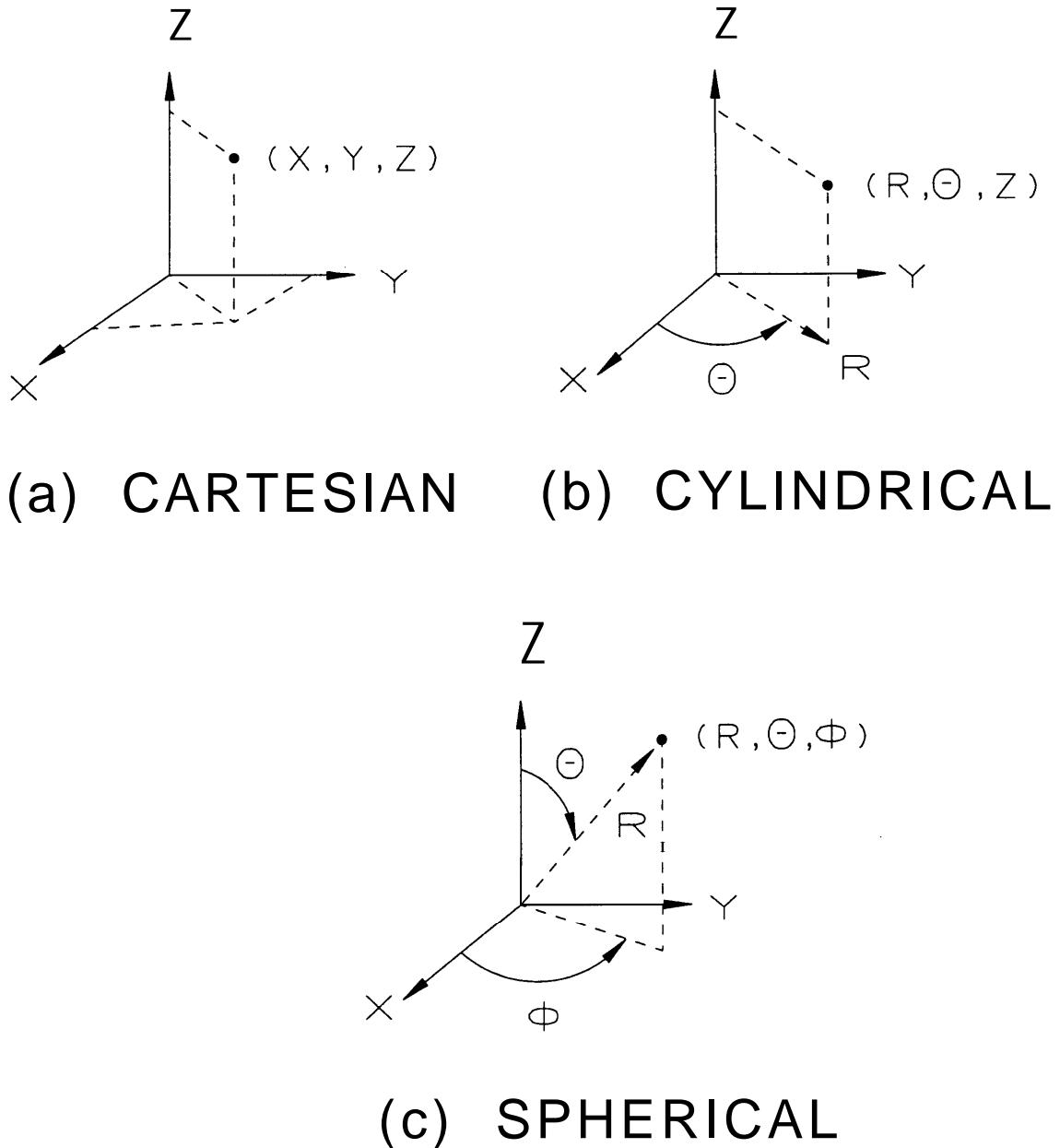


Figure 38. Nodal Displacement Coordinate Systems

**4.28 Finite Element Entity (Type 136)**

EC0630

A finite element is defined by an element topology (*i. e.*, node connectivity), along with physical and material properties. [Table 6](#) summarizes the available elements. [Table 7](#) and [Figure 39](#) through 45 illustrate the node connectivity for each element

In [Table 6](#) the element name (ETYP) is an English abbreviation or acronym describing the element. The element topology type (ITOP) is an integer number which shall appear as the first parameter of the parameter data. ITOP values greater than or equal 5001 are considered to be implementor-defined. The order is an integer identifying the order of an edge as follows:

<b>Value</b>	<b>Order of Edge</b>
0	Not applicable
1	Linear
2	Parabolic
3	Cubic

The number of nodes (N) from [Table 6](#) shall appear as the second parameter of the finite element parameter data. A missing node in the connectivity sequence shall have its corresponding pointer value set equal to zero.

**Directory Entry**

(1) Entity Type Number 136	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display 0, ⇒	(9) Status Number *****	(10) Sequence Number D #
(11) Entity Type Number 136	(12) Line Weight < n.a. >	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Entity Subscript shall contain the Element Number. The Entity Label optionally may contain the Element Label.

**Parameter Data**

EC0650

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	ITOP	Integer	Topology type
2	N	Integer	Number of nodes defining element ( <a href="#">See Section 4.27</a> ).
3	DE(1)	Pointer	Pointer to the DE of the first node defining element entity ( <a href="#">See Section 4.27</a> ).
:	:	:	
2+N	DE(N)	Pointer	Pointer to the DE of the last node defining element entity
3+N	ETYP	String	Element type name

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### 4.28 FINITE ELEMENT ENTITY (TYPE 136)

Table 6. Finite Element Topology Set

<b>Element Name (ETYP)</b>	<b>Element Topology Type (ITOP)</b>	<b>Order</b>	<b>Number of Nodes (N)</b>	<b>Number of Edges</b>	<b>Number of Faces</b>
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
TBEAM	33	1	3	1	0
OMASS	34‡	0	2	0	0
OFBEAM	35‡	1	4	1	0
PBEAM	36‡	2	3	1	0
CBEAM	37‡	2	3	1	0
CPSOW	38‡	3	21	9	5

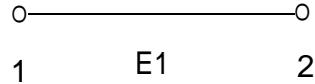
‡ Note: Elements 34-38 are untested.

#### 4.28 FINITE ELEMENT ENTITY (TYPE 136)

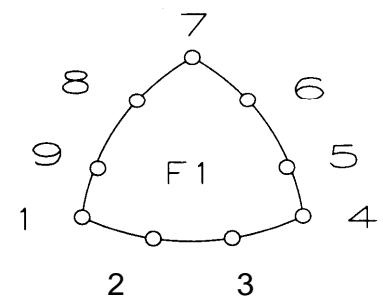
Table 7. Finite Element Topology

Element Type	Element Name	Edges	Faces
1.	BEAM	E1=1,2	
2.	LTRIA Linear Triangle	E1=1,2 E2=2,3 E3=3,1	F1=1,2,3
3.	PTRIA Parabolic Triangle	E1=1,2,3 E2=3,4,5 E3=5,6,1	F1=1,2,3,4,5,6
4.	CTRIA Cubic Triangle	E1=1,2,3,4 E2=4,5,6,7 E3=7,8,9,1	F1=1,2,3,4,5,6,7,8,9
5.	LQUAD Linear Quadrilateral	E1=1,2 E2=2,3 E3=3,4 E4=4,1	F1=1,2,3,4
6.	PQUAD Parabolic Quadrilateral	E1=1,2,3 E2=3,4,5 E3=5,6,7 E4=7,8,1	F1=1,2,3,4,5,6,7,8

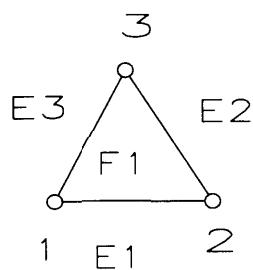
Refer to [Figure 39](#).



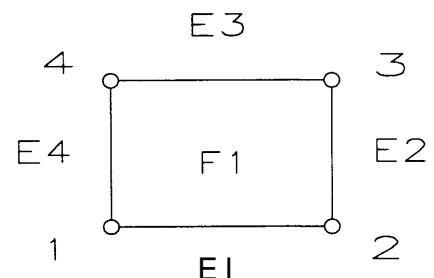
**1. BEAM**



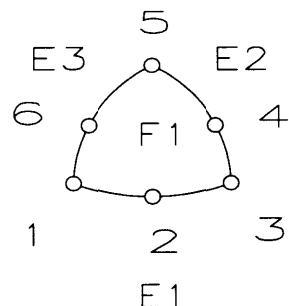
**4. CUBIC TRIANGLE**



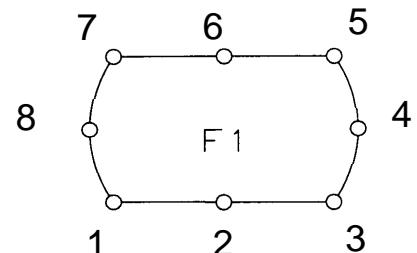
**2. LINEAR TRIANGLE**



**5. LINEAR QUADRILATERAL**



**3. PARABOLIC TRIANGLE**



**6. PARABOLIC QUADRILATERAL**

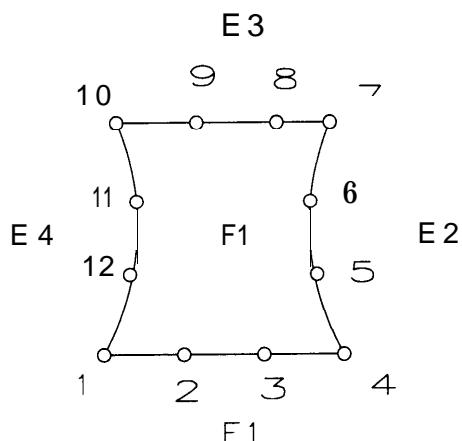
Figure 39. Finite Element Topology Set

## 4.28 FINITE ELEMENT ENTITY (TYPE 136)

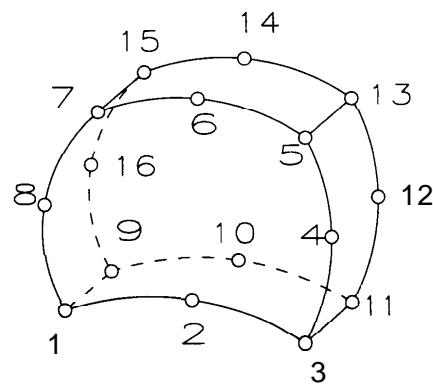
Table 7. Finite Element Topology (continued)

<b>Element Type</b>	<b>Element Name</b>	<b>Edges</b>	<b>Faces</b>
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=5,6,1,7,12,11
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 .	P T S 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=5,6,7,15,14,13 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 F5=7,8,9,10,22,21,20,19 F6=10,11,12,1,13,24,23,22

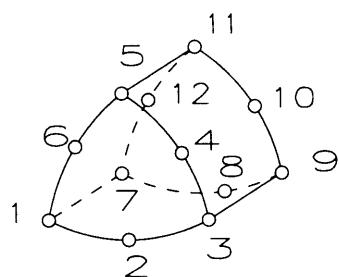
Refer to [Figure 40](#).



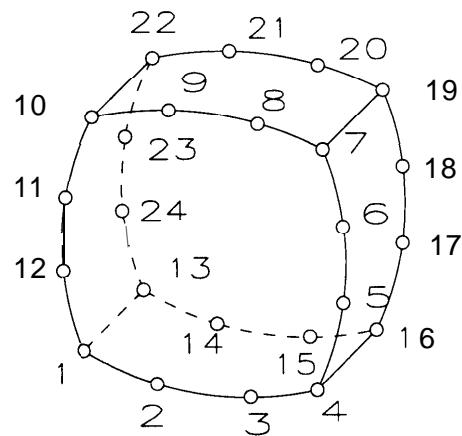
**7. CUBIC  
QUADRILATERAL**



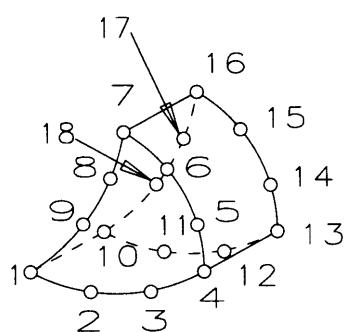
**10. PARABOLIC  
THICK SHELL**



**8. PARABOLIC THICK  
SHELL WEDGE**



**11. CUBIC  
THICK SHELL**



**9. CUBIC THICK  
SHELL WEDGE**

Figure 40. Finite Element Topology Set (continued)

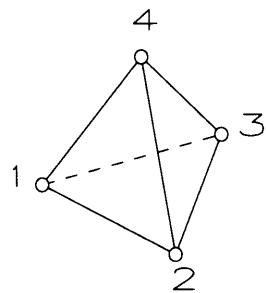
## 4.28 FINITE ELEMENT ENTITY (TYPE 136)

Table 7. Finite Element Topology (Continued)

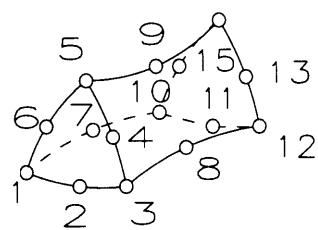
<b>Element Type</b>	<b>Element Name</b>	<b>Edges</b>	<b>Faces</b>
12.	LSOT Linear Solid Tetrahedron	E1=1,2 E2=2,3 E3=3,1 E4=1,4 E5=2,4 E6=3,4	F1=1,2,3 F2=1,2,4 F3=2,3,4 F4=3,1,4
13.	PSOT Parabolic Solid Tetrahedron	E1=1,2,3 E2=3,4,5 E3=5,6,1 E4=1,7,10 E5=3,8,10 E6=5,9,10	F1=1,2,3,4,5,6 F2=1,2,3,8,10,7 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=16,17,18,19,20,21,22,23,24 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

Refer to [Figure 41](#).

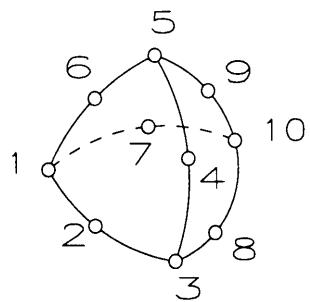
Table 7. Finite Element Topology (continued)



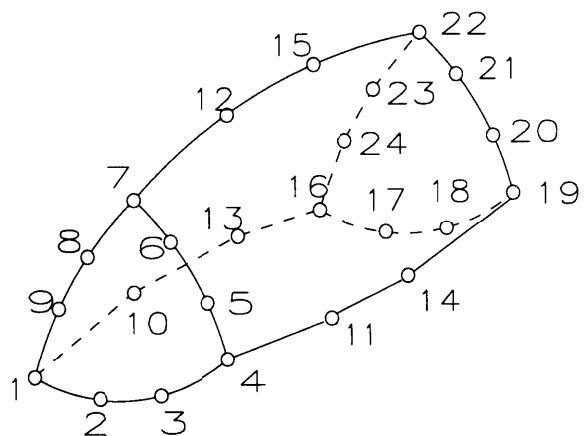
**12. LINEAR SOLID  
TETRAHEDRON**



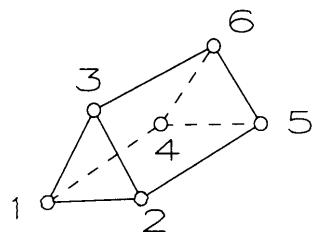
**15. PARABOLIC SOLID  
WEDGE**



**13. PARABOLIC SOLID  
TETRAHEDRON**



**16. CUBIC SOLID  
WEDGE**



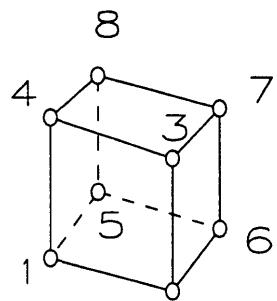
**14. LINEAR SOLID  
WEDGE**

Figure 41. Finite Element Topology Set (continued)

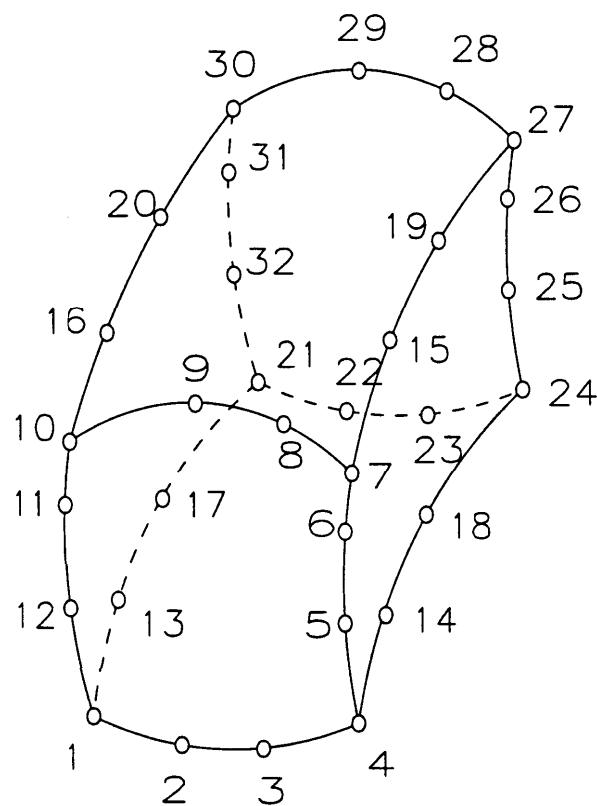
#### 4.28 FINITE ELEMENT ENTITY (TYPE 136)

<b>Element Type</b>	<b>Element Name</b>	<b>Edges</b>	<b>Faces</b>
17.	LSO Linear Solid	E1=1,2 E2=2,3 E3=3,4 E4=4, 1 E5=5,6 E6=6,7 E7=7,8 E8=8,5 E9=1,5 E10=2,6 E11=3,7 E12=4,8	F1=1,2,3,4 F2=5,6,7,8 F3=1,2,6,5 F4=2,3,7,6 F5=3,4,8,7 F6=4,1,5,8
18.	PSO Parabolic Solid	E1=1,2,3 E2=3,4,5 E3=5,6,7 E4=7,8,1 E5=13,14,15 E6=15,16,17 E7=17,18,19 E8=19,20,13 E9=1,9,13 E10=3,10,15 E11=5,11,17 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 F4=3,4,5,11,17,16,15,10 F5=5,6,7,12,19,18,17,11 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=1,2,3,4,5,6,7,8,9,10,11,12 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

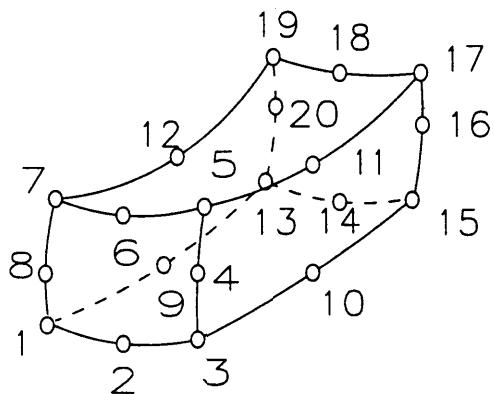
Refer to [Figure 42](#).



**17. LINEAR SOLID**



**19. CUBIC SOLID**



**18. PARABOLIC SOLID**

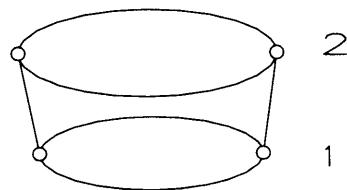
Figure 42. Finite Element Topology Set (continued)

## 4.28 FINITE ELEMENT ENTITY (TYPE 136)

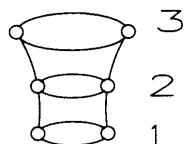
Table 7. Finite Element Topology (continued)

<b>Element Type</b>	<b>Element Name</b>	<b>Edges</b>	<b>Faces</b>
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 E3=3,1	No Faces
24.	APTRIA Axisymmetric Parabolic Triangle	E1=1,2,3 E2=3,4,5 E3=5,6,1	No Faces
25.	ALQUAD Axisymmetric Linear Quadrilateral	E1=1,2 E2=2,3 E3=3,4 E4=4,1	No Faces
26.	APQUAD Axisymmetric Parabolic Quadrilateral	E1=1,2,3 E2=3,4,5 E3=5,6,7 E4=7,8,1	No Faces

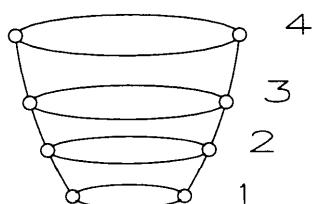
Refer to [Figure 43](#).



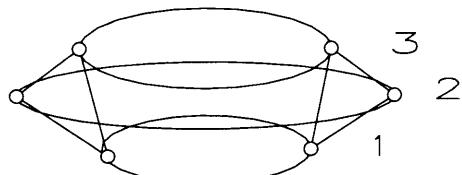
**20. AXISYMMETRIC LINEAR LINE**



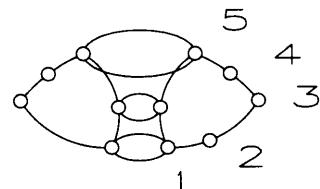
**21. AXISYMMETRIC PARABOLIC LINE**



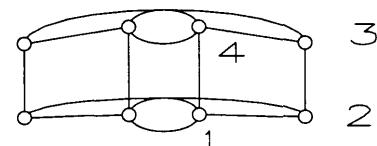
**22. AXISYMMETRIC CUBIC LINE**



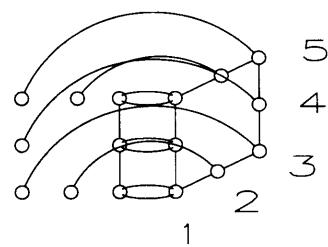
**23. AXISYMMETRIC LINEAR TRIANGLE**



**24. AXISYMMETRIC PARABOLIC TRIANGLE**



**25. AXISYMMETRIC LINEAR QUADRILATERAL**



**26. AXISYMMETRIC PARABOLIC QUADRILATERAL**

Figure 43. Finite Element Topology Set (continued)

#### **4.28 FINITE ELEMENT ENTITY (TYPE 136)**

Table 7. Finite Element Topology (continued)

<b>Element Type</b>	<b>Element Name</b>	<b>Edges</b>	<b>Faces</b>
27.	SPR Spring	No edges	No faces
28.	GSPR Grounded Spring		
29.	DAMP Damper		
30.	GDAMP Grounded damper		
31.	MASS Mass		
32.	RBDY Rigid Body		
33.	TBEAM Three-Noded Beam	E1 = 1,2	No faces

Refer to [Figure 44](#).

**4.28 FINITE ELEMENT ENTITY (TYPE 136)**



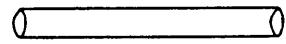
27. SPRING



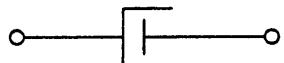
31. MASS



28. GROUNDED SPRING



32. RIGID BODY

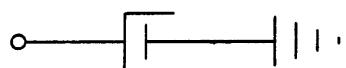


29. DAMPER

•3

1      o ————— o      2

E1



30. GROUNDED DAMPER

33. THREE NODED BODY

Figure 44. Finite Element Topology Set (continued)

#### 4.28 FINITE ELEMENT ENTITY (TYPE 136)

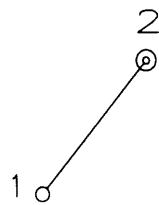
Table 7. Finite Element Topology (continued)

<b>Element Type</b>	<b>Element Name</b>	<b>Edges</b>	<b>Faces</b>
34.	OFMASS Offset Mass		Node 2 specifies the center of mass.
35.	OFBEAM Offset Beam	E1 = 3,4	
36.	PBEAM Three Node Beam	E1 = 1,2,3	
37.	CBEAM Curved Beam	E1 = 1,2	(Part of a circle) A <45 degrees
38.	CPSOW Cubic/Parabolic Solid Wedge	E1 = 1,2,3,4 E2 = 4,5,6,7 E3 = 7,8,9,1 E4 = 13,14,15,16 E5 = 16,17,18,19 E6 = 19,20,21,13 E7 = 1,10,13 E8 = 4,11,16 E9 = 7,12,19	F1 = 1,2,3,4,5,6,7,8,9 F2 = 1,2,3,4,11,16,15,14,13,10 F3 = 4,5,6,7,12,19,18,17,16,11 F4 = 7,8,9,1,10,13,21,20,19,12 F5 = 13,14,15,16,17,18,19,20,21
5001.	Implementor-Defined		

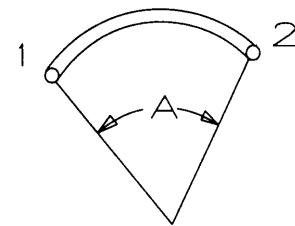
Refer to [Figure 45](#).

Note: Elements 34-38 and 5001 are untested

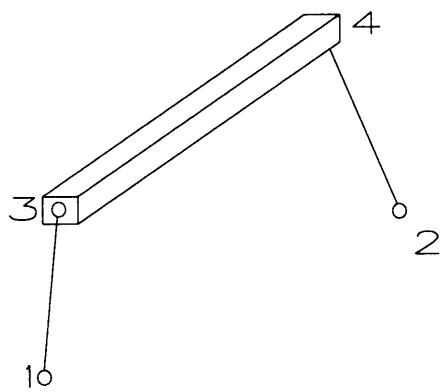
#### 4.28 FINITE ELEMENT ENTITY (TYPE 136)



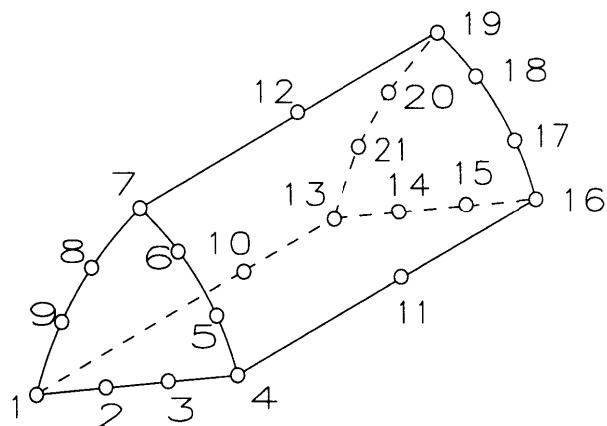
34. OFFSET MASS



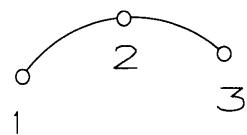
37. CURVED BEAM



35. OFFSET BEAM



38. CUBIC/PARABOLIC  
SOLID WEDGE



36. THREE NODE  
BEAM

Figure 45. Finite Element Topology Set (continued)

## 4.29 NODAL DISPLACEMENT AND ROTATION ENTITY (TYPE 138)

### 4.29 Nodal Displacement and Rotation Entity (Type 138)

The Nodal Displacement and Rotation Entity is used to communicate finite element postprocessing data. It contains the incremental displacements and rotations (expressed in radians) for each load case and each node in the model. It also contains a pointer to a General Note Entity (Type 212) for a description of the load cases. For each node it contains the node number identifier and the node DE pointer. The node number identifier is equivalent to the node number in the Directory Entry subscript field of the Node Entity (Type 134).

#### Directory Entry

EC0630

(1) Entity Type Number 138	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	I (7) Xformation Matrix < n.a. >	I (8) Label Display 0,⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 138	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	I (17) Reserved	I (18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

EC0650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NC	Integer	Number of analysis cases
2	GP(1)	Pointer	Pointer to the DE of the general note that describes the first analysis case
:	:		
1+NC	GP(NC)	Pointer	Pointer to the DE of the general note that describes the last analysis case
2+NC	NN	Integer	Number of nodes
3+NC	NO(1)	Integer	Node number identifier for first node
4+NC	NP(1)	Pointer	Pointer to the DE of the Node Directory Entry
5+NC	X(1,1)	Real	X-Incr. translation, first analysis case
6+NC	Y(1,1)	Real	Y-Incr. translation
7+NC	Z(1,1)	Real	Z-Incr. translation
8+NC	RX(1,1)	Real	RX-Incr. rotation
9+NC	RY(1,1)	Real	RY-Incr. rotation
10+NC	RZ(1,1)	Real	RZ-Incr. rotation
:	:		
-1+7*NC	X(1,NC)	Real	X-Incr. translation, last analysis case
7*NC	Y(1,NC)	Real	Y-Incr. translation
1+7*NC	Z(1,NC)	Real	Z-Incr. translation
2+7*NC	RX(1,NC)	Real	RX-Incr. rotation
:	:		
3+NC+(-1+NN)*(2+6*NC)	NO(NN)	Integer	Node number identifier for NNth node
4+NC+(-1+NN)*(2+6*NC)	NP(NN)	Pointer	Pointer to the DE of the Node Directory Entry
5+NC+(-1+NN)*(2+6*NC)	X(NN,1)	Real	X-Incr. translation, first analysis case
6+NC+(-1+NN)*(2+6*NC)	Y(NN,1)	Real	

## 4.29 NODAL DISPLACEMENT AND ROTATION ENTITY (TYPE 138)

7+NC+(-1+NN)*(2+6*NC)	Z(NN,1)	Real	
8+NC+(-1+NN)*(2+6*NC)	RX(NN,1)	Real	RX-Incr. rotation, first analysis case
9+NC+(-1+NN)*(2+6*NC)	RY(NN,1)	Real	
10+NC+(-1+NN)*(2+6*NC)	RZ(NN,1)	Real	
.	:	:	
-3+NC+NN*(2+6*NC)	X(NN,NC)	Real	X-Incr. translation, last analysis case
-2+NC+NN*(2+6*NC)	Y(NN,NC)	Real	
-1+NC+NN*(2+6*NC)	Z(NN,NC)	Real	
NC+NN*(2+6*NC)	RX(NN,NC)	Real	RX-Incr. rotation, last analysis case
1+NC+NN*(2+6*NC)	RY(NN,NC)	Real	
2+NC+NN*(2+6*NC)	RZ(NN,NC)	Real	

Additional pointers as required (see Section 2.2.4.5.2).

#### 4.30 Offset Surface Entity (Type 140)

The offset surface is a surface defined in terms of an existing surface.

ECO630

Let  $S = S(u, v)$  be a surface defined by this Specification, parameterized and oriented by  $N(u, v)$ , a differentiable field of unit normal vectors defined on the whole surface, and  $d$ , a fixed, nonzero real number. An offset surface to  $S$  is a parameterized surface  $O$  ( $u, v$ ) given by:

$$O(u, v) = S(u, v) + d \cdot N(u, v); \quad u1 \leq u \leq u2 \\ v1 \leq v \leq v2.$$

The base surface  $S(u, v)$  is referenced by a pointer in the parameter data section, while  $N(u, v)$  is found from  $S(u, v)$  as defined below. The value of  $d$  is provided as a parameter value in the parameter data section.

To determine which one of the two orientations of the orientable regular surface  $S(u, v)$  the offset surface will be used to define  $O$ , define

$$N(u, v) = \frac{\partial S / \partial u \times \partial S / \partial v}{\| \partial S / \partial u \times \partial S / \partial v \|}.$$

In order to avoid confusion with respect to the orientation of the base surface  $S(u, v)$ , an additional offset indicator is included. That indicator, shown in [Figure 46](#), consists of the vector  $(Nx, Ny, Nz)$  defined by the unit normal vector at the parameter values  $(Um, Vm)$ :

$$(Nx, Ny, Nz) = \frac{N(Um, Vm)}{\| N(Um, Vm) \|},$$

where, if the surface is bounded,

$$Um = (u1 + u2)/2 \text{ and } Vm = (v1 + v2)/2,$$

or, if the surface is unbounded,

$$Um = 0.0 \text{ and } Vm = 0.0.$$

This indicates the direction in which the offset distance,  $d$ , is measured positive at  $(Um, Vm)$ .

**CAUTION:** The vector  $(Nx, Ny, Nz)$  is simply an indicator of the direction with respect to the base surface  $S(u, v)$  where the offset distance,  $d$ , is measured positively. This vector does not participate in the evaluation of the offset surface as is evident from the formula for  $O$  that defines the offset surface.

**Directory Entry**

(1) Entity Type Number 140	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 140	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Parameter Data**

EC0630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NX	Real	The X-coordinate of the end of the offset indicator $N(Um, Vm)$
2	NY	Real	The Y-coordinate of the end of the offset indicator $N(Um, Vm)$
3	NZ	Real	The Z-coordinate of the end of the offset indicator $N(Um, Vm)$
4	D	Real	The distance by which the surface is normally offset on the side of the offset indicator if $d > 0$ and on the opposite side if $d < 0$
5	DE	Pointer	Pointer to the DE of the surface entity to be offset

Additional pointers as required (see Section 2.2.4.5.2).

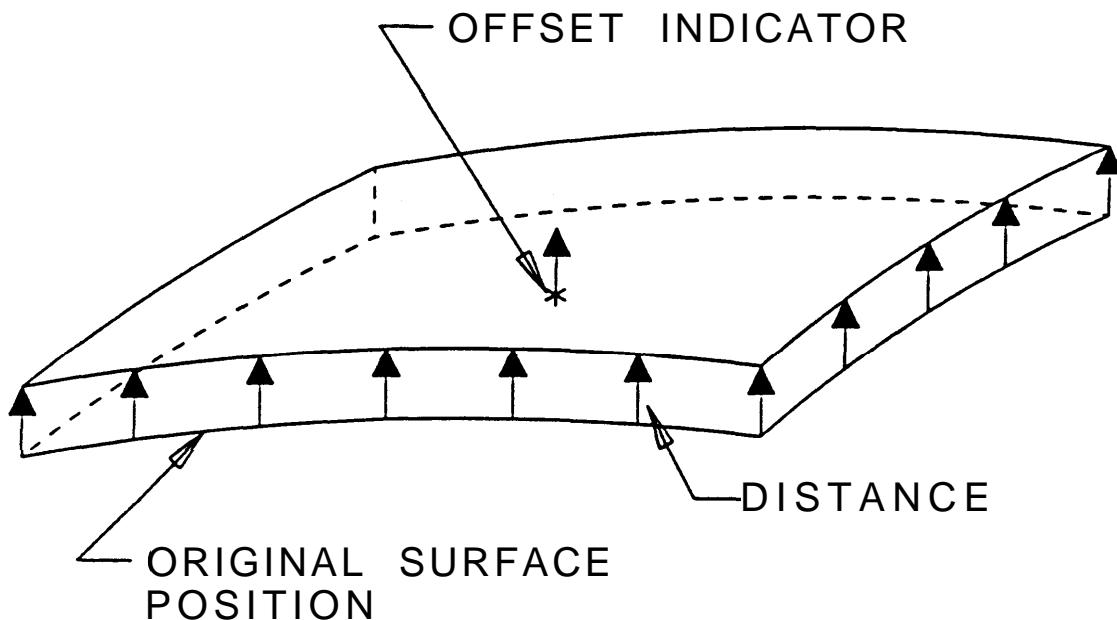


Figure 46. Offset Surface in 3-D Euclidean Space

**4.31 Boundary Entity (Type 141)**

ECO652

Each Boundary Entity (Type 141) identifies a surface boundary consisting of a set of curves lying on the surface. The properties of the surface, the boundary, and the curves comprising the boundary are defined below:

D1.  $S(u, v)$  may be used as a parameterized surface representation with the Boundary Entity (Type 141) if it meets the following criteria:

- (a) The untrimmed domain of  $S(u, v)$  is a rectangle,  $D$ , consisting of those points  $(u, v)$  such that  $a \leq u \leq b$  and  $c \leq v \leq d$  for given constants  $a, b, c$ , and  $d$  with  $a < b$  and  $c < d$ .
- (b) The mapping  $S = S(u, v) = (x(u, v), y(u, v), z(u, v))$  is defined for each ordered pair  $(u, v)$  in  $D$ .
- (c) It is one-to-one in the interior (but not necessarily on the boundary) of  $D$ .
- (d) It has continuous normal vectors at every point of  $D$  except those which map to poles (see definition D3).

D2. The isoparametric curves  $u = a$ ,  $u = b$ ,  $v = c$ , and  $v = d$  will be referred to as boundary curves of the parameter space or simply boundary curves.

D3. Let  $P$  be a 3-D Euclidean (model space) point. Then  $P$  is a pole of the surface defined by the mapping  $S(u, v)$  if any of the following are true:

- (a)  $P = S(a, v)$  for all  $v$  such that  $a \leq v \leq d$
- (b)  $P = S(b, v)$  for all  $v$  such that  $a \leq v \leq d$
- (c)  $P = S(u, c)$  for all  $u$  such that  $a \leq u \leq b$
- (d)  $P = S(u, d)$  for all  $u$  such that  $a \leq u \leq b$

D4. Let  $C$  be a 3-D Euclidean (model space) curve. Then  $C$  is a seam of the surface defined by the mapping  $S(u, v)$  if it is the image in model space of

- (a)  $C(v) = S(a, v)$  for all  $v$  such that  $c \leq v \leq d$  and  
 $C(v) = S(b, v)$  for all  $v$  such that  $c \leq v \leq d$   
or
- (b)  $C(u) = S(u, c)$  for all  $u$  such that  $a \leq u \leq b$  and  
 $C(u) = S(u, d)$  for all  $u$  such that  $a \leq u \leq b$

D5. A model space curve is represented parametrically, lies on the surface, and does not intersect itself except possibly at its endpoints.

D6. A boundary is an ordered list of model space curves ( $C_i, i = 1, n$ ) which has the following properties:

- (a) It is closed. This implies that the endpoint of  $C_n$  is the startpoint of  $C_1$ .
- (b) Each curve in the list is oriented such that the endpoint of the curve  $C_{i-1}$  is the startpoint of the curve  $C_i, i = 2, n$ .
- (c) It is not self-intersecting except at its endpoints.

The endpoints of the boundary are the startpoint of  $C_1$  and the endpoint of  $C_n$ . It does not intersect other boundaries except at isolated points (refer to D10(b) for related requirements).

#### 4.31 BOUNDARY ENTITY (TYPE 141)

- D7. The usage of a model-space trimming curve is oriented. It is part of an ordered list forming a boundary.
- D8. The positive surface normal is given by the cross product (in the order specified) of the partial derivative of  $S(u, v)$  with respect to  $u$  and the partial derivative of  $S(u, v)$  with respect to  $v$ .
- D9. The terminology “left of a model space trimming curve at a point  $p$ ” means “the direction of the vector formed as the cross product (in the order specified) of the surface normal and the tangent vector to the model space trimming curve at  $p$ .”
- D10. The region of the surface being communicated is called the active region; it shall satisfy the following:
- (a) The active region has finite area.
  - (b) Any two points on the interior of the active region shall be path-connected.
  - (c) The interior of the active region lies on the left of all of its boundaries.
  - (d) The active region consists of all of its boundaries and its interior.
  - (e) The closure of the interior of the active region (in the relative topology of the surface reduced by  $R3$ ) is the active region.
- D11.  $C_{a_i}$  is an associated parameter space curve of an arc,  $C_a$ , of a model space trimming curve,  $C$ , on the surface,  $S$ , with domain  $D$ , if  $C_{a_i}$  is contained in  $D$  and the composition  $S \circ C_{a_i} = C_a$ . An associated parameter space curve is assumed to be represented parametrically, and it shall not intersect itself except possibly at its endpoints.
- D12. An associated parameter space curve collection (or simply “collection”) is defined to be the associated parameter space curves ( $C_i$ ,  $i = 1, p$ ) such that the  $C_i$  given by the composition  $(S \circ C_i)$ ,  $i = 1, p$  form a composite curve. The  $C_i$  of the composite curve are ordered and oriented such that as the parameter goes from its initial to final value the complete model space trimming curve is produced in the direction indicated by the model space curve's orientation flag SENSE. ECO652

Figure 47 shows valid and invalid examples of a boundary.

The  $C_i$  forming the associated parameter space collections of a boundary are not required to satisfy the “closed” property for a boundary (see definition D6). The  $C_i$  can be formed into a boundary by adding the appropriate sections of the boundary curves of the parameter space (see definition D2).

**Directory Entry**

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
141	$\Rightarrow$	$< n.a. >$	$\#, \Rightarrow$	$\#, \Rightarrow$	$0, \Rightarrow$	$0, \Rightarrow$	$0, \Rightarrow$	??????**	D #
141	(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript
	#	#	#, $\Rightarrow$	#	0			#	D # + 1

**Parameter Data**

EC0650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	TYPE	Integer	The type of bounded surface representation: 0 = The boundary entities shall reference only model space trimming curves. The associated surface representation (located by SPTR) may be parametric. 1 = The boundary entities shall reference model space curves and associated parameter space curve collections. The associated surface (located by SPTR) shall be a parametric representation.
2	PREF	Integer	Indicates the preferred representation of the trimming curves in the sending system: 0 = Unspecified 1 = Model space 2 = Parameter space 3 = Representations are of equal preference
3	SPTR	Pointer	Pointer to the DE of the untrimmed surface entity to be bounded. If associated parameter space curves are being transferred (TYPE = 1) the surface representations shall be parametric.
4	N	Integer	Number of curves included in this boundary entity (N > 0)
5	CRVPT(1)	Pointer	Pointer to the DE of the first model space curve entity of this Boundary Entity
6	SENSE(1)	Integer	An orientation flag indicating whether the direction of the first model space curve should be reversed before use in the boundary. The possible values for the sense flag are: 1 = The direction of the model space curve does not require reversal; PSCPT and CRVPT orientations agree. 2 = The direction of the model space curve needs to be reversed; PSCPT and CRVPT orientations disagree.
7	K(1)	Integer	Number of associated parameter space curves in the collection for the first model space trimming curve. In the case of a TYPE = 0 transfer, this count shall be zero.
8	PSCPT(1,1)	Pointer	Pointer to the DE of the first associated parameter space entity curve of the collection for the first model space trimming curve
:	:	:	
7+K(1)	PSCPT(1,K(1))		Pointer Pointer to the DE of the last associated parameter space curve entity of the collection for the first model space trimming curve

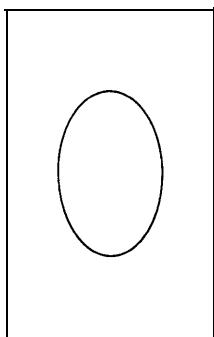
EC0652

#### 4.31 BOUNDARY ENTITY (TYPE 141)

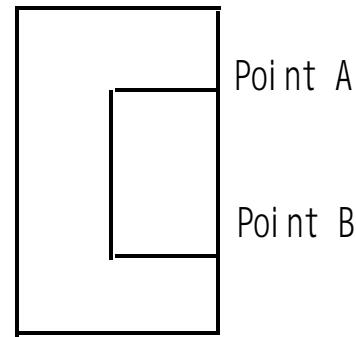
$\vdots$ M $1+M$ $2+M$ $3+M$ $\vdots$ $2+K(N)+M$	$\vdots$ CRVPT(N) Pointer SENSE(N) Integer K(N) Integer PSCPT(N,1) Pointer $\vdots$ PSCPT(N,K(N))	Let $M = 12 + 3*(N-1) + (K(1) + K(2) + \dots + K(N-1))$ Pointer to the DE of the last model space curve entity in this Boundary Entity An orientation flag indicating whether the direction of the last model space curve should be reversed before use in the boundary. The possible values for the sense flag are: 1 = The direction of the model space curve does not require reversal; PSCPT and CRVPT orientations agree. 2 = The direction of the model space curve needs to be reversed; PSCPT and CRVPT orientations disagree. Number of associated parameter space curves in the collection for the last model space trimming curve Pointer to the DE of the first associated parameter space curve entity of the collection for the last model space trimming curve $= 0$ transfer, this count shall be zero. Pointer to the DE of the last associated parameter space curve entity of the collection for the last model space trimming curve	
			ECO652

Additional pointers as required ([see Section 2.2.4.5.2](#)).

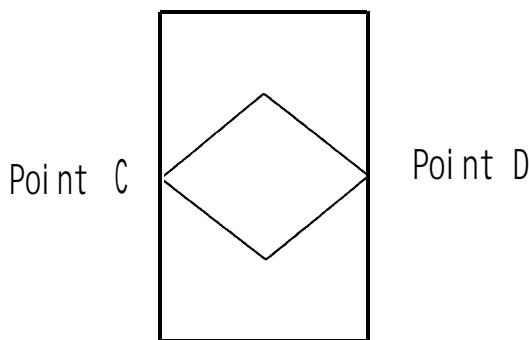
#### 4.31 BOUNDARY ENTITY (TYPE 141)



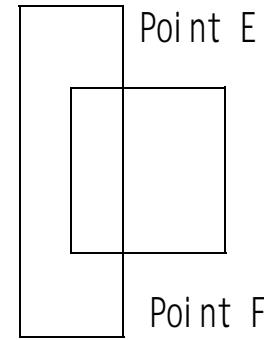
Valid



Invalid: violates D6  
(intersection is a segment )



Invalid



Invalid: violates D10

- Both loops share a common segment between Point A and Point B
- The outside loop and inside loop touch at points C and D
- At points E and F, the loops cross another segment of a different loop.

Figure 47. Examples of the Boundary Entity

## 4.32 CURVE ON A PARAMETRIC SURFACE ENTITY (TYPE 142)

### 4.32 Curve on a Parametric Surface Entity (Type 142)

The Curve on a Parametric Surface Entity associates a given curve with a surface and identifies the curve as lying on the surface. Let

$$S = S(u, v) = (x(u, v), y(u, v), z(u, v))$$

be a regular parameterized surface whose domain is a rectangle defined by

$$D = \{(u, v) \mid u1 \leq u \leq u2 \text{ and } v1 \leq v \leq v2\}.$$

Let  $B = B(t)$  be a curve defined by

$$B(t) = (u(t), v(t)) \text{ for } a \leq t \leq b,$$

taking its values in  $D$ .

A curve  $C_c(t)$  on the surface  $S(u, v)$  is the composition of two mappings,  $S$  and  $B$ , defined as follows: ECO630 ( $C_c(t)$  stands for “composition curve.” )

$$\begin{aligned} C_c(t) &\triangleq S \circ B(t) \\ &\triangleq S(B(t)) \\ &\triangleq S(u(t), v(t)) \\ &\triangleq (x(u(t), v(t)), y(u(t), v(t)), z(u(t), v(t))) \quad a \leq t \leq b. \end{aligned}$$

The curve  $B$  lies in the two dimensional space which is the domain of the surface  $S$ . Therefore, the representation used for  $B$  which has been derived from a curve defined in this Specification must be two dimensional: the  $X$  and  $Y$  coordinates of this curve pointed to by BPTR are used.

The Entity Use Flag (DE Field 9) of the entity  $B$  is set to 05, indicating that  $B$  is in the parameter space of the surface. Consequently,  $B$  cannot be scaled, and, if a transformation matrix is to be applied on  $B$ , it has to map it within the parameter space  $D$  in which it resides.

A curve on a parametric surface is given by:

1. the mapping  $C_c$  and an indication that the curve lies on the surface  $S(u, v)$
2. the mappings  $B$  and  $S$  whose composition gives the curve  $C_c$ .

A curve on a surface may have been created in one of a number of various ways:

1. as the projection on the surface of a given curve in model space in a prescribed way, for example, parallel to a given fixed vector
2. as the intersection of two given surfaces
3. by a prescribed functional relation between the surface parameters  $u$  and  $v$
4. by a special curve, such as a geodesic, emanating from a given point in a certain direction, ECO630 a principal curve (line of curvature) emanating from a certain point, an asymptotic curve emanating from a certain point, an isoparametric curve for a given value, or any other kind of special curve.

#### **4.32 CURVE ON A PARAMETRIC SURFACE ENTITY (TYPE 142)**

The Parameter Data section contains three pointers:

1. a pointer to the curve from which  $B(t)$  is derived
2. a pointer to the surface  $S(u, v)$
3. a pointer to a mapping  $C(r)$ , such that:
  - $C(r)$  and  $C_c(t)$  share the same image in model space.
  - $C(r)$  and  $C_c(t)$  have the same start and end points.
  - An implicit mathematical relationship exists between the parameters  $t$  and  $r$ .
  - $C(r)$  and  $C_c(t)$  must be such that  $t$  is related to  $r$  in a monotonically increasing fashion. This ensures that the orientations of  $C(r)$  and  $C_c(t)$  coincide, and no accidental multiple tracing of either curve occurs.

It also contains:

1. a flag to indicate how the curve was created
2. a flag to indicate which of the two alternate representations was preferred by the sending system.

## 4.32 CURVE ON A PARAMETRIC SURFACE ENTITY (TYPE 142)

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number	ECO630
142	$\Rightarrow$	$< n.a. >$	#, $\Rightarrow$	#, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	????00**	D #	
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number	D # + 1

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	CRTN	Integer	Indicates the way the curve on the surface has been created: 0 = Unspecified 1 = Projection of a givencurve on the surface 2 = Intersection of two surfaces 3 = Isoparametric curve, <i>i.e.</i> , either a <i>u</i> - parametric or a <i>v</i> - parametric curve
2	SPTR	Pointer	Pointer to the DE of the surface on which the curve lies
3	BPTR	Pointer	Pointer to the DE of the entity that contains the definition of the curve <i>B</i> in the parametric space ( <i>u</i> , <i>v</i> ) of the surface <i>S</i>
4	CPTR	Pointer	Pointer to the DE of the curve <i>C</i>
5	PREF	Integer	Indicates preferred representation in the sending system: 0 = Unspecified 1 = <i>S</i> o <i>B</i> is preferred 2 = <i>C</i> is preferred 3 = <i>C</i> and <i>S</i> o <i>B</i> are equally preferred

Additional pointers as required (see section 2.2.4.5.2).

### 4.33 Bounded Surface Entity (Type 143)

The Bounded Surface Entity (Type 143) is used to represent trimmed surfaces. The surface and ECO652 trimming curves are assumed to be represented parametrically and to comply with the definitions ECO630 D1 through D12 listed in [Section 4.31](#).

Two types of transfer are supported by the bounded surface. A TYPE = 0 transfer represents a ECO630 surface and its model space boundaries. A TYPE = 1 transfer represents a surface, its model space boundaries, and the associated parameter space curve collection for each model space trimming curve of each boundary. Because of seams and poles, the associated parameter space curve collections of a boundary do not necessarily enclose a region in parameter space.

The bounded surface information is represented using several entities. These are the Bounded Surface Entity (Type 143), the Boundary Entity ([Type 141](#)), the parametrically represented untrimmed surface entities, and the parametrically represented curve entities.

#### Directory Entry

(1) Entity Type Number 143	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????00**	(10) Sequence Number D #
(11) Entity Type Number 143	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	TYPE	Integer	The type of bounded surface representation: 0 = The boundary entities shall reference only model space curves. The associated surface representation (located by SPTR) may be parametric. 1 = The boundary entities shall reference both model space curves and the associated parameter space curve collections. The associated surface (located by SPTR) shall be a parametric representation.
2	SPTR	Pointer	Pointer to the DE of the untrimmed surface entity to be bounded. If parameter space trimming curves are being transferred (TYPE = 1) the surface representations shall be parametric.
3	N	Integer	The number of boundary entities
4	BDPT(1)	Pointer	Pointer to the DE of the first Boundary Entity ( <a href="#">Type 141</a> )
:	:	:	
3+N	BDPT(N)	Pointer	Pointer to the DE of the last Boundary Entity ( <a href="#">Type 141</a> )

Additional pointers as required (see [Section 2.2.4.5.2](#)).

## 4.34 TRIMMED (PARAMETRIC) SURFACE ENTITY (TYPE 144)

### 4.34 Trimmed (Parametric) Surface Entity (Type 144)

A simple closed curve in the Euclidean plane divides the plane into two disjoint open connected components, one bounded and one unbounded. The bounded component is called the interior region to the curve (herein called "interior" and the unbounded component is called the exterior region to the curve (herein called "exterior").

The domain of the trimmed surface is defined as the common region of the interior of the outer boundary and the exterior of each of the inner boundaries and includes the boundary curves. Note that the trimmed surface has the same mapping  $S(u, v)$  as the original (untrimmed surface) but a different domain. The curves that delineate either the outer or the inner boundary of the trimmed surface are curves on the surface  $S$ , and are to be exchanged by means of the Curve on a Parametric Surface Entity (Type 142).

Let  $S(u, v)$  be a regular parameterized surface, whose untrimmed domain is a rectangle  $D$  consisting of those points  $(u, v)$  such that  $a \leq u \leq b$  and  $c \leq v \leq d$  for given constants  $a, b, c$ , and  $d$  with  $a < b$  and  $c < d$ . Assume that  $S$  takes its values in three-dimensional Euclidean space so that it can be expressed as:

$$S = S(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix}$$

for each ordered pair  $(u, v)$  in  $D$ .

Also let the mapping  $S$  be subject to the following regularity conditions:

- It has a continuous normal vector in the interior of  $D$ . ECO630
- It is one-to-one in  $D$ .
- There are no singular points in  $D$ , i.e., the vectors of the first partial derivatives of  $S$  at any point in  $D$  are linearly independent.

Two types of simple closed curves are utilized to define the domain of the trimmed (parametric) surface.

**Outer boundary** There is exactly one. It lies in  $D$ , and in particular, it can be the boundary curve of  $D$ .

**Inner boundary** There can be any number of them, including zero. The set of inner boundaries satisfies two criteria:

1. The curves, as well as their interiors, are mutually disjoint. ECO630
2. Each curve lies in the interior of the outer boundary.

If the outer boundary of the surface being defined is the boundary of  $D$  and there are no inner boundaries, the trimmed surface being defined is untrimmed.

## 4.34 TRIMMED (PARAMETRIC) SURFACE ENTITY (TYPE 144)

### Directory Entry

(1) Entity Type Number 144	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????00**	(10) Sequence Number D #
(11) Entity Type Number 144	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>	
1	PTS	Pointer	Pointer to the DE of the surface entity that is to be trimmed	
2	N1	Integer	0 = the outer boundary is the boundary of D 1 = otherwise	
3	N2	Integer	This number indicates the number of simple closed curves which constitute the inner boundary of the trimmed surface. In case no inner boundary is introduced, this is set equal to zero.	
4	PTO	Pointer	Pointer to the DE of the Curve on a Parametric Surface Entity that constitutes the outer boundary of the trimmed surface or zero	ECO630
5	PTI(1)	Pointer	Pointer to the DE of the first simple closed inner boundary curve entity (Curve on a Parametric Surface Entity) according to some arbitrary ordering of these entities	
:	:	:		
4+N2	PTI(N2)	Pointer	Pointer to the DE of the last simple closed inner boundary curve entity (Curve on a Parametric Surface Entity)	

Additional pointers as required (see Section 2.2.4.5.2).

### **4.35 Nodal Results Entity (Type 146)‡**

‡The Nodal Results Entity has not been tested. See Section 1.9.

The number of analysis results data values per FEM node and their physical interpretation depends ECO630 upon specified values of the form number (TYPE) and NV (see Table 8). Also, the node number identifier shall be equal to the node number in the directory entry subscript field of the node entity.

#### **Directory Entry**

(1) Entity Type Number 146	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level <i>&lt; n.a. &gt;</i>	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display 0, ⇒	(9) status Number **?03**	(10) Sequence Number D #
(11) Entity Type Number 146	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number TYPE	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Entity Subscript field shall contain the Analysis Case Number. The Entity Label field optionally may contain the Analysis Label.

The value of TYPE (see Table 8) indicates the physical interpretation of the finite element analysis ECO630 results data. For a specific TYPE of data, multiple values are positioned within the Parameter Data record in the order in which they appear in the parenthetical expression in the description column of the table.

#### **Parameter Data**

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	GNOTE	Pointer	Pointer to the DE of the General Note Entity that describes the analysis case.
2	SCN	Integer	Analysis Subcase number. If there is no subcase, the value of this parameter shall be zero.
3	TIME	Real	Analysis time value used for this subcase. (This time value is not the time that the analysis was executed, nor does it have anything to do with the amount of time that a computer took to execute the job. It is the time at which transient analysis results occur in the mathematical FEM model.)
4	NV	Integer	Number of real values in array V for a FEM node. (The value of NV shall agree with the form number specified in the Directory Data, see Table 8.)
5	NN	Integer	Number of FEM nodes for which data is to be read.
6	NODE(1)	Integer	FEM node number identifier for first node.
7	NP(1)	Pointer	Pointer to the DE of the first FEM Node Entity
8	V(i)	Real	Values of the finite element analysis results data array for the first FEM node. There are NV data values in array V.
:	:	:	loop over number of nodes, NN

In subsequent index equations, let NNV = (NV+2)\*(NN-1)

#### **4.35 NODAL RESULTS ENTITY (TYPE 146)‡**

6+NNV	NODE(NN)	Integer FEM node number identifier for last node.
7+NNV	NP(NN)	Pointer Pointer to the DE of the last FEM Node Entity
8+NNV	V(i)	Real Values of the finite element analysis results data array for the last FEM node. There are NV data values in array V.

Additional pointers as required (see [Section 2.2.4.5.2](#)).

Table 8. Description of TYPE Numbers for the Nodal and Element Results Entities

Type	NV	Description
0	nv	Unknown/Miscellaneous (The number of values, nv, is not predefined for form type 0. The value of nv shall always be positive.)
1	1	Temperature
2	1	Pressure
3	3	Total Displacement (xx, yy, zz - consistent with the Nodal Displacement Coordinate System)
4	6	Total Displacement and Rotation (Dxx, Dyy, Dzz, Rxx, Ryy, Rzz - consistent with the Nodal Displacement Coordinate System)
5	3	Velocity
6	3	Velocity Gradient
7	3	Acceleration
8	3	Flux
9	3	Elemental Force
10	1	Strain Energy
11	1	Strain Energy Density
12	3	Reaction Force
13	1	Kinetic Energy
14	1	Kinetic Energy Density
15	3	Hydrostatic Pressure
16	1	Coefficient of Pressure
17	3	Symmetric 2-Dimensional Elastic Stress Tensor (xx, yy, xy)
18	3	Symmetric 2-Dimensional Total Stress Tensor (xx, yy, xy)
19	3	Symmetric 2-Dimensional Elastic Strain Tensor (xx, yy, xy)
20	3	Symmetric 2-Dimensional Plastic Strain Tensor (xx, yy, xy)
21	3	Symmetric 2-Dimensional Total Strain Tensor (xx, yy, xy)
22	3	Symmetric 2-Dimensional Thermal Strain (xx, yy, xy)
23	6	Symmetric 3-Dimensional Elastic Stress Tensor (xx, yy, zz, xy, yz, zx)
24	6	Symmetric 3-Dimensional Total Stress Tensor (xx, yy, zz, xy, yz, zx)
25	6	Symmetric 3-Dimensional Elastic Strain Tensor (xx, yy, zz, xy, yz, zx)
26	6	Symmetric 3-Dimensional Plastic Strain Tensor (xx, yy, zz, xy, yz, zx)
27	6	Symmetric 3-Dimensional Total Strain Tensor (xx, yy, zz, xy, yz, zx)
28	6	Symmetric 3-Dimensional Thermal Strain (xx, yy, zz, xy, yz, zx)
29	9	General Elastic Stress Tensor (xx, yx, zx, xy, yy, zy, xz, yz, zz)
30	9	General Total Stress Tensor (xx, yx, zx, xy, yy, zy, xz, yz, zz)
31	9	General Elastic Strain Tensor (xx, yx, zx, xy, yy, zy, xz, yz, zz)
32	9	General Plastic Strain Tensor (xx, yx, zx, xy, yy, zy, xz, yz, zz)
33	9	General Total Strain Tensor (xx, yx, zx, xy, yy, zy, xz, yz, zz)
34	9	General Thermal Strain (xx, yx, zx, xy, yy, zy, xz, yz, zz)

**4.36 Element Results Entity (Type 148) ‡**

‡The Element Results Entity has not been tested. See Section 1.9.

The number of results data values depends upon: (1) NV, the number of results data values per ECO630 reporting location; (2) NRL, the number of results data reporting locations in a FEM element per layer; and (3) NL, the number of layers in the FEM element. The physical interpretation and location of the results data depends upon: (1) TYPE, the type of results data which is specified by using the form number in the Directory Data section (see Table 8); (2) RRF, the results reporting flag which associates results data with FEM element location; and (3) DLF, the data layer flag which specifies the FEM element layer location of the results data.

**Directory Entry**

(1) Entity Type Number 148	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display 0,⇒	(9) Status Number **?03**	(10) Sequence Number D #
(11) Entity Type Number 148	(12) Line Weight < n.a. >	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number TYPE	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Entity Subscript field shall contain the Analysis Case Number. The Entity Label field optionally may contain the Analysis Label.

The value of TYPE (see Table 8) indicates the physical interpretation of the finite element analysis ECO630 results data. For a specific TYPE of data, multiple values are positioned within the Parameter Data record in the order in which they appear in the parenthetical expression in the description column of the table.

**Parameter Data**

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	GNOTE	Pointer	Pointer to the DE of the General Note Entity that describes the analysis case.
2	SCN	Integer	Analysis Subcase number. If there is no subcase, then the value of this parameter shall be zero.
3	TIME	Real	Analysis time value used for this subcase. (This time value is not the time that the analysis was executed, nor does it have anything to do with the amount of time that a computer took to execute the job. It is the time at which transient analysis results occur in the mathematical model.)
4	NV	Integer	Number of results values per FEM element reporting location. (The value of NV shall agree with the form number specified in the Directory Data; see Table 8.)
5	RRF	Integer	Results Reporting Flag. This flag is used to associate the data with a FEM location. The following values are possible: 0 - Indicates that the results data pertain to the FEM element's nodes. 1 - Indicates that the results data pertain to the FEM element's centroid.

#### 4.36 ELEMENT RESULTS ENTITY (TYPE 148) ‡

				2- Indicates that the results data are constant on all faces and throughout the entire volume of the FEM element. 3- Indicates that the results data pertain to the FEM element's Gauss points (reserved for future definition).
6	NE	Integer		Number of FEM elements defined in this entity.
7	EN(1)	Integer		FEM element number identifier for first element.
8	EP(1)	Pointer		Pointer to the DE of the first FEM Element Entity.
9	ITOP ( 1)	Integer		Element Topology type of first FEM element.
10	NL(1)	Integer		Number of layers per results data report location. This parameter, along with the form number, indicates the total number of results values to be read for a particular FEM element.
11	DLF(1)	Integer		Data Layer Flag. This flag indicates other information necessary to interpret the actual layer position of the data. Five values are possible. They are:  0 - Indicates that a layer is not special. (NL shall be 1 for this case.) 1- Indicates the layer is the top surface of a FEM plate element. (NL shall be 1 for this case.) 2 - Indicates the layer is the middle surface of a FEM plate element. (NL shall be 1 for this case.) 3 - Indicates the layer is the bottom surface of a FEM plate element. (NL shall be 1 for this case.) 4 - Indicates the layers are an ordered set of values from the top to the bottom surface of a FEM element. There are NL individual layers.
12	NRL(1)	Integer		Number of results data report locations for first FEM element.
13	RDRL(I)	Integer		The results data report locations for the FEM element. The values of RDRL depends on the results reporting flag, RRF. If RRF is: 0 - These are the node numbers for this FEM element at which results values are reported. There are NRL of them 1- This is FEM element centroidal results data. NRL shall be 1 and this value shall be zero. 2- This is FEM element constant results data. NRL shall be 1 and this value shall be zero. 3- These are a topologically ordered list of Gauss points (reserved for future definition). There are NRL values for RDRL.
:	:	:		
13+NRL	NUMV(1)	Integer		This value represents the total number of results contained in the following V array. It is the product of NV, NL, and NRL for this FEM element; e.g., for FEM element number one, NUMV(1) = NV*NL(1)*NRL(1).

## 4.36 ELEMENT RESULTS ENTITY (TYPE 148) ‡

14+NRL	V(J,K,L)	Real	The results data values of the FEM analysis for the first FEM element. The results data values are arranged in column major order; <i>i.e.</i> , the leftmost subscript changes most rapidly. The subscripts are: (1) J is the value number that is incremented from 1 to NV ( <a href="#">see Table 8</a> ); (2) K is the layer number that is incremented from 1 to NL(I); and (3) L is the results data report location index that is incremented from 1 to NRL(I). (The subscript I indicates that these values are dependent upon a particular FEM element.)
--------	----------	------	---

The loop through the V array is done by using the following FORTRAN code fragment:

```

DO 10 L = 1, NRL(I)
      DO 20 K = 1, NL(I)
            DO 30 J=1, NV
                  READ(unit,*) V(J,K,L)
                  30 CONTINUE
            20 CONTINUE
      10 CONTINUE

```

There are NUMV values for array V.

(loop over number of elements)

In subsequent index equations, let NLS =  $\sum (7 + (NL * NV + l) * NRL(I))$ ; where I = 1 to NE-1 and NE represents the number of elements. Also, let NLSE = NLS + NRL(NE).

7+NLS	EN(NE)	Integer	FEM element number identifier for last element.
8+NLS	EP(NE)	Pointer	Pointer to the DE of the last FEM Element Entity.
9+NLS	ITOP(NE)	Integer	Element Topology type of last FEM element.
10+NLS	NL (NE)	Integer	Number of layers per results data report location for last FEM element.
11+NLS	DLF(NE)	Integer	Data Layer Flag of last FEM element.
12+NLS	NRL(NE)	Integer	Number of results data report locations for the last FEM element.
13+NLS	RDRL(I)	Integer	The results data location list for the last FEM element.
13+NLSE	NUMV(NE)	Integer	This value represents the total number of results contained in the V array for the last FEM element.
14+NLSE	V(J,K,L)	Real	The results data values of the FEM element analysis for the last FEM element.

Additional pointers as required ([see Section 2.2.4.5.2](#)).

### **4.37 Block Entity (Type 150)**

The block is a rectangular parallelepipeds, defined with one vertex at (X1,Y1,Z1) and three edges lying along the local +X, +Y, and +Z axes. [Figure 48](#) shows an example. The local X-axis is defined by the unit vector (I1,J1,K1) and the local Z-axis by (I2,J2,K2). The local Y-axis is derived by taking the cross product of Z into X. The resulting local system shall be orthogonal, with (I1,J1,K1) values having the highest accuracy precedence. The block is specified by the positive lengths (LX, LY,LZ) along these axes as shown in [Figure 48](#).

#### **Directory Entry**

(1) Entity Type Number 150	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformat ion Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????00**	(10) Sequence Number D #
(11) Entity Type Number 150	(12) Line Weight #	(13) Color Number #,⇒	(14) Par ameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### **Parameter Data**

ECO630

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	LX	Real	Length in the local X-direction
2	LY	Real	Length in the local Y-direction
3	LZ	Real	Length in the local Z-direction
4	X1	Real	Corner point coordinates (default (0.0,0.0,0.0))
5	Y1	Real	
6	Z1	Real	
7	I1	Real	Unit vector defining local X-axis (default (1.0,0.0,0.0))
8	J1	Real	
9	K1	Real	
10	I2	Real	Unit vector defining local Z-axis (default (0.0,0.0,1.0))
11	J2	Real	
12	K2	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

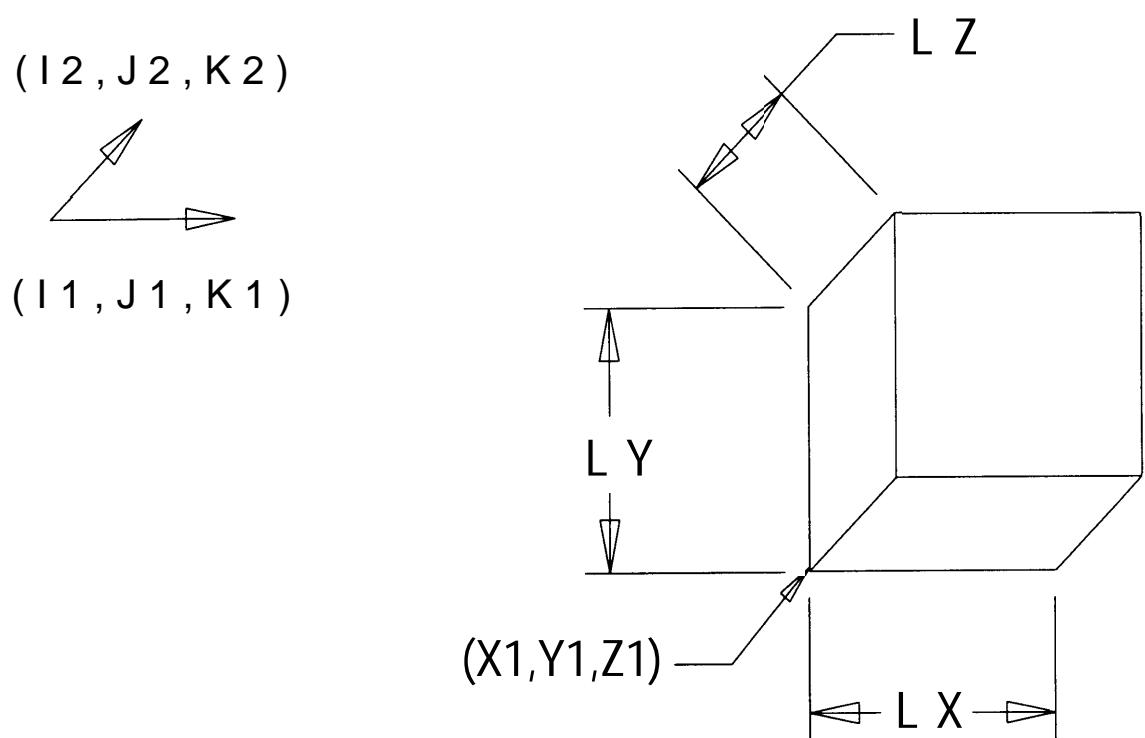


Figure 48. Parameters of the CSG Block Entity

**4.38 Right Angular Wedge Entity (Type 152)**

The right angular wedge is defined with one vertex at (X1,Y1,Z1) and three orthogonal edges lying ECO630 along the local +X, +Y, and +Z axes. [Figure 49](#) shows an example. A triangular/trapezoidal face lies in the local XY-plane. The local X-axis is defined by the unit vector (I1,J1,K1) and the local Z-axis by (I2, J2,K2). The local Y-axis is derived by taking the cross product of Z into X. The resulting local system shall be orthogonal, with (I1,J1,K1) values having the highest accuracy precedence. The wedge is specified by the positive lengths LX, LY, LZ along these axes and the length LTX (where LTX<LY) in the local positive X-direction at a distance LY (in the local Y-direction) from the local X-axis. If LTX=0, the wedge has five faces, two of which are triangular; otherwise, it has six faces.

**Directory Entry**

(1) Entity Type Number 152	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????00**	(10) Sequence Number D #
(11) Entity Type Number 152	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Parameter Data**

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	LX	Real	Length in the local X-direction at Y=0.0
2	LY	Real	Length in the local Y-direction
3	LZ	Real	Length in the local Z-direction
4	LTX	Real	Length in the local X-direction at distance LY from local X-axis
5	X1	Real	Corner point coordinates (default (0.0,0.0,0.0))
6	Y1	Real	
7	Z1	Real	
8	I1	Real	Unit vector defining local X-axis (default (1.0,0.0,0.0))
9	J1	Real	
10	K1	Real	
11	I2	Real	Unit vector defining local Z-axis (default (0.0,0.0,1.0))
12	J2	Real	
13	K2	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.38 RIGHT ANGULAR WEDGE ENTITY (TYPE 152)**

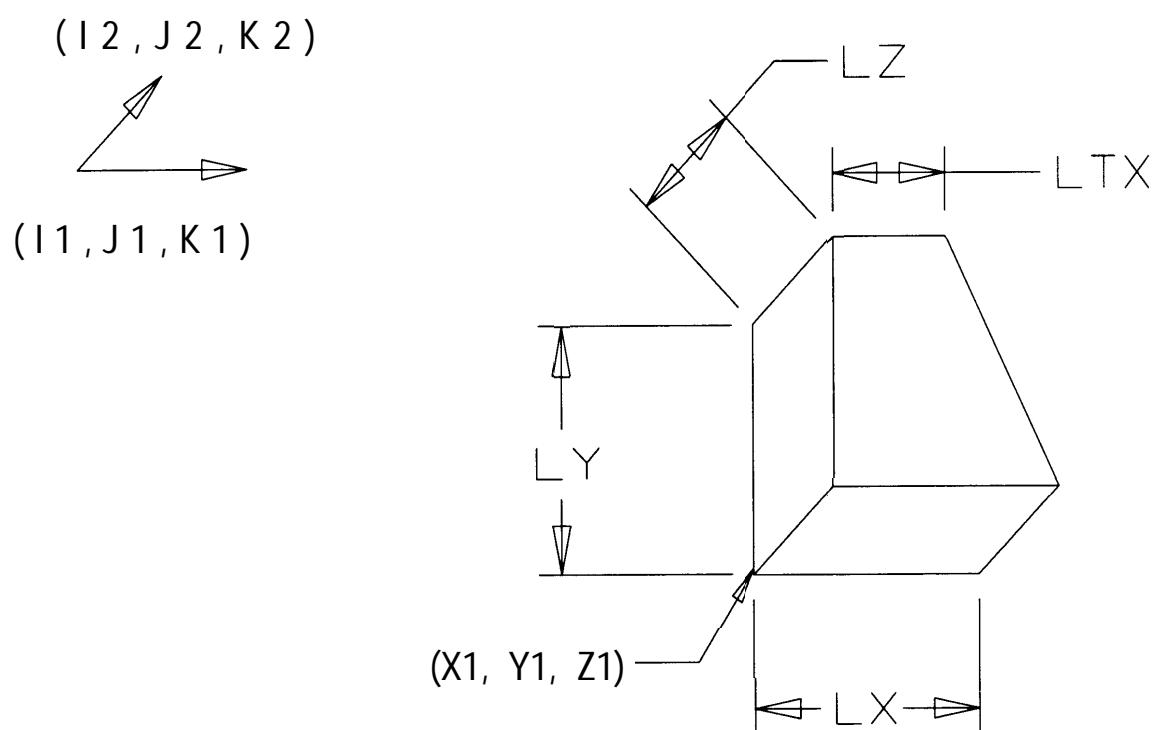


Figure 49. Parameters of the CSG Right Angular Wedge Entity

## 4.39 RIGHT CIRCULAR CYLINDER ENTITY (TYPE 154)

### 4.39 Right Circular Cylinder Entity (Type 154)

The right circular cylinder is defined by the center of one circular cylinder face, a unit vector, a ECO630 height, and a radius as shown in [Figure 50](#). The faces are perpendicular to the unit vector in the axis direction (I1,J1,K1) and are circular discs with the specified radius R (where R> 0.0). The height H (where H> 0.0) is the distance from the first circular face center in the positive direction of the unit vector to the second circular face center.

#### Directory Entry

(1) Entity Type Number 154	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????00**	(10) Sequence Number D #
(11) Entity Type Number 154	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	H	Real	Cylinder height
2	R	Real	Cylinder radius
3	X1	Real	First face center coordinates (default (0.0,0.0,0.0))
4	Y1	Real	
5	Z1	Real	
6	I1	Real	Unit vector in axis direction (default (0.0,0.0,1.0))
7	J1	Real	
8	K1	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.39 RIGHT CIRCULAR CYLINDER ENTITY (TYPE 154)**

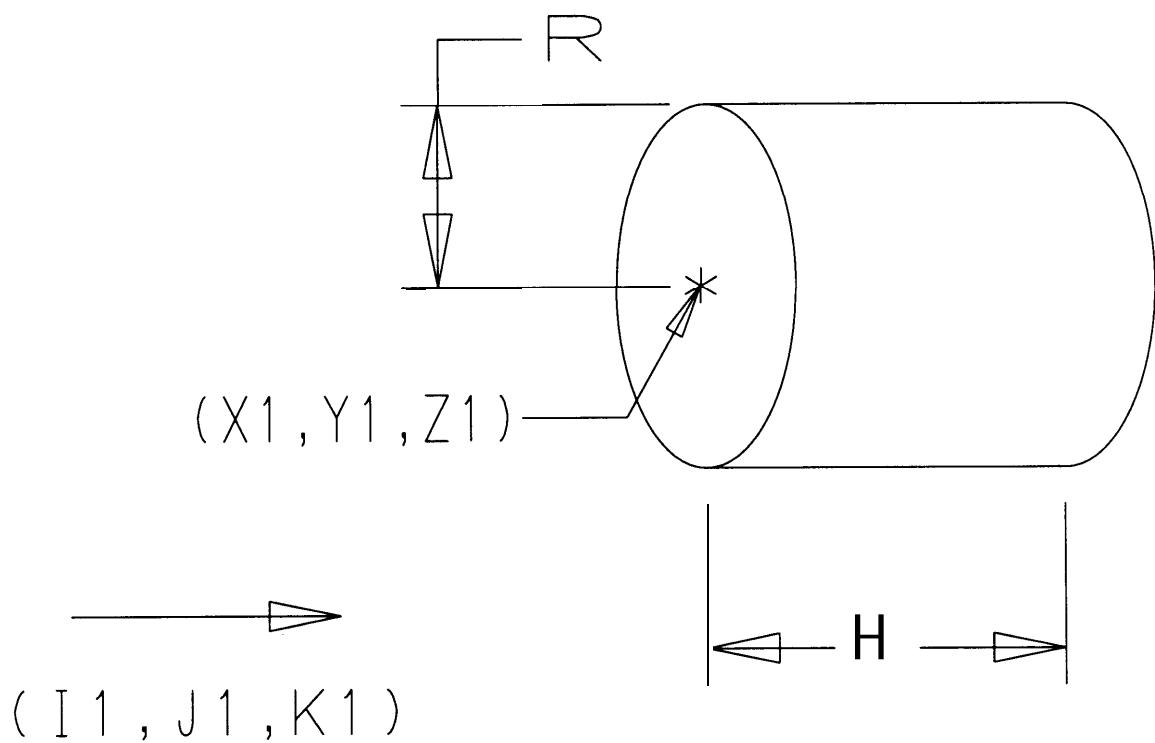


Figure 50. Parameters of the CSG Right Circular Cylinder Entity

## 4.40 RIGHT CIRCULAR CONE FRUSTUM ENTITY (TYPE 156)

### 4.40 Right Circular Cone Frustum Entity (Type 156)

The right circular cone frustum is defined by the center of the larger circular face of the frustum ECO630 (X1,Y1,Z1), its radius R1, a unit vector in the axis direction (I1,J1,K1), a height H in this direction, and a second circular face with radius R2, where  $R1 > R2 \geq 0.0$  and  $H > 0.0$ . As shown by [Figure 51](#), the circular faces are perpendicular to the unit vector (I1,J1,K1).

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
156	$\Rightarrow$	$< n.a. >$	$\#, \Rightarrow$	$\#, \Rightarrow$	$0, \Rightarrow$	$0, \Rightarrow$	$0, \Rightarrow$	????00**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
156	#	#, $\Rightarrow$	#	0				#	D # + 1

#### Parameter Data

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	H	Real	Height
2	R1	Real	Larger face radius
3	R2	Real	Smaller face radius (zero for cone apex - default)
4	X1	Real	Larger face center coordinates (default (0.0,0.0,0.0))
5	Y1	Real	
6	Z1	Real	
7	I 1	Real	Unit vector in axis direction (default (0.0,0.0,1.0))
8	J 1	Real	
9	K1	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.40 FLIGHT CIRCULAR CONE FRUSTUM ENTITY (TYPE 156)**

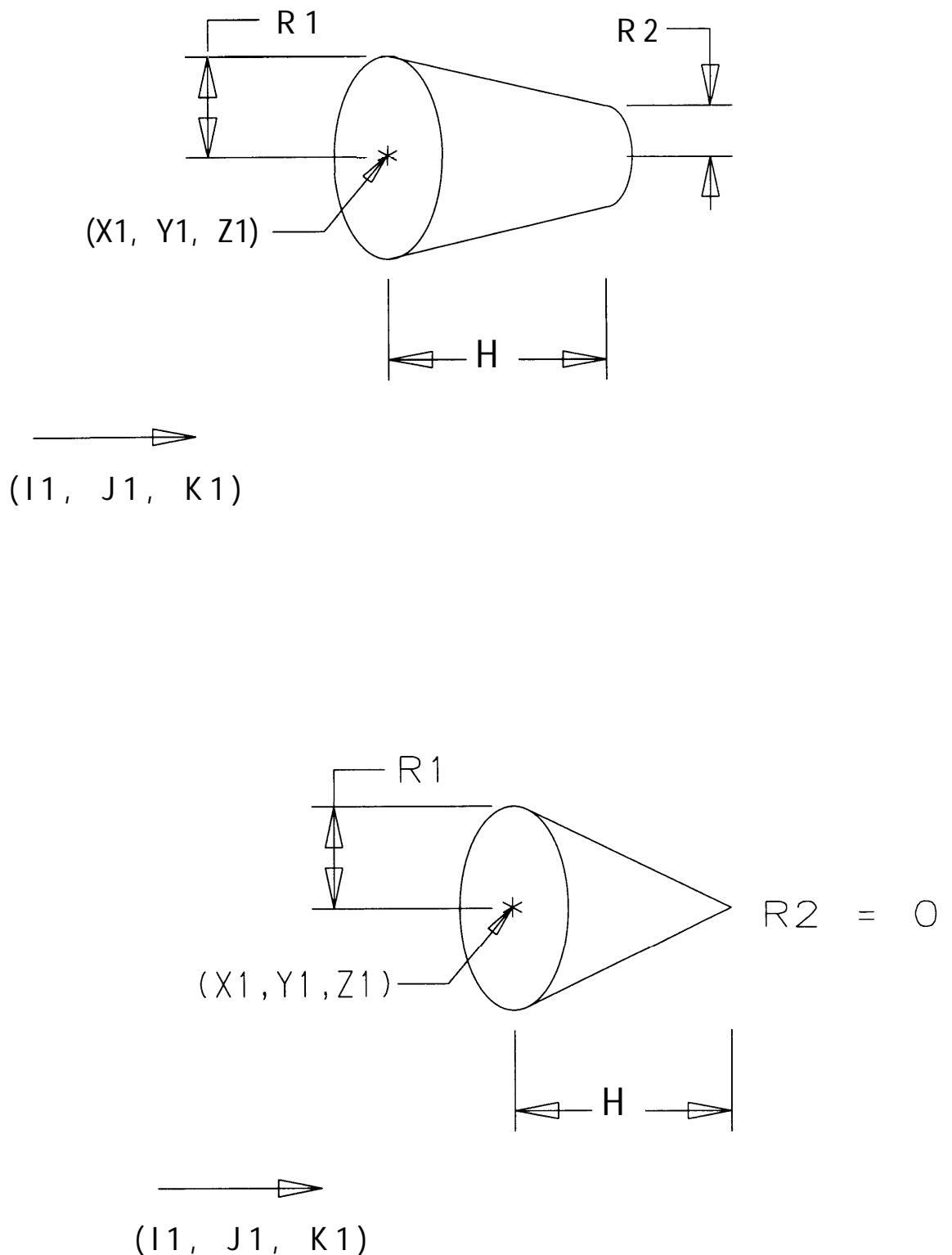


Figure 51. Parameters of the CSG Right Circular Cone Frustum Entity

#### 4.41 Sphere Entity (Type 158)

The sphere is defined with its center coordinates at (X1,Y1,Z1) and a radius R, where R >0.0. ECO630  
**Figure 52** shows an example.

##### Directory Entry

(1) Entity Type Number 158	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????00**	(10) Sequence Number D #
(11) Entity Type Number 158	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

##### Parameter Data

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	R	Real	Radius
2	X1	Real	Center coordinates (default (0.0,0.0,0.0))
3	Y1	Real	
4	Z1	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

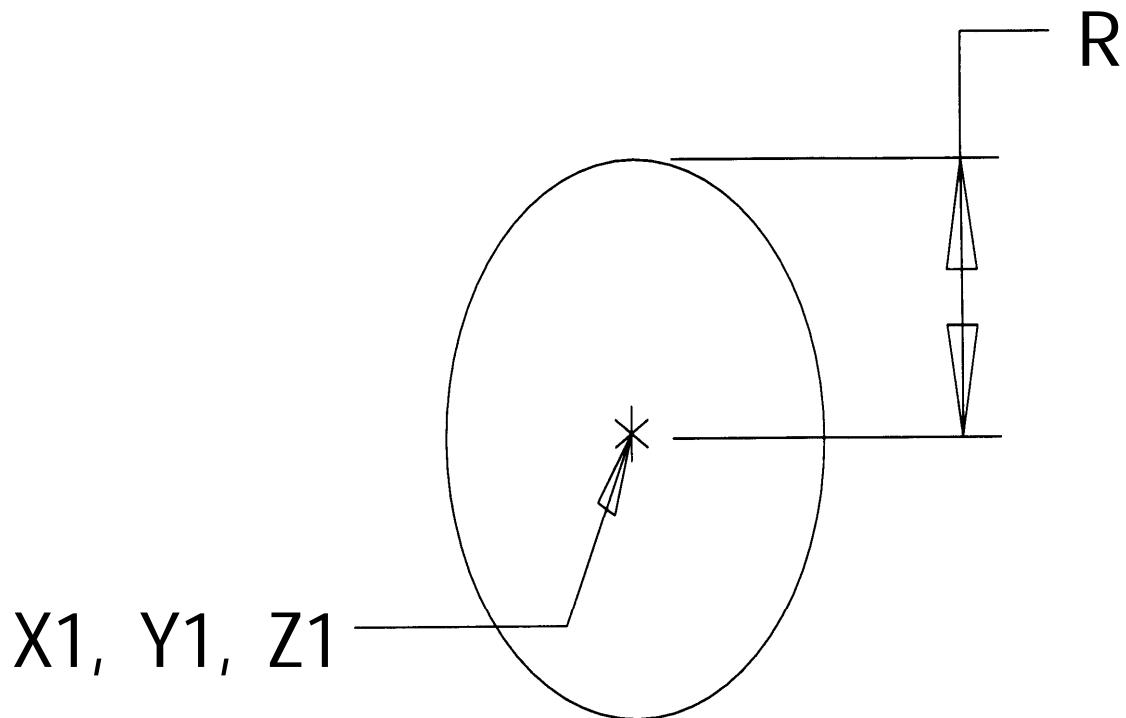


Figure 52. Parameters of the CSG Sphere Entity

#### 4.42 Torus Entity (Type 160)

The torus is the solid formed by revolving a circular disc about a specified coplanar axis. R1 is the ECO630 distance from the axis to the center of the defining disc, and R2 is the radius of the defining disc, where  $R1 > R2 > 0.0$ . The torus is located with its center at (X1,Y1,Z1), and its axis is oriented in the (I1,J1,K1) direction, as shown in [Figure 53](#).

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
160	$\Rightarrow$	$< n.a. >$	$\#, \Rightarrow$	$\#, \Rightarrow$	$0, \Rightarrow$	$0, \Rightarrow$	$0, \Rightarrow$	00000000	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
160	#	$\#, \Rightarrow$	#	0				#	D # + 1

#### Parameter Data

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	R1	Real	Distance from center of torus to center of circular disc to be revolved (perpendicular to axis)
2	R2	Real	Radius of circular disc
3	X1	Real	Torus center coordinates (default (0.0,0.0,0.0))
4	Y1	Real	
5	Z1	Real	
6	I1	Real	Unit vector in axis direction (default (0.0,0.0,1.0))
7	J1	Real	
8	K1	Real	

Additional pointers as required (see [Section 2.2.4.5.2](#)).

#### 4.42 TORUS ENTITY (TYPE 160)

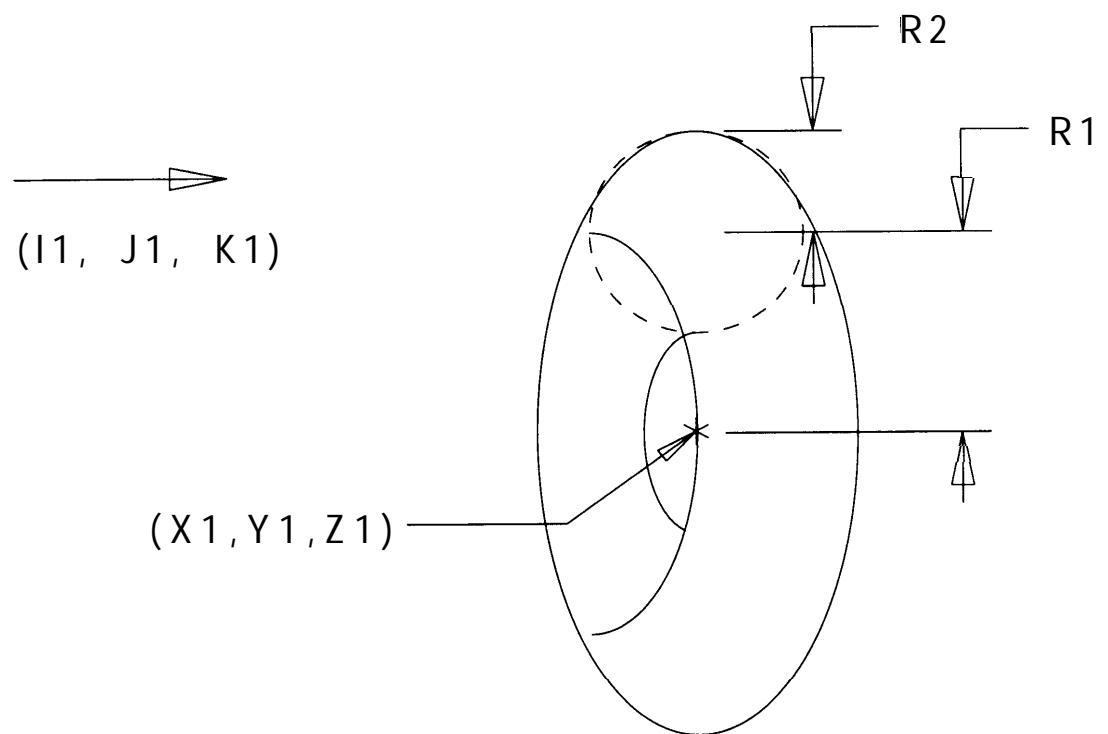


Figure 53. Parameters of the CSG Torus Entity

## 4.43 SOLID OF REVOLUTION ENTITY (TYPE 162)

### 4.43 Solid of Revolution Entity (Type 162)

The Solid of Revolution Entity defines the solid created by revolving the area determined by a ECO630 planar curve about a specified co-planar axis. The revolution is a given fraction of a full rotation  $F (0.0 < F \leq 1.0)$ , using the right-hand rule (counterclockwise when viewed from the positive direction). The curve shall not intersect itself. It shall not cross the axis but may touch it. [Figure 54](#) shows an example.

Two form numbers are used to indicate how the area is determined from the curve. If the curve is ECO630 closed, the form number shall be set to 1, and the area enclosed by the curve is used. If the curve is not closed and the form number is 0 projections are made from the ends of the curve to the rotation axis; the area enclosed by the curve, the projections, and the axis is used. In this case, the curve shall be such that it does not intersect the projections, except at the end points. If the curve is not closed and the form number is 1, the curve is closed by adding a line connecting its end points, and the area enclosed by the curve and the added line is used. In this case, the curve shall not intersect the added line, except at the end points.

For the Solid of Revolution Entity, the Form Numbers are as follows:

ECO630

Form	Meaning
0	Curve closed to axis
1	Curve closed to itself

### Directory Entry

(1) Entity Type Number 162	(2) Parameter Data ⇒	(3) Structure $<n.a.>$	(4) Line Font Pattern $\#, \Rightarrow$	(5) Level $\#, \Rightarrow$	(6) View $0, \Rightarrow$	(7) Xformation Matrix $0, \Rightarrow$	(8) Label Display $0, \Rightarrow$	(9) Status Number ????00**	(10) Sequence Number D #
(11) Entity Type Number 162	(12) Line Weight #	(13) Color Number $\#, \Rightarrow$	(14) Parameter Line Count #	(15) Form Number 0 - 1	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

### Parameter Data

ECO630

Index	Name	Type	Description
1	PTR	Pointer	Pointer to the DE of the curve entity to be revolved. The curve must be coplanar with rotation axis.
2	F	Real	Fraction of full rotation through which the curve entity will be revolved; default 1
3	X1	Real	Coordinates of point on axis (default (0.0,0.0,0.0))
4	Y1	Real	
5	Z1	Real	
6	I 1	Real	Unit vector in axis direction (default (0.0,0.0,1.0))
7	J 1	Real	
8	K 1	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.43 SOLID OF REVOLUTION ENTITY (TYPE 162)**

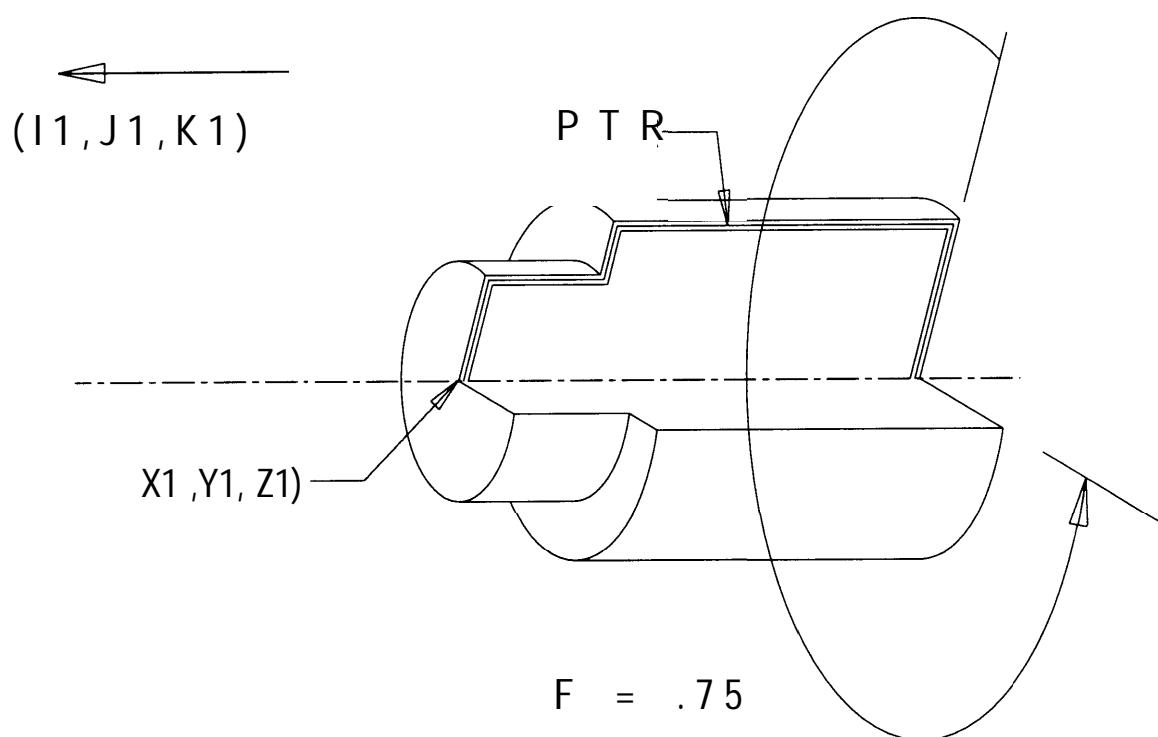


Figure 54. Parameters of the CSG Solid of Revolution Entity

## 4.44 SOLID OF LINEAR EXTRUSION ENTITY (TYPE 164)

### 4.44 Solid of Linear Extrusion Entity (Type 164)

The solid of linear extrusion is defined by translating an area determined by a planar curve. The curve as indicated by PTR in [Figure 55](#) must be closed and nonintersecting. The direction of the translation is defined by a unit vector (I1,J1,K1) and the length of the translation is defined by L, where  $L > 0.0$ . The vector (I1,J1,K1) must not be coplanar with the closed curve.

ECO630

#### Directory Entry

(1) Entity Type Number 164	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????00**	(10) Sequence Number D #
(11) Entity Type Number 164	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	PTR	Pointer	Pointer to the DE of the closed curve entity
2	L	Real	Length of extrusion along the vector positive direction
3	I1	Real	Unit vector specifying direction of extrusion (default (0.0,0.0,1.0))
4	J1	Real	
5	K1	Real	

ECO630

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.44 SOLID OF LINEAR EXTRUSION ENTITY (TYPE 164)**

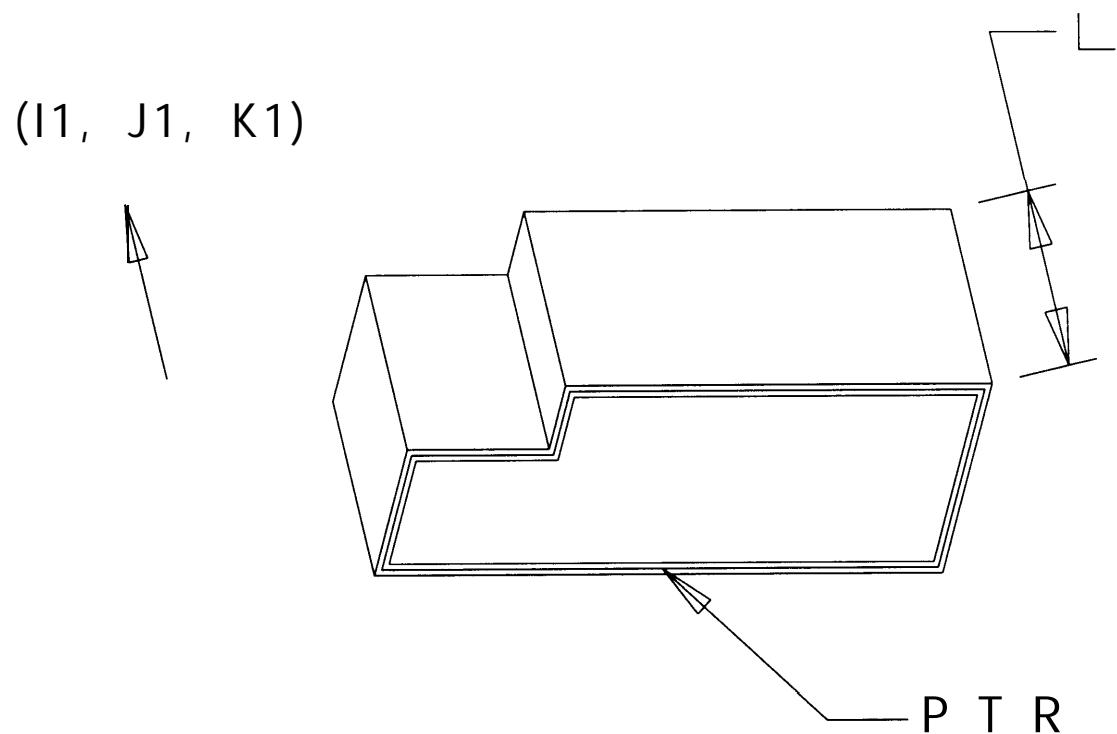


Figure 55. Parameters of the CSG Solid of Linear Extrusion Entity

## 4.45 Ellipsoid Entity (Type 168)

The ellipsoid is a solid bounded by the surface defined by:

$$\frac{X^2}{LX^2} + \frac{Y^2}{LY^2} + \frac{Z^2}{LZ^2} = 1$$

when centered at the origin and aligned with its major axis (LX) in the X direction and with the minor axis (LZ) in the Z direction. A major axis of an ellipsoid can be found by choosing a point on the surface farthest from the center and constructing the line from that point through the center. The plane through the center perpendicular to this major axis intersects the surface of the ellipsoid in an ellipse. The other two axes of the ellipsoid are the axes of this ellipse.

The ellipsoid is defined with its center at (X1,Y1,Z1) and its three axes coincident with the local ECO630 X, Y, Z axes, as shown in [Figure 56](#). The local X-axis is defined by the unit vector (I1,J1,K1) and the local Z-axis by (I2,J2,K2). The local Y-axis is derived by taking the cross product of Z into X. The resulting local system shall be orthogonal, with (I1,J1,K1) values having the highest accuracy precedence. The ellipsoid is specified by positive lengths (LX, LY, and LZ respectively, where LX >= LY >= LZ > 0.0) from the local origin to the surface along the local +X, +Y, +Z axes.

### Directory Entry

(1) Entity Type Number 168	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????00**	(10) Sequence Number D #
(11) Entity Type Number 168	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	LX	Real	Length in the local X-direction
2	LY	Real	Length in the local Y-direction
3	LZ	Real	Length in the local Z-direction
4	X1	Real	Coordinates of point in center of ellipsoid
5	Y1	Real	(default (0.0,0.0,0.0))
6	Z1	Real	
7	I1	Real	Unit vector defining local X-axis (Ellipsoid major axis)
8	J1	Real	(default (1.0,0.0,0.0))
9	K1	Real	
10	I2	Real	Unit vector defining local Z-axis (Ellipsoid minor axis)
11	J2	Real	(default (0.0,0.0,1.0))
12	K2	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

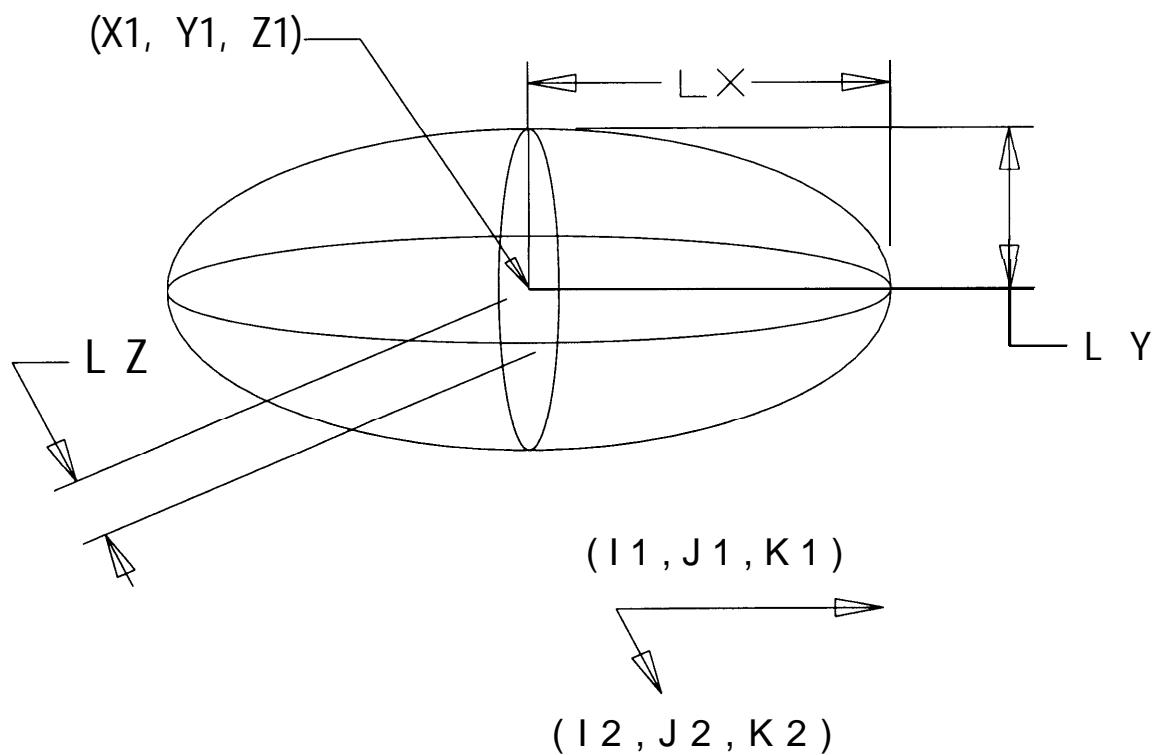


Figure 56. Parameters of the CSG Ellipsoid Entity

**4.46 Boolean Tree Entity (Type 180)**

The Boolean tree describes a binary tree structure composed of regularized Boolean operations and operands, in postorder notation. A regularized Boolean operation is defined as the closure of the interior of the result of a Boolean set operation. Specifically, denote the interior of a set  $X$  by  $X_o$ , the closure of  $X$  by  $\bar{X}$  and use  $\cup^*$ ,  $\cap^*$ , and  $-^*$  to denote the regularized Boolean operations intersection, and difference, respectively. Then:

$$\begin{aligned} X \cup^* Y &= \overline{(X \cup Y)_o} \\ X \cap^* Y &= \overline{(X \cap Y)_o} \\ X -^* Y &= \overline{(X - Y)_o} \end{aligned}$$

Since the topological space under consideration is a 3-dimensional space, all lower dimensional entities resulting from these operations will disappear. A discussion of regularized Boolean operations can be found in [TIL080].

All operations are assigned integers as follows:

<b>Integer</b>	<b>Operation</b>
1	Union
2	Intersection
3	Difference

Allowable operands are:

- Primitive entities
- Boolean Tree Entities
- Solid Instance Entities
- Manifold Solid B-Rep Object Entities

ECO644

The parameter data entries for the Boolean Tree Entity can be operation codes (integers) or pointers to operands. A positive (or unsigned) value in a parameter data entry implies an operation code; a negative value implies the absolute value is to be taken as a pointer to an operand.

A transformation matrix may be pointed to by Field 7 of the DE to position the resulting solid in any desired manner

For the Boolean Tree Entity, the Form Numbers are as follows:

ECO630

<b>Form</b>	<b>Meaning</b>
0	All operands are primitives, solid instances, or other Boolean trees
1	At least one operand is a manifold solid B-Rep object entity

## 4.46 BOOLEAN TREE ENTITY (TYPE 180)

[Figure 57](#) shows an example of a Boolean tree composed of five operands and four operations with values as follows:

Parameter	Value
1	9
2	PTRA (negative)
3	PTRB (negative)
4	PTRC (negative)
5	1
6	3
7	PTRD (negative)
8	PTRE (negative)
9	2
10	1

### Directory Entry

(1) Entity Type Number 180	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????00??	(10) Sequence Number D #
(11) Entity Type Number 180	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

**Note:** When the Hierarchy is set to Global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, Color Number, Level, View, and Blank Status.

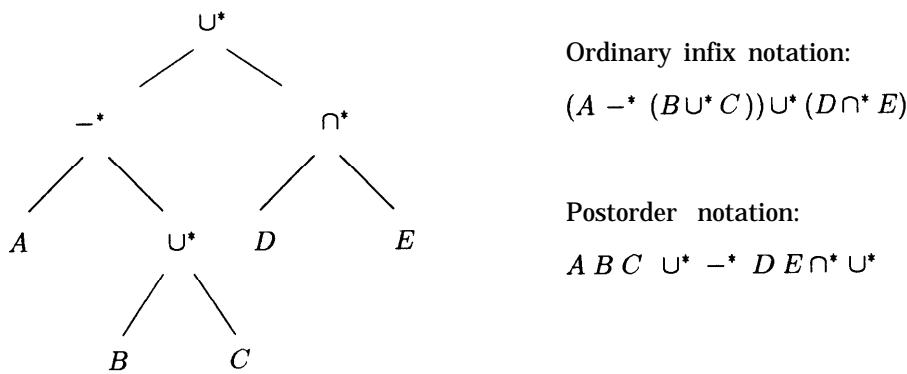
### Parameter Data

Index	Name	Type	Description
1	N	Integer	Length of post-order notation, including operations and operands ( $N > 2$ )
2	PTR(1)	Pointer	Negated pointer to the DE of the first operand
3	PTR(2)	Pointer	Negated pointer to the DE of the second operand
4	PTR(3)	Pointer	Negated pointer to the DE of the third operand
	or	or	or
	IOP(1)	Integer	Integer for the first operation
:	:	:	
N	PTR(M)	Pointer	Negated pointer to the DE of the last operand
	or	or	
	IOP(L-1)	Integer	Integer for next-to-last operation
N+1	IOP(L)	Integer	Integer for last operation

Additional pointers as required (see [Section 2.2.4.5.2](#)).

Notes: Parameters 2 and 3 will always be operands and thus will be negative numbers.

As L is the number of operations, and M is the number of operands,  $N = L+M$ .



Parameters: 9 A B C 13 D E 21

(A, B, C, D,& E are negative values representing pointers to operands.)

Figure 57. Example of a Boolean Tree

## 4.47 Selected Component Entity (Type 182) ‡

EC0630

‡The Selected Component Entity has not been tested. See Section 1.9.

The Selected Component Entity provides a means of selecting one component of a disjoint CSG solid.

**Directory Entry**

(1) Entity Type Number 182	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number **??03**	(10) Sequence Number D #
(11) Entity Type Number 182	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Parameter Data**

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	BTREE	Pointer	Pointer to the DE of the Boolean Tree Entity
2	SELX	Real	X component of a point in or on the desired component
3	SELY	Real	Y component of a point in or on the desired component
4	SELZ	Real	Z component of a point in or on the desired component

Additional pointers as required (see Section 2.2.4.5.2).

#### 4.48 Solid Assembly Entity (Type 184)

A solid assembly is a collection of items which possess a shared fixed geometric relationship. It differs from a union of the items in that each item retains its own structure, even if the items touch.

The transformation matrices are applied to the items individually before a matrix referenced by ECO630 Field 7 of the DE is applied to the collection. A value of zero in the pointer field indicates the identity matrix.

For the Solid Assembly Entity, the Form Numbers are as follows:

ECO644

Form	Meaning
0	All items are primitives, solid instances, Boolean trees, or other assemblies
1	At least one item is a manifold solid B-Rep object entity

#### Directory Entry

(1) Entity Type Number 184	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????02??	(10) Sequence Number D #
(11) Entity Type Number 184	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

**Note:** When the Hierarchy is set to Global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, Color Number, Level, View, and Blank Status.

#### Parameter Data

ECO650

Index	Name	Type	Description
1	N	Integer	Number of items
2	PTR(1)	Pointer	Pointer to the DE of the first item
⋮	⋮	⋮	⋮
1+N	PTR(N)	Pointer	Pointer to the DE of the last item
2+N	PTRM(1)	Pointer	Pointer to the DE of the Transformation Matrix Entity for the first item
⋮	⋮	⋮	⋮
1+2*N	PTRM(N)	Pointer	Pointer to the DE of the Transformation Matrix Entity for the last item

Additional pointers as required (see Section 2.2.4.5.2).

## 4.49 Manifold Solid B-Rep Object Entity (Type 186) ‡

ECO630

‡The Manifold Solid B-Rep Object Entity has not been tested. See Section 1.9.

A manifold solid is a bounded, closed, and finite volume  $V$  in three dimensional Euclidean space,  $R^3$ .  $V$  is restricted to be the closure of the interior of  $V$  which shall be arcwise connected. There is no restriction on the number of voids within  $V$  or on the genus of the boundary surfaces. Discussion of the manifold solid from a graph theoretic view is contained in [Appendix I](#).

The Manifold Solid B-Rep Object (MSBO) defines a manifold solid by enumerating its boundary. This boundary may be decomposed into its maximal connected components called closed shells. Each shell is composed of faces which have underlying surface geometry. The faces are bounded by loops of edges having underlying curve geometry. The edges are bounded by vertices whose underlying geometry is the point. Implicit in the representation is a concept of oriented uses of topological entities by containing entities. This allows the referencing entity to reverse the natural orientation of the referenced entity. The natural orientation is derived from the underlying geometry. [Figure 58](#) illustrates the hierarchical nature of this representation.

The vertex represents a location. The geometry underlying a vertex is a point in  $R^3$ .

An edge connects two vertices. It is bounded by two vertices ( $V_s$  and  $V_t$ ). It does not contain its bounds. The start and terminate vertices do not have to be distinct. Edges do not intersect except at their boundaries (i. e., vertices). The geometry underlying an edge is some portion of a curve in  $R^3$ . The edge has a natural orientation in the same direction as its underlying curve in  $R^3$ . Thus the edge is traced from start vertex to terminate vertex as the underlying curve is traced in the direction of increasing parameter value. Each edge is used once in each orientation and therefore shall be referenced exactly twice in an MSBO.

The loop is a path of oriented edges and vertices having the same start and terminate vertex. Typically, a loop represents a connected collection of face boundaries, seams, and poles of a single face (refer to Figures in [Appendix I](#)). Its underlying geometry is a connected curve or a single point in  $R^3$ . The loop is represented as an ordered list of oriented edges, edge-uses ( $EU_i$ ,  $i = 1, n$ ), which has the following properties:

- The terminal vertex of  $EU_1$  is the initial vertex of  $EU_{i+1}$ ,  $i = 1, n - 1$ .
- The loop is closed. This implies that the terminal vertex of  $EU_n$  is the same as the initial vertex of  $EU_1$ .
- The orientation of the loop is defined to be the same as its constituent edge-uses which reference edges. Therefore the direction of the loop at an edge-use which references a vertex,  $A$ , can be taken from any edge-use having an underlying edge which has  $A$  as either its start or terminate vertex.

The edge-use is an instancing of an edge or vertex into a loop. It consists of either an edge, an orientation, and optional parameter space curves (see the definitions of associated parameter space and collections in the Boundary Entity (Type 141)), or (in the case of a pole) a vertex and an optional parameter space curve.

If the edge-use references an edge, then the orientation describes whether the direction of this use of the edge is in agreement with the natural orientation of the edge. If the orientation of the edge-use is in agreement with the edge, then the use is directed from the start vertex to the terminate vertex of the edge. If the orientation is not in agreement, then the use of the edge is directed from the terminate vertex to the start vertex. At any point the direction of an edge-use is called its topological

#### 4.49 MANIFOLD SOLID B-REP OBJECT ENTITY (TYPE 186) ‡

tangent vector,  $T$ . See the face discussion to determine how to set the orientation. If the edge-use references a vertex, then no orientation is defined.

The face is a bound (partial) of an arcwise connected open subset of  $R^3$  and has finite area. It has an underlying surface,  $S$ , and is bounded by at least one loop. If more than one loop bounds a face, then the loops shall be disjoint. The cross product,  $N \times T$ , where  $N$  is in the same direction as the normal to  $S$  and  $T$  is the topological tangent vector of an edge-use in a loop bounding the face, points toward the material of the face. Note that this determines the edge-use orientation.

The MSBO shall point to one or more closed shells. The closed shell is represented as a set of edge connected oriented uses of faces (face-uses). The closed shell divides  $R^3$  into two arcwise-connected open subsets (parts). The normal of the shell is in the same direction as the normal of its face-uses. The normal of each face-use of the closed shell points toward the same part of  $R^3$ . The normal of the face-use is assumed to be in the direction of the normal of the underlying surface of the face unless the face-use orientation indicates it needs to be reversed. The faces used by the shell are connected to each other only via edges. Each edge shall be used exactly twice, once in each orientation, in the closed shell.

The MSBO describes the boundaries of the solid via oriented uses of shells (shell-use). It is the orientation of the use of the shells which define the volume of  $R^3$  the MSBO is describing. The orientation of the shell-use is determined by the shell-use normal which is either in the same or opposite direction as the shell normal. By convention, the direction of the shell-use normal points away from the part of  $R^3$  being described. One shell, the outer, shall completely enclose all the other shells and only the outer shell shall enclose a shell.

The geometric entities that may be used in an MSBO consist of the point, curve, and surface. The point data is embedded in the Vertex Entity for reasons of data compaction. The entities that may be used for a curve are restricted to the subset identified for Form 1 of the Edge Entity. The subset of surface entities that may be used is identified in Form 1 of the Face Entity. To avoid processing difficulties, the use of nested constructs is discouraged. For example, allowing the Edge to point at a Composite Curve which uses an Offset Curve as one of its components is not recommended.

The geometric surface definition used to specify the geometry of a face shall be a 2-manifold which is arcwise connected, oriented, bounded, non-self-intersecting, and has no handles within the region underlying the face. The surfaces can be represented implicitly,  $F(x, y, z) = 0$  or parametrically,  $S(u, v)$ . In the implicit representation the direction of the surface normal (orientation) is defined by the gradient of  $F(x, y, z)$ . If the surface is represented parametrically, the surface normal (orientation) is given by the cross product of the partial derivatives (in the order stated) with respect to  $u$  and  $v$ .

The model space ( $R^3$ ) curves underlying the edges are assumed to be parametrically represented, have a unique non-zero tangent vector at each point, lie on the two (2) intersecting surfaces, and be non-self intersecting on the open segment underlying the edge.

Note that, due to seams and poles, the representation of the pre-image of the curve,  $C$ , in the parameter space of the surfaces,  $S_1$  and  $S_2$ , can consist of ordered lists of curves,  $C_{ii}^j$ ,  $i = 1, n$  for surface  $S_1$  and  $C_{jj}^i$ ,  $j = 1, m$  for surface  $S_2$ . The  $C_{ii}^j$  given by the composition  $(S_1 \circ C_{ii}^j, i = 1, n)$  and the  $C_{jj}^i$  given by the composition  $(S_2 \circ C_{jj}^i, j = 1, m)$  form composite curves in  $R^3$  which are coincident with the curve  $C$ .

The optional parameter space curves,  $C_i^j$ ,  $i = 1, n$ , referenced by an edge-use are in the parameter space defined by the surface underlying the face bounded by the loop containing the referencing edge-use. These curves are assumed to be ordered in the list and oriented such that as the parameter goes from its initial to its final value for each parameter space curve the composition  $(S \circ C_i^j, i = 1, n)$

#### 4.49 MANIFOLD SOLID B-REP OBJECT ENTITY (TYPE 186) ‡

produces a composite curve,  $C_i$ ,  $i = 1, n$ , which is coincident with the curve underlying the edge. The orientation of  $C_i$ ,  $i = 1, n$  is in agreement with the orientation of the edge-use.

See Appendix I for examples that illustrate the general model for any entity modeling of a Cylinder, Sphere, and Torus.

The following is a summary of the major constraints on the topological and geometrical entities that may be used in representing the MSBO:

- The MSBO shall contain exactly one outer shell
- The volume described by the MSBO shall be arcwise connected. This implies that voids inside the outer shell shall not be contained in another void.
- The shells of an object shall be disjoint.
- The direction of the normals of the face-uses of a shell, reversed if the shell orientation flag is false, shall point away from the portion of  $R3$  that is in the volume being communicated by the MSBO.
- The shells of an object shall be closed shells.
- The face interiors, edge interiors, and vertices shall not intersect.
- Only the MSBO and the  $R3$  curve and surface entities shall have a transform.

ECO627

The following topological entities may be used in representing the MSBO:

**Manifold Solid B-Rep Object (MSBO) Entity (Type 186, Form 0)** Identifies the shell-uses (shell + orientation) which make up the MSBO.

**Closed Shell Entity (Type 514, Form 1)** defines a boundary for a region of  $R3$  by identifying ECO627 and orienting the use of faces.

**Face Entity (Type 510, Form 1)** implements the topological concept of a portion of a boundary of  $R3$ . The underlying surface is required.

**Loop Entity (Type 508, Form 1)** identifies and orients the use of edges as bounds (partial) of faces. It also establishes the optional association of parameter space geometry.

**Edge List Entity (Type 504, Form 1)** models an edge or a list of edges. Each edge referenced in an MSBO shall be modeled in only one Edge List Entity. Thus all references to a specific edge shall use the same Edge List Entity and list index. The underlying curve geometry in  $R3$  is required.

**Vertex List Entity (Type 502, Form 1)** models a vertex or a list of vertices. Each vertex referenced in an MSBO shall be modeled in only one Vertex List Entity. Thus all references to a specific vertex shall use the same Vertex List Entity and list index.

Figure 58 illustrates the hierarchical nature of a MSBO. Figure 59 illustrates the construction of a MSBO.

## 4.49 MANIFOLD SOLID B-REP OBJECT ENTITY (TYPE 186) ‡

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
186	$\Rightarrow$	$< n.a. >$	$< n.a. >$	$\#, \Rightarrow$	$< n.a. >$	$0, \Rightarrow$	$0, \Rightarrow$	?????????	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
186	$< n.a. >$	$< n.a. >$	#	0				#	D # + 1

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>	
1	SHELL	Pointer	Pointer to the DE of the shell	
2	SOF	Logical	Orientation flag of shell with respect to its underlying faces (True = agrees)	ECO627
3	N	Integer	Number of void shells, or zero	
4	VOID(1)	Pointer	Pointer to the DE of the first void shell	
5	VOF(1)	Logical	Orientation flag of first void shell	
:	:	:		
2+2*N	VOID(N)	Pointer	Pointer to the DE of the last void shell	
3+2*N	VOF(N)	Logical	Orientation flag of last void shell	

Additional pointers as required (see Section 2.2.4.5.2).

**4.49 MANIFOLD SOLID B-REP OBJECT ENTITY (TYPE 186) ‡**

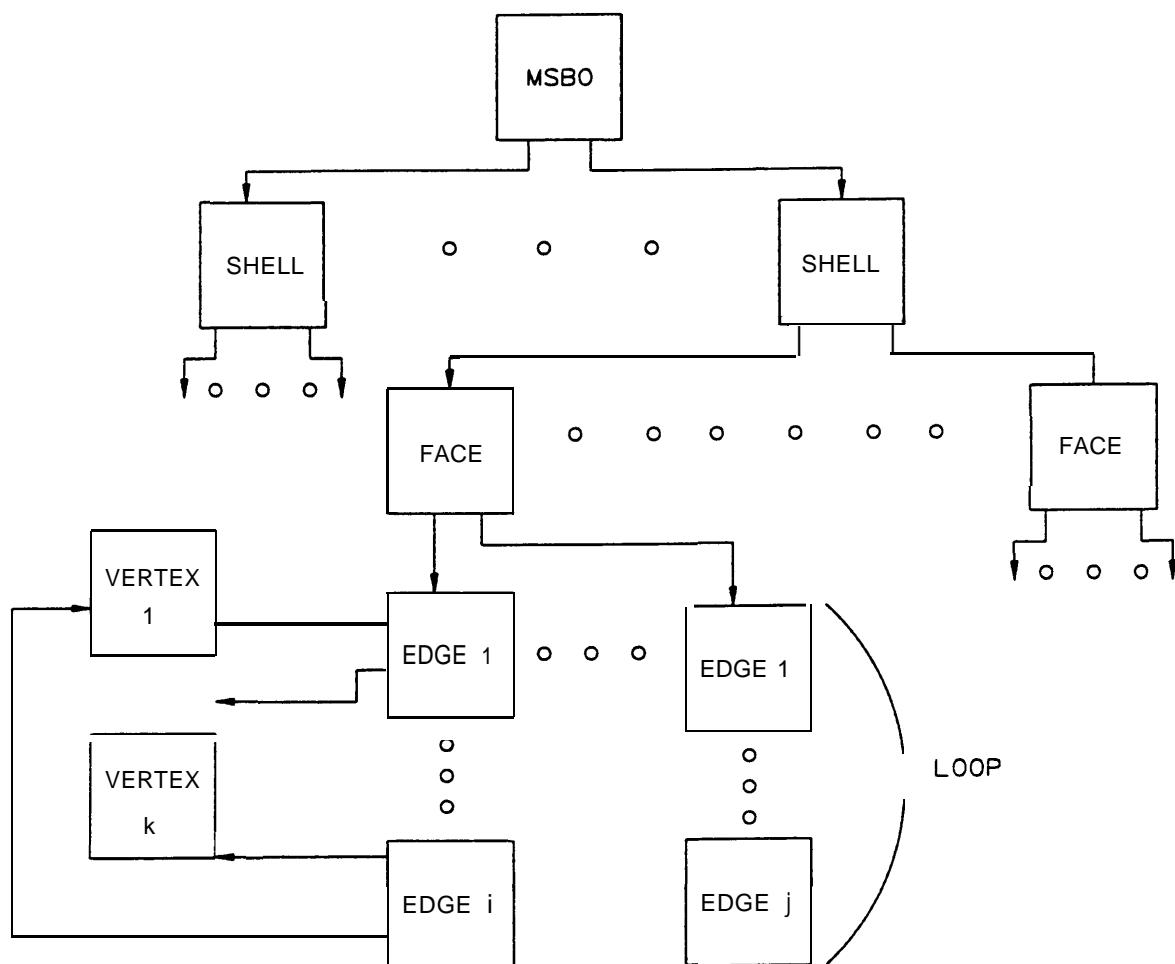


Figure 58. Hierarchical nature of the MSBO

**4.49 MANIFOLD SOLID B-REP OBJECT ENTITY (TYPE 186) ‡**

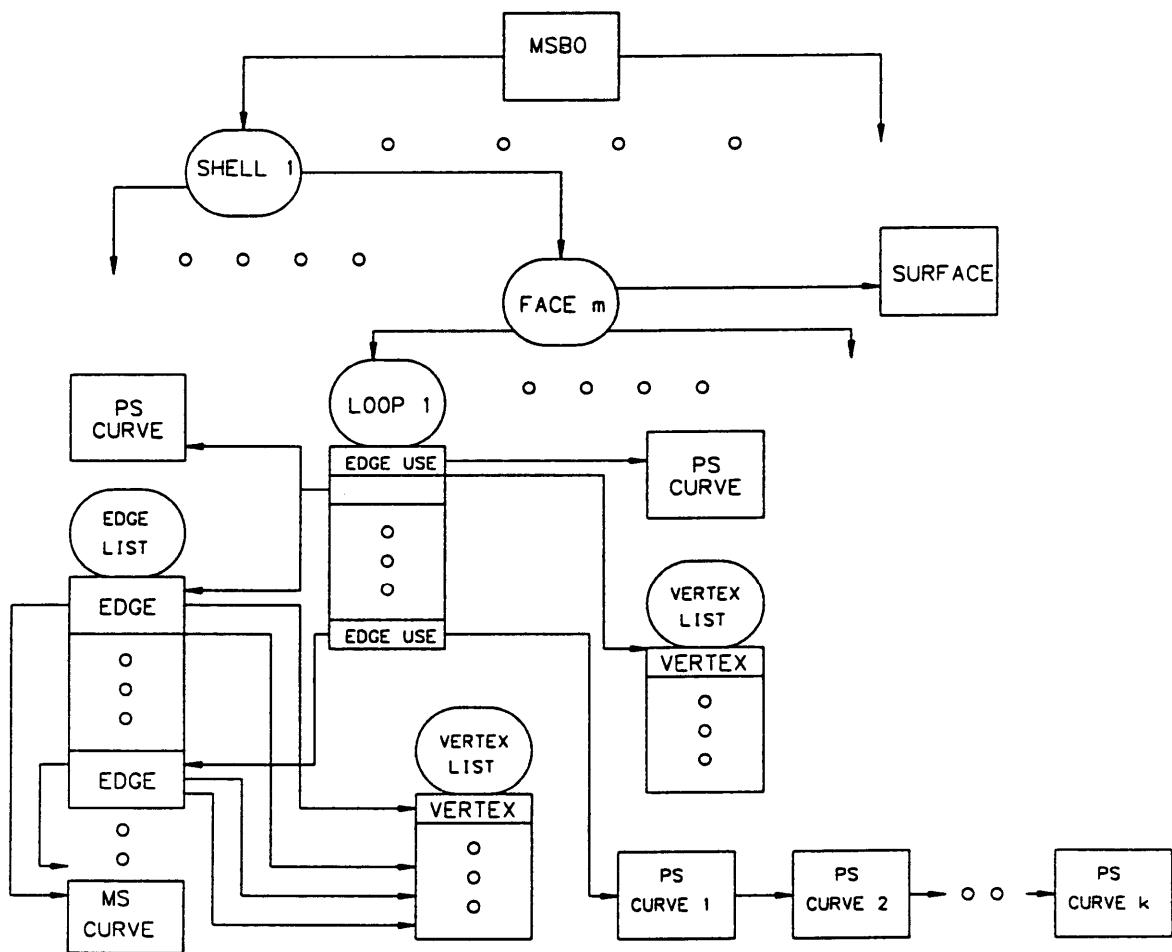


Figure 59. Construction of the MSBO

## 4.50 Plane Surface Entity (Type 190) ‡

ECO630

‡The Plane Surface Entity Entity has not been tested. [See Section 1.9.](#)

The plane surface is defined by a point on the plane and the normal direction to the surface. ([See Figure 60.](#))

If **C** is the point and **z** is the unitized normal direction, the plane surface is defined as the collection of all points **r** in Euclidean 3-space satisfying the equation

$$\mathbf{r} \cdot \mathbf{z} - \mathbf{C} \cdot \mathbf{z} = 0$$

The data ([Figure 61](#)) for the parameterized surface form is to be interpreted as follows:

<b>C</b>	= LOCATION
<b>z</b>	= <NORMAL>
<b>d</b>	= <REFDIR>
<b>x</b>	= < <b>d</b> - ( <b>d</b> • <b>z</b> ) <b>z</b> >
<b>y</b>	= < <b>z</b> × <b>x</b> >,

and the surface is parameterized as

$$\sigma(u, v) = \mathbf{C} + u \mathbf{x} + v \mathbf{y},$$

where the parameterization range is  $-\infty < u, v < \infty$ .

Note that **d** shall be distinct from **z** and shall be approximately perpendicular to **z**.

ECO630

For the Plane Surface Entity, the Form Numbers are as follows:

Form	Meaning
0	Unparameterized surface
1	Parameterized surface

The plane surface type is unbounded unless it is subordinate to another entity, such as the Bounded Surface Entity ([Type 143](#)) or the Trimmed Parametric Surface Entity ([Type 144](#)), that references its bounding geometry. If the Subordinate Entity Switch for this entity is set to Independent, the plane is infinite in extent.

This entity shall not be used as a clipping plane for a View Entity ([Type 410](#)).

ECO630

**Directory Entry**

(1) Entity Type Number 190	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number : *?*?*?*	(10) Sequence Number D #
(11) Entity Type Number 190	(12) Line Weight #	(13) Color Number	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

**Un-parameterized Plane Surface Entity (Type 190, Form 0)**

**Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DELOC	Pointer	Pointer to the DE of the point on the surface (LOCATION)
2	DENRML	Pointer	Pointer to the DE of the surface normal direction (NORMAL)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**Parameterized Plane Surface Entity (Type 190, Form 1)**

**Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DELOC	Pointer	Pointer to the DE of the point on the surface (LOCATION)
2	DENRML	Pointer	Pointer to the DE of the surface normal direction (NORMAL)
3	DEREFID	Pointer	Pointer to the DE of the reference direction (REFDIR)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

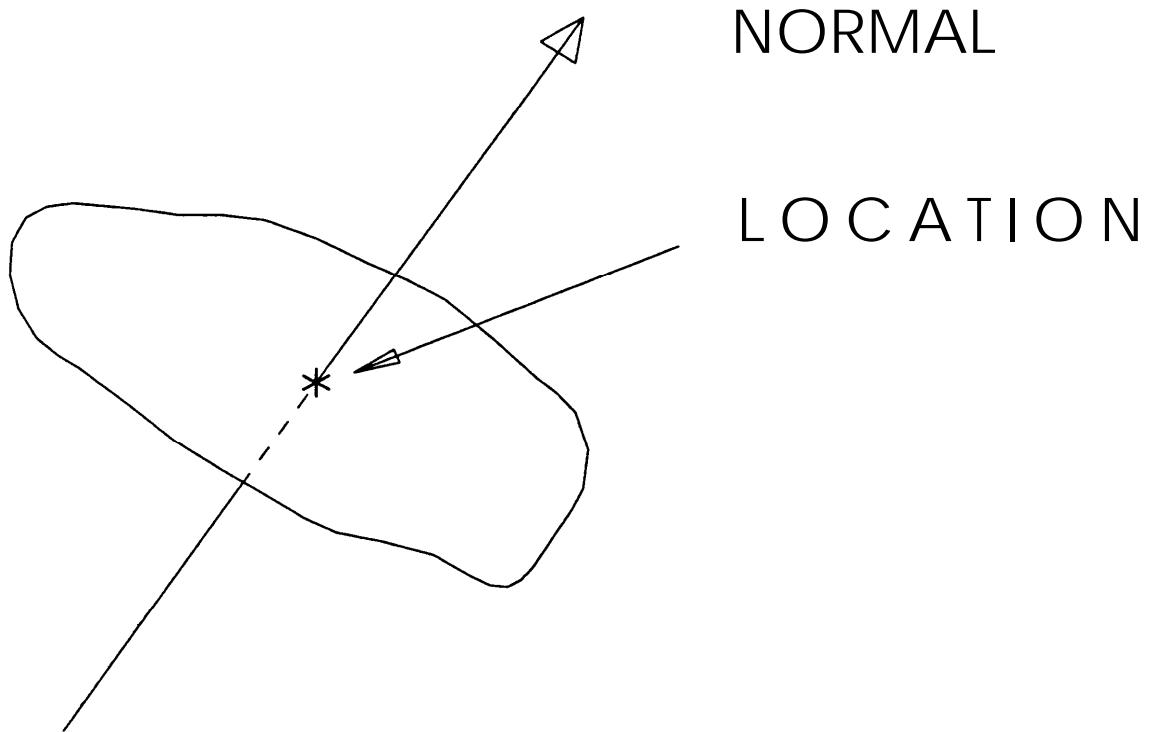


Figure 60. Defining data for un-parameterized plane surface (Form Number = 0)

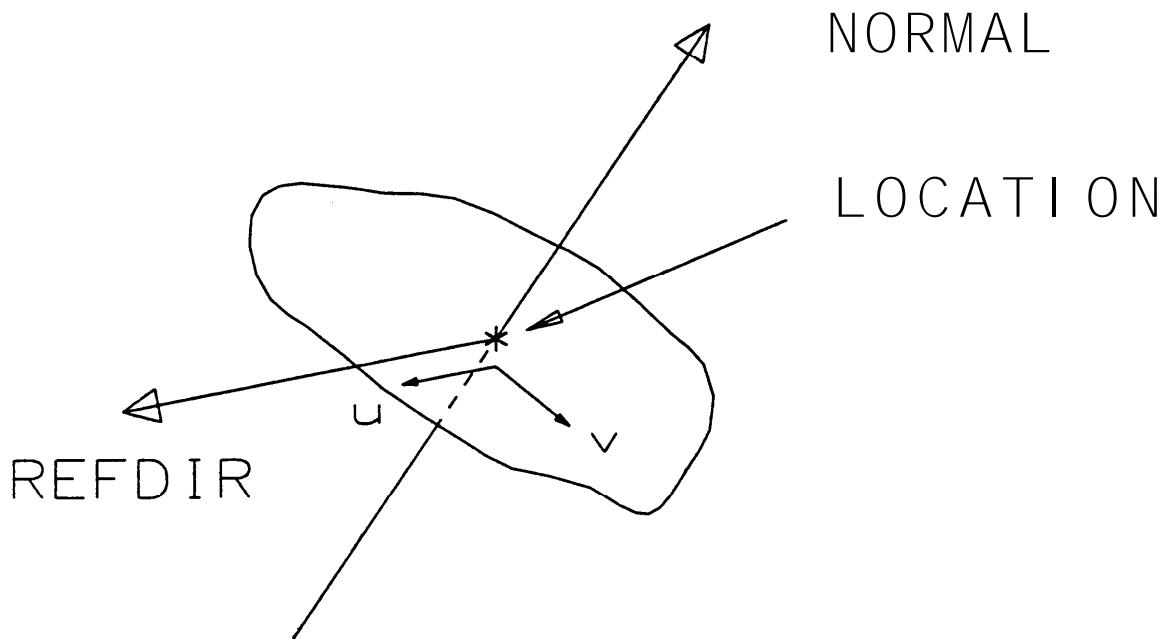


Figure 61. Defining data for parameterized plane surface (Form Number = 1)

## 4.51 RIGHT CIRCULAR CYLINDRICAL SURFACE ENTITY (TYPE 192)‡

### 4.51 Right Circular Cylindrical Surface Entity (Type 192)‡

ECO630

‡The Right Circular Cylindrical Surface Entity has not been tested. See Section 1.9.

The right circular cylindrical surface is defined by a point on the axis of the cylinder, the direction of the axis of the cylinder and a radius. (See Figure 62.) The positive direction of the surface normal is outwards from the axis.

If a local coordinate system is defined with the origin at the axis point and the Z axis in the axis direction, then the equation of the surface in this system is  $S = 0$  where

$$S(x, y, z) = x^2 + y^2 - r^2$$

and the positive direction of the surface normal is in the direction of increasing  $S$ . That is, the normal,  $\mathbf{N}$ , to the surface at any point on the surface is given by

$$\mathbf{N} = (S_x, S_y, S_z)$$

The data for the parameterized form of the surface (Figure 63) is to be interpreted as follows:

**C** = LOCATION

**z** = <AXIS>

**d** = <REFDIR>

**x** = <**d** - (**d** • **z**)**z**>

**y** = <**z** × **x**>

**r** = RADIUS

and the surface is parameterized as

$$\sigma(u, v) = \mathbf{C} + r(\cos(u)\mathbf{x} + \sin(u)\mathbf{y}) + v\mathbf{z}$$

where the parameterization range is  $0 \leq u \leq 360$  degrees and  $-\infty < v < \infty$ .

Note that **d** shall be distinct from **z** and shall be approximately perpendicular to **z**.

For the Right Circular Cylindrical Surface Entity, the Form Numbers are as follows:

Form	Meaning
0	Unparametrized Surface
1	Parameterized Surface

This surface type is intended to represent the geometry underlying topology, and shall only be referenced by a Face Entity (Type 510, Form 1). The Subordinate Entity Switch shall always be set to Physically Dependent; i.e., independent instances of this entity are not permitted.

## 4.51 RIGHT CIRCULAR CYLINDRICAL SURFACE ENTITY (TYPE 192)‡

### Directory Entry

(1) Entity Type Number 192	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number **01??**	(10) Sequence Number D #
(11) Entity Type Number 192	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0-1	(16) R e s e r v e d	(17) R e s e r v e d	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

### Un-parameterized Right Circular Cylindrical Surface Entity (Type 192, Form 0)

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DELOC	Pointer	Pointer to the DE of the point on axis (LOCATION)
2	DEAXIS	Pointer	Pointer to the DE of the axis direction (AXIS)
3	RADIUS	Real	Value of radius (> 0.0)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

### Parameterized Right Circular Cylindrical Surface Entity (Type 192, Form 1)

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DELOC	Pointer	Pointer to the DE of the point on axis (LOCATION)
2	DEAXIS	Pointer	Pointer to the DE of the axis direction (AXIS)
3	RADIUS	Real	Value of radius (> 0.0)
4	DREFD	Pointer	Pointer to the DE of the reference direction (REFDIR)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**RIGHT CIRCULAR CYLINDRICAL SURFACE ENTITY (TYPE 192)‡**

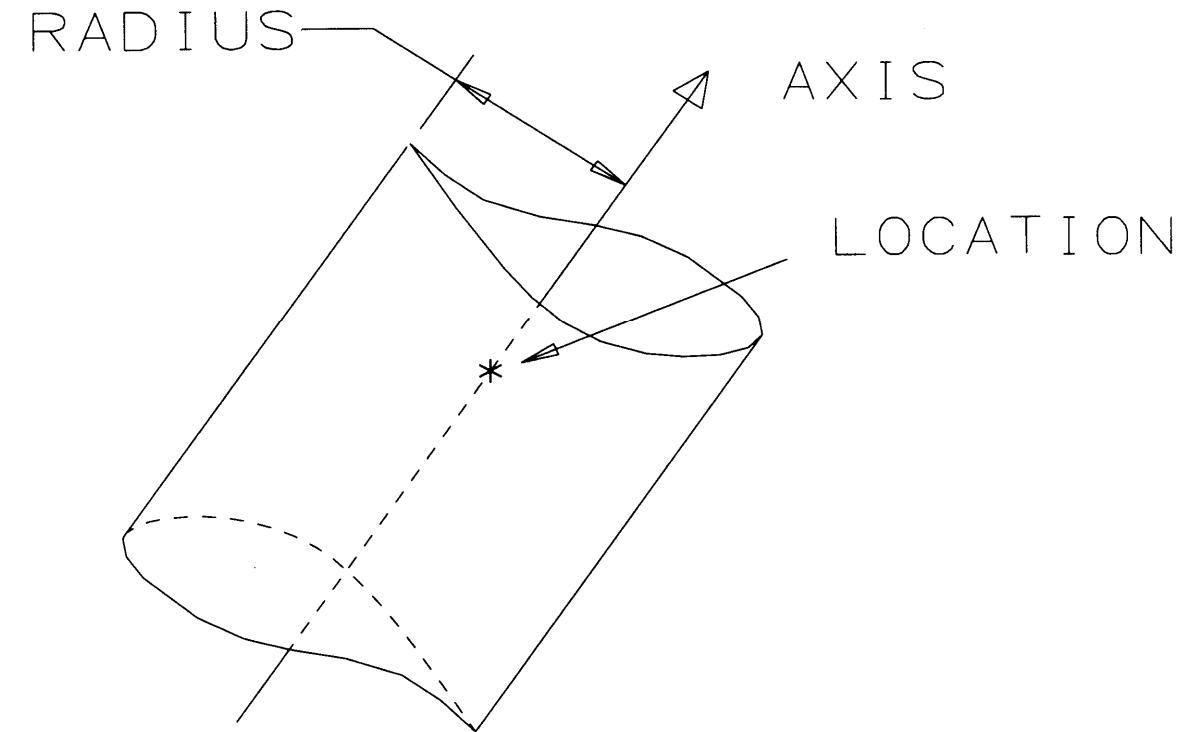


Figure 62. Defining data for un-parameterized right circular cylindrical surface (Form Number = 0)

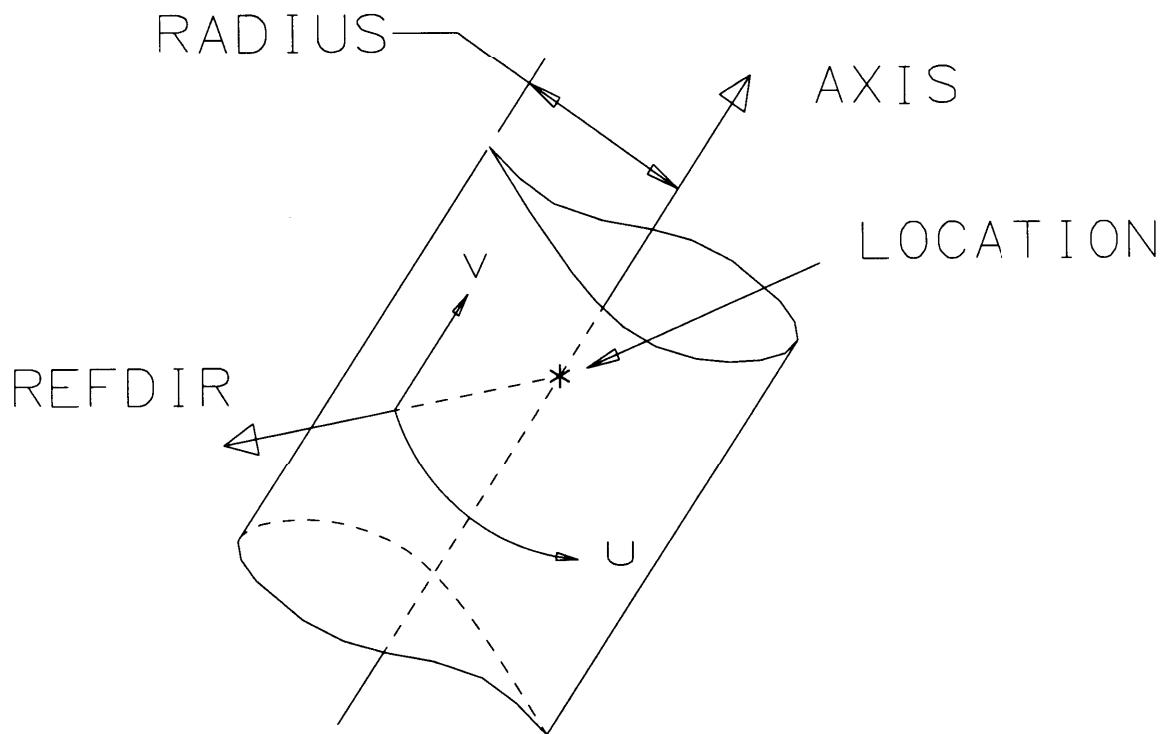


Figure 63. Defining data for parameterized right circular cylindrical surface (Form Number = 1)

## 4.52 RIGHT CIRCULAR CONICAL SURFACE ENTITY (TYPE 194)‡

### 4.52 Right Circular Conical Surface Entity (Type 194)‡

ECO630

‡The Right Circular Conical Surface Entity has not been tested. See Section 1.9.

The right circular conical surface is defined by a point on the axis of the cone, the direction of the axis of the cone, the radius of the cone at the axis point and the cone semi-angle. Figures 64 and 65 show examples. The positive direction of the surface normal is outwards from the axis.

If a local coordinate system is defined with the origin at the axis point and the Z axis in the axis direction, the equation of the surface in this system is  $S = 0$  where

$$S(x, y, z) = x^2 + y^2 - (r + z \tan s)^2$$

where  $s$  is the cone semi-angle and  $r$  is the given cone radius. The positive direction of the surface normal is in the direction of increasing  $S$ . At any point on the surface the surface normal  $\mathbf{N}$  is

$$\mathbf{N} = (S_x, S_y, S_z)$$

The data for the parameterized form of the surface (Figure 65) is to be interpreted as follows:

<b>C</b>	= LOCATION
<b>z</b>	= <AXIS>
<b>d</b>	= <REFDIR>
<b>x</b>	= < <b>d</b> - ( <b>d.z</b> ) <b>z</b> >
<b>y</b>	= < <b>z</b> × <b>x</b> >
<b>r</b>	= RADIUS
<b>s</b>	= ANGLE

and the surface is parameterized as

$$\sigma(u, v) = \mathbf{C} + (r + v \tan(s)) (\cos(u) \mathbf{x} + \sin(u) \mathbf{y}) + v \mathbf{z}$$

where the parameterization range is  $0 \leq u \leq 360$  degrees and  $-\infty < v < \infty$ .

Note that **d** shall be distinct from **z** and shall be approximately perpendicular to **z**.

For the Right Circular Conical Surface Entity, the Form Numbers are as follows:

Form	Meaning
0	Unparameterized surface
1	Parameterized surface

This surface type is intended to represent the geometry underlying topology, and shall only be referenced by a Face Entity (Type 510, Form 1). The Subordinate Entity Switch shall always be set to Physically Dependent; i.e., independent instances of this entity are not permitted.

## 4.52 RIGHT CIRCULAR CONICAL SURFACE ENTITY (TYPE 194)‡

### Directory Entry

(1) Entity Type Number 194	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number **01??**	(10) Sequence Number D #
(11) Entity Type Number 194	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # 1

ECO630

### Un-parameterized Right Circular Conical Surface Entity (Type 194, Form 0)

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DELOC	Pointer	Pointer to the DE of the point on axis (LOCATION)
2	DEAXIS	Pointer	Pointer to the DE of the axis direction (AXIS)
3	RADIUS	Real	Value of radius at axis point ( $\geq 0.0$ )
4	SANGLE	Real	Value of semi-angle in degrees ( $> 0.0$ and $< 90.0$ )

Additional pointers as required (see Section 2.2.4.5.2).

### Parameterized Right Circular Conical Surface Entity (Type 194, Form 1)

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DELOC	Pointer	Pointer to the DE of the point on axis (LOCATION)
2	DEAXIS	Pointer	Pointer to the DE of the axis direction (AXIS)
3	RADIUS	Real	Value of radius at axis point ( $\geq 0.0$ )
4	SANGLE	Real	Value of semi-angle in degrees ( $> 0.0$ and $< 90.0$ )
5	DEREFD	Pointer	Pointer to the DE of the reference direction (REFDIR)

Additional pointers as required (see Section 2.2.4.5.2).

**4.52 RIGHT CIRCULAR CONICAL SURFACE ENTITY (TYPE 194)‡**

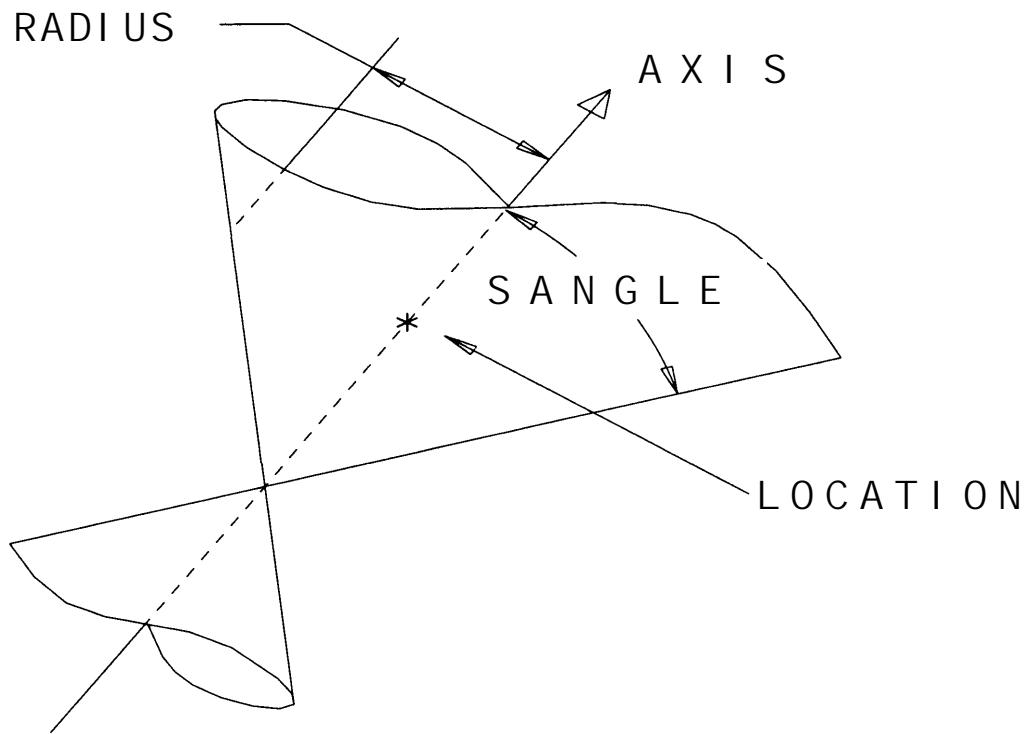


Figure 64. Defining data for un-parameterized right circular conical surface (Form Number = 0)

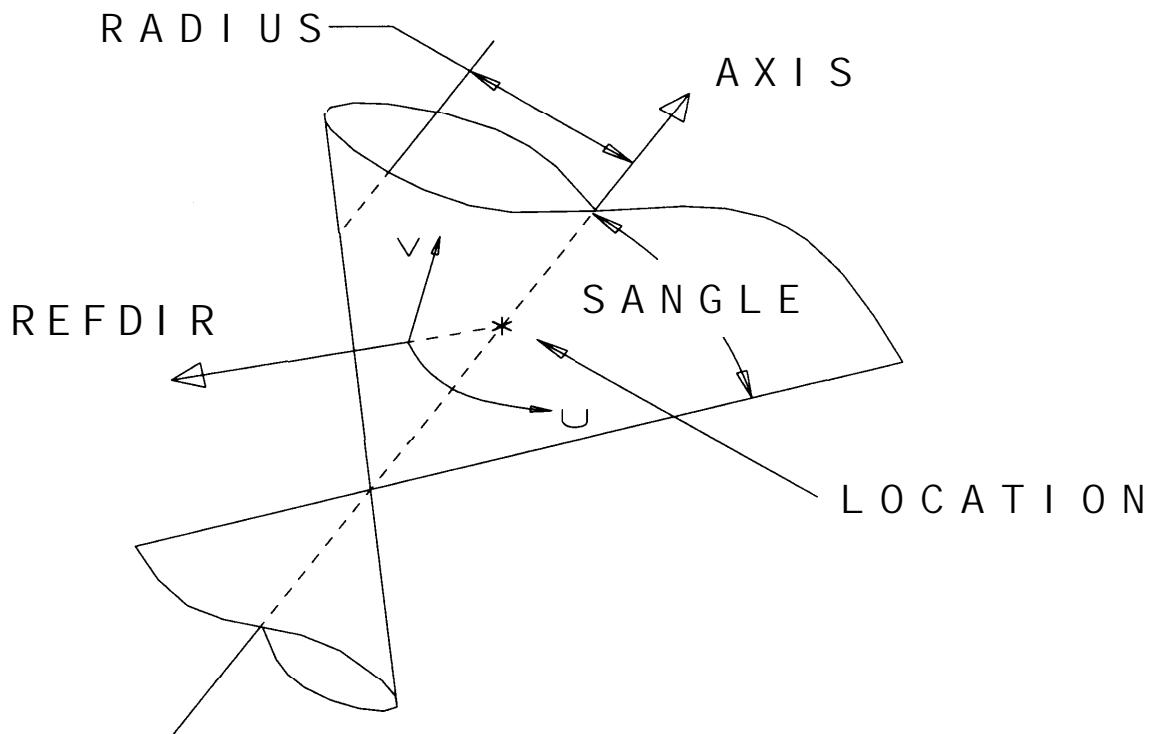


Figure 65. Defining data for parameterized right circular conical surface (Form Number = 1)

## 4.53 SPHERICAL SURFACE ENTITY (TYPE 196)‡

### 4.53 Spherical Surface Entity (Type 196)‡

ECO630

‡The Spherical Surface Entity has not been tested. [See Section 1.9.](#)

The spherical surface is defined by the center point and the radius. Figures 66 and 67 show examples. The positive direction of the surface normal is outwards from the center.

If a local coordinate system is defined with the origin at the center point then the equation of the surface in this system is  $S = 0$  where

$$S(x, y, z) = x^2 + y^2 + z^2 - r^2$$

and the positive direction of the surface normal is in the direction of increasing  $S$ . The normal,  $\mathbf{N}$ , to the surface at any point on the surface is given by

$$\mathbf{N} = (S_x, S_y, S_z)$$

The data for the parameterized form of the surface are to be interpreted as follows:

<b>C</b>	= LOCATION
<b>z</b>	= <AXIS>
<b>d</b>	= <REFDIR>
<b>x</b>	= ( <b>d</b> - ( <b>d.z</b> ) <b>z</b> )
<b>y</b>	= ( <b>z</b> × <b>x</b> )
<b>r</b>	= RADIUS

and the surface is parameterized as

$$\sigma(u, v) = \mathbf{C} + r(\cos(v)(\cos(u)\mathbf{x} + \sin(u)\mathbf{y}) + \sin(v)\mathbf{z})$$

where the parameterization range is  $0 \leq u \leq 360$  degrees and  $-90 \leq v \leq 90$  degrees.

Note that **d** shall be distinct from **z** and shall be approximately perpendicular to **z**.

For the Spherical Surface Entity, the Form Numbers are as follows:

Form	Meaning
0	Unparameterized surface
1	Parameterized surface

This surface type is intended to represent the geometry underlying topology, and shall only be referenced by a Face Entity ([Type 510, Form 1](#)). The Subordinate Entity Switch shall always be set to Physically Dependent; *i.e.*, independent instances of this entity are not permitted.

## 4.53 SPHERICAL SURFACE ENTITY (TYPE 196)‡

### Directory Entry

(1) Entity Type Number 196	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number **01??**	(10) Sequence Number D #
(11) Entity Type Number 196	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

### Un-parameterized Spherical Surface Entity (Type 196, Form 0)

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DELOC	Pointer	Pointer to the DE of the center point (LOCATION)
2	RADIUS	Real	Value of radius (> 0.0)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

### Parameterized Spherical Surface Entity (Type 196, Form 1)

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DELOC	Pointer	Pointer-to the DE of the center point (LOCATION)
2	RADIUS	Real	Value of radius (> 0.0)
3	DEAXIS	Pointer	Pointer to the DE of the axis direction (AXIS)
4	DEREFD	Pointer	Pointer to the DE of the reference direction (REFDIR)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.53 SPHERICAL SURFACE ENTITY (TYPE 196)‡**

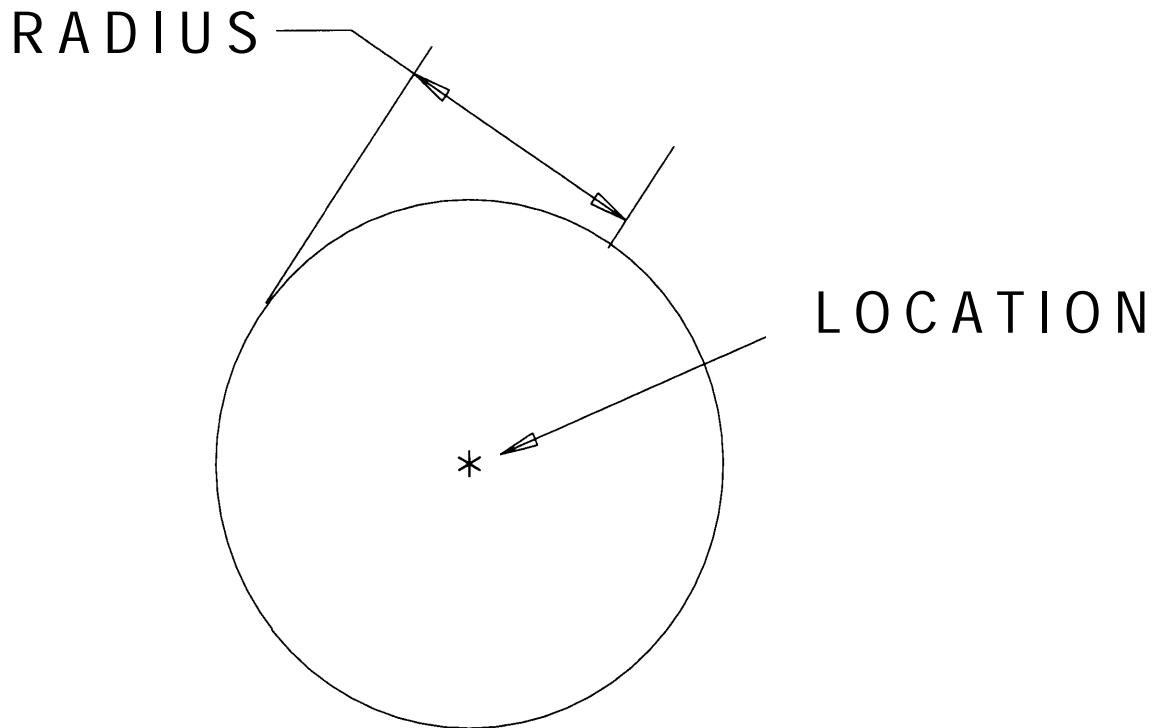


Figure 66. Defining data for un-parameterized spherical surface (Form Number = 0)

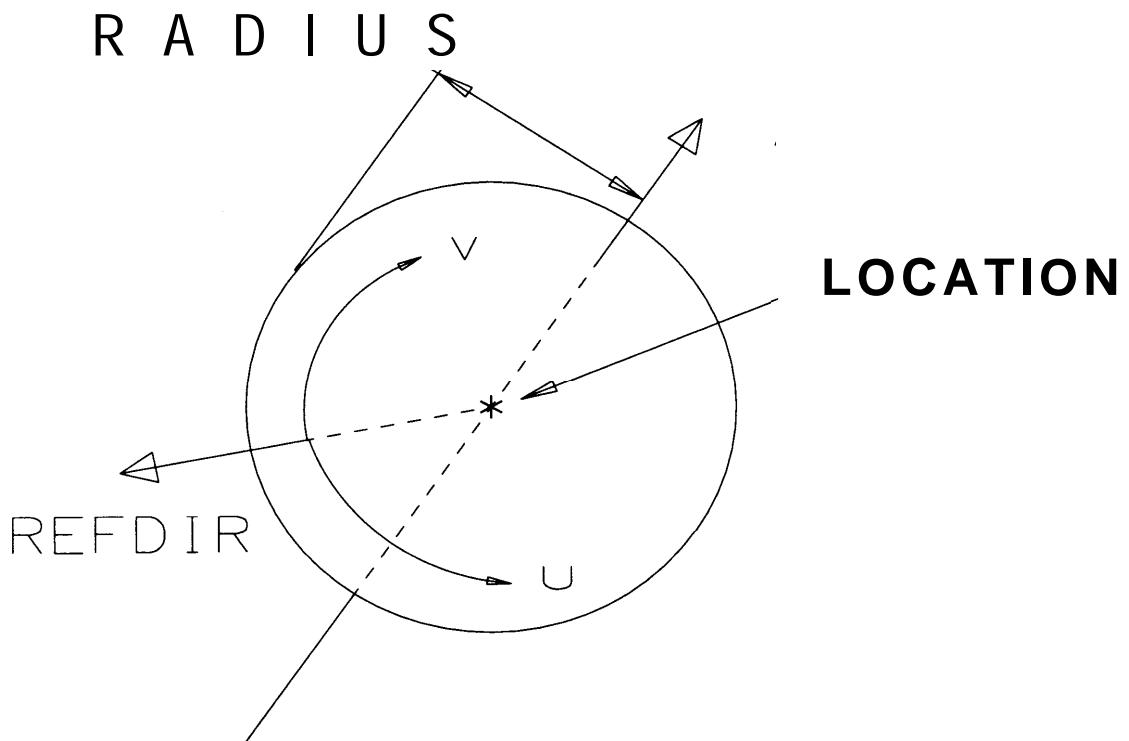


Figure 67. Defining data for parameterized spherical surface (Form Number = 1)

## 4.54 Toroidal Surface Entity (Type 198) ‡

EC0630

‡The Toroidal Surface Entity has not been tested. See Section 1.9.

The toroidal surface is defined by the center point, the axis direction and the major and minor radii. Figures 68 and 69 show examples. The positive direction of the surface normal is outwards from the center of the generating circle.

If a local coordinate system is defined with the origin at the axis point and the Z axis in the axis direction, then the equation of the surface in this system is  $S = 0$  where

$$S(x, y, z) = x^2 + y^2 + z^2 - 2R\sqrt{x^2 + y^2} - r^2 + R^2$$

and the positive direction of the surface normal is in the direction of increasing  $S$ . The surface normal,  $\mathbf{N}$ , at any point on the surface is given by

$$\mathbf{N} = (S_x, S_y, S_z)$$

The data for the parameterized form of the surface are to be interpreted as follows:

<b>C</b>	= LOCATION
<b>z</b>	= <AXIS>
<b>d</b>	= <REFDIR>
<b>x</b>	= ( <b>d</b> - ( <b>d</b> • <b>z</b> ) <b>z</b> )
<b>y</b>	= ( <b>z</b> × <b>x</b> )
<b>R</b>	= M A J R A D
<b>r</b>	= M I N R A D

and the surface is parameterized as

$$\sigma(u, v) = \mathbf{C} + (R + r \cos(u))(\cos(v)\mathbf{x} - \sin(v)\mathbf{y}) + r \sin(u)\mathbf{z}$$

where the parameterization range is  $0 \leq u, v \leq 360$  degrees.

Note that **d** shall be distinct from **z** and shall be approximately perpendicular to **z**.

For the Toroidal Surface Entity, the Form Numbers are as follows:

Form	Meaning
0	Unparameterized surface
1	Parameterized surface

This surface type is intended to represent the geometry underlying topology, and shall only be referenced by a Face Entity (Type 510, Form 1). The Subordinate Entity Switch shall always be set to Physically Dependent; i.e., independent instances of this entity are not permitted.

**Directory Entry**

(1) Entity Type Number 198	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number **01??**	(10) Sequence Number D #
(11) Entity Type Number 198	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

**Un-parametrized Toroidal Surface Entity (Type 198, Form 0)****Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DELOC	Pointer	Pointer to the DE of the center point (LOCATION)
2	DEAXIS	Pointer	Pointer to the DE of the axis direction (AXIS)
3	MAJRAD	Real	Value of major radius (> 0.0)
4	MINRAD	Real	Value of minor radius (> 0.0 and < MAJRAD)

Additional pointers as required (see Section 2.2.4.5.2).

**Parametrized Toroidal Surface Entity (Type 198, Form 1)****Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DELOC	Pointer	Pointer to the DE of the center point (LOCATION)
2	DEAXIS	Pointer	Pointer to the DE of the axis direction (AXIS)
3	MAJRAD	Real	Value of major radius (> 0.0)
4	MINRAD	Real	Value of minor radius (> 0.0 and < MAJRAD)
5	DEREFID	Pointer	Pointer to the DE of the reference direction (REFDIR)

Additional pointers as required (see Section 2.2.4.5.2).

**4.54 TOROIDAL SURFACE ENTITY (TYPE 198) ‡**

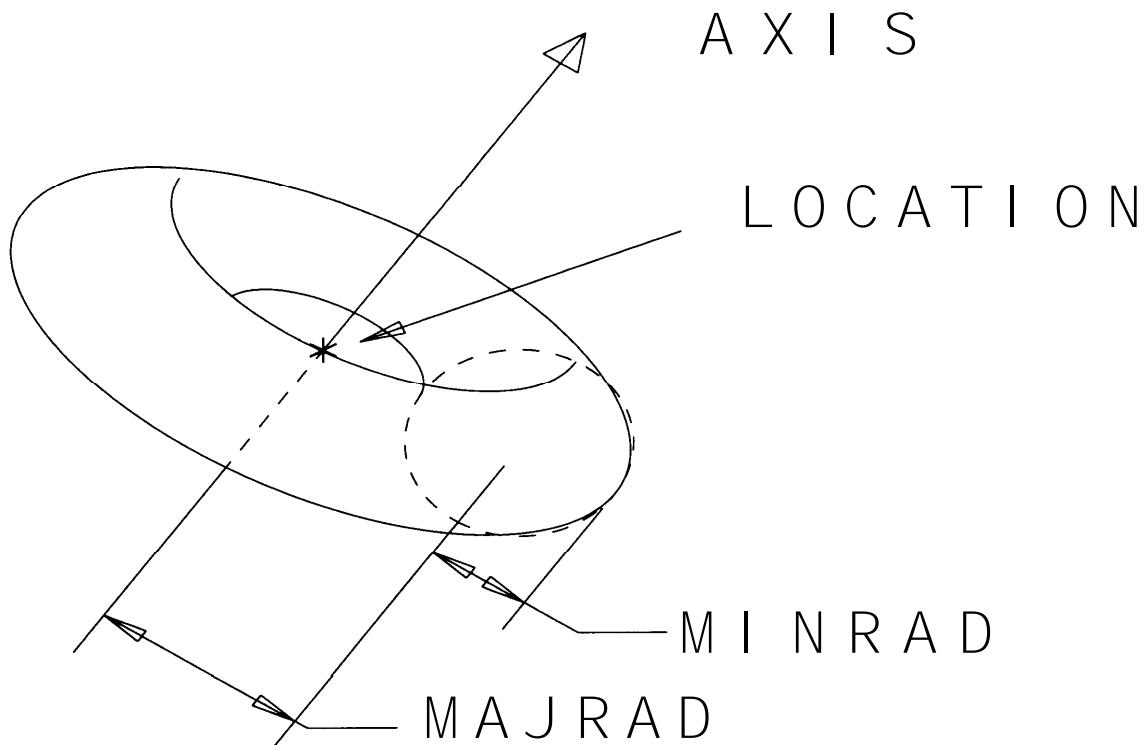


Figure 68. Defining data for un-parameterized toroidal surface (Form Number = 0)

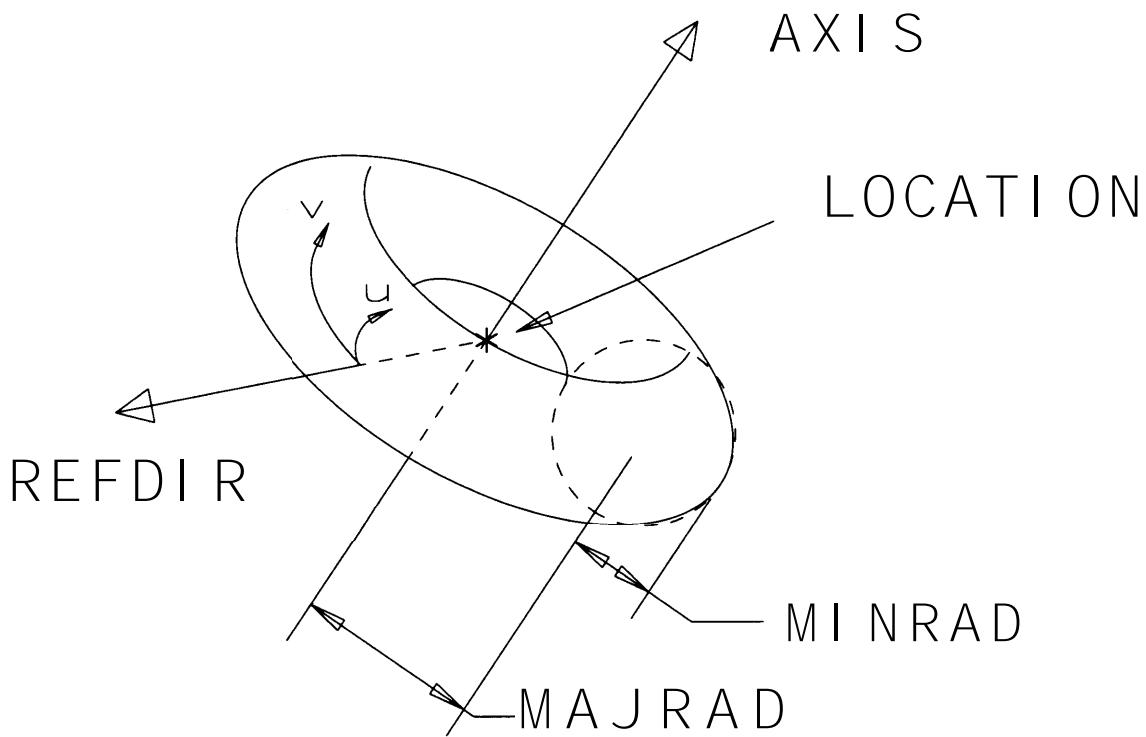


Figure 69. Defining data for parameterized toroidal surface (Form Number = 1)

### **4.55 Angular Dimension Entity (Type 202)**

An Angular Dimension Entity consists of a general note; zero, one, or two witness lines; two leaders; and an angle vertex point. [Figure 70](#) indicates the construction used. [Figure 71](#) shows examples of angular dimensions. If two witness lines are used, each is contained in its own Copious Data Entity (Type 106, Form 40).

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. The radius of the arc segments in the leader shall be calculated between the vertex point and the start point of the leader. ([Refer to Section 3.2.4](#) for information relating to the use of the term counterclockwise).

[Section 4.62](#) 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 shall be ignored. This convention arises to facilitate the definition of the second circular arc segment such as in the bottom leader in [Figure 70](#). The first example in [Figure 71](#) illustrates a leader with three segments.

See [Section 3.5.3](#) for coplanarity requirements for dimension entities.

ECO635

### **Directory Entry**

(1) Entity Type Number 202	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????01??	(10) Sequence Number D #
(11) Entity Type Number 202	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

### **Parameter Data**

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	DENOTE	Pointer	Pointer to the DE of the General Note Entity
2	DEWIT1	Pointer	Pointer to the DE of the first Witness Line Entity or zero
3	DEWIT2	Pointer	Pointer to the DE of the second Witness Line Entity or zero
4	XT	Real	Coordinates of vertex point
5	YT	Real	
6	R	Real	Radius of Leader arcs
7	DEARRW1	Pointer	Pointer to the DE of the first Leader Entity
8	DEARRW2	Pointer	Pointer to the DE of the second Leader Entity

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### 4.55 ANGULAR DIMENSION ENTITY (TYPE 202)

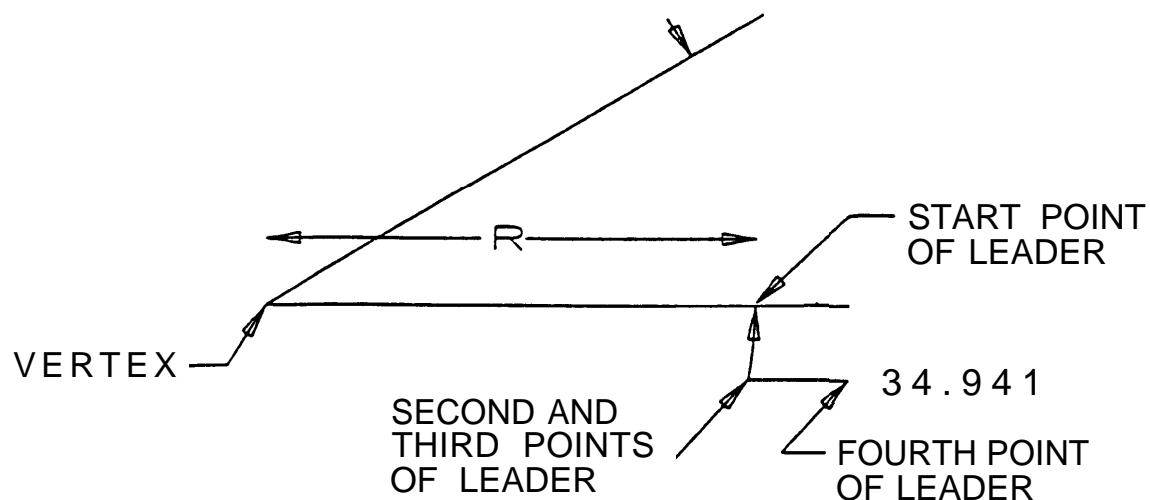


Figure 70. Construction of Leaders for the Angular Dimension Entity

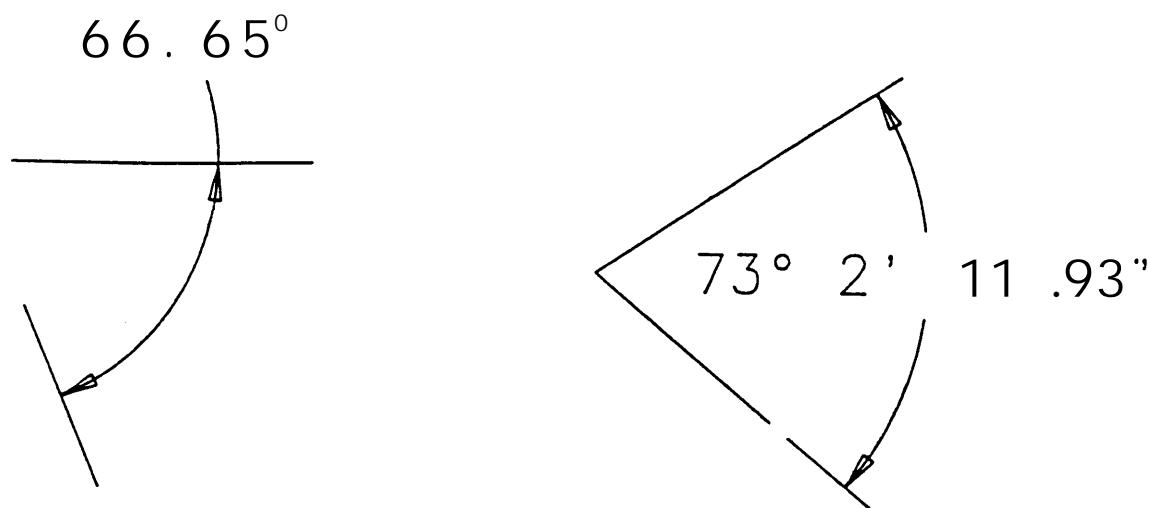


Figure 71. F202X.IGS Examples Defined Using the Angular Dimension Entity

## 4.56 CURVE DIMENSION ENTITY (TYPE 204) ‡

### 4.56 Curve Dimension Entity (Type 204) ‡

ECO630

‡The Curve Dimension Entity has not been tested. See Section 1.9.

A Curve Dimension Entity consists of a general note; one or two curves (which can be any of the parameterized curves); two leaders; and zero, one, or two witness lines. Refer to [Figure 72](#) for examples. Both parameterized curves shall not be Line Entities ([Type 110](#)); in this case a Linear Dimension ([Type 216](#)) is appropriate.

Each leader entity consists of one tail segment of non-zero length which begins with an arrowhead, and which serves only to define the orientation of the arrowhead.

The start and terminate point of a curve are determined by its parameterization. The start point of the curve has the lowest parameterization value; the terminate point of the curve has the highest parameterization value.

In the case where one curve is defined, the coordinates of the curve start point coincide with the coordinates of the arrowhead of the first leader. The coordinates of the curve terminate point coincide with the coordinates of the arrowhead of the second leader.

In the case where two curves are defined, the coordinates of the start point of the first curve coincide with the coordinates of the arrowhead of the first leader. The coordinates of the terminate point of the second curve coincide with the coordinates of arrowhead of the second leader.

#### Directory Entry

(1) Entity Type Number 204	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????01??	(10) Sequence Number D #
(11) Entity Type Number 204	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DENOTE	Pointer	Pointer to the DE of the General Note Entity
2	DECURV1	Pointer	Pointer to the DE of the first curve entity
3	DECURV2	Pointer	Pointer to the DE of the second curve entity, or zero
4	DEARR1	Pointer	Pointer to the DE of the first Leader Entity
5	DEARR2	Pointer	Pointer to the DE of the second Leader Entity
6	DEWIT1	Pointer	Pointer to the DE of the first Witness Line Entity, or zero
7	DEWIT2	Pointer	Pointer to the DE of the second Witness Line Entity, or zero

Additional pointers as required ([see Section 2.2.4.5.2](#)).

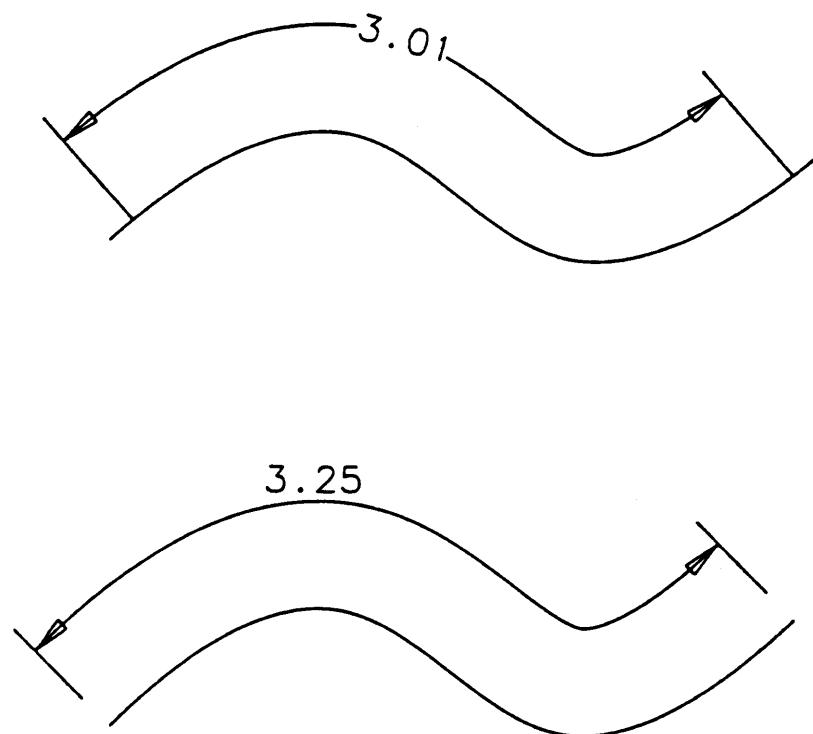


Figure 72. Examples Defined Using the Curve Dimension Entity

## 4.57 DIAMETER DIMENSION ENTITY (TYPE 206)

### 4.57 Diameter Dimension Entity (Type 206)

A Diameter Dimension Entity consists of a general note, one or two leaders, and an arc center point. Refer to [Figure 73](#) for examples of the Diameter Dimension Entity.

The arc center is used as a reference in constructing the diameter dimension but has no effect on ECO630 the dimension components.

See [Section 3.5.3](#) for coplanarity requirements for dimension entities.

ECO635

#### Directory Entry

(1) Entity Type Number 206	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????01??	(10) Sequence Number D #
(11) Entity Type Number 206	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DENOTE	Pointer	Pointer to the DE of the General Note Entity
2	DEARRW1	Pointer	Pointer to the DE of the first Leader Entity
3	DEARRW2	Pointer	Pointer to the DE of the second Leader Entity or zero
4	XT	Real	Arc center coordinates
5	YT	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### 4.57 DIAMETER DIMENSION ENTITY (TYPE 206)

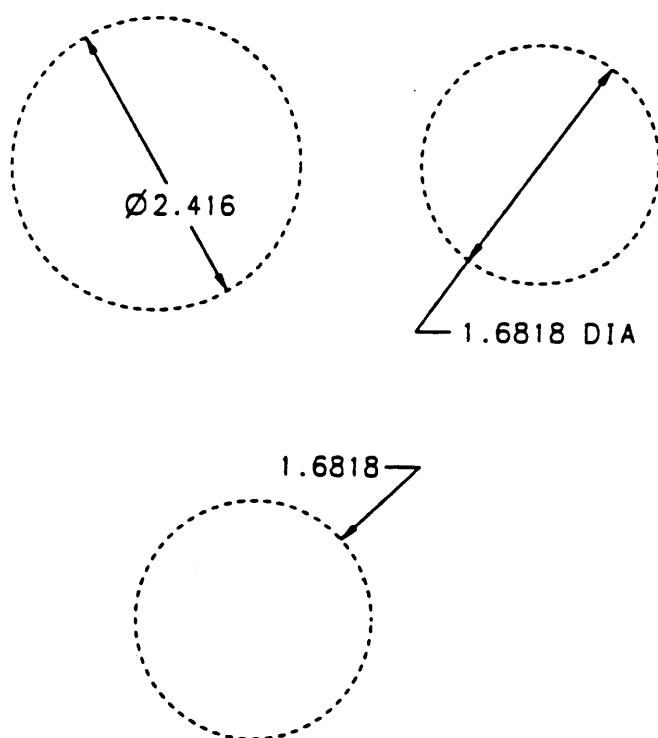


Figure 73. F206X.IGS Examples Defined Using the Diameter Dimension Entity

**4.58 Flag Note Entity (Type 208)**

A Flag Note Entity defines label information which is formatted as shown in [Figure 74](#). The geometric ECO630 parameters of the Flag Note Entity are defined using information from the General Note Entity as follows:

$$\begin{aligned}H &= 2Hc \\L &= W + 0.4Hc \\T &= 0.5H/\tan(35^\circ),\end{aligned}$$

where

*H* = Height  
*Hc* = Character Height (from General Note)  
*L* = Length  
*W* = Text Width (from General Note)  
*T* = Tip Length  
*A* = Rotation Angle (in radians).

*H* shall never be less than 0.3 in., and *L* shall never be less than 0.6 in. The box containing the text (as defined in the General Note Entity) shall be centered in the flag note box of size (*H* x *L*). The rotation angle and location of the lower left corner coordinate in the Flag Note Entity override the General Note Entity ([Type 212](#)) rotation angle and placement.

The Flag Note Entity may be defined with or without leaders.

The general note may consist of multiple text strings; however, they shall share a common baseline. The number of characters shall not be greater than 10.

Examples defined using the Flag Note Entity are shown in [Figure 75](#).

See [Section 3.5.3](#) for coplanarity requirements for dimension entities.

ECO635

## 4.58 FLAG NOTE ENTITY (TYPE 208)

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
208	$\Rightarrow$	$< n.a. >$	$\#, \Rightarrow$	$\#, \Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	????01??	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
208	#	$\#, \Rightarrow$	#	0			#		D # + 1

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	X T	Real	Lower left corner coordinate of the Flag
2	Y T	Real	
3	Z T	Real	
4	A	Real	Rotation angle in radians
5	DENOTE	Pointer	Pointer to the DE of the General Note Entity
6	N	Integer	Number of Arrows (Leaders) or zero
7	DEARRW(1)	Pointer	Pointer to the DE of the first associated Leader Entity
:	:	:	
6 + N	DEARRW(N)	Pointer	Pointer to the DE of the last associated Leader Entity

Additional pointers as required (see Section 2.2.4.5.2).

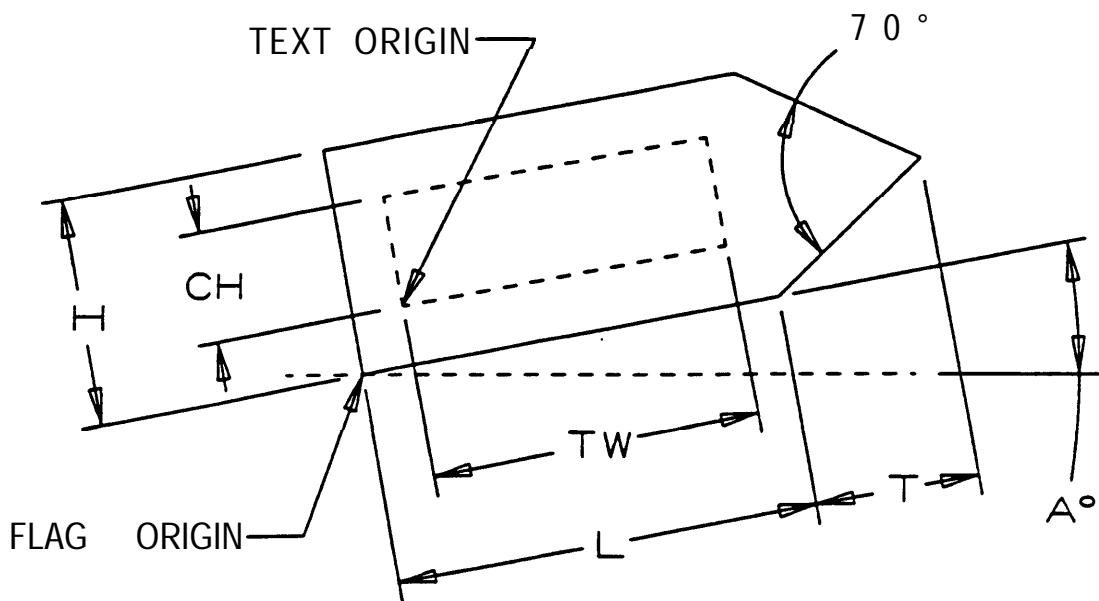


Figure 74. Parameters of the Flag Note Entity. Note that the box outlined within the flag illustrates the bounds of the text and is not a sub-symbol.

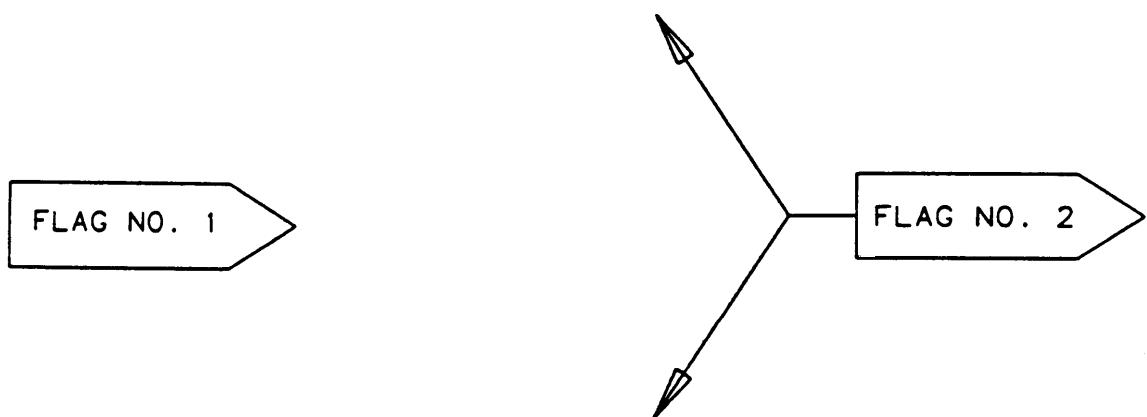


Figure 75. Examples Defined Using the Flag Note Entity

## 4.59 GENERAL LABEL ENTITY (TYPE 210)

### 4.59 General Label Entity (Type 210)

A General Label Entity consists of a general note with one or more associated leaders. Examples of general labels are shown in [Figure 76](#).

See [Section 3.5.3](#) for coplanarity requirements for dimension entities.

ECO635

#### Directory Entry

(1) Entity Type Number 210	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????01??	(10) Sequence Number D #
(11) Entity Type Number 210	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DENOTE	Pointer	Pointer to the DE of the associated General Note Entity
2	N	Integer	Number of Leaders
3	DEARRW(1)	Pointer	Pointer to the DE of the first associated Leader Entity
:	:	:	
2 + N	DEARRW(N)	Pointer	Pointer to the DE of the last associated Leader Entity

Additional pointers as required (see [Section 2.2.4.5.2](#)).

**4.69 GENERAL LABEL ENTITY (TYPE 210)**

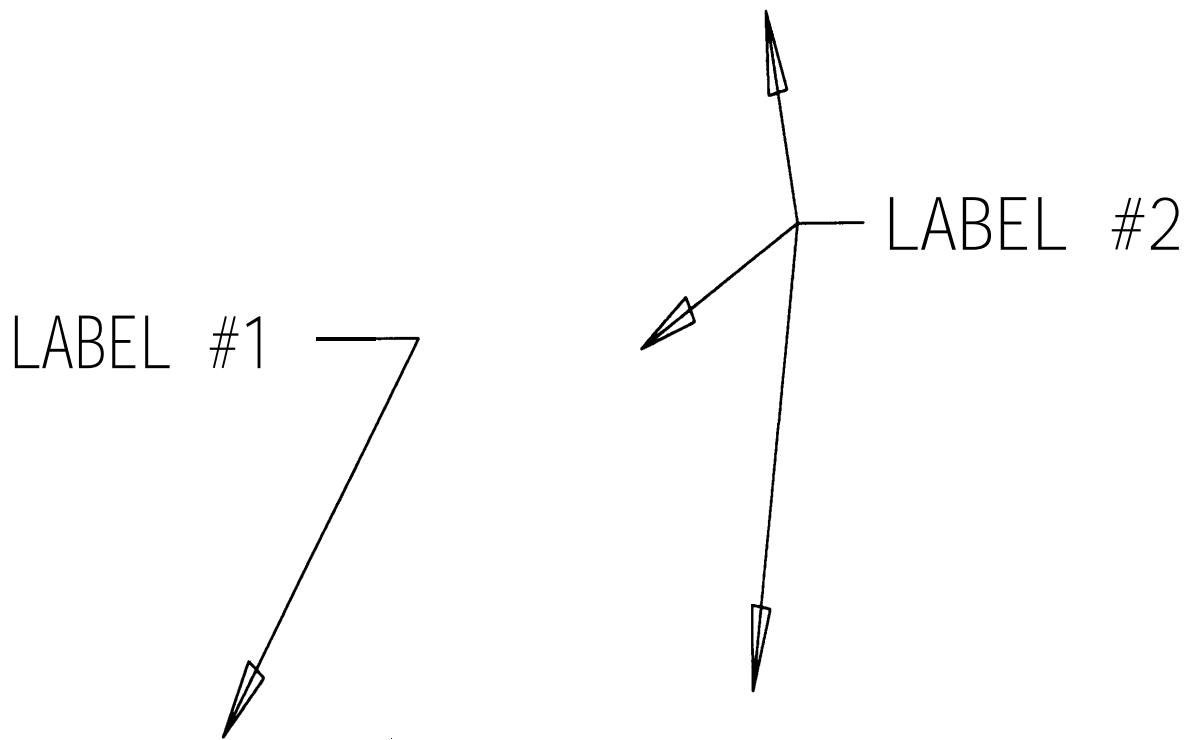


Figure 76. F210X.IGS Examples Defined Using the General Label Entity

#### 4.60 General Note Entity (Type 212)

A General Note Entity consists of one or more text strings. Each text string contains text, a starting point, a text size, and an angle of rotation of the text. Examples of general notes are shown in [Figure 77](#). The font code (FC) is an integer specifying the desired character set and its associated display characteristics. Positive values are pre-defined fonts. Negative values point to implementor-defined fonts or modifications to a pre-defined font, through the use of the Text Font Definition Entity ([Type 310](#)).

The following font codes are defined:

FC	Description
0	Symbol Font (no longer recommended)
1	Default Style for ASCII Character Set
2	LeRoy
3	Futura
6	Comp 80
12	News Gothic
13	Lightline Gothic
14	Simplex Roman
17	Century Schoolbook
18	Helvetica
19	OCR-B [ISO1073]‡
1001	Symbol Font 1
1002	Symbol Font 2
1003	Drafting Font
2001	Kanji [JIS6226]
3001	Latin-1 Alphabet‡

‡Font codes 19 and 3001 of the General Note Entity have not been tested. [See Section 1.9](#).

**FC 0** specifies an old symbol font and should no longer be used. [Figure F1](#) in [Appendix F](#) is a mapping symbol definition for FC 0

**FC 1** does not specify a defined display. Use of Font 1 implies that the receiving system may use any font which displays the appropriate ASCII format characters. The intent of this font is for usage when the actual display of the characters is not critical for the application.

**FC 19** ‡ specifies the OCR-B font [ISO01073] and is defined in [Figure 81](#). Display symbols shall be ECO630 represented using 7-bit ASCII codes with FC values in the 1000 series as shown in [Figures 82, 83](#) and [84](#). The 7-bit ASCII control characters, i.e., hexadecimal 00 through 1F and hexadecimal 7F, shall not be used to represent display symbols. They do not specify a character display font.

**FC 2001** specifies Japanese characters defined by the JIS Kanji (Kuten) Code Table [JIS6226]. ECO622 Values in that table are implemented here as a two hexadecimal digit row number followed by a two hexadecimal digit column number. (Leading or embedded zeroes, or both, shall be used to avoid confusion.) The fact that four consecutive ASCII characters are being used to represent one character in the alphabet is implicit in the FC, and a postprocessor which supports this FC shall behave accordingly.

## 4.60 GENERAL NOTE ENTITY (TYPE 212)

The hexadecimal row/column codes are biased by 20 (decimal 32). As an example, the characters represented by the decimal Kuten codes 20, 33 ("KAN") and 27, 90 ("JI") is coded as "8H34413B7A" ( $20 + 32 = 52_{10} = 34_{16}$  etc.).

The same value shall appear in the NC field of the PD record as appears in the Hollerith constant, e.g., even though 2 Kanji characters are represented as 8 Hollerith characters, NC shall have a value of 8 rather than 2.

Preprocessors shall define the text box height and box width so as to accurately reflect the display box size for the text string. Postprocessors which cannot display Japanese characters shall process this FC as if it were FC 1 (default style for ASCII character set).

The Rotate Internal Text Flag (VH) field in the PD record shall be used to convey vertical text orientation.

The Embedded Font Change form (Form 2) shall be used when the text note combines mixed English and Japanese fonts.

Embedded "escape" characters or metacharacters shall not be used; all of the characters are assumed to be for display.

**FC 3001‡** specifies European characters defined by the ISO 8859-1 standard [ISO8859], also known as the Latin-1 Alphabet. FC3001 is shown in [Figure 85](#). Values in ISO 8859-1 are implemented here as two ASCII characters, the leading character being either a space, or a period. The use of two consecutive ASCII characters to represent one character in the alphabet is implicit in the FC. A postprocessor which supports this FC shall behave accordingly.

Standard ASCII characters are preceded by a space. Non-ASCII characters from the Latin-1 alphabet are preceded by a period. For example, the phrase *déjà vu* is coded as "14H<sub>10</sub>.d.i<sub>10</sub>j.<sub>10</sub>v<sub>10</sub>u". The same value shall appear in the NC field of the PD record as appears in the Hollerith constant, e.g., when 7 French characters are represented as 14 Hollerith characters, NC shall have a value of 14 rather than 7.

Preprocessors shall define the text box-height and box-width so as to accurately reflect the display box size for the text string. Postprocessors which cannot display ISO 8859-1 characters shall process this FC as if it were FC 1 (default style for ASCII character set).

Embedded "escape" characters or metacharacters shall not be used; all of the characters are assumed to be for display.

[Table 9](#) provides names for the graphical characters defined in the symbol and drafting fonts (FC 1, FC 1001, FC 1002, and FC 1003).

If the pre-defined font codes are not sufficient to describe a desired character set or display characteristic, a Text Font Definition Entity (Type 310) 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 font code (FC) parameter. The use of the values WT, HT, SL, A, and text start point are shown in [Figure 78](#).

Within definition space, the parameters for the text block are applied in the following order (see [Figure 79](#)):

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. If the rotate internal text flag is set to 1 (vertical text) then characters are placed one below another instead of one beside another. The rotate internal flag has no effect on the orientation of individual characters; it only affects their positioning.

Regardless of the setting of the rotate internal text flag, the box width is measured as the sum of the widths of the N individual characters or symbols in the string, plus the width of N-1 inter-character spaces. For horizontal text, this may be interpreted as the width measured from the start of the left-most (first) text character or symbol in the positive XT direction along the text base line, and extending to the end of the right-most (last) character or symbol, extending N characters or symbols and N-1 inter-character spaces. ECO630

Regardless of the setting of the rotate internal text flag, the box height is measured in the positive YT direction and is the height of a single capital letter. It is equivalent to the symbol "h" used in [Appendix C](#) of [ANS182]. Special symbols, such as those appearing in [Appendix C](#) of [ANS182], which exceed "h" in height are centered vertically. Descenders and portions of symbols exceeding "h" extend outside the lower and upper borders of the box (see [Figure 80](#)). ECO630

The box height and width are measured before the rotation angle (A) is applied. The text start point is defined as the lower left corner of the first character or symbol box. ECO630

If the rotate internal flag is set to vertical text, then the vertical spacing between the baselines of consecutive characters is 1.5 times the box height. The inter-character spacing shall be assumed to be 0.1 times the width of a single character, unless this is overridden by the use of an Inter-character Spacing Property ([Type 406 Form 18](#)). ECO629 ECO630

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 counter-clockwise direction about the text start point. The plane of rotation is the XT, YT plane at the depth Z S(n) (where Z S(n) is the value given for the text start point). ECO630
4. The mirror operation is performed next. The value 1 indicates the mirror axis is the (rotated) line perpendicular to the text base line and through the text start point. The value 2 indicates the mirror axis is the (rotated) text base line.

Finally, the Transformation Matrix Entity is used to specify the relative position of definition space within model space.

The number of characters (NC(n)) shall be equal to the character count in its corresponding text string (TEXT(n)). ECO630

The graphical representation and recreation of notes with a special structure are handled by the use of the Form Number in Field 15 of the Directory Entry for this entity. A system to accommodate these notes is outlined below. Any strings after those specified by the form number are considered additional, appended strings that are not related in any particular manner to the previously referenced strings.

In the event that a string necessary for the defined structure is not present in the sending system's note, a null string (see NULL STRING in [Appendix K](#)) shall be inserted in the General Note Entity to take the place of the nonexistent string to maintain the structure of the data. ECO630

Notes that contain fractional notation shall be represented as mixed numerals. This is done through the use of four consecutive strings representing the whole number, the numerator, the denominator, and the divisor bar. These are examples of the divisor bar string

1H/ 1H- 2H-- 1H\_-

## 4.60 GENERAL NOTE ENTITY (TYPE 212)

The following form numbers for the general note are used to maintain the graphical representation of the originating system's note:

**Form 0** Simple Note (default) – A general note of one or more strings such that a text string is not related in any manner to another string in the same General Note Entity.

**Form 1:** Dual Stack – A general note of two or more strings where the first two are related in a manner such that they are both left justified and the second string is displayed “below” the first.

xxxxxx  
yyyyy

**Form 2:** Imbedded Font Change – A general note of two or more strings that is intended as a single string but was divided to accommodate a font change in the string.

**Form 3:** Superscript – A general note of two or more strings where the second string is a superscript of the first string.

xxx<sup>yyy</sup>

**Form 4:** Subscript – A general note of two or more strings where the second string is a subscript of the first string.

xxx<sub>yyy</sub>

**Form 5:** Superscript, Subscript – A general note of three or more strings where the second string ECO630 is a superscript of the first string and the third string is a subscript of the first string.

xxx<sup>yyy</sup><sub>zzz</sub>

**Form 6:** Multiple Stack, Left Justified – A general note where all strings are left justified to a common margin. These strings originated as a “paragraphed” note.

xxxxxxxxxx  
yyyyyyyy  
zzzzzzzzzz

**Form 7:** Multiple Stack, Center Justified A general note where all strings are center justified to a common axis.

xxxxxxxxxx  
yyyy  
zzzzzzzz

**Form 8:** Multiple Stack, Right Justified A general note where all strings are right justified to a common margin.

xxxxxxxxxx  
yyyyy  
zzzzzzzz

**Form 100:** Simple Fraction – A general note of four or more strings where the first four strings define a mixed numeral as defined previously.

yy  
xx —  
zz

**Form 101:** Dual Stack Fraction – A general note of eight or more strings which represent two mixed numerals as defined previously. These mixed numerals are related such that the fifth through the eighth strings are displayed below the first through the fourth strings respectively.

$$\begin{array}{c} \text{yy} \\ \text{xx} \quad \text{--} \\ \text{zz} \\ \\ \text{jj} \\ \text{ii} \quad \text{--} \\ \text{kk} \end{array}$$

**Form 102:** Imbedded Font Change, Double Fraction – This general note originated as a single ECO630 string but was split to accommodate a font change for a special character in the fifth string. This is a general note of nine or more strings where the first and sixth strings represent the whole number string of a mixed numeral as defined previously. The fifth string is a character (or characters) that was set apart to accommodate the font change.

$$\begin{array}{ccccc} \text{yy} & & & \text{jj} \\ \text{xx} \quad \text{--} & - & \text{ii} \quad \text{--} \\ & & & \text{kk} \end{array}$$

**Form 105:** Superscript, Subscript Fraction – A general note of twelve or more strings where the ECO630 first, fifth, and ninth strings represent the whole number string of a mixed numeral as defined previously. The second and third mixed numerals are the superscript and subscript respectively of the first mixed numeral.

$$\begin{array}{c} \left( \begin{array}{c} \text{ii} \quad \text{jj} \\ \text{--} \\ \text{kk} \end{array} \right) \\ \left( \begin{array}{c} \text{xx} \quad \text{yy} \\ \text{--} \\ \text{zz} \end{array} \right) \\ \left( \begin{array}{c} \text{rr} \quad \text{ss} \\ \text{--} \\ \text{tt} \end{array} \right) \end{array}$$

Note: The large parentheses are added to help convey the intent of [Form 105](#). They are not part of the General Note.

**Directory Entry**

(1) Entity Type Number 212	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern 1	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????01**	(10) Sequence Number D #
(11) Entity Type Number 212	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number #	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** Valid values of the Form Number are 0-8, 100-102, 105.

**Parameter Data**

ECO650

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	NS	Integer	Number of text strings in General Note
2	NC(1)	Integer	Number of characters in first string (TEXT(1)) or zero. The number of characters (NC(n)) shall always be equal to the character count of its corresponding text string (TEXT(n))
3	WT(1)	Real	Box width
4	HT(1)	Real	Box height
5	FC(1)	Integer or Pointer	Font code (default = 1)
6	SL(1)	Real	Pointer to the DE of the Text Font Definition Entity if negative Slant angle of TEXT1 in radians ( π/2 is the value for no slant angle and is the default value)
7	A(1)	Real	Rotation angle in radians for TEXT1
8	M(1)	Integer	Mirror flag: 0 = no mirroring 1 = mirror axis is perpendicular to text base line 2 = mirror axis is text base line
9	VH(1)	Integer	Rotate internal text flag: 0 = text horizontal 1 = text vertical
10	XS(1)	Real	First text start point
11	YS(1)	Real	
12	ZS(1)	Real	Z depth from XT, YT plane
13	TEXT(1)	String	First text string
14	NC(2)	Integer	Number of characters in second text string
⋮	⋮	⋮	
-10+12*NS	NC(NS)	Integer	Number of characters in last text string
⋮	⋮	⋮	
1+12*NS	TEXT (NS)	String	Last text string

ECO626

Additional pointers as required (see Section 2.2.4.5.2).

#### 4.60 GENERAL NOTE ENTITY (TYPE 212)

*A SINGLE LINE OF SLANTED TEXT*

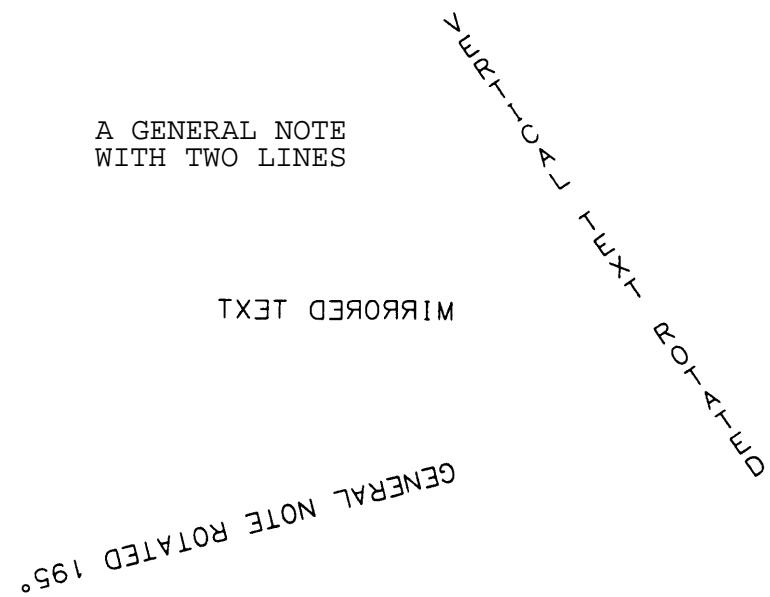


Figure 77. F212X.IGS Examples Defined Using the General Note Entity

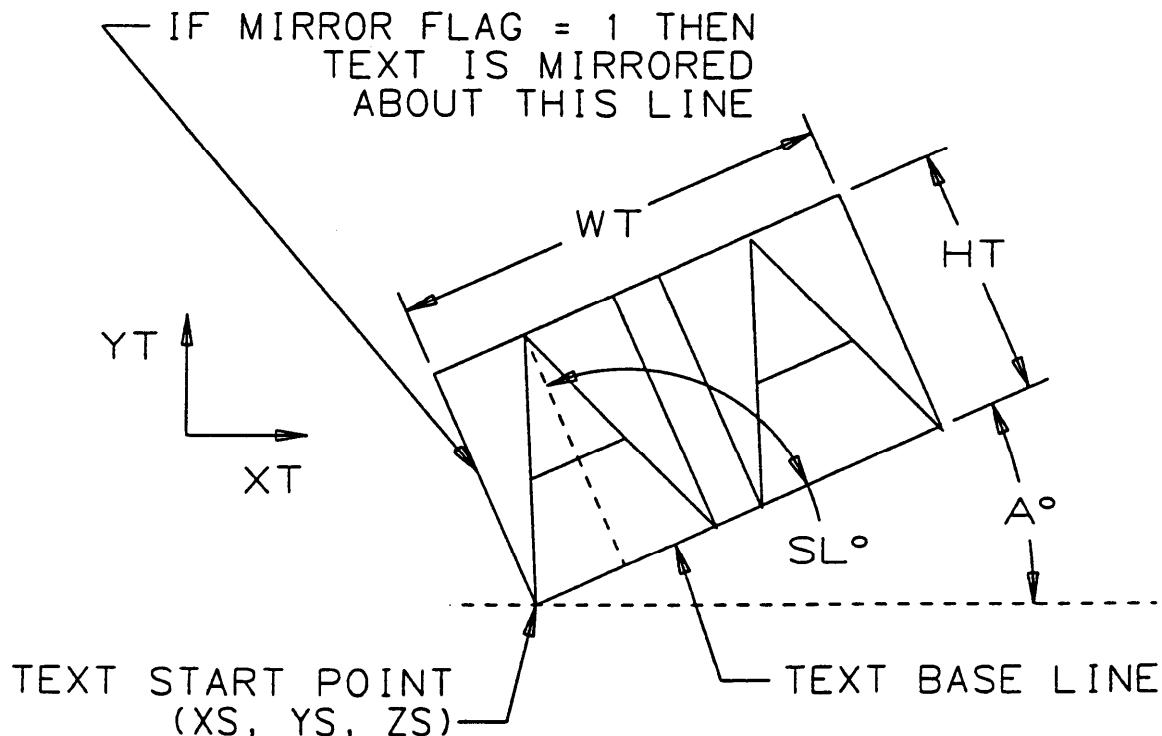


Figure 78. General Note Text Construction

#### 4.60 GENERAL NOTE ENTITY (TYPE 212)

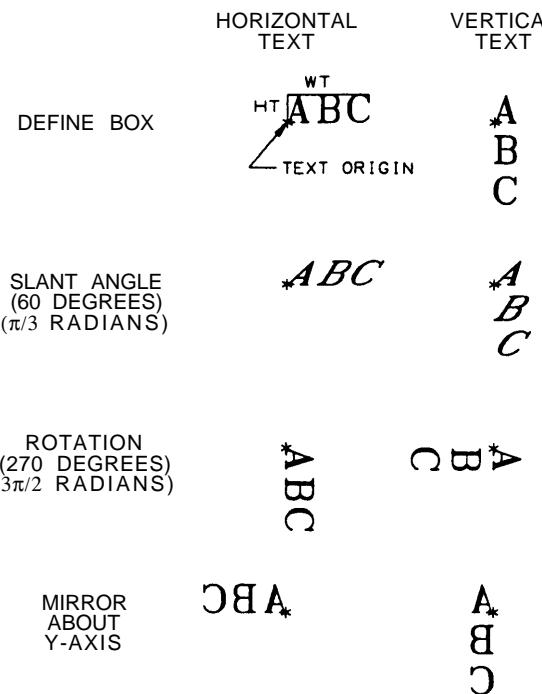


Figure 79. F212BX.IGS General Note Example of Text Operations

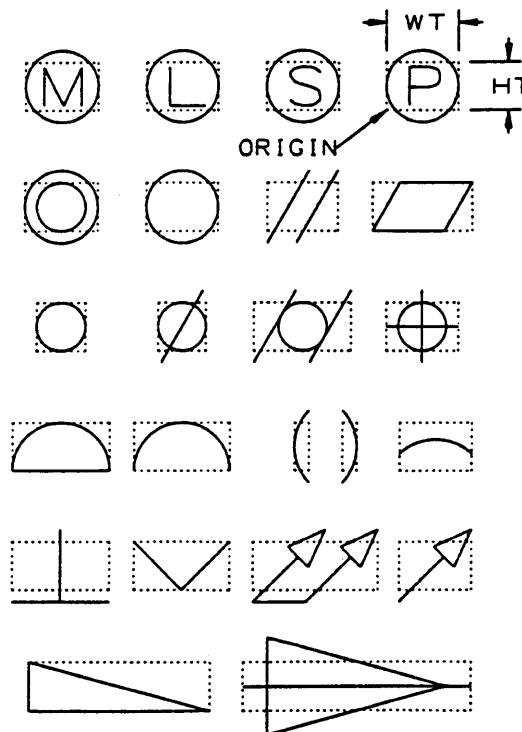


Figure 80. Examples of Drafting Symbols That Exceed Text Box Height

**4.60 GENERAL NOTE ENTITY (TYPE 212)**

BL		0	0	Q	Q	P	P	'	'	p	p
!	!	1	1	A	A	Q	Q	a	a	q	q
"	"	2	2	B	B	R	R	b	b	r	r
#	#	3	3	C	C	S	S	c	c	s	s
\$	\$	4	4	D	D	T	T	d	d	t	t
%	%	5	5	E	E	U	U	e	e	u	u
&	&	6	6	F	F	V	V	f	f	v	v
'	'	7	7	G	G	W	W	g	g	w	w
(	(	8	8	H	H	X	X	h	h	x	x
)	)	9	9	I	I	Y	Y	i	i	y	y
*	*	:	:	J	J	Z	Z	j	j	z	z
+	+	;	;	K	K	[	[	k	k	{	{
,	,	<	<	L	L	\	\	l	l		
-	-	=	=	M	M	]	]	m	m	}	}
.	.	>	>	N	N	-	-	n	n	~	~
/	/	?	?	O	O	-	-	o	o		

Figure 81. General Note Font (OCR-B) Specified by FC 19

#### 4.60 GENERAL NOTE ENTITY (TYPE 212)

BL		0	0	©	©	P	P	'	'	p	(P)
!	!	1	1	A	A	Q	Q	a	∠	q	€
"	"	2	2	B	B	R	R	b	₪	r	◎
#	#	3	3	C	C	S	S	c	□	s	⌚
\$	\$	4	4	D	D	T	T	d	△	t	□
%	%	5	5	E	E	U	U	e	○	u	○
&	&	6	6	F	F	V	V	f	//	v	△
'	'	7	7	G	G	W	W	g	↗	w	◊
(	(	8	8	H	H	X	X	h	↗	x	◆
)	)	9	9	I	I	Y	Y	i	≡	y	✗
*	*	:	:	J	J	Z	Z	j	⊕	z	Y
+	+	;	;	K	K	[	[	k	⌒	{	{
,	,	<	<	L	L	\	\	l	⊥		
-	-	=	=	M	M	]	]	m	Ⓜ	}	}
.	.	>	>	N	N	-	-	n	∅	~	~
/	/	?	?	O	O	-	-	o	○		

Figure 82. General Note Font Specified by FC 1001

**4.60 GENERAL NOTE ENTITY (TYPE 212)**

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

Figure 83. General Note Font Specified by FC 1002

#### 4.60 GENERAL NOTE ENTITY (TYPE 212)

BL		0	0	Q	Q	P	P	'	±	p	(P)
!	!	1	1	A	A	Q	Q	a	∠	q	¢
"	"	2	2	B	B	R	R	b	⊥	r	◎
#	#	3	3	C	C	S	S	c	□	s	⌚
\$	\$	4	4	D	D	T	T	d	▷	t	↶
%	%	5	5	E	E	U	U	e	○	u	—
&	&	6	6	F	F	V	V	f	//	v	└
,	'	7	7	G	G	W	W	g	∅	w	∨
(	(	8	8	H	H	X	X	h	↗	x	↓
)	)	9	9	I	I	Y	Y	i	≡	y	⇒
*	*	:	:	J	J	Z	Z	j	⊕	z	△
+	+	;	;	K	K	[	[	k	⌒	{	{
,	,	<	<	L	L	\	\	l	①		
-	-	=	=	M	M	]	]	m	Ⓜ	}	}
.	.	>	>	N	N	~	~	n	∅	~	◦
/	/	?	?	O	O	-	-	o	□		

Figure 84. General Note Font Specified by FC 1003

**4.60 GENERAL NOTE ENTITY (TYPE 212)**

BL	ÿ	ö	°	ø	À	P	Đ	`	à	p	ð
!	¡	1	±	À	Á	Q	Ñ	a	á	q	ñ
"	¢	2	²	B	Â	R	Ò	b	â	r	ò
#	£	3	³	C	Ã	S	Ó	c	ã	s	ó
\$	¤	4	'	D	Ä	T	Ô	d	ä	t	ô
%	¥	5	µ	E	Å	U	Õ	e	å	u	õ
&		6	¶	F	Æ	V	Ö	f	æ	v	ö
'	§	7	.	G	Ç	W	×	g	ç	w	÷
(	“	8	,	H	È	X	Ø	h	è	x	ø
)	©	9	¹	I	É	Y	Ù	i	é	y	ù
*	¤	:	º	J	Ê	Z	Ú	j	ê	z	ú
+	«	;	»	K	Ë	[	Û	k	ë	{	û
,	¬	<	$\frac{1}{4}$	L	Î	\	Ü	l	ì	l	ü
-		=	$\frac{1}{2}$	M	Í	]	Ý	m	í	}	ý
.	®	>	$\frac{3}{4}$	N	Î	~	Þ	n	î	~	þ
/	-	?	¿	O	Ï	-	ß	o	ï		

Figure 85. UNTESTED General Note Font Specified by FC 3001

Table 9. Character Names for the Symbol and Drafting Fonts

Name	Symbol	FC‡			
		1	1001	1002	1003
Space		20	20	20	20
Exclamation mark	!	21	21	21	21
Quotation marks	"	22	22	22	22
Pound sign	#	23	23		23
Plus/minus	±			23	60
Dollar sign	\$	24	24		24
Degree symbol	°			24	7E
Percent sign	%	25	25	25	25
Ampersand	&	26	26	26	26
Apostrophe	'	27	27	27	27
Left parenthesis	(	28	28	28	28
Right parenthesis	)	29	29	29	29
Asterisk	*	2A	2A	2A	2A
Plus sign	+	2B	2B	2B	2B
Comma	,	2C	2C	2C	2C
Minus sign/hyphen	-	2D	2D	2D	2D
Period	.	2E	2E	2E	2E
Slash	/	2F	2F	2F	2F
Numeric 0	0	30	30	30	30
Numeric 1	1	31	31	31	31
Numeric 2	2	32	32	32	32
Numeric 3	3	33	33	33	33
Numeric 4	4	34	34	34	34
Numeric 5	5	35	35	35	35
Numeric 6	6	36	36	36	36
Numeric 7	7	37	37	37	37
Numeric 8	8	38	38	38	38
Numeric 9	9	39	39	39	39
Colon	:	3A	3A	3A	3A
Semi-colon	;	3B	3B	3B	3B
Less than	<	3C	3C	3C	3C
Equal sign	=	3D	3D	3D	3D
Greater than	>	3E	3E	3E	3E
Question mark	?	3F	3F	3F	3F
Commercial at	@	40	40	40	40
Upper case letter A	A	41	41	41	41
Upper case letter B	B	42	42	42	42
Upper case letter C	C	43	43	43	43
Upper case letter D	D	44	44	44	44
Upper case letter E	E	45	45	45	45
Upper case letter F	F	46	46	46	46
Upper case letter G	G	47	47	47	47
Upper case letter H	H	48	48	48	48

‡Entries for each FC are hexadecimal ASCII equivalent

## 4.60 GENERAL NOTE ENTITY (TYPE 212)

Table 9. Character Names for the Symbol and Drafting Fonts (continued)

Name	Symbol	FC†			
		1	1001	1002	1003
Upper case letter I	I	49	49	49	49
Upper case letter J	J	4A	4A	4A	4A
Upper case letter K	K	4B	4B	4B	4B
Upper case letter L	L	4C	4C	4C	4C
Upper case letter M	M	4D	4D	4D	4D
Upper case letter N	N	4E	4E	4E	4E
Upper case letter O	O	4F	4F	4F	4F
Upper case letter P	P	50	50	50	50
Upper case letter Q	Q	51	51	51	51
Upper case letter R	R	52	52	52	52
Upper case letter S	S	53	53	53	53
Upper case letter T	T	54	54	54	54
Upper case letter U	U	55	55	55	55
Upper case letter V	V	56	56	56	56
Upper case letter W	W	57	57	57	57
Upper case letter X	X	58	58	58	58
Upper case letter Y	Y	59	59	59	59
Upper case letter Z	Z	5A	5A	5A	5A
Left bracket	[	5B	5B	5B	5B
Backward slash	\	5C	5C	5C	5C
Right bracket	]	5D	5D	5D	5D
Caret	^	5E	5E	5E	
Arc length	⌒				5E
Underscore	ˍ	5F	5F	5F	5F
Reverse quote	ˍ	60	60	60	
Lower case letter a	a	61			
Angularity	∠		61		61
Marker/symbol	ℳ			61	
Lower case letter b	b	62			
Marker/symbol	ℳ		62		
Division symbol	÷			62	
Perpendicularity	⊥				62
Lower case letter c	c	63			
Flatness	□		63		63
Less than or equal	≤			63	
Lower case letter d	d	64			
Profile of a surface	▷		64		
Greater than or equal	≥			64	
Lower case letter e	e	65			
Circularity	○		65		
Marker/symbol	△			65	

†Entries for each FC are hexadecimal ASCII equivalent

#### 4.60 GENERAL NOTE ENTITY (TYPE 212)

Table 9. Character Names for the Symbol and Drafting Fonts (continued)

Name	Symbol	FC†			
		1	1001	1002	1003
Lower case letter f	f	66			
Parallelism	/ /		66		66
Radical	✓			66	
Lower case letter g	g	67		67	
Cylindricity	◎		67		67
Cross product	x			67	
Lower case letter h	h	68			
Circular Runout	↗		68		68
Congruence	≡			68	
Lower case letter i	i	69			
Symmetry	≡		69		69
Not equal	≠			69	
Lower case letter j	j	6A			
Position	⊕		6A		6A
Integral	ʃ			6A	
Lower case letter k	k	6B			
Profile of a line	⌒		6B		6B
Implication	⊃			6B	
Lower case letter l	l	6C			
Perpendicularity	⊥		6C		
Union	∨			6C	
Least material condition	Ⓛ				6C
Lower case letter m	m	6D			
Maximum material condition	Ⓜ		6D		6D
Intersection	∧			6D	
Lower case letter n	n	6E			
Diameter	∅		6E		6E
Approximately equal	≈			6E	
Lower case letter o	o	6F			
All around applicability	○		6F		
Greek letter sigma (Sum)	Σ			6F	
Square (shape)	□				6F
Lower case letter p	p	70			
Projected tolerance zone	Ⓟ		70		70
Up arrow	↑			70	
Lower case letter q	q	71			
Centerline	℃		71		71
Down arrow	↓			71	
Lower case letter r	r	72			
Concentricity	◎		72		72
Right arrow	→			72	

†Entries for Each FC are hexadecimal ASCII equivalent

## 4.60 GENERAL NOTE ENTITY (TYPE 212)

Table 9. Character Names for the Symbol and Drafting Fonts (continued)

Name	Symbol	FC†			
		1	1001	1002	1003
Lower case letter s	s	73			
Regardless of feature size	⌚		73		73
Left arrow	←			73	
Lower case letter t	t	74			
Marker/symbol	□		74		
Greek letter phi	φ			74	
Total runout	↗				74
Lower case letter u	u	75			
Marker/symbol	○		75		
Greek letter theta	θ			75	
Straightness	—				75
Lower case letter v	v	76			
Marker/symbol	△		76		
Greek letter gamma	γ			76	
Counterbore	└				76
Lower case letter w	w	77			
Marker/symbol	◊		77		
Greek letter psi	ψ			77	
Countersink	▽				77
Lower case letter x	x	78			
Marker/symbol	✚		78		
Greek letter omega	ω			78	
Depth	↓				78
Lower case letter y	y	79			
Marker/symbol	⊗		79		
Greek letter lambda	λ			79	
Conical taper	⇒				79
Lower case letter z	z	7A			
Marker/symbol	Ү		7A		
Greek letter alpha	α			7A	
Slope	△				7A
Left brace	{	7B	7B		
Greek letter delta	δ			7B	
Vertical bar		7C	7C		7C
Greek letter mu	μ			7C	
Right brace	}	7D	7D		7D
Greek letter pi	π			7D	
Tilde	~	7E	7E		
Overscore	—			7E	

†Entries for each FC are hexadecimal ASCII equivalent

## 4.61 New General Note Entity (Type 213)‡

ECO630

‡The New General Note Entity has not been tested. See Section 1.9.

The New General Note Entity accommodates a wider range of text characteristics than the General Note Entity (Type 212). The sequence of strings within the note shall be from left to right and top to bottom within the defined “imaginary” text containment area. This entity assumes all text strings are related and coplanar.

ECO630

**4.61.1 Parameter Field Descriptions**

- 1- TXTCW** width of an imaginary text containment area drawn around all text strings within the note. There is no space between the characters and the imaginary box lines. The text containment width is established after the slant angle (SLn), rotation angle (An), character angle (CHRANGn), and mirror (Mn) are applied. See [Figure 86](#). All text must be within the defined text containment area, including descenders.
- 2- TXTCH** height of an imaginary text containment area drawn around all text strings within the note. There is no space between the characters and the imaginary box lines. The text containment height is established after the slant angle (SLn), rotation angle (An), character angle (CHRANGn), and mirror (Mn) are applied. See [Figure 86](#). All text must be within the defined text containment area, including descenders.
- 3- JUSTCD** justification of all text strings relative to the text containment area.
- 4- TXTCX, TXTCY, TXTCZ** location of the upper left corner of the imaginary containment area drawn around all text strings. See [Figure 86](#).
- 7- TXTAG** rotation angle of the text box in radians. See [Figure 86](#).
- 8- BASELX, BASELY, BASELZ** starting position of the first base line of the text strings. The base line is the imaginary line upon which the normal characters are placed. Control codes are used to place characters in a position away from the baseline. The superscript is an example of a control code which moves the character away from the baseline. The baseline is horizontal and can be rotated with TXTAG. See [Figure 86](#).
- 11- NILS** normal interline spacing between baselines. The distance is between two lines of text which only have the new line control code associated with both. The inter-line space would not be normal if fraction, superscript, subscript, etc., text is on the same base line. In Figure 86, the distance between the base line for the string “TOLERANCE AND’ and the base line for the string “CENTERED’ is the normal interline space. The distance between the baseline for the string “5.00” and the baseline for the string “TOLERANCE AND” is not the normal interline spacing. A negative NILS is allowed. See [Figure 86](#).
- 13- FIXVAR** integer switch indicating whether the character and font set specified is displayed with fixed spacing (*i. e.*, an “I” uses the same amount of space as a “M”) or is variable spaced. Box width WTn establishes the outer boundary into which the text TEXT(n) shall fit.
- 14- CHRWID** the width of a character excluding its preceding and succeeding spacing. The character width is not changed by the slant angle, rotation angle, or character angle.  
*Variable width character display fonts:* The width of the widest character in the font, typically the character capital “M.”  
*Fixed width character display fonts:* The width of any character.  
The character width shall be a positive non-zero value. See [Figures 88, 92, and 93](#). ECO630

## 4.61 NEW GENERAL NOTE ENTITY (TYPE 213)‡

**15-** **CHRHGT** the height of a capital character, typically the character “M”. The character ECO630 height is not changed by applying the slant angle, rotation angle, or character angle. The character height shall be a positive non-zero value. See [Figures 88, 92, and 93](#).

**16-** **CSPACE** inter-character spacing

ECO630

*Fixed width character display fonts:* the distance between the right side of one character and the left side of the next character in the same text string. A negative CSPACE is allowed which permits characters within a string to overlap. The lower limit of a negative CSPACE is the width of a character (CHRWIDn) within the string. See [Figures 88, 89, 92, and 93](#).

*Variable width character display fonts:* a fraction of the standard spacing in the kerning table ECO630 for the font. This value multiplies the standard spacing to obtain the actual spacing. A value of one is the default. Zero is the minimum value and indicates that the characters touch. Overlapping of variable-width fonts is not permitted.

The inter-character spacing is measured before the slant angle, rotation angle, or character angle are applied.

The Inter-character Spacing Property Entity ([Type 406, Form 18](#)) shall not be attached to ECO630 this entity.

**17-** **LSPACE** the distance between the base line of the n-th text string and the base line of the ECO630 previous line of text. LSPACE is only valid for a new line after the first line. It is not valid and must be set to zero for the first sub-string or for sub-strings which are a continuation of an existing line. This value is helpful when consecutive strings are not displayed horizontally or when the first sub-string of the new line is not placed on the baseline. For example, in a dimension with upper and lower tolerances and appended text followed by a second line of appended text, the LSPACE value for the second through fourth strings is meaningless since they are displayed on the same “line,” but the LSPACE value for the fifth string will give the proper distance between the first and fifth string. A negative LSPACE is allowed. See [Figure 87](#).

**18-** **FONT** font display style of the character set. For example, font 18 Helvetica is a font display ECO630 style (FONT), whereas the 1003 is a character set defining different symbols (CHRSET). Some special symbols within a character set may not be affected by the font style.

<b>FONT</b>	<b>Meaning</b>
1	Standard Block
2	Le Roy
3	Futura
6	Comp 80
12	News Gothic
13	Lightline Gothic
14	Simplex Roman
17	Century Schoolbook
18	Helvetica
19	OCR (ISO1073)]

**19-** **CHRANG** angle of the character relative to the base line ( $0.0 \leq \text{CHRANG} \leq 2\pi$ ). This value ECO630 is different from the rotation angle or slant angle. The default value is 0.0. The character angle is applied after the slant angle has been applied. See [Figures 91 and 93](#).

**20-** **CCTEXT** string of control code sequences that are applied to the string of displayed characters. The codes are expressed as pairs of characters which identify what action must be taken

## 4.61 NEW GENERAL NOTE ENTITY (TYPE 213)‡

prior to the display of the text string. The order that the control codes are presented in CC-TEXT shall be preserved when conditions call for imbedding unlike control codes (e.g., boxing, underscore, and overscore). Some control codes can be nested to any level. The corresponding ending control codes shall be defined in all cases.

- 2 2 - WT** the width of a box containing the character string. Box width is established after the rotation angle, slant angle, and character angle have been applied. See [Figures 88, 92, and 93](#). ECO630
- 2 3 - HT** the height of a box containing the character string. Box height is established after the rotation angle, slant angle, and character angle have been applied. [See Figure 93](#). ECO630
- CHRSET -24** the character set defining the string. It is the functional interpretation of the set of symbols. ECO630

Character Set	Meaning
1	Standard ASCII
1001	Symbol Font 1
1002	Symbol Font2
1003	Symbol Font3
2001	Kanji
3001	ISO 8859-1 (Latin-1)‡

- 2 5 - SL** the slant angle of an individual character. Slant angle is in addition to that already predefined in the font. See [Figures 90, 91, 92, and 93](#). ECO630
- 2 6 - A** the rotation angle of the text string. Rotation angle is relative to the positive x axis in construction space and is independent of the text containment angle (TXTAG). See [Figures 90 and 91](#). ECO630

### 4.61.2 Control Codes

Below is a list of currently defined control sequences.

NOTE: The character “Z” is used to indicate the end of a control code type. For example, SU...SZ represent superscript text start and superscript text end. All conditions are implicitly terminated at the end of the final text sub-string.

#### 4.61.2.1 Control Codes Which Cannot Be Nested

**CC** - Change character/font set. This is to indicate to a postprocessor that the only reason that this string is separate is to change character sets. This might be used if a special character is to be used within the context of a fraction or tolerance string.

**BL** - Base Line. The base line is defined as the first string in the note until a “new line” is encountered which redefines the “base line”. If the initial strings are such that they do not define a base line, the string which contains the base line control code can derive its origin from the X and Y start positions. The strings which do not define a base line are all tolerance, fraction, super-script, and subscript control-coded strings.

**NL** - New Line. This condition is used to indicate a new base line is being defined.

**BD** - Bold text display start.

**BZ** - Bold text display end.

**IT** - Italics text display start.

**I<sub>Z</sub>** - Italics text display end.

**T<sub>B</sub>** -Tolerance text, bilateral start.

**TT** -Tolerance, text being tolerated.

**TU** - Tolerance text, upper portion start.

**TL** - Tolerance text, lower portion start.

**TZ** -Tolerance text end (all types).

**US** - Underscore start.

**UZ** - Underscore end.

**OS** - Overscore start.

**OZ** - overscore end.

**ES** - Enclosing Separator. This causes the display of a vertical line relative to the rotation angle ECO630 to the text that extends from the top of the enclosing symbol to the bottom of the enclosing symbol at the point of the text. The ES code is not nested; therefore there is not an ending code for it. See [Figure 87](#).

#### 4.61.2.2 Control Codes Which Can Be Nested

**E n** - Enclosing Symbol start. The value n determines the type of symbol to be used:

ECO630

Value	Meaning
1	Standard Size Box - half character height above and below, half character width on each side
2	Oversized Box - full character height above and below, full character width on each side
3	Undersized Box - no space between characters and lines of the box
4	Bullet Right - box with semi-circle arc for right side
5	Bullet Left - box with semi-circle arc for left side
6	Capsule - box with both ends replaced by arcs
7	Flag Note Start - per Entity <a href="#">Type 208</a> definition
8	Lozenge - box where sides are replaced by < and >

**EZ** - Enclosing Symbol End. Stop the display of the lowest nested enclosing symbol that is currently ON.

**HU** - Horizontally aligned fraction, upper portion start.

**HL** - Horizontally aligned fraction, lower portion start.

**HZ** - Horizontally aligned fraction, upper or lower end.

**VU** - Vertically aligned fraction, upper portion start.

**VL** - Vertically aligned fraction, lower portion start.

#### 4.61 NEW GENERAL NOTE ENTITY (TYPE 213)‡

**VZ** - Vertically aligned fraction, upper or lower end

**DU** - Diagonally aligned fraction, upper portion start.

**DL** - Diagonally aligned fraction, lower portion start.

**DZ** - Diagonally aligned fraction, upper or lower end.

**SU** - Superscript text start.

**SL** - Subscript text start.

**SZ** - superscript or subscript text end.

ECO630

The following control code string and text string values apply to the example in [Figure 86](#), NS = 6: ECO630

Parameter	Value	I Parameter	Value
CCTEXT1	2HTU	TEXT1	5H+1.00
CCTEXT2	4HTLTZ	TEXT2	5H-1.00
WTEXT3	4HTTZ	TEXT3	4H5.00
CCTEXT4	4HTZNL	TEXT4	13HTOLERANCE AND
CCTEXT5	2HNL	TEXT5	8HCENTERED
CCTEXT6	2HNL	TEXT6	4HTEXT

The following control code string and text string values apply to the example in [Figure 87](#), NS = 6:

Parameter	Value	Parameter	Value
	2 H T T	TEXT1	3HAAA
CCTEXT2	2 H T U	TEXT2	3 H 5 6 7
CCTEXT3	4HTLTZ	TEXT3	4H1234
CCTEXT4	4HNL6TT	TEXT4	3HCCC
CCTEXT5	2 H T U	TEXT5	2 H 5 6
CCTEXT6	4HTLTZ	TEXT6	2 H 7 8

## 4.61 NEW GENERAL NOTE ENTITY (TYPE 213)‡

### Directory Entry

ECO630

(1) Entity Type Number 213	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern 1	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????01**	(10) Sequence Number D #
(11) Entity Type Number 213	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

ECO650  
ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	TXTCW	Real	Width of text containment area of all strings in the note
2	TXTCH	Real	Height of text containment area of all strings in the note
3	JUSTCD	Integer	Justification code of all strings within the note: 0 = no justification 1 = right justified 2 = center justified 3 = left justified
4	TXTCX	Real	Text containment area location point X
5	TXTCY	Real	Text containment area location point Y
6	TXTCZ	Real	Z depth from TXTCX,TXTCY plane
7	TXTAG	Real	Rotation Angle of text containment area in radians
8	BASELX	Real	Position of first base Line
9	BASELY	Real	Position of first base Line
10	BASELZ	Real	Z depth from BASELX,BASELY plane
11	NILS	Real	Normal Interline spacing
12	NS	Integer	Number of Text Strings
13	FIXVAR(1)	Integer	Fixed/Variable width character display: 0 = Fixed 1 = Variable
14	CHRWID(1)	Real	Character Width
15	CHRHGT(1)	Real	Character Height
16	CSPACE(1)	Real	Inter-character spacing
17	LSPACE(1)	Real	Interline spacing
18	FONT(1)	Integer	Font style
19	CHRANG (1)	Real	Character Angle
20	CCTEXT(1)	String	Control Code String
21	NC(1)	Integer	Number of characters in the first string (TEXT(1)) or zero. The number of characters (NC(n)) must always be equal to the character count of its corresponding text string (TEXT(n))
22	WT(1)	Real	Box Width
23	HT(1)	Real	Box Height
24	CHRSET(1)	Integer or Pointer	Character Set Interpretation (default = 1):
25	SL(1)	Real	Slant angle of TEXT(1) in radians ( $\pi/2$ is the value for no slant angle and is the default)
26	A(1)	Real	Rotation angle in radians for TEXT(1)

#### **4.61 NEW GENERAL NOTE ENTITY (TYPE 213)‡**

27	M(1)	Integer	Mirror Flag: 0 = no mirroring 1 = mirror axis is perpendicular to text base line 2 = mirror axis is text base line
28	VH(1)	Integer	Rotate internal text flag: 0 = text horizontal 1 = text vertical
29	XS(1)	Real	Text start point
30	YS(1)	Real	Text start point
31	ZS(1)	Real	Z depth from XT,YT plane
32	TEXT(1)	String	First text string
:	:	:	
20*NS-7	FIXVAR(NS)lnteger		Fixed/Variable width character display
:	:	:	
20*NS+12	TEXT(NS)	String	Last text string

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### 4.61 NEW GENERAL NOTE ENTITY (TYPE 213)‡

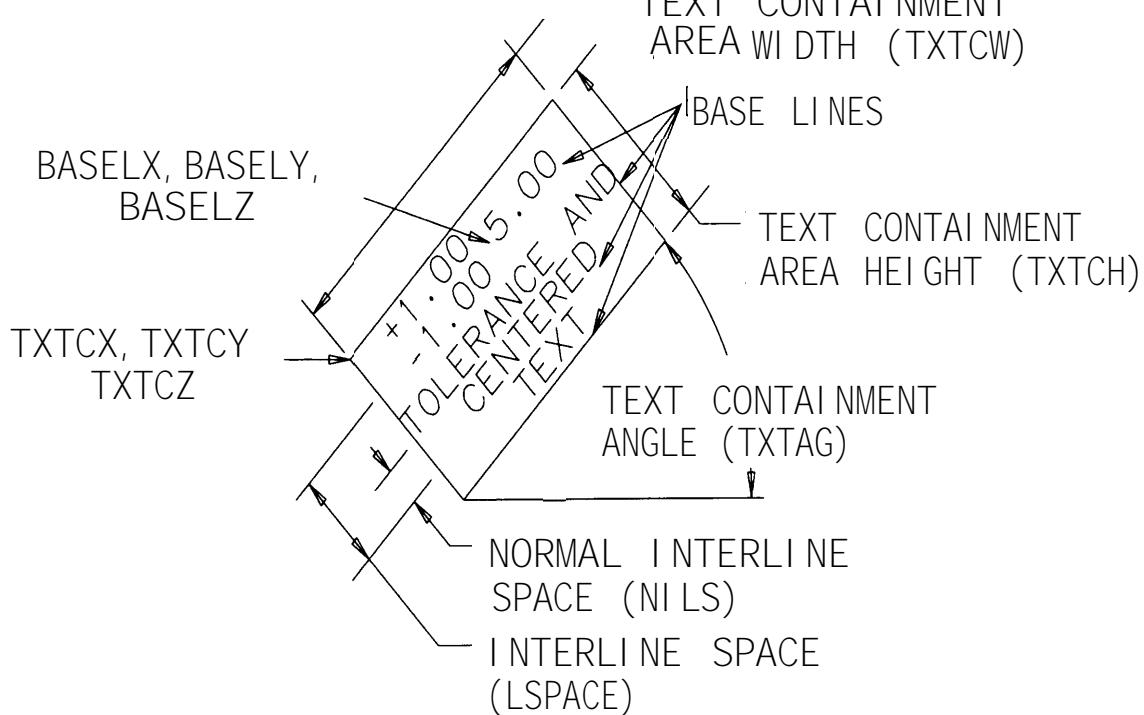


Figure 86. Text Containment Area (see text for CCTEXTn and TEXTn values)

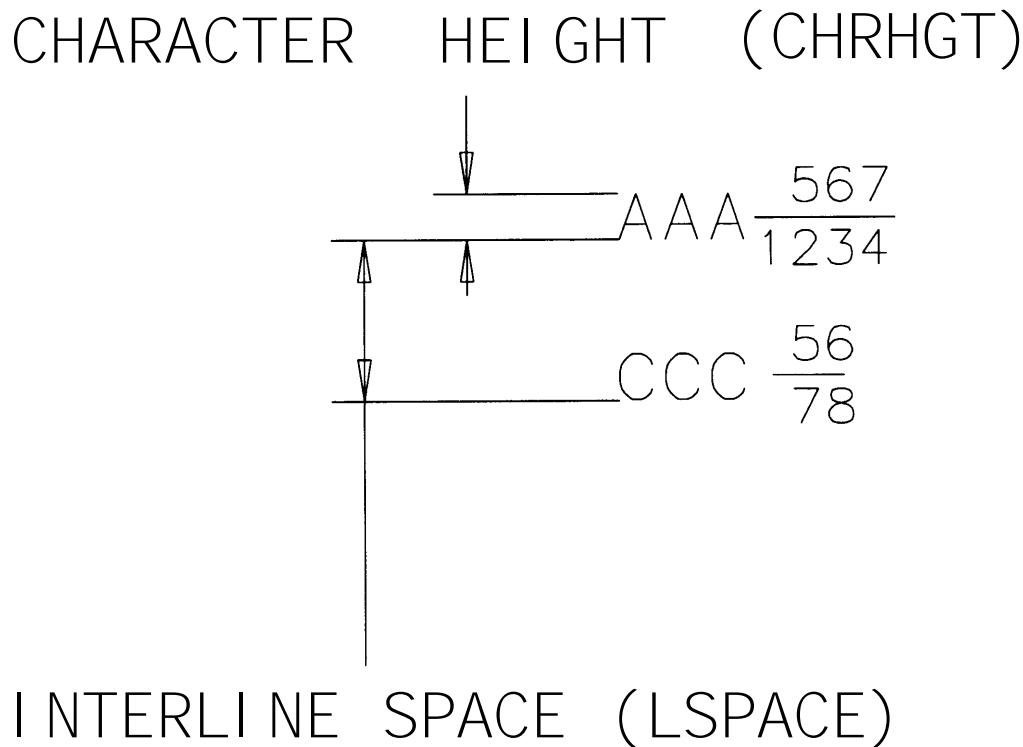


Figure 87. Character Height, Inter-line Spacing (see text for CCTEXTn and TEXTn values)

#### 4.61 NEW GENERAL NOTE ENTITY (TYPE 213)‡

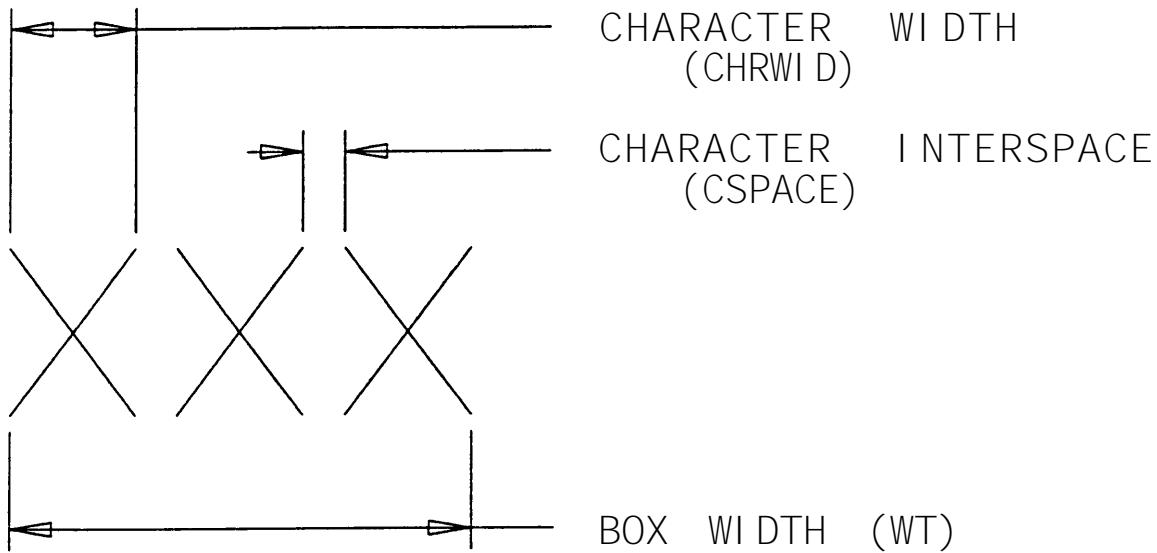


Figure 88. Character Width, Inter-space, Box Width

ZERO CHAR INTERSPACE

NEGATIVE CHAR INTERSPACE OF -.25

Figure 89. Examples of Fixed Width Character Inter-space

#### 4.61 NEW GENERAL NOTE ENTITY(TYPE 213)‡

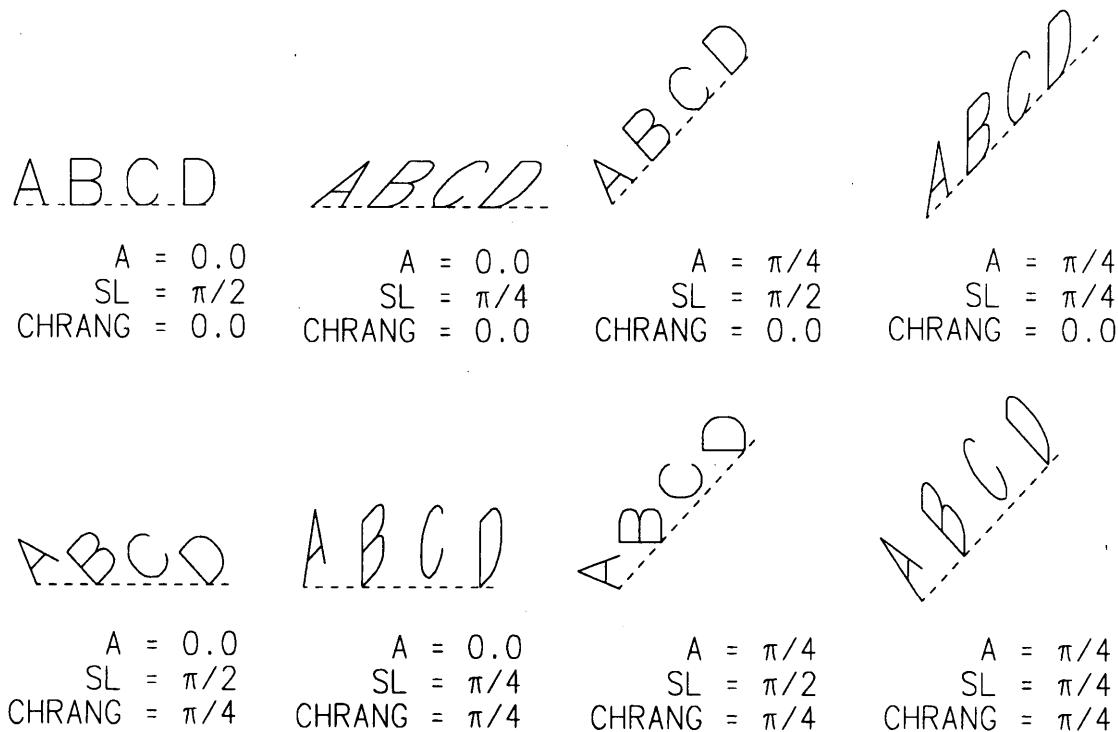


Figure 90. Rotation, Slant and Character Angle

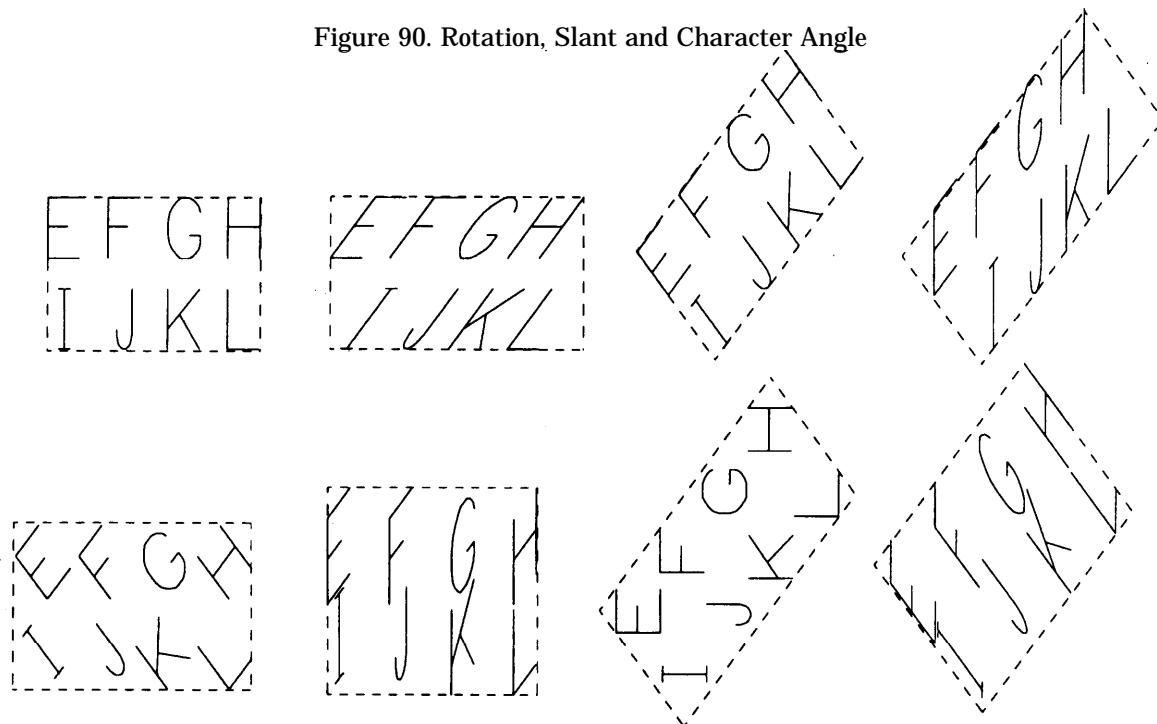


Figure 91. Text Containment Area

#### 4.61 NEW GENERAL NOTE ENTITY (TYPE 213)‡

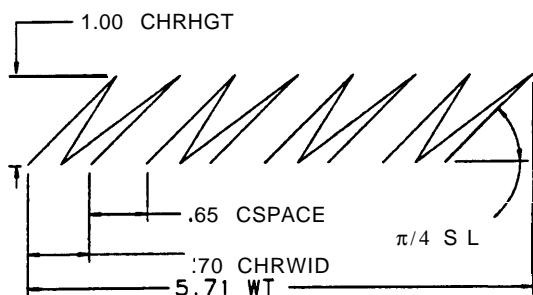
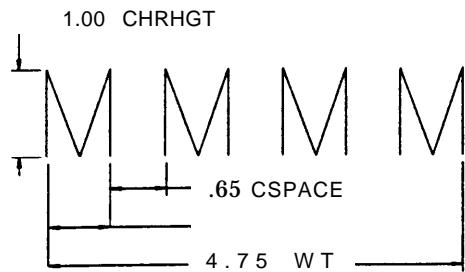


Figure 92. Character Height, Width, Inter-space, Box Width

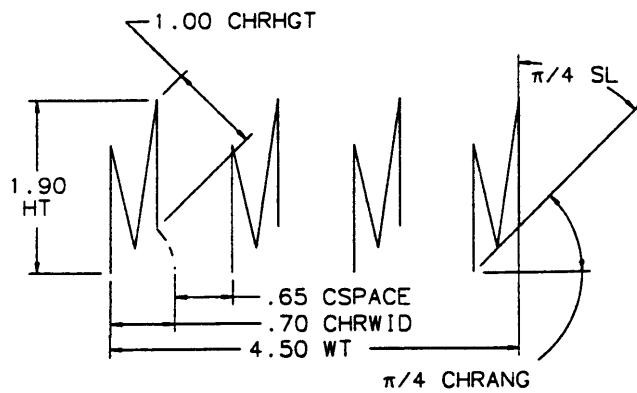
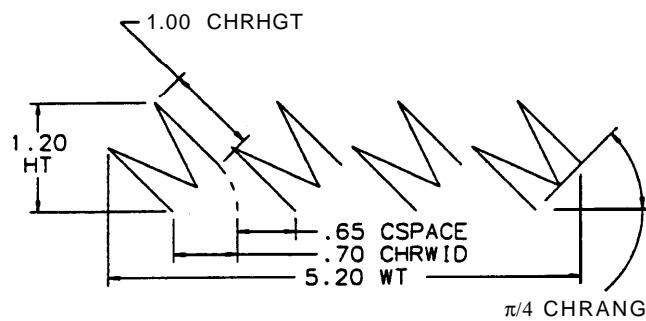


Figure 93. Character Height, Width, Inter-space, Box Width

**4.62 Leader (Arrow) Entity (Type 214)**

A Leader (Arrow) Entity consists of one or more line segments, except when the leader is part of an angular dimension (see Section 4.55). The first segment begins with an arrowhead. Remaining segments successively link to a presumed text item. 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 94. Examples of arrowheads for leaders are shown in Figure 96; the intersections of the vertical and horizontal lines define each arrow's head-point.

In the use of the angular, diameter, and linear dimension entities, 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 95.

Some leaders (e.g., the leader involved with the radius dimension in Figure 95) give the appearance of locating an arrow interior to a segment. 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 retraces the first in the opposite direction and extends it. Leaders of this type for other types of dimensions are constructed similarly. For the angular dimension entity, the first two segments are arcs.

For the Leader Entity, the Form Numbers are as follows (see Figure 96):

Form	Meaning
1	Wedge
2	Triangle
3	Filled Triangle
4	No Arrowhead
5	Circle
6	Filled Circle
7	Rectangle
8	Filled Rectangle
9	Slash
10	Integral Sign
11	Open Triangle
12	Dimension Origin

**Definitions.** The following definitions and abbreviations are used in the entity description.

ECO628

**Leader.** The line (or curve) extending from the arrowhead coordinate to the first segment tail coordinate.

**AD1.** The overall arrowhead height as measured parallel to the leader.

**AD2.** The overall arrowhead width as measured perpendicular to the leader. For the circular arrowhead styles (Forms 5, 6, and 12), AD1 and AD2 shall be greater than zero and equal. For the "no arrowhead" style (Form 4), AD1 and AD2 shall be zero. For all other styles, AD1 and AD2 shall be greater than zero.

**Wedge (Form 1).** The arrowhead is depicted as two line segments which form a "V".

The vertex of the "V" lies on the arrowhead coordinate with the open end extending over the leader.

**Triangle (Form 2).** The arrowhead is depicted as three line segments which form a triangle. The vertex of the triangle lies on the arrowhead coordinate and the portion of the leader within the triangle is displayed.

**Filled Triangle (Form 3).** The arrowhead is depicted as three line segments which form a triangle. The vertex of the triangle lies on the arrowhead coordinate and the interior of the triangle is shaded.

**No Arrowhead (Form 4).** The arrowhead does not appear to be depicted because AD1 and AD2 are zero.

**Circle (Form 5).** The arrowhead is depicted as a circle. The edge of the circle lies on the arrowhead coordinate and the center of the circle lies on the leader. The portion of the leader within the circle is not displayed. AD1 and AD2 shall be greater than zero and equal. ECO630

**Filled Circle (Form 6).** The arrowhead is depicted as a circle. The edge of the circle lies on the arrowhead coordinate and the center of the circle lies on the leader. The interior of the circle is shaded. AD1 and AD2 shall be greater than zero and equal. ECO630

**Rectangle (Form 7).** The arrowhead is depicted as four line segments which form a rectangle. One edge of the rectangle lies centered on the arrowhead coordinate and the center of the rectangle lies on the leader. The portion of the leader within the rectangle is not displayed.

**Filled Rectangle (Form 8).** The arrowhead is depicted as four line segments which form a rectangle. One edge of the rectangle lies centered on the arrowhead coordinate and the center of the rectangle lies on the leader. The interior of the rectangle is shaded.

**Slash (Form 9).** The arrowhead is depicted as a line segment which is the diagonal of a rectangle defined by AD1 and AD2 lying centered on the arrowhead coordinate.

**Integral Sign (Form 10).** The arrowhead is depicted as a curved segment in an elongated "S" which lies centered on the arrowhead coordinate. It fits within a rectangle defined by AD1 and AD2 lying centered on the arrowhead coordinate.

**Open Triangle (Form 11).** The arrowhead is depicted as three line segments which form a triangle. The vertex of the triangle lies on the arrowhead coordinate and the portion of the leader within the triangle is not displayed.

**Dimension Origin (Form 12).** The arrowhead is depicted as a circle. The center of the circle lies on the arrowhead coordinate and the portion of the leader within the circle is displayed. AD1 and AD2 shall be greater than zero and equal. ECO630

## 4.62 LEADER (ARROW) ENTITY (TYPE 214)

### Directory Entry

(1) Entity Type Number 214	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????01**	(10) Sequence Number D #
(11) Entity Type Number 214	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 1-12	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

ECO650

### Parameter Data

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	N	Integer	Number of segments
2	AD1	Real	Arrowhead height
3	AD2	Real	Arrowhead width
4	ZT	Real	Z depth
5	XH	Real	Arrowhead coordinates
6	YH	Real	
7	X(1)	Real	First segment tail coordinate pair
8	Y(1)	Real	
:	:		
5 + 2 * N	X(N)	Real	Last segment tail coordinate pair
6 + 2 * N	Y(N)	Real	

Additional pointers as required (see Section 2.2.4.5.2).

#### 4.62 LEADER (ARROW) ENTITY (TYPE 214)

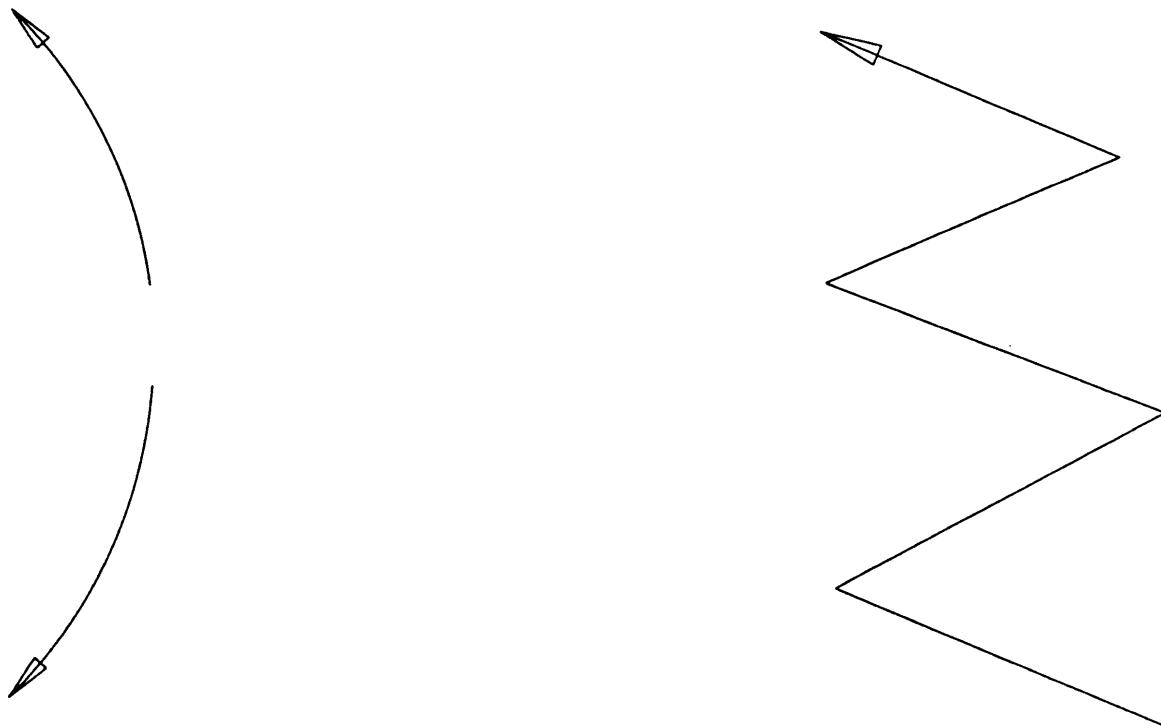
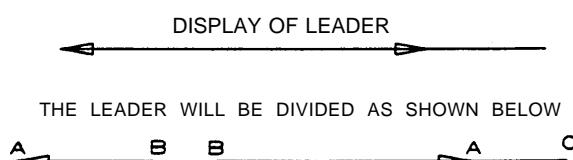
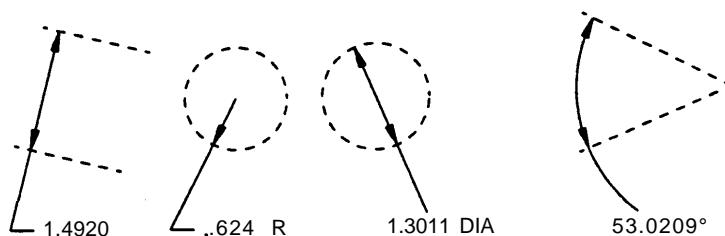


Figure 94. Examples Defined Using the Leader Entity



- 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)

Figure 95. Structure of Leaders Internal to a Dimension

#### 4.62 LEADER (ARROW) ENTITY (TYPE 214)

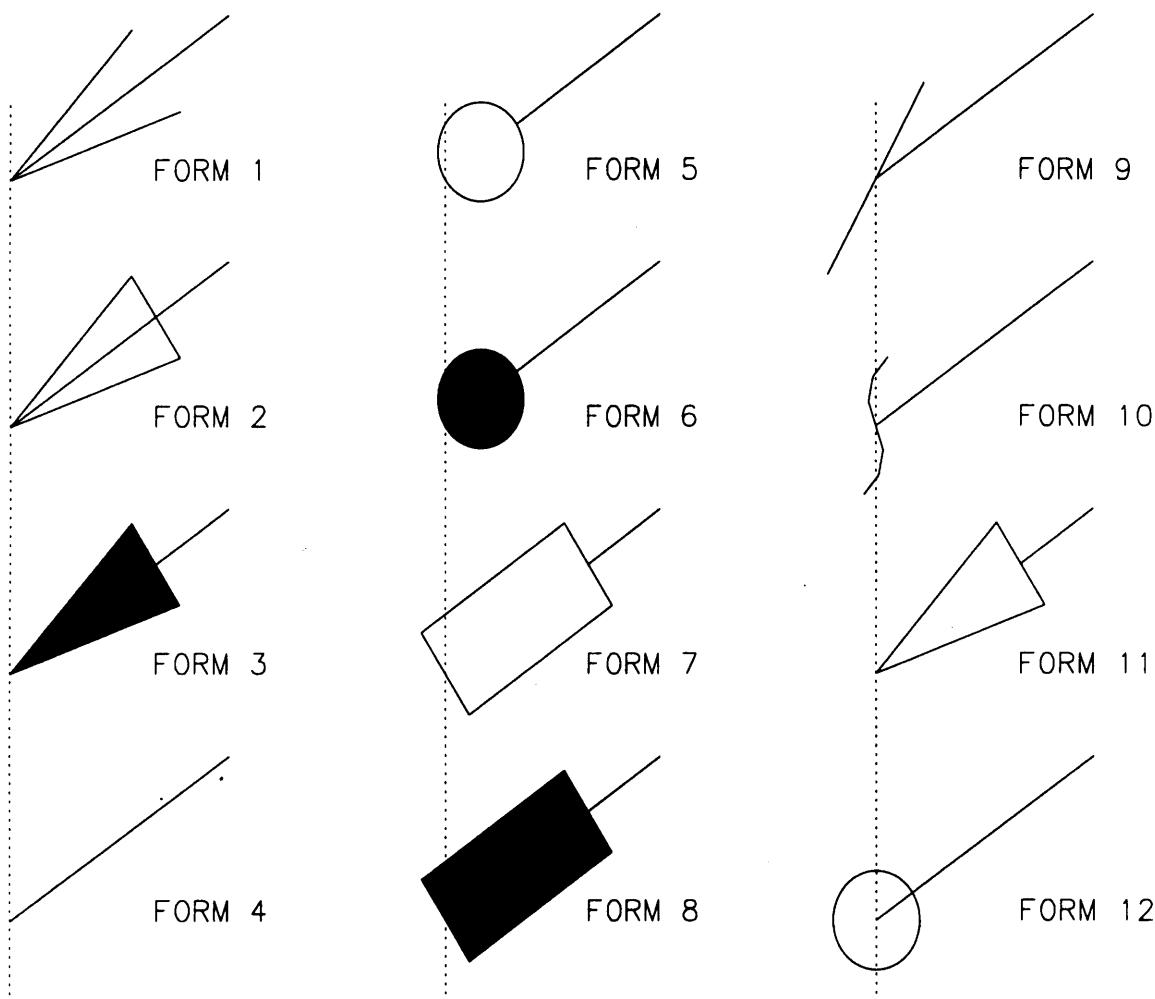
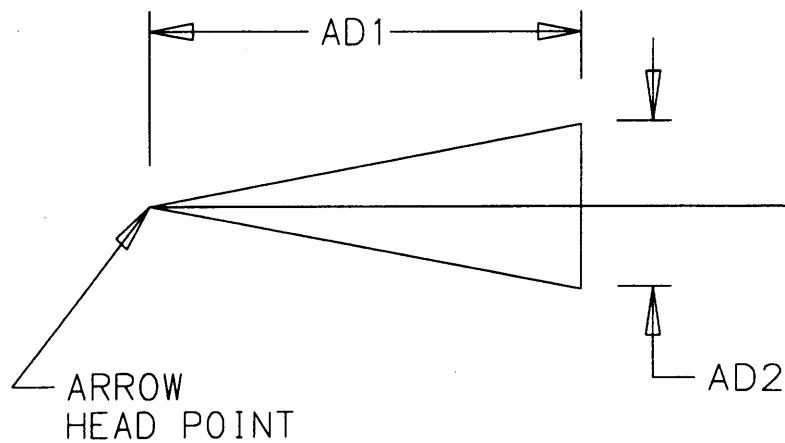


Figure 96. F214X.IGS Definition of Arrowhead Types for the Leader (Arrow) Entity

## 4.63 LINEAR DIMENSION ENTITY (TYPE 216)

### 4.63 Linear Dimension Entity (Type 216)

A Linear Dimension Entity consists of a general note; two leaders; and zero, one, or two witness lines. Refer to [Figure 97](#) for examples of linear dimensions.

For the Linear Dimension Entity, the Form numbers are defined below; examples are shown in [Figure 98](#).

<b>Form</b>	<b>Meaning</b>
0	Linear dimension of undetermined form
1 ‡	Linear dimension of diameter form
2 ‡	Linear dimension of radius form

‡Form Numbers 1 and 2 of the Linear Dimension Entity have not been tested. See [Section 1.9](#).

See [Section 3.5.3](#) for coplanarity requirements for dimension entities.

ECO635

### Directory Entry

(1) Entity Type Number 216	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ????01??	(10) Sequence Number D #
(11) Entity Type Number 216	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0-2	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

### Parameter Data

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	DENOTE	Pointer	Pointer to the DE of the General Note Entity
2	DEARRW1	Pointer	Pointer to the DE of the first Leader Entity
3	DEARRW2	Pointer	Pointer to the DE of the second Leader Entity
4	DEWIT1	Pointer	Pointer to the DE of the first Witness Line Entity, or zero if not defined
5	DEWIT2	Pointer	Pointer to the DE of the second Witness Line Entity, or zero if not defined

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### 4.63 LINEAR DIMENSION ENTITY (TYPE 216)

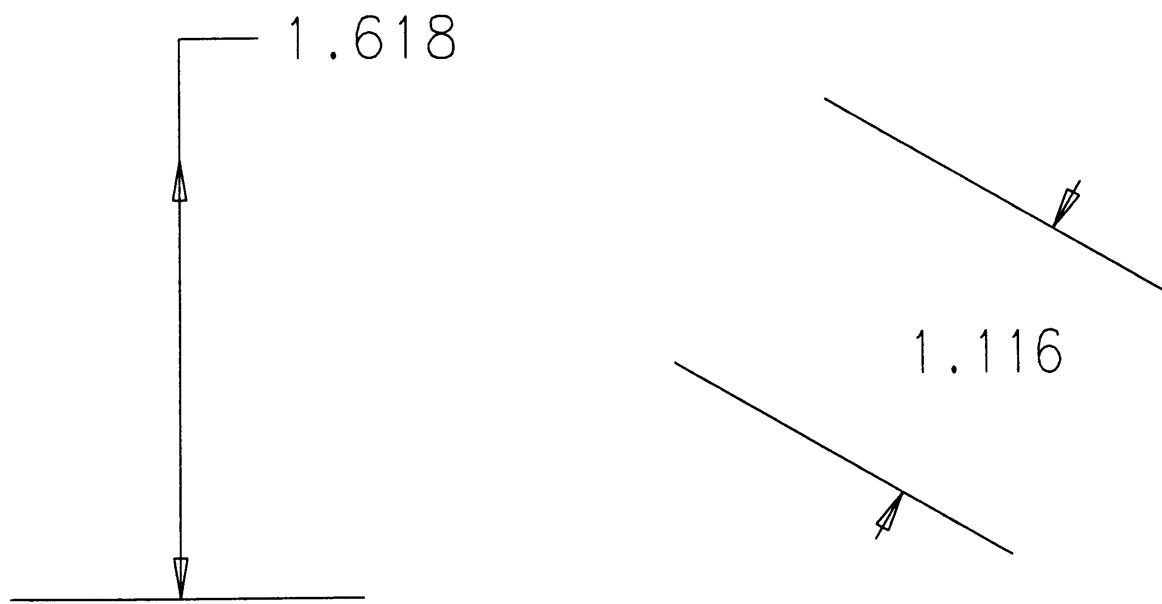


Figure 97. F216X.IGS Examples Defined Using Form 0 of the Linear Dimension Entity

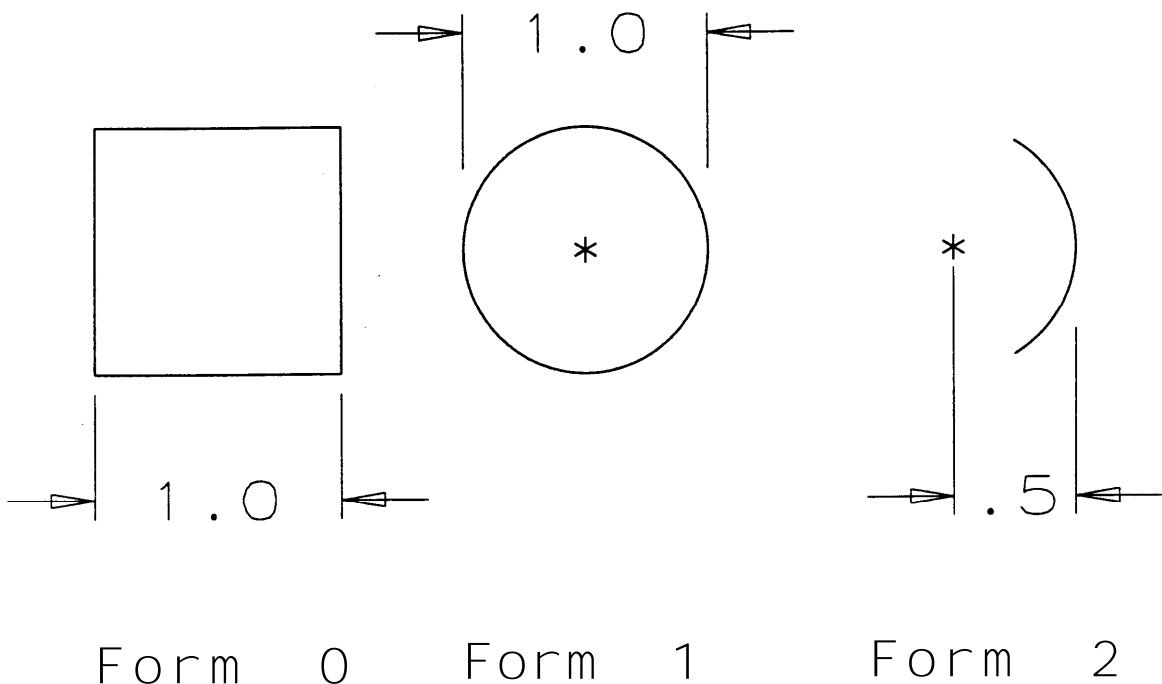


Figure 98. F21601X.IGS Examples of Linear Dimension Forms†

ECO630

## **4.64 ORDINATE DIMENSION ENTITY (TYPE 218)**

### **4.64 Ordinate Dimension Entity (Type 218)**

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

An Ordinate Dimension Entity consists of a general note and a witness line or leader. The values ECO630 stored are pointers to the Directory Entry for the associated General Note and Witness Line or Leader Entities. Examples of ordinate dimensions are shown in [Figure 99](#).

For the Ordinate Dimension Entity, the Form Numbers are as follows:

ECO630

<b>Form</b>	<b>Meaning</b>
0	Simple ordinate dimension
1‡	Ordinate dimension with supplemental leader

‡Form Number 1 of the Ordinate Dimension Entity has not been tested. [See Section 1.9](#).

Form 1 of the Ordinate Dimension Entity allows for both a witness and leader line and a supplemental leader as shown in the example in [Figure 100](#). The entity referenced by DEORD defines the ordinate position being dimensioned. The entity referenced by DESUPP is supplemental to those systems that may support it.

[See Section 3.5.3](#) for coplanarity requirements for dimension entities.

ECO635

## 4.64 ORDINATE DIMENSION ENTITY (TYPE 218)

### Directory Entry

(1) Entity Type Number 218	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????01??	(10) Sequence Number D #
(11) Entity Type Number 218	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

### Simple form of the Ordinate Dimension Entity

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DENOTE	Pointer	Pointer to the DE of the General Note Entity
2	DEWIT	Pointer	Pointer to the DE of the Witness Line Entity or Leader Entity

Additional pointers as required (see Section 2.2.4.5.2).

Ordinate Dimension Entity with Supplemental Leader

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DENOTE	Pointer	Pointer to the DE of the General Note Entity
2	DEORD	Pointer	Pointer to the DE of the Witness Line Entity
3	DESUPP	Pointer	Pointer to the DE of the Leader (Arrow) Entity

Additional pointers required (see Section 2.2.4.5.2)

**4.64 ORDINATE DIMENSION ENTITY (TYPE 218)**

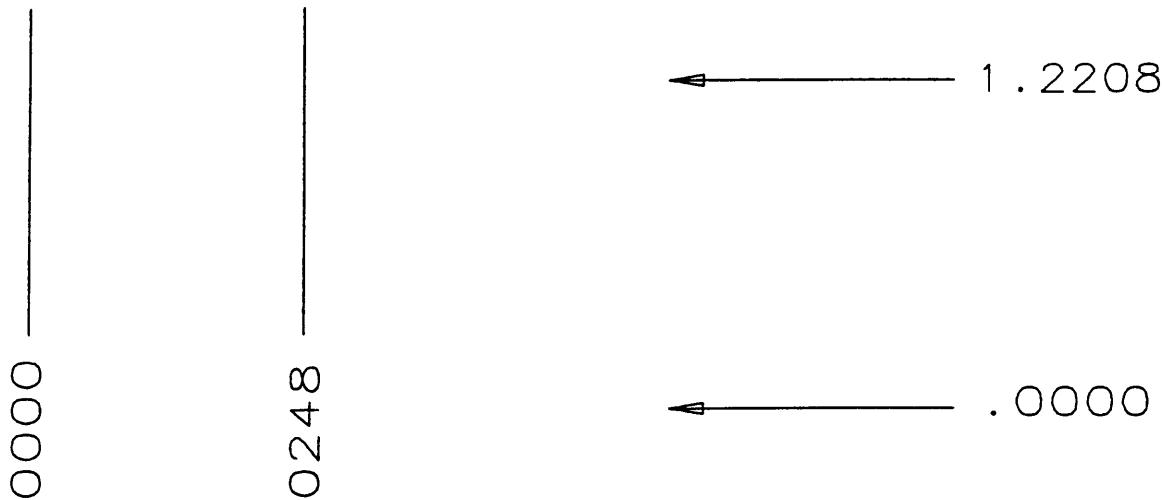


Figure 99. F218X.IGS Examples Defined Using the Ordinate Dimension Entity

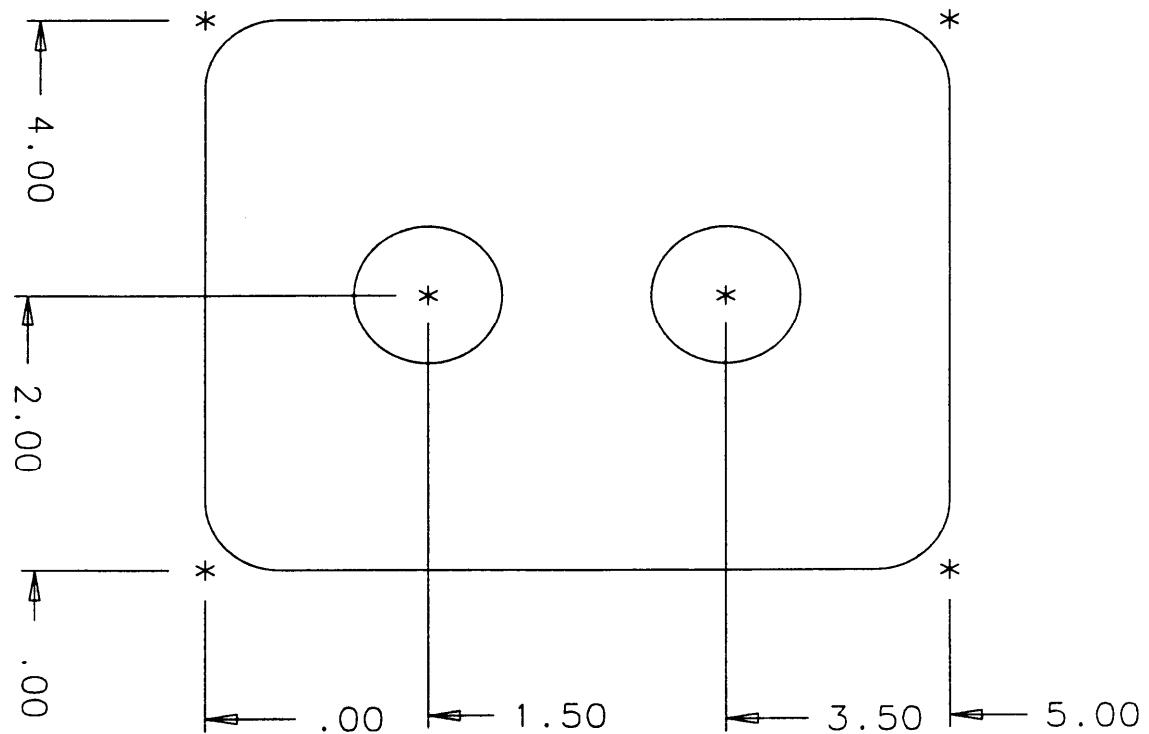


Figure 100. F21801X. IGS Example Defined Using Form 1 of the Ordinate Dimension Entity‡

### 4.65 Point Dimension Entity (Type 220)

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

The leader shall contain three segments. Its first and last segments shall be horizontal or vertical. If ECO630 a hexagon encloses the text, it shall be described by either a Composite Curve Entity ([Type 102](#)) or a Simple Closed Planar Curve Entity ([Type 106, Form 63](#)). If a circle or hexagon does not enclose the text, the last segment of the leader shall be horizontal and it shall underline the text.

Examples are shown in [Figure 101](#).

See [Section 3.5.3](#) for coplanarity requirements for dimension entities.

ECO635

#### Directory Entry

(1) Entity Type Number 220	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????01??	(10) Sequence Number D #
(11) Entity Type Number 220	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count , #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DENOTE	Pointer	Pointer to the DE of the General Note Entity
2	DEARRW	Pointer	Pointer to the DE of the Leader Entity
3	DEGEOM	Pointer	Pointer to the DE of the Circular Arc Entity, Composite Curve Entity, or Simple Closed Planar Curve Entity, or zero

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.65 POINT DIMENSION ENTITY (TYPE 220)**

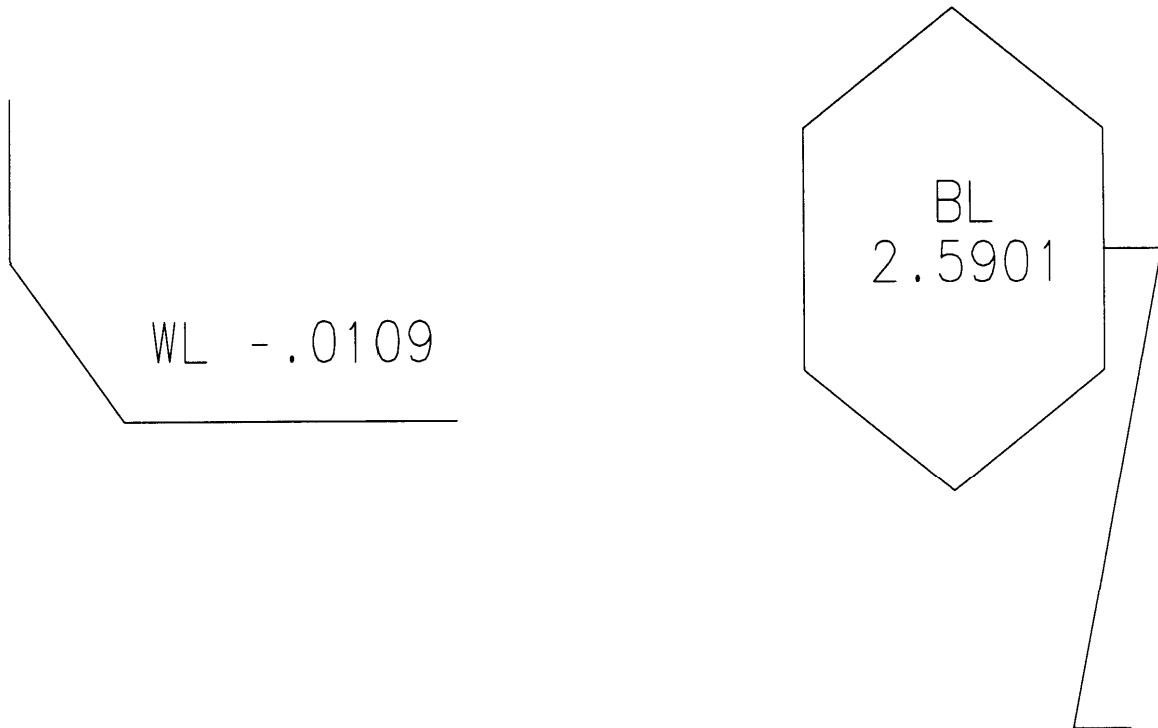


Figure 101. Examples Defined Using the Point Dimension Entity

## 4.66 RADIUS DIMENSION ENTITY (TYPE 222)

### 4.66 Radius Dimension Entity (Type 222)

A Radius Dimension Entity consists of a general note, a leader, and an arc center point, (XT, YT). ECO630 Refer to [Figure 102](#) for examples of radius dimensions.

The arc center is used as a reference in constructing the radius dimension but has no effect on the dimension components. ECO630

For the Radius Dimension Entity, the Form Numbers are as follows:

Form	Meaning
0	Simple radius dimension
1	Radius dimension with two leaders

Form 1 of the Radius Dimension Entity addresses the occasional need to have two Leader (Arrow) Entities referenced. In case of this form of the Radius Dimension Entity, the DEARRW2 pointer shall only reference a Leader Entity ([Type 214, Form 4](#)). An example is shown in [Figure 103](#). ECO634 ECO630

See [Section 3.5.3](#) for coplanarity requirements for dimension entities. ECO635

## 4.66 RADIUS DIMENSION ENTITY (TYPE 222)

### Directory Entry

(1) Entity Type Number 222	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????01??	(10) Sequence Number D #
(11) Entity Type Number 222	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

### Simple form of the Radius Dimension Entity

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DENOTE	Pointer	Pointer to the DE of the General Note Entity
2	DEARRW	Pointer	Pointer to the DE of the Leader Entity
3	XT	Real	Arc center coordinates
4	YT	Real	

Additional pointers as required (see Section 2.2.4.5.2).

### Radius Dimension Entity with two leaders

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DENOTE	Pointer	Pointer to the DE of the General Note Entity
2	DEARRW1	Pointer	Pointer to the DE of the first Leader (Arrow) Entity; the arrow head should touch the arc
3	XT	Real	Arc center coordinates
4	YT	Real	
5	DEARRW2	Pointer	Pointer to the DE of the second Leader (Arrow) Entity, or zero

Additional pointers as required (see Section 2.2.4.5.2).

**4.66 RADIUS DIMENSION ENTITY (TYPE 222)**

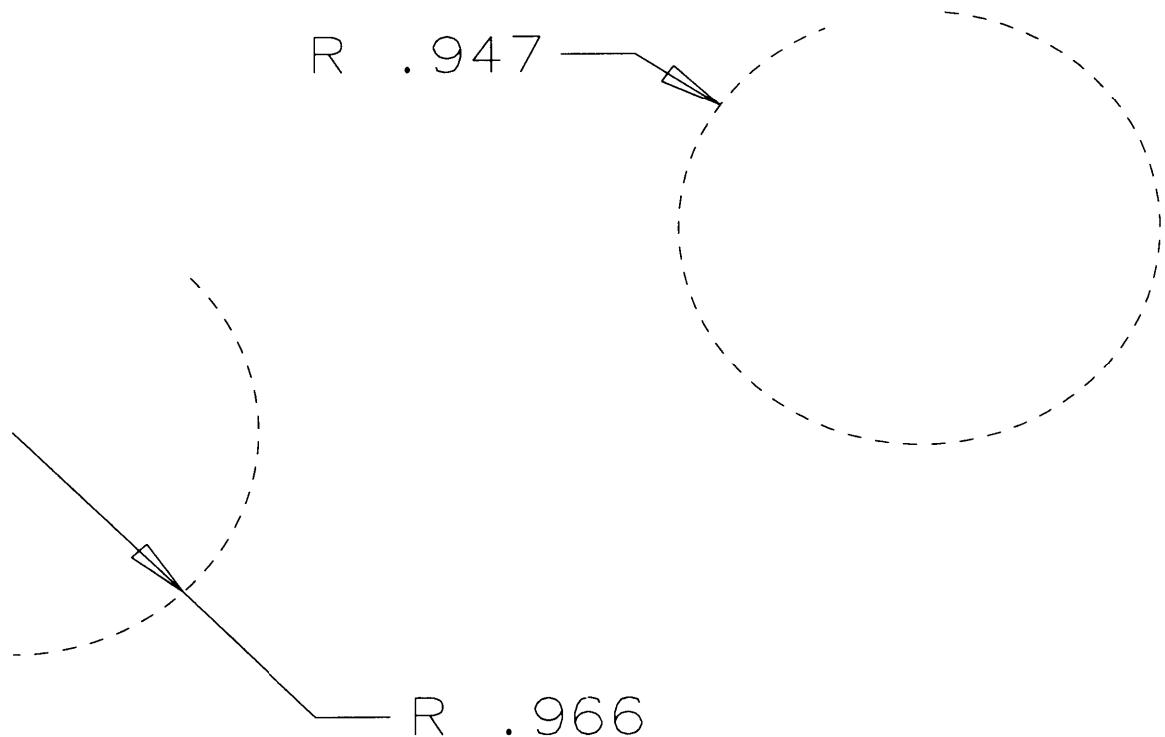


Figure 102. F222X.IGS Examples Defined Using the Radius Dimension Entity

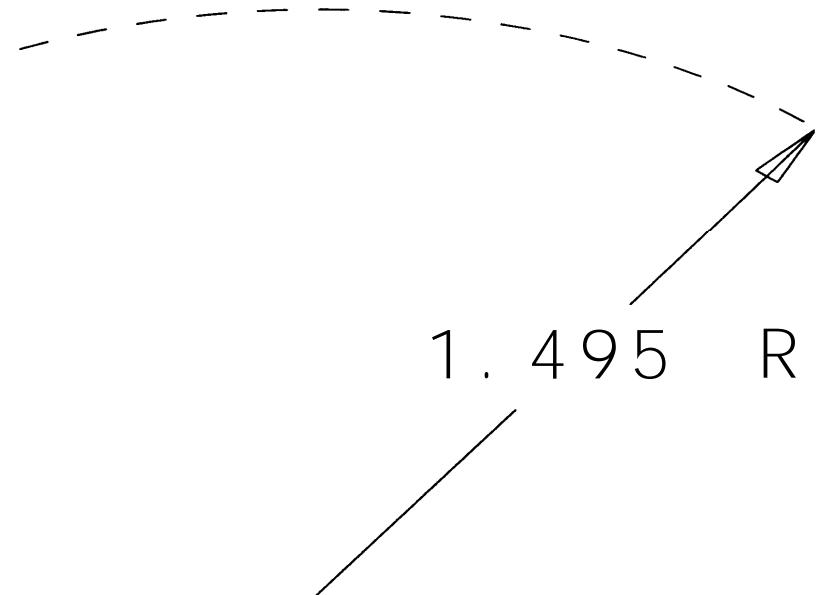


Figure 103. F22201X.IGS Example Defined Using Form 1 of the Radius Dimension Entity

### 4.67 General Symbol Entity (Type 228)

A General Symbol Entity is composed of zero or one general notes, zero or more associated leaders, ECO630 and one or more geometry entities which define a symbol. Examples of general symbols are shown in [Figure 104](#).

Any geometry entity used to create the symbol shall have a Subordinate Entity Switch of 01 and an Entity Use Flag of 01 in Field 9 of its Directory Entry Section.

For this entity, the form number is used to maintain the nature of the symbol on the sending system. ECO630 Note that each of the examples in [Figure 104](#) could be represented as Form 0 For the General Symbol Entity, the Form Numbers are as follows:

Form	Meaning (see [ANSI82])
0	General Symbol - Originated as a symbol which was not necessarily a standard symbol.
1‡	Datum Feature Symbol - the included data originated as a datum feature symbol consisting of a frame containing the datum identifying letter preceded and followed by a dash. The identifying letter is a letter of the alphabet (except I, O and Q). Where datum features are so numerous as to exhaust the single alpha series, the double alpha series is used - AA through AZ, BA through BZ, etc.
2‡	Datum Target Symbol - The included data originated as a datum target symbol consisting of a circle divided horizontally into two halves. The lower half contains a letter identifying the associated datum, followed by the target number assigned sequentially starting with one for each datum. Where the target is an area, the area size may be entered into the upper half of the symbol; otherwise, the upper half is blank. A radial line attached to the symbol is directed to the target point, line, or area, as applicable.
3‡	Feature Control name - The included data originated as a feature control frame consisting of a frame divided into compartments containing the geometric characteristic symbol followed by the tolerance. The tolerance may be preceded by a diameter symbol or followed by a material condition symbol.
5001-9999‡	Implementor - Defined

‡Forms 1, 2, 3, and 5001-9999 of the General Symbol Entity have not been tested. See [Section 1.9](#).

For Forms 1, 2, and 3, the general note is mandatory.

ECO630

Forms 1 (Datum Feature Symbol) and 3 (Feature Control Frame) may be related to the considered feature(s) by the Witness Line Entity ([Type 106, Form 40](#)) or by the Leader Entity ([Type 214](#)). See Section 3.5(b) or (c) of [ANSI82] for more information.

Form numbers in the range 5001-9999 are reserved to allow for implementor-defined meaning. Implementor-defined forms shall conform to the parameter requirements of the General Symbol Entity ([Type 228](#)), and are to be interpreted like those defined as Form 0 when the implementor-defined meaning is not understood.

**Directory Entry**

(1) Entity Type Number 228	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????01??	(10) Sequence Number D #
(11) Entity Type Number 228	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number #	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

Note: Valid values of the Form Number are 0-3 and 5001-9999.

**Simple General Symbol Entity (Form 0)****Parameter Data**

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DENOTE	Pointer	Pointer to the DE of the associated General Note Entity or zero (for Form 0 only)
2	N	Integer	Number of pointers to geometry
3	DEGEOM(1)	Pointer	Pointer to the DE of the first defining geometry entity
:	:	:	
2+N	DEGEOM(N)	Pointer	Pointer to the DE of the last defining geometry entity
3+N	L	Integer	Number of Leaders or zero
4+N	DEARRW(1)	Pointer	Pointer to the DE of the first associated Leader Entity
:	:	:	
3+L+N	DEARRW(L)	Pointer	Pointer to the DE of the last associated Leader Entity

Additional pointers as required (see Section 2.2.4.5.2).

Specific and Implementor-Defined Forms of the General Symbol Entity‡

**Parameter Data**

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DENOTE	Pointer	Pointer to the DE of the associated General Note Entity
2	N	Integer	Number of pointers to geometry
3	DEGEOM(1)	Pointer	Pointer to the DE of the first defining geometry entity
:	:	:	
2+N	DEGEOM(N)	Pointer	Pointer to the DE of the last defining geometry entity
3+N	L	Integer	Number of Leaders or zero
4+N	DEARRW(1)	Pointer	Pointer to the DE of the first associated Leader Entity
:	:	:	
3+L+N	DEARRW(L)	Pointer	Pointer to the DE of the last associated Leader Entity

Additional pointers as required (see Section 2.2.4.5.2).

**4.67 GENERAL SYMBOL ENTITY (TYPE 228)**

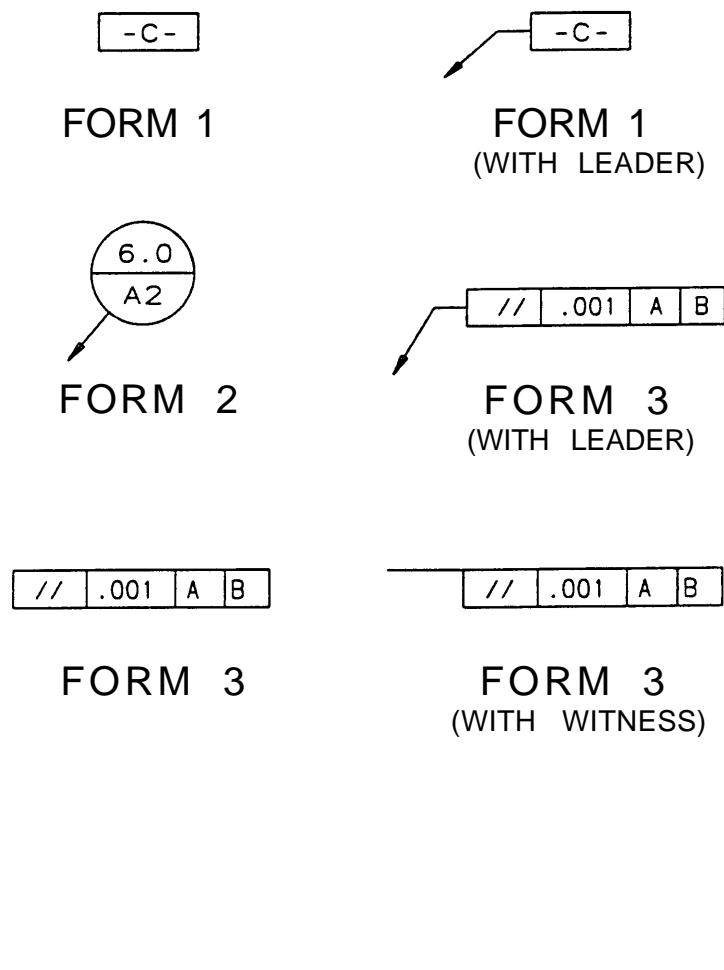


Figure 104. Examples of Symbols Defined Using the General Symbol Entity

#### 4.68 Sectioned Area Entity (Type 230)

A sectioned area is a portion of a design which is to be filled with a pattern of lines. Ordinarily ECO630 this entity is used to reveal or expose shape or material characteristics defined by other entities. The Sectioned Area Entity consists of a pointer to an exterior definition curve, a specification of the pattern of lines, the coordinates of a point on a pattern line, the distance between the pattern lines, the angle between the pattern lines and the X-axis of the definition space, and the specification of any enclosed definition curves (commonly known as islands). This entity is commonly used for annotative purposes but may be considered geometry if all definition curves are also part of the model geometry.

For the Sectioned Area Entity, the Form Numbers are as follows:

ECO630

Form	Meaning
0	Standard Crosshatching
1‡	Inverted Crosshatching

‡Form Number 1 of the Sectioned Area Entity has not been tested. See Section 1.9.

**Form 0 (Standard crosshatching)** used when the main boundary contains section lines. If there are nested island curves (*i.e.*,  $N > 0$  section lines change state for each nested interior boundary).

**Form 1 (Inverted crosshatching)** used when the main boundary *does not* contain section lines, or when there is no main boundary defined. At least one island curve shall be defined; *i.e.*,  $N > 0$  is required. Unnested island curves contain section lines; if any island curves are nested, section lines change state for each nested interior boundary.

Refer to [Figure 110](#) for examples of both standard and inverted crosshatching.

A definition curve is a simple closed curve that defines an area on the plane. The list of curve entity types and form numbers that may be used is given below:

Type	Name
100	Circular Arc (full circle only)
102	Composite Curve
104/1	Conic Arc (full ellipse only)
106/63	Simple Closed Planar Curve
112	Parametric Spline Curve
126	Rational B-Spline Curve

In cases where the definition curve duplicates or projects model geometry into the definition space solely for the purpose of defining the sectioned area, the definition curve shall be flagged as an annotation entity and physically subordinate to the Sectioned Area Entity. In cases where the definition curve is the model geometry, the definition curves shall remain as geometric entities and are not physically subordinate to the Sectioned Area Entity.

The XT and YT coordinates, which may be specified, indicate a location which is on one of the pattern lines. This point allows applications which require specific placements of the lines to constrain them appropriately. This point explicitly defines a point interior to the filled portion of the Sectioned Area which may be used as an anchor point for those systems which produce explicit ray

#### **4.68 SECTIONED AREA ENTITY (TYPE 230)**

traces to create the fill pattern. If not specified, *i.e.*, indicated by default, the lines need only be within the exterior definition curve.

The angle of the lines has a default value of  $\pi/4$ , measured in radians.

The fill pattern is specified by a fill pattern code according to predefined definitions illustrated in [Figure 105](#). Where possible, the intention of the pattern is shown in Table 10.

Table 10. Predefined Fill Patterns for the Sectioned Area Entity

PATRN	Description
0	(no fill pattern specified)
1	Iron, brick, stone masonry
2	steel
3	(no intended meaning)
4	Rubber, plastic, electrical insulation
5	Marble, slate, glass, porcelain
6	(no intended meaning)
7	(no intended meaning)
8	(no intended meaning)
9	Bronze, brass, copper, composition
10	(no intended meaning)
11	(no intended meaning)
12	Titanium, refractory material
13	(no intended meaning)
14	(no intended meaning)
15	(no intended meaning)
16	White metal, zinc, lead, babbitt, alloys
17	Magnesium, aluminum, aluminum alloys
18	Electrical windings, electromagnets, resistance, <i>etc.</i>
19	Solid fill

ECO630

## 4.68 SECTIONED AREA ENTITY (TYPE 230)

Table 10. Predefined Fill Patterns for the Sectioned Area Entity (continued)

PATRN	Meaning	Authority†
20‡	Earth	[ANSI79]
22‡	Rock	[ANSI79]
26‡	Cliffs	[ANSI72]
28‡	Sand	[ANSI79]
29‡	Sand and Sand Dunes	[ANSI72]
32‡	Stone Fill	AIA
34‡	Tailings or Mining Debris	[ANSI72]
36‡	Hill Shading	[ANSI72]
38‡	Water and Other Liquids	[ANSI79]
40‡	Wood (Across Grain)	[ANSI79]
41‡	Wood (With Grain)	[ANSI79]
42‡	Finish Wood	AIA
46‡	Large Scale Plywood	AIA
50‡	Shingles Siding	AIA
60‡	Glass	AIA
70‡	Cork, Felt, Fabric, Leather and Fiber	[ANSI79]
72‡	Sound Insulation	[ANSI79]
80‡	Thermal Insulation	[ANSI79]
82‡	Insulation (Loose Fill or Batts)	AIA
84‡	Insulation (Boards or Quilt)	AIA
86‡	Insulation (Solid Core)	AIA
90‡	Concrete	[ANSI79]
92‡	Structural Concrete	AIA
94‡	Light Weight Concrete	AIA
110‡	Concrete Block	AIA
124‡	Fire Brick on Common	AIA
134‡	Running Bond Masonry	AIA
136‡	Stack Bond Masonry	AIA
140‡	Marble	AIA
142‡	Slate, Bluestone, Soapstone	AIA
152‡	Cut Stone	AIA
154‡	Ashlar Stone	AIA
156‡	Cast Stone (Concrete)	AIA
157‡	Rubble	AIA
158‡	Rubble Stone	AIA
159‡	Squared Stone	AIA
172‡	Plaster, Sand and Cement	AIA
174‡	Concrete Plaster	AIA
178‡	Terrazzo	AIA

† AIA refers to the publications of the American Institute of Architects

‡Patterns of the Sectioned Area Entity with PATRN >19 have not been tested. See Section 1.9. ECO630

## 4.68 SECTIONED AREA ENTITY (TYPE 230)

Table 10. Predefined Fill Patterns for the Sectioned Area Entity [continued]

PATRN	Meaning	Authority†
210‡	Glaciers	[ANSI72]
220‡	Fresh Marsh	[ANSI72]
224‡	Salt Marsh	[ANSI72]
226‡	Submerged Marsh	[ANSI72]
234‡	Tidal Flat	[ANSI72]
236‡	Water Line	[ANSI72]
240‡	Cleared Land	[ANSI72]
244‡	Cultivated Land	[ANSI72]
246‡	Meadow	[ANSI72]
252‡	Deciduous Trees	[ANSI72]
254‡	Evergreen Trees	[ANSI72]
256‡	Oak Trees	[ANSI72]
262‡	Orchard	[ANSI72]
264‡	Vineyard	[ANSI72]
265‡	Willows	[ANSI72]
266‡	Corn	[ANSI72]
268‡	Tobacco	[ANSI72]

†AIA refers to the publications of the American Institute of Architects

‡Patterns of the Sectioned Area Entity with PATRN >19 have not been tested. See Section 1.9.

ECO630

For the fill pattern codes 0 and 19, the Parameter Data values for indices 3 through 7 are defaulted to 0.0 because they do not apply to the specified fill patterns. For the fill pattern codes 20 through 268, which use a composite of geometry entities, preprocessors shall set the indices 3 through 7 to 0.0, and postprocessors shall ignore them.

It is not intended that exact visual equivalence be preserved. The receiving system is to use similar, but not necessarily identical, patterns based on the pattern codes; the intent is to preserve the *functionality* implicit in the code. If the receiving system does not have a similar pattern, the default shall be no pattern fill.

The specification of enclosed definition curves allows for the nesting of curves so long as they meet the criteria specified below. This specification makes it possible to identify closed areas which are alternately filled and not filled as the pattern lines trace inward from the exterior definition curve. Since no nesting levels are required, interior definition curves may be interior to, or at the same level as, another interior definition curve. (See Figure 106).

All definition curves used to create a sectioned area shall meet the following criteria:

ECO630

1. The Sectioned Area Entity and all its constituent components shall be defined in the same definition space (model or drawing). ECO630
2. A definition curve shall be a simple closed curve. A curve is called a *simple closed curve* if its start point and terminate point coincide and, furthermore, as one traverses the curve from start point to terminate point, this common end point is occupied by no other point during the traversal. The curve intersects itself only at its endpoints. The definition curve separates a plane into two distinct areas, the interior area and the exterior area. Figure 107 shows cases of invalid definition curves. ECO630
3. The interior areas defined by two or more definition curves shall be related in one of two ways. The two areas shall either be completely disjoint or one shall completely enclose the other, in ECO630

## 4.68 SECTIONED AREA ENTITY (TYPE 230)

the sense that the interior area of the island or islands is a subset of the interior area of the outer definition curve. [Figure 108](#) shows cases of invalid relationships for definition curves.

4. Pattern lines and all definition curves shall be coplanar (*i.e.*, the Sectioned Area Entity and ECO630 all its constituent components shall share the same plane).

[Figure 109](#) shows an example of how implementations may differ but still accomplish the same result. ECO630 The example is a cross section detail of a T-slot. The T-slot may be represented as a single enclosed area to be sectioned ([Figure 109-A](#)). The area not to be sectioned may be represented by a second definition curve (island [Figure 109-B](#)). Note that the interior and exterior curves in [Figure 109-B](#) have coincident edges.

### Directory Entry

(1) Entity Type Number 230	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern #, ⇒	(5) Level #, ⇒	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ????01??	(10) Sequence Number D #
(11) Entity Type Number 230	(12) Line Weight #	(13) Color Number #, ⇒	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

### Sectioned Area Entity with Standard Crosshatching (Form 0)

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	BNDP	Pointer	Pointer to the DE of the exterior definition curve - a closed planar curve
2	PATRN	Integer	Fill pattern code
3	XT	Real	X coordinate through which a line shall pass (if not defaulted)
4	YT	Real	Y coordinate through which a line shall pass (if not defaulted)
5	ZT	Real	Z depth of lines
6	DIST	Real	Normal distance between adjacent lines
7	ANGLE	Real	Angle measured in radians from the XT axis to the lines of the sectioning. Default = $\pi/4$
8	N	Integer	Number of island curves or zero
9	ISLPT(1)	Pointer	Pointer to the DE of the first interior definition curve for an island
:	:	:	
8+N	ISLPT(N)	Pointer	Pointer to the DE of the last interior definition curve for an island

Additional pointers as required ([see Section 2.2.4.5.2](#)).

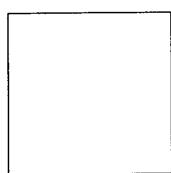
**Sectioned Area Entity with Inverted Crosshatching (Form 1)‡****Parameter Data**

ECO650

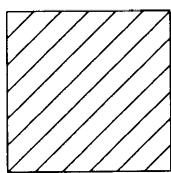
<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	BNDP	Pointer	Pointer to the DE of the exterior definition curve entity, or zero
2	PATRN	Integer	Fill pattern code
3	XT	Real	X coordinate through which a line shall pass (if not defaulted)
4	YT	Real	Y coordinate through which a line shall pass (if not defaulted)
5	ZT	Real	Z depth of lines
6	DIST	Real	Normal distance between adjacent lines
7	ANGLE	Real	Angle measured in radians from the XT axis to the lines of the sectioning. Default = $\pi/4$
8	N	Integer	Number of island curves ( $N > 0$ )
9	ISLPT(1)	Pointer	Pointer to the DE of the first interior definition curve entity for an island
:	:	:	
8+N	ISLPT(N)	Pointer	Pointer to the DE of the last interior definition curve entity for an island

Additional pointers as required (see Section 2.2.4.5.2).

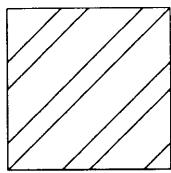
#### 4.68 SECTIONED AREA ENTITY (TYPE 230)



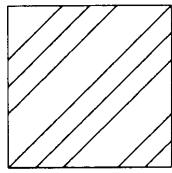
CODE 0



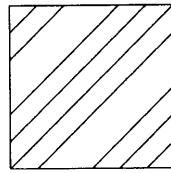
CODE 1



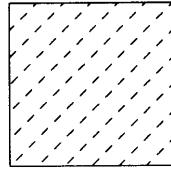
CODE 2



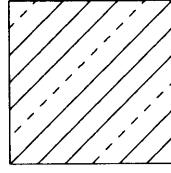
CODE 3



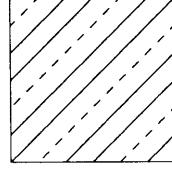
CODE 4



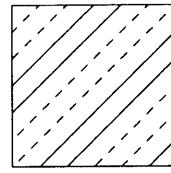
CODE 5



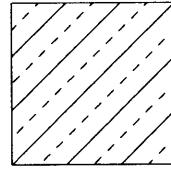
CODE 6



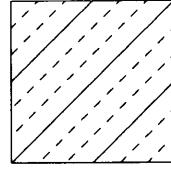
CODE 7



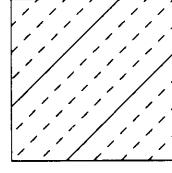
CODE 8



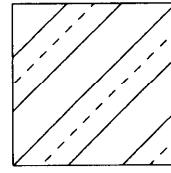
CODE 9



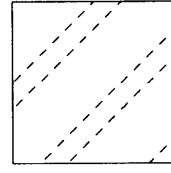
CODE 10



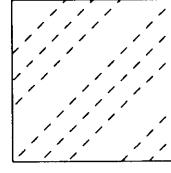
CODE 11



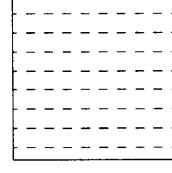
CODE 12



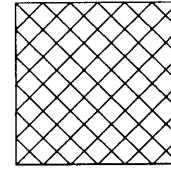
CODE 13



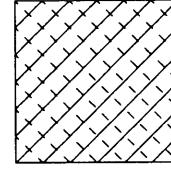
CODE 14



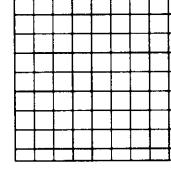
CODE 15



CODE 16



CODE 17



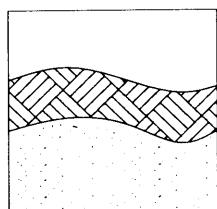
CODE 18



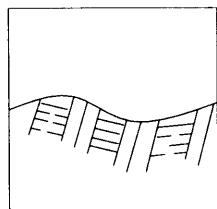
CODE 19

Figure 105. F230X.IGS Predefined Fill Patterns for the Sectioned Area Entity

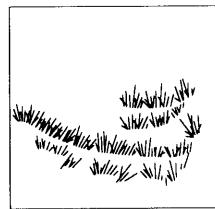
#### 4.68 SECTIONED AREA ENTITY (TYPE 230)



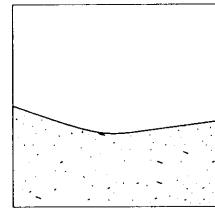
CODE 20



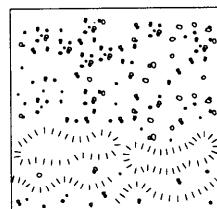
CODE 22



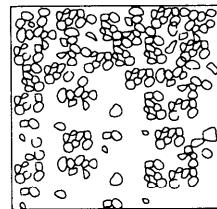
CODE 26



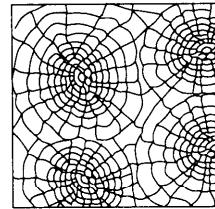
CODE 28



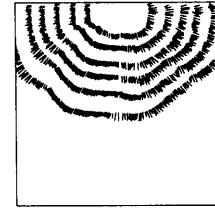
CODE 29



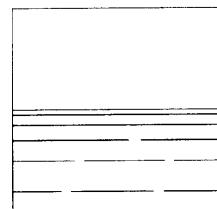
CODE 32



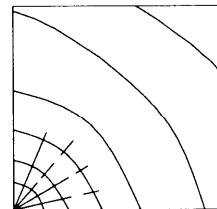
CODE 34



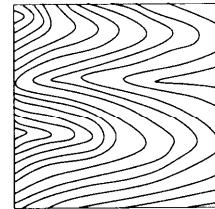
CODE 36



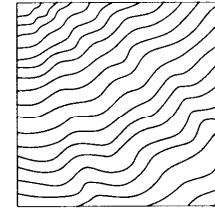
CODE 38



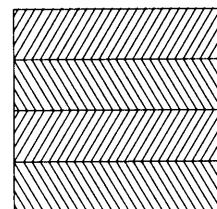
CODE 40



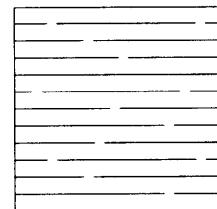
CODE 41



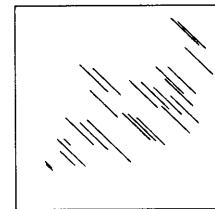
CODE 42



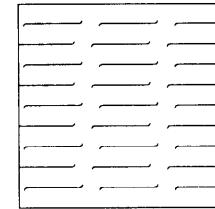
CODE 46



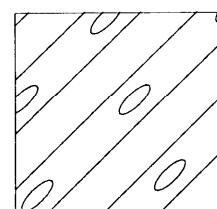
CODE 50



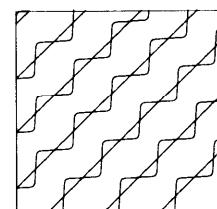
CODE 60



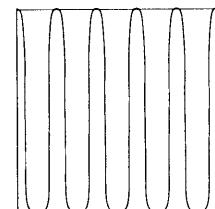
CODE 70



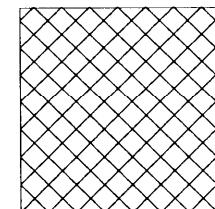
CODE 72



CODE 80



CODE 82



CODE 84

Figure 105. Predefined Fill Patterns‡ for the Sectioned Area Entity (continued)

#### 4.68 SECTIONED AREA ENTITY (TYPE 230)

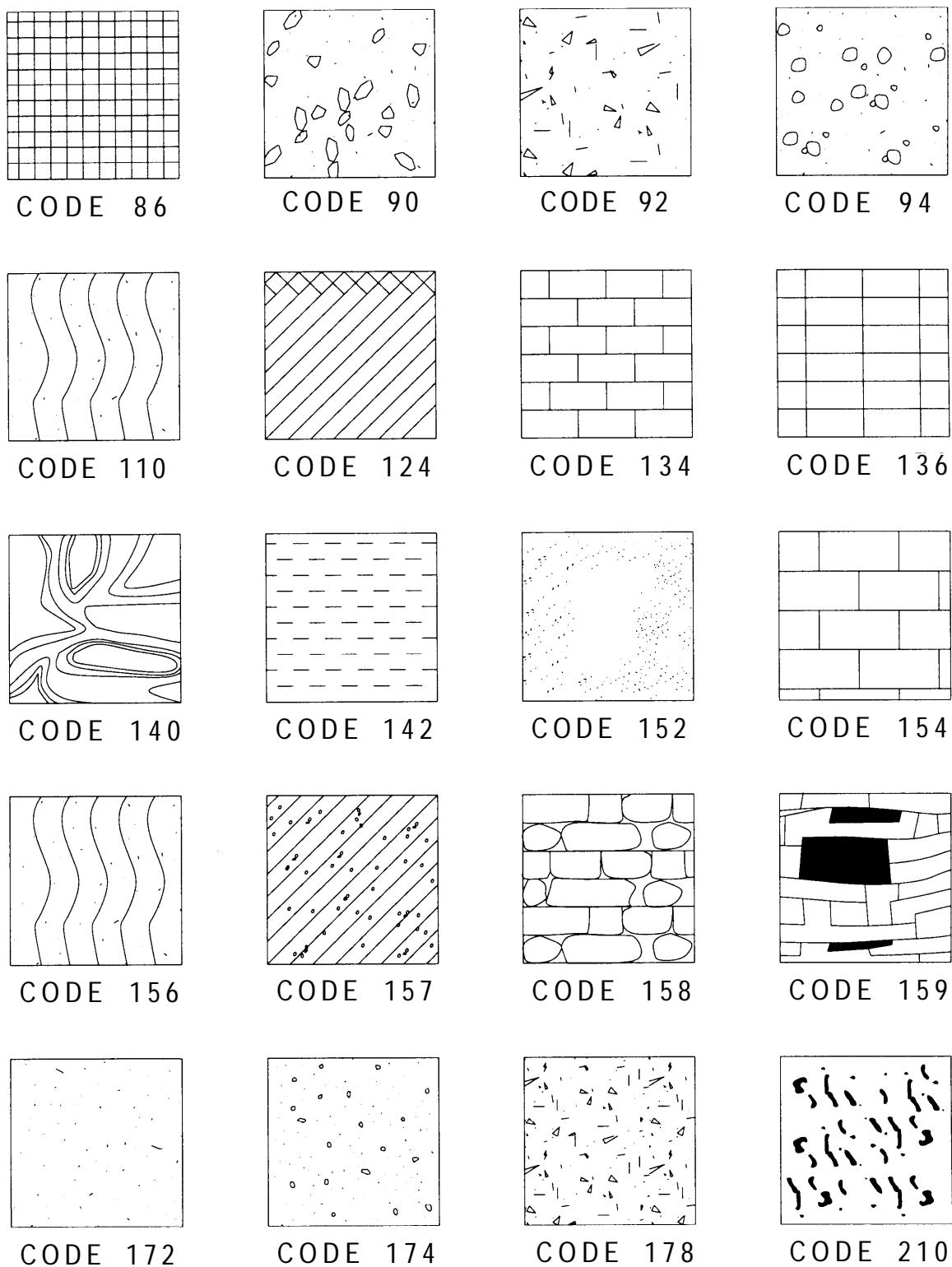
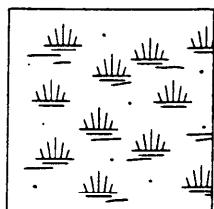
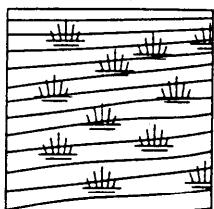


Figure 105. Predefined Fill Patterns‡ for the Sectioned Area Entity (continued)

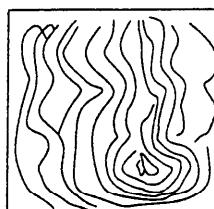
#### 4.68 SECTIONED AREA ENTITY (TYPE 230)



CODE 220



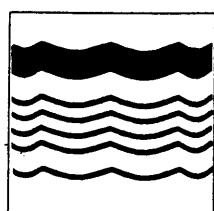
CODE 224



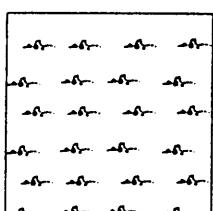
CODE 226



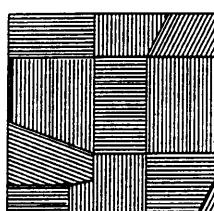
CODE 234



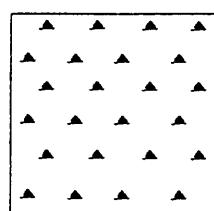
CODE 236



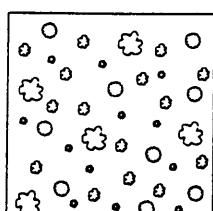
CODE 240



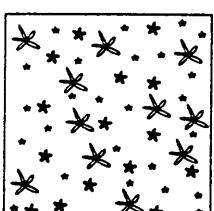
CODE 244



CODE 246



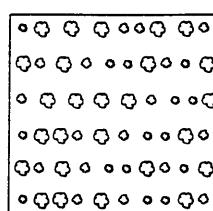
CODE 252



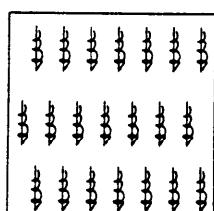
CODE 254



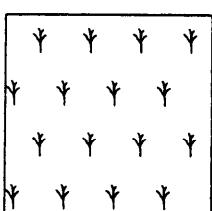
CODE 256



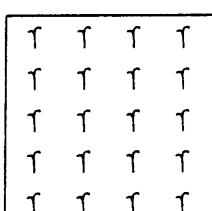
CODE 262



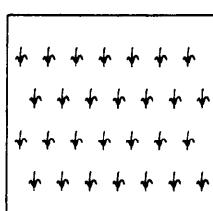
CODE 264



CODE 265



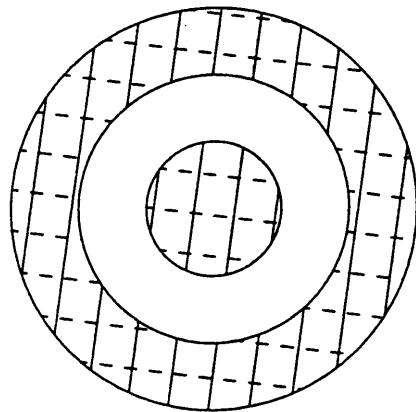
CODE 266



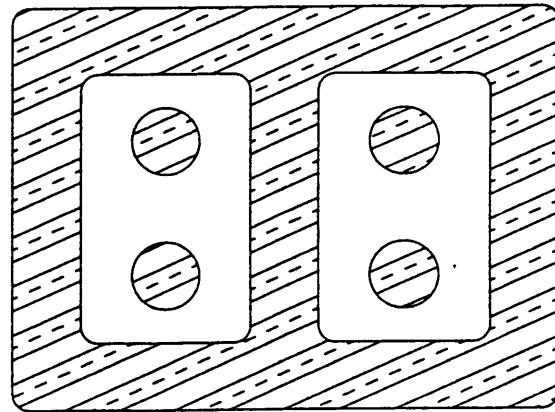
CODE 268

Figure 105. Predefined Fill Patterns for the Sectioned Area Entity (continued)

#### 4.68 SECTIONED AREA ENTITY (TYPE 230)

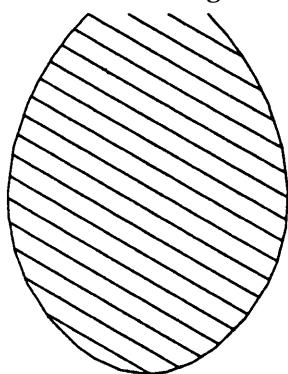


2 INTERNAL  
DEFINITION  
CURVES

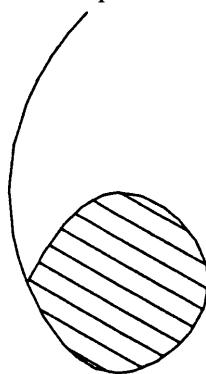


6 INTERNAL  
DEFINITION  
CURVES

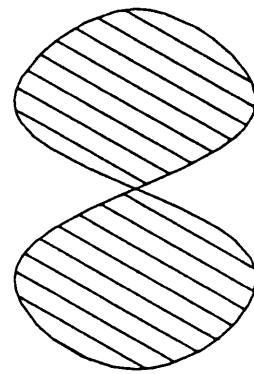
Figure 106. F230\_4X.IGS Examples of Nested Definition Curves



NOT CLOSED



SELF INTERSECTION  
AT OTHER THAN  
END POINTS



INTERSECTING  
AT MORE THAN  
ONE POINT

Figure 107. Examples of Invalid Definition Curves

**4.68 SECTIONED AREA ENTITY (TYPE 230)**

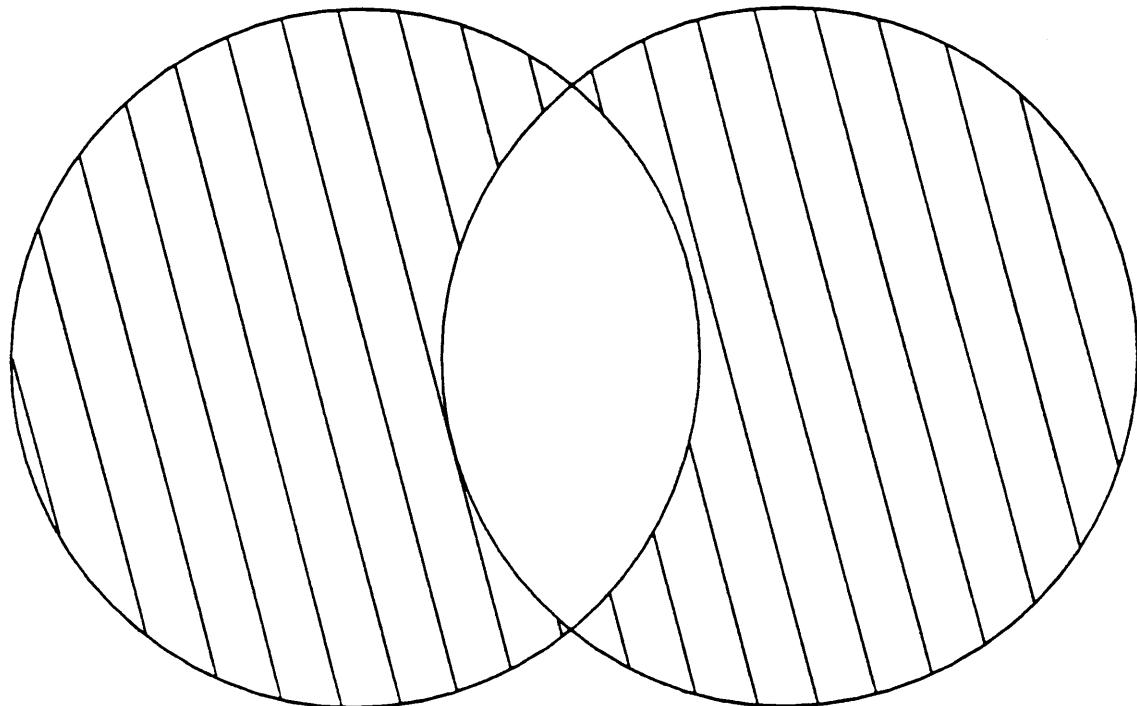
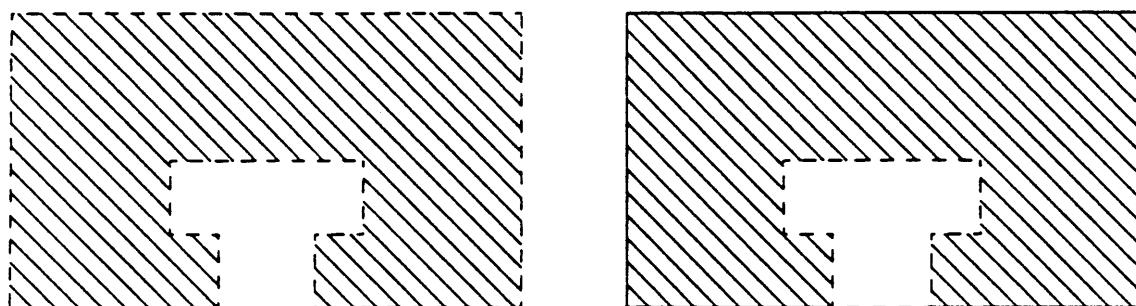


Figure 108. Example of an Invalid Relationship for Definition Curves

ECO630

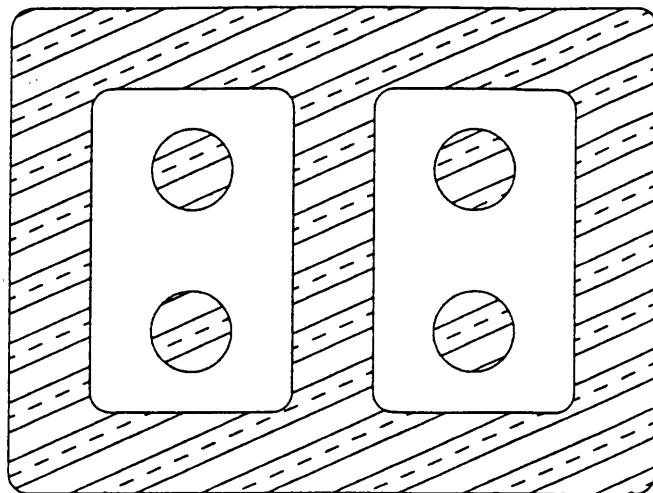


A ) NO ISLANDS  
(EXTERIOR DEFINITION  
CURVE DASHED)

B) ONE ISLAND  
(INTERIOR DEFINITION  
CURVE DASHED)

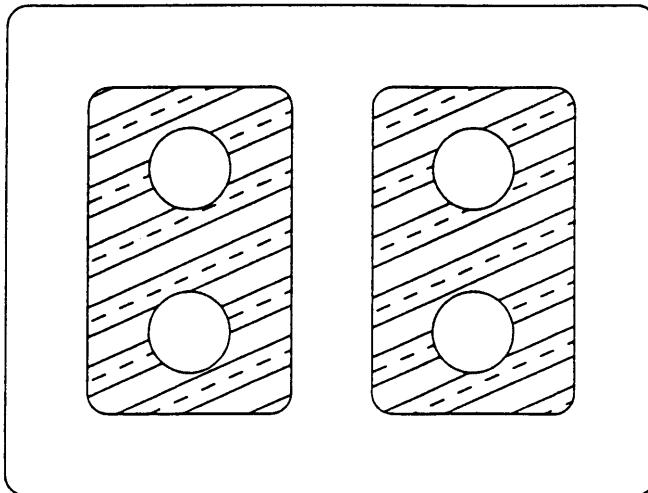
Figure 109. Example of Two Ways to Define an Area

**4.68 SECTIONED AREA ENTITY (TYPE 230)**



PATRN = 12  
BNDP ≠ 0  
FORM 0

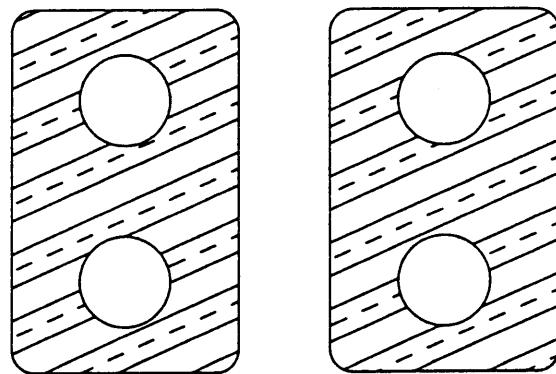
Figure 110. F23000X.IGS Examples of Standard and Inverted Crosshatching‡



PATRN = 12  
BNDP ≠ 0  
FORM 1

Figure 110. F23001AX.IGS Examples of Standard and Inverted Crosshatching‡ (continued).

**4.68 SECTIONED AREA ENTITY (TYPE 230)**



PATRN = 12  
BNDP = 0  
FORM 1

Figure 110. F230018X Examples of Standard and Inverted Crosshatching‡ (continued).

### 4.69 Associativity Definition Entity (Type 302)

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.

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.

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 in its back pointer parameter section to the directory entry of the associativity instance.

The provision in the schema which specifies whether or not a class is ordered indicates if the order of appearance of entries in the class is significant.

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.

The associativity definition fixes the number of classes for an Associativity and the number of items per entry in a particular class. Each associativity instance has 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.

Two kinds of Associativity Instance Entity (Type 402) are permitted within the file. Pre-defined associativities have form numbers in the range of 1 to 5000 and are defined in [Section 4.80.1](#). Definitions for pre-defined associativities do not appear in the file. The second kind of associativity is defined in the file by a preprocessor using the Associativity Definition Entity. Instances of these associativities have form numbers in the range of 5001–9999. These definitions appear once in the file for each form of Associativity defined.

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) is considered a class. [See Figure 111](#) for a complete example of associativity.

## 4.69 ASSOCIATIVITY DEFINITION ENTITY (TYPE 302)

### Directory Entry

(1) Entity Type Number 302	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **0002**	(10) Sequence Number D #
(11) Entity Type Number 302	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 5001 -9999	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

ECO650

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	K	Integer	Number of class definitions
2	BP(1)	Integer	1 = back pointers required 2 = back pointers not required
3	OR(1)	Integer	1 = ordered class 2 = unordered class
4	N(1)	Integer	Number of items per entry
5	IT(1,1)	Integer	1 = pointer to a directory entry 2 = value 3 = parameter is a value or a pointer if parameter >= 0 it is a value if parameter < 0, it is a pointer
:	:	:	
4+N(1)	IT(1,N1)	Integer	

Additional pointers as required (see Section 2.2.4.5.2).

The items in parameters 2 through 4+N(1) are repeated for each of the K classes.

#### 4.69 ASSOCIATIVITY DEFINITION ENTITY (TYPE 302)

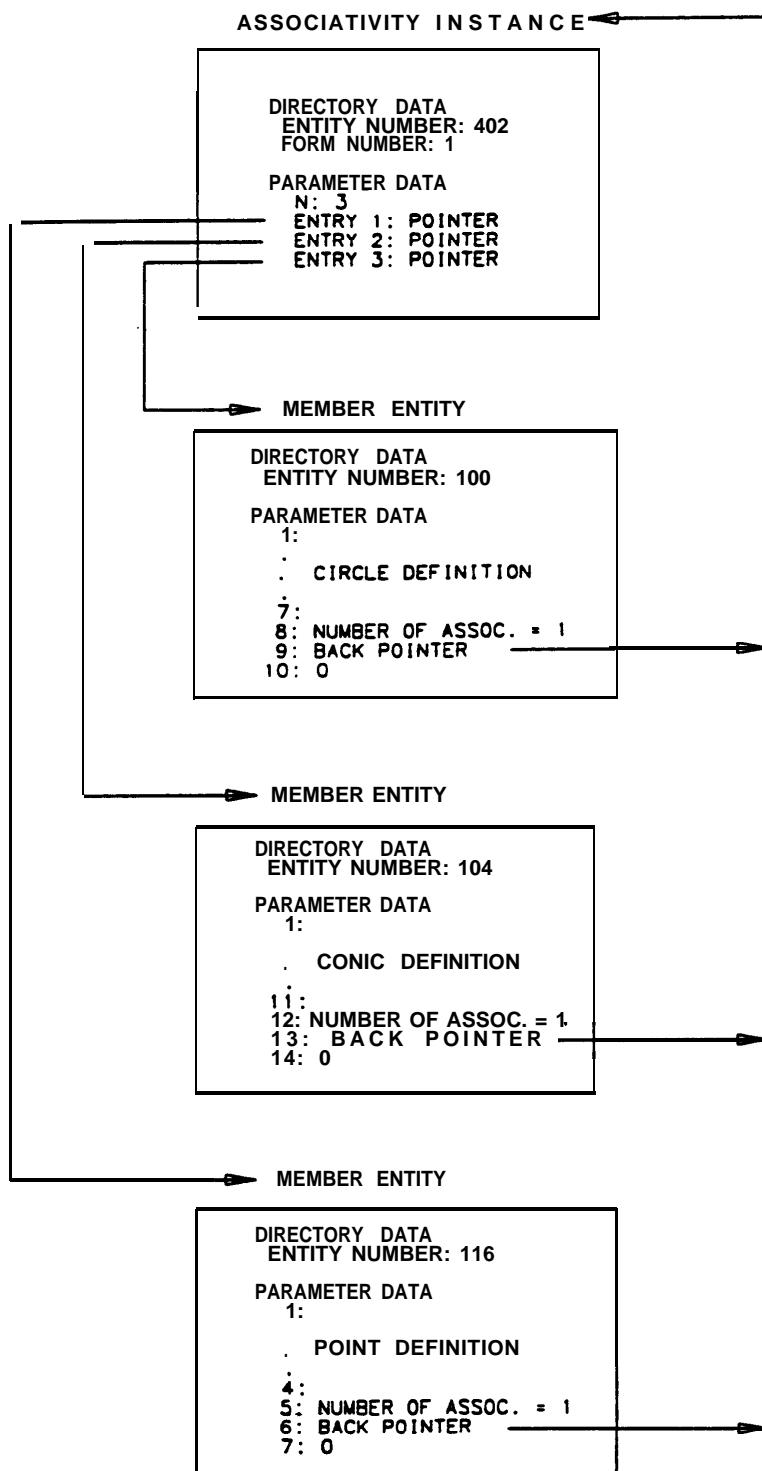


Figure 111. Relationships Between Entities in an Associativity

#### 4.70 Line Font Definition Entity (Type 304)

Two types of line fonts may be defined. One type considers a line font as a repetition of a basic pattern of visible-blank (or, on-off) segments superimposed on a line or a curve. The line or curve is then displayed according to the basic pattern. The other type considers a line font as a repetition of a template figure that is displayed at regularly spaced locations along a planar anchoring curve. The anchoring curve itself has no visual purpose.

Any line or curve geometry entity type may reference a Line Font Definition Entity by inserting a pointer to that entity in its Directory Entry Field 4, the line font pattern field. The type of line font being specified is then indicated by a form number in the Line Font Definition Entity.

The preprocessor shall select one of the line font patterns (see Section 2.2.4.4.4) and place the value in Directory Entry Field 4 of the Line Font Definition Entity. This value shall be the closest functional equivalent or the most visually similar. The value will be used by postprocessors which cannot support the Line Font Definition Entity. Examples of the standard line font patterns are shown in Figure 114.

For the Line Font Definition Entity, the Form Numbers are as follows:

ECO630

Form	Meaning
1	Line font specified by a repeating template subfigure
2	Line font specified by a repeating visible-blank pattern

ECO630

**Form 1:** specifies that the line font type is to be a repetition of template figure displays along the referencing anchoring curve. The template figure is specified as a Subfigure Definition Entity (Type 308). In this case, four values specify the entity as follows:

- The first parameter specifies the orientation of the template displays. This may remain constant, or it may vary with the direction of the anchoring curve at the point of each template figure display location.
- The second parameter is a pointer to the Subfigure Definition Entity containing the template display.
- The third parameter specifies display locations on the anchoring curve by giving the common arc length distance between corresponding points on successive template figure displays.
- The fourth parameter gives a scale factor to be applied to the template subfigure at each display location.

Figure 112 illustrates two examples of a line font using Form Number 1. In each case, the anchoring curve is a straight line.

**Form 2:** specifies that the line font type is to be a repetition of a basic visible blank pattern superimposed on the referencing line or curve. An arbitrary number of segments ( $M$ ) is used in the basic pattern. When the basic pattern is laid out horizontally, the first segment is the leftmost one; the  $M$ -th segment is the rightmost one. The length (in the units of the curve on which the pattern is being superimposed) of each segment of the pattern may be specified individually. This allows the visible blank sequence of the pattern to alternate between visible and blank regardless of the lengths of the segments but does not prohibit adjacent segments from being either both visible or both blanked when unequal lengths are employed. Another option for some patterns is to hold the length constant across segments, and achieve variation

ECO630

#### **4.70 LINE FONT DEFINITION ENTITY (TYPE 304)**

in the lengths of the visible and blanked segments by making the visible or blank segments be adjacent as required.

For example, a basic pattern whose left two-thirds is visible and whose right third is blanked, ECO630 may be described either by the sequence visible-blank with the length of the first segment twice that of the second, or else by the sequence visible-visible-blank, with the lengths of all three segments equal.

The visible-blank sequence is specified by correlating it with the rightmost M bits in the binary representation of a string of hexadecimal digits, the M-th segment being associated with the units bit of the binary representation of the rightmost hexadecimal digit. A 0 represents a blank, or off segment; a 1 represents a visible, or on segment.

For this line font type, the first parameter is the positive integer M giving the number of segments in the basic pattern. Then, parameter values 2 through M+1 give the lengths of the M segments. Finally, parameter value M+2 is the minimal string of hexadecimal digits whose significance has been described above.

Figure 113 shows an example of the Form Number 2 with 5 segments of unequal length. Two ECO630 repetitions of the basic font are illustrated.

## 4.70 LINE FONT DEFINITION ENTITY (TYPE 304)

### Directory Entry

(1) Entity Type Number 304	(2) Parameter Date ⇒	(3) Structure < n.a. >	(4) Line Font P-ttern 1-5	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix 0, ⇒	(8) Label Display < n.a. >	(9) Status Number **0002**	(10) Sequence Number D #
(11) Entity Type Number 304	(12) Line Weight <n.a. >	(13) Color Number <n.a. >	(14) Parameter Line Count #	(15) Form Number 1-2	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

### Line font specified by a repeating template subfigure (Form 1)

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	M	Integer	Display flag 0= Each template display is oriented by aligning the axes of the subfigure definition coordinate system with the axes of the definition space of the anchoring curve. 1= Each template display is oriented by aligning the X-axis of the subfigure definition coordinate system with the tangent vector of the anchoring curve at the point of incidence of the curve and the origin of the subfigure. The Z-axis of the subfigure definition coordinate system is aligned with the Z-axis of the definition space of the anchoring curve.
2	L1	Pointer	Pointer to the DE of the Subfigure Definition Entity for the template displays
3	L2	Real	Common arc length distance between corresponding points on successive template figure displays
4	L3	Real	Scale factor to be applied to the subfigure

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.70 LINE FONT DEFINITION ENTITY (TYPE 304)

**Line font specified by a repeating visible-blank pattern (Form 2)**

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	M	Integer	Number of segments in the basic pattern of visible-blank segments
2	L(1)	Real	Length of the first segment of the basic pattern
:	:	:	
1+M	L(M)	Real	Length of the last segment of the basic pattern
2+M	B	String	((M-1)/4 + 1) hexadecimal digits indicating which segments of the basic pattern are visible and which are blanked, where the expression represents the greatest integer result. (e.g., "5" indicates that segments 1 and 3 are visible. ) Bits are right justified.

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### 4.70 LINE FONT DEFINITION ENTITY (TYPE 304)

$M = 0$   
 $L_1 = \text{POINTER TO SUBFIGURE DEFINITION}$   
 $L_2 = \text{LENGTH OF SEPARATION}$   
 $L_3 = 0.5 \text{ (SCALE)}$

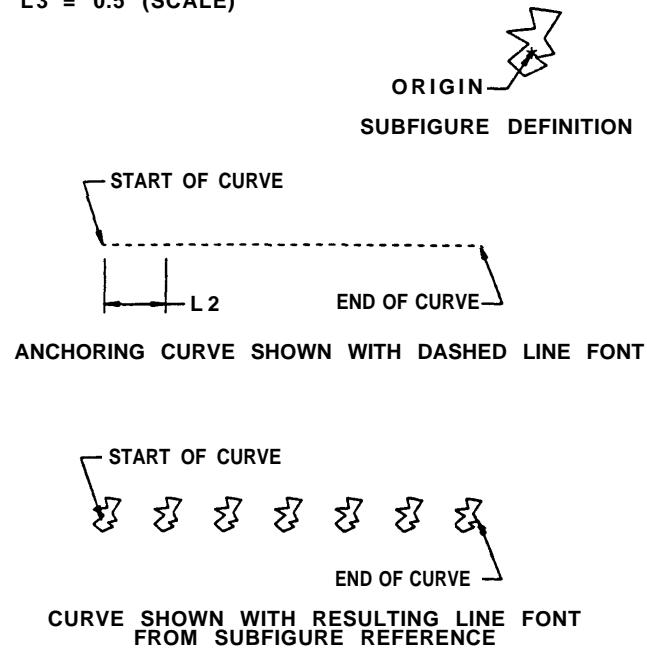


Figure 112. Line Font Definition Using Form Number 1 (Template Subfigure)

$M = 5$

$L_1 = L_3 = L_4 = L_5 = 2.0$

$L_2 = 1.0$

BIT PATTERN = 10110

HEXADECIMAL STRING = 2H16

RESULTING LINE FONT:

(2 CYCLES)

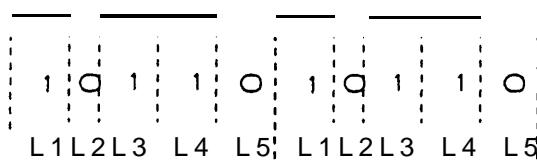


Figure 113. Line Font Definition Using Form Number 2 (Visible-Blank Pattern)

**4.70 LINE FONT DEFINITION ENTITY (TYPE 304)**

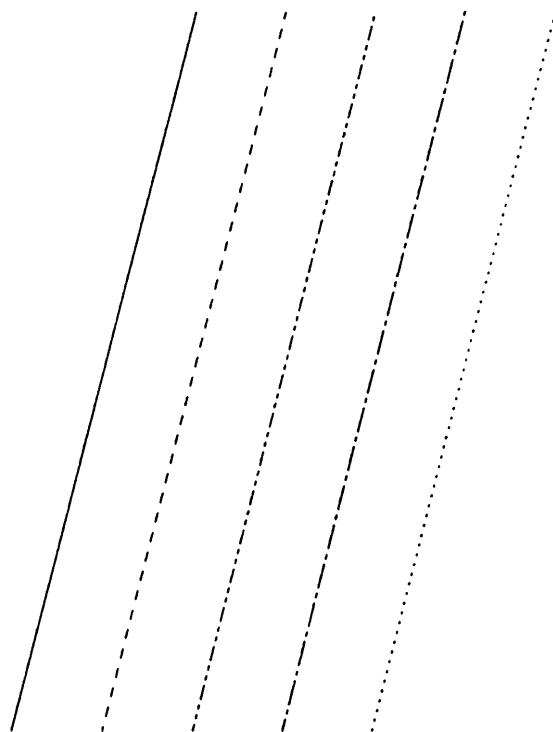


Figure 114. F30402X.IGS Examples of Standard Line Font Patterns

## 4.71 MACRO Definition Entity (Type 306) ‡

‡The Macro Definition Entity has not been tested. See Section 1.9.

ECO630

**4.71.1 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. Instances of these “new” entities are replaced during the MACRO processing by the constituent entities specified in the corresponding MACRO definition.

A MACRO definition is written using the MACRO Definition Entity (Type 306). The Parameter Data section of the entity contains the MACRO body. In the MACRO body, eleven types of language statements are usable. The statements are **LET**, **SET**, **REPEAT**, **CONTINUE**, **BREAK**, **IF**, **LABEL**, **GOTO**, **MACRO**, **ENDM**, **MREF**. The details of the MACRO syntax are given in Section 4.71.2. Each of the statements in a MACRO Definition Entity is terminated by a record delimiter.

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 and 11 of the Directory Entry record and refers to the definition by a pointer in the Structure Field (DE Field 3). The parameters for the instance are placed in the Parameter Data record of the instance.

The Directory Entry record of a MACRO definition has a standard form.

The attributes 4 through 9, 12, 13, 15, 18, and 19 have no significance. The default values for these attributes are taken from the Directory Entry record of the MACRO instance (described in Section 4.72).

The Parameter Data records of a MACRO definition consist of MACRO language statements. 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 (see Section 2.2.3). Every statement is terminated by a record delimiter.

### 4.71.2 MACRO Syntax

**4.71.2.1 Constants.** Constants may be integer, real, double precision real, pointer or string (See Section 2.2.2).

**4.71.2.2 Variables.** The significant part of a variable name is from one to six characters in length. The first character shall 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

## 4.71 MACRO DEFINITION ENTITY (TYPE 306) ‡

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 shall be alphabetic (A-Z or a-z), or numeric (0-9).

Variable Naming Convention

Variable Type	First Character
<b>Integer</b>	I-N, i-n
<b>Real</b>	A-H, O-Z a-h, o-z
<b>Double precision real</b>	!
<b>String</b>	\$
<b>Pointer</b>	#
<b>Label</b>	&

Examples of valid variable names.

Variable Type	Valid Names				
Integer	IJK	ICOUNT	K101	NTIMES	max
Real	XYZ	x1	y2	QrsTu1	
Double precision real	! h	!x1	!yz	!12341	
String	\$str	\$TITLE	\$label		
Pointer	#line	#note	#REF	#XYZ1	

Some examples of invalid variable names are:

\$\$\$\$ \$ not permitted after first character  
 1X43B 1 shall not be first character  
 A.BC . is invalid in variable name

ECO630

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.

**4.71.2.3 Functions.** Functions similar to FORTRAN library functions are provided. The rules for mixed mode have been relaxed so that it is not necessary to use **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, the functions do have a specific type of value that they return, *i.e.*, integer, real, double precision real, or string. The functions are listed here by the type of value returned. The type of argument usually used is also noted for clarity. 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 upper case.

**Functions Returning Integer Values:**

Function	Value Returned
IABS(i)	Absolute value of i
IDINT(!d)	Integer part of !d
IFIX(x)	Integer part of x
INT(x)	Integer part of x
ISIGN(i)	1 if i is positive, or 0 if it is zero, or -1 if it is negative

**Functions Returning Real Values:**

Function	Value Returned
ABS(x)	Absolute value of x
AINT(x)	Integer part of x, in real form
ALOG(x)	Natural logarithm of x
ALOG10(x)	Common (base 10) logarithm of x
ATAN(x)	Arctangent of x; angle returned in radians
COS(x)	Cosine of angle x; angle in radians
EXP(x)	Natural anti-logarithm of x ( <i>i.e.</i> , e to the x)
FLOAT(i)	Real (floated) value for i, <i>e.g.</i> , FLOAT(2) returns 2.
SIGN(x)	1. if x is positive, or 0 if x is zero, or -1. if x is negative
SIN(x)	Sine of angle x; angle in radians
SNGL(!d)	Single precision (real) value of double precision variable !d. As many significant digits of !d as possible are used in the returned value
SQRT(x)	Square root of x
TAN(x)	Tangent of angle x; angle in radians

**Functions Returning Double Precision Real Values:**

Function	Value Returned
DABS(!d)	Absolute value of !d
DATAN(!d)	Arctangent of !d; value returned in radians
DBLE(x)	Returns double precision real value of x. Note that this is merely conversion, not an extension. Thus, DBL (. 33333333) will return .33333333D0, but not .333333333333333333333333D0. Thus, DBLE(1./3.) is not necessarily equal to 1D0/3D0
DCOS(!d)	Cosine of angle !d; angle in radians
DEXP(!d)	Natural anti-logarithm of !d ( <i>i.e.</i> , e to the !d)
DLOG(!d)	Natural logarithm of !d
DLOG10(!d)	Common (base 10) logarithm of !d
DSIGN(!d)	1D0 if !d is positive, or 0D0 if zero, or -1D0 if negative
DSIN(!d)	Sine of angle !d; angle in radians
DSQRT(!d)	Square root of !d
DTAN(!d)	Tangent of angle !d; angle in radians

**Functions Returning String Values:**

Function	Value Returned
STRING (expression, format )	Character representation of the argument “expression”. See Section 4.71.3 for a description of the argument “format”.

**4.71.2.4 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, *i.e.*, 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 permissible 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 invalid, and AB does not imply A\*B, but rather the separate variable AB.

**Mode of expression evaluation.** Mixed mode (integer mixed with real, *etc.*) is permitted. Whenever two different types are to be operated upon, the calculation is performed in the “higher” type. Integer is the lowest type, real is next, and double precision real 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 1.0, 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.

**Conditional expressions.** Conditional expressions may be formed using functions, variables, and constants, and the following six standard relational operators:

Function	Symbol
less than or equal	.LE.
less than	.LT.
equal	.EQ.
greater than or equal	.GE.
greater than	.GT.
not equal	.NE.

Examples of conditional expressions include:

X.GT.3  
 SQRT(A+B) .NE. I+1  
 (!A-!B).GE.3.14159

**4.71.3 Language Statements.** There are eleven language statements that can be used. They are:

<b>BREAK</b>	<b>IF</b>	<b>MREF</b>
<b>CONTINUE</b>	<b>LABEL</b>	<b>REPEAT</b>
<b>ENDM</b>	<b>LET</b>	<b>SET</b>
<b>GOTO</b>	<b>MACRO</b>	

These “keywords” are recognized only in uppercase, and every statement shall begin with one of ECO630 these keywords. Statements are free format; blanks and tabs are ignored except within strings. Statements may extend over several lines, or more than one statement may be present on a line. All statements are terminated by a record delimiter which shall be present.

#### 4.71.3.1 LET Statement(Arithmetic)

This is the 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 shall have been previously defined; it cannot be assumed that variables will default to zero if they are undefined. Some examples of valid **LET** statements:

```
LET HYPOT = SQRT(A**2+B**2) ;
LET X = X + 1 ;
LET ROOT1 = -B + SQRT(B*B - 4*A*C) ;
LET I = 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 arithmetic operations (*i.e.*, +, -, \*, /, \*\*) are not possible with string variables. Two forms of LET statements for string variables are possible:

```
LET $str = 23Hstring of 23 characters;
      or
LET $str1 = $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 “\$str1.” Examples of these statements include:

```
LET $title = 3HBox;
LET $subti = 6HBottom;
LET $x = $subti;
```

Note that if a string variable appears on the right hand side of the statement, it shall have been ECO630 previously defined. Spaces are not ignored within a string constant; they become part of the string. Any printable 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 **LET** statement including string function is:

## 4.71 MACRO DEFINITION ENTITY (TYPE 306) ‡

LET \$str = STRING (expression, format);

where “expression” is any normal integer, real or double precision real expression, “\$ str“ is a string variable name, and “format” is a format specification as used in a FORTRAN FORMAT statement. The allowable format specifications 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 specifications follow the standard FORTRAN convention. “Iw” causes ECO630 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 precision real values. Note that this is one place where mixed mode is not allowed. The type of format specification and the type of the expression’s result shall be identical.

### LET Statements (Attributes)

Attributes (those appearing in the directory entry record for the MACRO instance) may be set using the LET statement. The format 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. Attributes may also be set to string constants or string variables but not to the result of a STRING function. Examples include:

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

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

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

The word "/HDR" is the only nonconstant 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.

Attribute	Name	Default
Line font pattern	/LFP	/HDR
Level	/LEV	/HDR
View	/VIEW	/HDR
Transformation matrix	/MTX	/HDR
Label display Associativity	/CE	0
Blank status	/BS	/HDR
Subordinate entity	/SE	/HDR
Entity use	/ET	/HDR
Hierarchy	/HF	/HDR
Line weight	/LW	/HDR
Color Number	/PN	/HDR
Form Number	/FORM	0
Entity label	/LABEL	blanks
Entity subscript	/SUB	0

**4.71.3.2 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 entity type number, such as “110”; and “argument list” is a group of variables which is the parameter data of the entity. Examples of this type of **SET** statement include:

```
SET #LINE = 110,X1,Y1,Z,X2,Y2,Z,0,0;
SET #ABC = 828,Z,A+B/C,Y1,X2,Y2+1,0,0;
SET #qwe = 864,15Hstrings allowed, X,Y,$this2;
```

The argument list may contain expressions and may spread over more than one line. At least one argument shall be present, *i.e.*, the argument list may not be null. The entity type number may not be an expression; *i.e.*, it shall 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. ECO630

“Forward referencing” of pointers is valid 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 shall appear on the left hand side of one SET statement. ECO630

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.

**4.71.3.3 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 precision real type; in the case of real or double precision real 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 are:

```
REPEAT 3;
REPEAT N+1;
REPEAT 0;
REPEAT X+Y;
```

REPEAT statements may be nested only to a depth of ten.

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

**4.71.3.4 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** statements and **CONTINUE** statements in a MACRO shall be the same. A **CONTINUE** statement(s) is not implied by **ENDM**.

**4.71.3.5 BREAK Statement** A **BREAK** statement may be used within a **REPEAT** construct to terminate the processing of statements of the **REPEAT** construct before the completion of the specified number of loops, such as upon detection of a condition during the processing. The form of a **BREAK** statement is:

**BREAK;**  
or  
**IF** conditional expression, **BREAK;**

When a **BREAK** statement is encountered, processing of MACRO statements resumes with the statement immediately following the **CONTINUE** statement marking the end of the affected **REPEAT** construct.

**4.71.3.6 IF Statement** The **IF** statement causes a single language statement to be executed if a certain condition is true.

The form of an **IF** statement is:

**IF** conditional expression, language statement;

where “conditional expression” is a conditional expression as described in [Section 4.71.2](#), “language statement” can be any statement allowed in a MACRO except:

**MACRO**  
**ENDM**  
**IF**  
**LABEL**

Examples of **IF** statements:

```
IF A. LT.3, LET A=3;
IF B. EQ. 0, SET #LIN1=110, . . . ;
IF SWITCH. EQ .1, GOTO &A;
```

## 4.71 MACRO DEFINITION ENTITY (TYPE 306) ‡

**4.71.3.7 LABEL Statement** The **LABEL** statement is used to mark a position in a MACRO where the execution control is transferred to using a **GOTO** statement. The form of a **LABEL** statement is:

**LABEL** label name;

where “label name” is any character string beginning with an ampersand (&). It should be from ECO630 one to six characters long (including the &). Within a single MACRO definition, all label names shall be unique. Examples are:

```
LABEL &loop;  
LABEL &end;
```

**4.71.3.8 GOTO Statement** This statement is used to transfer the execution control to a particular point which is marked by a **LABEL** statement. The form of a **GOTO** statement is:

**GOTO** label name;

where “label name” is any label name specified in a **LABEL** statement.

The **GOTO** statement can be used to jump both forward and backward, but both the **GOTO** statement and the target **LABEL** shall be at the same nesting level and within the same **REPEAT** construct. Examples are:

```
GOTO &start;  
GOTO &end;
```

**4.71.3.9 MACRO Statement** The MACRO statement is used to signify the start of a MACRO definition. The first statement in every MACRO definition shall be a MACRO statement. The form of ECO630 a MACRO statement is:

306, MACRO, entity type number, argument list;

where “entity type number” is the assigned entity number of the MACRO, and “argument list” is a list of parameters that are to be assigned values at execution time. Entity type numbers in the range of 600 to 699 and 10000 to 99999 will be used.

The argument list may not be null. The parameters in the argument list take the form of the variables described in [Section 4.71.2](#). Note that the argument list may not contain expressions, only symbolic variable names.

One additional type of variable, the “class variable” can be used in an argument list. The class variable takes the form:

size\_variable (class\_var\_1, class\_var\_2, . . . , class\_var\_n)

where size-variable precedes the occurrence of the class variable in the argument list and class\_var\_i members are referenced in the MACRO body by means of a subscript:

class\_var\_i (J)

In the **MACRO** statement, the size\_variable is used to identify the class variable collection being defined. In the MACRO body, the size\_variable indicates the number of sets of class variable members that are included (*i.e.*, the number of times the class variable collection is to be repeated).

A simple class variable example is:

N(ITEM1, ITEM2, ITEM3, ITEM4)

which specifies a class variable collection with four members. For an instance of the MACRO, using the example class variable with N equal to 3, three sets of class variable data for the collection are available to the macro body statements. The parameter list for the associated MACRO instance is:

ITEM1(1), ITEM2(1), ITEM3(1), ITEM4(1), . . . , ITEM3(3), ITEM4(3)

Each value for each member of the class variable may be referenced individually:

ITEM1(1), ITEM1(2), ITEM1(3), ITEM1(4), etc.

in any order, or implicitly, by using an index variable, *i.e.*, J:

ITEM3 (J) with J ranging from 1 to 3

Use of the class variable to represent a MACRO with a parameter list identical to the Views Visible Associativity, Form 4, would be specified as:

306, MACRO, 681, N1, N2, N1(#DEV, LF, #DEF, IPN, LW),  
N2(#DE), N, N(#DEA), M, M(#DEP) ;

where:

N1(#DEV,LF,#DEF,ICN,LW)	indicates the blocks of views visible, line font, color number, and line weight information
N2(#DE)	contains the pointers to the entities included in the view
N(#DEA)	contains the back pointers/text pointers
M(#DEP)	contains the pointers to properties

Note that zero is a valid value for a size\_variable in a MACRO instance.

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

**ENDM ;**

All MACROS shall have an ENDM statement as their last statement. ENDM is not implied by the end ECO630 of the parameter data section.

#### **MREF Statement**

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

MREF, ptr, entity type number, argument list;

where "ptr" may be either a pointer variable or an integer expression. The value is the sequence number of the Directory Entry record of the MACRO definition being referenced. "Entity type number" is the assigned entity number of the MACRO being referenced. "Argument list" is 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. (All variables not in the argument list are treated as local variables. ) 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;
```

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 precision real 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.

**4.71.4 The 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 of only language statements in the parameter data. The character strings constituting the language statements in the MACRO definition are not set off by means of the nH structure of string constants but rather consist of only the actual character string terminated by a record delimiter (see [Section 2.2.2.5](#)).

## 4.71 MACRO DEFINITION ENTITY (TYPE 306) ‡

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
306	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**0002**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
306	< n.a. >	< n.a. >	#	0				#	D # + 1

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	ID	Literal	MACRO
2	NE	Integer	Entity Type ID
3	TEXT	Language	First statement statement
:	:	:	
2+N	TEXT	Language	Last statement statement
3+N	T	Literal	ENDM

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.72 MACRO Instance Entity‡**

ECO630

‡The Macro Instance Entity has not been tested. [See Section 1.9.](#)

ECO630

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 structure field (Directory Entry Field 3), which contains a negated pointer to the Directory Entry of the MACRO Definition Entity ([Type 306](#)).

Five examples are given to illustrate some of the capabilities of a MACRO.

ECO630

- 1 . Isosceles Triangle
- 2 . Repeated Parallelograms
- 3 . Concentric Circles
- 4 . Ground Symbol
- 5 . Useful Features

**Directory Entry**

**Entity Type Number:** As defined for each MACRO in the range 600 to 699 or 10000 to 99999.

**Structure Field:** Negated Pointer to the DE of the MACRO Definition Entity ([Type 306](#)).

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

**Parameter Data**

The parameter data section for an instance has the following form: With all parameter data entities, the first record begins with the entity type number as defined in the MACRO.

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1,...,K	As appropriate		Values for the arguments to the MACRO must agree in format and number with the arguments in the MACRO statement of the definition.

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.72.1 Example 1: Isosceles Triangle** The following MACRO definition creates an isosceles triangle, given a vertex point, a height, a base width and a scale.

**Directory Entry**

**Entity Type Number: 306**

**Parameter Data**

```
306, MACRO, 621, X1, Y1, A1, A2, K;
LET Z = 0;
SET #Line1 = 110,X1,Y1,Z,X1+(K*A1),Y1+(K*A2/2.),Z,0,0;
SET #Line2 = 110,X1+(K*A1),Y1+(K*A2/2.),Z,
           X1+(K*A1),Y1-(K*A2/2.),Z,0,0;
SET #Line3 = 110,X1+(K*A1),Y1-(K*A2/2.),Z,
           X1,Y1,Z,0,0;
ENDM;
```

The MACRO can be used to create a triangle by using a MACRO instance which supplies the needed values for X1, Y1, A1, A2, and K. The parameter data section for the MACRO instance would have the following format:

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	X1	Real	X coordinate of vertex
2	Y1	Real	Y coordinate of vertex
3	A1	Real	Height of triangle
4	A2	Real	Base of triangle
5	K	Integer	Scaling factor

Additional pointers as required ([see Section 2.2.4.5.2](#)).

In particular, to create a triangle shown in [Figure 115](#) with a base of 5. and a height of 17., a vertex at (0,0)), and a scale factor 1, the following instance could be placed into the file:

**Directory Entry**

**Entity Type Number: 621**

Structure: -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;
```

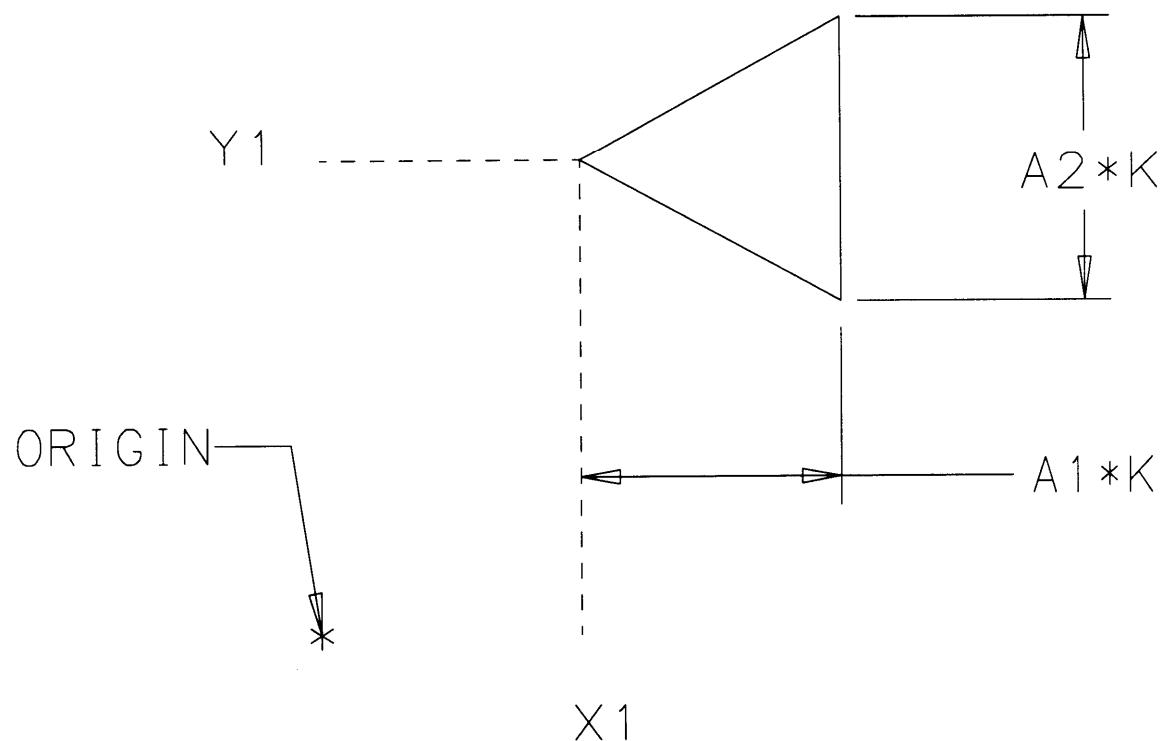


Figure 115. Parameters of the Isosceles Triangle Macro in Example 1 in Text

**4.72.2 Example 2: Repeated parallelograms** The following MACRO takes the coordinates of three points and a repetition number as arguments and creates a pattern of repeated parallelograms as shown in [Figure 116](#). The three points represent the vertices of the initial parallelogram. The repetition number argument (NTANG) controls how many additional parallelograms will be drawn offset in the positive X and Y direction from the initial one. For simplicity, the points have been constrained to all lie in a plane parallel to the X-Y plane.

**Directory Entry****Entity Type Number: 306****Parameter Data**

```

306,MACRO, 600, X1, Y1, X2, Y2, X3, Y3, Z, NTANG;
IF NTANG.EQ.0, GOTO &END;
LET XDEL = (X2-X1)/NTANG;
LET YDEL = (Y3-Y1)/NTANG;
LET K = 0;
REPEAT NTANG +1;
    SET #VLINE = 110,X1+K*XDEL,Y1+K*YDEL,Z,
                X2+K*XDEL,Y2+K*YDEL,Z,0,0;
    SET #HLINE = 110,X1+K*XDEL,Y1+K*YDEL,Z,
                X3+K*XDEL,Y3+K*YDEL,Z,0,0;
    SET #VLINE = 110,X3+K*XDEL,Y3+K*YDEL,Z,
                X3+(X2-X1)+K*XDEL,Y2+(Y3-Y1)+K*YDEL,
                Z,0,0;
    SET #HLINE = 110,X2+K*XDEL,Y2+K*YDEL, Z,
                X3+(X2-X1)+K*XDEL,Y2+(Y3-Y1)+K*YDEL,
                Z,0,0;
    LET K = K+1;
CONTINUE;
LABEL &END;
ENDM;
```

Parameter Data for an instance of this MACRO looks like this:

**Directory Entry****Entity Type Number: 600****Parameter Data**

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

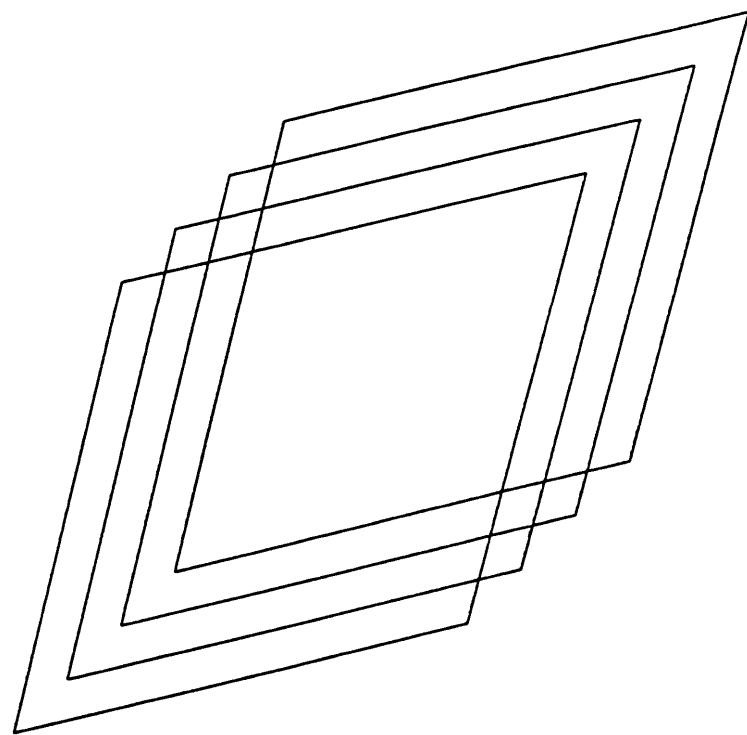


Figure 116. Repeated Parallelograms Created by Macro Example 2 in Text

**4.72.3 Example 3: Concentric circles** The following MACRO, given a coordinate, a radius, and a number, creates concentric circles out to the radius. A point is put into the center. [Figure 117](#) shows the result.

**Directory Entry**  
**Entity Type Number: 306**

**Parameter Data**

```
306,MACRO,601,XC,YC,ZC,R,NCIRC;
IF NCIRC .EQ. 0, GOTO &END;
LET DELTR = R/NCIRC;
REPEAT NCIRC;
  SET #CIR = 100,ZC,XC,YC,XC,YC+R,XC,YC+R,0,0;
  LET R = R - DELTR;
CONTINUE;
SET #PT = 116, XC, YC, ZC, 0, 0, 0;
LABEL &END;
ENDM;
```

Parameter Data for an instance of the MACRO which would create four concentric circles around the origin out to a radius of 20 looks like this:

**Directory Entry**  
**Entity Type Number: 600**

**Parameter Data**

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

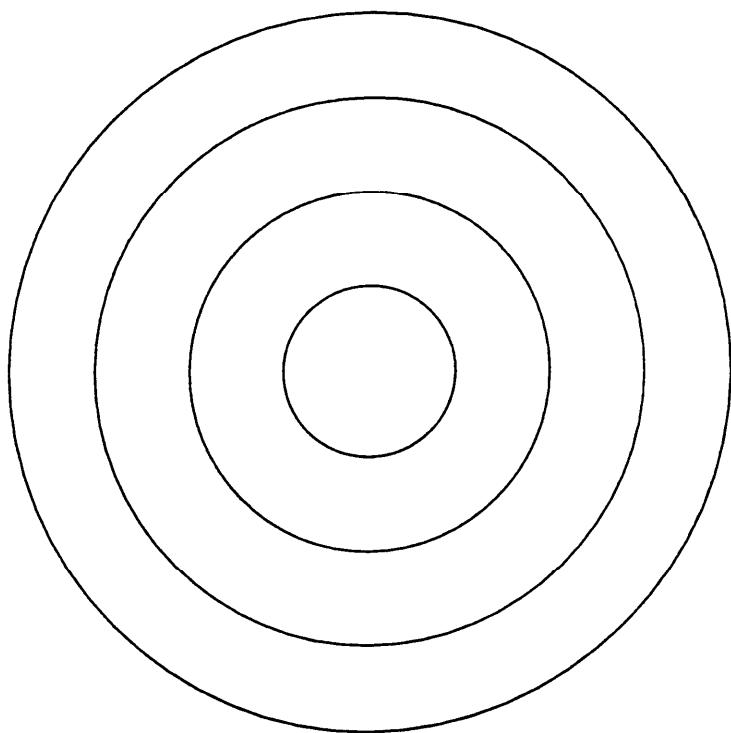


Figure 117. Concentric Circles Created by Macro Example 3 in Text

**4.72.4 Example 4: Electrical ground symbol** This MACRO takes a point and a base length and constructs a ground symbol (horizontally) at that point. [Figure 118](#) shows the result.

**Directory Entry**  
**Entity Type Number: 306**

**Parameter Data**

```
306, MACRO, 602, X, Y, Z, B;  
IF B.EQ.0, GOTO &A;  
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;  
LABEL &A;  
ENDM;
```

Parameter Data for an instance of this MACRO looks like this:

**Directory Entry**  
**Entity Type Number: 602**

**Parameter Data**

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



Figure 118. Ground Symbol Created by Macro Example 4 in Text

**4.72.5 Example 5: Useful features** This last example demonstrates the use of various MACRO features. It is not meant as an example of a “useful” MACRO.

#### Parameter Data

```

306,MACRO,613,NROW,NCOL,VDIST,HDIST,!ANGLE;
LET/LABEL = 6HPOINTS;
LET !SIN=DSIN(!ANGLE); LET !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., 0,0,0;
        LET ICC = ICC + 1;
    CONTINUE;
    LET IRC = ICC + 1;
CONTINUE;
LET $NPTS = STRING(NROW*NCOL,17);
LET/LABEL = $NPTS;
SET #LINE= 110, 0., 0., 0., 10., 0., 0. ;
SET #CIRC "100, 0., 0., 0., 10., 0., 10., 0. ;
MREF, 22, 601, 0., 0., 0., 10., 5;
ENDM ;

```

Parameter Data for an instance of this MACRO looks like this:

**Directory Entry**  
**Entity Type Number: 613**

#### Parameter Data

```
613, 4, 5, 0.2, 0.1, 7.85398D-01;
```

## 4.73 SUBFIGURE DEFINITION ENTITY (TYPE 308)

### 4.73 Subfigure Definition Entity (Type 308)

The Subfigure Definition Entity supports multiple instantiation of a defined collection of entities. ECO630 This reduces file size and simplifies maintenance when an identical feature (e.g. a bolt) is used repeatedly in the file. Each Subfigure Definition Entity may reference any other entities, including other Subfigure Instance Entities ([Type 408](#)). When a Subfigure Definition references a Subfigure Instance, it is called *nesting*. DEPTH indicates the amount of nesting. 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 an instance of a subfigure definition with DEPTH N-1.

#### Directory Entry

(1) Entity Type Number 308	(2) Parameter Data ⇒	(3) Structure $< n.a. >$	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View $< n.a. >$	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number **??02??	(10) Sequence Number D #
(11) Entity Type Number 308	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

Note: When the Hierarchy is set to Global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, Color Number, Level, View, and Blank Status.

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
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(1)	Pointer	Pointer to the DE of the first associated entity
:	:	:	
3+N	DE(N)	Pointer	Pointer to the DE of the last associated entity

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## **4.74 TEXT FONT DEFINITION ENTITY (TYPE 310)**

### **4.74 Text Font Definition Entity (Type 310)**

This entity defines the appearance of characters in a text font. The data describing the appearance ECO630 of a character may be located by the Font Code (FC) and the ASCII character code (AC). This entity may describe any or all the characters in a character set. Thus, this entity may be used to describe a complete font or a modification to a subset of characters in another font. Font Code (FC) and Font Name (FNAME) are the number and name used to reference the font on the sending system. When this entity is a modification to another font, the Supersedes Font (SF) value (Parameter 3) indicates which font the entity modifies. When it is not, this field is ignored. This value is an integer which indicates the font number to be modified or, if negative, is the pointer value to the Directory Entry of another Text Font Definition Entity. When this entity modifies another font, *i.e.*, Parameter 3 references another font, the definitions in this entity supersede the definition in the original font. For example, a complete set of characters may have their font definition specified by this entity. Another Text Font Definition Entity could reference the first definition and modify a subset of the characters.

Each character is defined by overlaying an equally spaced square grid over the character. The character is decomposed into straight line segments which connect grid points. Grid points are referenced by standard Cartesian coordinates. The position of the character relative to the grid is defined by two points. The character's origin point is placed at the origin (0,0) of the grid and defines the position of the character relative to the text origin of that character. The second point defines the origin point of the character following the character being defined. This allows the spacing between characters to be specified. Construction of text strings consists of placing the character origin of the first character at the text string origin and placing subsequent character origins at the location specified in the previous character as the location of the next character's origin.

The parameterization of the character appearance is described by the motion of an imaginary pen moving between grid points. Commands to move the pen reference the grid location to which the pen is to move. The pen may be "lifted" such that its movement is not displayed. The representation of the movement of the pen is a sequence of pen commands and grid locations. Each movement of the pen is represented by a pen up/down 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.

**Directory Entry**

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
310	$\Rightarrow$	$< n.a. >$	$< n.a. >$	$< n.a. >$	$< n.a. >$	$< n.a. >$	$< n.a. >$	**0002**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
310	$<n.a. >$	$<n.a. >$	#	0				#	D # + 1

**Parameter Data**ECO650  
ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	FC	Integer	Font Code
2	FNAME	String	Font Name
3	SF	Integer or Pointer	Number of the font which this definition supersedes or Pointer to the DE of the Text Definition Entity if negative
4	SCALE	Integer	Number of grid units which equal one text height unit
5	N	Integer	Number of characters in this definition
6	AC(1)	Integer	ASCII code for first character
7	NX(1)	Integer	Grid location of the first character's origin
8	NY(1)	Integer	
9	NM(1)	Integer	Number of pen motions for first character
10	PF(1,1)	Integer	Pen up/down flag: 0 Down (default), 1 = Up
11	X(1,1)	Integer	Grid location to which the pen is to move
12	Y(1,1)	Integer	
:	:	:	
9+3*NM(1)	Y(1,NM(1))	Integer	Last grid location of first character
10+3*NM(1)	AC(2)	Integer	ASCII code for second character
11+3*NM(1)	NX(2)	Integer	Grid location of the second character origin
12+3*NM(1)	NY(2)	Integer	
13+3*NM(1)	NM(2)	Integer	Number of pen motions for second character
:	:	:	
5+4*N+	Y(N,NM(N))	Integer	Last grid location of last character
$3*\sum_{i=1}^N NM(i)$			

Additional pointers as required (see Section 2.2.4.5.2).

#### **4.74 TEXT FONT DEFINITION ENTITY (TYPE 310)**

Examples of character definitions are shown in [Figures 119](#) and [120](#). The parameters for the first example are:

Parameter Name	Value
FC	1
FNAME	8HSTANDARD
SF	
SCALE	8
N	60
AC1	65
NX1	11
NY1	0
NM1	4
PF1	0
X1	4
Y1	8
PF2	0
X2	8
Y2	0
PF3	1
X3	2
Y3	4
PF4	0
X4	6
Y4	4

In the Parameter Data Section of the file, this definition would look like:

1,8HSTAND,,8,60,65,11,0,4,,4,8,,8,0,1,2,4,,6,4....

#### 4.74 TEXT FONT DEFINITION ENTITY (TYPE 310)

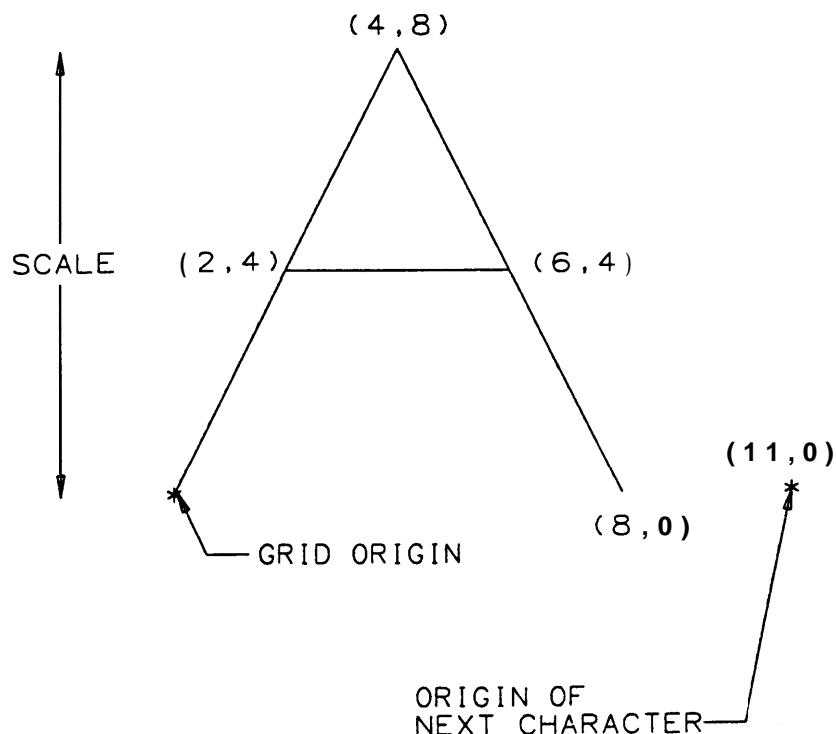


Figure 119. Example of a Character Definition

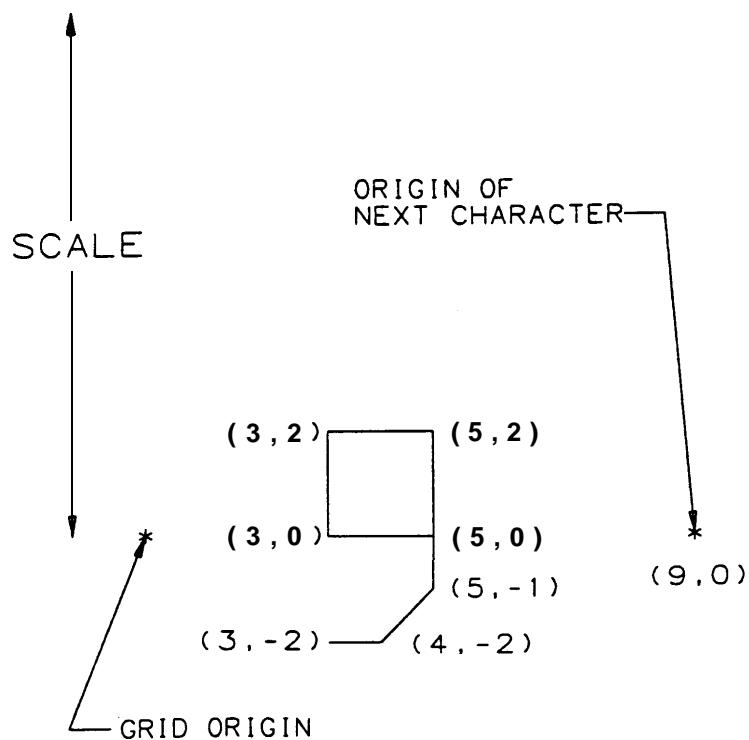


Figure 120. Example of a Character Definition Including Descenders

## 4.75 Text Display Template Entity (Type 312)

The Text Display Template is used to set parameters for display of information which has been logically included in another entity as a parameter value. The text to be displayed is derived from the indicated parameter value of the entity which is pointing to the Text Display Template. In addition to string constants, the parameter values to be displayed may be integer, real or logical constants. The parameter value shall be processed into text as defined in [Section 2.2.2](#) and according to the processing rules presented in the following. Furthermore, the pointer to the Text Display Template may be explicitly defined in the pointing entity description or it may be an implicitly defined additional pointer available with all entities ([see Section 2.2.4.5.2](#)). When the pointer is explicitly defined (e.g., Class 6 of the Flow Associativity (Type 402, Form 18)), the specified explicit parameter shall be processed for display. (In the example cited, the parameter to be displayed is the first flow name listed in Class 5.) When, on the other hand, the pointer to the template is one of the additional pointers (as defined in [Section 2.2.4.5.2](#)) the parameter to be processed for display is the first information value in the pointing entity. (The information value to be displayed shall be data as opposed to meta-data. For example, if the first parameter of the pointing entity is the number of property values, that should be skipped and the first actual property value should be processed and displayed.) For a more detailed description of the parameters, see the General Note Entity ([Type 212](#)).

### Processing Rules for Text Display

The following rules are provided for the sake of uniformity in case postprocessed files are to be tested for identical textual presentation of Integer, Real, and Logical values. Strict application of these rules may lead to overwriting other entities, impaired legibility, and reduced visual association with pertinent nearby structures. Consequently, local site standards and conventions may supersede these rules whenever identical textual presentation is not required.

**Integer Values.** All integers shall be processed such that the resultant text string contains only the valid decimal digits (0–9) and a sign. A leading minus sign shall denote negative values. A leading plus sign shall be provided for positive values.

**Real Values.** All reals shall be processed so that the resulting string represents a valid approximation of the number in scientific notation. The decimal point shall appear immediately to the left of the most significant digit. The decimal point shall be preceded by a zero. A leading minus sign shall denote negative values. A leading plus sign shall be provided for positive values.

**Logical Values.** The logical value .TRUE. (indicated by a 1 in the file) shall be processed as if the string constant “4HTRUE” was to be displayed. The logical value .FALSE. (indicated by a 0 in the file) shall be processed as if the string constant “5HFALSE” was to be displayed.

#### Absolute Text Display Template (Form 0).

ECO630

This form of the Text Display Template specifies the parameters for the text block at the specified starting point.

#### Incremental Text Display Template (Form 1).

ECO630

The Incremental Text Display Template (Form 1) specifies text block parameters for a block whose starting point is located incrementally from the text origin of the entity referencing it.

## 4.75 TEXT DISPLAY TEMPLATE ENTITY (TYPE 312)

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
312	$\Rightarrow$	$< n.a. >$	$< n.a. >$	#, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	0, $\Rightarrow$	??000200	<b>D #</b>
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
312	$< n.a. >$	#, $\Rightarrow$	#	0				#	<b>D # + 1</b>

### Absolute Text Display Template (Form 0)

ECO630

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	CBW	Real	Character box width
2	CBH	Real	Character box height
3	FC	Integer	Font code (Default = 1) or Pointer to the DE of the Text Font Definition Entity if negative
4	SL	Real	Slant angle (Radians) (Default = $\pi/2$ )
5	A	Real	Rotation angle (Radians)
6	M	Integer	Mirror Flag
7	VH	Integer	Rotate internal text flag
8	XS	Real	Coordinates of lower left corner of first character box
9	YS	Real	
10	ZS	Real	

Additional pointers as required (see Section 2.2.4.5.2).

### Incremental Text Display Template (Form 1).

ECO630

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	CBW	Real	Character box width
2	CBH	Real	Character box height
3	FC	Integer	Font code (Default = 1) or Pointer if negative
4	SL	Real	Slant angle (Radians) (Default = $\pi/2$ )
5	A	Real	Rotation angle (Radians)
6	M	Integer	Mirror Flag
7	VH	Integer	Rotate internal text flag
8	DXS	Real	Increment in X from X coordinate found in parent entity
9	DYS	Real	Increment in Y from Y coordinate found in parent entity
10	DZS	Real	Increment in Z from Z coordinate found in parent entity

Additional pointers as required (see Section 2.2.4.5.2).

## 4.76 COLOR DEFINITION ENTITY (TYPE 314)

### 4.76 Color Definition Entity (Type 314)

The Color Definition Entity specifies the relationship of the primary (red, green, and blue) colors to ECO630 the intensity level of the respective graphics devices as a percent of the full intensity range.

These red, green, blue coordinates (RGB) can be readily transformed to cyan, magenta, yellow (CMY) and to hue, lightness, saturation (HLS) using transformations that are given in [Appendix D](#).

The preprocessor shall select one of the Color Numbers ([see Section 2.2.4.13](#)) and place the value ECO630 in Directory Entry Field 13 of the Color Definition Entity. This value shall be the closest functional equivalent, or the most visually similar. The value shall be used by postprocessors which cannot support the Color Definition Entity.

ECO630

#### Directory Entry

(1) Entity Type Number 314	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level <i>&lt; n.a. &gt;</i>	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **0002**	(10) Sequence Number D #
(11) Entity Type Number 314	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number 1-8	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	CC1	Real	First color coordinate (red) as a percent of full intensity (range 0.0 to 100.0)
2	CC2	Real	Second color coordinate (green) as a percent of full intensity (range 0.0 to 100.0)
3	CC3	Real	Third color coordinate (blue) as a percent of full intensity (range 0.0 to 100.0)
4	CNAME	String	Color name; this is an optional character string which may contain some verbal description of the color. If the color name is not provided and additional pointers are required, the color name shall be defaulted.

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.77 Units Data Entity (Type 316)‡

ECO630

‡The Units Data Entity has not been tested. See Section 1.9.

This entity stores data about a model's fundamental units. The first entry (NP) is the number of data strings in the PD. The entity then contains records, each of which contains a pair of string variables and a real scale factor. The first variable contains the unit to be set, the second variable contains one of the valid entries, and the third variable contains a scale factor to be applied to the unit.

If the real data associated with any entity is not expressed in the units of length defined in the global ECO630 section or the SI (MKSA) defaults for the Tabular Data Entity ([Type 406, Form 11](#)), a Units Data Entity ([Type 316](#)) shall be attached to the data entity via a property pointer.

There are seven base units and two supplementary units from which all other units can be derived. ECO630 Therefore, the value of TYP in the above parameter data shall be chosen from the following list of valid TYP strings:

TYP	Indicates unit of
LENGTH	Length
MASS	Mass
TIME	Time
CURRENT	Electric Current
TEMPERATURE	Thermodynamic Temperature
AMOUNT	Amount of Substance
INTENSITY	Luminous Intensity
PLANE	Plane Angle
SOLID	Solid Angle

A given TYP determines which of the following lists shall be used to specify the particular units. ECO630

Valid VAL strings for TYP = LENGTH:

VAL	Description
A	Angstrom
AU	Astronomical Unit
FT	Foot
IN	Inch
LY	Light Year
M	Meter
UM	Micron
MIL	Mil (.001 Inch)
MI	Mile
KN	Nautical Mile
Y	Yard

#### **4.77 UNITS DATA ENTITY (TYPE 316)‡**

Valid VAL strings for TYP = MASS:

VAL	Description
C	Carat
DR	Dram
GA	Grain
KG	Kilogram
MT	Metric Tonne
OU	Ounce
LB	Pound
S	Slug

Valid VAL strings for TYP = TIME:

VAL	Description
D	Day
HR	Hour
M	Minute
S	Second
W	Week
Y	Year

Valid VAL strings for TYP = CURRENT:

VAL	Description
A	Ampere

Valid VAL strings for TYP = TEMPERATURE:

VAL	Description
C	Centigrade
F	Fahrenheit
K	Kelvin
R	Rankine

Valid VAL strings for TYP = AMOUNT:

VAL	Description
M	Mole

Valid VAL strings for TYP = INTENSITY:

VAL	Description
C	Candela

Valid VAL strings for TYP = PLANE:

VAL	Description
D	Degree
G	Grad
M	Minute
R	Radian
REV	Revolution
S	Second

Valid VAL strings for TYP = SOLID:

VAL	Description
C	Steradian

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
316	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**0002**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
316	< n.a. >	< n.a. >	#	0				#	D # + 1

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of units defined by this entity
2	TYP(l)	String	Type of first unit being defined
3	VAL(l)	String	Units of first unit being defined
4	SF(1)	Real	A multiplicative scale factor to be applied to the first unit
:	:	:	
-1+3*NP	TYP(NP)	String	Type of last unit being defined
3*NP	VAL(NP)	String	Units of last unit being defined
1+3*NP	SF(NP)	Real	A multiplicative scale factor to be applied to the last unit

Additional pointers as required (see Section 2.2.4.5.2).

**4.78 Network Subfigure Definition Entity (Type 320)**

The Network Subfigure Definition Entity supports multiple instantiation of a defined collection of ECO630 entities, similar to the Subfigure Definition Entity ([Type 308](#)). It differs from the ordinary subfigure definition in that it defines a specialized subfigure, one whose instances may participate in networks. To participate in a network, points of connection (Connect Point Entity ([Type 132](#))) shall be defined (see indices 7+NA and following) and instanced along with the subfigure. Often, products which contain networks are designed first as schematics (showing the logical connections or relationships), which are then converted into the designs of the physical products. Whenever both a logical design and a physical design are present in the same file, the processor needs a way to determine which entities belong in which design. The Type Flag field (index 4+NA) implements this distinction. Other fields, such as NAME and DEPTH, function in exactly the same manner as in the Subfigure Definition Entity ([Type 308](#)).

There is a direct relationship between the points of connection in the Network Subfigure Definition Entity ([Type 320](#)) and the Network Subfigure Instance Entity ([Type 420](#)). The number of associated (child) Connect Point Entities ([Type 132](#)) in the instance shall match the number in the definition, their order shall be identical, and any unused points of connection in the instance shall be indicated by a null (zero) pointer.

Note: The depth of the subfigure is inclusive of both the Network Subfigure Definition Entity ([Type 320](#)) and the ordinary Subfigure Definition Entity ([Type 308](#)). Thus, the two may be nested but shall indicate that in the depth parameter.

## 4.78 NETWORK SUBFIGURE DEFINITION ENTITY (TYPE 320)

### Directory Entry

(1) Entity Type Number 320	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View < n.a. >	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number *??02??	(10) Sequence Number D #
(11) Entity Type Number 320	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

Note: When the Hierarchy is set to Global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, Color Number, Level, View, and Blank Status.

### Parameter Data

ECO650  
ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DEPTH	Integer	Depth of subfigure (indicating the amount of nesting)
2	NAME	String	Subfigure name
3	NA	Integer	Number of associated child entities in the subfigure, exclusive of primary reference designator and Connect Points
4	APTR(1)	Pointer	Pointer to the DE of the first associated entity
:	:	:	
3+NA	APTR(NA)	Pointer	Pointer to the DE of the last associated entity
4+NA	TF	Integer	Type Flag: 0 = not specified 1 = logical 2 = physical
5+NA	PRD	String	Primary reference designator
6+NA	DPTR	Pointer	Pointer to the DE of the primary reference designator Text Display Template, or null. If null, no Text Display Template specified.
7+NA	NC	Integer	Number of associated (child) Connect Point Entities
8+NA	CPTR(1)	Pointer	Pointer to the DE of the first associated Connect Point Entity, or zero
:	:	:	
7+NC+NA	CPTR(NC)	Pointer	Pointer to the DE of the last associated Connect Point Entity, or zero

Additional pointers as required (see Section 2.2.4.5.2).

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

### 4.79 Attribute Table Definition Entity (Type 322)

The Attribute Table Definition Entity supports the concept of a well-defined collection of attributes ECO630 ([Section 3.6.7](#)), whether it is a table or a single row of attributes. The entity provides a template for the instance of attribute tables ([see Section 4.141](#)), or for the combination of template and instance ([see Section 4.79 Form 1 and Form 2](#)). The entity includes a table name (NAME), and for each attribute, an attribute type (AT), data type (AVDT), and a count (AVC).

**Definitions.** The following definitions and abbreviations are used in the entity description:

**Attribute List Type (ALT).** The designated attribute list contains the names (or descriptors) ECO630 of each attribute type appearing in the attribute table. Within each attribute list the integer numbers representing the attributes shall be unique. As an aid to implementors and users, the attribute list also may contain useful supporting information such as suggested units, suggested data types, a footnote for reference, or a range of acceptable values.

<b>Value</b>	<b>Designated List</b>	ECO649
0	See Name Property Entity ( <a href="#">Type 406, Form 15</a> ) for the name of the specific engineering standard that defines the attribute list	
1	General attribute list	
2	Electrical attribute list ( <a href="#">see Table 11</a> )	
3	AEC attribute list ( <a href="#">see Table 12</a> )	
4	Process plant attribute list ( <a href="#">see Table 13</a> )	
5	Electrical and LEP manufacturing attribute list ( <a href="#">see Table 14</a> )	
6-5000	other application areas	
5001-9999	implementor defined lists	

**Attribute Type (AT).** Each integer number designates an attribute type defined in the designated ECO630 attribute list. The number shall exist in the list.

**Attribute Value Data Type (AVDT).** Each attribute has one or more associated value types ECO630 which may be presented in this entity using one of the following data type indicators (there is no default - a value shall be specified):

<b>Value</b>	<b>Data Type</b>
0	No value
1	Integer
2	Real
3	Character string
4	Pointer
5	Not used
6	Logical

Note that these are the same types and are in the same order as the constants described in [Section 2.2.2](#).

**Attribute Value Count (AVC).** The number of values ([Form 0](#) or [Form 1](#)) or pairs of values and pointers ([Form 2](#)) which follow. The default count is 1. A count of zero implies that values exist and will be recorded at some future time but are currently unknown. In this special case, no values or pairs of values and pointers are required.

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

**Attribute Value (AV).** Each attribute contains zero, one or more values as counted by the AVC field. Each AV is specified in the data type field indicated by the current AVDT.

**Attribute Value Pointer (AW).** A pointer to a Text Display Template Entity ([Type 312](#)) may ECO630 be associated with each AV. If the pointer contains a non-null value, the AV is displayed by the Text Display Template at either the absolute location given (Form 0 or by combining the increment given (Form 1) with the location of the parent entity (to which this entity is attached, if it is dependent).

### Attribute Table Definition (Form 0).

This form of the entity is for the definition only of a group of attributes (name, type, and count). It is to be used for the one-to-many case where there will be many instances of a single attribute table definition (*i.e.*, the file shall contain one or more Attribute Table Instance Entities referencing the Attribute Table Definition Entity).

### Attribute Table Definition (Form 1).

This form of the entity is to be used for the one-to-one case where there will be few, or only one, instance of the group of attributes. (The attribute values shall follow their respective attribute definitions.)

### Attribute Table Definition (Form 2).

This form is similar to Form 1 with the addition of a pointer to a Text Display Template Entity following each attribute value.

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

### Directory Entry

(1) Entity Type Number 322	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View < n.a. >	(7) Information Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number **0002??	(10) Sequence Number D #
(11) Entity Type Number 322	(12) Line Weight #	(13) Color Number #,⇒	(14) Paramet er Line Count #	(15) Form Number 0-2	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** When the Hierarchy is set to Global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, Color Number, Level, View, and Blank Status.

### Attribute Table Definition (Form 0).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NAME	String	Attribute Table name, or comment (default = blank, no name)
2	ALT	Integer	Attribute list type
3	NA	Integer	Number of attributes (first attribute definition)
4	AT(1)	Integer	First attribute type
5	AVDT(1)	Integer	First attribute value data type
6	AVC(1)	Integer	First attribute value count
:	:	:	

Let M = 3\*NA

(last attribute definition)

M+1	AT(NA)	Integer	Last attribute type
M+2	AVDT(NA)	Integer	Last attribute value data type
M+3	AVC(NA)	Integer	Last attribute value count

Additional pointers as required ([see Section 2.2.4.5.2](#)).

### Attribute Table Definition (Form 1).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NAME	String	Attribute Table name, or comment (default=blank, no name)
2	ALT	Integer	Attribute list type
3	NA	Integer	Number of attributes (first attribute definition and values)
4	AT(1)	Integer	First attribute type
5	AVDT(1)	Integer	First attribute value data type
6	AVC(1)	Integer	First attribute value count
7	AV(1,1)	Variable	First attribute value
:	:	:	

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

6+AVC(1)	AV(1, AVC(1))	Variable	Last attribute value
:	:	:	
Let M = 3*NA + AVC(1) + ... + AVC(NA-1) (last attribute definition and values)			
M+1	AT(NA)	Integer	Last attribute type
M+2	AVDT(NA)	Integer	Last attribute value data type
M+3	AVC(NA)	Integer	Last attribute value count
M+4	AV(NA,1)	Variable	First attribute value
:	:		
M+4+AVC(NA)	AV(NA,AVC(NA))	Variable	Last attribute value

Additional pointers as required ([see Section 2.2.4.5.2](#)).

### Attribute Table Definition (Form 2).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NAME	String	Attribute Table name, comment
2	ALT	Integer	Attribute list type
3	NA	Integer	
	(first attribute definition)		
4	AT(1)	Integer	
5	AVDT(1)	Integer	First attribute value data type
6	AVC(1)	Integer	
7	AV(1,1)	Variable	
8	AVP(1,1)	Pointer	Pointer to the DE of the Text Display Template
:	:	:	
6+AVC(1)	AV(1,AVC(1))	Variable	Last attribute value
7+AVC(1)	AVP(1,AVC(1))	Pointer	Pointer to the DE of the Text Display Template
:	:	:	
Let M = 3*NA + 2*(AVC(1) + ... + AVC(NA-1)) (last attribute definition)			
M+1	AT(NA)	Integer	Last attribute type
M+2	AVDT(NA)	Integer	Last attribute value data type
M+3	AVC(NA)	Integer	Last attribute value count
M+4	AV(NA,1)	Variable	First attribute value
M+5	AVP(NA,1)	Pointer	Pointer to the DE of the Text Display Template
:	:	:	
M+4+AVC(NA)	AV(NA,AVC(NA))	Variable	Last attribute value
M+5+AVC(NA)	AVP(NA,AVC(NA))	Pointer	Pointer to the DE of the Text Display Template

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2)

No.	Definition	Symbol	Unit	Type	Ref.
1	Access time, memory		second	Real	[IEEE84]
2	Accuracy (percent of full scale)			Real	[IEEE84]
3	Ambient air temperature	$T_a$	C	Real	[MIL195]
4	Amplification	$A$		Real	[IEEE84]
5	Amplification Factor	$A_f$		Real	[IEEE84]
6	Angle (of a waveform)		degree	Real	[IEEE84]
7	Anode voltage	$V_a$	volt	Real	[IEEE84]
8	Async. input pulse width, minimum	$A_{PW}$	second	Real	[MIL133]
9	Automatic gain control	AGC		Real	[MIL133]
10	Average forward-current rating		ampere	Real	[MIL195]
11	Average gate power dissipation	$W_{gpd}$	watt	Real	[MIL195]
12	Average gate-power-dissipation rating		watt	Real	[MIL195]
13	Backlash		degree	Real	[IEEE84]
14	Bandwidth	BW	hertz	Real	[MIL133]
15	Base resistance	$R_B$	ohm	Real	[IEEE84]
16	Base current	$I_B$	ampere	Real	[MIL195]
17	Base current, instantaneous total	$i_B$	ampere	Real	[MIL195]
18	Base spreading resistance	$r_b$	ohm	Real	[MIL195]
19	Base supply voltage	$V_{BB}$	volt	Real	[MIL195]
20	Base to emitter voltage	$V_B$	volt	Real	[MIL195]
21	LEP thickness		inch	Real	[IPCT85]
22	Breakdown voltage	$V_{BR}$	volt	Real	[MIL133]
23	Capacitance	$C$	farad	Real	[IEEE84]
24	Carrier frequency	$C_f$	hertz	Real	[IEEE84]
25	Case temperature	$T_C$	C	Real	[MIL195]
26	Cathode current	$I_c$	ampere	Real	[IEEE84]
27	Circuit commutated turn-off time		second	Real	[MIL195]
28	Clearance		inch	Real	[IEEE84]
29	Clock level transition time	$t_{TC}$	second	Real	[MIL133]
30	Clock pulse width, minimum	$C_{PW}$	second	Real	[MIL133]
31	Clock repetition rate	$C_{RR}$	hertz	Real	[MIL133]
32	Collector current	$I_C$	ampere	Real	[MIL195]
33	Collector current, instantaneous total	$i_C$	ampere	Real	[MIL195]
34	Collector cut-off current	$I_{CEX}$	ampere	Real	[MIL133]
35	Collector efficiency (percent)			Real	[MIL195]
36	Collector power dissipation	$P_C$	watt	Real	[MIL195]
37	Collector to base voltage	$V_{CB}$	volt	Real	[MIL195]
38	Collector to emitter voltage	$V_{CE}$	volt	Real	[MIL195]
39	Common-mode input voltage	$V_{ICR}$	volt	Real	[MIL133]
40	Common-mode output voltage	$V_{OC}$	volt	Real	[MIL133]

ECO649

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
41	Common-mode rejection ratio	$C_{MRR}$		Real	[MIL133]
42	Common-mode voltage amplification	$AVC$		Real	[MIL133]
43	Conductance	$G$	siemens	Real	[IEEE84]
44	Conductor spacing		inch	Real	[IPCT85]
45	Conductor width		inch	Real	[IPCT85]
46	Conversion efficiency (percent)			Real	[MIL195]
47	Conversion time, analog-to-digital		second	Real	[IEEE84]
48	Coupling coefficient			Real	[IEEE84]
49	Critical anode voltage		volt	Real	[IEEE84]
50	Current	$I$	ampere	Real	[IEEE84]
51	Current capacity	$I_c$	ampere	Real	[IEEE84]
52	Current limit	$I_l$	ampere	Real	[IEEE84]
53	Current rating	$I_r$	ampere	Real	[IEEE84]
54	Cut-off grid voltage	$V_{gc(off)}$	volt	Real	[IEEE84]
55	Cutoff current	$I_c$	ampere	Real	[MIL195]
56	Data rate		hertz	Real	[IEEE84]
57	Delay time	$t_d$	second	Real	[MIL195]
58	Dielectric constant	$K$		Real	[IEEE84]
59	Dielectric strength		volt/inch	Real	[IEEE84]
60	Differential control voltage		volt	Real	[IEEE84]
61	Differential input impedance	$Z_{id}$	ohm	Real	[MIL133]
62	Differential output impedance	$Z_{od}$	ohm	Real	[MIL133]
63	Differential output voltage		volt	Real	[IEEE84]
64	Differential voltage amplification	$A_{vd}$		Real	[MIL133]
65	Distortion (percent)			Real	[IEEE84]
66	Drain current	$I_D$	ampere	Real	[MIL195]
67	Drain cutoff current	$I_{D(off)}$	ampere	Real	[MIL195]
68	Drain supply voltage	$V_{DD}$	volt	Real	[MIL195]
69	Drain to gate voltage	$V_{DG}$	volt	Real	[MIL195]
70	Drain to source voltage	$V_{DS}$	volt	Real	[MIL195]
71	Drain to substrate voltage	$V_{Ds}$	volt	Real	[MIL195]
72	Drift (percent of full scale)			Real	[IEEE84]
73	Driving point impedance	$Z_{dp}$	ohm	Real	[IEEE84]
74	Dynamic impedance	$Z_D$	ohm	Real	[MIL195]
75	Efficiency (percent)			Real	[IEEE84]
76	Emission current	$I_e$	ampere	Real	[IEEE84]
77	Emission efficiency (percent)			Real	[IEEE84]
78	Emitter current	$I_E$	ampere	Real	[MIL195]
79	Emitter current, instantaneous total	$i_E$	ampere	Real	[MIL195]
80	Emitter supply voltage	$V_{EE}$	volt	Real	[MIL195]

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
81	Emitter to base voltage	$V_{EB}$	volt	Real	[MIL195]
82	Emitter to collector voltage	$V_{EC}$	volt	Real	[MIL195]
83	External base resistance	$R_B$	ohm	Real	[MIL195]
84	External collector resistance	$R_C$	ohm	Real	[MIL195]
85	External emitter resistance	$R_E$	ohm	Real	[MIL195]
86	Extrapolated unity gain frequency	$f_T$	hertz	Real	[MIL195]
87	Failure rate			Real	[IEEE84]
88	Fall time	$t_f$	second	Real	[MIL195]
89	Field emission				[IEEE84]
90	Figure of merit				[IEEE84]
91	Filament voltage	$V_F$	volt	Real	[IEEE84]
92	Floating potential	$V_{FP}$	volt	Real	[MIL195]
93	Forward blocking voltage	$V_{FBO}$	volt	Real	[MIL195]
94	Forward breakover current	$I_{FBR}$	ampere	Real	[MIL195]
95	Forward breakover voltage	$V_{FBR}$	volt	Real	[MIL195]
96	Forward current	$I_F$	ampere	Real	[MIL195]
97	Forward current, overload	$I_{F(OV)}$	ampere	Real	[MIL195]
98	Forward current, surge peak	$I_{FSM}$	ampere	Real	[MIL195]
99	Forward gate current	$I_{GF}$	ampere	Real	[MIL195]
100	Forward gate-to-source breakdown voltage	$V_{FBRGS}$	volt	Real	[MIL195]
101	Forward gate-to-source voltage	$V_{FGS}$	volt	Real	[MIL195]
102	Forward power dissipation	$P_F$	watt	Real	[MIL195]
103	Forward recovery time	$t_{fr}$	second	Real	[MIL195]
104	Forward voltage	$V_F$	volt	Real	[MIL195]
105	Frequency	$F$	hertz	Real	[IEEE84]
106	Gain			Real	[IEEE84]
107	Gate controlled turn-off time	$t_{gc(off)}$	second	Real	[MIL195]
108	Gate current	$I_G$	ampere	Real	[MIL195]
109	Gate supply voltage	$V_{GG}$	volt	Real	[MIL195]
110	Gate to source voltage	$V_{GS}$	volt	Real	[MIL195]
111	Gate trigger current	$I_{gt}$	ampere	Real	[MIL195]
112	Gate trigger voltage	$V_{gt}$	volt	Real	[MIL195]
113	Gate turn-off current	$I_{gt(off)}$	ampere	Real	[MIL195]
114	Gate turn-off voltage	$V_{gt(off)}$	volt	Real	[MIL195]
115	Gate voltage	$V_g$	volt	Real	[MIL195]
116	Gate-to-source cutoff voltage	$V_{gs(off)}$	volt	Real	[MIL195]
117	Gate-to-source threshold voltage	$V_{gst}$	volt	Real	[MIL195]
118	Gate-to-source voltage	$V_{gs}$	volt	Real	[MIL195]
119	Grid control ratio			Real	[IEEE84]
120	Grid current	$I_g$	ampere	Real	[IEEE84]

## ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
121	Grid voltage	$V_g$	volt	Real	[IEEE84]
122	High clock level	$V_{CH}$	volt	Real	[MIL133]
123	High level input current	$I_{IH}$	ampere	Real	[MIL133]
124	High level node input current	$I_{INH}$	ampere	Real	[MIL133]
125	High level node input voltage	$V_{INH}$	volt	Real	[MIL133]
126	High level output current	$I_{OH}$	ampere	Real	[MIL133]
127	High level output voltage	$V_{OH}$	volt	Real	[MIL133]
128	High level supply current drain	$I_{CCH}$	ampere	Real	[MIL133]
129	Holding current	$I_H$	ampere	Real	[MIL195]
130	Impedance	$Z$	ohm	Real	[IEEE84]
131	Incremental resistance	$R_{inc}$	ohm	Real	[IEEE84]
132	Inductance	$L$	henry	Real	[IEEE84]
133	Input bias current	$I_{IB}$	ampere	Real	[MIL133]
134	Input bias current temp. sensitivity	$I_{IB/T}$	amp/deg C	Real	[MIL133]
135	Input impedance	$Z_i$	ohm	Real	[IEEE84]
136	Input offset current	$I_{IO}$	ampere	Real	[MIL133]
137	Input offset current temp. sensitivity	$I_{IO/T}$	amp/deg C	Real	[MIL133]
138	Input offset voltage	$V_{IO}$	volt	Real	[MIL133]
139	Input offset voltage temp. sensitivity	$V_{IO/T}$	volt/deg C	Real	[MIL133]
140	Input signals timing relationships	$I_{TR}$		Real	[MIL133]
141	Insulation resistance	$R_i$	ohm	Real	[IEEE84]
142	Interelectrode capacitance	$C_{IE}$	farad	Real	[IEEE84]
143	Interrupting current	$I_i$	ampere	Real	[IEEE84]
144	Ionization time		second	Real	[IEEE84]
145	Junction temperature	$T_J$	deg C	Real	[MIL195]
146	Knee impedance	$Z_k$	ohm	Real	[MIL195]
147	Knee voltage	$V_K$	volt	Real	[MIL195]
148	Latching current	$I_l$	ampere	Real	[MIL195]
149	Leakage current	$I_{lk}$	ampere	Real	[IEEE84]
150	Limiting current	$I_L$	ampere	Real	[MIL195]
151	Limiting voltage	$V_L$	volt	Real	[MIL195]
152	Load immittance			Real	[MIL195]
153	Logic level high	$H$		Real	[IEEE84]
154	Logic level low	$L$		Real	[IEEE84]
155	Low clock level	$V_{CL}$	volt	Real	[MIL133]
156	Low level input current	$I_{IL}$	ampere	Real	[MIL133]
157	Low level node input current	$I_{INL}$	ampere	Real	[MIL133]
158	Low level node input voltage	$V_{INL}$	volt	Real	[MIL133]
159	Low level output current	$I_{OL}$	ampere	Real	[MIL133]
160	Low level output voltage	$V_{OL}$	volt	Real	[MIL133]

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
161	Low level supply current drain	$I_{CCL}$	ampere	Real	[MIL133]
162	Luminous energy	$c_d$		Real	[MIL195]
163	Material			String	[IEEE84]
164	Maximum frequency of oscillation	$f_{max}$	hertz	Real	[MIL195]
165	Maximum surge current, nonrepetitive	$I_{smax}$	ampere	Real	[MIL195]
166	Max. surge-current rating, nonrepetitive		ampere	Real	[MIL195]
167	Minimum on-voltage	$V_{min}$	volt	Real	[MIL195]
168	Mode of operation			String	[IEEE84]
169	Moisture-resistant			Logical	[IEEE84]
170	Mutual impedance	$Z_m$	ohm	Real	[IEEE84]
171	Mutual conductance	$G_m$	siemens	Real	[IEEE84]
172	N-channel field-effect transistor	$N_{fet}$		Real	[MIL195]
173	Negative logic			Logical	[IEEE84]
174	Noise figure	$N_F$	dB	Real	[MIL133]
175	Noise margin	$V_N$	volt	Real	[MIL133]
176	Noise temperature	$T_N$	deg C	Real	[MIL195]
177	Nonlinearity (percent of full scale)			Real	[IEEE84]
178	Nonrepetitive peak off-state voltage	$V_{DSM}$	volt	Real	[MIL195]
179	Nonrepetitive peak reverse current	$I_{RSM}$	ampere	Real	[MIL195]
180	Nonrepetitive peak reverse voltage	$V_{RSM}$	volt	Real	[MIL195]
181	Off-state current	$I_D$	ampere	Real	[MIL195]
182	Off-state voltage	$V_D$	volt	Real	[MIL195]
183	Offset error (percent of full range)			Real	[IEEE84]
184	On-state current	$I_T$	ampere	Real	[MIL195]
185	On-state drain current	$I_{D(on)}$	ampere	Real	[MIL195]
186	On-state drain-to-source voltage	$V_{DS(on)}$	volt	Real	[MIL195]
187	On-state voltage	$V_T$	volt	Real	[MIL195]
188	Open loop gain	$A_v$		Real	[IEEE84]
189	Operating temperature	$T_{op}$	deg C	Real	[MIL195]
190	Operator (the person's name or initials)			String	[IEEE84]
191	Orientation			String	[IEEE84]
192	Output impedance	$Z_o$	ohm	Real	[IEEE84]
193	Output leakage current high	$i_{ozh}$	ampere	Real	[IEEE84]
194	Output current		ampere	Real	[IEEE84]
195	Output frequency	$F_o$	hertz	Real	[IEEE84]
196	Output frequency high	$F_{oh}$	hertz	Real	[IEEE84]
197	Output frequency low	$F_{ol}$	hertz	Real	[IEEE84]
198	Output leakage current low	$i_{ozl}$	ampere	Real	[IEEE84]
199	Output offset voltage	$V_{OO}$	volt	Real	[MIL133]
200	Output polarity			String	[IEEE84]

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
201	Output pulse duration	$t_{op}$	second	Real	[IEEE84]
202	Output pulse duration high	$t_{oph}$	second	Real	[IEEE84]
203	Output pulse duration low	$t_{opl}$	second	Real	[IEEE84]
204	Output short circuit current	$I_{OS}$	ampere	Real	[MIL133]
205	Output swing bandwidth, maximum	$B_{OM}$	hertz	Real	[MIL133]
206	Output type			String	[IEEE84]
207	Output voltage	$V_o$	volt	Real	[IEEE84]
208	Output voltage high	$V_{oh}$	volt	Real	[IEEE84]
209	Output voltage low	$V_{ol}$	volt	Real	[IEEE84]
210	Output voltage regulation ((high-low)/nominal, percent)			Real	[IEEE84]
211	Output voltage sensitivity to temperature	$V_{OPP}$	$(degC)^{-1}$	Real	[IEEE84]
212	Output voltage swing, maximum		volt	Real	[MIL133]
213	Outputs			String	[IEEE84]
214	Over-voltage sense			String	[IEEE84]
215	Overall average noise figure	$N_{foa}$		Real	[MIL195]
216	Overload recovery time	$t_{or}$	second	Real	[MIL133]
217	P-channel field-effect transistor	$P_{fet}$		Real	[MIL195]
218	Parallel entry			String	[IEEE84]
219	Parametric electrical test required			Logical	[IEEE84]
220	Peak current rating		ampere	Real	[IEEE84]
221	Peak forward-blocking voltage rating		volt	Real	[MIL195]
222	Peak forward-current rating, repetitive		volt	Real	[MIL195]
223	Peak forward-voltage rating		volt	Real	[MIL195]
224	Peak gate current	$I_{GP}$	ampere	Real	[MIL195]
225	Peak gate power dissipation	$W_{GP}$	watt	Real	[MIL195]
226	Peak gate voltage	$V_{GP}$	volt	Real	[MIL195]
227	Peak gate-current rating		ampere	Real	[MIL195]
228	Peak gate-power-dissipation rating		watt	Real	[MIL195]
229	Peak gate-voltage rating		volt	Real	[MIL195]
230	Peak repetitive on-state current	$I_p$	ampere	Real	[MIL195]
231	Phase margin	$\phi_m$	degree	Real	[MIL133]
232	Phase shift	$\phi_s$	degree	Real	[IEEE84]
233	Photoelectric current	$I_\lambda$	ampere	Real	[IEEE84]
234	Pin size		inch	Real	[IEEE84]
235	Plate efficiency (percent)			Real	[IEEE84]
236	Polarity			String	[IEEE84]
237	Polarity on stud			String	[IEEE84]
238	Positive logic			Logical	[IEEE84]
239	Power	$W$	watt	Real	[IEEE84]
240	Power dissipation	$P_D$	watt	Real	[MIL133]

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
241	Power gain	$G_p$	dB	Real	[MIL133]
242	Power per diode	$W_D$	watt	Real	[IEEE84]
243	Power rating		watt	Real	[IEEE84]
244	Power supply high	$W_{sh}$	volt	Real	[IEEE84]
245	Power supply low	$W_{sl}$	volt	Real	[IEEE84]
246	Power supply rejection ratio	$P_{SRR}$		Real	[MIL133]
247	Procurement spec # 1 MIL-M-38510/xxx			String	[IEEE84]
248	Procurement spec # 2 desc xxxx			String	[IEEE84]
249	Programmable			Logical	[IEEE84]
250	Programmable gain			Real	[IEEE84]
251	Propagation delay time		second	Real	[IEEE84]
252	Propagation delay time, high to low level	$t_{PHL}$	second	Real	[MIL133]
253	Propagation delay time, low to high level	$t_{PLH}$	second	Real	[MIL133]
254	Propagation delay to output high	$t_{pdh}$	second	Real	[IEEE84]
255	Propagation delay to output low	$t_{pdl}$	second	Real	[IEEE84]
256	Protocol			String	[IEEE84]
257	Pulse storage time	$t_{ps}$	second	Real	[MIL195]
258	Pulse time	$t_p$	second	Real	[MIL195]
259	Pulse width	$t_w$	second	Real	[IEEE84]
260	Quiescent input voltage	$V_I$	volt	Real	[MIL133]
261	Quiescent output voltage	$V_O$	volt	Real	[MIL133]
262	Radiant energy	$J_\omega$	joule	Real	[MIL195]
263	Radiation hardened			Logical	[IEEE84]
264	Reach-through voltage		volt	Real	[MIL195]
265	Reactance	$X$	ohm	Real	[IEEE84]
266	Receiver input impedance	$Z_i$	ohm	Real	[IEEE84]
267	Rectification efficiency (percent)			Real	[MIL195]
268	Rectified voltage	$V_{rect}$	volt	Real	[IEEE84]
269	Register number			String	[IEEE84]
270	Register types			String	[IEEE84]
271	Regulator current	$I_S$	ampere	Real	[MIL195]
272	Regulator voltage	$V_S$	volt	Real	[MIL195]
273	Regulator impedance	$Z_s$	ohm	Real	[MIL195]
274	Repetitive peak reverse voltage	$V_{RRM}$	volt	Real	[MIL195]
275	Repetitive peak off-state current	$I_{DRM}$	ampere	Real	[MIL195]
276	Repetitive peak off-state voltage	$V_{DRM}$	volt	Real	[MIL195]
277	Repetitive peak on-state current	$I_{TRM}$	ampere	Real	[MIL195]
278	Repetitive peak reverse voltage	$V_{RRM}$	volt	Real	[MIL195]
279	Repetitive peak-reverse voltage rating		volt	Real	[MIL195]
280	Resettable			Logical	[IEEE84]

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
281	Resistance	$R$	ohm	Real	[IEEE84]
282	Resistance tolerance, high (percent)			Real	[IEEE84]
283	Resistance tolerance, low (percent)			Real	[IEEE84]
284	Resistance value (measured)		ohm	Real	[IEEE84]
285	Resistivity (in ohms/square)	$\Omega$	ohm	Real	[IEEE84]
286	Resistor construction class			String	[IEEE84]
287	Resolution in bits			Integer	[IEEE84]
288	Response time	$t_r$	second	Real	[IEEE84]
289	Resultant carry			String	[IEEE84]
290	Reverse blocking current	$I_{RBO}$	ampere	Real	[MIL195]
291	Reverse breakdown current	$I_{rb}$	ampere	Real	[MIL195]
292	Reverse breakdown voltage	$V_{rb}$	volt	Real	[MIL195]
293	Reverse current	$I_R$	ampere	Real	[MIL195]
294	Reverse current, repetitive peak	$I_{RRM}$	ampere	Real	[MIL195]
295	Reverse current, surge peak	$I_{RSM}$	ampere	Real	[MIL195]
296	Reverse gate current	$I_{GR}$	ampere	Real	[MIL195]
297	Reverse gate-to-source voltage	$V_{SG}$	volt	Real	[MIL195]
298	Reverse power dissipation	$P_R$	watt	Real	[MIL195]
299	Reverse recovery current	$I_{RM(REC)}$	ampere	Real	[MIL195]
300	Reverse recovery time	$t_{rr}$	second	Real	[MIL195]
301	Reverse voltage	$V_R$	volt	Real	[MIL195]
302	Rise time	$t_r$	second	Real	[MIL195]
303	Saturation current	$I_{sat}$	ampere	Real	[IEEE84]
304	Saturation resistance	$R_{sat}$	ohm	Real	[MIL195]
305	Saturation voltage	$V_{sat}$	volt	Real	[MIL195]
306	Screening test			String	[IEEE84]
307	Sealed			String	[IEEE84]
308	Settling time	$t_s$	second	Real	[IEEE84]
309	Settling time high	$t_{sh}$	second	Real	[IEEE84]
310	Settling time low	$t_{sl}$	second	Real	[IEEE84]
311	Settling time to p-percent of final max.	$t_{pmax}$	second	Real	[IEEE84]
312	Settling time to p-percent of final min.	$t_{pmin}$	second	Real	[IEEE84]
313	Shaft diameter		inch	Real	[IEEE84]
314	Shift direction (e.g. 4Hleft)			String	[IEEE84]
315	Shift out clock frequency	$F_{shift}$	hertz	Real	[IEEE84]
316	Short circuit			String	[MIL195]
317	Signal to noise ratio	$SNR$	dB	Real	[IEEE84]
318	Single-ended input impedance	$Z_{is}$	ohm	Real	[MIL133]
319	Single-ended input voltage	$V_{ISR}$	volt	Real	[MIL133]
320	Single-ended output impedance	$Z_{os}$	ohm	Real	[MIL133]

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
321	Single-ended voltage amplification	$A_{vs}$		Real	[MIL133]
322	Slew rate	$S_R$	volt/second	Real	[MIL133]
323	Small-signal breakdown impedance	$z_{BR}$	ohm	Real	[MIL195]
324	Small-signal forward impedance	$z_f$	ohm	Real	[MIL195]
325	Small-signal resistance	$r$	ohm	Real	[MIL195]
326	Soft start			String	[IEEE84]
327	Solderability			Logical	[IEEE84]
328	Source current	$I_S$	ampere	Real	[MIL195]
329	Source cutoff current	$I_{S(off)}$	ampere	Real	[MIL195]
330	Source supply voltage	$V_{SS}$	volt	Real	[MIL195]
331	Source terminal			Real	[MIL195]
332	Source-to-substrate voltage	$V_{ss}$	volt	Real	[MIL195]
333	Space charge density			Real	[IEEE84]
334	Standard reference			Real	[MIL195]
335	Standing wave ratio	$SWR$		Real	[MIL195]
336	Static drain-to-source on-state resistance	$R_{sds(on)}$	ohm	Real	[MIL195]
337	Static forward current transfer-ratio	$I_{sft}$		Real	[MIL195]
338	Static input resistance	$R_{in}$	ohm	Real	[MIL195]
339	Static transconductance	$G_M$	siemens	Real	[MIL195]
340	Storage temperature	$T_{STG}$	deg C	Real	[MIL195]
341	Storage time	$t_s$	second	Real	[MIL195]
342	Stored charge	$Q_S$	coulomb	Real	[MIL195]
343	Strip force		lbf	Real	[IEEE84]
344	Strobing input currents	$I_{st}$	ampere	Real	[IEEE84]
345	Strobing pulse width	$t_{st}$	hertz	Real	[IEEE84]
346	Stud torque		inch-lbf	Real	[IEEE84]
347	Subcarrier			Real	[IEEE84]
348	Substrate size length		inch	Real	[IEEE84]
349	Substrate size width		inch	Real	[IEEE84]
350	Substrate terminal			Real	[MIL195]
351	Supply voltage	$V_{CC}$	volt	Real	[MIL133]
352	Surge current	$I_S$	ampere	Real	[MIL195]
353	Surge, on-state current	$I_{TSM}$	ampere	Real	[MIL195]
354	Symbol function			String	[IEEE84]
355	Temperature	$T$	deg C	Real	[IEEE84]
356	Temperature coefficient			Real	[IEEE84]
357	Terminal type			String	[IEEE84]
358	Thermal equilibrium			Real	[MIL195]
359	Thermal noise			Real	
360	Thermal resistance	$R_\theta$	ohm	Real	[MIL195]

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
361	Thermal resistance, case to ambient	$R_{\theta CA}$	ohm	Real	[MIL195]
362	Thermal resistance, junction to ambient	$R_{\theta JA}$	ohm	Real	[MIL195]
363	Thermal resistance, junction to case	$R_{\theta JC}$	ohm	Real	[MIL195]
364	Thermal resistance, junction to lead	$R_{\theta jl}$	ohm	Real	[MIL195]
365	Thermal resistance, junction to reference	$R_{\theta JR}$	ohm	Real	[MIL195]
366	Threshold current	$I_{th}$	ampere	Real	[IEEE84]
367	Threshold current high	$I_{thh}$	ampere	Real	[IEEE84]
368	Threshold current low	$I_{thl}$	ampere	Real	[IEEE84]
369	Threshold voltage	$V_{th}$	volt	Real	[IEEE84]
370	Threshold voltage high	$V_{thh}$	volt	Real	[IEEE84]
371	Threshold voltage low	$V_{thl}$	volt	Real	[IEEE84]
372	Tolerance			Real	[IEEE84]
373	Tolerance at d deg C (percent)			Real	[IEEE84]
374	Total duration	$t_{td}$	second	Real	[IEEE84]
375	Total harmonic distortion (percent)	$THD$		Real	[MIL133]
376	Total power dissipation, all terminals	$P_T$	watt	Real	[MIL195]
377	Transducer power gain	$G_T$	dB	Real	[MIL133]
378	Transient response	$T_R$	second	Real	[MIL133]
379	Transition duration		second	Real	[IEEE84]
380	Transition time, high to low level	$t_{THL}$	second	Real	[MIL133]
381	Transition time, low to high level	$t_{TLH}$	second	Real	[MIL133]
382	Turn off time	$t_{off}$	second	Real	[MIL195]
383	Turn on time	$t_{on}$	second	Real	[MIL195]
384	Turn-off delay time	$t_{d(off)}$	second	Real	[MIL195]
385	Turn-on delay time	$t_{d(on)}$	second	Real	[MIL195]
386	Unbalance voltage	$V_{OU}$	volt	Real	[MIL133]
387	Under-voltage protection			String	[IEEE84]
388	Under-voltage release			String	[IEEE84]
389	Units			String	[IEEE84]
390	Unity gain bandwidth	$G_b$	hertz	Real	[IEEE84]
391	Unity gain bandwidth, maximum	$G_{bmax}$	hertz	Real	[IEEE84]
392	Unity gain bandwidth, minimum	$G_{bmin}$	hertz	Real	[IEEE84]
393	Update time			String	[IEEE84]
394	Utilization factor (maximum demand/rated capacity)			Real	[IEEE84]
395	Value			String	[IEEE84]
396	Voltage	$V$	volt	Real	[IEEE84]
397	Voltage control	$V_c$	volt	Real	[IEEE84]
398	Voltage control oscillator frequency	$F_{vco}$	hertz	Real	[IEEE84]
399	Voltage control oscillator frequency, high	$F_{vcoh}$	hertz	Real	[IEEE84]
400	Voltage control oscillator frequency, low	$F_{vclo}$	hertz	Real	[IEEE84]

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
401	Voltage control, maximum	$V_{cmax}$	volt	Real	[IEEE84]
402	Voltage control, minimum	$V_{cmin}$	volt	Real	[IEEE84]
403	Voltage rating		volt	Real	[IEEE84]
404	Voltage reference	$V_{ref}$	volt	Real	[IEEE84]
405	Voltage-temperature coefficient		volt/deg C	Real	[MIL195]
406	Wavelength		hertz	Real	[IEEE84]
407	Width		inch	Real	[IEEE84]
408	Wire diameter		inch	Real	[IEEE84]
409	Working peak off-state voltage	$V_{wp(off)}$	volt	Real	[MIL195]
410	Working peak reverse voltage	$V_{RWM}$	volt	Real	[MIL195]
411	Working peak off-state voltage	$V_{DWM}$	volt	Real	[MIL195]
412	Working peak-reverse voltage rating		volt	Real	[MIL195]
413	Write pulse width	$T_w$	second	Real	[IEEE84]
414	Write pulse width, maximum	$T_{wmax}$	ampere	Real	[IEEE84]
415	Write pulse width, minimum	$T_{wmin}$	ampere	Real	[IEEE84]
416	Zero gate voltage drain current	$I_{DDS}$	ampere	Real	[MIL195]
417	Zero gate voltage source current	$I_{SDS}$	ampere	Real	[MIL195]
418	Zero power resistance at $T$ deg C	$r_T$	ohm	Real	[IEEE84]
Circuit Simulation Parameters for Junction Diodes					
419	Saturation Current	IS	A	Real	[SPICE]
420	Ohmic Resistance	RS	ohm	Real	[SPICE]
421	Emission Coefficient	N		Real	[SPICE]
422	Transit Time	TT	s	Real	[SPICE]
423	Zero-bias P-N Capacitance	CJO	F	Real	[SPICE]
424	Junction Potential	VJ	V	Real	[SPICE]
425	Junction Grading Coefficient	M		Real	[SPICE]
426	Activation Energy	EG	eV	Real	[SPICE]
427	Saturation Current Temperature Coeff.	XTI		Real	[SPICE]
428	Reverse Breakdown Voltage	BV	V	Real	[SPICE]
429	Current at Breakdown Voltage	IBV	A	Real	[SPICE]
430	Forward-bias Depletion Capacitance Coeff.	FC		Real	[SPICE]
431	Flicker Noise Coefficient	KF		Real	[SPICE]
432	Flicker Noise Exponent	AF		Real	[SPICE]

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
Circuit Simulation Parameters for JFET Transistors					
433	Threshold Voltage	VTO	V	Real	[SPICE]
434	Transconductance	BETA	A/V <sup>2</sup>	Real	[SPICE]
435	Channel Length Modulation	LAMBDA	1/V	Real	[SPICE]
436	Drain Ohmic Resistance	RD	ohm	Real	[SPICE]
437	Source Ohmic Resistance	RS	ohm	Real	[SPICE]
438	Zero-bias Gate-Source Junction Cap.	CGS	F	Real	[SPICE]
439	Zero-bias Gate-Drain Junction Cap.	CGD	F	Real	[SPICE]
440	Gate Junction Potential	PB	V	Real	[SPICE]
441	Gate Junction Saturation Current	IS	A	Real	[SPICE]
442	Forward-bias Depletion Capacitance Coeff.	FC		Real	[SPICE]
443	Threshold Voltage Temperature Coeff.	VTOTC	V/degC	Real	[SPICE]
444	BETA Exponential Temperature Coeff.	BETATCE	%/degC	Real	[SPICE]
445	Flicker Noise Coefficient	KF		Real	[SPICE]
446	Flicker Noise Exponent	AF		Real	[SPICE]
Circuit Simulation Parameters for Bipolar Transistors					
447	Transport Saturation Current	IS	A	Real	[SPICE]
448	Ideal Maximum Forward Beta	BF		Real	[SPICE]
449	Forward Current Emission Coefficient	NF		Real	[SPICE]
450	Forward Early Voltage	VAF	V	Real	[SPICE]
451	High Current Beta Rolloff	IKF	A	Real	[SPICE]
452	B-E Leakage Saturation Current	ISE	A	Real	[SPICE]
453	B-E Leakage Emission Coefficient	NE		Real	[SPICE]
454	Ideal Maximum Reverse Beta	BR		Real	[SPICE]
455	Reverse Current Emission Coefficient	NR		Real	[SPICE]
456	Reverse Early Voltage	VAR	V	Real	[SPICE]
457	High Reverse Current (Irev) Beta Rolloff	IKR	A	Real	[SPICE]
458	B-C Leakage Saturation Current	ISC	A	Real	[SPICE]
459	B-C Leak Emission Coefficient	NC		Real	[SPICE]
460	Zero Bias Base Resistance (Rbase)	RB	ohm	Real	[SPICE]
461	Rbase Cutoff Current	IRB	A	Real	[SPICE]
462	Minimum Base Resistance	RBM	ohm	Real	[SPICE]
463	Emitter Resistance	RE	ohm	Real	[SPICE]
464	Collector Resistance	RC	ohm	Real	[SPICE]
465	BE Zero Bias Depletion Capacitance	CJE	F	Real	[SPICE]
466	B-E Built-in Potential	VJE	V	Real	[SPICE]
467	B-E Junction Exponential Factor	MJE		Real	[SPICE]
468	Ideal Forward Transit Time	TF	s	Real	[SPICE]
469	TF Bias Depletion Coefficient	XTF		Real	[SPICE]
470	VBC Dependence of TF	VTF	V	Real	[SPICE]

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
471	TF High-current Parameter	ITF	A	Real	[SPICE]
472	Excess Phase	PTF	deg	Real	[SPICE]
473	B-C Zero Bias Depletion Capacitance	CJC	F	Real	[SPICE]
474	B-C Built-in Potential	VJC	V	Real	[SPICE]
475	B-C Junction Exponential Factor	MJC		Real	[SPICE]
476	B-C Capacitance to Base Connection Ratio	XCJC		Real	[SPICE]
477	Ideal Reverse Transit Time	TR	s	Real	[SPICE]
478	Zero Bias Collector-Substrate Capacitance	CJS	F	Real	[SPICE]
479	Substrate-Junction Built-in Potential	VJS	V	Real	[SPICE]
480	Substrate-Junction Exponential Factor	MJS		Real	[SPICE]
481	Beta-Temperature Exponent	XTB	eV	Real	[SPICE]
482	IS-Temperature Energy Gap	EG		Real	[SPICE]
483	IS-Temperature Exponent	XT		Real	[SPICE]
484	Flicker Noise Coefficient	KF		Real	[SPICE]
485	Flicker Noise Exponent	AF		Real	[SPICE]
486	Forward-Bias Capacitance Coefficient	FC		Real	[SPICE]
487	Area Scale-Factor	AREA		Real	[SPICE]
Circuit Simulation Parameters for MOS Transistors					
488	Model Complexity identifier	LEVEL		Integer	[SPICE]
489	Surface Potential	PHI	V	Real	[SPICE]
490	Drain Ohmic Resistance	RD	ohm	Real	[SPICE]
491	Source Ohmic Resistance	RS	ohm	Real	[SPICE]
492	Zero-bias Body-Drain Capacitance	CBD	F	Real	[SPICE]
493	Zero-bias Body-Source Capacitance	CVS	F	Real	[SPICE]
494	Bulk Junction Saturation Current	IS	A	Real	[SPICE]
495	Gate-Source Overlap Cap./Channel Width	CGSO	F/m	Real	[SPICE]
496	Gate-Drain Overlap Cap./Channel Width	CGDO	F/m	Real	[SPICE]
497	Gate-Bulk Overlap Cap./Channel Width	CGBO	F/m	Real	[SPICE]
498	Drain, Source Diffusion Sheet Resistance	RSH	ohm	Real	[SPICE]
499	Zero-bias Bulk Junction Cap./Area	CJ	F/m <sup>2</sup>	Real	[SPICE]
500	Bulk Junction Bottom grading coefficient	MJ		Real	[SPICE]
501	Zero-bias Bulk Junction perimeter Cap./l	CJSW	F/m	Real	[SPICE]
502	Bulk Junction Sidewall grading coeff.	MJSW		Real	[SPICE]
503	Bulk Junction Saturation Current/Area	JS	A/m <sup>2</sup>	Real	[SPICE]
504	Oxide Thickness	TOX	m	Real	[SPICE]
505	Substrate Doping Density	NSUB	cm <sup>-3</sup>	Real	[SPICE]
506	Surface State Density	NSS	cm <sup>-2</sup>	Real	[SPICE]
507	Fast Surface State Density	NFS	cm <sup>-2</sup>	Real	[SPICE]
508	Type of Gate Matl. +1=opp. 0=Al -1=sub.	TPG		Real	[SPICE]

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	Definition	Symbol	Unit	Type	Ref.
509	Metallurgical Junction Depth	XJ	m	Real	[SPICE]
510	Lateral Diffusion Length	LD	m	Real	[SPICE]
511	Surface Mobility	UO	cm <sup>2</sup> /V*s	Real	[SPICE]
512	Maximum Drift Velocity	VMAX	m/s	Real	[SPICE]
513	Width Effect on Threshold Voltage	DELTA		Real	[SPICE]
514	Mobility Modulation term	THETA	1/V	Real	[SPICE]
515	Static Feedback term	ETA		Real	[SPICE]
516	Saturation Field factor	KAPPA		Real	[SPICE]
517	Channel Length	L	m	Real	[SPICE]
518	Channel Width	W	m	Real	[SPICE]
519	Lateral Diffusion Width	WD	m	Real	[SPICE]
520	Zero-bias Threshold Voltage	VTO	V	Real	[SPICE]
521	Transconductance	KP	A/V <sup>2</sup>	Real	[SPICE]
522	Bulk Threshold parameter	GAMMA	V <sup>0.5</sup>	Real	[SPICE]
523	Channel-length Modulation	LAMBDA	1/V	Real	[SPICE]
524	Gate Ohmic Resistance	RG	ohm	Real	[SPICE]
525	Bulk Ohmic Resistance	RB	ohm	Real	[SPICE]
526	Drain-Source Shunt Resistance	RDS	ohm	Real	[SPICE]
527	Bulk Junction Potential	PB	V	Real	[SPICE]
528	Bulk Junction Forward-bias Cap. Coeff.	FC		Real	[SPICE]
529	Mobility Degradation critical field	UCRIT	V/cm	Real	[SPICE]
530	Mobility Degradation exponent	UEXP		Real	[SPICE]
531	Mobility Degradation Transverse field k	UTRA		Real	[SPICE]
532	Channel Charge coefficient	NEFF		Real	[SPICE]
533	Fraction of Channel Charge due to Drain	XQC		Real	[SPICE]
534	Flicker Noise coefficient	KF		Real	[SPICE]
535	Flicker Noise exponent	AF		Real	[SPICE]
Hybrid MicroCircuit Resistor Attributes†					
536	Hybrid Resistor Laser Trim Value	N/A	N/A	Real	LHR1
537	Hybrid Resistor Laser Trim Process	N/A	N/A	String	LHR2
538	Hybrid Resistor Ink ID	N/A	N/A	String	LHR3
539	Hybrid Resistor Shape	N/A	N/A	String	LHR4
540	Hybrid Resistor Length	N/A	square	Real	LHR5
541	Hybrid Resistor Width	N/A	Model	Real	LHR6
542	Hybrid Resistor Hat Length	N/A	Model	Real	LHR7
543	Hybrid Resistor Hat Width	N/A	Model	Real	LHR8
544	Hybrid Resistor Terminal Length Extension	N/A	Model	Real	LHR9
545	Hybrid Resistor Terminal Width Extension	N/A	Model	Real	LHR10
546	Hybrid Resistor Terminal Overlap	N/A	Model	Real	LHR11

ECO651

†Definitions for hybrid microcircuit resistor attributes are given in the explanatory notes following this table.

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

EC0651

Table 11. Electrical Attribute List (ALT=2) (continued)

Layered Electrical Product Design Rule Attributes						
No.	AVC	Definition	Symbol	Unit	Type	Ref.
547	2	Component Placement Grid	N/A	Model	Real	LDR1
548	2	Pin Placement Grid	N/A	Model	Real	LDR2
549	2	Via Placement Grid	N/A	Model	Real	LDR3
550	2	Conductor Path and Area Grid	N/A	Model	Real	LDR4
551	1	Default Padstack Size	N/A	Model	Real	LDR5
552	1	Component to Component Placement Clearance	N/A	Model	Real	LDR6
553	1	Padstack to Padstack Placement Clearance	N/A	Model	Real	LDR7
554	1	Conductor to Conductor Placement Clearance	N/A	Model	Real	LDR8
555	1	Conductor to Padstack Placement Clearance	N/A	Model	Real	LDR9
556	1	Dielectric to Padstack Clearance	N/A	Model	Real	LDR10
557	1	Dielectric to Deposition Component Clearance	N/A	Model	Real	LDR11
558	1	Dielectric to Conductor Overlap	N/A	Model	Real	LDR12
559	2	Component Placement Orientation Rule:	N/A	N/A		LDR13
		• Layer			String	
		• Orientation				
		· Orthogonal_Only			String	
		· Diagonal_Allowed			String	
		· All_Angle			String	
560	3	AutoRouting Orientation Rule	N/A	N/A		LDR14
		• Layer			String	
		• Direction				
		· Vertical			String	
		· Horizontal			String	
		· Orthogonal			String	
		· Diagonal_Allowed			String	
		· All_Angle			String	
		• TJunctions				
		· TJunctions_Allowed			String	
		· No_TJunctions			String	
561	1	Padstack No Connect Rule	N/A	N/A	String	LDR15
562	1	Via Under Padstack Rule	N/A	N/A	Pointer	LDR16
563	2	Net Length Rule	N/A	Model	Real	LDR17

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

ECO651

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	AVC	Definition	Symbol	Unit	Type	Ref.
564	6	ECL Net Length Rule:	N/A	Model		LDR18
		• TN_Max			Real	
		• TN_Min			Real	
		• RT_Max			Real	
		• RT_Min			Real	
		• ST_Max			Real	
		• ST_Min			Real	
565	1	Net Configuration Rule:	< n.a. >	N/A		LDR19
		• Starburst			String	
		• Daisy_Chain			String	
		• ECL_Daisy_Chain			String	
		• User			String	
		• ECL_User			String	
566	1	Net Priority	N/A	N/A	Integer	LDR20
567	1	Net Source Pin Rule	N/A	N/A	Pointer	LDR21
568	1	Net Restrictions Rule	N/A	N/A		LDR22
		• TJunctions			String	
569	1	Net Terminating Resistor Rule	N/A	N/A	Pointer	LDR23
570	2	Generic Design Rule:	N/A	N/A		LDR24
		• Name_Of.Rule			String	
		• Rule_Value			String	
571	1	Component Pins Modified Rule:	N/A	N/A		LDR25
		• Move_Pin			String	
		• New_Padstack			String	
572	1	Hybrid Resistor Length Step Size	N/A	Model	Real	LDR26
573	1	Hybrid Resistor Minimum Dimension	N/A	Model	Real	LDR27
574	1	Hybrid Resistor Maximum Dimension	N/A	Model	Real	LDR28
575	1	Hybrid Resistor Minimum Allowed Area	N/A	(Model) <sup>2</sup>	Real	LDR29
576	1	Hybrid Resistor Maximum Allowed Area	N/A	(Model) <sup>2</sup>	Real	LDR30
577	1	Hybrid Resistor Minimum Aspect Ratio	N/A	N/A	Real	LDR31
578	1	Hybrid Resistor Maximum Aspect Ratio	N/A	N/A	Real	LDR32
579	1	Hybrid Resistor Trim to Conductor Clearance	N/A	Model	Real	LDR33

Table 11. Electrical Attribute List (ALT=2) (continued)

No.	AVC	Definition	Symbol	Unit	Type	Ref.
580	1	Hybrid Resistor Default Direction	N/A	N/A		LDR34
		•Horizontal			String	
		•Vertical			String	
581	1	Hybrid Resistor Required Direction	N/A	N/A		LDR35
		•Horizontal			String	
		•Vertical			String	
582	1	Hybrid Resistor Required Layer	N/A	N/A	String	LDR36
583	1	Hybrid Resistor Ink Deviation (Length)	N/A	Model	Real	LDR37
584	1	Hybrid Resistor Ink Deviation (Percent of Value)	N/A	N/A	Real	LDR38

### Hybrid Microcircuit Resistor Attributes

Explanatory notes for Hybrid Microcircuit Resistor Attributes (Numbers 536 through 546, ALT=2) in [Table 11](#) (note: details of the application of these attributes may be found in [ISHM82]).

**LHR1 Hybrid Resistor Laser Trim Value** Shall state the laser trim factor of a resistor. The resistor's trim factor is stated as a percent of the nominal value of the resistor. A trim factor of 100.0 indicates that the resistor is not trimmed.

**LHR2 Hybrid Resistor Laser Trim Process** Shall state the process which shall be one of the following (case insensitive);

- N** No trimming.
- S** Static (specific amount).
- D** Dynamic (trim to within functioning circuit).
- SD** Static or dynamic.

**LHR3 Hybrid Resistor Ink ID** Shall state the identification of the resistor ink.

**LHR4 Hybrid Resistor Shape** Shall state the type of hybrid resistor shape which shall be one of the following (case insensitive);

- Top\_Hat**
- Rectangular**
- Serpentine**

**LHR5 Hybrid Resistor Length** Shall state the length of the resistor in squares.

**LHR6 Hybrid Resistor Width** Shall state the width of the resistor in model space units.

**LHR7 Hybrid Resistor Hat Length** Shall state the hat length for a top-hat resistor in model space units.

**LHR8 Hybrid Resistor Hat Width** Shall state the hat width for a top-hat resistor in model space units.

**LHR9 Hybrid Resistor Terminal Length Extension** Shall state the distance the conductor terminal extends beyond the length of the resistor in model space units.

**LHR10 Hybrid Resistor Terminal Width Extension** Shall state the distance the conductor terminal extends beyond the width of the resistor in model space units.

**LHR11 Hybrid Resistor Terminal Overlap** Shall state the distance the resistor overlaps the conductor terminal in model space units.

### Design Rule Attributes

Explanatory notes for Design Rules (Numbers 547 through 584, ALT-2) in [Table 11](#) (note: details of the application of these attributes may be found in [CH84] and [HON80] ).

**LDR1 Component Placement Grid** Specifies the X & Y distance between component placement grid points. The model space origin shall coincide with one of the grid points.

**LDR2 Pin Placement Grid** Specifies the X & Y distance between component pin placement grid points. The model space origin shall coincide with one of the grid points.

**LDR3 Via Placement Grid** Specifies the X & Y distance between via placement grid points. The model space origin shall coincide with one of the grid points.

**LDR4 Conductor Path and Area Grid** Specifies the X & Y distance between conductor path and area placement grid points. The model space origin shall coincide with one of the grid points.

**LDR5 Default Padstack Size** Shall state the default size (diameter) of padstacks (pin, via or hole).

**LDR6 Component to Component Placement Clearance** Shall state the minimum separation between any pair of component placement boundaries after placement.

**LDR7 Padstack to Padstack Placement Clearance** Shall state the minimum separation between any pair of padstack (pin, via or hole) boundaries after placement.

**LDR8 Conductor to Conductor Placement Clearance** Shall state the minimum separation between any pair of conductors (path or area) after placement.

**LDR9 Conductor to Padstack Placement Clearance** Shall state the minimum separation between any conductor (path or area) and any padstack (pin, via or hole), after placement.

**LDR10 Dielectric to Padstack Clearance** Shall state the minimum separation between any dielectric crossover and any padstack (pin, via or hole), after placement.

**LDR11 Dielectric to Deposition Component Clearance** Shall state the minimum separation between any dielectric crossover and any deposition component (hybrid screened resistor), after placement.

**LDR12 Dielectric to Conductor Overlap** Shall state the minimum distance that a dielectric crossover extends (overlaps) beyond the edge of a conductor.

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

**LDR13 Component Placement Orientation Rule** There are two fields defined as follows:

**Layer** The layer to which the rule applies shall be one of the predefine functional layers from the Level to LEP Layer Map property.

**Orientation** The type of path orientation allowed for the specific layer (case insensitive) shall be one of the following three:

1. Orthogonal\_Only
2. Diagonal\_Allowed
3. All\_Angle

**LDR14 Auto Routing Orientation Rule** There are three fields defined as follows:

**Layer** The layer to which the rule applies shall be one of the predefine functional layers from the Level to LEP Layer Map property.

**Direction** The dominant direction for autorouting paths on the specified layer (case insensitive) shall be one of the following five:

1. Vertical
2. Horizontal
3. Orthogonal
4. Diagonal\_Allowed
5. All\_Angle

**TJunctions** T-junctions maybe allowed or disallowed by specifying one of the following (case insensitive) strings.

- TJunctions\_Allowed
- No\_TJunctions

**LDR15 Padstack No Connect Rule** The indicated layer shall not be used to connect to a padstack (pin or via). This layer shall be one of the predefine functional layers from the Level to LEP Layer Map property.

**LDR16 Via Under Padstack Rule** The via padstack subfigure definition identified may be placed directly under a surface mount padstack and shall connect to it.

**LDR17 Net Length Rule** The minimum and the maximum length for a net shall both be specified. The default minimum shall be 0.0.

**LDR18 ECL Net Length Rule** There are six fields defined as follows:

**TN\_Max** Maximum length of the total net.

**TN\_Min** Minimum length of the total net.

**RT\_Max** Maximum net length between the terminating resistor and the next to last pin.

**RT\_Min** Minimum net length between the terminating resistor and the next to last pin.

**ST\_Max** Maximum net length between the source pin and the terminating resistor.

**ST\_Min** Minimum net length between the source pin and the terminating resistor.

**LDR19 Net Configuration Rule** The configuration of the net shall be one of the following five (case insensitive):

**Starburst** (Default configuration) The router is free to connect each net as efficiently as it can.

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

**Daisy\_Chain** Configure the net in a system generated daisy chain. (No physical connect point may have more than two joins.)

**ECL\_Daisy\_Chain** Configure as a system generated daisy chain and check as an ECL net.

User Configure in a user specified daisy chain according to the pin order in the Network Flow Associativity.

**ECL\_User** Configure as a user defined daisy chain but check as an ECL net.

**LDR20 Net Priority** Nets with high priority shall be routed even if more nets of lower priority must then remain unrouted. The routing priority is expressed as an integer from 0 to 100, where 100 is the highest priority.

**LDR21 Net Source Pin Rule** Shall state the source pin of a potential ECL net. The value shall be the DE pointer to the Connect Point entity of the potential source pin.

**LDR22 Net Restrictions Rule** Any special restriction that pertains to the net shall be indicated by the following (case insensitive) restriction:

**TJunctions** TJunctions shall not be part of the net

**LDR23 Net Terminating Resistor Rule** The terminating pin for an ECL net shall be identified by the DE pointer to its Connect Point entity.

**LDR24 Generic Design Rule** A generic design rule (one that has not yet been incorporated into this specification) shall be represented by two fields.

**Name\_Of\_Rule** The name of the rule in the native system.

**Rule\_Value** The value associated with the rule in the native system. If multiple fields are part of the value, they shall be delimited with parameter delimiter characters.

**LDR25 Component Pins Modified Rule** There are two fields that specify that the component pins may be moved, or a separate padstack may be assigned upon instantiation (case insensitive).

**Move\_Pin** Shall state that the pins may be moved (null value specifies that the pins shall not be moved).

**New\_Padstack** Shall state that a new padstack may be assigned (null value specifies that a new padstack shall not be assigned).

**LDR26 Hybrid Resistor Length Step Size** Step size (in model space units) by which the resistor length may be increased to meet specific power dissipation requirements.

**LDR27 Hybrid Resistor Minimum Dimension** The minimum dimension (length or width) in model space units for a resistor.

**LDR28 Hybrid Resistor Maximum Dimension** The maximum dimension (length or width) in model space units for a resistor.

**LDR29 Hybrid Resistor Minimum Allowed Area** The minimum area for a resistor.

**LDR30 Hybrid Resistor Maximum Allowed Area** The maximum area for a resistor.

**LDR31 Hybrid Resistor Minimum Aspect Ratio** Minimum aspect ratio of length to width for a resistor.

**LDR32 Hybrid Resistor Maximum Aspect Ratio** Maximum aspect ratio of length to width for a resistor.

#### **4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)**

**LDR33 Hybrid Resistor Trim to Conductor Clearance** The minimum placement distance from a resistor to a conductor path or area to allow for laser trimming.

**LDR34 Hybrid Resistor Default Direction** The default direction that a resistor should be screened (Horizontal or Vertical).

**LDR35 Hybrid Resistor Required Direction** The required direction that a resistor shall be screened (Horizontal or Vertical).

**LDR36 Hybrid Resistor Required Layer** The required layer on which a resistor shall be placed. This layer shall be one of the predefine functional layers from the Level to LEP Layer Map property.

**LDR37 Hybrid Resistor Ink Deviation (Length)**

**LDR38 Hybrid Resistor Ink Deviation (Percent of Value)**

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 12. AEC Attribute List (ALT=3)

No.	Definition	Unit†	Data Type
1	Air changes per hour	hour <sup>-1</sup>	Real
2	Area of glazing	foot <sup>2</sup>	Real
3	Bearing wall		Logical
4	Bearing wall capacity	pound/foot	Real
5	Building name		String
6	Building occupancy type		String
7	Capacity per unit of egress width		Integer
8	Ceiling cavity depth	inch	Real
9	Ceiling type		String
10	Combustible		Logical
11	Concentrated dead load	pound	Real
12	Concentrated live load	pound	Real
13	Cooled		Logical
14	Cost per square foot	dollar(US)	Real
15	Finish color		String
16	Finish type		String
17	Finished floor elevation	foot	Real
18	Finished opening height	inch	Real
19	Finished opening width	inch	Real
20	Fire door		Logical
21	Fire protection		Logical
22	Fire rating	hour	Real
23	Fire suppression system		Logical
24	Fire wall		Logical
25	Floor name		String
26	Floor to ceiling height	foot	Real
27	Floor to floor height	foot	Real
28	Floor type		String
29	Frame type		String
30	Gross area	foot <sup>2</sup>	Real
31	Gross floor area per occupant	foot <sup>2</sup>	Real
32	Hardware type		String
33	Heated		Logical
34	Hydrostatic pressure	pound/foot <sup>2</sup>	Real
35	Illumination level	foot-candle	Real
36	Infiltration	foot <sup>3</sup> /minute	Real
37	Latent heat gain	BTU/hour	Real
38	Light reflectance (percent)		Real
39	Lintel height	inch	Real
40	Live load reduction (percent)		Real
41	Means of egress		Logical

†The use of English rather than SI units follows the current practice of the AEC industry.

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 12. AEC Attribute List (ALT=3) (continued)

No.	Definition	Unit†	Data Type
42	Net area	foot <sup>2</sup>	Real
43	Net floor area per occupant	foot <sup>2</sup>	Real
44	Number of stories		Integer
45	Occupant load		Integer
46	Opening type		String
47	Operable		Logical
48	Relative humidity (percent)		Real
49	Riser height	inch	Real
50	Riser height, maximum	inch	Real
51	Riser height, minimum	inch	Real
52	Room activity		String
53	Room name		String
54	Rough opening height	inch	Real
55	Rough opening width	inch	Real
56	Sensible heat gain	BTU/hour	Real
57	Shading coefficient (percent)		Real
58	Shear wall		Logical
59	Shear wall capacity	pound/foot	Real
60	Sill height	inch	Real
61	Slope (percent)		Real
62	Smoke door		Logical
63	Smoke rating	hour	Real
64	Smoke wall		Logical
65	Snow load	pound/foot <sup>2</sup>	Real
66	Soil bearing capacity	pound/foot <sup>2</sup>	Real
67	Soil density	pound/foot <sup>3</sup>	Real
68	Soil type		String
69	Sound level	dB	Real
70	Sound reflectance (percent)		Real
71	Sound transmission class		Integer
72	Temperature	deg F	Real
73	Thermal transmittance	BTU/hour/foot	Real
74	Tread width	inch	Real
75	Tread width, maximum	inch	Real
76	Tread width, minimum	inch	Real
77	Uniform dead load	pound/foot <sup>2</sup>	Real
78	Uniform live load	pound/foot <sup>2</sup>	Real
79	Unit of egress width	inch	Real
80	Ventilation	foot <sup>3</sup> /minute	Real
81	Wall type		String
82	Wind pressure	pound/foot <sup>2</sup>	Real
83	Work plane height	inch	Real

†The use of English rather than S1 units follows the current practice of the AEC industry.

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 13. Process Plant Attribute List (ALT=4)

No.	Definition	Data Type	Default Units †
1	Nominal pipe size	Real	model units
2	Material name	String	
3	End preparation	String	
4	Wall thickness	Real	model units
5	User part number	String	
6	Joint identification number	String	
7	Configuration non-deviation code	Integer	
8	Material non-deviation code	Integer	
9	Specification body	String	
10	Material control level	String	
11	Criticality class	String	
12	Joint type	String	
13	Dry weight/mass value	Real	lbs (pounds)
14	Dry weight/mass units	String	
15	Pipe spool/detail name	String	
16	Functional group code	String	
17	Part type	String	
18	Nominal pipe size type	String	
19	Object identifier	String	
20	Object revision	String	
21	Drawing and item concatenation (find number)	String	
22	Applicability	String	
23	Schedule	String	
24	Service	String	
25	Shock rating	String	
26	Sub-safe	String	
27	Noise critical	String	
28	Wall thickness units	String	
29	Equipment name	String	
30	Equipment function	String	
31	Dry center of gravity (x)	Real	model units
32	Dry center of gravity (y)	Real	model units
33	Dry center of gravity (z)	Real	model units
34	Dry center of gravity units	String	
35	Function	String	
36	Class	String	
37	Pipe specification	String	
38	Type	String	
39	Insulation specification	String	
40	Paint specification	String	

†Model units refer to the units specified in Global Parameters 14 and 15

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 13. Process Plant Attribute List (ALT=4) (continued)

No.	Definition	Data Type	Default Units †
41	Tracing specification	String	
42	Status (detailed, stressed, ...)	String	
43	Tag number	String	
44	Maximum design pressure	Real	psia
45	Maximum design pressure units	String	
46	Maximum design temperature	Real	degree, F
47	Maximum design temperature units	String	
48	Vapor pressure	Real	psia
49	Vapor pressure units	String	
50	Material description	String	
51	Gasket thickness	Real	model units
52	Gasket thickness units	String	
53	Integral gasket code	String	
54	Isometric split point indicator	String	
55	Line/sequence number	String	
56	Instrument loop number	String	
57	Nominal pipe size units	String	
58	Design area	String	
59	Project	String	
60	Valve type	String	
61	Valve size	Real	model units
62	Valve size units	String	
63	Valve code	String	
64	Molecular weight	Real	lb/lb mole
65	Molecular weight units	String	
66	Flow rate	Real	lb/hr
67	Flow rate units	String	
68	Flow material	String	
69	Type instrument	String	
70	Tube specification	String	
71	Density	Real	lb/ft <sup>3</sup>
72	Density units	String	
73	Viscosity	Real	centipoise
74	Viscosity units	String	
75	Heat capacity	Real	BTU/lb F
76	Heat capacity units	String	
77	Operating temperature	Real	degree, F
78	Operating temperature units	String	
79	Operating pressure	Real	psia
80	Operating pressure units	String	

†Model units refer to the units specified in Global Parameters 14 and 15

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 13. Process Plant Attribute List (ALT=4) (continued)

No.	Definition	Data Type	Default Units †
81	Composition	String	
82	Bldg/bent	String	
83	Elevation/floor	Real	model units
84	Elevation/floor units	String	
85	Pressure rating	Real	psi
86	Pressure rating units	String	
87	Flange finish	String	
88	Flow direction	String	
89	Project area	String	
90	Note	String	
91	Note identifier	String	
92	Wet weight/mass value	Real	lbs (pounds)
93	Wet weight/mass units	String	
94	Wet center of gravity (x)	Real	model units
95	Wet center of gravity (y)	Real	model units
96	Wet center of gravity (z)	Real	model units
97	Wet center of gravity units	String	
98	Outside diameter	Real	model units
99	Outside diameter units	String	
100	Dimensional standard	String	
101	Weld i.d.	String	
102	Component name	String	
103	Spec option code	String	
104	Fabrication category	String	
105	Material takeoff indicator	String	
106	Through-bolted indicator	Logical	
107	Valve operator type	String	
108	Reinforcing pad thickness	Real	model units
109	Reinforcing pad thickness units	String	
110	Reinforcing pad width	Real	model units
111	Reinforcing pad width units	String	
112	Chain length	Real	model units
113	Chain length units	String	
114	Pipe support type	String	
115	Support detail reference	String	
116	Bolt type	String	
117	Bolt length	Real	model units
118	Bolt length units	String	
119	Bolt diameter	Real	model units
120	Bolt diameter units	String	

†Model units refer to the units specified in Global Parameters 14 and 15

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 13. Process Plant Attribute List (ALT=4) (continued)

No.	Definition	Data Type	Default Units †	
121	Insulation thickness	Real	model units	
122	Insulation thickness units	String		
123	Insulation density	Real	lbs/ft <sup>3</sup>	
124	Insulation density units	String		
125	Fluid code	String		
126	Unit number	String		
127	Item number	String		
128	Part number	String		
129	Safety zone	String		
130	Design code	String		
131	Nozzle length #1	Real	model units	
132	Nozzle length #2	Real	model units	
133	Nozzle length units	String		
134	Nozzle bend radius	Real	model units	
135	Nozzle bend radius units	String		
136	Nozzle type	String		
137	Quantity	Real	piece	
138	Quantity units	String		
139	Pipe fit-up	Real	model units	
140	Pipe fit-up units	String		
141	Minimum design wall thickness	Real	model units	
142	Minimum design wall thickness units	String		
143	Commodity component class	Integer		
144	Commodity component subclass	Integer		
145	Flag for flanged component	Integer		
146	Flag for component attachment	Integer		
147	Component angle	Real	degrees	
148	Component angle units	String		
149	Component length from origin to bottom of socket at A end	Real	model units	
150	Component A length units	String		
151	Component length from origin to bottom of socket at B end	Real	model units	
152	Component B length units	String		
153	Component length from origin to bottom of socket at C end	Real	model units	
154	Component C length units	String		
155	Component length from origin to bottom of socket at D end	Real	model units	
156	Component D length units	String		
157	Component socket depth at A end	Real	model units	
158	Component socket depth A units	String		
159	Component socket depth at B end	Real	model units	
160	Component socket depth B units	String		

ECO645

†Model units refer to the units specified in Global Parameters 14 and 15

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 13. Process Plant Attribute List (ALT=4) (continued)

No.	Definition	Data Type	Default Units †	ECO645
161	Component socket depth at C end	Real	model units	
162	Component socket depth C units	String		
163	Component socket depth at D end	Real	model units	
164	Component socket depth D units	String		
165	Component inner width at A end	Real	model units	
166	Component inner width A units	String		
167	Component inner width at B end	Real	model units	
168	Component inner width B units	String		
169	Component inner width at C end	Real	model units	
170	Component inner width C units	String		
171	Component inner width at D end	Real	model units	
172	Component inner width D units	String		
173	Component wall thickness at A end	Real	model units	
174	Component wall thickness A units	String		
175	Component wall thickness at B end	Real	model units	
176	Component wall thickness B units	String		
177	Component wall thickness at C end	Real	model units	
178	Component wall thickness C units	String		
179	Component wall thickness at D end	Real	model units	
180	Component wall thickness D units	String		
181	Component radius of curvature	Real	model units	
182	Component radius of curvature units	String		
183	Offset of component centerlines	Real	model units	
184	Units for offset of component centerlines	String		
185	Outer diameter of flange	Real	model units	
186	Units for outer diameter of flange	String		
187	Thickness of flange	Real	model units	
188	Flange thickness units	String		
189	Diameter of raised face of flange	Real	model units	
190	Units for diameter of raised face of flange	String		
191	Thickness of raised face of flange	Real	model units	
192	Units for thickness of raised face of flange	String		
193	Length from bottom of socket to flange face	Real	model units	
194	Units for length from bottom of socket to flange face	String		
195	Diameter of bore	Real	model units	
196	Bore diameter units	String		
197	Friction factor	Real		
198	Friction factor descriptor	String		

†Model units refer to the units specified in Global Parameters 14 and 15

#### 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

Table 14. Electrical and LEP Manufacturing Attribute List (ALT=5)

ECO649

No.	AVC	Definition	Symbol	Unit	Type	†Ref.
1	9	Component Physical Orientation X Rotation, Minimum Y Rotation, Minimum Z Rotation, Minimum X Rotation, Nominal Y Rotation, Nominal Z Rotation, Nominal X Rotation, Maximum Y Rotation, Maximum Z Rotation, Maximum		degrees degrees degrees degrees degrees degrees degrees degrees degrees	Real Real Real Real Real Real Real Real Real	LMA1
2	9	LEP Physical Orientation X Rotation, Minimum Y Rotation, Minimum Z Rotation, Minimum X Rotation, Nominal Y Rotation, Nominal Z Rotation, Nominal X Rotation, Maximum Y Rotation, Maximum Z Rotation, Maximum		degrees degrees degrees degrees degrees degrees degrees degrees degrees	Real Real Real Real Real Real Real Real Real	LMA2
3	3	Component Physical Thickness Minimum Nominal Maximum		inch inch inch	Real Real Real	LMA3
4	2	Component Placement Form Code Form Code Description			String String	LMA4
5	1	Component Placement Depth Stop			String	LMA5
6	3	Component Placement Force Minimum Nominal Maximum		ounces ounces ounces	Real Real Real	LMA6
7	1	Component Placement Machine			String	LMA7
8	1	Component Placement Tool			String	LMA8
9	2	Component Placement Feeder ID Description			String String	LMA9
10	3	Component Placement Feeder X location Y Location Z location		inch inch inch	Real Real Real	LMA10

†References are to the explanatory notes following this table.

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

ECO649

Table 14. Electrical and LEP Manufacturing Attribute List (ALT=5) (continued)

No.	AVC	Definition	Symbol	Unit	Type	†Ref.
11	1	Component Placement Pick Point ID			String	LMA11
12	1	Allowable Test Point ID			String	LMA12
13	1	Actual Test Point ID			String	LMA13
14	1	Physical Component Device ID			String	LMA14
15	1	Printed Wire Assembly ID			String	LMA15
16	1	LEP Assembly ID			String	LMA16
17	1	LEP Thru Via ID			String	LMA17
18	1	LEP Blind Via ID			String	LMA18
19	1	LEP Fiducial ID			String	LMA19
20	1	Electrostatic Discharge Rating			String	LMA20
21	3	Component Placement Bonding ID Material Spec. Process Spec.			String String String	LMA21
22	3	LEP Design Thickness Top Bottom Total		inch inch inch	Real Real Real	LMA22

†References are to the explanatory notes following this table.

Explanatory notes for the Electrical and LEP Manufacturing Attribute List (ALT=5) in [Table 14](#):

### **LMA1 Component Physical Orientation**

ECO649

The Component Physical Orientation Attribute specifies the location of the referencing entity within the dispensing mechanism while it is attaching the component represented to the LEP. The property values pertain to the actual physical placement orientation of the component, about the X, Y, or Z axis. The placement on the LEP may be done by hand, auto-inserter, pick-and-place machine, robot, or other assembly technique. This is not necessarily the same orientation that is applied to the Network Subfigure Instance in the exchange file. These values are with respect to the placement of the component in its dispensing mechanism (feeder, DIP tube, part carousel, waffle pack position, etc.). When the rotation is applied to the component, the resulting orientation is the actual component placement orientation on the assembled LEP.

This attribute shall be referenced by a Network Subfigure Instance which represents a physical electronic component. This attribute may also be referenced by a Subfigure Instance which represents a mechanical component or fastening device.

### **LMA2 LEP Physical Orientation**

ECO649

The LEP Physical Orientation Attribute physically specifies the location of the LEP, about the X, Y, and Z axis, in a given work cell. These values are used by the manufacturing postprocessing software to compensate for the rotation of the LEP with respect to the original CAD model orientation. When the rotation is applied to the LEP, the resulting orientation is the actual LEP orientation in the coordinate system of the work cell.

For a particular LEP substrate, there may be several different work cell environments used during

ECO630

## **4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)**

the assembly process. Therefore, the LEP Physical Orientation X, Y, and Z Rotation Attribute shall be referenced by the Group Associativity that defines the components that are associated with a particular process. If there is only one work cell environment associated with the manufacturing process, the attribute shall be referenced by the entity defining the LEP. If no such entity exists, then the attribute may have independent status.

### **LMA3 Component Physical Thickness**

ECO649

The Component Physical Thickness attribute specifies the design thickness of the component after it has been assembled on the LEP. It is defined as the distance from the surface of the LEP, where the component is attached or mounted, to the highest point (most protruding) on the component. This value may be used for component interference checking, insertion postprocessing, and robotic tool path generation.

The Component Physical Thickness Minimum value is the minimum design thickness of the component after it has been assembled on the LEP. The component Physical Thickness Nominal value is the nominal design thickness of the component after it has been assembled on the LEP. The Component Physical Thickness Maximum value is the maximum design thickness of the component after it has been assembled on the LEP.

This attribute shall be referenced by a Network Subfigure Definition/Instance which represents a physical electrical component. This attribute may also be referenced by a Subfigure Definition/Instance which represents a mechanical component or fastening device.

### **LMA4 Component Placement Form**

ECO649

The Component Placement Form Attribute specifies the way that the leads of certain components shall be shaped, whether they are formed automatically, semi-automatically, or by hand. Although there is not a universally accepted document to control form code naming, most organizations have come up with their own internal naming conventions which shall be used as the Component Placement Form Code Value. This data is typically used for transistors, op- amps, transformers, and vertically mounted components.

The Component Placement Form Code Description value is additional information about a specific form code such as a particular die or machine setting.

This attribute shall be referenced by a Network Subfigure Definition/Instance which represents a physical electronic component.

### **LMA5 Component Placement Depth Stop**

ECO649

The Component Placement Depth Stop Attribute specifies the actual placement machine depth stop or distance that the insertion machine shall use when placing the component.

Many insertion machines have a set of graduated depth stops, with fixed increments, that may regulate placement of a component. Other insertion machines read a string that controls the height of the mounted component from the surface of the LEP after assembly. If the value is an insertion machine code such as "C43", "D6", or "1a", then the string shall be placed in the attribute. The insertion machine postprocessing software uses the code to calibrate the machine. If the value is the actual linear distance that the insertion machine uses to place the component, then the value shall be placed in the attribute as a real number such as "0.125". The value shall reflect the linear distance measured in the units specified.

This attribute shall be referenced by a Network Subfigure Definition/Instance which represents a physical electronic component. This attribute may also be referenced by a Subfigure Definition/Instance which represents a mechanical component or fastening device.

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

### LMA6 Component Placement Force

ECO649

The Component Placement Force Attribute specifies the force that shall be exerted on a component during the placement process. The insertion machine uses this data and the feedback from one or more tactile sensors to determine if the component has met the design criteria for placement. This is very important for SMT applications, where the component leads are to be pushed into solder paste on the LEP, with a predetermined force to insure good bonding during the solder process. The force is variable from component to component depending on package type and pin count.

The Component Placement Force Minimum value is the minimum force that shall be exerted on a component during the placement process. The Component Placement Force Nominal value is the nominal force that shall be exerted on a component during the placement process. The Component Placement Force Maximum value is the maximum force that shall be exerted on a component during the placement process.

This attribute shall be referenced by a Network Subfigure Definition/Instance which represents a physical electronic component. This attribute may also be referenced by a Subfigure Definition/Instance which represents a mechanical component or fastening device.

### LMA7 Component Placement Machine

ECO649

The Component Placement Machine Attribute specifies the name, number, or other identifier of the machine that is used to install the component on the LEP. This data is used in CAD or CAPP systems to assign components to machines during the design phase of the LEP.

For a particular assembly there may be several different insertion sequences. Therefore, the Component Placement Machine Attribute shall be referenced by the Group Associativity that defines the components that are associated with each insertion sequence. If there is no insertion sequence specified for the assembly, the attribute shall be referenced by the entity defining the LEP. If no such entity exists, then the attribute may have independent status.

### LMA8 Component Placement Tool

ECO649

The Component Placement Tool Attribute specifies the name of the tool that is used to handle a part during the assembly process. This data element is used by manufacturing post-processing software to determine which end-effector or tooling head should be used to pick up a component.

This attribute shall be referenced by a Network Subfigure Definition/Instance which represents a physical electronic component. This attribute may also be referenced by a Subfigure Definition/Instance which represents a mechanical component or fastening device.

### LMA9 Component Placement Feeder

ECO649

The Component Placement Feeder Attribute specifies the feeder machine ID, or organization dependent code for the placement feeder machine, and its description. The ID value is typically a short string of characters that has significance to a particular machine process.

The Component Placement Feeder Description value is the name of the part feeder that is used for component placement. This could include names such as waffle.pack-2, TR09, or others that are machine specific identifiers. The name could apply to SMT, through-hole technology, and mechanical components by stating where they are physically located in the work cell or placement machine. (Note: Most components have preset locations for each placement configuration.)

This attribute shall be referenced by a Network Subfigure Definition/Instance which represents a physical electronic component. This attribute may also be referenced by a Subfigure Definition/Instance which represents a mechanical component or fastening device.

## 4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)

### LMA10 Component Placement Feeder Location

ECO649

The Component Placement Feeder Location Attribute specifies the coordinate location for the feeder input. This value is typically a machine absolute coordinate. By analyzing the Component Placement Feeder X, Y, and Z Location and the Component Placement Pick Point X, Y, and Z Location, the exact location of a component in its placement machine may be determined. Furthermore, by analyzing the component Placement Physical orientation, and LEP Physical orientation, the exact three-dimensional movement that is required to place a component may be determined.

This attribute shall be referenced by a Network Subfigure Definition/Instance which represents a physical electronic component. This attribute may also be referenced by a Subfigure Definition/Instance which represents a mechanical component or fastening device.

### LMA11 Component Placement Pick Point ID

ECO649

The Component Placement Pick Point ID Attribute specifies the location of that feature on a component which the placement head, tool, or robot end-effector shall use for attaching to the part during the placement process. This is generally the center of an SMT component, but may vary for some odd shaped parts or special assembly methods that require holding the part at an offset and/or different angle.

This attribute may be referenced by any entity that locates a feature. If the attribute is referenced by an entity, the parent entity is tagged as a component placement pick point, and the string value in the attribute further describes it.

**LMA12 Allowable Test Point ID** The Allowable Test Point Attribute specifies that a physical feature on the LEP (via, through-hole component pin, SMT land area, or dedicated test point) is available for test point access or probing. Interference from other components, tool fixtures, and/or physical parameters of an available feature (*e.g.* a SMT land pad is too fragile for a particular probing technique) could make a candidate test point unacceptable.

More than one Allowable Test Point Attribute may be associated with the same point to represent its availability for several test sequences (*e.g.*, a particular point may be used for bare-board testing, in-circuit testing, and robotic probing while another might be available only for bare-board tests). Typical values for this attribute are strings such as: "PWB" (for printed wiring bare-board test), "ICT" (for in-circuit or combinational testing), "R" (for robotic probing), or "NA" (for not available).

This attribute may be referenced by any entity that locates a conductive feature. If the attribute is referenced by an entity, the parent entity is tagged as an allowable test point, and the string value in the attribute further describes it. If there is no Allowable Test Point Attribute referenced by an entity, it is not an allowable test point.

### LMA13 Actual Test Point ID

ECO649

The Actual Test Point Attribute specifies that a physical feature on the LEP (via, through-hole component pin, SMT land area, or dedicated test point) has been assigned as a test point.

If the attribute is referenced by an entity, the parent entity is tagged as a test point, and the string value in the attribute further describes it.

More than one Actual Test Point Attribute may be associated with the same point to represent that it is used in several test schemes (*e.g.*, a particular point may be used for PWB test, in-circuit, and robotic probing while another might be available only for PWB test). Typical values for this attribute are strings that define the name/model number of the tester such as DITMCO\_9 100, HP3065, GR2750, L293, *etc.* Another use could be to organize the test points according to the class of test

## **4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)**

such as PWB, ICT, functional, *etc.* The primary purpose of this attribute is to convey the design intent of where and how the test point information is to be used.

This attribute may be referenced by any entity that locates a feature.

### **LMA14 Physical Component Device ID**

ECO649

The Physical Component Device ID Attribute specifies that an object is a physical component.

If the attribute is referenced by an entity, the parent entity is tagged as a physical component device, and the string value in the attribute further describes it. This attribute shall be referenced by a Network Subfigure Definition or Subfigure Definition which represents a physical component device.

### **LMA15 Printed Wire Assembly ID**

ECO649

The Printed Wire Assembly ID Attribute specifies that an object is a PWA.

If the attribute is referenced by an entity, the parent entity is tagged as a PWA, and the string value ECO630 in the attribute further describes it. This attribute shall be referenced by the entity defining the Printed Wire Assembly (PWA). If no such entity exists, then the attribute may have independent status.

### **LMA16 LEP Assembly ID**

ECO649

The LEP Assembly ID Attribute specifies that an object is a Layered Electrical Product.

If the attribute is referenced by an entity, the parent entity is tagged as a Layered Electrical Product, ECO630 and the string value in the attribute further describes it. This attribute shall be referenced by the entity defining the LEP. If no such entity exists, then the attribute may have independent status.

### **LMA17 LEP Through Via ID**

ECO649

The LEP Through Via ID Attribute specifies that an object is a through via, a conductive hole which penetrates all the LEP strata.

If the attribute is referenced by an entity, the parent entity is tagged as a through via, and the string value in the attribute further describes it. This attribute shall be referenced by a Network Subfigure Definition or Subfigure Definition which represents a through via.

### **LMA18 LEP Blind Via ID**

ECO649

The LEP Blind Via ID Attribute specifies that an object is a blind via, a conductive hole which penetrates only one exterior surface of the LEP.

If the attribute is referenced by an entity, the parent entity is tagged as a blind via, and the string value in the attribute further describes it. This attribute shall be referenced by a Network Subfigure Definition or Subfigure Definition which represents a blind via.

### **LMA19 LEP Fiducial ID**

ECO649

The LEP Fiducial ID Attribute specifies that an object is a fiducial, a feature useable as a designated point of reference. A fiducial is a feature (e.g., a hole) used by LEP insertion equipment to line up the LEP substrate or a component, such that the pins are properly inserted on or in the LEP substrate.

If the attribute is referenced by an entity, the parent entity is tagged as a fiducial, and the string value in the attribute further describes it. This attribute shall be referenced by a Subfigure Definition which represents a fiducial.

## **4.79 ATTRIBUTE TABLE DEFINITION ENTITY (TYPE 322)**

### **LMA20 Electrostatic Discharge Rating**

ECO649

The Electrostatic Discharge Rating Attribute specifies how sensitive a component or LEP is to electrostatic energy. The value of the attribute is a string which defines the ESD rating.

This attribute shall be referenced by a Network Subfigure Definition which represents a physical electronic component or LEP. This attribute shall be referenced by the entity defining the component or LEP. If no such entity exists, the attribute may have independent status.

### **LMA21 Component Placement Bonding**

ECO649

The Component Placement Bonding Attribute specifies that a feature represents the location for bonding material. The feature may be depicted as a single point, in which case it will most likely represent a spot of glue. The feature may also be depicted as a line or a polygon, in which case it may represent an area to be coated with solder paste. In either case, Component Placement Bonding Material Specification and Component Placement Bonding Process Specification values further state the intended meaning of the data.

The Component Placement Bonding Material Specification value specifies the type of bonding material to be used at the specified location. The value of the attribute is a string indicating the material specification.

The Component Placement Bonding Process Specification value specifies the type of bonding process for the specified location. The value of the attribute is a string indicating the process specification.

If the attribute is referenced by an entity, the parent entity is tagged as a bonding area, and the ID string value further describes it (e.g., adhesive, glue, solder, paste, etc.). This attribute may be referenced by any entity that locates a feature.

### **LMA22 LEP Design Thickness (Top, Bottom, and Total)**

ECO649

The LEP Design Thickness Attribute specifies the maximum allowable distance from the top surface of the LEP substrate to the top of the highest component, the maximum allowable distance from the bottom surface of the LEP substrate to the bottom of the lowest component, and the maximum total thickness of the LEP substrate. All three of the distances are measured from the LEP substrate after component placement.

The top and bottom surface of the LEP substrate is based on which side is represented by the COMP\_LACEMENT\_T functional level and the COMP\_PLACEMENT\_B functional level respectively, in the Level to LEP Layer Map Property Entity ([Type 406, Form 24](#)).

This attribute shall be referenced by the entity defining the LEP. If no such entity exists, the attribute [may have independent status](#).

## 4.80 ASSOCIATIVITY INSTANCE ENTITY (TYPE 402)

### 4.80 Associativity Instance Entity (Type 402)

Each time an associativity relation is needed, an Associativity Instance Entity shall be used. ECO630

The Form Number of the associativity instance identifies the meaning of the entity. If the Form ECO630 Number is between 1 and 5000, the definition is specified as described in [Section 4.80.1](#) and following sections. If the Form Number is between 5001 and 9999, an Associativity Definition Entity ([Type 302](#)) shall occur in the file, and the Structure Field of the instance (DE Field 3) shall reference the Directory Entry of this definition entity.

Each entity that is a member of an Associativity Instance may contain a back pointer to the Associativity Instance ([see Section 2.2.4.5.2](#)). ECO630

The parameters K and N(1), N(2), . . . . N(K) are specified in the Associativity Definition (see [Section 4.69](#)). ECO630

**4.80.1 Pre-defined Associativities.** As defined in [Section 4.69](#), the Associativity Definition Entity ([Type 302](#)) shall only occur in the file for Form Numbers 5001 through 9999. The following Sections contain the definitions of the pre-defined associativities as they would appear if they were defined by an implementor. Also included in these Sections are the descriptions of each associativity's parameters in a manner similar to other entities in this Specification. ECO630

The general format of the parameter data for an Associativity Instance Entity is:

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NE(1)	Integer	Number of class one entries
2	NE(2)	Integer	Number of class two entries
:	:		
K	NE(K)	Integer	Number of class K entries

For K classes with (NE(1),..., NE(K)) entries with (N(1),..., N(K)) items per entry

1+K	I(1,1,1)	Variable	Class 1, Entry 1, Item 1
.	I(1,1,2)	Variable	Class 1, Entry 1, Item 2
:	:	:	:
.	I(1,1,N(1))	Variable	Class 1, Entry 1, Item N(1)
.	I(1,2,1)	Variable	Class 1, Entry 2, Item 1
:	:	:	:
.	I(1,2,N(1))	Variable	Class 1, Entry 2, Item N(1)
:	:	:	:
.	I(1,NE(1),1)	Variable	Class 1, Entry NE(1), Item 1
:	:	:	:
.	I(1,NE(1),N(1))	Variable	Class 1, Entry NE(1), Item N(1)
.	I(2,1,1)	Variable	Class 2, Entry 1, Item 1
:	:	:	:
.	I(2,1,N(2))	Variable	Class 2, Entry 1, Item N(2)
.	I(2,2,1)	Variable	Class 2, Entry 2, Item 1

#### **4.80 ASSOCIATIVITY INSTANCE ENTITY (TYPE 402)**

⋮	⋮
I(2,2,N(2))	Variable Class 2, Entry 2, Item N(2)
⋮	⋮
I(2,NE(2),N(2))	Variable Class 2, Entry NE(2), Item N(2)
⋮	⋮
I(K,1,1)	Variable Class K, Entry 1, Item 1
⋮	⋮
I(K,NE(K),N(K))	Variable Class K, Entry NE(K), Item N(K)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.81 GROUP ASSOCIATIVITY (TYPE 402, FORM 1)

### 4.81 Group Associativity (Type 402, Form 1)

The Group Associativity allows a collection of entities to be maintained as a single, logical entity. [Figure 111](#) is an example.

There are four form numbers which specify group associativities:

ECO630

<b>Form</b>	<b>Meaning</b>
1	Unordered group with back pointers
7	Unordered group without back pointers
14	Ordered group with back pointers
15	Ordered group without back pointers

The first (Form=1) is defined here; the others are defined in [Sections 4.85 \(Form=7\)](#), [4.89 \(Form=14\)](#), and [4.90 \(Form= 15\)](#), respectively.

### DEFINITION

<b>Index</b>	<b>Set Value</b>	<b>Meaning</b>
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 Entry

(1) Entity Type Number 402	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **????**	(10) Sequence Number D #
(11) Entity Type Number 402	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 1	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

ECO650

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	N	Integer	Number of entries
2	DE(1)	Pointer	Pointer to the DE of the first entity
⋮	⋮	⋮	⋮
1+N	DE(N)	Pointer	Pointer to the DE of the last entity

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## **4.82 VIEWS VISIBLE ASOCIATIVITY (TYPE 402, FORM 3)**

### **4.82 Views Visible Associativity (Type 402, Form 3)**

When an entity is to be displayed in a single view, a pointer to that View Entity ([Type 410](#)) is ECO630 entered in Field 6 of the entity's DE.

If one or more entities are to be displayed in more than one view, but not in all views, Field 6 ECO630 of their Directory Entries shall reference an instance of this entity. This form of the associativity contains two classes of information. The first class contains the number of views in which an entity is visible, followed by references to those views. The optional second class contains the number of entities whose display is specified by this instance, followed by pointers to each of the entities.

#### **DEFINITION**

<b><u>Index</u></b>	<b><u>Set Value</u></b>	<b><u>Meaning</u></b>
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	Back pointers not required
7	2	Unordered
8	1	One item per entry
9	1	Item is a pointer (to other entity)

**DESCRIPTION****Directory Entry**

(1) Entity Type Number 402	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level <i>&lt; n.a. &gt;</i>	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **0001**	(10) Sequence Number D#
(11) Entity Type Number 402	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 3	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence D#+1

**Parameter Data**

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N1	Integer	Number of views visible
2	N2	Integer	Number of entities displayed in these views, or zero
3	DEV(1)	Pointer	Pointer to the DE of the first View Entity
:	:	:	
2+N1	DEV(N1)	Pointer	Pointer to the DE of the last View Entity
3+N1	DE(1)	Pointer	Pointer to the DE of the first entity whose display is being specified by this associativity instance
:	:	:	
2+N2+N1	DE(N2)	Pointer	Pointer to the DE of the last entity whose display is being specified by this associativity instance

Additional pointers as required (see Section 2.2.4.5.2).

## **4.83 VIEWS VISIBLE, COLOR LINE WEIGHT ASSOCIATIVITY (FORM 4)**

### **4.83 Views Visible, Color, Line Weight Associativity (Form 4)**

This associativity is an extension of Form Number 3. Entities that are visible in multiple views, but have a different line font, color number, or line weight in each view, shall reference an instance of this entity from DE Field 6.

In the parameter data portion of the associativity instance, the Parameter N1 shall indicate the number of blocks containing the views visible, line font, color number, and line weight specifications. Each block shall contain a pointer to the View Entity ([Type 410](#)), a line font value or 0, a pointer to a Line Font Definition Entity ([Type 304](#)) if the line font value was 0, a color value or pointer to a Color Definition Entity ([Type 314](#)), and a line weight value. Parameter N2 shall contain the number of entities which are members of this associativity (*i.e.*, entities which have this particular display characteristic) or zero.

If more than one entity appears in Class 2, the complete set of display characteristics in Class 1 applies to each entity in Class 2.

#### **DEFINITION**

<b><u>Index</u></b>	<b><u>Set Value</u></b>	<b><u>Meaning</u></b>
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 Entity
6	2	Line Font value
7	1	Pointer to Line Font Definition Entity
8	3	Color Number (value) or pointer
9	2	Line Weight (value)
		Class 2 (Entity)
10	2	Back pointers not required
11	2	Unordered
12	1	One item per entry
13	1	Item is a pointer (to entity)

## 4.83 VIEWS VISIBLE, COLOR LINE WEIGHT ASSOCIATIVITY (FORM 4)

### DESCRIPTION

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
402	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**0001**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
402	< n.a. >	< n.a. >	#	4			#		D # + 1

#### Parameter Data

ECO650  
ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N1	Integer	Number of blocks containing the view visible, line font, color number, and line weight information
2	N2	Integer	Number of entities which have this particular set of display characteristics, or zero
3	DEV(1)	Pointer	Pointer to the DE of the first View Entity
4	LF(1)	Integer	Line font value or zero
5	DEF(1)	Pointer	Pointer to the DE of the Line Font Definition Entity or zero (only used if LF(1) = 0)
6	CN(1)	Integer or Pointer	Color number value 1 or Pointer to the DE of the Color Definition Entity
7	LW(1)	Integer	Line weight value 1
8	DEV(2)	Pointer	Pointer to the DE of the second View Entity
:	:	:	
2+5*N1	LW(N1)	Integer	Last line weight value
3+5*N1	DE(1)	Pointer	Pointer to the DE of the first entity
:	:	:	
2+N2+5*N1	DE(N2)	Pointer Pointer	Pointer to the DE of the last entity

Additional pointers as required (see Section 2.2.4.5.2).

## **4.84 ENTITY LABEL DISPLAY ASSOCIATIVITY (TYPE 402, FORM 5)**

### **4.84 Entity Label Display Associativity (Type 402, Form 5)**

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, the Label Display Field (Field 8) of the DE contains a pointer to an instance of this associativity.

In the parameter data portion of the associativity instance, the parameter N shall indicate the ECO630 number of blocks containing label placement information. Each block shall reference a View Entity ([Type 410](#)) which specifies the view of visibility. The remaining information (text location, leader, and level number) applies to the label for that view.

#### **DEFINITION**

<b><u>Index</u></b>	<b><u>Set Value</u></b>	<b><u>Meaning</u></b>
1	1	One class
2	2	Back pointers not required
3	1	Ordered
4	7	Seven items per entry
5	1	Pointer to View Entity
6	2	XT of text location
7	2	YT of text location
8	2	ZT of text location
9	1	Pointer to Leader Entity
10	2	Entity label level number
11	1	Pointer to entity

## 4.84 ENTITY LABEL DISPLAY ASSOCIATIVITY (TYPE 402, FORM 5)

### DESCRIPTION

#### Directory Entry

(1) Entity Type Number 402	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **????**	(10) Sequence Number D #
(11) Entity Type Number 402	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 5	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of label placements
2	DEV(1)	Pointer	Pointer to the DE of the first View Entity
3	XT(1)	Real	XT coordinate of text location in first view
4	YT(1)	Real	YT coordinate of text location in first view
5	ZT(1)	Real	ZT coordinate of text location in first view
6	DEARRW(1)	Pointer	Pointer to the DE of the Leader Entity in first view
7	LLN(1)	Integer	Entity label level number in first view
8	DE(1)	Pointer	Pointer to the DE of the first entity being displayed
:	:	:	
-5+7*N	DEV(N)	Pointer	Pointer to the DE of the last View Entity
-4+7*N	XT(N)	Real	XT coordinate of text location in last view
-3+7*N	YT(N)	Real	YT coordinate of text location in last view
-2+7*N	ZT(N)	Real	ZT coordinate of text location in last view
-1+7*N	DEARRW(N)	Pointer	Pointer to the DE of the Leader Entity in last view
7*N	LLN(N)	Integer	Entity label level number in last view
1+7*N	DE(N)	Pointer	Pointer to the DE of the last entity being displayed

Additional pointers as required (see Section 2.2.4.5.2).

## 4.85 GROUP WITHOUT BACK POINTERS ASSOCIATIVITY (FORM 7)

### 4.85 Group Without Back Pointers Associativity (Form 7)

See [Section 4.80](#) for a discussion of Groups.

#### DEFINITION

<u>Index</u>	<u>Set Value</u>	<u>Meaning</u>
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 Entry

(1) Entity Type Number  402	(2) Parameter Data  ⇒	(3) Structure  < n.a. >	(4) Line Font Pattern  < n.a. >	(5) Level  < n.a. >	(6) View  < n.a. >	(7) Xformation Matrix  < n.a. >	(8) Label Display  < n.a. >	(9) Status Number  **?????**	(10) Sequence Number  D #
(11) Entity Type Number  402	(12) Line Weight  < n.a. >	(13) Color Number  < n.a. >	(14) Parameter Line Count  #	(15) Form Number  7	(16) Reserved  	(17) Reserved  	(18) Entity Label  #	(19) Entity Subscript  #	(20) Sequence Number  D # + 1

##### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of entries
2	DE(1)	Pointer	Pointer to the DE of the first entity
:	:	:	
1+N	DE(N)	Pointer	Pointer to the DE of the last entity

Additional pointers as required (see [Section 2.2.4.5.2](#)).

## 4.86 SINGLE PARENT ASSOCIATIVITY (TYPE 402, FORM 9)

### 4.86 Single Parent Associativity (Type 402, Form 9)

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 specified by the parent entity. ECO630

#### **DEFINITION**

<u>Index</u>	<u>Set Value</u>	<u>Meaning</u>
1	2	Two classes
		Class 1 (parent)
2	1	Back pointers required
3	2	Unordered
4	1	One item per entry
5	1	Item is pointer to parent entity
		Class 2 (children)
6	1	Back pointers required
7	1	Ordered
8	1	One item per entry
9	1	Item is pointer to child entity

#### **DESCRIPTION**

##### **Directory Entry**

(1) Entity Type Number	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number ***????**	(10) Sequence Number D #
402									
(11) Entity Type Number	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 9	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

##### **Parameter Data**

ECO650  
ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of parent entities (NP=1 is required)
2	NC	Integer	Number of children
3	DE	Pointer	Pointer to the DE of the parent entity
4	DE(1)	Pointer	Pointer to the DE of the first child entity
:	:	:	
2+NC	DE(NC)	Pointer	ZPointer to the DE of the last child entity

Additional pointers as required (see Section 2.2.4.5.2).

## 4.87 EXTERNAL REFERENCE FILE INDEX ASSOCIATIVITY (FORM 12)

### 4.87 External Reference File Index Associativity (Form 12)

The External Reference File Index Entity appears in one file which contains definitions referenced by another file. It contains a list of the symbolic names used by the referencing files and the DE pointers to the corresponding definitions within the referenced file. [See Section 3.6.4](#) and the External Reference Entity ([Type 416](#)) for more detail.

#### DEFINITION

<u>Index</u>	<u>Set Value</u>	<u>Meaning</u>
1	1	One class (externally referenced entities)
2	2	Back pointers not required
3	2	Unordered list of entries in a class
4	2	Number of items in an entry
5	2	First item is a value (External Reference Entity symbolic name)
6	1	Second item is a pointer (internal entity DE pointer)

#### DESCRIPTION

##### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
402	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**?????**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
402	< n.a. >	< n.a. >	#	12				#	D # + 1

##### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of index entries
2	NAME(1)	String	First External Reference Entity symbolic name
3	PTR(1)	Pointer	Pointer to the DE of the first internal entity
:	:		
2N	NAME(N)	String	Last External Reference Entity symbolic name
1+2*N	PTR(N)	Pointer	Pointer to the DE of the last internal entity

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## **4.88 DIMENSIONED GEOMETRY ASSOCIATIVITY (TYPE 402, FORM 13)**

### **4.88 Dimensioned Geometry Associativity (Type 402, Form 13)**

This entity has been replaced by the new form of the Dimensioned Geometry Associativity Entity ([Type 402, Form 21](#)) and should no longer be used by preprocessors. When that entity ([Type 402, Form 21](#)) has been tested sufficiently, this entity ([Type 402, Form 13](#)) will be moved to the Obsolete Entities Appendix. Its use will then be deprecated.

This associativity links a dimension entity with the geometry entities it is dimensioning. The pointers to the entities being dimensioned have interpretations related to the type of dimension entity. See [Figure 121](#).

#### **DEFINITION**

<b>Index</b>	<b>Set Value</b>	<b>Meaning</b>
1	2	Two classes
	Class 1	(Dimension Entity)
2	1	Back pointers required
3	2	Unordered
4	1	One item (pointer to dimension)
5	1	Item is pointer
	Class 2 (Related Geometry)	
6	2	Back pointers not required
7	2	Unordered
8	1	One item per entry (pointers to geometry)
9	1	Item is pointer

## 4.88 DIMENSIONED GEOMETRY ASSOCIATIVITY (TYPE 402, FORM 13)

### DESCRIPTION

#### Directory Entry

(1) Entity Type Number 402	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level <i>&lt; n.a. &gt;</i>	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **?????**	(10) Sequence Number D #
(11) Entity Type Number 402	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 13	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

EC0650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	ND	Integer	Number of dimensions (ND=1 is required)
2	NG	Integer	Number of associated geometry entities
3	DIMPTR	Pointer	Pointer to the DE of the dimension entity
4	GEOM(1)	Pointer	Pointer to the DE of the first geometry entity
:	:	:	
3+NG	GEOM(NG)	Pointer	Pointer to the DE of the last geometry entity

Additional pointers as required (see Section 2.2.4.5.2).

**4.88 DIMENSIONED GEOMETRY ASSOCIATIVITY (TYPE 402, FORM 13)**

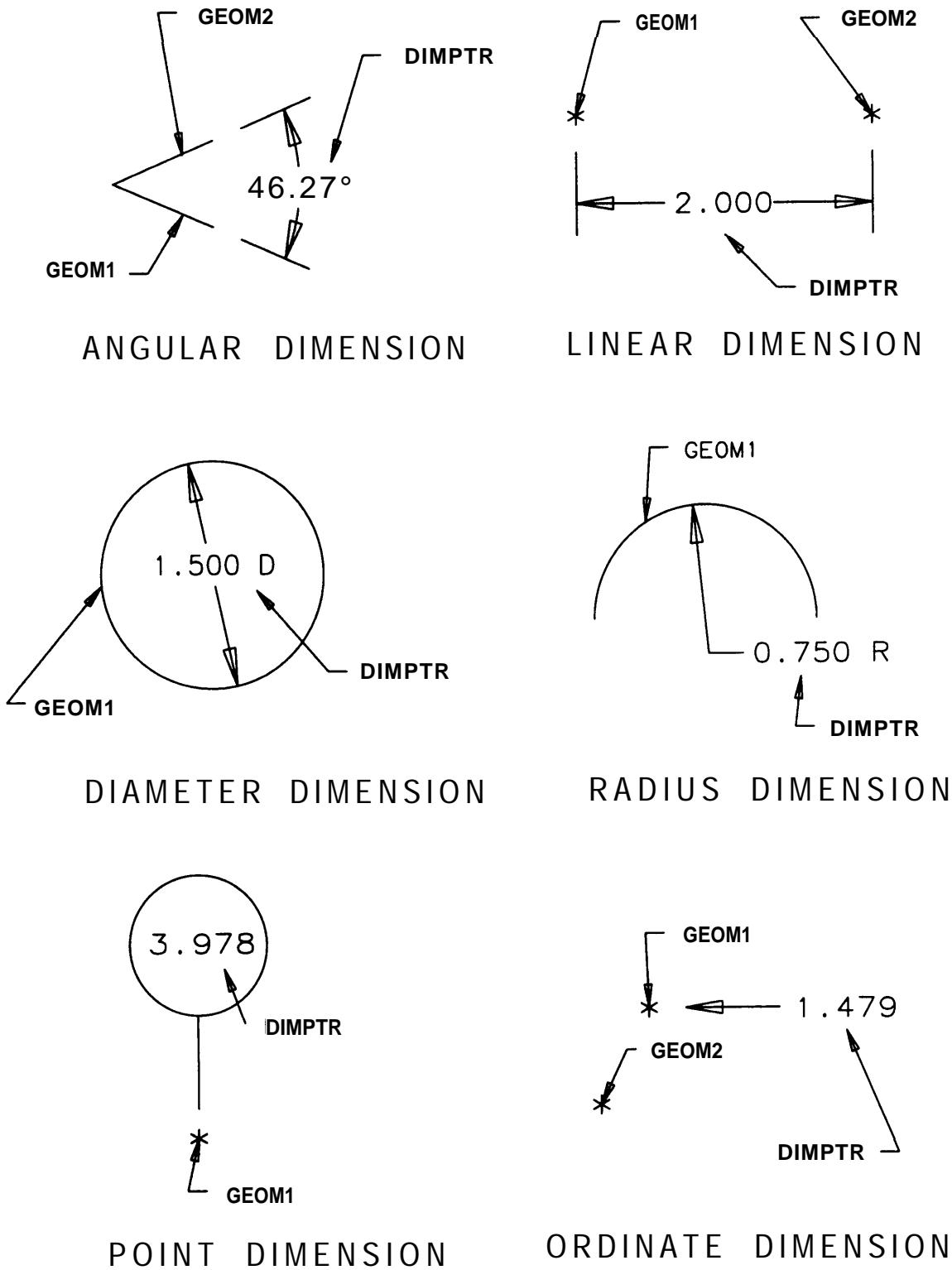


Figure 121. Dimensioned Geometry Associativity

## 4.89 ORDERED GROUP WITH BACK POINTERS ASSOCIATIVITY (FORM 14)

### 4.89 Ordered Group with Back Pointers Associativity (Form 14)

See [Section 4.80](#) for a discussion of Groups.

#### DEFINITION

<u>Index</u>	<u>Set Value</u>	<u>Meaning</u>
1	1	One class
2	1	Back pointers required
3	1	Ordered
4	1	One item per entry
5	1	The item is a pointer

#### DESCRIPTION

##### Directory Entry

(1) Entity Type Number 402	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **?????**	(10) Sequence Number D #
(11) Entity Type Number 402	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 14	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

##### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of entries
2	DE(1)	Pointer	Pointer to the DE of the first entity
:	:	:	
1+N	DE(N)	Pointer	Pointer to the DE of the last entity

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.90 ORDERED GROUP, NO BACK POINTERS ASSOCIATIVITY (FORM 15)

### 4.90 Ordered Group, no Back Pointers Associativity (Form 15)

See Section 4.80 for a discussion of Groups.

#### DEFINITION

<u>Index</u>	<u>Set Value</u>	<u>Meaning</u>
1	1	One class
2	2	Back pointers not required
3	1	Ordered
4	1	One item per entry
5	1	The item is a pointer

#### DESCRIPTION

##### Directory Entry

(1) Entity Type Number 402	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **?????**	(10) Sequence Number D #
(11) Entity Type Number 402	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 15	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript D # + 1	(20) Sequence Number

##### Parameter Data

EC0650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of entries
2	DE(1)	Pointer	Pointer to the DE of the first entity
⋮	⋮	⋮	⋮
1+N	DE(N)	Pointer	Pointer to the DE of the last entity

Additional pointers as required (see Section 2.2.4.5.2).

**4.91 Planar Associativity (Type 402, Form 16)**

This associativity is used to indicate that a collection of entities is coplanar. The entities in the ECO630 collection may be geometric, annotative, or structural. If an entity references subordinate entities, they shall also be coplanar.

The first class contains the pointer to the Transformation Matrix Entity (Type 124) indicating ECO630 the plane to which the entities have been moved. The plane in question is the image, under this transformation, of the XY plane. As noted in the description for DE Field 7, the value 0 may be used to indicate the identity transformation matrix. This matrix is informational only for the associativity; the constituent entities shall be properly positioned in model space.

The second class contains the pointers to the coplanar entities.

**DEFINITION**

<b>Index</b>	<b>Set Value</b>	<b>Meaning</b>
1	2	Two classes
		Class 1 (Transformation Matrix)
2	2	Back pointers not required
3	1	Ordered class
4	1	Number of items per entry
5	1	Pointer
		Class 2 (Coplanar Entities)
6	2	Back pointers not required
7	2	Unordered class
8	1	Number of items per entry
9	1	Pointer

## 4.91 PLANAR ASSOCIATIVITY (TYPE 402, FORM 16)

### DESCRIPTION

#### Directory Entry

(1) Entity Type Number 402	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level <i>&lt; n.a. &gt;</i>	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **?05**	(10) Sequence Number D #
(11) Entity Type Number 402	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 16	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

ECO650  
ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NTR	Integer	Number of Transformation Matrices (NTR= 1 is required)
2	N	Integer	Number of entities in this plane pointed to by this associativity
3	DETR	Pointer	Pointer to the DE of the Transformation Matrix moving data from XY plane into plane of co-planarity, or zero
4	DE(1)	Pointer	Pointer to the DE of the first entity on plane specified
:	:	:	
3+N	DE(N)	Pointer	Pointer to the DE of the last entity on plane specified

Additional pointers as required (see Section 2.2.4.5.2).

#### 4.92 Flow Associativity (Form 18)

The Flow Associativity represents a single signal or a single fluid flow path. The associativity contains seven classes.

Class one contains the type and function flags:

Type Flag	Meaning
0	Not specified (Default)
1	Logical flow
2	Physical flow

The use of the Type Flag is mandatory when both the logical (e.g., schematic) and physical (e.g., printed board) product definitions are in the same file. In such a file, the Type Flag shall not be zero.

The Function Flag differentiates between a fluid path and an electrical conductor:

Function Flag	Meaning
0	Not specified (Default)
1	Electrical signal
2	Fluid flow path

A fluid flow path is a single path from a starting Connect Point entity. The path may include additional intermediate Connect Points, but separate Flow Associativity Entities are required to describe the branch flow paths. The join entities (Class four) and connection entities (Class three) shall be ordered as they occur along the flow path; *i.e.*, the start of the fluid flow path shall be the one listed first; the end of the fluid flow path shall be listed last. ECO630 ECO656

Class two contains pointers to other associated Flow Associativities. These other associativities may implement alternative flow representations. The obvious example of this is a file containing both the schematic and physical product definitions. The corresponding Flow Associativities of each type would be paired. ECO630

Class three is the Link, which contains the list of pointers to the Connect Point Entities involved in the signal or flow. ECO630

Class four is the Join, which contains the list of pointers to the entities representing the graphical implementation of the signal or flow. ECO630

Class five contains the flow names which are associated with the signal or flow. ECO630

Class six contains a list of pointers to the name display entities used to display the first flow name listed in Class five. The reference may point to either a Text Display Template or a General Note which represents a variable text entity. In the case of the Text Display Template Entities, these entities provide the locations and attributes for the signal name display; the text string for display is obtained from the first flow name listed in Class five. ECO630 ECO656

Class seven contains a list of pointers to the flow continuation entities. The flow continuations are represented through a tree of Flow Associativities, where each Flow Associativity represents a single branch within the overall flow. This is an ordered list, where the “main” continuation of the path, if any, shall be listed last. A null pointer shall be used if there is no flow continuation. ECO630 ECO656

**DEFINITION**

<b>Index</b>	<b>Set Value</b>	<b>Meaning</b>	
1	7	Seven classes	
		Class 1 (Context Flag)	
2	2	Back pointers not required	
3	1	Ordered	
4	1	One item per entry	
5	2	Item is value	
		Class 2 (Associated Flows)	
6	2	Back pointers not required	
7	2	Unordered	
8	1	One item per entry	
9	1	Pointer to Flow Associativity	
		Class 3 (Connect Points (Link))	
10	1	Back pointers required	
11	1	Ordered	
12	1	One item per entry	
13	1	Pointer to Connect Point Entity or Group Associativity	EC0656
		Class 4 (Join)	
14	1	Back pointers required	
15	1	Ordered	
16	1	One item per entry	
17	1	Pointer to geometry or Subfigure Instance Entity	
		Class 5 (Flow Name)	
18	2	Back pointers not required	
19	2	Unordered	
20	1	One item per entry	
21	2	Item is value	
		Class 6 (Flow Name Display)	
22	2	Back pointers not required	
23	2	Unordered	
24	1	One item per entry	
25	1	Pointer to Text Display Template Entity or General Note Entity	EC0656
		Class 7 (Flow Continuations)	
26	1	Back pointers required	
27	1	Ordered	
28	1	One item per entry	
29	1	Item is a pointer to a Flow Associativity or a Net Connection As- sociativity	EC0656

**DESCRIPTION****Directory Entry**

(1) Entity Type Number 402	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **?03**	(10) Sequence Number D #
(11) Entity Type Number 402	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 18	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Parameter Data**

ECODH

Index	Name	Type	Description	
1	NCF	Integer	Count of context flags (NCF=2 is required)	
2	NF	Integer	Count of associated Flow Associativities	ECO656
3	NC	Integer	Count of connection entities	
4	NJ	Integer	Count of Join entities (geometry or subfigure)	ECO656
5	NN	Integer	Count of flow names	ECO656
6	NT	Integer	Count of name display entities	ECO656
7	NP	Integer	Count of continuation flow associativities	
8	TF	Integer	Type flag: 0 = not specified (Default) 1 = logical flow 2 = physical flow	
9	FF	Integer	Function flag 0 = not specified (Default) 1 = electrical signal 2 = fluid flow path	
10	SPTR(1)	Pointer	Pointer to the DE of the first Flow Associativity Entity	
⋮	⋮	⋮	⋮	
NF+9	SPTR(NF)	Pointer	Pointer to the DE of the last Flow Associativity Entity	
NF+10	CPTR(1)	Pointer	Pointer to the DE of the first connection entity	ECO656
⋮	⋮	⋮	⋮	
NF+NC+9	CPTR(NC)	Pointer	Pointer to the DE of the last connection entity	ECO656
NF+NC+10	JPTR(1)	Pointer	Pointer to the DE of the first Join Entity	
⋮	⋮	⋮	⋮	
NF+NC+NJ+9	JPTR(NJ)	Pointer	Pointer to the DE of the last Join Entity	
NF+NC+NJ+10	NAME(1)	String	First Flow name	
⋮	⋮	⋮	⋮	
NF+NC+NJ+NN+9	NAME(NN)	String	Last Flow name	
NF+NC+NJ+NN+10	GPTR(1)	Pointer	Pointer to the DE of the first name display entity	ECO656
⋮	⋮	⋮	⋮	
NF+NC+NJ+NN+NT+9	GPTR(NT)	Pointer	Pointer to the DE of the last name display entity	ECO656
NF+NC+NJ+NN+NT+10	CFPTR(1)	Pointer	Pointer to the DE of the first continuation	

#### **4.92 FLOW ASSOCIATIVITY (FORM 18)**

⋮	⋮	⋮	Flow Associativity Entity	
NF+NC+NJ+NN+NT+NP+9	CFPTR(NP)	Pointer	Pointer to the DE of the last continuation entity (the “main” continuation)	ECO656

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.93 SEGMENTED VIEWS VISIBLE ASSOCIATIVITY (TYPE 402, FORM 19)‡

### 4.93 Segmented Views Visible Associativity (Type 402, Form 19)‡

‡The Segmented Views Visible Associativity Entity has not been tested. See Section 1.9.

ECO630

This entity associates display parameters with curves in a view. This entity works in the same way ECO630 as the Views Visible Associativity Entity (Type 402, Form 3 or 4). It is referenced by an entity's DE Field 6 (View).

The curve to be displayed is broken into segments. The display parameters are associated with these ECO630 segments. Segments are defined by the breakpoints between them. The first segment starts at the minimum parameterization value and ends at the first parameter breakpoint. The second segment starts at the first parameter breakpoint and runs to the second breakpoint, etc.

The data in the instances of this entity shall be ordered in ascending parameter breakpoint order ECO630 within a given view. That is, the parameter breakpoints for a given view shall be adjacent in increasing parametric order. There is no particular order for the sequence of views. The last parameter breakpoint shall be equal to the maximum parameterization value for the entity for the final view defined. Negative values for parameters 5 and 6 indicate pointers to Color Definition and Line Font Definition Entities, respectively.

If the data for the display parameters of a segment of the curve are defaulted, *i.e.*, consecutive ECO630 delimiters, the display parameters for that segment shall be taken from the curve's DE values for color, line font, or line weight as required. Receiving systems which do not have the ability to display segments of a curve differently shall apply the curve's DE values for color, line font, and line weight across the entire curve.

### DEFINITION

<b>Index</b>	<b>Set Value</b>	<b>Meaning</b>
1	1	One class
2	2	No Back pointers required
3	1	Ordered class
4	6	Six items per entry
5	1	Pointer to View Entity
6	2	Parameter of Breakpoint
7	2	Display Flag (DE Field 9a)
8	3	Color
9	3	Line Font
10	2	Line Weight

## 4.93 SEGMENTED VIEWS VISIBLE ASSOCIATIVITY (TYPE 402, FORM 19)‡

### DESCRIPTION

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
402	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**0001**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
402	< n.a. >	< n.a. >	#	19			#		D # + 1

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of view/segment blocks
2	PT(1)	Pointer	Pointer to the DE of the first View Entity
3	P(1)	Real	Parameter of first breakpoint
4	DF(1)	Integer	First display flag
5	C(1)	Integer or Pointer	First color value or Pointer to the DE of the first Color Definition Entity if negative
6	L(1)	Integer or Pointer	First line font or Pointer to the DE of the first Line Font Definition Entity if negative
7	W(1)	Integer	First line weight
8	PT(2)	Pointer	Pointer to the DE of the second View Entity (may be the same as PT1 or different)
9	P(2)	Real	Parameter of second breakpoint
:	:	:	
2+6*(N-1)	PT(N)	Pointer	Pointer to the DE of the last View Entity for last segment
3+6*(N-1)	P(N)	Real	Parameter of last segment
:	:	:	
7+6*(N-1)	W(N)	Integer	Last line weight

Additional pointers as required (see Section 2.2.4.5.2).

## 4.94 PIPING FLOW ASSOCIATIVITY (TYPE 402, FORM 20)‡

### 4.94 Piping Flow Associativity (Type 402, Form 20)‡

‡The Piping Flow Associative Entity has not been tested. See Section 1.9.

ECO630

The Piping Flow Associativity represents a single fluid flow path. The associativity contains seven classes.

Class one contains the type flag:

Type Flag	Meaning
0	Not specified (Default)
1	Logical
2	Physical

The use of the Type Flag is mandatory when both the logical (e.g., piping and instrumentation diagrams) and physical (e.g., piping product model) product definitions are in the same file. In such a file, the Type Flag shall not be zero.

Class two contains pointers to other associated Piping Flow Associativities. These other associations may implement alternative flow representations (e.g., a file containing both the logical and physical product definitions). The corresponding Piping Flow Associativities of each type are paired. ECO630

Class three is the Link, which contains the list of pointers to the Connect Point Entities involved in the flow.

Class four is the Join, which contains the list of pointers to the entities representing the graphical implementation of the flow.

Class five contains the flow names which are associated with the flow.

Class six contains a list of pointers to the Text Display Template Entities which specify the display of the first flow name listed in class five. ECO630

Class seven contains a list of pointers to flow paths which branch from the present flow path. This is an ordered list, and the “main” continuation of the path, if any, shall always be listed last. A null pointer shall be used if there is no continuation of the main path. ECO630

**DEFINITION**

<b><u>Index</u></b>	<b><u>Set Value</u></b>	<b><u>Meaning</u></b>
1	7	Seven classes
		Class 1 (Context Flag)
2	2	Back pointers not required
3	1	Ordered
4	1	One item per entry
5	2	Item is value
		Class 2 (Associated Flows)
6	2	Back pointers not required
7	2	Unordered
8	1	One item per entry
9	1	Pointer to Piping Flow Associativity
		Class 3 (Connect Points (Link))
10	2	Back pointers not required
11	1	Ordered
12	1	One item per entry
13	1	Pointer to Connect Point Entity
		Class 4 (Join)
14	2	Back pointers not required
15	1	Ordered
16	1	One item per entry
17	1	Pointer to geometry or Subfigure Instance Entity
		Class 5 (Flow Name)
18	2	Back pointers not required
19	2	Unordered
20	1	One item per entry
21	2	Item is value
		Class 6 (Flow Name Display)
22	2	Back pointers not required
23	2	Unordered
24	1	One item per entry
25	1	Pointer to Text Display Template Entity
		Class 7 (Flow Continuations)
26	2	Back pointers not required
27	1	Ordered
28	1	One item per entry
29	1	Item is a pointer

## 4.94 PIPING FLOW ASSOCIATIVITY (TYPE 402, FORM 20)‡

### DESCRIPTION

#### Directory Entry

(1) Entity Type Number 402	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **??03**	(10) Sequence Number D #
(11) Entity Type Number 402	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 20	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

ECO650  
ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NCF	Integer	Count of context flags (NCF=1 is required)
2	NF	Integer	Count of associated Piping Flow Associativities
3	NC	Integer	Count of Connect Point Entities
4	NJ	Integer	Count of Join entities (geometry or subfigure)
5	NN	Integer	Count of flow names
6	NT	Integer	Count of Text Display Templates for flow name display
7	NP	Integer	Count of continuation piping flow associativities
8	TF	Integer	Type flag: 0 = not specified (Default) 1 = logical 2 = physical
9	SPTR(1)	Pointer	Pointer to the DE of the first Piping Flow Associativity Entity
:	:	:	
NF+8	SPTR(NF)	Pointer	Pointer to the DE of the last Piping Flow Associativity Entity
NF+9	CPTR(1)	Pointer	Pointer to the DE of the first Connect Point Entity
:	:	:	
NF+NC+8	CPTR(NC)	Pointer	Pointer to the DE of the last Connect Point Entity
NF+NC+9	JPTR(1)	Pointer	Pointer to the DE of the first Join Entity
:	:	:	
NF+NC+NJ+8	JPTR(NJ)	Pointer	Pointer to the DE of the last Join Entity
NF+NC+NJ+9	NAME(1)	String	First Flow name
:	:	:	
NF+NC+NJ+NN+8	NAME(NN)	String	Last Flow name
NF+NC+NJ+NN+9	GPTR(1)	Pointer	Pointer to the DE of the first Text Display Template Entity
:	:	:	

**4.94 PIPING FLOW ASSOCIATIVITY (TYPE 402, FORM 20)‡**

NF+NC+NJ+NN+NT+8	GPTR(NT)	Pointer	Pointer to the DE of the last Text Display Template Entity
NF+NC+NJ+NN+NT+9	CFPTR(1)	Pointer	Pointer to the DE of the first continuation Flow Associativity Entity
:	:	:	
NF+NC+NJ+NN+NT+NP+8	CFPTR(NP)	Pointer	Pointer to the DE of the last continuation Flow Associativity Entity (the “main” continuation)

Additional pointers as required (see Section 2.2.4.5.2).

## 4.95 DIMENSIONED GEOMETRY ASSOCIATIVITY (TYPE 402, FORM 21)‡

### 4.95 Dimensioned Geometry Associativity (Type 402, Form 21)‡

‡The Dimensioned Geometry Associativity Entity has not been tested. See Section 1.9.

ECO630

This entity is intended to replace the existing Dimensioned Geometry Associativity Entity (Type 402, Form 13). When this entity has been tested, that entity (Type 402, Form 13) will be moved to the Obsolete Entities Appendix and its use will be deprecated.

ECO630

This associativity links a dimension entity with the geometry entities it is dimensioning, so that later, in the receiving database, the dimension can be automatically recalculated and redrawn should the geometry be changed.

Due to the generality of the allowed geometry, the GEOMn\_PNTs have been introduced to help postprocessors resolve ambiguous situations. GEOMn\_PNT refers to the coordinates (GPXn, GPYn, GPZn) which are associated with the pointer GEOMn. A GEOMn\_PNT is a point in model space at the point of interest on the geometry. Examples of “points of interest” are a corner of a solid or an endpoint of a line.

If there are two arrowheads in the dimension, there shall be two geometry pointers (GEOM1 and GEOM2, NG=2) even if that means repeating a single pointer twice, as in the case of dimensioning a line. This allows space for GEOM1\_PNT and GEOM2\_PNT. In a straightforward case like dimensioning a line, some postprocessors may ignore the two points as redundant information, but others may depend on them.

ECO630

The dimension orientation flag (DOF) is used for angular, ordinate, and linear dimensions. The values 0-3 are used by the Angular Dimension (Type 202). The values 47 are used by the Linear Dimension (Type 216, all forms), and the Ordinate Dimension (Type 218).

In the case of angular dimensioning, the angle to be measured is between the first piece of geometry (GEOM1) and the second (GEOM2). Both pieces of geometry are then projected onto the plane of the dimension, and the measurement of the angle is taken in a counterclockwise direction. There are four such angles. The one intended is indicated by the DOF and GEOM1\_PNT and GEOM2\_PNT, where neither of these two points shall be the vertex.

ECO630

In the case of ordinate dimensioning, GEOM1 shall point to the particular geometry whose distance from a baseline is to be measured. GEOM2 shall be set to 0, and the related GEOM2\_PNT shall be set to the origin of the baseline. The DOF value shall be determined on the basis of whether the distance to be dimensioned is horizontal or vertical in the dimension's definition space.

The values of DOF are as follows (see [Figure 122](#) and [Figure 123](#)):

- 0 The angle starts on the same side of the vertex as GEOM1\_PNT and ends on the same side as GEOM2\_PNT. This is the default.
- 1 The angle starts on the opposite side of the vertex from GEOM1\_PNT, but ends on the same side as GEOM2\_PNT.
- 2 The angle starts on the same side of the vertex as GEOM1\_PNT, and ends on the opposite side as GEOM2\_PNT.
- 3 The angle starts on the opposite side of the vertex from GEOM1\_PNT and ends on the opposite side from GEOM2\_PNT.
- 4 The dimension is a true dimension measuring the Euclidean distance between GEOM1\_PNT and GEOM2\_PNT in model space.

#### **4.95 DIMENSIONED GEOMETRY ASSOCIATIVITY (TYPE 402, FORM 21)‡**

- 5 The dimension is a parallel dimension measuring the distance between GEOM1\_PNT and GEOM2\_PNT in the XT-YT plane of the definition space of the dimension.
- 6 The dimension is a vertical dimension measuring the YT distance in definition space of the dimension between GEOM1\_PNT and GEOM2\_PNT.
- 7 The dimension is a horizontal dimension measuring the XT distance in definition space of the dimension between GEOM1\_PNT and GEOM2\_PNT.
- 8 The dimension is an AT-ANGLE dimension measuring the distance between GEOM1\_PNT and GEOM2\_PNT in definition space parallel to a line at angle AV with respect to the XT axis.

The dimension location flag (DLF) indicates the relationship between the associated geometry and the position being dimensioned. The values of DLF are as follows:

- 0 End Point (default). The position being dimensioned is the endpoint of the associated geometry nearest the corresponding GEOMn\_NT.
- 1 Center. The position being dimensioned is the center of the associated geometry. The corresponding GEOMn\_PNT is ignored.
- 2 Tangent Point. The position being dimensioned is the point on the associated geometry where the tangent to the geometry is perpendicular to the direction in which the dimension is being measured. If multiple points qualify, the one nearest the corresponding GEOMn\_PNT is used.
- 3 Perpendicular Point. The position being dimensioned is the point on the associated geometry where the normal to the geometry is perpendicular to the direction in which the dimension is being measured. If multiple points qualify, the one nearest the corresponding GEOMn\_PNT is used.
- 4 Relative Parameter Value. The position being dimensioned is the point on the associated geometry nearest the corresponding GEOMn\_PNT. If the geometry is modified, the new dimension position is at the same relative parameter value on the resulting geometry as the original position was on the original geometry.
- 5 Relative Arc Length. The position being dimensioned is the point on the associated geometry nearest the corresponding GEOMn\_PNT. If the geometry is modified, the new dimension position is at the same relative arc length on the resulting geometry as the original position was on the original geometry.

[Figure 124](#) illustrates the effects of each value of DLF.

An instance of this property shall have its Subordinate Entity Switch set to Physically Dependent; ECO630 it shall be referenced by exactly one dimension entity backpointer

**DEFINITION**

<b><u>Index</u></b>	<b><u>Set Value</u></b>	<b><u>Meaning</u></b>
1	2	Two classes
		Class 1 (Dimension Entity)
2	1	Back pointers required
3	2	Unordered
4	3	Three items per entry
5	1	Pointer to a Dimension Entity
6	2	Dimension Orientation Flag
7	2	Angle value
		Class 2 (Related Geometry)
8	2	Back pointers not required
9	2	Ordered
10	5	Five items per entry
11	1	Pointer to geometry
12	2	Dimension location flag
13	2	X coordinate of a point on the geometry
14	2	Y coordinate of a point on the geometry
15	2	Z coordinate of a point on the geometry

## 4.95 DIMENSIONED GEOMETRY ASSOCIATIVITY (TYPE 402, FORM 21)‡

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
402	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**0102**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
402	< n.a. >	< n.a. >	#	21				#	D # + 1

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	ND	Integer	Number of dimensions (ND=1)
2	NG	Integer	Number of associated geometry entities
3	DIMPTR	Pointer	Pointer to the DE of the dimension entity
4	DOF	Integer	Dimension Orientation Flag
5	AV	Real	Angle Value
6	GEOM(1)	Pointer	Pointer to the DE of the first geometry entity
7	DLF(1)	Integer	Dimension location flag for GEOM(1)
8	GPX(1)	Real	Coordinate of point on GEOM(1)
9	GPY(1)	Real	
10	GPZ(1)	Real	
:	:	:	
NG*5+1	GEOM(NG)	Pointer	Pointer to the DE of the last geometry entity or zero
NG*5+2	DLF(NG)	Integer	Dimension location flag for GEOM(NG)
NG*5+3	GPX(NG)	Real	Coordinate of point on GEOM(NG) or origin of base line
NG*5+4	GPY(NG)	Real	
NG*5+5	GPZ(NG)	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.95 DIMENSIONED GEOMETRY ASSOCIATIVITY (TYPE 402, FORM 21)‡**

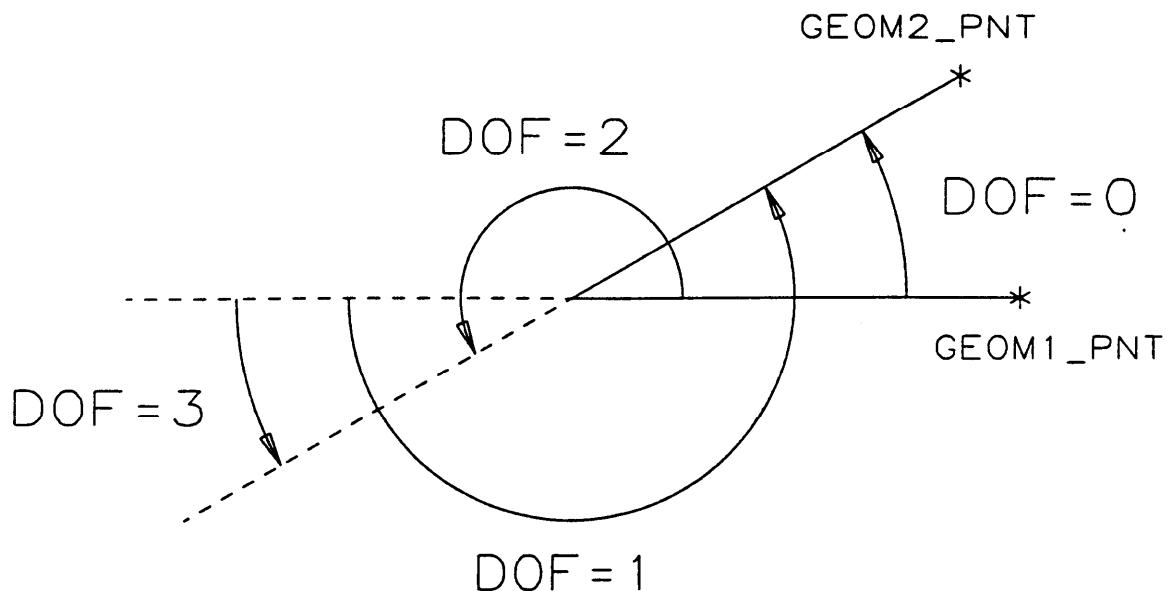


Figure 122. Use of DOF with Angular Dimensions

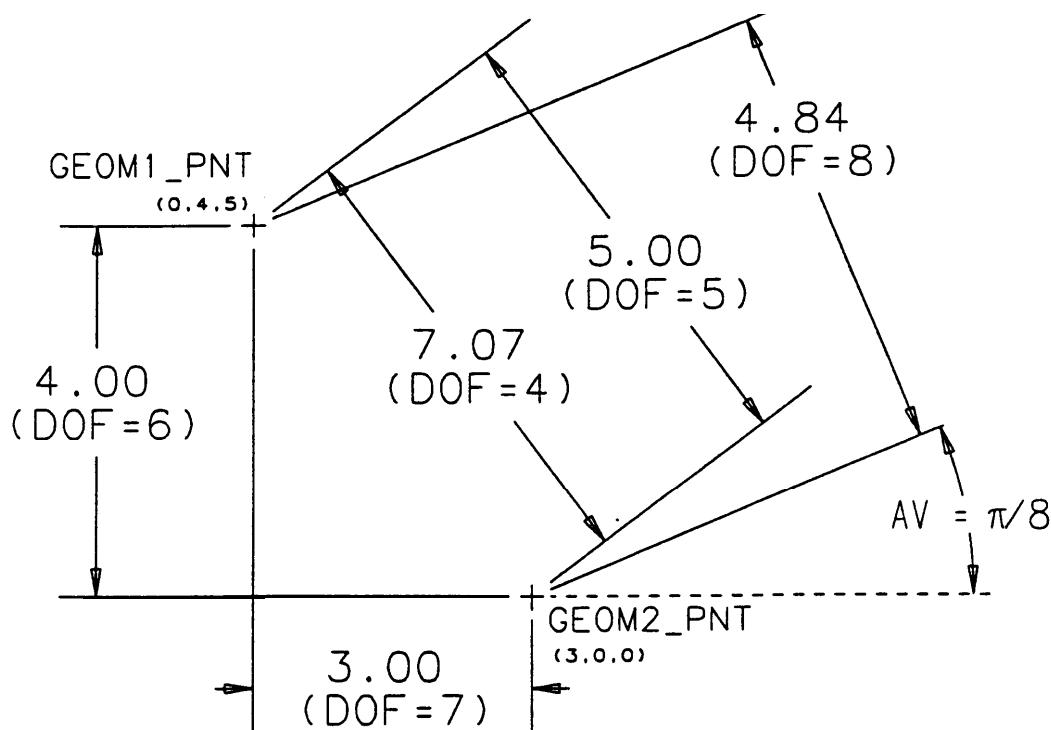
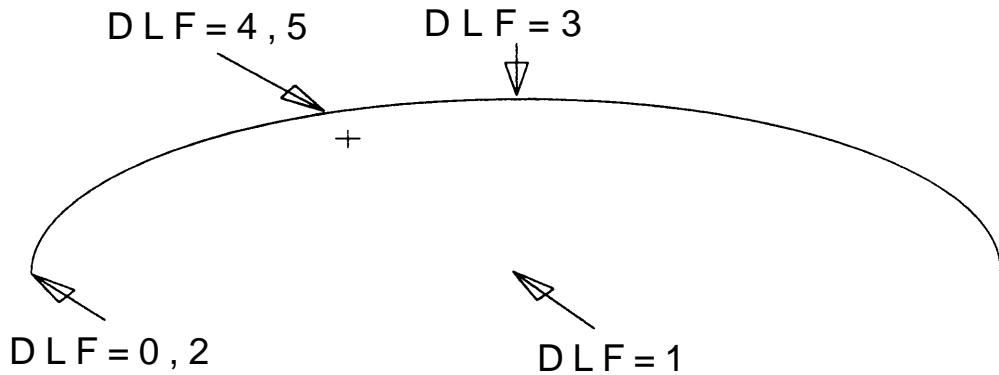
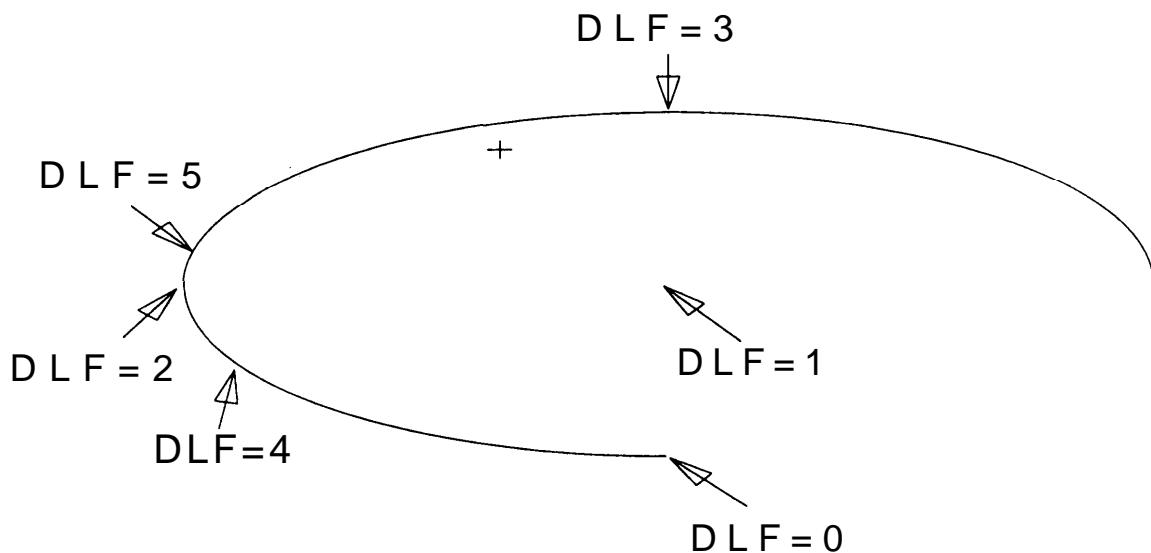


Figure 123. Use of DOF with Linear and Ordinate Dimensions

**4.95 DIMENSIONED GEOMETRY ASSOCIATIVITY (TYPE 402, FORM 21)‡**



Original



After Modification

Figure 124. Use of DLF

#### 4.96 Drawing Entity (Type 404)

The Drawing Entity specifies a drawing as a collection of annotation entities *i.e.*, any entity with its Entity Use Flag set to 01) defined in drawing space, and views (*i.e.*, projections of model space data in view space). The collection depicts a part in the same way that an engineering drawing depicts a part in standard drafting practice. Views are specified by referencing View Entities (Type 410). If desired, multiple drawings can be included in a single file, referring to the same model space.

Drawings are located in drawing space as illustrated in Figure 125, with sides coincident with the drawing coordinate system axes and with the lower left corner at the origin (0,0). The drawing space coordinate system (XD, YD) is a special 2-dimensional coordinate system used for view origin locations in the Drawing Entity and for annotation entities referenced by the Drawing Entity. Any Z coordinates are ignored in the referenced annotation entities, and any transformation matrix from definition space to drawing space must be 2-dimensional (*i.e.*, in the Transformation Entity (Type 124),  $T_3 = R_{13} = R_{31} = R_{32} = R_{23} = 0.0$  and  $R_{33} = 1.0$ ).

Annotation entities can be defined in drawing space and be referenced by the Drawing Entity directly, or can be defined in model space and appear in individual views. When defined in drawing space, the annotation entities shall have physically dependent (01) status. A View Entity referenced by the Drawing Entity shall have logically dependent (02) status.

The transformation of a view from view space to drawing space is controlled by the view scale factor S, specified in the View Entity, and the view origin drawing locations, specified in the Drawing Entity. For orthographic parallel projection, the transformation is:

$$\begin{bmatrix} XD \\ YD \end{bmatrix} = \begin{bmatrix} S & 0 & 0 \\ 0 & S & 0 \end{bmatrix} \begin{bmatrix} XV \\ YV \\ ZV \end{bmatrix} + \begin{bmatrix} X\text{ORIGIN} \\ Y\text{ORIGIN} \end{bmatrix}$$

where S represents the SCALE parameter in the View Entity,

$$\begin{bmatrix} XV \\ YV \\ ZV \end{bmatrix}$$

denotes the view space coordinates, and

$$\begin{bmatrix} X\text{ORIGIN} \\ Y\text{ORIGIN} \end{bmatrix}$$

denotes the drawing space coordinates of the origin of the transformed view (see Section 4.134).

The following formula defines view scale:

$$S = L_d / L_m$$

where S = View scale

$L_d$  = Length in drawing space units

$L_m$  = Length in model space units

The following formula relates the view scale (parameter 2 of the View Entity (Type 410)), the length of an entity as measured in model space units, and the length of an entity as measured in drawing space units:

$$L_d = L_m \cdot S$$

The above formulas always apply, even when drawing units differ from model space units (see Drawing Units Property (Type 406, Form 17)).

**EXAMPLES:** In a file where the model space units are inches, and the drawing space units are centimeters, the following cases illustrate correct scale factor usage:

- A view scale of 2.54 means that a line which is 1 inch long in model space is to be presented on the drawing as 2.54 centimeters long.
- A view scale of 5.08 means that a line which is 1 inch long in model space is to be presented on the drawing as 5.08 centimeters long.

Some CAD systems maintain a rotation, in addition to a translation and scaling, between the view and drawing coordinate systems. It is not possible to correctly capture the relationships among all three coordinate systems—model, view and drawing—using [Form 0](#) of the Drawing Entity. A rotation is needed in addition to the translation for transforming view to drawing coordinates provided by [Form 0](#). A [Form 1](#) is defined which shall be used in this case.

As with Form 0, the transformation for Form 1 is controlled by the view scale factor S and the view origin drawing location. In addition, a rotation angle q is applied as follows:

$$\begin{bmatrix} XD \\ YD \end{bmatrix} = S \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \end{bmatrix} \begin{bmatrix} XV \\ YV \\ ZV \end{bmatrix} + \begin{bmatrix} X_{ORIGIN} \\ Y_{ORIGIN} \end{bmatrix}$$

Systems not having the ability to apply a rotation between their view and drawing coordinate systems will have to choose which of the two to keep correctly. It is recommended that drawing coordinates be maintained in preference to view coordinates in all cases where both coordinate systems cannot be maintained in the receiving system. To do this, the rotation must be incorporated into the transformation from Model to View coordinates.

If there is plane clipping, the situation is more complex, as clipping is done in View coordinates. In this case, conceptually (there are other ways of obtaining the same result), the following must be done:

- Transform from model to view space.
- Perform clipping.
- Perform projection onto the view plane.
- Transform from view space to drawing space.

The name of the drawing may be provided by using the Name Property ([Type 406, Form 15](#)).

The size of the drawing may be specified by using the Drawing Size Property ([Type 406, Form 16](#)).

The units for drawing space may be set differently from the model space units specified in the Global Section by use of the Drawing Units Property Entity ([Type 406, Form 17](#)). When this property is not referenced by a drawing, that drawing's units are the same as the model units.

The following values are given in drawing units:

- view origin drawing locations
- drawing size
- coordinates of annotation entities referenced directly

Refer to [Figures 125](#) and [126](#) for examples of the use of the Drawing Entity.

**Directory Entry**

(1) Entity Type Number 404	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **0001**	(10) Sequence Number D #
(11) Entity Type Number 404	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

ECO630

**Drawing Entity, Form 0****Parameter Data**

ECO650

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of View pointers or zero (default)
2	VPTR(1)	Pointer	Pointer to the DE of the first View Entity
3	XORIGIN(1)	Real	Drawing space coordinate of the origin of the first View Entity
4	YORIGIN(1)	Real	Drawing space coordinate of the origin of the first View Entity
5	VPTR(2)	Pointer	Pointer to the DE of the second View Entity
:	:	:	
2+3*N	M	Integer	Number of Annotation Entities or zero (default)
3+3*N	DPTR(1)	Pointer	Pointer to the DE of the first annotation entity in this Drawing
:	:	:	
2+M+3*N	DPTR(M)	Pointer	Pointer to the DE of the last annotation entity in this Drawing

Additional pointers as required (see Section 2.2.4.5.2).

**Drawing Entity, Form 1****Parameter Data**

ECO650

ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of View pointers or zero (default)
2	VPTR(1)	Pointer	Pointer to the DE of the first View Entity
3	XORIGIN(1)	Real	Drawing space <i>x</i> coordinate of the origin of the first View Entity
4	YORIGIN(1)	Real	Drawing space <i>y</i> coordinate of the origin of the first View Entity
5	ANGLE(1)	Real	Orientation angle in radians for first View Entity (default = 0.0)
:	:	:	
-2+4*N	VPTR(N)	Pointer	Pointer to the DE of the last View Entity
-1+4*N	XORIGIN(N)	Real	Drawing space <i>x</i> coordinate of the origin of the last View Entity
4*N	YORIGIN(N)	Real	Drawing space <i>y</i> coordinate of the origin of the last View Entity
1+4*N	ANGLE(N)	Real	Orientation angle in radians for last View Entity (default= 0.0)
2+4*N	M	Integer	Number of Annotation Entities or zero (default)
3+4*N	DPRT(1)	Pointer	Pointer to the DE of the first annotation entity in this Drawing
:	:	:	

#### **4.96 DRAWING ENTITY (TYPE 404)**

2+M+4\*N DPRT(M) Pointer Pointer to the DE of the last annotation entity in this Drawing

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### 4.96 DRAWING ENTITY (TYPE 404)

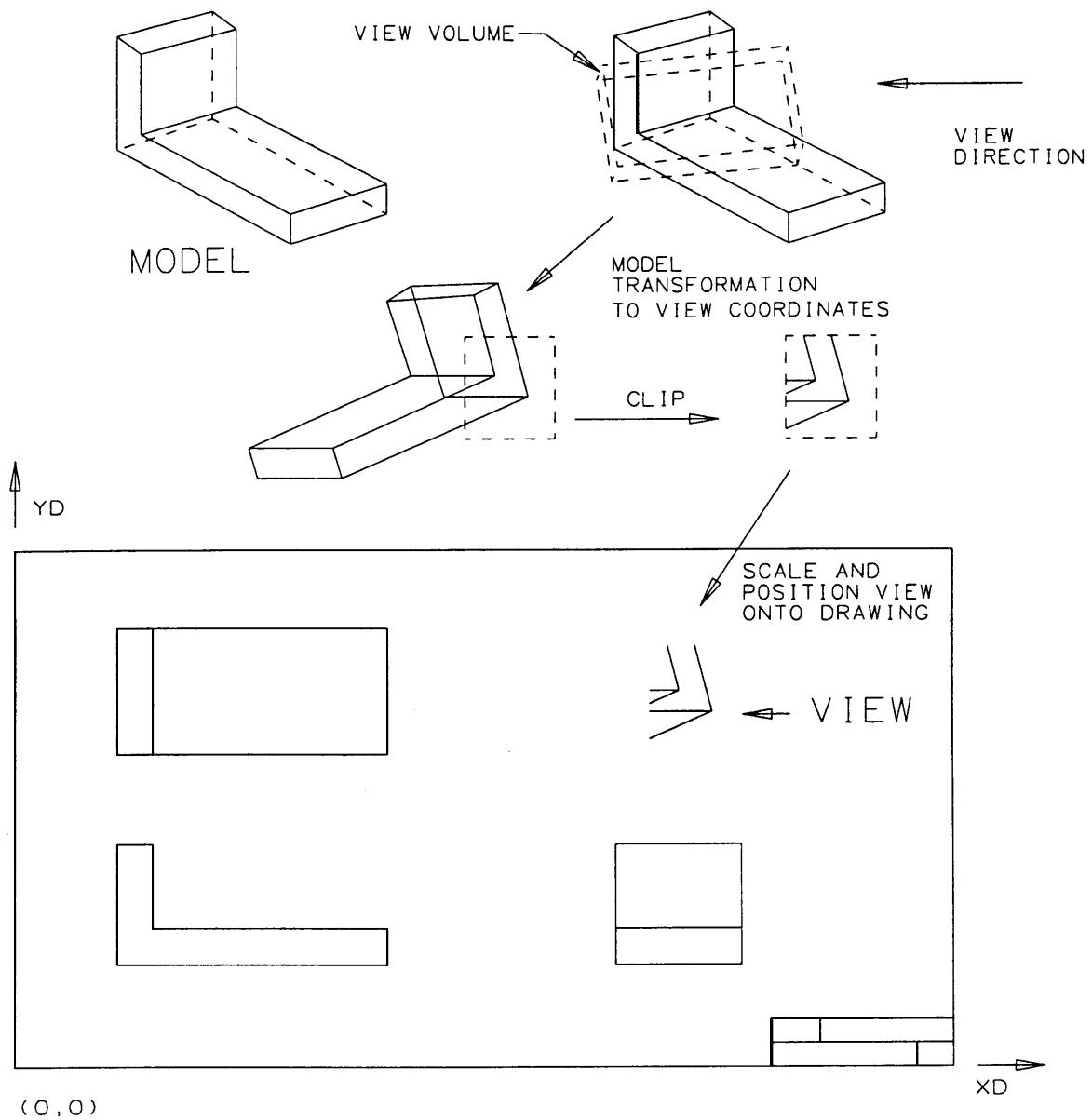


Figure 125. Using Clipping Planes with a View in a Drawing

#### 4.96 DRAWING ENTITY (TYPE 404)

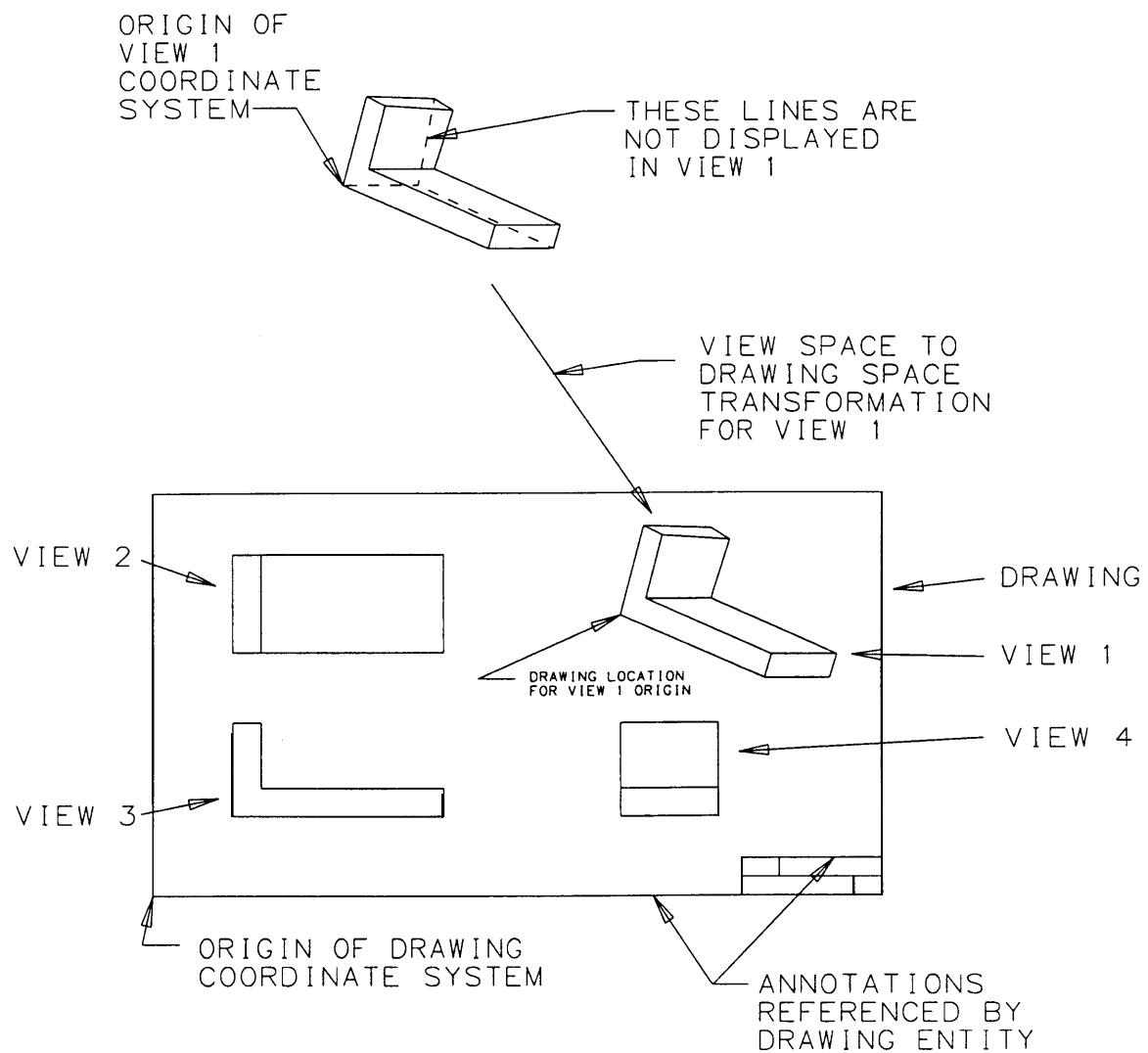


Figure 126. Parameters of the Drawing Entity

**4.97 Property Entity (Type 406)**

The Property Entity contains numerical or textual data. Its Form Number specifies its meaning. ECO630 Form Numbers in the range 5001–9999 are reserved for implementors.

Note that properties may also reference other properties, participate in associativities, reference ECO630 related general notes, or display text by referencing a Text Display Template Entity ([Type 312](#)).

Properties usually are referenced by a pointer in the second group of additional pointers as described ECO630 in [Section 2.2.4.5.2](#); however, as stated in [Section 1.6.1](#), when a property is independent, it applies to all entities on the same level as its Directory Entry Level attribute.

The parameter data values have the following common format for all Property Entities:

**Parameter Data**

ECO650

<b><u>Index</u></b>	<b><u>Name</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
1	N P	Integer	Number of property values
2	V(1)	Variable	First property value
:	:	:	
1+NP	V(NP)	Variable	Last property value

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.98 DEFINITION LEVELS PROPERTY (FORM 1)

### 4.98 Definition Levels Property (Form 1)

For one or more entities in the file that are defined on a set of multiple levels, there shall be ECO630 an occurrence of the Property Instance (Form 1). In the parameter data portion of the property instance, the first parameter, NP, shall contain the number of multiple levels followed by a list of those levels. Each entity that is defined on this set of levels shall contain a pointer (in the level field of the directory entry) to this property instance. A different set of multiple levels shall result in a different property instance.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #, ⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number ***?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 1	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

#### Parameter Data

ECO650

Index	Name	Type	Description
1	NP	Integer	Number of property values
2	L(1)	Integer	First level number
:	:	:	
1+NP	L(NP)	Integer	Last level number

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.99 REGION RESTRICTION PROPERTY (FORM 2)

### 4.99 Region Restriction Property (Form 2)

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

The Directory Entry attribute Level Number is used to specify the physical LEP layers to which the Region Restriction Property is applied. The method used to convey this information is as follows:

1. Create a Definition Levels Property ([Type 406, Form 1](#)).
2. Include a level number for each physical LEP layer to which the Region Restriction Property is applied.
3. Reference the Definition Levels Property from the Directory Entry Level attribute of the Region Restriction Property through a negated pointer.

The values in the Definition Level Property are exchange file level numbers. In order to determine the actual physical LEP layers, the postprocessor must refer to the physical layer number in the Level to LEP Layer Map Property ([Type 406, Form 24](#)).

**Note:** The Directory Entry Level attribute of the boundary curve is used to determine the level(s) upon which the graphic representation of the region is displayed. If the graphic representation is to be displayed on each level to which the Region Restriction is applied, the boundary curve shall point to the same Definition Levels Property as the Region Restriction Property.

Each of the property values in this property shall have one of three values indicating the region restriction relevant to the application's item.

Property	Value	Description
	0	No Restriction
	1	Item must be inside region
	2	Item must be outside region

## 4.99 REGION RESTRICTION PROPERTY (FORM 2)

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	⇒	< n.a. >	< n.a. >	#,⇒	< n.a. >	< n.a. >	< n.a. >	**??****	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16)	(17)	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
406	< n.a. >	< n.a. >	#	2	Reserved	Reserved		#	D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see Sections 4.97 and 1.6.1).

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=3)
2	EVR	Integer	Electrical vias restriction (EVR=0,1 or 2)
3	ECPR	Integer	Electrical components restriction (ECPR=0,1 or 2)
4	ECRR	Integer	Electrical circuitry restriction (ECRR=0, 1 or 2)

Additional pointers as required (see Section 2.2.4.5.2).

## 4.100 LEVEL FUNCTION PROPERTY (FORM 3)

### 4.100 Level Function Property (Form 3)

This property specifies the meaning or intended use of a level in the sending system. An instance ECO630 of this property shall apply to all entities in the same file with the same DE level value (Field 5), without the requirement of a pointer to it ([see Section 1.6.1](#)). Parameter 2 is used to record an integer code number when the sending system uses a level-use index or table. Parameter 3 is used to record the level-use text, whether such text is obtained from the index which provided Parameter 2, or exists independently. Either Parameter 2 or Parameter 3 may have a default value. This property may be readily added to a file (by edit or data merge) when level-use information is required by the receiving system or archive. The Parameter (2 and 3) values of an instance of this property shall apply to multiple levels if the instance's level value is a pointer to an instance of the Definition Levels Property Entity ([Type 406, Form 1](#)).

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level #, ⇒	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **00****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 3	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=2)
2	FC	Integer	Function description code (Default = 0)
3	FD	String	Function description (Default = null string)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.101 LINE WIDENING PROPERTY (FORM 5)

### 4.101 Line Widening Property (Form 5)

This property defines the characteristics of entities used to define the location of items such as strips ECO649 of metalization on LEPs. ECO630

The justification flag terminology is interpreted as follows: Right justified means that a defining line ECO630 segment forms the right edge of the widened line in the direction from first defining point to second. (The entire widened line appears to the left of the defining line. Side is determined from point 1 to point 2. See [Figure 127](#).) Left justified is the opposite, while center justified indicates that the defining line segment splits the widening exactly in half. [Figure 127](#) indicates the measurement of the property values.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #, ⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number ***?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 5	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=5)
2	WM	Real	Width of metalization
3		Integer	Cornering codes: 0 = rounded 1 = squared
4	EF	Integer	Extension flag: 0 = No extension 1 = One-half width extension 2 = Extension set by Parameter 6
5	JF	Integer	Justification flag: 0 = center justified 1 = left justified 2 = right justified
6	E	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#))

#### 4.101 LINE WIDENING PROPERTY (FORM 5)

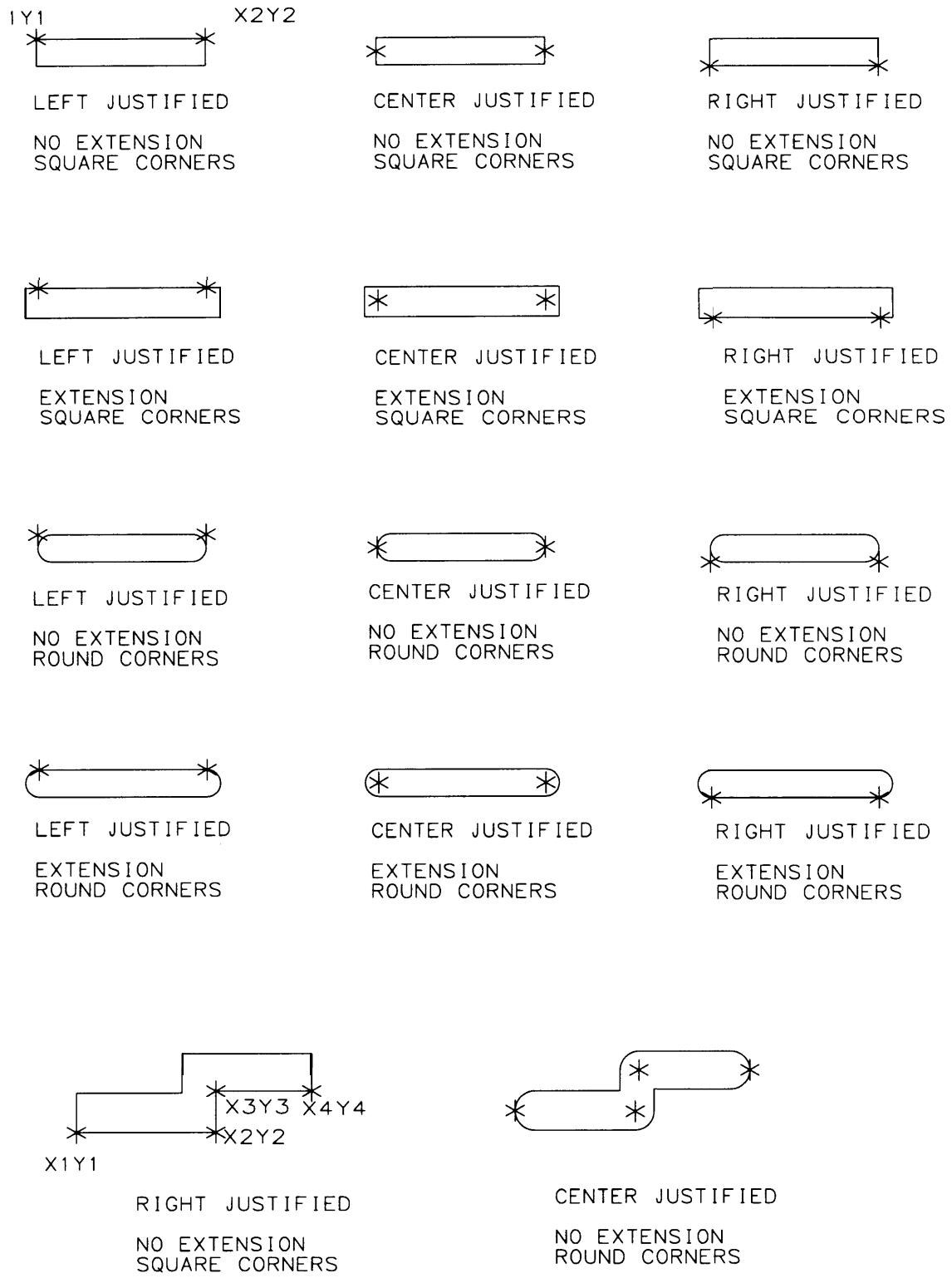


Figure 127. Measurement of the Line Widening Property Values

## 4.102 DRILLED HOLE PROPERTY (FORM 6)

### 4.102 Drilled Hole Property (Form 6)

The Drilled Hole Property identifies an entity representing a drilled hole through a LEP. 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 LEP.

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	$\Rightarrow$	$< n.a. >$	$< n.a. >$	#, $\Rightarrow$	$< n.a. >$	$< n.a. >$	$< n.a. >$	**?*****	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
406	$< n.a. >$	$< n.a. >$	#	6				#	D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97](#) and [1.6.1](#)).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=5)
2	DDS	Real	Drill diameter size
3	FDS	Real	Finish diameter size
4	PF	Integer	Plating indication flag: 0 = no 1 = yes
5	LNL	Integer	Lower numbered layer
6	HNL	Integer	Higher numbered layer

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.103 REFERENCE DESIGNATOR PROPERTY (FORM 7)

### 4.103 Reference Designator Property (Form 7)

The Reference Designator Property attaches a text string containing the value of a component reference designator to an entity representing a component. This property shall not be used for the primary reference designator when a component is represented by a Network Subfigure Instance Entity ([Type 420](#)), as reference designator is included in the subfigure parameters.

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	$\Rightarrow$	$< n.a. >$	$< n.a. >$	$\#, \Rightarrow$	$< n.a. >$	$< n.a. >$	$< n.a. >$	**?*****	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
406	$< n.a. >$	$< n.a. >$	#	7				#	D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=1)
2	RD	String	Reference designator text

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.104 PIN NUMBER PROPERTY (FORM 8)

### 4.104 Pin Number Property (Form 8)

The Pin Number Property attaches a text string representing a component pin number to an entity representing an electrical component's pin. This property shall not be used when a pin is represented by a Connect Point Entity ([Type 132](#)), as the pin number is included in the Connect Point parameters.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level #, ⇒	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 8	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97](#) and [1.6.1](#)).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=1)
2	PN	String	Pin Number Value

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.105 PART NUMBER PROPERTY (FORM 9)

### 4.105 Part Number Property (Form 9)

The Part Number Property attaches a set of text strings that define the common part numbers to ECO630 an entity representing a physical component. Defaulted strings in any parameter imply that the defaulted value is not relevant to the data.

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	⇒	< n.a. >	< n.a. >	#, ⇒	< n.a. >	< n.a. >	< n.a. >	**??*****	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
406	< n.a. >	< n.a. >	#	9				#	D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

#### Parameter Data

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	NP	Integer	Number of property values (NP=4)
2	GPN	String	Generic part number or name
3	MPN	String	Military Standard (MIL-STD) part number
4	VPN	String	Vendor part number or name
5	IPN	String	Internal part number

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.106 HIERARCHY PROPERTY (FORM 10)

### 4.106 Hierarchy Property (Form 10)

The Hierarchy Property specifies the hierarchy of each directory entry attribute. This property is ECO630 referenced when the directory entry status digits 7 and 8 are 02. Acceptable values for Parameters 2 through 7 are 0 and 1. (See definition in [Section 2.2.4.4.9.4](#)).

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	⇒	< n.a. >	< n.a. >	#, ⇒	< n.a. >	< n.a. >	< n.a. >	**?*****	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
406	< n.a. >	< n.a. >	#	10				#	D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=6)
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	CO	Integer	Color number

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.107 TABULAR DATA PROPERTY (FORM 11)

### 4.107 Tabular Data Property (Form 11)

ECO630

The Tabular Data Property provides a structure to accommodate point-form data. The basic structure is a two-dimensional array organized in column-row order. In a simplified form, this structure may contain a single list of values; the more complex forms contain multiple lists of independent and dependent variables.

The Property Type is the key used to define the dependent variable data values. Property Types 1 to 5000 are reserved for defining finite element material properties. Property Types are listed in the following table and are defined in the following Subsections.

PTYPE	Property Type	ND
1	Young's Modulus	3
2	Poisson's Ratio	3
3	Shear Modulus	3
4	Material Matrix	21
5	Mass Density	1
6	Thermal Expansion Coefficient	3
7	Laminate Material Stiffness Matrix	6
8	Bending Material Stiffness Matrix	6
9	Transverse Shear Material Stiffness Matrix	3
10	Bending Coupling Material Stiffness Matrix	6
11	Material Coordinate System	3
12	Nodal Load/Constraint Data	Number of Degrees of Freedom
13	Sectional Properties for Beam Elements if the properties are the same at both ends of the beam otherwise,	8 16
14	Beam End Releases	12
15	Offsets	9 or 18
16	Stress Recovery Information	12 or 24
17	Element Thickness	1 or n
18	Non-Structural Mass	1
19	Thermal Conductivity	3
20	Heat Capacity	1
21	Convective Film Coefficient	1
22	Electromagnetic Radiation Parameters	4

The type of the first independent variable is given in the following table:

TYPI	Variable Type
1	Temperature
2	Pressure
3	Relative humidity
4	Rate of Strain
5	Velocity
6	Acceleration
7	Time
8	Strain

#### 4.107 TABULAR DATA PROPERTY (FORM 11)

The default units used for this property shall follow the International System of Units (SI) practice for base units and derived units (IEEE76). Typical SI units are as follows:

Base Units	Unit	Symbol
Length	meter	m
Mass	kilogram	kg
Time	second	s
Electric Current	ampere	A
Thermodynamic Temperature	K	
Amount	mole	mol
Luminous Intensity	candela	cd
Plane Angle	radian	rad
Solid Angle	steradian	s r

Derived Units	Unit	Symbol	Formula	I
Force	Newton	N	$(\text{kg} \cdot \text{m}/\text{s})/\text{s}$	
Energy	Joule	J	$\text{N} \cdot \text{m}$	

**Young's Modulus (PTYPE = 1)** Young's Modulus relates stress to strain in materials. In the simple case:

$$\bar{\sigma} = E \bar{\epsilon},$$

where

- $\bar{\sigma}$  = Stress vector (row of elements  $\sigma_x, \sigma_y, \sigma_z$ ),
- $\bar{\epsilon}$  = Strain vector (row of elements  $\epsilon_x, \epsilon_y, \epsilon_z$ ), and
- $E$  = Young's Modulus (matrix of diagonal elements  $E_{xx}, E_{yy}, E_{zz}$ ).

The modulus is a vector with three principal values  $E_{xx}, E_{yy}$ , and  $E_{zz}$ . ND = 3.

In matrix form:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{bmatrix} = \begin{bmatrix} E_{xx} & 0 & 0 \\ 0 & E_{yy} & 0 \\ 0 & 0 & E_{zz} \end{bmatrix}$$

**Poisson's Ratio (PTYPE = 2)** Poisson's ratio is the ratio of transverse strain in the j-direction when stressed in the i-direction, *i.e.*,

$$\nu_{ij} = \epsilon_j / \epsilon_i,$$

where

- $\nu$  = Poisson's Ratio,
- $\epsilon$  = Strain,
- $i$  = One orthogonal direction, and
- $j$  = Another orthogonal direction.

#### 4.107 TABULAR DATA PROPERTY (FORM 11)

The Poisson's Ratio is a vector consisting of matrix elements with the three principal values,  $\nu_{xy}$ ,  $\nu_{yz}$ ,  $\nu_{zx}$ . The off-diagonal matrix values are reciprocals of the principal values, *i.e.*,

$$\nu_{xy} = 1/\nu_{yx}.$$

ND = 3.

In matrix form, for an orthotropic material,

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \end{bmatrix} = \begin{bmatrix} 1/E_{xx} & -\nu_{yx}/E_{yy} & -\nu_{zx}/E_{zz} \\ -\nu_{xy}/E_{xx} & 1/E_{yy} & -\nu_{zy}/E_{zz} \\ -\nu_{xz}/E_{xx} & -\nu_{yz}/E_{yy} & 1/E_{zz} \end{bmatrix} \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{bmatrix}.$$

**Shear Modulus (PTYPE = 3)** Shear Modulus is the ratio of shear stress to shear strain.

$$G = \tau_s / \gamma,$$

where

- $G$  is the Shear Modulus,
- $\tau_s$  is the Shear Stress, and
- $\gamma$  is the Shear Strain.

The Shear Modulus is a vector with the three principal values  $G_{xy}$ ,  $G_{yz}$ , and  $G_{zx}$ . ND = 3.

In matrix form, for orthotropic materials,

$$\begin{bmatrix} \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} = \begin{bmatrix} 1/G_{xy} & 0 & 0 \\ 0 & 1/G_{yz} & 0 \\ 0 & 0 & 1/G_{zx} \end{bmatrix} \begin{bmatrix} \tau_{s_{xy}} \\ \tau_{s_{yz}} \\ \tau_{s_{zx}} \end{bmatrix}.$$

**Material Matrix (PTYPE = 4)** Material matrix defines the tensor qualities of the material.

For example:

$$\bar{\sigma} = [C] \bar{\epsilon},$$

where

- $\bar{\sigma}$  is the Stress Vector,
- $\bar{\epsilon}$  is the Strain Vector, and
- $[C]$  is the Material Matrix.

Because of symmetry, the elements  $C_{ji} = C_{ij}$ . Therefore, 21 elements define the material matrix:

$$\begin{array}{ccc} C_{11} & C_{24} & C_{55} \\ C_{12} & C_{34} & C_{16} \\ C_{22} & C_{44} & C_{26} \\ C_{13} & C_{15} & C_{36} \\ C_{23} & C_{25} & C_{46} \\ C_{33} & C_{35} & C_{56} \\ C_{14} & C_{45} & C_{66} \end{array}$$

ND = 21.

In Matrix form:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix}$$

**Mass Density (PTYPE = 5)** Mass Density is the mass per unit volume. ND = 1.

Mass Density =  $\rho$

**Thermal Expansion Coefficient (PTYPE = 6)** The Thermal Expansion Coefficient is a material property that computes the strain given a temperature differential, *i.e.*,

$$\epsilon = \alpha \Delta T$$

or

$$\alpha = \epsilon / \Delta T,$$

where

- $\epsilon$  = strain,
- $\alpha$  = thermal expansion coefficient, and
- $\Delta T$  = the temperature differential.

The Thermal Expansion Coefficient may be represented as a vector with three principal values:

$$\alpha_{xx}, \alpha_{yy}, \text{ and } \alpha_{zz}.$$

ND = 3.

**Composite Materials (PTYPES 7- 11)** Composite materials will be represented with linkages to the Tabular Data Property as described in [Figure 128](#). PTYPES required are:

PTYPE	Description
7	Laminate material stiffness matrix
8	Bending material stiffness matrix
9	Transverse shear material stiffness matrix
10	Bending coupling material stiffness matrix
11	Material Coordinate System

**Laminate Material Stiffness Matrix (PTYPE = 7)** The membrane material stiffness matrix defines anisotropic material properties for shell membrane action. For example:

$$\bar{f} = t[M]\bar{\epsilon}$$

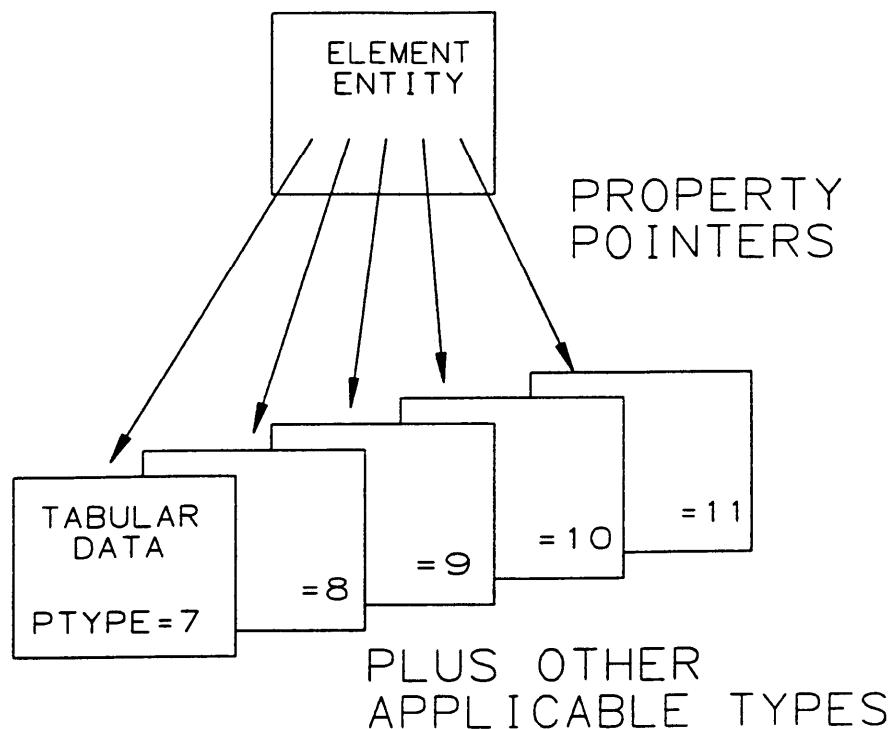


Figure 128. Relationship Between Properties Used to Represent a Composite Material

where

$\bar{f}$  = Forces per unit length (row of elements  $f_x, f_y, f_{xy}$ ),

$\bar{\epsilon}$  = Midplane strains (row of elements  $\epsilon_x, \epsilon_y, \epsilon_{xy}$ ),

$t$  = Shell thickness - see element property, and

$[M]$  = Membrane material stiffness matrix.

$\bar{f}$  and  $\bar{\epsilon}$  are defined in the shell material coordinate system, PTYPE = 11.

Because of symmetry, the elements  $M_{ji} = M_{ij}$ . Therefore, six elements define the membrane material stiffness matrix:

$M_{11}$   
 $M_{12}$   
 $M_{13}$   
 $M_{22}$   
 $M_{23}$   
 $M_{33}$

**ND = 6.**

The matrix  $[M]$  is a laminate material stiffness matrix which is calculated from lamina stress strain matrices  $[G]_n$ . One method for calculating  $[M]$  for a laminate containing  $m$  plies is:

$$[M_{ij}] = \frac{1}{t} \sum_{m=1}^m [G_{ij}]_n \Delta t_n,$$

#### 4.107 TABULAR DATA PROPERTY (FORM 11)

where

- $\Delta t_n$  is thickness of  $n^{\text{th}}$  ply,
- $t$  is total thickness of laminate, and
- $[G]_n$  is stress strain matrix for  $n^{\text{th}}$  ply of laminate, defined in the material coordinate system.

**Bending Material Stiffness Matrix (PTYPE = 8)** The bending material stiffness matrix defines the anisotropic material properties for shell bending. For example:

$$\bar{m} = \frac{1}{12}t^3 [B]\bar{\chi},$$

where

- $\bar{m}$  = Shell bending moments per unit length (row of elements  $M_x, M_y, M_{xy}$ ),
- $\bar{\chi}$  = Shell curvature (row of elements  $\chi_x, \chi_y, \chi_{xy}$ ),
- $t$  = Shell thickness (see element property), and
- $[B]$  = Bending material stiffness matrix.

$\bar{M}$  and  $\bar{\chi}$  are defined in shell material coordinate system, PTYPE= 11.

Because of symmetry, the elements  $B_{ij} = B_{ji}$ . Therefore, six elements define the bending stress strain matrix:

$$\begin{matrix} B_{11} \\ B_{12} \\ B_{13} \\ B_{22} \\ B_{23} \\ B_{33} \end{matrix}$$

ND = 6.

The matrix  $[B]$  is a laminate matrix for bending which is calculated from lamina matrices  $[G]_n$ . One method for calculating  $[B]$  for a laminate containing  $m$  plies is:

$$[B_{ij}] = \frac{1}{12}t^{-3} \sum_{n=1}^m (Z_n^2 [G_{ij}]_n$$

where

- $[G]_n$  is the stress strain matrix for the  $n^{\text{th}}$  ply of laminate,
- $\Delta t_n$  is thickness of  $n^{\text{th}}$  ply,
- $t$  is total thickness of laminate, and
- $Z_n$  is the normal distance from midplane of shell to the centroid of the ply.

**Transverse Shear Material Stiffness Matrix (PTYPE = 9)** The transverse shear material stiffness matrix defines anisotropic material properties for transverse shear flexibility in shell structure. For example:

$$\bar{V} = t_s [S]\bar{\gamma},$$

#### 4.107 TABULAR DATA PROPERTY (FORM 11)

where

$\bar{V}$  = Transverse shear force per unit length (row of elements  $V_x, V_y$ ),

$\bar{\gamma}$  = Transverse shear strains, dimensionless (row of elements  $\gamma_x, \gamma_y$ ),

$t_s$  = 5/6 of effective transverse shear thickness, and

$[S]$  = Transverse shear material stiffness matrix.

$\bar{V}$  and  $\bar{\gamma}$  are defined in the material coordinate system.

Because of symmetry, the elements  $S_{ij} = S_{ji}$ . Therefore, three elements define the transverse shear material stiffness matrix:

$$\begin{matrix} S_{11} \\ S_{12} \\ S_{22} \end{matrix}$$

ND = 3.

The matrix  $[S]$  is a laminate material stiffness matrix for transverse shear flexibility. If the matrix is not defined, deflections normal to the shell do not include contributions from transverse shear strain.

**Bending Coupling Material Stiffness Matrix (PTYPE = 10)** The membrane-bending coupling material stiffness matrix defines the anisotropic material properties for shell structure with the neutral axis for bending offset from the midplane of the shell. For example:

$$\bar{f} = t^2 [C] \bar{\chi}$$

and

$$\bar{m} = t^2 [C]^T \bar{\epsilon},$$

where

$\bar{f}$  = Forces per unit length (row of elements  $f_x, f_y, f_{xy}$ ),

$\bar{m}$  = Bending moments per unit length (row of elements  $m_x, m_y, m_{xy}$ ),

$\bar{\chi}$  = Curvature (measurement of bending strain, (meter)<sup>-1</sup>) (row of elements  $\chi_x, \chi_y, \chi_{xy}$ ),

$\bar{\epsilon}$  = Midplane strain (row of elements  $\epsilon_x, \epsilon_y, \epsilon_{xy}$ ), and

$t$  = Shell thickness.

$\bar{f}, \bar{m}, \bar{\chi}$ , and  $\bar{\epsilon}$  are defined in the shell material coordinate system.

Because of symmetry, the elements  $C_{ij} = C_{ji}$ . Therefore, six elements define the membrane bending coupling material matrix:

$$\begin{matrix} C_{11} \\ C_{12} \\ C_{13} \\ C_{22} \\ C_{23} \\ C_{33} \end{matrix}$$

## 4.107 TABULAR DATA PROPERTY (FORM 11)

ND = 6.

The matrix [C] is a laminate matrix for membrane-bending coupling, which is calculated from lamina stress strain matrices [G]<sub>n</sub>. One method for calculating [C] for a laminate containing  $m$  plies is:

$$[C_{ij}] = t^{-2} \sum_{n=1}^m (Z_n [G_{ij}]_n \Delta t_n),$$

where

- [G]<sub>n</sub> = stress strain matrix for  $n^{th}$  ply,  
defined in shell material coordinate system,
- $\Delta t_n$  = thickness of  $n^{th}$  ply,
- $Z_n$  = the normal distance from midplane  
of the shell to centroid of  $n^{th}$  ply, and
- $t$  = thickness of shell.

**Material Coordinate System (PTYPE = 11)** The orientation of the element material coordinate system is specified by a set of direction cosines defining a vector  $\bar{D}$ . The use of the vector  $\bar{D}$  depends upon the element type.

For Element Topology Types 1 and 33 of the Finite Element Entity ([Type 136](#)), [Figure 129](#) illustrates the use of the vector  $\bar{D}$  to define the element material coordinate system. For Topology [Type 33](#), the vector  $\bar{D}$  is defined by the Reference Node 3.

The cosines for vector  $\bar{D}$  are translated to the location of the shear center offset to establish the reference planes for material property definition (vector  $\bar{DT}$ ).

For Element Topology [Types 2](#) through [26](#) of the Finite Element Entity, the following paragraphs discuss the use of vector  $\bar{D}$  to define the material coordinate system.

The projection of  $\bar{D}$  on the plane of the element face (F1) (outward normal  $\bar{N}$ ) defines a vector in the X direction of the material coordinate system:

$$\bar{X} = \bar{N} \times \bar{D} \times \bar{N},$$

where

- $\bar{N}$  = positive outward normal of the element face (F1) and is defined  
by nodal connection of the element, i.e.,  $\bar{N} = \bar{S}_1 \times \bar{S}_2$ ,
- $\bar{S}_1$  = vector from first to second corner of element face (F1), and
- $\bar{S}_2$  = vector from second to third corner of element face (F1).

Three direction cosines are required to define the vector  $\bar{D}$  in the global coordinate system:

$$\begin{matrix} D_1 \\ D_2 \\ D_3 \end{matrix}$$

ND = 3.

## 4.107 TABULAR DATA PROPERTY (FORM 11)

The vectors  $\bar{D}$  and  $\bar{N}$  define the element material X and Z axes, respectively. The internal load and strain sign conventions are described to ensure consistent definition of Material [Types 7-10](#). See [Figure 130](#).

### Internal Load Relationships:

$$\begin{Bmatrix} f_x \\ f_y \\ f_{xy} \end{Bmatrix} = \bar{f} \text{ forces per unit length}$$

$$\begin{Bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_{xy} \end{Bmatrix} = \epsilon \text{ midplane strains}$$

$$\begin{Bmatrix} M_x \\ M_y \\ M_{xy} \end{Bmatrix} = \bar{M} \text{ moments per unit length}$$

$$\begin{Bmatrix} \chi_x \\ \chi_y \\ \chi_{xy} \end{Bmatrix} = \chi \text{ bending curvatures}$$

$$\begin{Bmatrix} V_x \\ V_y \end{Bmatrix} = \bar{V} \text{ transverse shear forces per unit length}$$

$$\begin{Bmatrix} \gamma_x \\ \gamma_y \end{Bmatrix} = \gamma \text{ transverse shear strains}$$

### Strain Displacement Relationships:

$$\epsilon_x = \frac{\partial u}{\partial x}$$

$$\epsilon_y = \frac{\partial v}{\partial y}$$

$$\epsilon_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$$

$$\chi_x = \frac{\partial^2 u}{\partial x^2}$$

$$\chi_y = \frac{\partial^2 v}{\partial y^2}$$

$$\gamma_x \approx \frac{\partial w}{\partial x}$$

$$\gamma_y \approx \frac{\partial w}{\partial y}$$

where

$x, y, z$  are material coordinate system axes, and

$u, v, w$  are displacements of a point in the material coordinate system.

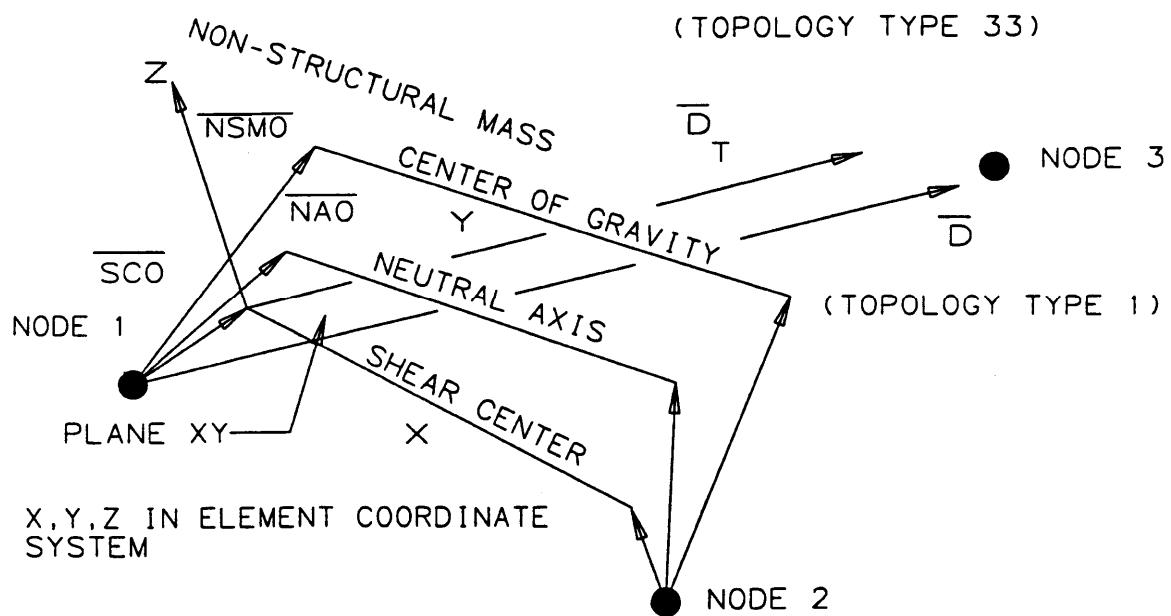


Figure 129. Use of the Vector  $\bar{D}$  to Define the Element Material Coordinate System

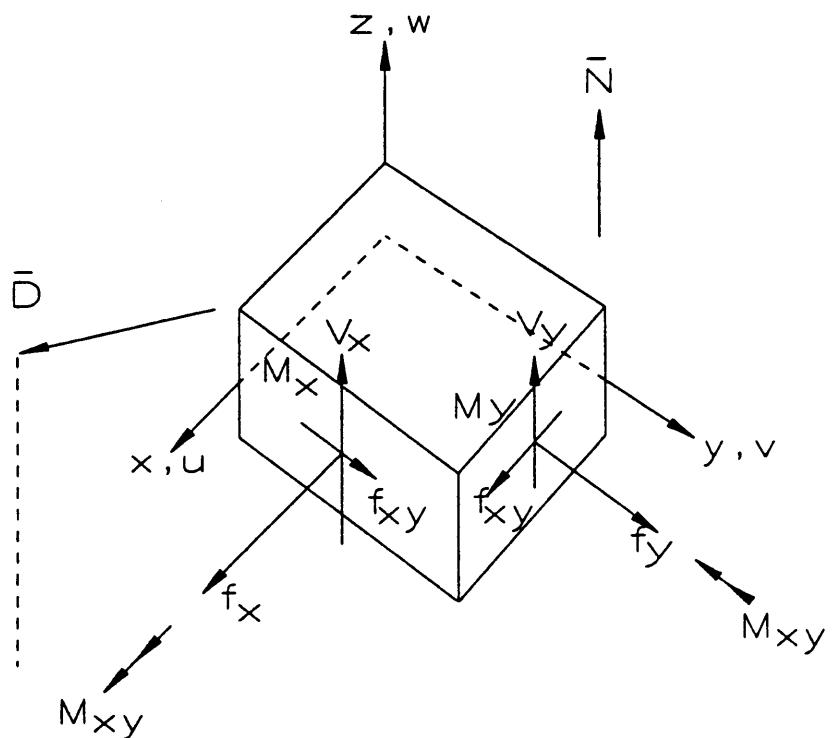


Figure 130. Internal Load and Strain Sign Convention

#### 4.107 TABULAR DATA PROPERTY (FORM 11)

**Nodal Loads and Constraints Data (PTYPE = 12)** The nodal load and constraint data shall be stored in following manner:

PTYPE = 12

ND = Number of degrees of freedom.

For example, if the load vector has X, Y, Z,  $M_x$ ,  $M_y$ , and  $M_z$  components, ND=6. (Note:  $M_x$ ,  $M_y$ , and  $M_z$  refer to moments. ) If the load vector has X and Y components, ND=2. If the load vector has only a Z-component, ND=3. In other words, the X-component is the 1st degree of freedom, the Y-component is the 2nd degree of freedom, and the Z-component is the 3rd degree of freedom. Other components are treated in a similar manner starting with the 4th degree of freedom for the rotation X component.

The constraint vector has X, Y, Z,  $M_x$ ,  $M_y$ , and  $M_z$  constraints. These constraints are represented by 0 (= No Constraint) and 1 (= Constraint). For constraints, ND = 6; otherwise, ND = the number of degrees of freedom.

**Sectional Properties for Beam Elements (PTYPE = 13)** Sectional properties for beam elements define the structural characteristics of the beam. These properties are:

Property Name	Units	Description
AREA	$M^2$	Area of section
IX	$M^4$	Area moment of inertia about the element x-axis
IY	$M^4$	Area moment of inertia about the element y-axis
IXY	$M^4$	Product of inertia
J	$M^4$	Torsional stiffness parameter
SRXY	Unitless	Shear stiffness ratio
SRXZ	Unitless	Shear stiffness ratio
WC	$M^6$	Warping coefficient.

If the properties are the same at both ends of the beam, ND=8; otherwise ND=16. If ND=16, two sets of section properties shall be specified. They shall be stated in order of the topology set grid number scheme.

**Beam End Releases (PTYPE = 14)** Beam end releases specify whether the ends of the beam are constrained or free to move. If free to move then both translation and rotational freedoms of the end are considered.

Property Name	Description
For each end	X X direction translation freedom/constraint
	Y Y direction translation freedom/constraint
	Z Z direction translation freedom/constraint
	MX X direction rotational freedom/constraint
	MY Y direction rotational freedom/constraint
	MZ Z direction rotational freedom/constraint

The value for each property X, Y, Z,  $M_x$ ,  $M_y$ , and  $M_z$  is a 0 (= unconstrained), +1 (= constrained to the global coordinate system), or -1 (= constrained to the element coordinate system).

The beam release shall be specified at both ends; therefore, ND=12.

The beam ends shall be defined by the topology set grid number scheme.

**Offsets (PTYPE = 15)** Offsets are global x,y,z values used to define the location of the shear center axis, neutral axis, and non-structural center of mass relative to the element end nodes. [Figure 129](#) shows the Shear Center Offset, SCO; the Neutral Axis Offset, NAO; and the Non-Structural Mass Offset, NSMO.

These offsets are vectors in the global coordinate system (model space) relative to the end of the beam.

Property Name	Description
For each end	SCOX Shear Center Offset in global x direction
	SCOY Shear Center Offset in global y direction
	SCOZ Shear Center Offset in global z direction
	NAOX Neutral Axis Offset in global x direction
	NAOY Neutral Axis Offset in global y direction
	NAOZ Neutral Axis Offset in global z direction
	NSMOX Non – Structural Mass Offset in global x direction
	NSMOY Non – Structural Mass Offset in global y direction
	NSMOZ Non – Structural Mass Offset in global z direction

## 4.107 TABULAR DATA PROPERTY

ND=9, or ND=18, depending on whether or not both ends must be specified. They in order of the topology set grid number scheme.

**Stress Recovery Information (PTYPE = 16)** Stress Recovery Information is used to define up to four offset points at each beam end at which stress levels will be recovered from the finite element analysis program.

These offset points are described as global x, y, z offsets from each end node. All offset points are in a plane which is normal to the beam element axis. These points occur in pairs from one end of the beam to the other.

Property Name	Description
END1 first pair offsets	$\left\{ \begin{array}{l} \text{SRI1X1 Stress Recovery Information beam end 1 x - direction pair 1} \\ \text{SRI1Y1 Stress Recovery Information beam end 1 y - direction pair 1} \\ \text{SRI1Z1 Stress Recovery Information beam end 1 z - direction pair 1} \end{array} \right.$
END2 first pair offsets	$\left\{ \begin{array}{l} \text{SRI12X1 Stress Recovery Information beam end 2 x — direction pair 1} \\ \text{SRI2Y1 Stress Recovery Information beam end 2 y — direction pair 1} \\ \text{SRI2Z1 Stress Recovery Information beam end 2 z — direction pair 1} \\ \text{(for the first pair of offsets)} \end{array} \right.$
⋮	⋮
END1 fourth pair offsets	$\left\{ \begin{array}{l} \text{SRI1X4 Stress Recovery Information beam end 1 x — direction pair 4} \\ \text{SRI1Y4 Stress Recovery Information beam end 1 y — direction pair 4} \\ \text{SRI1Z4 Stress Recovery Information beam end 1 z — direction pair 4} \end{array} \right.$
END2 fourth pair offsets	$\left\{ \begin{array}{l} \text{SRI2X4 Stress Recovery Information beam end 2 x — direction pair 4} \\ \text{SRI2Y4 Stress Recovery Information beam end 2 y — direction pair 4} \\ \text{SRI2Z4 Stress Recovery Information beam} \\ \text{(for the fourth pair of offsets)} \end{array} \right.$

**Element Thickness (PTYPE = 17)** Element Thickness defines the net section thickness for a homogeneous element or individual plate thickness for a laminate or sandwich plate. ND=1 or n and is defined as follows:

ND	Description
1	Scalar thickness of element
n	Thickness of the $n^{th}$ plate of a sandwich or laminate

#### 4.107 TABULAR DATA PROPERTY (FORM 11)

The thickness is ordered from 1 to n.

The thicknesses are measured in the positive z direction in order of increasing z in local element coordinate system.

<b>Property Name</b>	<b>Description</b>
T 1	Thickness for homogeneous plate or the thickness for the first lamina of the plate
⋮	⋮
Tn	Thickness for the nth lamina of the plate

**Non-Structural Mass (PTYPE = 18)** Non-Structural Mass is defined as the mass not accounted for in volume and density information for the structural elements. ND=1.

<b>Property Name</b>	<b>Description</b>
NSM	Mass per unit length or Mass per unit area or Mass per unit volume

The description depends on the type of element.

**Thermal Conductivity (PTYPE = 19)** Thermal Conductivity relates heat flow across a surface as a function of temperature. The heat balance equation shows this relationship:

$$\frac{\partial}{\partial x} \left[ k_x \frac{\partial T}{\partial x} \right] + \frac{\partial}{\partial y} \left[ k_y \frac{\partial T}{\partial y} \right] + \frac{\partial}{\partial z} \left[ k_z \frac{\partial T}{\partial z} \right] = \rho C_p \frac{\partial T}{\partial t} - \dot{Q}_I,$$

where

- $k_n$  = Thermal conductivity coefficient ( $n=x, y, z$ ),
- $C_p$  = Heat capacity at constant pressure,
- $\rho$  = Material density,
- $T$  = Temperature,
- $t$  = Time, and
- $\dot{Q}_I$  = Rate that energy is converted to internal heat.

x, y, and z are defined in the local element coordinate system.

If we consider  $k$  (thermal conductivity coefficient) as independent of direction,  $k$  can be represented as constant in the x, y, z directions: i.e.,

$$\left[ k_n \frac{\partial T}{\partial n} \right] = k_n \frac{\partial^2 T}{\partial n^2},$$

where  $n$  is  $x$ ,  $y$ , or  $z$ .

#### 4.107 TABULAR DATA PROPERTY (FORM 11)

Therefore, the thermal conductivity is a vector with three principal values,  $k_x$ ,  $k_y$ , and  $k_z$ . This implies that ND (Number of Dependent Variables) is equal to three. In matrix form, the heat balance equation is:

$$[k_x \ k_y \ k_z] \begin{bmatrix} \frac{\partial^2 T}{\partial x^2} \\ \frac{\partial^2 T}{\partial y^2} \\ \frac{\partial^2 T}{\partial z^2} \end{bmatrix} = \rho C_p \frac{\partial T}{\partial t} - \dot{Q}_I.$$

Now assume that there are no internal heat sources  $\dot{Q}_I$ , and the heat flow is steady state ( $\partial T / \partial t = 0$ ). Then, integrating the above equation in one dimension yields

$$\dot{Q}_x = k_x A \frac{\partial T}{\partial x},$$

where  $\dot{Q}_x$  is the heat flux in the x direction across a surface of area  $A$  normal to the x direction. Likewise, solutions can be found in the y and z directions.

Property Name	Description
KX	Thermal Conductivity coefficient x direction
KY	Thermal Conductivity coefficient y direction
KZ	Thermal Conductivity coefficient z direction

**Heat Capacity (PTYPE = 20)** Heat Capacity is a material's ability to store heat. The heat balance equation shows the relationship of heat capacity to the spatial variation and time variation of temperature.

$$\frac{\partial}{\partial x} \left[ k_x \frac{\partial T}{\partial x} \right] + \frac{\partial}{\partial y} \left[ k_y \frac{\partial T}{\partial y} \right] + \frac{\partial}{\partial z} \left[ k_z \frac{\partial T}{\partial z} \right] = \rho C_p \frac{\partial T}{\partial t} - \dot{Q}_I,$$

where

- $k_n$  = Thermal conductivity coefficient where  $n = x, y, z$ ,
- $C_p$  = Heat capacity at constant pressure,
- $P$  = Material density,
- $T$  = Temperature,
- $t$  = Time, and
- $\dot{Q}_I$  = Rate that energy is converted to internal heat.

If we consider constant pressure, the heat capacity can be considered a constant. ND=1.

Property Name	Description
CP	Heat capacity at constant pressure

#### 4.107 TABULAR DATA PROPERTY (FORM 11)

**Convective Film Coefficient (PTYPE = 21)** Convective film coefficient relates to the amount of heat flux that is convected to adjacent materials at the interface boundary of a heat source.

$$\dot{Q} = -h_c A \Delta T,$$

where

$\dot{Q}$	= the heat flux,
$h_c$	= the convective film coefficient,
$A$	= the surface area through which the heat flows, and
$\Delta T$	= the temperature differential between the materials.

The convective film coefficient may be represented as a constant. ND=1.

Property Name	Description
HC	Convective Film Coefficient

**Electromagnetic Radiation Parameters (PTYPE = 22)** Properties for Absorptivity, Transmissivity, Reflectivity, and Emissivity are defined for structural elements using four values. ND=4.

Property Name	Description
A	Absorptivity Constant
T	Transmissivity Constant
R	Reflectivity Constant
E	Emissivity Constant

**Directory Entry**

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level #, ⇒	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **??*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 11	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see Sections 4.97 and 1.6.1).

**Parameter Data**

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of Property values
2	PTYPE	Integer	Property Type:
3	ND	Integer	Number of dependent variables
4	NI	Integer	Number of independent variables
5	TYPI(1)	Integer	Type of first independent variable: ⋮ ⋮
5+NI	TYPI(NI)	Integer	Type of the last independent variable
6+NI	NVALI(1)	Integer	Number of different values of the first independent variable ⋮ ⋮
6+2*NI	NVALI(NI)	Integer	Number of different values of the last independent variable
7+2*NI	VALI(1,1)	Real	First value of the first independent variable ⋮ ⋮ ⋮
	VALI(1, NVALI(1))	Real	Last value of the first independent variable
	⋮ ⋮ ⋮	⋮ ⋮ ⋮	Last value of the last independent variable
	VALD(1,1)	Real	Value of the first dependent variable at the first data point
	⋮ ⋮ ⋮	⋮ ⋮ ⋮	Value of the j-th dependent variable at the k-th data point
	⋮ ⋮ ⋮	⋮ ⋮ ⋮	Value of the last dependent variable at the last data point
N	VALD(ND, NVALI(NI))	Real	

Additional pointers as required (see Section 2.2.4.5.2).

## 4.107 TABULAR DATA PROPERTY (FORM 11)

### Examples of the use of the Tabular Data Form of the Property Entity:

Consider the representation of the mass density (PTYPE = 5) as a function of pressure. In this case, there is one independent variable. Suppose the density is known for two values of pressure. The Parameter Data Section contains:

Index	Name	Recorded Value
1	N P	9
2	PTYPE	5
3	ND	1
4	NI	1
5	TYPI	2
6	NVALI	2
7	VALI1	50
8	VALI2	25
9	VALD(1,1)	33
10	VALD(1,2)	46

as well as additional pointers as required ([see Section 2.2.4.5.2](#)).

Consider the representation of Young's modulus (PTYPE = 1) for a linear, static, independent case. In this case, there is no independent variable. The Parameter Data Section contains:

Index	Name	Recorded Value
1	NP	6
2	PTYPE	1
3	ND	3
4	N I	0
5	E <sub>xx</sub>	30.0E6
6	E <sub>yy</sub>	30.0E6
7	E <sub>zz</sub>	30.0E6

as well as additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.108 EXTERNAL REFERENCE FILE LIST PROPERTY (FORM 12)

### 4.108 External Reference File List Property (Form 12)

The External Reference File List appears in a file which references definitions that reside in another file. It contains a list of the names of the files directly referenced by entities within this file. See [Section 3.6.4](#) and the External Reference Entity ([Type 416](#)) for more detail.

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	⇒	< n.a. >	< n.a. >	#, ⇒	< n.a. >	< n.a. >	< n.a. >	**?*****	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
406	< n.a. >	< n.a. >	#	12				#	D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N P	Integer	Number of List Entries
2	NAME (1)	String	First External Reference File Name
⋮	⋮	⋮	
1+NP	NAME(NP)	String	Last External Reference File Name

Additional pointers as required (see [Section 2.2.4.5.2](#)).

## 4.109 NOMINAL SIZE PROPERTY (FORM 13)

### 4.109 Nominal Size Property (Form 13)

The Nominal Size Property attaches a value, a name, and, optionally, a reference to an engineering standard to entities which require special dimensioning. The nominal size value is a real value in the units appropriate for the specified name. The name is a string data type, but the following names have pre-defined meanings:

Nominal Size Name	Pre-defined Meaning
3HAWG	American Wire Gauge
3HIPS	Iron Pipe Size
2HOD	Outside Diameter schedule, i.e., tubing.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level #, ⇒	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 13	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=2 or 3)
2	SZ	Real	Nominal size value
3	NM	String	Nominal size name
4	SP	String	Name of relevant engineering standard (optional)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.110 FLOW LINE SPECIFICATION PROPERTY (FORM 14)

### 4.110 Flow Line Specification Property (Form 14)

The Flow Line Specification Property attaches one or more text strings to entities being used to represent a flow line.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #, ⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 14	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6,1](#)).

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values
2	L(1)	String	Primary flow line specification name
3	L(2)	String	Modifier (optional)
⋮	⋮	⋮	⋮
1+NP	L(NP)	String	Modifier (optional)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.111 Name Property (Form 15)**

This property attaches a string which specifies a user-defined name. It can be used for any entity ECO630 that does not have a name explicitly specified in the parameter data for the entity.

**Directory Entry**

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #, ⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **??****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 15	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

**Parameter Data**

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	NP	Integer	Number of property values (NP=1)
2	NAME	String	Entity Name

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.112 DRAWING SIZE PROPERTY (FORM 16)

### 4.112 Drawing Size Property (Form 16)

This property specifies the size of the drawing in drawing units. The origin of the drawing is defined to be (0,0) in drawing space.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #, ⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 16	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=2)
2	XS	Real	X Size (Extent of Drawing along positive XD axis)
3	YS	Real	Y Size (Extent of Drawing along positive YD axis)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.113 DRAWING UNITS PROPERTY (FORM 17)

### 4.113 Drawing Units Property (Form 17)

This property specifies the drawing space units as outlined in the Drawing Entity ([Type 404](#)). The drawing units are given in the same form as the model space units in the Global Section (see [Section 2.2.4.3.15](#)).

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	⇒	< n.a. >	< n.a. >	#, ⇒	< n.a. >	< n.a. >	< n.a. >	***??****	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16)	(17)	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
406	< n.a. >	< n.a. >	#	17	Reserved	Reserved	#	D # + 1	

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97](#) and [1.6.1](#)).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=2)
2	FLAG	Integer	Units Flag
3	UNIT	String	Units Name

Additional pointers as required (see [Section 2.2.4.5.2](#)).

## 4.114 INTERCHARACTER SPACING PROPERTY (FORM 18)‡

### 4.114 Intercharacter Spacing Property (Form 18)‡

‡The Intercharacter Spacing Property Entity has not been tested. See Section 1.9.

ECO630

The Intercharacter Spacing Property specifies the gap between letters when fixed-pitch spacing is used. It is applicable to text generated by the General Note and Text Template Entities. The gap shall be calculated as a percentage of the text height.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #, ⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **??*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 18	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see Sections 4.97 and 1.6.1).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP = 1)
2	ISPACE	Real	Intercharacter Space in percent of text height (Range 0. to 100.)

Additional pointers as required (see Section 2.2.4.5.2).

## 4.115 LINE FONT PROPERTY (FORM 19)‡

### 4.115 Line Font Property (Form 19)‡

‡The Line Font Property Entity has not been tested. See Section 1.9.

ECO630

This property specifies a line font pattern from a pre-defined list rather than from Directory Entry Field 4 (either the default line font patterns, or those available by defining a repeating pattern using the Line Font Definition Entity (Type 304)). The list is given in [Table 15](#); illustrations of line font patterns are found in [Figure 131](#).

It is not intended that exact visual equivalence be preserved. The receiving system is to use similar but not necessarily identical patterns based on the pattern codes; the intent is to preserve the *functionality* implicit in the code. If the receiving system does not have a similar pattern, the postprocessor shall use the pattern specified by DE Field 4 of the entity pointing to this property.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level #, ⇒	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number ***?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 19	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=1)
2	LFPC	Integer	Line Font Pattern Code ( <a href="#">see Figure 131</a> )

Additional pointers as required ([see Section 2.2.4.5.2](#)).

Table 15. Line Font Property Patterns

<b>LFPC</b>	<b>Meaning</b>	<b>Authority</b>
12	Compressed Air Line	[ANSI72]
14	Duct & Air	[ANSI72]
16	Mech. Pipe & Air	[ANSI72]
18	Mech. Pipe Duct & Air	[ANSI72]
22	Gas Pipe Line	[ANSI72]
42	High-Pressure Steam	[ANSI79a]
44	High-Pressure Return	[ANSI79a]
46	Medium-Pressure Steam	[ANSI79a]
48	Medium-Pressure Return	[ANSI79a]
52	Feedwater Pump Discharge	[ANSI79a]
54	Condensate or Vacuum Pump Discharge	[ANSI79a]
152	Fence (on Street Line)	[ANSI72]
154	Fence (on Railway Property Line)	[ANSI72]
156	Rail Fence	[ANSI72]
162	Woven Wire Fence	[ANSI72]
164	Barbwire Fence	[ANSI72]
166	Picket Fence	[ANSI72]
172	Hedge Fence	[ANSI72]
174	Stone Fence	[ANSI72]
176	Snow Fence	[ANSI72]
178	Worm Fence	[ANSI72]
192	City	[ANSI72]
194	City Limit	[ANSI72]
198	Fire Limit	[ANSI72]
200	Coke Ovens	[ANSI72]
203	Soil, Waste, or Leader (Below Grade)	[ANSI79a]
206	Vent	[ANSI79a]
223	Cold Water	[ANSI79a]
227	Hot Water	[ANSI79a]
230	Hot Water Return	[ANSI79a]
232	Makeup Water	[ANSI79a]
237	Acid Waste	[ANSI79a]

**4.115 LINE FONT PROPERTY (FORM 19)‡**

Table 15. Line Font Property Patterns (continued)

LFPC	Meaning	Authority
239	Acid Vent	[ANSI79a]
240	Indirect Drain	[ANSI79a]
253	Fire Line	[ANSI79a]
270	Vacuum Cleaning	[ANSI79a]
330	Pneumatic Tubes/Tube Runs	[ANSI79a]
355	Low Pressure Steam Return	[ANSI79a]
360	Boiler Blow Off	[ANSI79a]
380	Air Relief Line	[ANSI79a]
385	Fuel Oil Return	[ANSI79a]
390	Fuel Oil Tank Vent	[ANSI79a]
395	Hot Water Heating Supply	[ANSI79a]
400	Hot Water Heating Return	[ANSI79a]
405	Refrigerant Liquid	[ANSI79a]
410	Refrigerant Discharge	[ANSI79a]
415	Humidification Line	[ANSI79a]
420	Drain	[ANSI79a]
425	Brine Supply	[ANSI79a]
430	Brine Return	[ANSI79a]
445	Branched Head Sprinkler	[ANSI79a]
485	Fence Intertrack	[ANSI79a]

#### 4.115 LINE FONT PROPERTY (FORM 19)‡

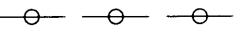
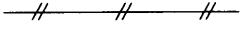
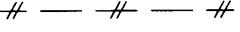
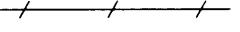
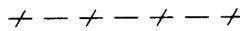
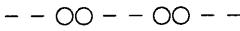
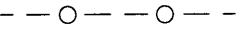
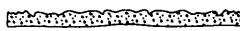
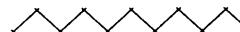
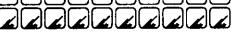
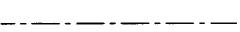
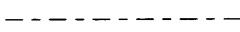
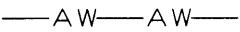
			
CODE 12	CODE 14	CODE 16	CODE 18
			
CODE 22	CODE 42	CODE 44	CODE 46
			
CODE 48	CODE 52	CODE 54	CODE 152
			
CODE 154	CODE 156	CODE 162	CODE 164
			
CODE 166	CODE 172	CODE 174	CODE 176
			
CODE 178	CODE 192	CODE 194	CODE 198
			
CODE 200	CODE 203	CODE 206	CODE 223
			
CODE 227	CODE 230	CODE 232	CODE 237

Figure 131. Illustrations of Line Font Patterns for Different Values of LFPC

**4.115 LINE FONT PROPERTY (FORM 19)‡**

---AV---AV---	— D — D —	— F — F —	— V — V —
CODE 239	CODE 240	CODE 253	CODE 270
— PN — PN —	— LPR —	— BD — BD —	— V — V —
CODE 330	CODE 355	CODE 360	CODE 380
— FOR —	— FOV —	— HW — HW —	— HWR —
CODE 385	CODE 390	CODE 395	CODE 400
— RL — RL —	— RD — RD —	— H — H —	— D — D —
CODE 405	CODE 410	CODE 415	CODE 420
— B — B —	— BR — BR —	— O — O —	— — — —
CODE 425	CODE 430	CODE 445	CODE 485

Figure 131. Illustrations of Line Font Patterns for Different Values of LFPC (Continued)

## 4.116 HIGHLIGHT PROPERTY (FORM 20)‡

### 4.116 Highlight Property (Form 20)‡

‡The Highlight Property Entity has not been tested. See Section 1.9.

ECO630

The Highlight Property attaches information that an entity shall be displayed in some system-dependent manner, as it is in GKS (see [ANSI85, IS07942]), to draw attention to the display of an entity. Blinking or increasing intensity are two possible methods of accomplishing this.

Hierarchical application of the Highlight Property shall be the same as is done for Blank Status. For application of hierarchy, see Section 2.2.4.4.9.4.

ECO630

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level #, ⇒	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 20	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see Sections 4.97 and 1.6.1).

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=1)
2	HIGHLIGHT	Integer	Highlight Flag: 0 = entity is not highlighted (default) 1 = entity is highlighted

Additional pointers as required (see Section 2.2.4.5.2).

**4.117 Pick Property (Form 21)‡**

‡The Pick Property Entity has not been tested. See Section 1.9.

ECO630

The Pick Property attaches information that an entity may be picked by whatever pick device is used in the receiving system. See [ANSI85, ISO7942] for a discussion of picking in the context of the Graphical Kernel System (GKS).

ECO630

Hierarchical application of the Pick Property shall be the same as is done for Blank Status. For application of hierarchy, see Section 2.2.4.9.4.

ECO630

**Directory Entry**

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level #, ⇒	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **??*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 21	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see Sections 4.97 and 1.6.1).

**Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=1)
2	PICK	Integer	Pick flag: 0 = entity is pickable (default) 1 = entity is not pickable

Additional pointers as required (see Section 2.2.4.5.2).

## 4.118 UNIFORM RECTANGULAR GRID PROPERTY (FORM 22)‡

### 4.118 Uniform Rectangular Grid Property (Form 22)‡

‡The Uniform Rectangular Grid Property Entity has not been tested. See Section 1.9.

ECO630

This property specifies sufficient information for the creation of a uniform rectangular grid within a drawing. It shall be attached to the Drawing Entity ([Type 404](#)). ECO630

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #,⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 22	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

#### Parameter Data

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	NP	Integer	Number of property values (NP = 9)
2	FFLAG	Integer	Finite/infinite grid flag: 0 = infinite, 1 = finite
3	LFLAG	Integer	Line/point grid flag: 0 = points, 1 = lines
4	WFLAG	Integer	Weighted/unweighted grid flag (Weighting means the nearest grid point will be selected by screen position indication by cursor, light pen or other such means.): 0 = weighted, 1 = unweighted
5	PX	Real	X coordinate of a point on the grid in drawing coordinates. If the grid is finite, this point shall be the lower left corner of the grid. If the grid is infinite, this point is an arbitrary point on the grid.
6	PY	Real	Y coordinate of a point on the grid in drawing coordinates. If the grid is finite, this point shall be the lower left corner of the grid. If the grid is infinite, this point is an arbitrary point on the grid.
7	DX	Real	Grid spacing in X direction in drawing coordinates
8	DY	Real	Grid spacing in Y direction in drawing coordinates
9	NX	Integer	Number of points/lines in X direction (ignored if grid is infinite)
10	NY	Integer	Number of points/lines in Y direction (ignored if grid is infinite)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.119 ASSOCIATIVITY GROUP TYPE PROPERTY (FORM 23)‡

### 4.119   Associativity Group Type Property (Form 23)‡

‡The Associativity Group Type Property Entity has not been tested. See Section 1.9.

ECO630

The Associativity Group Type Property is used to assign an unambiguous identification to a Group ECO630 Associativity. This allows for the automated processing of the Unordered Group with Back Pointers Associativity Entity (Type 402, Form 1), the Unordered Group without Back Pointers Associativity Entity (Type 402, Form 7), the Ordered Group with Back Pointers Associativity Entity (Type 402, Form 14), and the Ordered Group without Back Pointers Associativity Entity (Type 402, Form 15). This property shall be attached only to these four associativity types. It includes a TYPE and a NAME.

The following definitions and abbreviations are used in the entity description.

**TYPE.** The Type field is an enumerated list, specifying a particular associativity type.

Value	Designated Type
1	Insertion Sequence
2	Functional Group
3	Work Cell
4	Fiducial
5	Drill Path
6	Profile Routing Sequence
7	Component Trimming Sequence
8-5000	other associativity types
5001-9999	implementor-defined types

ECO630

**NAME.** The Name field further identifies the associativity. The Name field is specified by native CAD/CAM system properties, by the user, or by other means.

One example of the usage of the Associativity Group Type Property is to group electronic components for proper insertion sequence. In this example, the entities to be inserted are grouped with the Ordered Group without Back Pointers Associativity Entity (Type 402, Form 15) and the Associativity Group Type Property is attached to the associativity. The Type field contains the value 1, indicating the associativity is an insertion sequence. The Name field contains the string "DIPS" (or other meaningful, user-specified name), distinguishing it from other insertion sequences (e.g., "RESISTORS" ).

In some cases (e.g., Drill Path), it may be necessary to specify an overall group of smaller groups. For example, if each Drill Path is for a unique drill size, the sequence of individual Drill Paths may be specified. In these cases, one of the group associativities (as appropriate to the application) shall be used as a parent (Subordinate Entity Switch = 00) to the individual (child) group associativities (Subordinate Entity Switch = 02). The parent associativity shall have an Associativity Group Type Property attached which specifies the same type of associativity as the child associativities.

## 4.119 ASSOCIATIVITY GROUP TYPE PROPERTY (FORM 23)‡

### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #,⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number ***?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 23	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

### Parameter Data

Index	Name	Type	Description
1	NP	Integer	Specifies the number parameter data fields (NP=2)
2	TYPE	Integer	Specifies the type of the attached associativity
3	NAME	String	Uniquely identifies a particular instance of an associativity of type TYPE

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## **4.120 LEVEL TO LEP LAYER MAP PROPERTY (FORM 24)‡**

### **4.120 Level to LEP Layer Map Property (Form 24)‡**

‡The Level to LEP Layer Map Property Entity has not been tested. See Section 1.9.

ECO630

The Level to LEP Layer Map property is used to correlate an exchange file level number with its corresponding native level identifier, physical LEP layer number, and predefine functional level identification. Therefore, the postprocessor of the exchange file can interpret the individual entity level number in terms of the physical LEP layer to which it maps. Furthermore, the postprocessor can determine what the functional use of the level was in the native system by analyzing the predefine functional level identification. This property shall be attached to the entity defining the LEP or, if no such entity exists, the property shall stand alone in the file.

In order to unambiguously represent what the intended functionality of the level was in the native system, the functional level identification shall be selected from a predefine list. There is no specific way to determine which side of a LEP is the top or bottom; it is an arbitrary decision. However, it is necessary to set a baseline orientation from which many other functions will be associated. Through the assignment of the functional level identifiers, the top and bottom side of the LEP becomes established. All consequent functions that require identifying the top and/or bottom of the LEP shall utilize this method of identification. The following list represents the current set of keywords.

If the level identification keyword is followed by the string (T#/B), it specifies that the actual level identification can be any of (level, level\_T, level\_# or level\_B).

**level** Represents data on a generic level. (A generic level attribute specifies that the base entity is associated with one or more levels based on a set of corresponding specific levels.)

**level\_T** Represents data on a specific level that maps into the top LEP layer.

**level\_#** Represents data on a specific level that maps into an internal LEP layer (where # is equal to the internal physical layer number 2,3,4,5, ..., etc.)

**level\_B** Represents data on a specific level that maps into the bottom LEP layer.

The following list represents the current predefine list of functional level names. The names shall be case insensitive.

ECO655

## 4.120 LEVEL TO LEP LAYER MAP PROPERTY (FORM 24)‡

<b>Level Identification</b>	<b>Description</b>	ECO630
Annotation	General comment text and graphics.	
Bond_Pad (T/#/B)	Component bonding pad geometry.	
Breakout (T/#/B)	Component breakout leads.	
Chip_Pad (T/#/B)	Component chip pad.	
Component_Outline (T/#/B)	Component boundary outlines.	
Component_Placement (T/#/B)	Component placement instances.	
Crossover (T/#/B)	Crossover conductor data.	
Deposition_Components (T/#/B)	Deposition component instances.	
Dielectric (T/#/B)	Dielectric crossover geometry.	
Drilled_Holes	Drilled hole geometry.	
Errors	Error data.	
Glue_Mask (T/#/B)	Glue mask outlines.	
Ground (T/#/B)	Conductive ground planes.	
Hole_Fill (T/#/B)	Conductive fill for holes.	
Laser-Trim-Path (T/#/B)	Laser trim paths.	
Pad (T/#/B)	Component pad geometry.	
Panel_Outline	Panel Outline.	
Pin_ID (T/#/B)	Component pin identification text.	
Pin_Placement (T/#/B)	Component pin instances.	
Placement_Keepin	Component placement keepin outlines.	
Placement_Keepout	Component placement keepout outlines.	
Power (T/#/B)	Conductive power plane geometry.	
PRD_ID	Primary Reference Designator text.	
Routing_Keepin	Routing keep-in outlines.	
Routing_Keepout	Routing keep-out outlines.	
Sheet_Dielectric (T/#/B)	Sheet dielectric data.	
Signal (T/#/B)	Signal routing geometry.	
Signal_Guide	Signal guide wires.	
Signal_ID (T/#/B)	Signal identification text.	
Silkscreen (T/#/B)	Silkscreen data.	
Solder_Mask (T/#/B)	Solder mask outlines.	
Solder_Paste-Mask (T/#/B)	Solder paste mask outlines.	
Substrate_Outline	Substrate data.	
Thermal_Outline (T/#/B)	Component thermal outlines.	
Trace_Keepin	Trace keep-in outlines.	
Trace_Keepout	Trace keep-out outlines.	
Undefined	Undefined data. Indicates that the functionality of exchange file numbers is not currently defined within the set of level identifications in the native system.	
Unplaced_Components	Unplaced components.	
Via_Keepin	Via keep-in outlines.	
Via_Keepout	Via keep-out outlines.	
Via_Placement	Via instances.	
Wire-Bond (T/#/B)	Wire bonds.	

**Note:** If a level number is found in an entity subordinate to the Network Subfigure Definition that is used to define the LEP, and the level number is not listed in the Level to LEP Layer Map property, it is assumed that the level does not correspond to a physical layer of the LEP and is intended to be a general annotation of the model. However, for levels that do not correspond

## 4.120 LEVEL TO LEP LAYER MAP PROPERTY (FORM 24)‡

to a physical layer, they shall be entered in this property, specified with a physical layer equal to zero-and a functional identification equal to UNDEFINED.

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	⇒	< n.a. >	< n.a. >	#, ⇒	< n.a. >	< n.a. >	< n.a. >	***?*****	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16)	(17)	(18)	(19)	(20) Sequence Number
406	< n.a. >	< n.a. >	#	24	Reserved	Reserved	Entity Label	#	D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97 and 1.6.1](#)).

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>	
1	NP	Integer	Number of property values	
2	NLD	Integer	Number of level to layer definitions	
3	IL(1)	Integer	Exchange file level number for the first level definition	
4	NLID(1)	String	Identification that the sending system used to identify the native level that was mapped to the first exchange file level number	
5	PLN(1)	Integer	Physical layer number to which the first level number applies. If the level does not apply to data that maps to a physical layer of the LEP, this field shall be set to zero	ECO630
6	FLN(1)	String	Exchange file level identification for the first level number	
7	IL(2)	Integer	Exchange file level number for the second level definition	
:	:	:		
2+4*NLD	FLN(NLD)	String	Exchange file level identification for the last level number	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.121 LEP ARTWORK STACKUP PROPERTY (FORM 25)‡

### 4.121 LEP Artwork Stackup Property (Form 25)‡

‡The LEP Artwork Stackup Property Entity has not been tested. See Section 1.9.

ECO630

The LEP Artwork Stackup Property is used to communicate which exchange file levels are to be combined in order to create the artwork for a printed wire board (or other LEP).

This property shall be attached to the entity defining the LEP or, if no such entity exists, the ECO630 property shall stand alone in the file.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #, ⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 25	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

Note: The Level shall be ignored if this property is subordinate (see Sections 4.97 and 1.6.1).

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values
2	ID	String	Artwork stackup identification
3	NV	Integer	Number of level number values
4	L(1)	Integer	First level number
:	:	:	
3+NV	L(NV)	Integer	Last level number

Additional pointers as required (see Section 2.2.4.5.2).

## **4.122 LEP DRILLED HOLE PROPERTY (FORM 26)‡**

### **4.122 LEP Drilled Hole Property (Form 26)‡**

‡The LEP Drilled Hole Property Entity has not been tested. [See Section 1.9.](#)

ECO630

The LEP Drilled Hole Property is used to identify an entity that locates a drilled hole and to specify ECO630 the characteristics of the drilled hole.

The DE attribute Level Number is used to specify which physical LEP layers the drilled hole pierces. The method used to convey this information is as follows:

1. Create a Definition Levels Property ([Type 406, Form 1](#)).
2. Include a level number for each physical LEP layer that the drilled hole is to pierce.
3. Reference the Definition Levels Property from the DE Level attribute of the LEP Drilled Hole ECO630 Property through a negated pointer.

The values included in the Definition Levels Property are exchange file level numbers (DE field 5). ECO630 In order to determine the actual physical LEP layers, the postprocessor shall refer to the physical layer number in the Level to LEP Layer Map Property ([Type 406, Form 24](#)).

The LEP Drilled Hole Property shall be attached to the following base entities:

- Connect Point Entity ([Type 132](#)), when the drilled hole is used to define a component thru-pin.
- Point Entity ([Type 116](#)), when the drilled hole is used to define a via, mounting, or tooling hole .

## 4.122 LEP DRILLED HOLE PROPERTY (FORM 26)‡

### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level ⇒	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number ***?*****	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 26	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** The Level shall be ignored if this property is subordinate (see [Sections 4.97](#) and [1.6.1](#)).

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=3)
2	DDS	Real	Drill diameter size
3	FDS	Real	Finish diameter size
4	FC	Integer	Function code for the drilled hole: 1 = Nonplated hole for general assembly purposes 2 = Plated hole for general assembly purposes 3 = Nonplated tooling hole 4 = Plated tooling hole 5 = Plated hole for component pins and vias 5001–9999 = Implement or-defined hole types

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.123 GENERIC DATA PROPERTY (FORM 27)‡

### 4.123 Generic Data Property (Form 27)‡

‡The Generic Data Property Entity has not been tested. See Section 1.9.

ECO630

The Generic Data Property is used to communicate information which is defined by the system operator while creating the model. The information is system-specific and does not map into one of the pre-defined properties or associativities.

Properties and property values can be defined by multiple instances of this property. An instance of ECO630 this property shall have its Subordinate entity switch set to Physically Dependent; it is dependent upon either a single entity or a group of geometric entities.

In cases where the system cannot process operator-defined properties, these entities may either be ECO630 ignored or be inserted as text at some logical location.

Definitions. The following definitions and abbreviations are used in the entity description.

**Property Name (NAME).** The NAME field is used to identify the property. The Name field is ECO630 specified by native CAD/CAM system properties, the user, or other means.

**Property Type (TYP).** The TYP field is an enumerated list, specifying a particular property type. The list of Type field values may be extended by modification of the Specification.

Value	Property Type
0	No value
1	Integer
2	Real
3	Character string
4	Pointer
5	Not used
6	Logical

ECO630

**Property Value (VAL).** Each VAL field contains a property value whose type is specified by the ECO630 associated Type field.

## 4.123 GENERIC DATA PROPERTY (FORM 27)‡

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**0102**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16)	(17)	(18)	(19)	(20) Sequence Number
406	< n.a. >	< n.a. >	#	27	Reserved	Reserved	Entity Label	#	D # + 1

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values
2	NAME	String	Property name
3	NV	Integer	Number of TYPE/VALUE pairs
4	TYP(1)	Integer	First property value data type
5	VAL(1)	Variable	First property value
:	:	:	
2+2*NV	TYP(NV)	Integer	Last property value data type
3+2*NV	VAL(NV)	Variable	Last property value

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.124 Dimension Units Property (Form 28)‡**

‡The Dimension Units Property Entity has not been tested. See Section 1.9.

ECO630

The Dimension Units Property describes the units and formatting details of the nominal value of ECO630 a dimension. One or two properties may be associated with the same dimension, depending on whether single or dual dimensioning is being used.

The Unit Indicator (UI) parameter defines the units to be used for calculating and displaying this dimension value. The following table defines the available units:

<b>Value</b>	<b>Meaning</b>
0	Use units from Global Section
1-11	See <a href="#">section 2.2.4.3.14</a> for meaning
100	Degrees
101	Degrees/minutes
102	Degrees/minutes/seconds
103	Radians
104	Grads
105	Feet/inches
106	Key-in text

ECO630

The CHRSET font characteristic parameter is used in conjunction with USTRING to allow specification of font characteristic (FC) with special symbols (e.g., the degree symbol). (See General Note Entity ([Type 212](#)).)

The USTRING shall be appended to the numeric value of the dimension to form the value displayed. For dimensions in which multiple numeric values are generated, (e.g., degrees/minutes/seconds), the USTRING consists of n subparts separated by the character "/" (slash). For example, USTRING could be 3H' /" for distances in feet and inches.

A single instance of this property may be pointed to by several dimensions.

**Directory Entry**

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**0202**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
406	< n.a. >	< n.a. >	#	28				#	D # + 1

**Parameter Data**

Index	Name	Type	Description
1	NP	Integer	Number of property values (NP=6)
2	SPOS	Integer	Position of secondary dimension with respect to primary dimension 0 = This is main text 1 = Secondary dimension before primary dimension 2 = Secondary dimension after primary dimension 3 = Secondary dimension above primary dimension 4 = Secondary dimension below primary dimension
3	UI	Integer	Units indicator
4	CHRSET	Integer	Character Set Interpretation (default= 1): 1 = Standard ASCII 1001 = Symbol Font 1 1002 = Symbol Font 2 1003 = Drafting Font
5	USTRING	String	String used in formatting value
6	FFLAG	Integer	Fraction Flag 0 = Show value as decimal 1 = Show value as fraction
7	PREC	Integer	Precision/Denominator Number of decimal places when FFLAG=0 Denominator of fraction when FFLAG=1

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.125 DIMENSION TOLERANCE PROPERTY (FORM 29)‡

### 4.125 Dimension Tolerance Property (Form 29)‡

‡The Dimension Tolerance Property Entity has not been tested. See Section 1.9.

ECO630

The Dimension Tolerance Property provides tolerance information for a dimension. This information can be used by the receiving system to regenerate the dimension.

A dimension may point to 0, 1, or 2 Dimension Tolerance Properties. SFLAG indicates whether the property applies to the primary or to the secondary dimension value.

TYP indicates which tolerance format should be displayed. Figure 132 illustrates the available tolerance formats.

UTOL and LTOL are the upper and lower tolerance values, in the units of the value being tolerated. For bilateral tolerances, UTOL is used as the tolerance value. When only one tolerance value is to be displayed, the other value is ignored.

SSPFLG indicates whether the plus sign should be suppressed when the upper tolerance is displayed. TRUE implies suppress the display of the plus sign.

When FFLAG is 0, values are displayed as decimal numbers and PREC specifies the number of digits to be displayed to the right of the decimal point. When FFLAG is 1, values are displayed as mixed fractions and PREC specifies the value to be used as the denominator of the fraction. When FFLAG is 2, values are displayed as fractions. The following table illustrates the use of FFLAG and PREC:

PREC	Value	FFLAG = 0	FFLAG = 1	FFLAG = 2
0	2.65	3	3	3
1	2.65	2.7	3	$\frac{3}{1}$
2	2.65	2.65	$2\frac{1}{2}$	$\frac{5}{2}$
2	2.4	2.40	$2\frac{1}{2}$	$\frac{5}{2}$
3	2.65	2.650	$2\frac{1}{2}$	$\frac{8}{3}$
3	1.93	1.930	2	$\frac{6}{3}$

If PREC is 0, then values are displayed as whole numbers.

A single instance of this property may be pointed to by several dimensions.

## 4.125 DIMENSION TOLERANCE PROPERTY (FORM 29)‡

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**0202**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
406	< n.a. >	< n.a. >	#	29			#		D # + 1

### Parameter Data

Index	Name	Type	Description
1	NP	Integer	Number of property values (NP=8)
2	SFLAG	Integer	Secondary tolerance flag 0 = Tolerance applies to primary dimension 1 = Tolerance applies to secondary dimension 2 = Display values as fractions
3	TYP	Integer	Tolerance type (no default ) 1 = Bilateral 2 = Upper/Lower 3 = Unilateral upper 4 = Unilateral lower 5 = Range - min before max 6 = Range - min after max 7 = Range - min above max 8 = Range - min below max 9 = Nominal + Range - min above max 10 = Nominal + Range - min below max
4	TPFLAG	Integer	Tolerance placement (default = 2) 1 = Placement before nominal value 2 = Placement after nominal value 3 = Placement above nominal value 4 = Placement below nominal value
5	UTOL	Real	Upper or bilateral tolerance value
6	LTOL	Real	Lower tolerance value
7	SSPFLG	Logical	Sign suppression flag (TRUE implies suppress the display of the plus sign.)
8	FFLAG	Integer	Fraction flag 0 = Display values as decimal numbers 1 = Display values as mixed fractions 2 = Display values as fractions
9	PREC	Integer	Precision for value display

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.125 DIMENSION TOLERANCE PROPERTY (FORM 29)‡**

1. 00±0. 01      1. 00  $^{+0. 01}_{-0. 02}$       1. 00  $^{+0. 01}$

1 - BILATERAL

2 - UPPER/LOWER

3 - UPPER

1. 00  $_{-0. 02}$ 

0. 98-1. 01

1. 01-0. 98

4 - LOWER

5 - RANGE - BEFORE

6 - RANGE - AFTER

0. 98  
1. 011. 01  
0. 981. 00 0. 98  
1. 01

7 - RANGE-ABOVE

8 - RANGE-BELOW

9 - NOMINAL +  
RANGE-ABOVE1. 00 1. 01  
0. 9810 - NOMINAL +  
RANGE - BELOW

Figure 132. Examples of tolerance formats (UTOL = 0.01, LTOL = -0.02)

## **4.126 DIMENSION DISPLAY DATA PROPERTY (FORM 30)‡**

### **4.126 Dimension Display Data Property (Form 30)‡**

‡The Dimension Display Data Property Entity has not been tested. [See Section 1.9.](#)

ECO630

The Dimension Display Data Property is optional but, when present, shall be referenced by a ECO630 dimension entity. The information it contains could be extracted from the text, leader, and witness line data with difficulty. Display data is saved with dimensions by many systems.

DT=2 if, and only if, a Basic Dimension Property ([Type 406, Form 31](#)) is also associated with the ECO630 same dimension.

An example of a label in a dimension is “Radius” in “Radius 3 ft.” In this example the preferred ECO630 label position LP= 1 (before) and the label string LS=6HRadius. Had the text instead been “3 Ft. Radius,” LP=2. The word “preferred” is used because a system may have to place the label above instead of before if the space between the witness lines is too small to accommodate strung-out text. CHRSET, the font characteristic for the label, is particularly important when the label is a special character like a diameter symbol that only exists in some fonts. The diameter symbol in font 1003 has the same ASCII code as lowercase “n” in conventional fonts. Thus, CHRSET=1003, LS=1Hn conveys that the label is a diameter symbol.

The witness line angle is the angle in dimension definition space (the plane of the dimension text) measured counterclockwise between the first witness line and the line between the arrowheads.

TA=0 means that the text is to appear parallel to the XT-axis in dimension definition space. TA= 1 means that the text is to run parallel to the line between the two arrowheads.

TP=0 means that, if the text can fit between the witness lines, it should be placed there as in ECO630 [Figure 133](#). TP=1 means that the text ideally belongs outside the first-listed witness line, as in [Figure 133](#).

Sometimes extra text, called a note, is affixed to the dimension. If one or more notes exist, the ECO630 Supplemental Note Position (SNP) indicates where each block of text is to be placed relative to the rest of the dimension text. The Note Start (NS) and Note End (NE) fields specify which strings in the General Note, pointed to by the dimension, comprise each supplemental note. The note starts with the NSth string and ends with the NEth, inclusive.

An instance of this property shall be pointed to by more than one dimension if, and only if, there ECO630 are no supplemental notes. A particular dimension entity shall reference at most one instance of this property.

## 4.126 DIMENSION DISPLAY DATA PROPERTY (FORM 30)‡

### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level <i>&lt; n.a. &gt;</i>	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **0202**	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 30	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP=14)
2	DT	Integer	Dimension Type 0 = Ordinary 1 = Reference (usually with parentheses) 2 = Basic (boxed)
3	LP	Integer	Preferred label position 0 = Does not exist 1 = Before measurement 2 = After measurement 3 = Above measurement 4 = Below measurement
4	CHRSET	Integer	Character Set Interpretation (default=1) Meaningful only if LS is non-empty: 1 = Standard ASCII 1001 = Symbol Font 1 1002 = Symbol Font 2 1003 = Drafting Font
5	LS	String	e.g., 8HDIA METER
6	DS	Integer	Decimal symbol 0 = " ." (period) 1 = " , " (comma)
7	WLA	Real	Witness line angle in radians. Default is $\pi/2$
8	TA	Integer	Text alignment 0 = Horizontal 1 = Parallel
9	TL	Integer	Text level 0 = Neither above nor below the leaders(s) (default) 1 = Above 2 = Below
10	TP	Integer	Preferred text placement 0 = Between the witness lines (default) 1 = Outside, near the first the witness line 2 = Outside, near the second the witness line
11	AH	Integer	Arrowhead orientation 0 = In, pointing out 1 = Out, pointing in
12	IV	Real	The primary dimension initial value
13	K	Integer	Number of supplemental notes, or zero

#### 4.126 DIMENSION DISPLAY DATA PROPERTY (FORM 30)‡

14	SNP ( 1 )	Integer	First supplemental note 1 = Before the rest of the dimension text 2 = After, but starting at the same level 3 = Above 4 = Below
15	NS ( 1 )	Integer	First note start index
16	NE ( 1 )	Integer	First note end index
:	:	:	
11+3*K	SNP ( K )	Integer	Last supplemental note
12+3*K	NS ( K )	Integer	Last note start index
13+3*K	NE ( K )	Integer	Last note end index

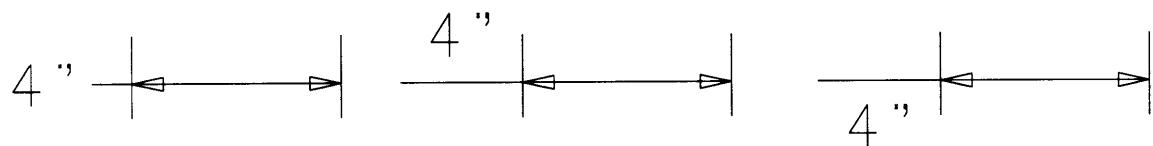
Additional pointers as required (see Section 2.2.4.5.2).



$TL=0, \quad TP=0$

$TL=1, \quad TP=0$

$TL=2, \quad TP=0$



$TL=0, \quad TP=1$

$TL=1, \quad TP=1$

$TL=2, \quad TP=1$

Figure 133. Placement of Text Using TP and TL

## 4.127 BASIC DIMENSION PROPERTY (FORM 31)‡

### 4.127 Basic Dimension Property (Form 31)‡

‡The Basic Dimension Property Entity has not been tested. See Section 1.9.

ECO630

The Basic Dimension Property indicates that the referencing dimension entity is to be displayed ECO630 with a box around the text. Preprocessors are responsible for providing the coordinates of the box corners. The coordinates may be ignored by postprocessors for systems that support the functionality of a Basic dimension; systems without this intrinsic functionality shall draw a box by using the coordinates provided.

The coordinates represent an ordered list beginning in the lower left corner proceeding counter-clockwise. A rectangular box is drawn connecting these points, starting and terminating at the first point.

This property inherits the Hierarchy attributes (line font, view, level, blank status, line weight, and color number) of the dimension that points to it, and it shall have the same transformation matrix processing applied to it.

An instance of this property shall not be pointed to by more than one dimension. An instance of this property shall have its Subordinate Entity Switch set to Physically Dependent.

An example of the Basic Dimension Property is shown in [Figure 134](#).

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **0102**	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 31	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

Index	Name	Type	Description
1	NP	Integer	Number of property values (NP=8)
2	LLX	Real	Coordinates of Lower Left corner
3	LLY	Real	
4	LRX	Real	Coordinates of Lower Right corner
5	LRY	Real	
6	URX	Real	Coordinates of Upper Right corner
7	URY	Real	
8	ULX	Real	Coordinates of Upper Left corner
9	ULY	Real	

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.127 BASIC DIMENSION PROPERTY (FORM 31)‡**

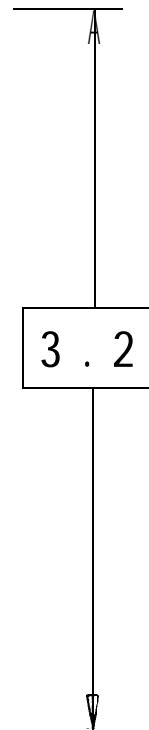


Figure 134. F40631X.IGS Example of Basic Dimension property

## 4.128 DRAWING SHEET APPROVAL PROPERTY (TYPE 406, FORM 32)‡

### 4.128 Drawing Sheet Approval Property (Type 406, Form 32)‡

‡The Drawing Sheet Approval Property Entity has not been tested. [See Section 1.9.](#)

ECO630

The Drawing Sheet Approval Property specifies the authorizing notation that signifies a drawing has been reviewed and accepted. It contains fields for the individual's name (NAME), their department or organizational function (ORG), and a date and time stamp (DATE).

ECO630

This property may be referenced only by a Drawing Entity ([Type 404](#)), and represents approval for one or more drawings or sheets within a drawing.

Multiple instances of this property may be referenced by the same entity, indicating that different individuals have given their approval. A single instance of this property may be referenced by multiple entities, indicating that the same individual has approved multiple sheets at the same time.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **0101**	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 32	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

Index	Name	Type	Description
1	NP	Integer	Number of property values (NP=3)
2	NAME	String	Individual's name
3	ORG	String	Individual's department or organization
4	DATE	String	Date & time of approval (same format as Global Section, <i>i.e.</i> , 15YYYYMMDD.HHNNSS or 13HYYMMDD.HHNNSS)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.129 DRAWING SHEET ID PROPERTY (TYPE 406, FORM 33)‡

### 4.129 Drawing Sheet ID Property (Type 406, Form 33)‡

‡The Drawing Sheet ID Property Entity has not been tested. See Section 1.9.

ECO630

The Drawing Sheet ID Property Property is used to identify (a) the sequence of a particular sheet in relation to other sheets of the drawing, and (b) a specific version of the drawing sheet.

The drawing sheet number (SNUM) is typically in a sequential series. The drawing sheet revision identifier (SID) is an alphanumeric string.

This property shall be referenced only from a Drawing Entity (Type 404), and only one instance ECO630 shall be referenced per drawing sheet. Each instance within a file shall be unique and referenced only once; *i.e.*, two drawing sheets within a file shall not have the same Sheet ID.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level <i>&lt; n.a. &gt;</i>	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display <i>&lt; n.a. &gt;</i>	(9) Status Number **0101**	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 33	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

Index	Name	Type	Description
1	NP	Integer	Number of property values (NP=2)
2	SNUM	Integer	Drawing sheet number
3	SID	String	Drawing sheet revision identifier

Additional pointers as required (see Section 2.2.4.5.2).

## 4.130 UNDERSCORE PROPERTY (TYPE 406, FORM 34)‡

### 4.130 Underscore Property (Type 406, Form 34)‡

‡The Underscore Property Entity has not been tested. See Section 1.9.

ECO630

The Underscore Property is used to communicate underscoring in text strings of a General Note Entity (Type 212). The underscoring for a text string is specified by the index number of the text string in the General Note and by the index numbers of the first and last characters in the text string to be underscored. Note: multiple underscore specifications can occur for each text string of a General Note. The exact positioning of the underscoring is system dependent. Examples of the Underscore Property are shown in Figure 135 with:

$$T(1) = 1, F(1) = 1, L(1) = 10$$

$$T(2) = 3, F(2) = 1, L(2) = 4$$

$$T(3) = 4, F(3) = 5, L(3) = 11$$

**Requirements:** An instance of this property shall only be referenced by one General Note Entity (Type 212). The color of the underscoring shall be the same as the color of the General Note Entity.

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
406	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**0101**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
406	< n.a. >	< n.a. >	#	34				#	D # + 1

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP = 1+ND*3)
2	ND	Integer	Number of underscore specifications (ND >= 1)
3	T(1)	Integer	Index of first text string with underscoring
4	F(1)	Integer	Index of first character to be underscored in text string T(1)
5	L(1)	Integer	Index of last character to be underscored in text string T(1)
:	:	:	
ND*3	T(ND)	Integer	Index of last text string with underscoring
1+ND*3	F(ND)	Integer	Index of first character to be underscored in text string T(ND)
2+ND*3	L(ND)	Integer	Index of last character to be underscored in text string T(ND)

Additional pointers as required (see Section 2.2.4.5.2).

## 4.131 OVERSCORE PROPERTY (TYPE 406, FORM 35)‡

### 4.131 Overscore Property (Type 406, Form 35)‡

‡The Overscore Property Entity has not been tested. See Section 1.9.

ECO630

The Overscore Property is used to communicate overscoring in text strings of a General Note Entity (Type 212) or Text Display Template Entity (Type 312). The overscoring for a text string is specified by the index number of the text string in the General Note and by the index numbers of the first and last characters in the text string to be overscored. Note: multiple overscore specifications can occur for each text string of a General Note. The exact positioning of the overscoring is system dependent. Examples of the overscore Property are shown in Figure 135 with:

$$T(1) = 2, F(1) = 1, L(1) = 9$$

$$T(2) = 3, F(2) = 1, L(2) = 4$$

$$T(3) = 4, F(3) = 1, L(3) = 7$$

**Requirements:** An instance of this property shall only be referenced by one General Note Entity (Type 212). The color of the overscoring shall be the same as the color of the General Note Entity.

#### Directory Entry

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **0101**	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight 	(13) Color Number 	(14) Parameter Line Count #	(15) Form Number 35	(16) Reserved 	(17) Reserved 	(18) Entity Label 	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

ECODH

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NP	Integer	Number of property values (NP = 1+ND*3)
2	ND	Integer	Number of overscore specifications (ND>=1)
3	T(1)	Integer	Index of first text string with overscoring
4	F(1)	Integer	Index of first character to be overscored in text string T(1)
5	L(1)	Integer	Index of last character to be overscored in text string T(1)
:	:	:	
ND*3	T(ND)	Integer	Index of last text string with overscoring
1+ND*3	F(ND)	Integer	Index of first character to be overscored in text string T(ND)
2+ND*3	L(ND)	Integer	Index of last character to be overscored in text string T(ND)

Additional pointers as required (see Section 2.2.4.5.2).

**4.131 OVERSCORE PROPERTY (TYPE 406, FORM 35)‡**

Underscore

---

Overscore

---

Both

---

Overlapping

Figure 135. F40635X.IGS Examples defined using the underscore and overscore properties

**4.132 Closure Property (Type 406, Form 36)‡**

‡The Closure Property Entity has not been tested. See Section 1.9.

ECO630

The Closure Property ([Type 406, Form 36](#)) exchanges the concept of closure for curve or surface entities. The property distinguishes between closure and the more restrictive case of “simple” closure (e.g., both a circle and a figure 8 are “closed,” but only the circle is a simple closed curve).

U and V are defined as follows: The untrimmed domain of  $S(u, v)$  is a rectangle,  $D$ , consisting of those points  $(u, v)$  such that  $a \leq u \leq b$  and  $c \leq v \leq d$  for given constants  $a, b, c$ , and  $d$  with  $a < b$  and  $c < d$ . The mapping  $S = S(u, v) = (x(u, v), y(u, v), z(u, v))$  is defined for each ordered pair  $(u, v)$  in  $D$ .

A surface is closed in  $u$  if the model-space images of the parameter-space curves  $u = \text{minimum}$  and  $u = \text{maximum}$  are the same, and similarly for  $v$ . A surface is “simple closed” if it is not self-intersecting except possibly along the parametric boundaries  $u = \text{minimum}$ ,  $u = \text{maximum}$ ,  $v = \text{minimum}$ ,  $v = \text{maximum}$ .

[Figure 136](#) illustrates use of this property.

- The cylinder is a simple closed surface in U (values= 1,2)
- The torus is a simple closed surface in U and V (values= 2,2,2)
- The trapezoid wrapped into a cylinder and the self-intersecting rectangle are partially closed and shall not reference this property.

Requirements: Multiple geometry entities may reference a single instance of this property if the entities have exactly the same closure situation. This property shall not be referenced by the Bounded Surface Entity ([Type 143](#)) nor by the Trimmed (Parametric) Surface Entity ([Type 144](#)) directly; however, the boundary curves referenced by these surface entities may reference this property. Partially closed surfaces shall not reference this property.

**Directory Entry**

(1) Entity Type Number 406	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number 00010300	(10) Sequence Number D #
(11) Entity Type Number 406	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 36	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Parameter Data**

Index	Name	Type	Description
1	N P	Integer	Number of property values (1 or 2; may not be defaulted)
2	CLOSEDU	Integer	U flag for curves or surfaces: 0=not specified (default), 1=closed, 2=simple closed
3	CLOSEDV	Integer	V flag for surfaces only: 0=not specified (default), 1=closed, 2=simple closed

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.132 CLOSURE PROPERTY (TYPE 406, FORM 36)‡**

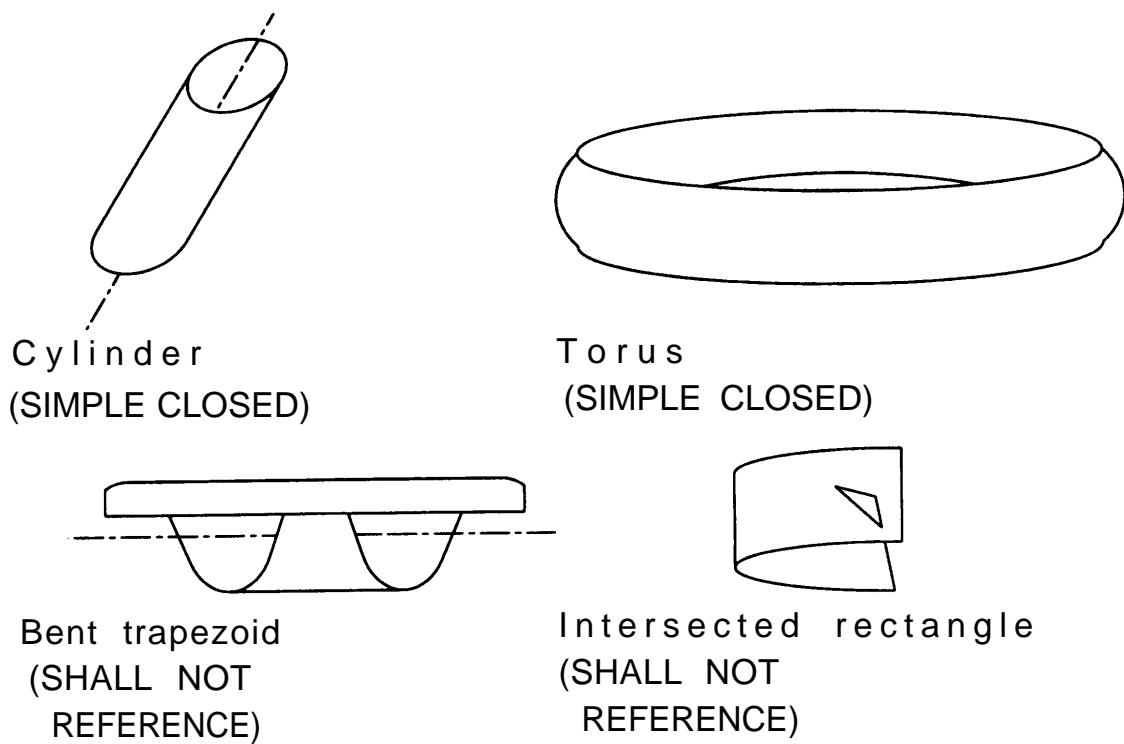


Figure 136. Use of the Closure Property

## 4.133 SINGULAR SUBFIGURE INSTANCE ENTITY (TYPE 408)

### 4.133 Singular Subfigure Instance Entity (Type 408)

This entity defines the occurrence of a single instance of the defined subfigure ([Type 308](#)). See [Figure 137](#) and [Section 3.6.2](#). [Figure 138](#) shows examples of subfigure instances:

Normal	Scale is 1.0; rotation is zero.
45°	Scale is 1.0; rotation is $\pi/4$ .
Twice	Scale is 2.0; rotation is $\pi$ .
One-half	Scale is 0.5; rotation is $\pi/2$ .

Note: The rotations are contained in the associated transformation matrices.

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
408	⇒	< n.a. >	#,⇒	#,⇒	0,⇒	0,⇒	0,⇒	?????????	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
408	#	#,⇒	#	0				#	D # + 1

**Note:** When the Hierarchy is set to Global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, Color Number, Level, View, and Blank Status.

#### Parameter Data

Index	Name	Type	Description
1	DE	Pointer	Pointer to the DE of the Subfigure Definition Entity
2	X	Real	Translation data relative to either model space or to the definition space of a referring entity
3	Y	Real	
4	Z	Real	
5	S	Real	Scale factor (default = 1.0)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### 4.133 SINGULAR SUBFIGURE INSTANCE ENTITY (TYPE 408)

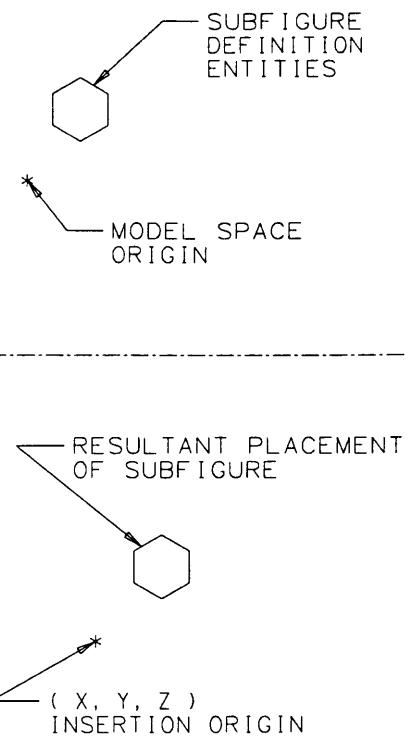


Figure 137. Relationship Between Subfigure Definition and Subfigure Instance

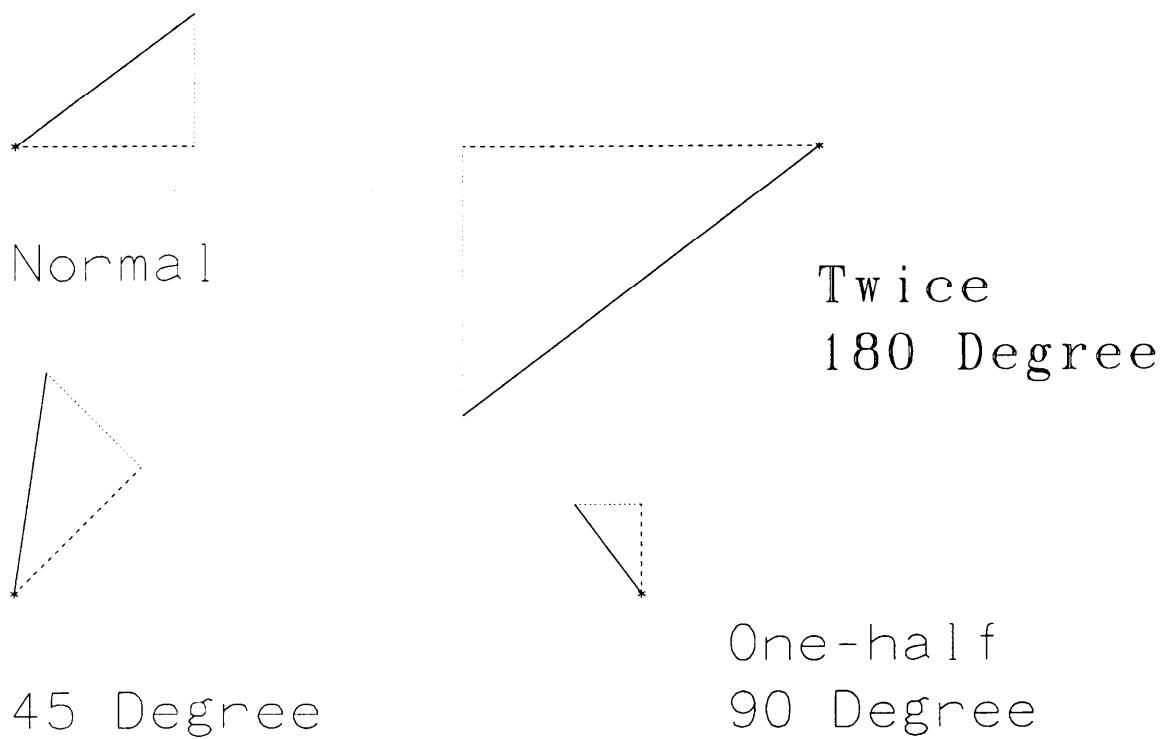


Figure 138. F408X.IGS Examples of Subfigure Instances at Various Scales and Orientations

#### 4.134 View Entity (Type 410)

The View Entity defines a framework for specifying a viewing orientation of an object in three ECO630 dimensional model space (X, Y, Z). The framework is also used to support the projection of all or part of model space onto a view plane. Two types of projection are specified, an orthographic parallel projection, described in this Section and a perspective projection, described in [Section 4.135](#). The perspective projection is untested. [See Section 1.9](#).

**Orthographic Parallel Projection.** An orthographic parallel projection onto a view plane of an object in model space is formed by passing rays normal to the view plane through each point of the object and finding the intersection with the view plane as shown in [Figure 139](#).

**View Coordinate System.** The view plane can be described by introducing a right-handed view coordinate system, (XV, YV, ZV) into model space. The view plane is the XV, YV plane, *i.e.*, the plane ZV=0. The view direction is along the positive ZV axis toward the view plane, *i.e.*, in the direction of the vector (0,0,-1). The positive YV axis points in the “up” direction in the resulting view. The point (0,0,0) in the view coordinate system ([see Figure 140](#)) is called the view origin. Thus, a complete viewing orientation is specified by a view coordinate system.

**View Coordinates Obtained from Model Coordinates.** View coordinates are obtained from model coordinates through translation and rotation. There are several ways that systems specify the data required to transfer from model to view coordinates. However, in each case, the data can be recorded using Form 0 of the Transformation Matrix Entity such that the model coordinates are taken as input and the view coordinates are produced as output, as follows, where R denotes the rotation matrix and T the translation vector ([see Section 4.21](#)):

$$\begin{bmatrix} XV \\ YV \\ ZV \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

In this situation, R is called the view matrix.

The View Entity specifies the view matrix and the translation vector by use of a pointer to a Transformation Matrix Entity in DE Field 7. In the special case when the view matrix is the identity matrix and there is zero translation, a zero value in DE Field 7 may be used.

**Example 1:** (View coordinates obtained from model coordinates by a translation and then a rotation.)

The system defines a viewing orientation by specifying a view origin (XO,YO, ZO) in model space and a rotation matrix so that:

$$\begin{bmatrix} XV \\ YV \\ ZV \end{bmatrix} = \begin{bmatrix} 3 \times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix} \left( \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \begin{bmatrix} XO \\ YO \\ ZO \end{bmatrix} \right)$$

or:

$$\begin{bmatrix} XV \\ YV \\ ZV \end{bmatrix} = \begin{bmatrix} 3 \times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \begin{bmatrix} 3 \times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix} \begin{bmatrix} XO \\ YO \\ ZO \end{bmatrix}.$$

Therefore, the rotation matrix is the view matrix, and in the Transformation Matrix Entity:

$$R = \begin{bmatrix} 3 \times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix}, \quad T = - \begin{bmatrix} 3 \times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix} \begin{bmatrix} XO \\ YO \\ ZO \end{bmatrix}$$

**Example 2:** (View coordinates obtained from model coordinates by a rotation and then a translation.)

The system defines a viewing orientation by specifying a rotation matrix and a translate or pan vector (XL, YL, ZL) expressed in the rotated coordinate system, so that:

$$\begin{bmatrix} XV \\ YV \\ ZV \end{bmatrix} = \begin{bmatrix} 3 \times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \begin{bmatrix} XL \\ YL \\ ZL \end{bmatrix}$$

Therefore, the rotation matrix is the view matrix, and in the Transformation Matrix Entity:

$$R = \begin{bmatrix} 3 \times 3 \\ \text{Rotation} \\ \text{Matrix} \end{bmatrix}, \quad T = - \begin{bmatrix} XL \\ YL \\ ZL \end{bmatrix}.$$

**Simple Form of the View Entity.** The View Entity provides a view number for the purpose of identifying differing view orientations. However, no standard indexing scheme is presumed to exist.

In its simplest form, the View Entity consists of a pointer to the Transformation Matrix Entity (in DE Field 7), and a view number. The Transformation Matrix Entity specifies a view matrix R and a translation vector T as given in the preceding section.

**Projection of a View Volume.** In some cases, a view volume and a scale factor may be required to control the projection of the view into a two-dimensional drawing space specified by a Drawing Entity (see Section 4.96).

The view volume bounds that portion of the data which will be projected after clipping is performed. The view volume is a rectangular parallelepipeds with limits specified by Plane Entities (Type 108) defined in the model coordinate system. The absence of clipping in a particular direction may be indicated by setting the pointer for the appropriate Plane Entity equal to zero.

The Plane Entities used to define the view volume shall not be arbitrary planar definitions (see Figure 141). After the transformation from model coordinates to view coordinates, each plane shall be perpendicular to the appropriate view coordinate system axis (e.g., the left side of the view volume shall transform into a plane XV=constant). Only the unbounded form of the Plane Entity (Type 108, Form 0) is required for use as a clipping plane; if another form is encountered, the bounding curve and display symbol shall be ignored.

**Projection Operations.** The order of operations for the View Entity is as follows:

1. Transform from model to view space.
2. Perform clipping (if included).
3. Perform projection onto the view plane.
4. Transform from view space to drawing space.

For Form 0 of the Drawing Entity (Type 404), the projection onto the view plane and the transform from view space to drawing space can be controlled by the following equation in the case of orthographic parallel projection, where S is the scale factor and XORIGIN and YORIGIN are defined in the Drawing Entity (see Section 4.96):

$$\begin{bmatrix} XD \\ YD \end{bmatrix} = \begin{bmatrix} S & 0 & 0 \\ 0 & S & 0 \end{bmatrix} \begin{bmatrix} XV \\ YV \\ ZV \end{bmatrix} + \begin{bmatrix} XORIGIN \\ YORIGIN \end{bmatrix}$$

As with Form 0, the transformation for Form 1 of the Drawing Entity (Type 404) is controlled by the view scale factor S and the view origin drawing location. In addition, a rotation angle  $\theta$  applied as follows:

$$\begin{bmatrix} XD \\ YD \end{bmatrix} = S \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \end{bmatrix} \begin{bmatrix} XV \\ YV \\ ZV \end{bmatrix} + \begin{bmatrix} XORIGIN \\ YORIGIN \end{bmatrix}$$

**Entity Display.** The display of an entity in a particular view is controlled by the use of the view value in Field 6 of the Directory Entry for the entity. If this value is zero or undefined, the entity is displayed with its own characteristics in all views unless display is controlled by other parameters (e.g., pointed to by another entity such as Subfigure Definition or Drawing). If this value is a pointer to a View Entity, the entity is displayed with its own characteristics in only the one view.

The selection of multiple views, display characteristics, or both, for an entity may be made by using one of the Views Visible Associativity Entities (Type 402, Form 3, 4, or 19). The view value for the entity then is a pointer to this associativity instead of to a View Entity.

## 4.134 VIEW ENTITY (TYPE 410)

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
410	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	0, ⇒	< n.a. >	????01**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
410	< n.a. >	< n.a. >	#	0				#	D # + 1

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	VNO	Integer	View number
2	SCALE	Real	Scale factor (Default = 1.0)
3	XVMINP	Pointer	Pointer to left side of view volume (XVMIN plane), or zero
4	YVMAXP	Pointer	Pointer to top of view volume (YVMAX plane) or zero
5	XVMAXP	Pointer	Pointer to right side of view volume (XVMAX plane), or zero
6	YVMINP	Pointer	Pointer to bottom of view volume (YVMIN plane), or zero
7	ZVMINP	Pointer	Pointer to back of view volume (ZVMIN plane), or zero
8	ZVMAXP	Pointer	Pointer to front of view volume (ZVMAX plane), or zero

Additional pointers as required ([see Section 2.2.4.5.2](#)).

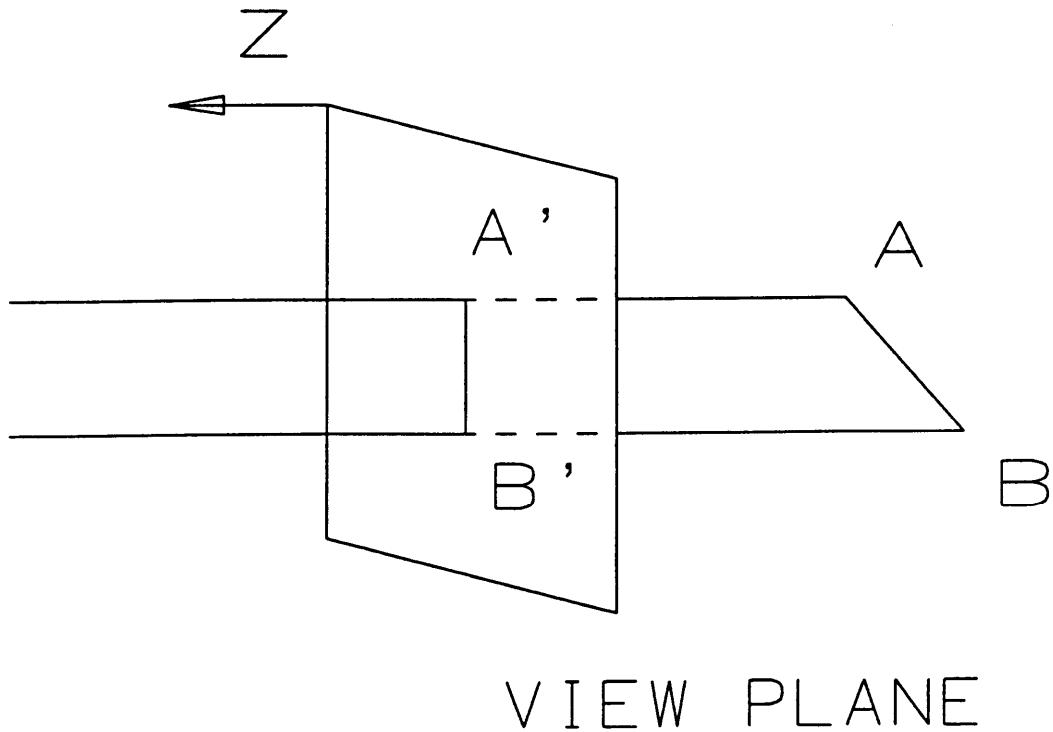


Figure 139. Orthographic Parallel Projection of AB on a View Plane

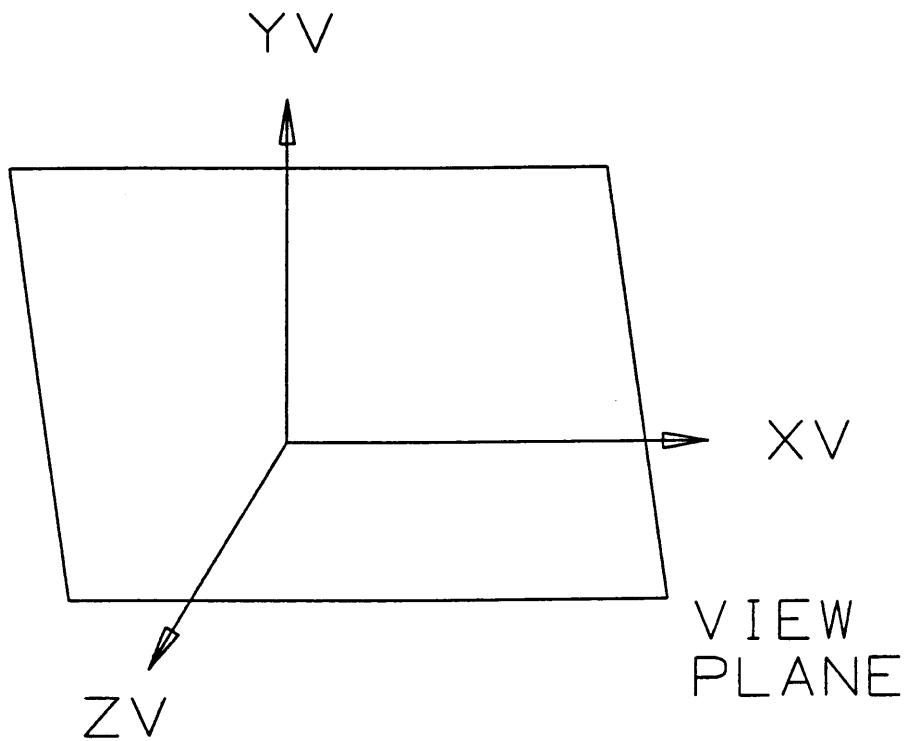


Figure 140. View Coordinate System (View Origin at  $XV=YV=ZV=0$ )

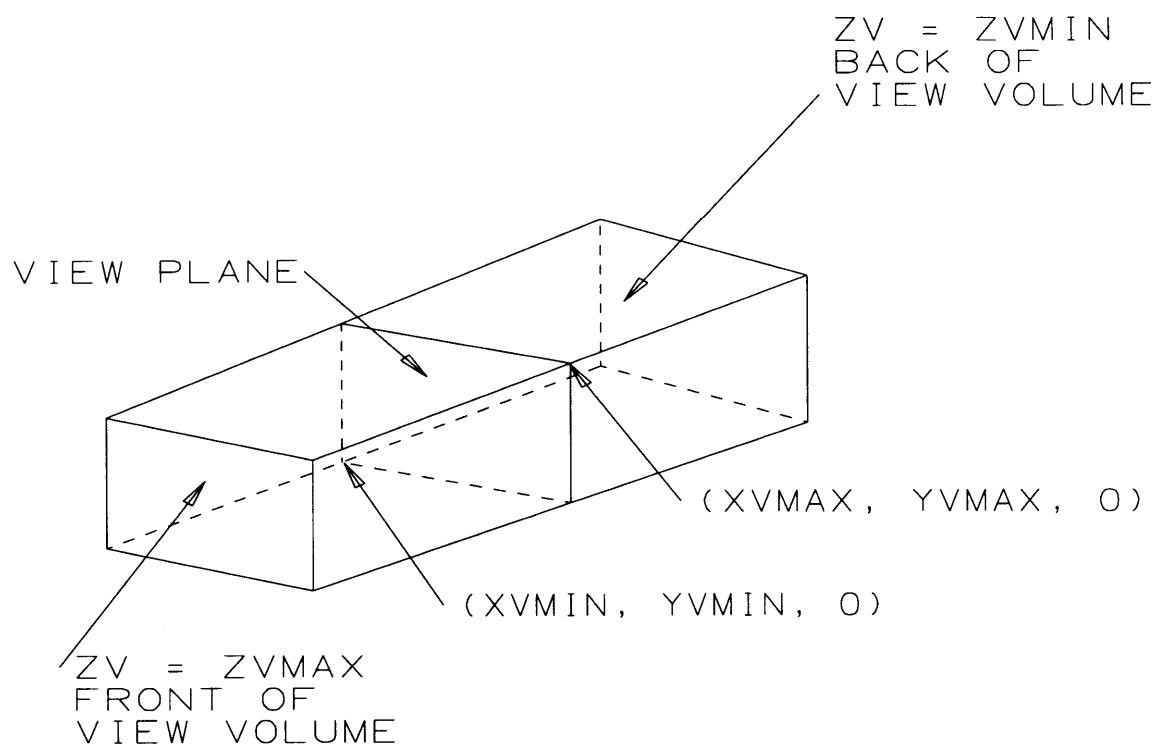


Figure 141. Planes Defining the View Volume

#### **4.135 PERSPECTIVE VIEW ENTITY (TYPE 410, FORM 1)‡**

##### **4.135 Perspective View Entity (Type 410, Form 1)‡**

‡The Perspective View Entity has not been tested. See Section 1.9.

The second form of the View Entity (Type 410, Form 1) supports a perspective view (see Figure 142). To avoid confusion, DE field 7 (pointer to a Transformation Matrix Entity) shall contain the value zero. For systems that require an orthogonal Transformation Matrix Entity (Type 124), see Appendix G for information on how to construct one from the information provided in the Parameter Data record.

Any geometric projection is defined by a view plane and the projectors that pass through the view plane. It is instructive to think of projectors as rays of light that form an image by passing through the viewed object and striking the view plane.

The *view plane* is positioned perpendicular to the view plane normal vector, at a specified view plane distance from the view reference point. ECO630

The *projectors* are defined via a point called the center of projection (also known as eye point). ECO630

In perspective views, all projectors emanate from the center of projection and pass through the view plane, as shown in Figure 142.

The view *coordinate system* is defined to be right-handed, with its origin at the view reference point. ECO630  
The view coordinate system has U, V, and W axes, where the V-axis is formed by orthographically projecting the VIEW UP vector onto the view plane. The U-axis is the cross-product of the V-axis crossed with the view plane normal. The W-axis corresponds to the view plane normal, offset to pass through the view reference point.

The view coordinate system is used in defining clipping windows and depth planes. The left and right sides of the clipping window are specified in view coordinates along the U-axis. The top and bottom sides of the clipping window are specified in view coordinates along the V-axis. The back and front clipping planes are specified in view coordinates along the W-axis. The use of view coordinates implies that the values for clipping windows and depth planes can be negative.

## 4.135 PERSPECTIVE VIEW ENTITY (TYPE 410, FORM 1)‡

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
410	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	0	< n.a. >	????01**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
410	< n.a. >	< n.a. >	#	1			#		D # + 1

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	VNO	Integer	View number
2	SCALE	Real	Scale factor
3	VPNX	Real	View plane normal vector (model space)
4	VPNY	Real	
5	VPNZ	Real	
6	VRPX	Real	View reference point (model space)
7	VRPY	Real	
8	VRPZ	Real	
9	CPX	Real	Center of projection (model space)
10	CPY	Real	
11	CPZ	Real	
12	VUPX	Real	View up vector (model space)
13	VUPY	Real	
14	VUPZ	Real	
15	VPD	Real	View plane distance (model space)
16	UMIN	Real	View coordinate denoting left side of clipping window
17	UMAX	Real	View coordinate denoting right side of clipping window
18	VMIN	Real	View coordinate denoting bottom of clipping window
19	VMAX	Real	View coordinate denoting top of clipping window
20	DCI	Integer	Depth clipping indicator: 0 = No depth clipping 1 = Back clipping plane ON 2 = Front clipping plane ON 3 = Back and front clipping planes ON
21	WMIN	Real	View coordinate denoting location of back clipping plane
22	WMAX	Real	View coordinate denoting location of front clipping plane

Additional pointers as required (see Section 2.2.4.5.2).

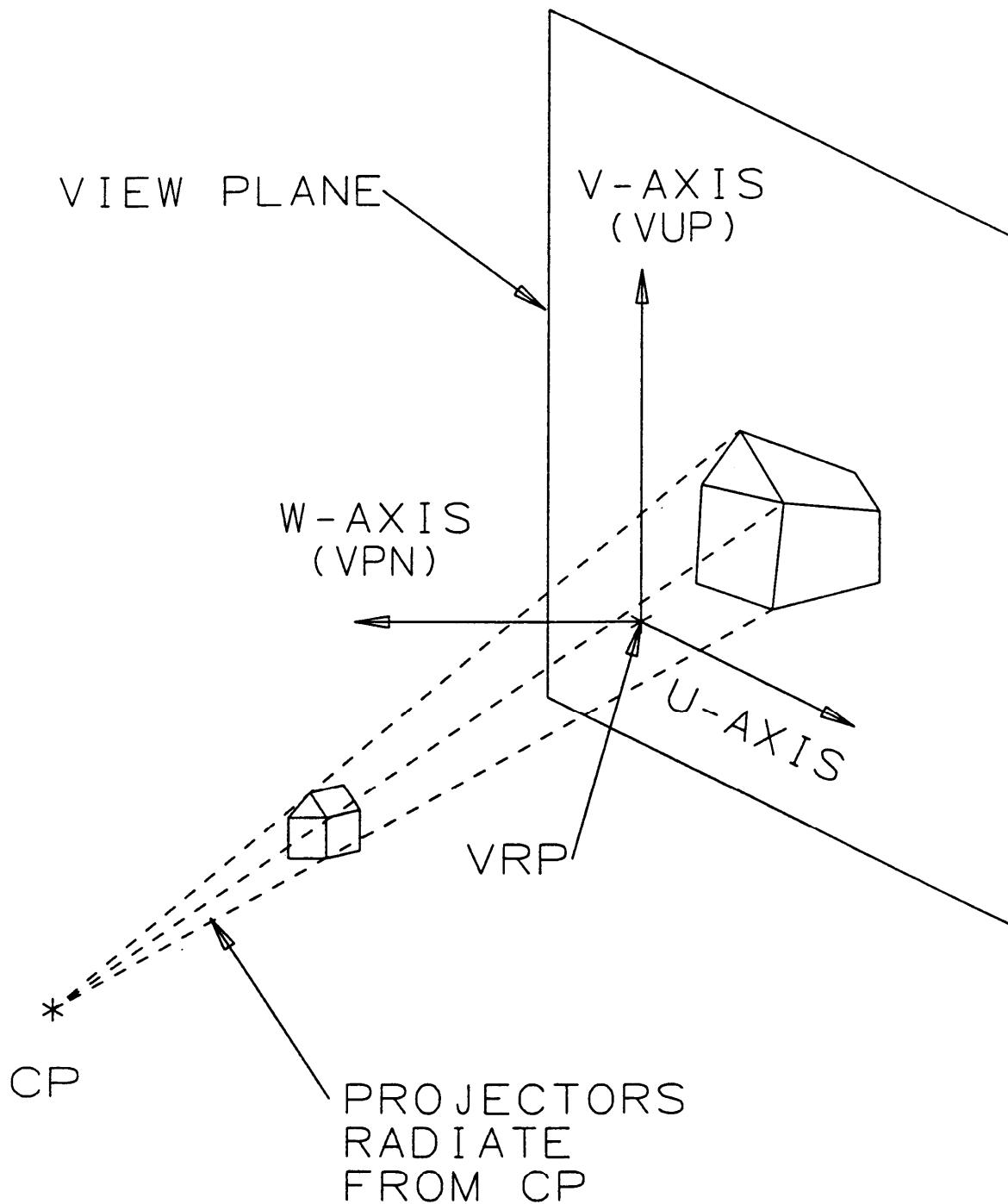


Figure 142. Definition of a perspective view entity. Broken lines indicate projectors. The abbreviations are: CP = center of projection, VPN = view plane normal, VRP = view reference point, and VUP = view up vector. The V-axis is the orthographic projection of the VUP vector onto the view plane.

## **4.136 RECTANGULAR ARRAY SUBFIGURE INSTANCE ENTITY (TYPE 412)**

### **4.136 Rectangular Array Subfigure Instance Entity (Type 412)**

The Rectangular Array Subfigure Instance Entity produces copies of an object called the base entity, ECO630 arranging them in equally spaced rows and columns. The following types of entities are valid for use as a base entity: Group Associativity Instance, Point, Line, Circular Arc, Conic Arc, Parametric Spline Curve, Rational B-Spline Curve, any annotation entity, Rectangular Array Subfigure Instance, Circular Array Subfigure Instance, or Subfigure Definition. 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 are given. This is where the first entity in the reproduction process is placed and is called position number 1. The successive positions are counted vertically up the first column, then vertically up the second column to the right, and so on.

The array of instance locations for the base entity is rotated about the line through the point (X,Y), parallel to the ZT-axis. The angle of rotation is specified in radians counterclockwise from the positive XT-axis. The instances of the base entity are not rotated from their original orientation.

A DO-DON'T flag controls which portion of the array is displayed. If the DO value is chosen, half or fewer of the elements of the rectangular array are to be defined. If the DON'T value is chosen, half or more of the elements of the rectangular array are to be defined.

## 4.136 RECTANGULAR ARRAY SUBFIGURE INSTANCE ENTITY (TYPE 412)

### Directory Entry

(1) Entity Type Number 412	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ?????????	(10) Sequence Number D #
(11) Entity Type Number 412	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** When the Hierarchy is set to Global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, Color Number, Level, View, and Blank Status.

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DE	Pointer	Pointer to the DE of the base entity
2	S	Real	Scale factor (default = 1.0)
3	X	Real	Coordinates of point to be used as lower left corner of array
4	Y	Real	
5	Z	Real	
6	NC	Integer	Number of columns
7	NR	Integer	Number of rows
8	DX	Real	Horizontal distance between columns
9	DY	Real	Vertical distance between rows
10	AX	Real	Rotation angle in radians
11	LC	Integer	DO-DON'T list count (LC=0 indicates all to be displayed.)
12	DDF	Integer	DO-DON'T flag: 0 = DO 1 = DON'T
13	N(1)	Integer	Number of first position to be proceed (DO), or not to be processed (DON'T)
:	:	:	
12+LC	N(LC)	Integer	Number of last position

Additional pointers as required (see Section 2.2.4.5.2).

#### **4.137 CIRCULAR ARRAY SUBFIGURE INSTANCE ENTITY (TYPE 414)**

##### **4.137 Circular Array Subfigure Instance Entity (Type 414)**

The Circular Array Subfigure Instance Entity produces copies of an object called the base entity, ECO630 arranging them around the edge of an imaginary circle whose center and radius are specified. The following types of entities are valid for use as a base entity: Group Associativity Instance, Point, Line, Circular Arc, Conic Arc, Parametric Spline Curve, Rational B-spline Curve, any annotation entity, Rectangular Array Subfigure Instance, Circular Array Subfigure Instance, or Subfigure Definition. The number of possible instance locations for the base entity is specified, and the location of the first instance position is specified in terms of a radius and a start angle measured positive, counterclockwise in radians from the line through the point (X, Y), parallel to the ZT-axis. 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 controls which portion of the array is displayed. If the DO value is chosen, half or fewer of the elements of the circular array are to be defined. If the DON'T value is chosen, half or more of the elements of the circular array are to be defined.

## 4.137 CIRCULAR ARRAY SUBFIGURE INSTANCE ENTITY (TYPE 414)

### Directory Entry

(1) Entity Type Number 414	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ?????????	(10) Sequence Number D #
(11) Entity Type Number 414	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** When the Hierarchy is set to global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, Color Number, Level, View, and Blank Status.

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	DE	Pointer	Pointer to the DE of the base entity
2	NE	Integer	Total number of possible instance locations
3	X	Real	Coordinates of center of imaginary circle
4	Y	Real	
5	Z	Real	
6	R	Real	Radius of imaginary circle
7	AS	Real	Start angle in radians
8	AD	Real	Delta angle in radians
9	LC	Integer	DO-DON'T list count (LC=0 indicates all replicated entities to be displayed)
10	DDF	Integer	DO-DON'T Flag: 0 = DO 1 = DON'T
11	N(1)		Number of first position to be processed (DO), or to be not processed (DON'T)
:	:	:	
10+LC	N(LC)	Integer	Number of last position

Additional pointers as required (see Section 2.2.4.5.2).

## 4.138 EXTERNAL REFERENCE ENTITY (TYPE 416)

### 4.138 External Reference Entity (Type 416)

The External Reference Entity provides a link between an entity in a referencing file and the definition of a logically related entity in a referenced file. In keeping with the concept of treating this entity as the definition which it replaces, the subordinate entity switch should be set as it would be on the definition replaced. [See Section 3.6.4](#) for the entities used in the linkage.

Five forms of the External Reference Entity are defined. Two of these forms are used to reference a definition and one form is a logical reference. Form 0 is used when a single definition from the referenced file is desired. This would be the case where the referenced file contained a collection of definitions. Form 1 is used when the entire file is to be instanced as a single definition. This would be the case where the referenced file contained a complete subassembly. Form 2 is used for external logical references where an entity in one file relates to an entity in a separate file (*e.g.*, when each sheet of a drawing is a separate file, and a flange on one sheet is also depicted on, or mates with, a flange on another sheet).

Forms 3 and 4 are used when a copy of the subfigure exists in native form on the receiving system. These forms shall only be used to replace the Subfigure Definition (Type 308) and Network Subfigure Definition (Type 320). Forms 3 and 4 have not been tested. [\(See Section 1.9.\)](#)

Form 3‡ of the External Reference Entity is used when a copy of the subfigure exists in native form on the receiving system; this form shall only be used to replace the Subfigure Definition Entity (Type 308), and Network Subfigure Definition Entity (Type 320).

Form 4‡ of the External Reference Entity is used when a copy of the subfigure exists in native form in a library on the receiving system; this form shall only be used to replace the Subfigure Definition Entity (Type 308) and Network Subfigure Definition Entity (Type 320).

‡Note: Forms 3 and 4 of the External Reference Entity have not been tested. [See Section 1.9.](#)

Forms 0, 2, 3, and 4 require an entity-unique symbolic name. The following entities and the parameter which supplies the symbolic name are identified for use.

Entity Type Number	Entity Name	Parameter Index	Description
132	Connect Point	9	CP Function Name (unique)
302	Associativity Definition	( )	Implementor assigned
304	Line Font Definition	( )	Implementor assigned
306	MACRO Definition	2	Entity Type Identification
308	Subfigure Definition	2	Subfigure name (unique)
310	Text Font Definition	2	Font name
312	Text Display Template	( )	Implementor assigned
314	Color Definition	( )	Implementor assigned
320	Network Subfigure Def.	2	Subfigure name (unique)

Possible alternatives for the entity-unique symbolic name for those entities marked "Implementor assigned" could be: (1) the property Reference Designator (Type 406, Form 7), or (2) the Entity Label, Entity Subscript (Directory Entry Fields 18 and 19).

## 4.138 EXTERNAL REFERENCE ENTITY (TYPE 416)

### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
416	⇒	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	< n.a. >	**?????**	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
416	< n.a. >	< n.a. >	#	0-4				#	D # + 1

ECO630

### Forms 0 and 2 of the External Reference Entity

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	EXTFID	String	External Reference File Identifier (contained as Global Parameter Number 4 in the referenced file)
2	EXTNAM	String	External Reference Entity Symbolic Name

Additional pointers as required ([see Section 2.2.4.5.2](#)).

### Form 1 of the External Reference Entity

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	EXTFID	String	External Reference File Identifier (contained as Global Parameter Number 4 in the referenced file)

Additional pointers as required ([see Section 2.2.4.5.2](#)).

#### **4.138 EXTERNAL REFERENCE ENTITY (TYPE 416)**

##### **Form 3‡ of the External Reference Entity**

###### **Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	EXTNAM	String	External Reference Entity Symbolic Name

Additional pointers as required ([see Section 2.2.4.5.2](#)).

##### **Form 4‡ of the External Reference Entity**

###### **Parameter Data**

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	LIBNAM	String	Name of library in which EXTNAM resides
2	EXTNAM	String	External Reference Entity Symbolic Name

Additional pointers as required ([see Section 2.2.4.5.2](#)).

‡Note: Forms 3 and 4 of the External Reference Entity have not been tested. [See Section 1.9](#).

## 4.139 NODAL LOAD/CONSTRAINT ENTITY (TYPE 418)

### 4.139 Nodal Load/Constraint Entity (Type 418)

This entity relates loads or constraints to specific nodes in the Finite Element Model. This is accomplished by creating a relation between Node Entities and the Tabular Data Property that contains the load or constraint data. Each load and constraint case will require a Nodal Load/Constraint Entity and a Tabular Data Property (Form 11) with PTYPE= 12. ECO630

[Figure 143](#) shows the relationship or linkage between the Nodal Load/Constraint Entity and the ECO630 Tabular Data Property which carries the load or constraint vector. The relationship or linkage is also shown on the General Note Entity which describes the load or constraint test case being performed. There is a one-to-one correspondence between the load case description and the General Note Entity and the pointer to the Tabular Data Property containing the load case magnitudes (see also [Section 3.6.6](#)).

#### Directory Entry

(1) Entity Type Number 418	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View 0, ⇒	(7) Xformation Matrix 0, ⇒	(8) Label Display 0, ⇒	(9) Status Number ??????**	(10) Sequence Number D #
(11) Entity Type Number 418	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

#### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NC	Integer	Total number of cases
2	TYPE	Integer	1 = Loads 2 = Constraints
3	DE	Pointer	Pointer to Node
4	PTR(1)	Pointer	Pointer to the DE of the first Tabular Data Property
:	:	:	
3+NC	PTR(NC)	Pointer	Pointer to the DE of the last Tabular Data Property

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.139 NODAL LOAD/CONSTRAINT ENTITY (TYPE 418)**

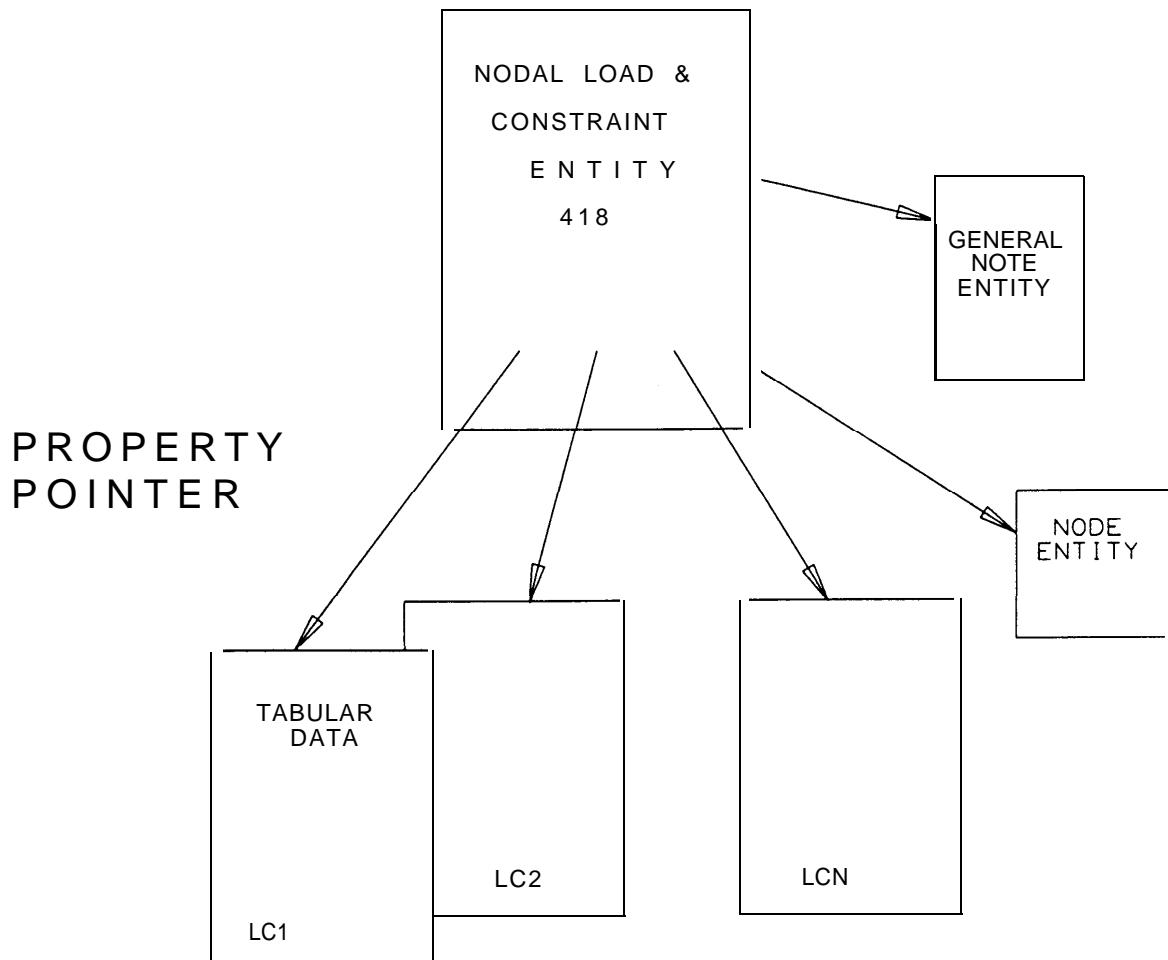


Figure 143. Relationship Between the Nodal Load/Constraint Entity and Tabular Data Properties

## **4.140 NETWORK SUBFIGURE INSTANCE ENTITY (TYPE 420)**

### **4.140 Network Subfigure Instance Entity (Type 420)**

Each instance of a Network Subfigure Definition Entity ([Type 320](#)) is specified by a Network Subfigure Instance Entity. Its use is described in [Section 3.6.2](#).

In addition, the points of connection (Connect Point Entity, [Type 132](#)) specified by the Network Subfigure Definition Entity must be instanced and associated with each Network Subfigure Instance (see indices 11 and 12).

There is a direct relationship between the points of connection in the Network Subfigure Definition Entity ([Type 320](#)) and the Network Subfigure Instance Entity ([Type 420](#)). The number of associated (child) Connect Point Entities ([Type 132](#)) in the instance shall match the number in the definition, their order shall be identical, and any unused points of connection in the instance shall be indicated by a null (zero) pointer.

The Type Flag Field (Index 8) implements the distinction between logical design and physical design data, and is required if both are present in the file.

The Network Subfigure Instance Entity allows different scale factors in the x, y, and z directions. This scaling is performed before the translation from X, Y, and Z and before the Transformation Matrix Entity pointed to in the Directory Entry (if any) is applied. The scaling does not apply to the model space placement coordinates (X, Y, Z).

## 4.140 NETWORK SUBFIGURE INSTANCE ENTITY (TYPE 420)

### Directory Entry

(1) Entity Type Number 420	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ?????????	(10) Sequence Number D #
(11) Entity Type Number 420	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** When the Hierarchy is set to Global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, Color Number, Level, View, and Blank Status.

### Parameter Data

ECO650

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	DE	Pointer	Pointer to the DE of the Network Subfigure Definition Entity
2	X	Real	Translation data relative to either model space or to the definition space of a referring entity
3	Y	Real	
4	Z	Real	
5	XS	Real	Scale factor in definition space x axis (default 1.0)
6	YS	Real	Scale factor in definition space y axis (default XS)
7	ZS	Real	Scale factor in definition space z axis (default XS)
8	TF	Integer	Type flag 0 = not specified (default) 1 = logical 2 = physical
9	PRD	String	Primary reference designator
10	DPTR	Pointer	Pointer to the DE of the primary reference designator Text Display Template Entity, or null. If null, no Text Display Template Entity specified.
11	NC	Integer	Number of associated (child) Connect Point Entities
12	CPTR(1)	Pointer	Pointer to the DE of the first associated Connect Point Entity, or zero
:	:	:	
11+NC	CPTR(NC)	Pointer	Pointer to the DE of the last associated Connect Point Entity, or zero

Additional pointers as required (see Section 2.2.4.5.2).

## 4.141 ATTRIBUTE TABLE INSTANCE ENTITY (TYPE 422)

### 4.141 Attribute Table Instance Entity (Type 422)

Each occurrence of an Attribute Table ([Type 322, Form 0](#) is represented by an Attribute Table Instance Entity ([Type 422](#)). Directory Entry Field 3 (Structure) of each instance contains a negated pointer to the Directory Entry of its corresponding Attribute Table Definition Entity. All forms of this entity have independent or dependent status. [See Section 2.2.4 .4.9.2](#) for more details.

**4.141.1 Attribute Table Instance (Form 0).** This form of the entity is for an instance of a single row or tuple.

#### Directory Entry

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
422	$\Rightarrow$	$\Rightarrow$	$< n.a. >$	$< n.a. >$	$< n.a. >$	$< n.a. >$	$< n.a. >$	$**?????**$	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16)	(17)	(18)	(19)	(20) Sequence Number
422	$< n.a. >$	$< n.a. >$	#	0	Reserved	Reserved	Entity Label	Entity Subscript	D # + 1

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
(first attribute instance)			
1	AV(1,1)	Variable	First attribute value
2	AV(1,2)	Variable	Second attribute value
:	:	:	
AVC(1)	AV(1,AVC(1))	Variable	Last attribute value
:	:	:	
Let M = AVC(1) + ... + AVC(NA-1)			
(last attribute instance)			
M+1	AV(NA,1)	Variable	First attribute value
M+2	AV(NA,2)	Variable	Second attribute value
:	:	:	
M+AVC(NA)	AV(NA,AVC(NA))	Variable	Last attribute value

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## 4.141 ATTRIBUTE TABLE INSTANCE ENTITY (TYPE 422)

**4.141.2 Attribute Table Instance (Form 1).** This form of the entity is for a table of attributes in row-major order; *i.e.*, the values for the first attribute through the last attribute for the first row are followed by the values for the first attribute through the last attribute for the second row, etc. ECO630

### Directory Entry

(1) Entity Type Number 422	(2) Parameter Data ⇒	(3) Structure ⇒	(4) Line Font Pattern < n.a. >	(5) Level < n.a. >	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display < n.a. >	(9) Status Number **?????**	(10) Sequence Number D #
(11) Entity Type Number 422	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 1	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
(attributes, first row, first column)			
1	NR	Integer	Number of rows
2	AV(1,1,1)	Variable	First attribute value
3	AV(1,1,2)	Variable	Second attribute value
⋮	⋮	⋮	⋮
1+AVC(1)	AV(1,1,AVC(1))	Variable	Last attribute value
⋮	⋮	⋮	⋮
Let M = 1 + AVC(1) + ... + AVC(NA-1)			
(attributes, first row, last column)			
M+1	AV(1,NA,1)	Variable	First attribute value
M+2	AV(1,NA,2)	Variable	Second attribute value
⋮	⋮	⋮	⋮
M+AVC(NA)	AV(1,NA,AVC(NA))	Variable	Last attribute value
⋮	⋮	⋮	⋮
Let M = 1 + NR*(AVC(1) + ... + AVC(NA)) - AVC(NA)			
(attributes, last row, last column)			
M+1	AV(NR,NA,1)	Variable	First attribute value
M+2	AV(NR,NA,2)	Variable	Second attribute value
⋮	⋮	⋮	⋮
M+AVC(NA)	AV(NR,NA,AVC(NA))	Variable	Last attribute value

Additional pointers as required (see Section 2.2.4.5.2).

## **4.142 SOLID INSTANCE ENTITY (TYPE 430)**

### **4.142 Solid Instance Entity (Type 430)**

The Solid Instance Entity provides a mechanism for replicating a solid representation. The solid pointed to in this entity is allowed to be:

- Primitive Entity
- Boolean Tree Entity
- Solid Assembly Entity
- Solid Instance Entity
- Manifold Solid B-Rep Object Entity

ECO644

Note that a transformation matrix may be pointed to by Field 7 of the DE to position this instance in any desired manner.

For the Solid Instance Entity, the Form numbers are:

ECO644

Form	Meaning
0	The solid pointed to is a primitive, solid instance, Boolean tree, or solid assembly
1	The solid pointed to is a manifold solid B-Rep object entity

## 4.142 SOLID INSTANCE ENTITY (TYPE 430)

### Directory Entry

(1) Entity Type Number 430	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Xformation Matrix 0,⇒	(8) Label Display 0,⇒	(9) Status Number ?????????	(10) Sequence Number D #
(11) Entity Type Number 430	(12) Line Weight #	(13) Color Number #,⇒	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Note:** When the Hierarchy is set to Global Defer (01), all of the following are ignored and may be defaulted: Line Font Pattern, Line Weight, ColorNumber, Level, View, and Blank Status.

### Parameter Data

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	PTR	Pointer	Pointer to the DE of the solid

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.143 Vertex Entity (Type 502)‡**

‡The Vertex Entity has not been tested. See Section 1.9.

ECO630

The geometry underlying a vertex is a point in  $R3$ . A vertex is the bound of an edge and can participate in the bounds of a face.

- There are no default values for the vertex.
- Transformations cannot be applied to a vertex.

**4.143.1 Vertex List Entity (Type 502, Form 1)** Form 1 of the Vertex Entity is the Vertex List Entity which contains one or more vertices. The Subordinate Entity Switch shall be set to Physically Dependent. (Independent Vertex Lists are not permitted.)

To avoid ambiguity, the Vertex List Entity shall not point to a transformation Matrix Entity (Type 124). The vertex coordinates are defined in model space such that if they were post-processed as Point Entities (Type 116), they would be properly oriented in 3D space such that tests for verification of tolerance could be performed.

The Vertex List Entity requires a list of 3D coordinates. Any properties associated with this entity apply to all vertices in the lists.

- The order of vertices in this list is not significant.

## 4.143 VERTEX LIST ENTITY (TYPE 502, FORM 1)

### Directory Entry

(1) Entity Type Number 502	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #, ⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display 0, ⇒	(9) Status Number ??01??**	(10) Sequence Number D #
(11) Entity Type Number 502	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 1	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

### Parameter Data

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of vertex tuples in list ( $N > 0$ )
2	X(1)	Real	Coordinates of first vertex
3	Y(1)	Real	
4	Z(1)	Real	
:	:	:	
-1+3*N	X(N)	Real	Coordinates of last vertex
3*N	Y(N)	Real	
1+3*N	Z(N)	Real	

Additional pointers as required (see Section 2.2.4.5.2).

**4.144 Edge Entity (Type 504)‡**

‡The Edge Entity has not been tested. See Section 1.9.

ECO630

The Edge Entity represents the topological construct corresponding to a line segment between two vertices. The edge is not closed since it does not contain the vertices ( $V_1$  and  $V_2$ ) which bound it. The start and terminate vertices do not have to be distinct.

Underlying curve geometry in  $R^3$  is required. These curves shall be represented parametrically and shall be continuous and non-self intersecting in the arc of the curve underlying the edge.

The natural orientation of the edge is in the same direction as its underlying curve in  $R^3$ . Thus the edge is traced from start vertex to terminate vertex as the underlying curve is traced in the direction of increasing parameter value.

**4.144.1 Edge List Entity (Type 504, Form 1)** Form 1 of the Edge Entity is the Edge List Entity. The list of curve entity types that may be used with the Edge List Entity is given below:

ECO630  
ECO630

Entity Type Number	Entity Type
100	Circular Arc
102	Composite Curve
104	Conic Arc
106/11	2D Path
106/12	3D Path
106/63	Simple Closed Planar Curve
110	Line
112	Parametric Spline Curve
126	Rational B-Spline Curve
130	Offset Curve

The Edge List Entity shall have its Subordinate Entity Switch set to Physically Dependent. (Independent Edge Lists are not permitted.) Its Hierarchy Flag shall be set to 01.

The start and terminate vertices are represented by a pointer to the DE of a Vertex List Entity (Type 502, Form 1) and by a list index into the list.

The Edge List Entity requires underlying curve geometry in  $R^3$ . Any properties associated with the entity are associated with all members of the list.

- 'The order of edges in this list is not significant.

**Directory Entry**

(1) Entity Type Number	(2) Parameter Data	(3) Structure	(4) Line Font Pattern	(5) Level	(6) View	(7) Xformation Matrix	(8) Label Display	(9) Status Number	(10) Sequence Number
504	⇒	< n.a. >	< n.a. >	#,⇒	< n.a. >	< n.a. >	0,⇒	??01??01	D #
(11) Entity Type Number	(12) Line Weight	(13) Color Number	(14) Parameter Line Count	(15) Form Number	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript	(20) Sequence Number
504	< n.a. >	< n.a. >	#	1				#	D # + 1

**Parameter Data**

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of edge tuples in list (N > 0)
2	CURV(1)	Pointer	Pointer to the DE of the first model space curve
3	SVP(1)	Pointer	Pointer to the DE of the Vertex List Entity ( <a href="#">Type 502, Form 1</a> ) for the first start vertex
4	SV(1)	Integer	List Index of the first start vertex in the Vertex List Entity
5	TVP(1)	Pointer	Pointer to the DE of the Vertex List Entity for the first terminate vertex
6	TV(1)	Integer	List Index of the first terminate vertex in the Vertex List Entity
:	:	:	
-3+5*N	CURV(N)	Pointer	Pointer to the DE of the last model space curve
-2+5*N	SVP(N)	Pointer	Pointer to the DE of the Vertex List Entity for the last start vertex
-1+5*N	SV(N)	Integer	List Index of the last start vertex in the Vertex List Entity
5 * N	TVP(N)	Pointer	Pointer to the DE of the Vertex List Entity for the last terminate vertex
1+5*N	TV(N)	Integer	List Index of the last terminate vertex in the Vertex List Entity

Additional pointers as required ([see Section 2.2.4.5.2](#)).

**4.145 Loop Entity (Type 508)‡**

‡The Loop Entity has not been tested. See Section 1.9.

ECO630

Form 1 of the Loop Entity specifies a bound of a face. Typically, a loop represents a connected collection of face boundaries, seams, and poles of a single face (refer to figures in Appendix I). Its underlying geometry is a connected curve or a single point in  $R3$ .

This form of the Loop Entity consists of a repeating construct, the edge use. This construct consists ECO630 of either an edge, an orientation, and optional parameter space curves, or (in the case of a pole) a vertex and an optional parameter space curve. If the edge use references an edge, the orientation describes whether the direction of this use of the edge is in agreement with the natural orientation of the edge. An edge-use shall be only used once in the shell.

Let  $P$  be a point on the arc of the  $R3$  curve,  $C$ , underlying an edge,  $E$ . Both  $P$  and  $C$  lie on surface ECO630  $S$ . Let  $N$  be the vector normal to  $S$  at point  $P$ .  $T$  is a vector at  $P$  whose direction is that of  $C$  at  $P$ .  $RT$  is the vector derived by reversing the direction of  $T$ . If the edge orientation is TRUE, the cross product  $N \times T$  points to the left of  $E$ . If the orientation is FALSE, the cross product  $N \times RT$  points to the left of the edge.

By convention, loops are oriented so that the material of the face they bound lies on the left.

The loop is represented as an ordered list of edge-uses ( $EU_i, i = 1, n$ ) which has the following properties:

- The terminal vertex of  $EU_i$  is the initial vertex of  $EU_{i+1}$ ,  $i = 1, n - 1$ .
- The loop is closed. This implies that the terminal vertex of  $EU_n$  is the same as the initial vertex of  $EU_1$ .
- The orientation of the loop is defined to be the same as its constituent edge-uses which reference edges. Therefore the direction of the loop at an edge-use which references a vertex,  $A$ , can be taken from any edge-use having an underlying edge which has  $A$  as either its start or terminate vertex.
- Material of the face lies on the left of the edge-uses which make up the loop.

This form of the Loop Entity is physically dependent on its parent entity, the Face Entity (Type 510, Form 1). (Independent Loops are not permitted.)

Each edge can be represented by either a list index into a Vertex List Entity (Type 502, Form 1), or a list index into an Edge List Entity (Type 504, Form 1).

**Directory Entry**

(1) Entity Type Number 508	(2) Parameter Data ⇒	(3) Structure < n.a. >	(4) Line Font Pattern < n.a. >	(5) Level #, ⇒	(6) View < n.a. >	(7) Xformation Matrix < n.a. >	(8) Label Display 0, ⇒	(9) Status Number ??01????	(10) Sequence Number D #
(11) Entity Type Number 508	(12) Line Weight < n.a. >	(13) Color Number < n.a. >	(14) Parameter Line Count #	(15) Form Number 0-1	(16) Reserved	(17) Reserved	(18) Entity Label	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Parameter Data**ECO650  
ECO630

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of edge tuples
2	TYPE(1)	Integer	Type of first edge 0 = Edge 1 = Vertex
3	EDGE(1)	Pointer	Pointer to the DE of the first Vertex List or Edge List Entity
4	NDX(1)	Integer	List Index into Vertex List or Edge List Entity
5	OF(1)	Logical	Orientation flag of first edge with respect to direction of the model space curve(s) (True = agrees)
6	K(1)	Integer	Number of underlying parameter space curves, or zero
7	ISOP(1,1)	Logical	Isoparametric flag of first parameter space curve (True = curve is isoparametric on the surface underlying the face which this loop bounds)
8	CURV(1,1)	Pointer	Pointer to the DE of the first parameter space curve in first edge
:	:	:	
5+2*K(1)	ISOP(1,K(1))	Logical	Isoparametric flag of last parameter space curve
6+2*K(1)	CURV(1,K(1))	Pointer	Pointer to the DE of the last parameter space curve in first edge
:	:	:	
M	TYPE(N)	Integer	Type of last edge
1+M	EDGE(N)	Pointer	Pointer to the DE of the last Vertex List or Edge List Entity
2+M	NDX(N)	Integer	List Index into Vertex List or Edge List Entity
3+M	OF(N)	Logical	Orientation flag of last edge with respect to direction of the model space curve(s)
4+M	K(N)	Integer	Number of underlying parameter space curves, or zero
5+M	ISOP(N,1)	Logical	Isoparametric flag of first parameter space curve
6+M	CURV(N,1)	Pointer	Pointer to the DE of the first parameter space curve in last edge
:	:	:	
3+M+2*K(N)	ISOP(N,K(N))	Logical	Isoparametric flag of last parameter space curve
4+M+2*K(N)	CURV(N,K(N))	Pointer	Pointer to the DE of the last parameter space curve in last edge

Additional pointers as required (see Section 2.2.4.5.2).

**4.146 Face Entity (Type 510)‡**

‡The Face Entity has not been tested. See Section 1.9.

ECO630

Form 1 of the Face Entity is a bound (partial) of  $R^3$  which has finite area. The face,  $F$ , has an underlying surface,  $S$ , and is bounded by one or more loops ( $L_i$ ,  $i = 1, m$ ). If more than one loop bounds a face, the loops shall be disjoint. The material of the face lies on the left of all the loops bounding the face. See the Loop Entity (Type 508, Form 1) for a definition of left.

This form of the Face Entity is physically dependent on its parent entity, the Shell Entity (Type 514). This form of the Face Entity requires an underlying surface which shall be one of the following entity types:

<b>Entity Type Number</b>	<b>Entity Type</b>
114	Parametric Spline Surface
118/1	Ruled Surface
120	Surface of Revolution
122	Tabulated Cylinder
128	Rational B-spline Surface
140	Offset Surface
190	‡Plane Surface
192	‡Right Circular Cylindrical Surface
194	‡Right Circular Conical Surface
196	‡Spherical Surface
198	‡Toroidal Surface

- The portion of the underlying surface of the face covered by the face interior (not including its bounding loops) shall be an oriented, connected, finite 2-manifold having no handles. The surface covered by the faces, together with its bounding loops, is not so restricted (see Figure 13 in Appendix I).
- The face does not contain its bounds.
- The bounds of a face are loops. The outer loop may be chosen arbitrarily by the sending system.
- The bounds of a face are disjoint.

**Directory Entry**

(1) Entity Type Number 510	(2) Parameter Data ⇒	(3) Structure <i>&lt; n.a. &gt;</i>	(4) Line Font Pattern <i>&lt; n.a. &gt;</i>	(5) Level #, ⇒	(6) View <i>&lt; n.a. &gt;</i>	(7) Xformation Matrix <i>&lt; n.a. &gt;</i>	(8) Label Display 0, ⇒	(9) Status Number ??01????	(10) Sequence Number D #
(11) Entity Type Number 510	(12) Line Weight <i>&lt; n.a. &gt;</i>	(13) Color Number <i>&lt; n.a. &gt;</i>	(14) Parameter Line Count #	(15) Form Number 1	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Parameter Data**

ECO650

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	SURF	Pointer	Pointer to the DE of the underlying surface
2	N	Integer	Number of loops ( $N > 0$ )
3	OF	Logical	Outer loop flag (True implies that the loop identified by LOOP1 is to be considered the outer loop. False implies that no outer loop is identified.)
4	LOOP(1)	Pointer	Pointer to the DE of the first loop of the face
:	:	:	
3+N	LOOP(N)	Pointer	Pointer to the DE of the last loop of the face

Additional pointers as required (see Section 2.2.4.5.2).

**4.147 Shell Entity (Type 514)‡**

‡The Shell Entity has not been tested. See Section 1.9.

ECO627  
ECO630

The shell is represented as a set of edge connected, oriented uses of faces (face-uses). The normal of the shell is in the same direction as the normal of its face-uses. The normal of the face-use is assumed to be in the direction of the normal of the underlying surface of the face unless the face-use orientation indicates it needs to be reversed. The faces used by the shell are connected to each other only via edges.

For the Shell Entity, the Form Numbers are as follows:

ECO630

Form	Meaning
1	Closed Shell
2	Open Shell

Each edge shall be referenced at least once, but not more than twice, by the loops of the faces of an Open Shell. Each edge shall be referenced exactly twice by the loops of the faces of a Closed Shell. Forms 1 and 2 of the Shell Entity may exist independently. All Face Entities (Type 510) referenced by these forms of the shell shall be Form 1 and have underlying surface geometry.

- The shell shall be an orientable surface with the same orientation maintained.
- The shell shall contain at least one use of a face.
- Faces used by the shell shall not intersect themselves or each other, except at their edges.
- Edges used by the shell shall not intersect except at their vertices.

ECO630

For additional details on the structure of the Shell Entity, and its relations to other topological entities, see the discussion of the Manifold Solid B-Rep object Entity (Type 186) (Section 4.49).

**Closed Shell Entity (Type 514, Form 1)** Form 1 of the Shell Entity is the Closed Shell Entity. A closed shell is a connected entity of dimensionality 2 which divides  $R^3$  into two arcwise-connected, open subsets (parts), one of which is finite. The inside of the shell is defined to be the finite region. If this form of the Shell Entity is referenced by a Manifold Solid B-Rep Object Entity (Type 186) (MSBO), it shall be physically dependent on its parent entity, the MSBO.

ECO630

**Open Shell Entity (Type 514, Form 2)** Form 2 of the Shell Entity is the Open Shell Entity.

The open shell is a set of faces which form a connected, orientable manifold with boundary which does not separate space. This form of the shell shall not be pointed to by an MSBO (Type 186).

**Directory Entry**

(1) Entity Type Number 514	(2) Parameter Data ⇒	(3) Structure $< n.a. >$	(4) Line Font Pattern $< n.a. >$	(5) Level #,⇒	(6) View $< n.a. >$	(7) Xformation Matrix $< n.a. >$	(8) Label Display 0,⇒	(9) Status Number ?????????	(10) Sequence Number D #
(11) Entity Type Number 514	(12) Line Weight $< n.a. >$	(13) Color Number $< n.a. >$	(14) Parameter Line Count #	(15) Form Number 1-2	(16) Reserved	(17) Reserved	(18) Entity Label #	(19) Entity Subscript #	(20) Sequence Number D # + 1

**Parameter Data**

ECO650

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	N	Integer	Number of faces ( $N > 0$ )
2	FACE(1)	Pointer	Pointer to the DE of the first face
3	OF(1)	Logical	Orientation flag of first face with respect to the direction of the underlying surface (True = agrees)
:	:	:	
$2^*N$	FACE(N)	Pointer	Pointer to the DE of the last face
$1+2^*N$	OF(N)	Logical	Orientation flag of last face

Additional pointers as required (see Section 2.2.4.5.2).

## Appendix A. Part File Examples

This appendix contains three sample parts encoded in the ASCII Form. These files are included to provide guidance in the usage of this Specification and, as such, they do not represent all design application uses. The files are a two-dimensional application using structure entities, a two-dimensional drawing of a mechanical part with dimensioning, and a three-dimensional part with two-dimensional drawing views defined.

Example file 1 is an integrated circuit (IC) cell. The IC application was selected because of the predominance of two-dimensional geometry used in electrical designs. The geometry used in the cell in [Figure A1](#) consists of Simple Closed Planar Curve Entities ([Type 106, Form 63](#)), linear path entities and the Line Widening Property Entity ([Type 406, Form 5](#)). The structure entities are nested subfigures using a Network Subfigure Definition Entity ([Type 320](#)) and array subfigure instance entities. A Connect Point Entity ([Type 132](#)) is included to identify the signal port. The geometry is on five different levels, each representing a process mask. The entity label field of each Directory Entry record contains (optional) text included to describe the entity's use. The entities in this file would be typical of those used in an IC application to transfer either cell libraries or a complete design between design systems. The file of a design prepared for pattern generation, with subfigures resolved and the geometry fractured, would use the Flash Entity ([Type 125](#)) exclusively. The cell file was adapted from a cell library in [HON80] with kind permission from the author.

Example file 2 is a two-dimensional drawing of a mechanical part containing geometry entities and annotation entities typically found on engineering drawings. Included as geometry are points, lines, circular arcs and conics. For annotation, the file includes linear dimensions, angular dimensions, radius dimensions, ordinate dimensions, a general label and general notes. [Figure A2](#) shows the mechanical part, which was used during one of the early public demonstrations of inter-system data exchange.

Example file 3 is included to show the use of View Entities and Drawing Entities in conjunction with a three-dimensional part model to convey a drawing to the receiving system. [Figure A3](#) shows the example drawing. In this way, model geometry and viewing parameters are logically separate. A three-dimensional model, as well as the drawing, is received enabling additional views to be created if necessary, and changes to the part model are to be reflected in all views.

## A1. ELECTRICAL PART EXAMPLE

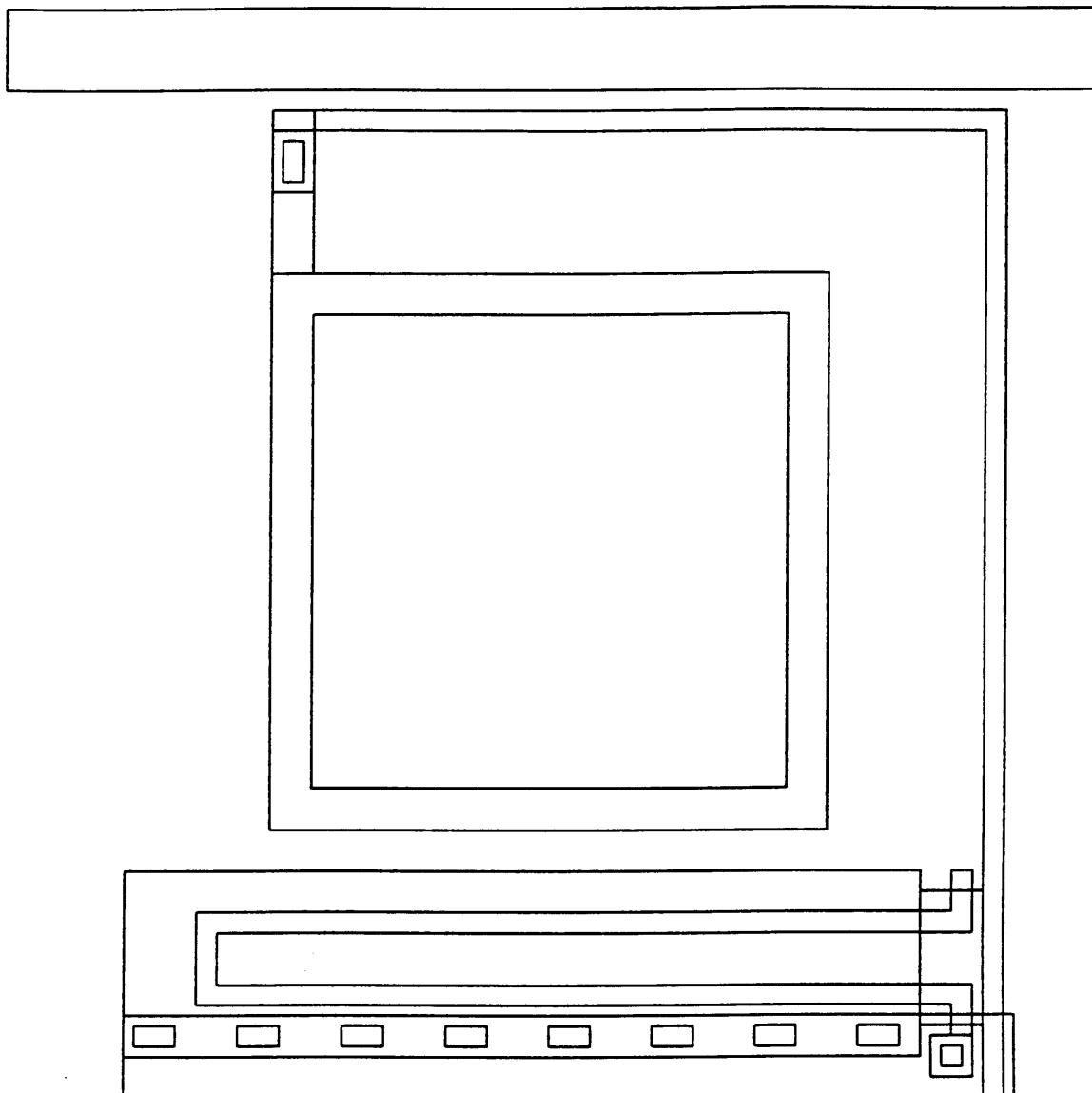


Figure A1. Electrical Part Example

## A1. ELECTRICAL PART EXAMPLE

### Example 1 Electrical Part

INTEGRATED CIRCUIT SEMICUSTOM CELL (ONE PART OF A LIBRARY FILE)	S	1
USED IN APPENDIX A OF IGES VERSION 3.0 AND MODIFIED FOR VERSION 4.0	S	2
1H,,1H;,10H5MICRONLIB,5HPADIN,9HEXAMPLE 1,4HHAND,16,38,06,38,13,	G	1
10HIC.LIBRARY,1.0,9,2HUM,1,,13H900729.231212,0.01,265.0,	G	2
25HIGES RFC Review Committee,8HIPO/NIST,6,0;	G	3
308 01 1 0 0 00020201D 01		
308 0 1 SUBFIG1 D 02		
106 02 1 1 0 00020200D 03		
106 0 4 1 63 VDDPORT D 04		
106 03 1 1 0 00020200D 05		
106 0 4 1 63 GNDPORT D 06		
106 04 1 1 0 00020200D 07		
106 0 4 1 63 BONDPAD D 08		
106 05 1 7 0 00020200D 09		
106 0 5 1 63 GLASSBOX D 10		
320 06 1 0 0 00010201D 11		
320 0 1 CELLFIG D 12		
408 07 1 0 0 00030200D 13		
408 0 1 INST1 D 14		
106 08 1 3 0 00020200D 15		
106 0 3 2 63 ACTBOX D 16		
106 10 1 3 0 00020200D 17		
106 0 3 1 63 ACTBOX D 18		
106 11 1 3 0 00020200D 19		
106 0 3 1 11 ACTSTG D 20		
106 12 1 3 0 00020200D 21		
106 0 3 2 63 ACTBOX D 22		
132 14 1 3 0 00020400D 23		
132 0 1 SIGPORT D 24		
106 15 1 6 0 00020200D 25		
106 0 8 2 63 CUT D 26		
106 17 1 6 0 00020200D 27		
106 0 8 2 63 CUT D 28		
308 19 1 0 0 00020201D 29		
308 0 1 SUBFIG2 D 30		
106 20 1 6 0 00030200D 31		
106 0 8 1 63 CUTDEF D 32		
412 21 1 0 0 00030201D 33		
412 0 1 CUTARR D 34		
106 22 1 2 0 00020200D 35		
106 0 2 2 11 GATESTG D 36		
106 24 1 2 0 00020200D 37		
106 0 2 2 63 GATEBOX D 38		
106 26 1 1 0 00020200D 39		
106 0 4 1 63 GATEBOX D 40		
406 27 1 0 0 00010200D 41		
406 0 1 5 LINWIDTH D 42		

## A1. ELECTRICAL PART EXAMPLE

308,0,6HPADBLK,4,03,05,07,09;	01P	01
106,1,5,0.,0.,0.,265.,0.,265.,-20.,0.,-20.,0.,0.;	03P	02
106,1,5,0.,30.,-245.,245.,-245.,245.,-265.,30.,-265.,30.,-245.;	05P	03
106,1,5,0.,65.,-65.,200.,-65.,200.,-200.,65.,-200.,65.,-65.;	07P	04
106,1,5,0.,75.,-75.,190.,-75.,190.,-190.,75.,-190.,75.,-75.;	09P	05
320,1,5HPADIN,11,13,15,17,19,21,25,27,33,35,37,39,2,,,1,23;	11P	06
408,01,0.,0.,0.;	13P	07
106,1,5,0.,30.,-210.,222.5,-210.,222.5,-255.,30.,-255.,30.,	15P	08
-210.;	15P	09
106,1,5,0.,65.,-25.,75.,-25.,75.,-45.,65.,-45.,65.,-25.;	17P	10
106,1,3,0.,77.5,-27.5,240.,-27.5,240.,-262.5,0,1,41;	19P	11
106,1,5,0.,222.5,-215.,237.5,-215.,237.5,-247.5,222.5,-247.5,	21P	12
222.5,-215. ;	21P	13
132,240.,-265.,0,,,2,1,,,,,01,1,1,11;	23P	14
106,1,5,0.,67.5,-32.5,72.5,-32.5,72.5,-42.5,67.5,-42.5,67.5,	25P	15
-32.5;	25P	16
106,1,5,0.,227.5,-252.5,232.5,-252.5,232.5,-257.5,227.5,-257.5,	27P	17
227.5,-252.5;	27P	18
308,0,7HCONTACT,1,31;	29P	19
106,1,5,0.,-5.,2.5,5.,2.5,5.,-2.5,-5.,-2.5,-5.,2.5;	31P	20
412,29,1.0,37.5,-250.,0.,8,1,25.,0.,0.,0;	33P	21
106,1,6,0.,232.5,-212.5,232.5,-222.5,50.,-222.5,50.,-240.,	35P	22
232.5,-240.,232.5,-247.5,0,1,41;	35P	23
106,1,5,0.,225.,-250.,235.,-250.,235.,-260.,225.,-260.,225.,	37P	24
-250. ;	37P	25
106,1,5,0.,65.,-30.,75.,-30.,75.,-65.,65.,-65.,65.,-30.;	39P	26
406,5,5,0,1,1,0,0;	41P	27
S        2G        3D        42P        27	T	1

## A2. MECHANICAL PART EXAMPLE

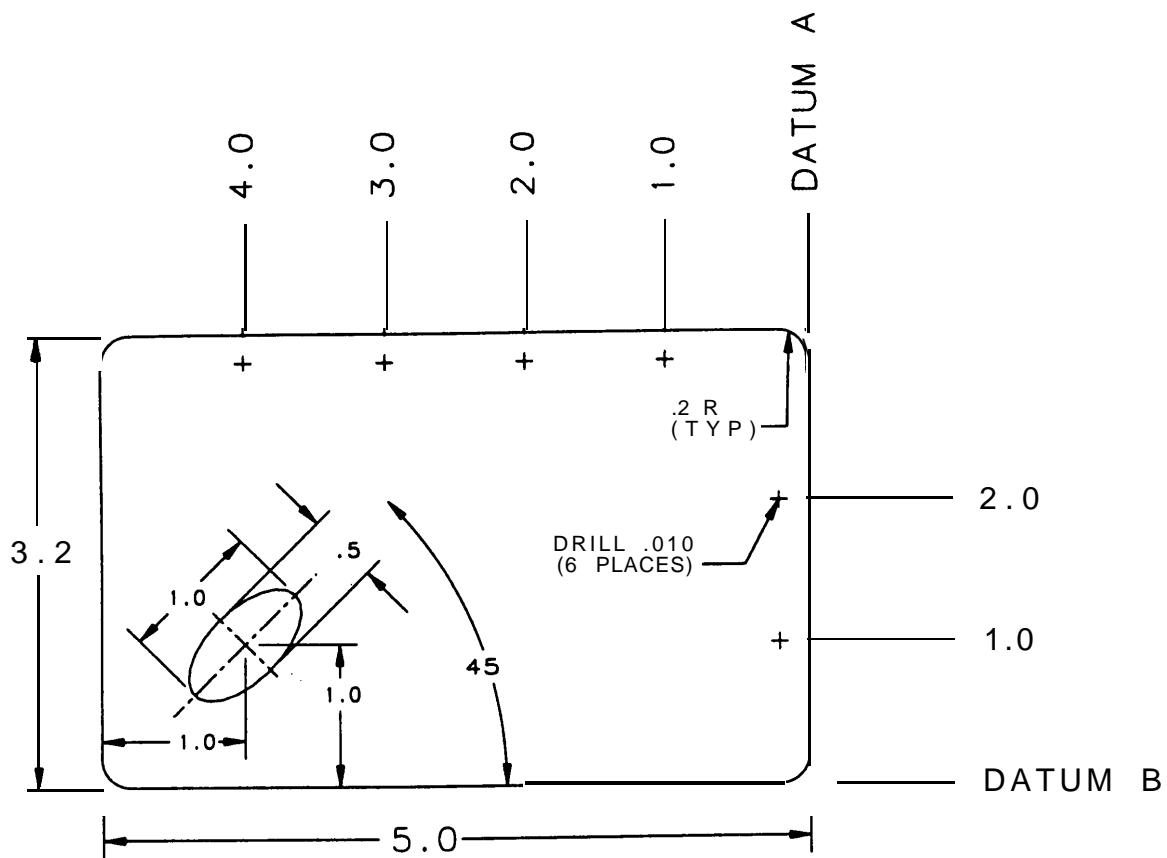


Figure A2. Mechanical Part Example

## A2. MECHANICAL PART EXAMPLE

### Example 2 Mechanical Part

PLATE.001	SAMPLE MECHANICAL PART WITH ANNOTATION	S	1		
USED AT AUTOFAC - OCTOBER 1982 AND IN VERSION 3.0 APPENDIX A			S	2	
		S	3		
ENTITY CONTENT: POINT, LINE, ARC and CONIC			S	4	
LINEAR, ANGULAR, RADIUS, POINT and ORDINATE DIMENSION			S	5	
GENERAL NOTE, GENERAL LABEL			S	6	
		S	7		
,,8HPANEL123,10HPANEL.IGES,4HEX 2,4HHAND,16,38,7,38,14,8HPANEL123,1.0,		G	1		
1,4HINCH,1,0.028,13H900729.231652,0.0005,100.0,		G	2		
25HIGES RFC Review Committee,8HIPO/NIST,6,0;			G	3	
124	1	1	0	OD	1
124	0	0	2	OD	2
212	3	1	1	5	0
212	0	0	2	0	
214	5	1	1	5	0
214	0	0	1	2	
210	6	1	1	5	0
210	0	0	1	0	
110	7	1	1	1	0
110	0	0	1	0	
110	8	1	1	1	0
110	0	0	1	0	
110	9	1	1	1	0
110	0	0	1	0	
110	10	1	1	1	0
110	0	0	1	0	
100	11	1	1	1	0
100	0	0	1	0	
100	12	1	1	1	0
100	0	0	1	0	
100	13	1	1	1	0
100	0	0	1	0	
100	14	1	1	1	0
100	0	0	1	0	
116	15	1	1	2	0
116	0	0	1	0	
116	16	1	1	2	0
116	0	0	1	0	
116	17	1	1	2	0
116	0	0	1	0	
116	18	1	1	2	0
116	0	0	1	0	
104	19	1	1	3	0
104	0	0	2	1	
116	21	1	1	2	0
116	0	0	1	0	
116	22	1	1	2	0
116	0	0	1	0	

## A2. MECHANICAL PART EXAMPLE

212	23	1	1	4	0	0	0	10100D	39
212	0	0	1	0				OD	40
214	24	1	1	4	0	0	0	10100D	41
214	0	0	1	2				OD	42
214	25	1	1	4	0	0	0	10100D	43
214	0	0	1	2				OD	44
106	26	1	1	4	0	0	0	10100D	45
106	0	0	1	40				OD	46
106	27	1	1	4	0	0	0	10100D	47
106	0	0	1	40				OD	48
216	28	1	1	4	0	0	0	101D	49
216	0	0	1	0				OD	50
212	29	1	1	4	0	0	0	10100D	51
212	0	0	1	0				OD	52
214	30	1	1	4	0	0	0	10100D	53
214	0	0	1	2				OD	54
214	31	1	1	4	0	0	0	10100D	55
214	0	0	1	2				OD	56
106	32	1	1	4	0	0	0	10100D	57
106	0	0	1	40				OD	58
106	33	1	1	4	0	0	0	10100D	59
106	0	0	1	40				OD	60
216	34	1	1	4	0	0	0	101D	61
216	0	0	1	0				OD	62
212	35	1	1	5	0	0	0	10100D	63
212	0	0	1	0				OD	64
106	36	1	1	5	0	0	0	10100D	65
106	0	0	1	40				OD	66
218	37	1	1	5	0	0	0	101D	67
218	0	0	1	0				OD	68
212	38	1	1	5	0	0	0	10100D	69
212	0	0	1	0				OD	70
106	39	1	1	5	0	0	0	10100D	71
106	0	0	1	40				OD	72
218	40	1	1	5	0	0	0	101D	73
218	0	0	1	0				OD	74
212	41	1	1	5	0	0	0	10100D	75
212	0	0	1	0				OD	76
106	42	1	1	5	0	0	0	10100D	77
106	0	0	1	40				OD	78
218	43	1	1	5	0	0	0	101D	79
218	0	0	1	0				OD	80
212	44	1	1	5	0	0	0	10100D	81
212	0	0	1	0				OD	82
106	45	1	1	5	0	0	0	10100D	83
106	0	0	1	40				OD	84
218	46	1	1	5	0	0	0	101D	85

## A2. MECHANICAL PART EXAMPLE

218	0	0	1	0				OD	86
212	47	1	1	5	0	0	0	10100D	87
212	0	0	1	0				OD	88
106	48	1	1	5	0	0	0	10100D	89
106	0	0	1	40				OD	90
218	49	1	1	5	0	0	0	101D	91
218	0	0	1	0				OD	92
212	50	1	1	5	0	0	0	10100D	93
212	0	0	1	0				OD	94
106	51	1	1	5	0	0	0	10100D	95
106	0	0	1	40				OD	96
218	52	1	1	5	0	0	0	101D	97
218	0	0	1	0				OD	98
212	53	1	1	5	0	0	0	10100D	99
212	0	0	2	0				OD	100
106	55	1	1	5	0	0	0	10100D	101
106	0	0	1	40				OD	102
218	56	1	1	5	0	0	0	101D	103
218	0	0	1	0				OD	104
212	57	1	1	5	0	0	0	10100D	105
212	0	0	2	0				OD	106
106	59	1	1	5	0	0	0	10100D	107
106	0	0	1	40				OD	108
218	60	1	1	5	0	0	0	101D	109
218	0	0	1	0				OD	110
212	61	1	1	5	0	0	0	10100D	111
212	0	0	2	0				OD	112
214	63	1	1	5	0	0	0	10100D	113
214	0	0	1	2				OD	114
222	64	1	1	5	0	0	0	101D	115
222	0	0	1	0				OD	116
212	65	1	1	6	0	0	0	10100D	117
212	0	0	1	0				OD	118
214	66	1	1	6	0	0	0	10100D	119
214	0	0	1	2				OD	120
214	67	1	1	6	0	0	0	10100D	121
214	0	0	1	2				OD	122
106	68	1	1	6	0	0	0	1010100D	123
106	0	0	1	40				OD	124
106	69	1	1	6	0	0	0	10100D	125
106	0	0	1	40				OD	126
216	70	1	1	6	0	0	0	101D	127
216	0	0	1	0				OD	128
212	71	1	1	6	0	0	0	10100D	129
212	0	0	1	0				OD	130
214	72	1	1	6	0	0	0	10100D	131
214	0	0	1	2				OD	132

## A2. MECHANICAL PART EXAMPLE

214	73	1	1	6	0	0	0	10100D	133
214	0	0	1	2				0D	134
106	74	1	1	6	0	0	0	10100D	135
106	0	0	1	40				0D	136
106	75	1	1	6	0	0	0	1010100D	137
106	0	0	1	40				0D	138
216	76	1	1	6	0	0	0	101D	139
216	0	0	1	0				0D	140
212	77	1	1	6	0	0	0	10100D	141
212	0	0	1	0				0D	142
214	78	1	1	6	0	0	0	10100D	143
214	0	0	1	2				0D	144
214	79	1	1	6	0	0	0	10100D	145
214	0	0	1	2				0D	146
106	80	1	1	6	0	0	0	10100D	147
106	0	0	1	40				0D	148
106	81	1	1	6	0	0	0	10100D	149
106	0	0	1	40				0D	150
216	82	1	1	6	0	0	0	101D	151
216	0	0	1	0				0D	152
212	83	1	1	6	0	0	0	10100D	153
212	0	0	1	0				0D	154
214	84	1	1	6	0	0	0	10100D	155
214	0	0	1	2				0D	156
214	85	1	1	6	0	0	0	10100D	157
214	0	0	1	2				0D	158
106	86	1	1	6	0	0	0	10100D	159
106	0	0	1	40				0D	160
106	87	1	1	6	0	0	0	10100D	161
106	0	0	1	40				0D	162
216	88	1	1	6	0	0	0	101D	163
216	0	0	1	0				0D	164
110	89	1	4	6	0	0	0	100D	165
110	0	0	1	0				0D	166
110	90	1	4	6	0	0	0	100D	167
110	0	0	1	0				0D	168
212	91	1	1	6	0	0	0	10100D	169
212	0	0	1	0				0D	170
214	92	1	1	6	0	0	0	10100D	171
214	0	0	1	2				0D	172
214	93	1	1	6	0	0	0	10100D	173
214	0	0	1	2				0D	174
106	94	1	1	6	0	0	0	1010100D	175
106	0	0	1	40				0D	176
106	95	1	1	6	0	0	0	1010100D	177
106	0	0	1	40				0D	178
202	96	1	1	6	0	0	0	101D	179

## A2. MECHANICAL PART EXAMPLE

202	0	0	1	0	OD	180
124,0.70710678,-0.70710678,0.0,1.0,0.70710678,0.70710678,0.0,					1P	1
1.0,0.0,0.0,1.0,0.0,0,0;					1P	2
212,2,10,0.98,0.1,1,1.571,0.0,0,0,3.21,1.656,0.0,10HDRILL .010,					3P	3
10,1.02,0.1,1,1.571,0.0,0,0,3.210,1.506,0.0,10H(6 PLACES),0,0;					3P	4
214,2,0.150,0.050,0.0,4.800,2.000,4.562,1.546,4.262,1.546,0,0;					5P	5
210,3,1,5,0,0;					7P	6
110,0.0,0.200,0.0,0.0,3.000,0.0,0,0;					9P	7
110,0.200,0.0,0.0,4.800,0.0,0.0,0,0;					11P	8
110,5.000,0.200,0.0,5.000,3.000,0.0,0,0;					13P	9
110,0.200,3.200,0.0,4.800,3.200,0.0,0,0;					15P	10
100,0.0,4.800,3.000,5.000,3.000,4.800,3.200,0,0;					17P	11
100,0.0,0.200,3.000,0.200,3.200,0.0,3.000,0,0;					19P	12
100,0.0,0.200,0.200,0.0,0.200,0.200,0.00,0,0;					21P	13
100,0.0,4.800,0.200,4.800,0.0,5.000,0.200,0,0;					23P	14
116,4.000,3.000,0.0,0,0,0;					25P	15
116,3.000,3.000,0.0,0,0,0;					27P	16
116,2.000,3.000,0.0,0,0,0;					29P	17
116,1.000,3.000,0.0,0,0,0;					31P	18
104,4.000,0.0,16.000,0.0,0.0,-1.000,0.0,0.500,0.0,0.500,0.0,					33P	19
0,0;					33P	20
116,4.800,1.000,0.0,0,0,0;					35P	21
116,4.800,2.000,0.0,0,0,0;					37P	22
212,1,3,0.421,0.156,1,1.571,0.0,0,0,-0.639,1.614,0.0,3H3.2,0,0;					39P	23
214,1,0.150,0.050,0.0,-0.454,3.200,-0.454,1.870,0,0;					41P	24
214,1,0.150,0.050,0.0,-0.454,0.0,-0.454,1.514,0,0;					43P	25
106,1,3,0.0,0.0,3.200,-0.094,3.200,-0.579,3.200,0,0;					45P	26
106,1,3,0.0,0.0,0.0,-0.094,0.0,-0.579,0.0,0,0;					47P	27
216,39,41,43,45,47,0,0;					49P	28
212,1,3,0.437,0.156,1,1.571,0.0,0,0,2.032,-0.447,0.0,3H5.0,0,0;					51P	29
214,1,0.150,0.050,0.0,0.0,-0.369,1.932,-0.369,0,0;					53P	30
214,1,0.150,0.050,0.0,5.000,-0.369,2.519,-0.369,0,0;					55P	31
106,1,3,0.0,0.0,0.0,0.0,-0.094,0.0,-0.494,0,0;					57P	32
106,1,3,0.0,5.000,0.0,5.000,-0.094,5.000,-0.494,0,0;					59P	33
216,51,53,55,57,59,0,0;					61P	34
212,1,3,0.374,0.156,1,1.571,1.571,0,0,4.078,4.181,0.0,3H1.0,0,0;					63P	35
106,1,3,0.0,4.000,3.094,4.000,3.188,4.000,3.993,0,0;					65P	36
218,63,65,0,0;					67P	37
212,1,3,0.421,0.156,1,1.571,1.571,0,0,3.078,4.183,0.0,3H2.0,0,0;					69P	38
106,1,3,0.0,3.000,3.094,3.000,3.188,3.000,3.996,0,0;					71P	39
218,69,71,0,0;					73P	40
212,1,3,0.437,0.156,1,1.571,1.571,0,0,2.078,4.177,0.0,3H3.0,0,0;					75P	41
106,1,3,0.0,2.000,3.094,2.000,3.188,2.000,3.989,0,0;					77P	42
218,75,77,0,0;					79P	43
212,1,3,0.437,0.156,1,1.571,1.571,0,0,1.078,4.177,0.0,3H4.0,0,0;					81P	44
106,1,3,0.0,1.000,3.094,1.000,3.188,1.000,3.989,0,0;					83P	45
218,81,83,0,0;					85P	46

## A2. MECHANICAL PART EXAMPLE

212,1,3,0.374,0.156,1,1.571,0.0,0,0,6.211,0.922,0.0,3H1.0,0,0;	87P	47
106,1,3,0.0,4.894,1.000,4.988,1.000,6.024,1.000,0,0;	89P	48
218,87,89,0,0;	91P	49
212,1,3,0.421,0.156,1,1.571,0.0,0,0,6.211,1.922,0.0,3H2.0,0,0;	93P	50
106,1,3,0.0,4.894,2.000,4.988,2.000,6.024,2.000,0,0;	95P	51
218,93,95,0,0;	97P	52
212,1,7,1.248,0.156,1,1.571,0.,0,0,6.23,-0.078,0.0,7HDATUM B,0,	99P	53
0;	99P	54
106,1,3,0.0,5.094,0.0,5.188,0.0,6.042,0.0,0,0;	101P	55
218,99,101,0,0;	103P	56
212,1,7,1.232,0.156,1,1.571,1.571,0,0,5.078,4.193,0.,7HDATUM A,	105P	57
0,0;	105P	58
106,1,3,0.0,5.000,3.094,5.000,3.187,5.000,4.006,0,0;	107P	59
218,105,107,0,0;	109P	60
212,2,4,0.35,0.100,1,1.571,0.,0,0,4.029,2.611,0.,4H.2 R,5,0.500,	111P	61
0.100,1,1.571,0.0,0,0,4.029,2.461,0.0,5H(TYP),0,0;	111P	62
214,2,0.150,0.050,0.0,4.877,3.185,4.862,2.511,4.562,2.511,0,0;	113P	63
222,111,113,4.800,3.000,0,0;	115P	64
212,1,3,0.240,0.100,1,1.571,0.0,0,0,1.559,0.602,0.0,3H1.0,0,0,	117P	65
214,1,0.150,0.050,0.0,1.663,0.0,1.663,0.538,0,0;	119P	66
214,1,0.150,0.050,0.0,1.663,1.000,1.663,0.766,0,0;	121P	67
106,1,3,0.0,0.200,0.0,0.294,0.0,1.788,0.0,0,0;	123P	68
106,1,3,0.0,1.000,1.000,1.094,1.000,1.788,1.000,0,0;	125P	69
216,117,119,121,123,125,0,0;	127P	70
212,1,3,0.240,0.100,1,1.571,0.0,0,0,0.537,0.275,0.0,3H1.0,0,0,	129P	71
214,1,0.150,0.050,0.0,1.000,0.325,0.809,0.325,0,0;	131P	72
214,1,0.150,0.050,0.0,0.0,0.325,0.473,0.325,0,0;	133P	73
106,1,3,0.0,1.000,1.000,1.000,0.906,1.000,0.200,0,0;	135P	74
106,1,3,0.0,0.0,0.012,0.0,0.106,0.0,0.200,0,0;	137P	75
216,129,131,133,135,137,0,0;	139P	76
212,1,3,0.240,0.100,1,1.571,0.0,0,0,0.470,1.289,0.0,3H1.0,0,0,	141P	77
214,1,0.150,0.050,0.0,0.971,1.736,0.688,1.453,0,0;	143P	78
214,1,0.150,0.050,0.0,0.264,1.029,0.459,1.225,0,0;	145P	79
106,1,3,0.0,1.354,1.354,1.287,1.420,0.882,1.825,0,0;	147P	80
106,1,3,0.0,0.646,0.646,0.580,0.713,0.175,1.118,0,0;	149P	81
216,141,143,145,147,149,0,0;	151P	82
212,1,2,0.170,0.100,1,1.571,0.0,0,0,1.631,1.622,0.0,2H.5,0,0;	153P	83
214,1,0.150,0.050,0.0,1.863,1.509,2.146,1.226,0,0;	155P	84
214,1,0.150,0.050,0.0,1.509,1.863,1.226,2.146,0,0;	157P	85
106,1,3,0.0,1.177,0.823,1.243,0.890,1.951,1.598,0,0;	159P	86
106,1,3,0.0,0.823,1.177,0.890,1.243,1.598,1.951,0,0;	161P	87
216,153,155,157,159,161,0,0;	163P	88
110,0.500,0.500,0.0,1.500,1.500,0.0,0,0;	165P	89
110,1.225,0.775,0.0,0.775,1.225,0.0,0,0;	167P	90
212,1,2,0.220,0.100,1,1.571,0.0,0,0,2.566,0.786,0.0,2H45,0,0;	169P	91
214,1,0.150,0.050,0.0,2.847,0.0,2.754,0.722,0,0;	171P	92
214,1,0.150,0.050,0.0,2.013,2.013,2.683,0.951,0,0;	173P	93

## A2. MECHANICAL PART EXAMPLE

106,1,3,0.0,4.988,0.0,4.894,0.0,2.972,0.0,0,0;	175P	94
106,1,3,0.0,2.000,2.000,2.066,2.066,2.101,2.101,0,0;	177P	95
202,169,175,177,0.0,0.0,2.847,171,173,0,0;	179P	96
S 7G 3D 180P 96	T	1

### A3. DRAWING AND VIEW EXAMPLE

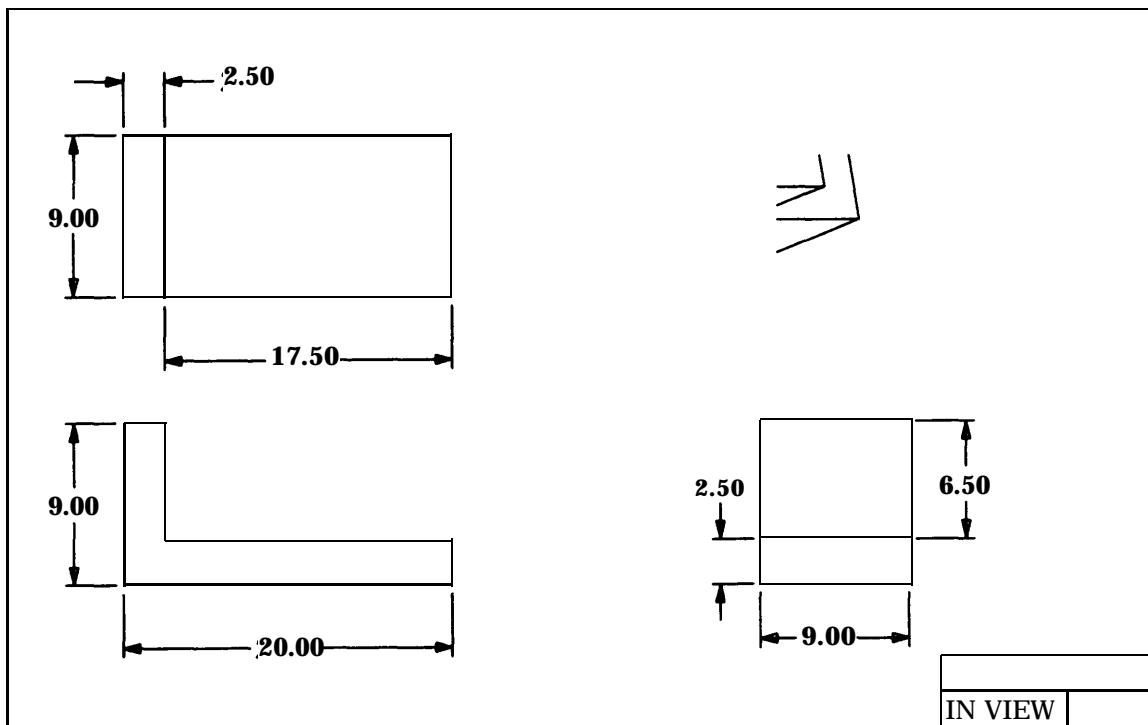


Figure A3. Drawing and View Example

### A3. DRAWING AND VIEW EXAMPLE

#### Example 3 Drawing and View

Test file of model with DRAWING (404) and VIEW (410) entities	S	1
	S	2
This file demonstrates annotation attached to the VIEWS, i.e., the dimensions entities are flagged as INDEPENDENT, and their DE field 6 points to a VIEW entity. The coordinates of the dimensions are in MODEL space, and they have a transformation matrix which is the inverse of the VIEW matrix.	S	3
	S	4
	S	5
	S	6
	S	7
	S	8
A companion file demonstrates annotation attached to the DRAWING, <i>i.e.</i> , the dimension entities are flagged as DEPENDENT, and they are pointed to by the PD of the DRAWING entity. The coordinates of the dimensions are in DRAWING space.	S	9
	S	10
	S	11
	S	12
	S	13
1H,,1H;,,8HVIEWDWG2,12HVIEWDWG2.IGS,13H<unspecified>,13H<unspecified>, G 1	G	1
32,38,6,38,15,8HVIEWDWG2,1.,1,2HIN,8,0.016,13H900729.231904,0.0001,71., G 2	G	2
25HIGES RFC Review Committee,8HIPO/NIST,6,0; G 3	G	3
406 1 0 0 0 0 0 0 0 10300D 1		
406 0 0 1 15 0 0 0 0 OD 2		
124 2 0 0 0 0 0 0 0 10300D 3		
124 0 0 2 0 0 0 0 0 OD 4		
108 4 0 0 0 0 0 0 0 10201D 5		
108 0 0 2 0 0 0 0 0 OD 6		
108 6 0 0 0 0 0 0 0 10201D 7		
108 0 0 2 0 0 0 0 0 OD 8		
108 8 0 0 0 0 0 0 0 10201D 9		
108 0 0 2 0 0 0 0 0 OD 10		
108 10 0 0 0 0 0 0 0 10201D 11		
108 0 0 2 0 0 0 0 0 OD 12		
410 12 0 0 0 0 0 3 0 20201D 13		
410 0 0 1 0 0 0 0 0 OD 14		
406 13 0 0 0 0 0 0 0 10300D 15		
406 0 0 1 15 0 0 0 0 OD 16		
124 14 0 0 0 0 0 0 0 10300D 17		
124 0 0 1 0 0 0 0 0 OD 18		
108 15 0 0 0 0 0 0 0 10201D 19		
108 0 0 1 0 0 0 0 0 OD 20		
108 16 0 0 0 0 0 0 0 10201D 21		
108 0 0 1 0 0 0 0 0 OD 22		
108 17 0 0 0 0 0 0 0 10201D 23		
108 0 0 1 0 0 0 0 0 OD 24		
108 18 0 0 0 0 0 0 0 10201D 25		
108 0 0 1 0 0 0 0 0 OD 26		
410 19 0 0 0 0 0 17 0 20201D 27		
410 0 0 1 0 0 0 0 0 OD 28		
406 20 0 0 0 0 0 0 0 10300D 29		
406 0 0 1 15 0 0 0 0 OD 30		
124 21 0 0 0 0 0 0 0 10300D 31		

### A3. DRAWING AND VIEW EXAMPLE

124	0	0	1	0				OD	32
108	22	0	0	0	0	0	0	10201D	33
108	0	0	1	0				OD	34
108	23	0	0	0	0	0	0	10201D	35
108	0	0	1	0				OD	36
108	24	0	0	0	0	0	0	10201D	37
108	0	0	1	0				OD	38
108	25	0	0	0	0	0	0	10201D	39
108	0	0	1	0				OD	40
410	26	0	0	0	0	31	0	20201D	41
410	0	0	1	0				OD	42
406	27	0	0	0	0	0	0	10300D	43
406	0	0	1	15				OD	44
108	28	0	0	0	0	0	0	10201D	45
108	0	0	1	0				OD	46
108	29	0	0	0	0	0	0	10201D	47
108	0	0	1	0				OD	48
108	30	0	0	0	0	0	0	10201D	49
108	0	0	1	0				OD	50
108	31	0	0	0	0	0	0	10201D	51
108	0	0	1	0				OD	52
410	32	0	0	0	0	0	0	20201D	53
410	0	0	1	0				OD	54
110	33	0	1	2	0	0	0	10100D	55
110	1	4	1	0				OD	56
110	34	0	1	2	0	0	0	10100D	57
110	1	4	1	0				OD	58
110	35	0	1	2	0	0	0	10100D	59
110	1	4	1	0				OD	60
110	36	0	1	2	0	0	0	10100D	61
110	1	4	1	0				OD	62
110	37	0	1	2	0	0	0	10100D	63
110	1	4	1	0				OD	64
110	38	0	1	4	0	0	0	10100D	65
110	1	4	1	0				OD	66
110	39	0	1	4	0	0	0	10100D	67
110	1	4	1	0				OD	68
110	40	0	1	4	0	0	0	10100D	69
110	1	4	1	0				OD	70
110	41	0	1	4	0	0	0	10100D	71
110	1	4	1	0				OD	72
406	42	0	1	0	0	0	0	10300D	73
406	0	0	1	15				OD	74
404	43	0	1	0	0	0	0	201D	75
404	0	0	2	0				OD	76
110	45	0	1	2	0	0	0	OD	77
110	1	2	1	0				OD	78

### A3. DRAWING AND VIEW EXAMPLE

110	46	0	1	2	0	0	0	OD	79
110	1	2	1	0				OD	80
110	47	0	1	2	0	0	0	OD	81
110	1	2	1	0				OD	82
110	48	0	1	2	0	0	0	OD	83
110	1	2	1	0				OD	84
110	49	0	1	2	0	0	0	OD	85
110	1	2	1	0				OD	86
110	50	0	1	2	0	0	0	OD	87
110	1	2	1	0				OD	88
110	51	0	1	2	0	0	0	OD	89
110	1	2	1	0				OD	90
110	52	0	1	2	0	0	0	OD	91
110	1	2	1	0				OD	92
110	53	0	1	2	0	0	0	OD	93
110	1	2	1	0				OD	94
110	54	0	1	2	0	0	0	OD	95
110	1	2	1	0				OD	96
110	55	0	1	2	0	0	0	OD	97
110	1	2	1	0				OD	98
110	56	0	1	2	0	0	0	OD	99
110	1	2	1	0				OD	100
110	57	0	1	2	0	0	0	OD	101
110	1	2	1	0				OD	102
110	58	0	1	2	0	0	0	OD	103
110	1	2	1	0				OD	104
110	59	0	1	2	0	0	0	OD	105
110	1	2	1	0				OD	106
110	60	0	1	2	0	0	0	OD	107
110	1	2	1	0				OD	108
110	61	0	1	2	0	0	0	OD	109
110	1	2	1	0				OD	110
110	62	0	1	2	0	0	0	OD	111
110	1	2	1	0				OD	112
106	63	0	1	2	53	0	0	10101D	113
106	1	3	1	40				OD	114
214	64	0	1	2	53	0	0	10101D	115
214	1	3	1	2				OD	116
212	65	0	1	2	53	0	0	10101D	117
212	1	3	1	0				OD	118
214	66	0	1	2	53	0	0	10101D	119
214	1	3	1	2				OD	120
106	67	0	1	2	53	0	0	10101D	121
106	1	3	1	40				OD	122
216	68	0	1	2	53	0	0	100D	123
216	1	3	1	0				OD	124
106	69	0	1	2	27	0	0	10101D	125

## A. DRAWING AND VIEW EXAMPLE

106	1	3	1	40				OD	126
214	70	0	1	2	27	0	0	10101D	127
214	1	3	1	2				OD	128
212	71	0	1	2	27	0	0	10101D	129
212	1	3	1	0				OD	130
214	72	0	1	2	27	0	0	10101D	131
214	1	3	1	2				OD	132
106	73	0	1	2	27	0	0	10101D	133
106	1	3	1	40				OD	134
216	74	0	1	2	27	211	0	100D	135
216	1	3	1	0				OD	136
106	75	0	1	2	41	0	0	10101D	137
106	1	3	1	40				OD	138
214	76	0	1	2	41	0	0	10101D	139
214	1	3	1	2				OD	140
212	77	0	1	2	41	0	0	10101D	141
212	1	3	1	0				OD	142
214	78	0	1	2	41	0	0	10101D	143
214	1	3	1	2				OD	144
106	79	0	1	2	41	0	0	10101D	145
106	1	3	1	40				OD	146
216	80	0	1	2	41	213	0	100D	147
216	1	3	1	0				OD	148
106	81	0	1	2	41	0	0	10101D	149
106	1	3	1	40				OD	150
214	82	0	1	2	41	0	0	10101D	151
214	1	3	1	2				OD	152
212	83	0	1	2	41	0	0	10101D	153
212	1	3	1	0				OD	154
214	84	0	1	2	41	0	0	10101D	155
214	1	3	1	2				OD	156
106	85	0	1	2	41	0	0	10101D	157
106	1	3	1	40				OD	158
216	86	0	1	2	41	213	0	100D	159
216	1	3	1	0				OD	160
106	87	0	1	2	53	0	0	10101D	161
106	1	3	1	40				OD	162
214	88	0	1	2	53	0	0	10101D	163
214	1	3	1	2				OD	164
212	89	0	1	2	53	0	0	10101D	165
212	1	3	1	0				OD	166
214	90	0	1	2	53	0	0	10101D	167
214	1	3	1	2				OD	168
106	91	0	1	2	53	0	0	10101D	169
106	1	3	1	40				OD	170
216	92	0	1	2	53	0	0	100D	171
216	1	3	1	0				OD	172

## A. DRAWING AND VIEW EXAMPLE

106	93	0	1	2	41	0	0	10101D	173
106	1	3	1	40				OD	174
214	94	0	1	2	41	0	0	10101D	175
214	1	3	1	2				OD	176
212	95	0	1	2	41	0	0	10101D	177
212	1	3	1	0				OD	178
214	96	0	1	2	41	0	0	10101D	179
214	1	3	1	2				OD	180
106	97	0	1	2	41	0	0	10101D	181
106	1	3	1	40				OD	182
216	98	0	1	2	41	213	0	100D	183
216	1	3	1	0				OD	184
106	99	0	1	2	27	0	0	10101D	185
106	1	3	1	40				OD	186
214	100	0	1	2	27	0	0	10101D	187
214	1	3	1	2				OD	188
212	101	0	1	2	27	0	0	10101D	189
212	1	3	1	0				OD	190
214	102	0	1	2	27	0	0	10101D	191
214	1	3	1	2				OD	192
106	103	0	1	2	27	0	0	10101D	193
106	1	3	1	40				OD	194
216	104	0	1	2	27	211	0	100D	195
216	1	3	1	0				OD	196
106	105	0	1	2	27	0	0	10101D	197
106	1	3	1	40				OD	198
214	106	0	1	2	27	0	0	10101D	199
214	1	3	1	2				OD	200
212	107	0	1	2	27	0	0	10101D	201
212	1	3	1	0				OD	202
214	108	0	1	2	27	0	0	10101D	203
214	1	3	1	2				OD	204
106	109	0	1	2	27	0	0	10101D	205
106	1	3	1	40				OD	206
216	110	0	1	2	27	211	0	100D	207
216	1	3	1	0				OD	208
212	111	0	1	2	0	0	0	10101D	209
212	1	3	1	0				OD	210
124	112	0	0	0	0	0	0	10300D	211
124	0	0	1	0				OD	212
124	113	0	0	0	0	0	0	10300D	213
124	0	0	1	0				OD	214
406	114	0	1	0	0	0	0	10300D	215
406	0	0	1	16				OD	216
406	115	0	1	0	0	0	0	10300D	217
406	0	0	1	17				OD	218

406,1,7HCLIPPED;

1P 1

## A. DRAWING AND VIEW EXAMPLE

124,-0.67499,-0.171,0.71774,0.,-0.24401,0.96977,0.00157,0.,	3P	2
-0.69631,-0.17408,-0.69631,0.;	3P	3
108,0.67499,0.171,-0.71774,5.0055,0,6.390347,2.455541,-0.379235,	5P	4
0.;	5P	5
108,-0.24401,0.96977,0.00157,3.55308,0,3.025032,4.420965,	7P	6
2.494731,0.;	7P	7
108,-0.67499,-0.171,0.71774,2.99094,0,0.992841,1.088152,	9P	8
5.360119,0.;	9P	9
108,0.24401,-0.96977,-0.00157,1.9103,0,4.358156,-0.877273,	11P	10
2.486153,0.;	11P	11
410,4,1.,5,7,9,11,0,0,0,1,1;	13P	12
406,1,5HRIGHT;	15P	13
124,0.,0.,-1.,0.,0.,1.,0.,0.,1.,0.,0.,0.;	17P	14
108,0.,0.,1.,36.,0,0.,0.,36.,0.;	19P	15
108,0.,1.,0.,24.59627,0,0.,24.59627,0.,0.;	21P	16
108,0.,0.,-1.,36.,0,0.,0.,-36.,0.;	23P	17
108,0.,-1.,0.,24.59627,0,0.,-24.59627,0.,0.;	25P	18
410,3,1.,19,21,23,25,0,0,0,1,15;	27P	19
406,1,3HTOP;	29P	20
124,1.,0.,0.,0.,0.,0.,-1.,0.,0.,1.,0.,0.;	31P	21
108,-1.,0.,0.,36.,0,-36.,0.,0.,0.;	33P	22
108,0.,0.,-1.,24.59627,0,0.,0.,-24.59627,0.;	35P	23
108,1.,0.,0.,36.,0,36.,0.,0.,0.;	37P	24
108,0.,0.,1.,24.59627,0,0.,0.,24.59627,0.;	39P	25
410,2,1.,33,35,37,39,0,0,0,1,29;	41P	26
406,1,5HFRONT;	43P	27
108,-1.,0.,0.,36.,0,-36.,0.,0.,0.;	45P	28
108,0.,1.,0.,24.59627,0,0.,24.59627,0.,0.;	47P	29
108,1.,0.,0.,36.,0,36.,0.,0.,0.;	49P	30
108,0.,-1.,0.,24.59627,0,0.,-24.59627,0.,0.;	51P	31
410,1,1.,45,47,49,51,0,0,0,1,43;	53P	32
110,67.,0.,0.,67.,2.,0.;	55P	33
110,60.,4.,0.,60.,2.,0.;	57P	34
110,57.,4.,0.,57.,0.,0.;	59P	35
110,70.,4.,0.,57.,4.,0.;	61P	36
110,70.,2.,0.,57.,2.,0.;	63P	37
110,0.,40.,0.,0.,0.,0.;	65P	38
110,70.,40.,0.,0.,40.,0.;	67P	39
110,70.,0.,0.,70.,40.,0.;	69P	40
110,0.,0.,0.,70.,0.,0.;	71P	41
406,1,7HDRAWING;	73P	42
404,4,13,51.,29.,27,46.,8.,41,7.,24.,53,7.,8.,10,55,57,59,61,63,	75P	43
65,67,69,71,209,0,3,73,215,217;	75P	44
110,0.,9.,0.,0.,9.,-9.;	77P	45
110,2.5,9.,0.,2.5,9.,-9.;	79P	46
110,2.5,2.5,0.,2.5,2.5,-9.;	81P	47
110,20.,2.5,0.,20.,2.5,-9.;	83P	48

## A. DRAWING AND VIEW EXAMPLE

110,20.,0.,0.,20.,0.,-9.;	85P	49
110,0.,0.,0.,0.,0.,-9.;	87P	50
110,0.,9.,-9.,0.,0.,-9.;	89P	51
110,2.5,9.,-9.,0.,9.,-9.;	91P	52
110,2.5,2.5,-9.,2.5,9.,-9.;	93P	53
110,20.,2.5,-9.,2.5,2.5,-9.;	95P	54
110,20.,0.,-9.,20.,2.5,-9.;	97P	55
110,0.,0.,-9.,20.,0.,-9.;	99P	56
110,0.,9.,0.,0.,0.,0.;	101P	57
110,2.5,9.,0.,0.,9.,0.;	103P	58
110,2.5,2.5,0.,2.5,9.,0.;	105P	59
110,20.,2.5,0.,2.5,2.5,0.;	107P	60
110,20.,0.,0.,20.,2.5,0.;	109P	61
110,0.,0.,0.,20.,0.,0.;	111P	62
106,1,3,0.,0.,0.,-0.5,0.,-4.;	113P	63
214,1,1.,0.5,0.,0.,-3.5,7.625,-3.5;	115P	64
212,1,5,3.75,1.,1,1.5707963267949,0.,0,0,8.,-4.,0.,5H20.00;	117P	65
214,1,1.,0.5,0.,20.,-3.5,12.125,-3.5;	119P	66
106,1,3,0.,20.,0.,20.,-0.5,20.,-4.;	121P	67
216,117,115,119,113,121;	123P	68
106,1,3,0.,0.,0.,-0.5,0.,-3.5;	125P	69
214,1,1.,0.5,0.,0.,-3.,2.125,-3.;	127P	70
212,1,4,3.,1.,1,1.5707963267949,0.,0,0,2.5,-3.5,0.,4H9.00;	129P	71
214,1,1.,0.5,0.,9.,-3.,5.875,-3.;	131P	72
106,1,3,0.,9.,0.,9.,-0.5,9.,-3.5;	133P	73
216,129,127,131,125,133;	135P	74
106,1,3,0.,2.5,0.,2.5,-0.5,2.5,-4.;	137P	75
214,1,1.,0.5,0.,2.5,-3.5,8.625,-3.5;	139P	76
212,1,5,3.75,1.,1,1.5707963267949,0.,0,0,9.,-4.,0.,5H17.50;	141P	77
214,1,1.,0.5,0.,20.,-3.5,13.125,-3.5;	143P	78
106,1,3,0.,20.,0.,20.,-0.5,20.,-4.;	145P	79
216,141,139,143,137,145;	147P	80
106,1,3,0.,0.,9.,0.,9.5,0.,12.5;	149P	81
214,1,1.,0.5,0.,0.,12.,-3.,12.;	151P	82
212,1,4,3.,1.,1,1.5707963267949,0.,0,0,6.,11.5,0.,4H2.50;	153P	83
214,1,1.,0.5,0.,2.5,12.,5.5,12.;	155P	84
106,1,3,0.,2.5,9.,2.5,9.5,2.5,12.5;	157P	85
216,153,151,155,149,157;	159P	86
106,1,3,0.,0.,0.,-0.5,0.,-3.5,0.;	161P	87
214,1,1.,0.5,0.,-3.,0.,-3.,3.5;	163P	88
212,1,4,3.,1.,1,1.5707963267949,0.,0,0,-4.5,4.,0.,4H9.00;	165P	89
214,1,1.,0.5,0.,-3.,9.,-3.,5.5;	167P	90
106,1,3,0.,0.,9.,-0.5,9.,-3.5,9.;	169P	91
216,165,163,167,161,169;	171P	92
106,1,3,0.,0.,-0.5,0.,-3.5,0.;	173P	93
214,1,1.,0.5,0.,-3.,0.,-3.,3.5;	175P	94
212,1,4,3.,1.,1,1.5707963267949,0.,0,0,-4.5,4.,0.,4H9.00;	177P	95

## A. DRAWING AND VIEW EXAMPLE

214,1,1.,0.5,0.,-3.,9.,-3.,5.5;	179P	96
106,1,3,0.,0.,9.,-0.5,9.,-3.5,9.;	181P	97
216,177,175,179,173,181;	183P	98
106,1,3,0.,0.,-0.5,0.,-3.,0.;	185P	99
214,1,1.,0.5,0.,-2.5,0.,-2.5,-2.;	187P	100
212,1,4,3.,1.,1,1.5707963267949,0.,0,0,-4.,5.,0.,4H2.50;	189P	101
214,1,1.,0.5,0.,-2.5,2.5,-2.5,4.5;	191P	102
106,1,3,0.,0.,2.5,-0.5,2.5,-3.,2.5;	193P	103
216,189,187,191,185,193;	195P	104
106,1,3,0.,9.,2.5,9.5,2.5,13.,2.5;	197P	105
214,1,1.,0.5,0.,12.5,2.5,12.5,4.5;	199P	106
212,1,4,3.,1.,1,1.5707963267949,0.,0,0,11.,5.,0.,4H6.50;	201P	107
214,1,1.,0.5,0.,12.5,9.,12.5,6.5;	203P	108
106,1,3,0.,9.,9.,9.5,9.,13.,9.;	205P	109
216,201,199,203,197,205;	207P	110
212,1,7,8.25,1.,1,1.5707963267949,0.,0,0,58.,0.5,0.,7HIN VIEW;	209P	111
124,0.,0.,1.,0.,0.,1.,0.,0.,-1.,0.,0.,0.;	211P	112
124,1.,0.,0.,0.,0.,0.,1.,0.,0.,-1.,0.,0.;	213P	113
406,2,70.,40.;	215P	114
406,2,1,2HIN;	217P	115
S      13G      3D      218P      115	T	1

## **Appendix B. Spline Curves and Surfaces**

### **B.1 Introduction**

Chapter 4 of this Specification includes four different types of spline representations:

1. A parametric piecewise cubic polynomial curve,
2. A rational B-spline curve,
3. A grid of bicubic patches (for surfaces), and
4. 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 Chapter 4 include parametric cubics, piecewise linear, Wilson-Fowler, modified Wilson-Fowler, rational and nonrational B-splines, and rational and nonrational Cartesian product B-spline surfaces. Spline types not supported include splines under tension and extended Coons patches.

Software to convert between parametric spline curves or surfaces and the corresponding rational B- ECO630 spline curves or surfaces is available from the IGES/PDES Administration Office. Materials provided include Pascal source code and accompanying documentation.

### **B.2 Spline Functions**

In [Section 4.14](#), 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:

1.  $N$ : The number of segments,
2.  $T(1), \dots, T(N + 1)$ : The endpoints and the breakpoints separating the cubic polynomial segments,
3.  $A(i), B(i), C(i), D(i)$ ,  $i = 1, \dots, N$ : The coefficients of the polynomials representing the spline in each of the  $N$  segments,
4. CTYPE: The spline type (1=linear, 2=quadratic, 3=cubic, 4=Wilson-Fowler, 5= Modified ECO630 Wilson-Fowler, 6=B-spline) of the sending system. [See Section 4.14](#).
5.  $H$ : Degree of continuity. [See Section 4.14](#).

To evaluate the spline at a point  $u$ , first determine the segment containing  $u$ , i.e., the segment  $i$  ECO630 such that  $T(i) \leq u \leq T(i + 1)$ , then evaluate the cubic polynomial in that segment, i.e., compute

$$S(u) = A(i) + B(i) \cdot (uu) + C(i) \cdot (uu)^2 + D(i) \cdot (uu)^3$$

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.,  $S(T(i)) = A(i), i = 1, \dots, N$ , and  $S(T(N + 1)) = TP0$ ). Computations using the relative displacement also have less floating-point round-off 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:

1. End points  $E1, E2$  and end slopes  $S1, S2$ : The spline can be evaluated using the “Hermite” basis (see [DEB078], p. 59).
2. Values at four points: The spline value can be computed from the Lagrange or Newton interpolation formulas (see [DEB078]).
3. End points and “control” points: There are a number of schemes for computing splines from control points which will not be described here.

DeBoor [DEB078] 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 ECO630 systems, B-splines have been used directly in curve fitting (e.g., the spline Bezier polygon (see [GORD74])) and indirectly in various spline calculations (e.g., computing a cubic spline interpolant). For every set of breakpoints  $T(1), \dots, T(N+1)$  and degree of continuity  $H$ , a set of B-spline functions  $B(1, u), B(2, u), \dots, B(n^*, u)$  can be constructed (see [DEB078]). Then, for any piecewise polynomial  $S(u)$  with these breakpoints and continuity there is a set of B-spline coefficients  $a(1), \dots, a(n^*)$  such that  $S(u)$  can be represented as a linear combination of these B-splines,

$$S(u) = a(1) \cdot B(1, u) + a(2) \cdot B(2, u) + \dots + a(n^*) \cdot B(n^*, u)$$

where

$$n^* = (N - 1) \cdot (3 - H) + 4.$$

B-splines can be computed from piecewise polynomials and *vice versa* (see [DEB078], p. 116).

### B.3 Spline Curves

The comments in this section pertain primarily to [Section 4.14](#).

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),

$$X(u) = S_x(u),$$

$$Y(u) = S_y(u),$$

and

$$Z(u) = S_z(u)$$

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 [IITR68]). It is still used in many turn-key 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 4.14](#) by rotating the parametric spline ( $u$ ,  $Swf(u)$ ) back into the current coordinate system; however, most types of splines defined in [Section 4.14](#) cannot be converted to Wilson-Fowler splines.

#### B.4 Rational B-Spline Curves

The comments in this section pertain primarily to [Section 4.23](#).

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 weights (positive real numbers).

The  $P(i)$  are the control points (points in  $R^3$ ).

The  $b_i$  are the B-spline basis functions.

These are defined as soon as their degree,  $M$ , and underlying knot sequence,  $T$ , are specified. This is done as follows:

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

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  $[T(i - M), T(i + 1)]$ . Between any two adjacent knot values  $T(j), T(j + 1)$  the function can be expressed as a single polynomial of degree  $M$ .

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

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

As 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  $N(t|t_{i-M}, \dots, t_{i+1})$  denote the B-spline basis function of degree  $M$  supported by the interval  $[t_{i-M}, t_{i+1}]$ .

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

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

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

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

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

Rational Bezier curves can be expressed exactly as rational B-spline curves. An unpublished paper by Fuhr [FUHR81] on this subject is available from the IGES/PDES Administration Office.

For further information, see [FAR188]. Note that the indexing of the knot vectors is done differently.

## B.5 Spline Surfaces

The spline surface defined in [Section 4.15](#) is the analog of the spline curve defined in [Section 4.14](#), i.e., it is also pieced together out of other primitive functions. The surface is a grid of parametric bicubic patches defined by:

1.  $M$ : The number of grid lines in  $u$ ,
2.  $TU(1), \dots, TU(M + 1)$ : The breakpoints in  $u$  ( $u$  values of grid lines),
3.  $N$ : The number of grid lines in  $v$ ,
4.  $TV(1), \dots, TV(N + 1)$ : The breakpoints in  $v$  ( $v$  values of grid lines),
5.  $Ax(i, j), Bx(i, j), \dots, Ay(i, j), \dots, Az(i, j), \dots$ , for  $i = 1, \dots, M; j = 1, \dots, N$ :  
The  $M * N$  sets of  $3 * 16$  coefficients defining the bicubic polynomial for each of the three coordinates of the patch,
6. CTYPE: The spline type. (1=linear, 2=quadratic, 3=cubic, 4=Wilson-Fowler, 5=Modified Wilson-Fowler, 6 = B-spline), and
7. PTYPE: The patch type. (1=Cartesian product, 0=unspecified).

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:

$$\begin{aligned} X(u, v) &= Ax(i, j) \cdot vv^0 \cdot uu^0 + Bx(i, j) \cdot vv^0 \cdot uu^1 + Cx(i, j) \cdot vv^0 \cdot uu^2 + \\ &\quad Dx(i, j) \cdot vv^0 \cdot uu^3 + Ex(i, j) \cdot vv^1 \cdot uu^0 + Fx(i, j) \cdot vv^1 \cdot uu^1 + \\ &\quad Gx(i, j) \cdot vv^1 \cdot uu^2 + Hx(i, j) \cdot vv^1 \cdot uu^3 + Kx(i, j) \cdot vv^2 \cdot uu^0 + \\ &\quad Lx(i, j) \cdot vv^2 \cdot uu^1 + Mx(i, j) \cdot vv^2 \cdot uu^2 + Nx(i, j) \cdot vv^2 \cdot uu^3 + \\ &\quad Px(i, j) \cdot vv^3 \cdot uu^0 + Qx(i, j) \cdot vv^3 \cdot uu^1 + Rx(i, j) \cdot vv^3 \cdot uu^2 + \\ &\quad Sx(i, j) \cdot vv^3 \cdot uu^3 \\ Y(u, v) &= Ay(i, j) \dots \\ Z(u, v) &= Az(i, j) \dots \end{aligned}$$

where  $uu = u - TU(i)$  and  $vv = v - TV(j)$ .

The patches in the spline surface are equivalent to the bicubic surface patch (see [ROGE76], p. 170 for the conversion details). The parameters of the bicubic surface 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.

For further information, see [FAR188]. Note that the indexing of the knot vectors is done differently.

## B.6 Rational B-spline Surfaces

The comments in this section pertain primarily to [Section 4.24](#).

A rational B-spline surface is expressed parametrically in the form,

$$G(s, t) = \frac{\sum_{i=0}^{K_1} \sum_{j=0}^{K_2} W(i, j) P(i, j) b_i(s) b_j(t)}{\sum_{i=0}^{K_1} \sum_{j=0}^{K_2} W(i, j) b_i(s) b_j(t)}$$

where the notation is analogous to that used for rational B-spline curves:

The  $W(i, j)$  are the weights (positive real numbers).

The  $P(i, j)$  are the control points (points in  $\mathbb{R}^3$ ).

The  $b_i$  are the B-spline basis functions of degree  $M_1$  determined by the knot sequence  $S(-M_1), \dots, S(N_1 + M_1)$ . The  $b_j$  are the B-spline basis functions of degree  $M_2$  determined by the knot sequence  $T(-M_2), \dots, T(N_2 + M_2)$ . Here,  $N_1 = K_1 - M_1 + 1$  and  $N_2 = K_2 - M_2 + 1$ .

The surface itself is parameterized for  $U(0) \leq s \leq U(1)$  and for  $V(0) \leq t \leq V(1)$  where  $S(0) \leq U(0) < U(1) \leq S(N_1)$  and  $T(0) \leq V(0) < V(1) \leq T(N_2)$ . Rational Bezier surfaces can be expressed exactly as rational B-spline surfaces. An unpublished paper by Fuhr [FUHR81] on this subject is available from the IGES/PDES Administration Office.

## B.6 RATIONAL B-SPLINE SURFACES

If the surface is periodic with respect to the first parametric variable, set PROP4 to 1; otherwise set PROP4 to 0. If the surface is periodic with respect to the second parametric variable, set PROP5 to 1; otherwise set PROP5 to 0. The periodic flags are to be interpreted as purely informational. The surfaces which are flagged to be periodic are to be evaluated exactly the same as in the nonperiodic case.

Software to convert between parametric spline curves or surfaces and the corresponding rational B-spline curves or surfaces is available from the IGES/PDES Administration Office at the National Institute of Standards and Technology. Materials provided include Pascal source code and accompanying documentation.

## Appendix C. Conic Arcs

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

**Accuracy.** The conic equation is numerically sensitive; small changes in the coefficients can cause large changes in the locations of the points satisfying the conic equation. ECO630

**Stability.** The determination of the conic type depends upon whether certain invariants are positive, zero or negative. Working in floating point arithmetic, a machine value of 0.0 is unlikely to be encountered. Furthermore, small changes in coefficient values can easily result in positive values when negative ones are intended, and conversely. ECO630

It is assumed that data are represented by a Conic Arc Entity with the intent of preserving geometric properties (major and minor semi-axes, asymptotes, directrices, *etc.*) in addition to describing the points on the curve.

If the geometric properties are desired, the Conic Arc Entity ([Type 104](#)) should be used as described below.

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

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

The following are suggested sets of values for the cases of an ellipse, a hyperbola, and a parabola: ECO630

Ellipse	
A = AXISY <sup>2</sup>	B = 0
C = AXISX <sup>2</sup>	D = 0
E = 0	F = -A · C
where AXISY and AXISX are the lengths of the major and minor semi-axes (not necessarily in order).	

Note: In the following suggested representations for the hyperbola, the F term is always a positive ECO630 number.

Hyperbola; Case # 1 (Figure C1)			
$A = -\text{AXISY}^2$	$B = 0$	$C = +\text{AXISX}^2$	$D = 0$
$E = 0$	$F = -A \cdot C$		
where $\text{AXISX} = 1/2$ the length of the transverse axis; $\text{AXISY} = 1/2$ the length of the conjugate axis.			

Hyperbola; Case # 2 (Figure C2)			
$A = +\text{AXISY}^2$	$B = 0$	$C = -\text{AXISX}^2$	$D = 0$
$E = 0$	$F = -A \cdot C$		
where $\text{AXISX} = 1/2$ the length of the conjugate axis; $\text{AXISY} = 1/2$ the length of the transverse axis.			

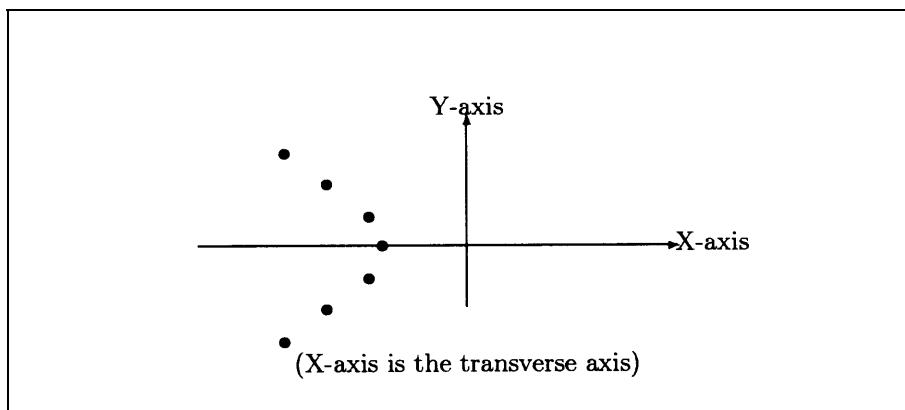


Figure C1. Case 1: Hyperbola oriented (aligned) along the X-axis

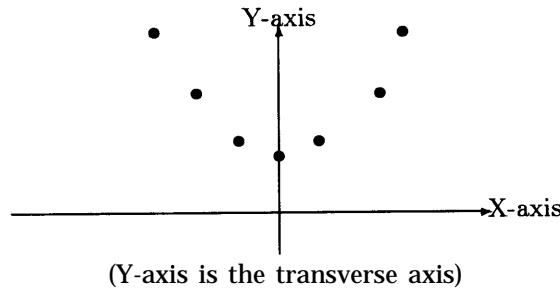


Figure C2. Case 2: Hyperbola oriented (aligned) along the Y-axis

ECO630

<b>Parabola</b>	
A = 0 (or 1)	B = 0
C = 1(or 0, if A=1)	D = 4*DIST (or 0, if A=1)
E = 0 (or 4*DIST, if A=1)	F = 0
where DIST is the distance of the vertex from the focus.	

### Preprocessor Conic Handling

The conic arc shall be put into standard form, parallel to the X or Y axis and centered about the ECO630 origin. A Transformation Matrix Entity ([Type 124](#)) shall be used to move the conic arc into its desired position in space. In this form, the coefficients in the format that should be 0.0 will be exactly so. In particular, for the ellipse and hyperbola B, D, and E shall be 0.0, and for the parabola B and F and either A and E or C and D shall be 0.0.

Determination of the conic type from the equations becomes straightforward for the postprocessor. For further mathematical details, see [THOM60].

## Appendix D. Color-Space Mappings

It is often more convenient to operate in some color space other than RGB. The relationship between R, G, and B (Red, Green, and Blue) and C, M, and Y (Cyan, Magenta, and Yellow) is given by: ECO630

$$R = 100.0 - C$$

$$R = \text{red} \quad C = \text{cyan}$$

$$G = 100.0 - M$$

$$\text{where: } G = \text{green} \quad M = \text{magenta}$$

$$B = 100.0 - Y$$

$$B = \text{blue} \quad Y = \text{yellow}$$

The HSL (Hue, Saturation, Lightness) color space can be defined in terms of RGB in several ways with subtle variations. A typical approach is given by the following transformation:

$$H = \frac{1}{2\pi} \tan^{-1} \left( \frac{2R - G - B}{\sqrt{3}(G - B)} \right)$$

$$S = \sqrt{R^2 + G^2 + B^2 - RG - RB - BG}$$

$$L = \frac{1}{3}(R + G + B)$$

where:

ECO630

H = Hue

S = Saturation

L = Lightness

Variations on this transformation are given in [JOBL78] and [SMIT78].

## **Appendix E. ASCII Form Conversion Utility**

This appendix gives details of a utility program to convert a file in the ASCII Form from the regular ECO630 (fixed line length) ASCII Format to the Compressed ASCII Format and back again. The program is written in FORTRAN 77 code. The program is known to fail to convert a file from Compressed ASCII Format to regular ASCII Format if the file contains any string constant compressed such that an ASCII character "D" falls into column one. The source code is available from the IGES/PDES Administration Office.

```
C*****
C      THIS PROGRAM IS WRITTEN IN VAX 4.2 FORTRAN 77 SOURCE. ITS PURPOSE IS
C      TO CONVERT BETWEEN REGULAR ASCII FORMAT AND COMPRESSED ASCII FORMAT.
C
C      PROGRAM IGES
C
C      PROGRAM ORIGINALLY WRITTEN BY J. M. SPAETH 7-24-84
C                           GENERAL ELECTRIC CORP. RE. & DEV.
C      RE-WRITTEN BY LEE KLEIN 9-20-84
C                           GENERAL DYNAMICS CAD/CAM POMONA DIV.
C      REVISED BY LEE KLEIN 7-28-86
C                           GENERAL DYNAMICS CAD/CAM POMONA DIV.
C      REVISED BY LEE KLEIN 8-1-86
C                           GENERAL DYNAMICS CAD/CAM POMONA DIV.
C      REVISED BY LEE KLEIN 8-7-86
C                           GENERAL DYNAMICS CAD/CAM POMONA DIV.
C      REVISED BY ROBERT COLSHER 22 AUG 1986
C                           IGES DATA ANALYSIS COMPANY
C
C      PURPOSE:
C          TO CONVERT NEW FORM OF IGES OUTPUT TO OLD FORM AND
C          OLD FORM TO NEW.
C
C      INPUT:
C          YOU MUST GIVE THE NAME ( INCLUDING DIRECTORY IF DIFFERENT ) OF
C          THE FILE CONTAINING THE NEW FORM OF OUTPUT. YOU MUST ALSO
C          GIVE THE NAME OF THE FILE TO CONTAIN THE CONVERTED OUTPUT.
C
C*****
C      SPECIAL NOTES:
C
C      1. THE DOLLAR SIGN IN I/O FORMAT STATEMENTS IS THERE TO SUPPRESS
C         THE CARRIAGE RETURN AT THE END OF THE PROMPT LINE.
```

## E. ASCII FORM CONVERSION UTILITY

```
C      2. IN COMPILERS THAT DO NOT ACCEPT A VARIABLE LENGTH OUTPUT FORMAT,
C          SOME MEANS OF COMPRESSING BLANK PADDED LINES MUST BE USED.
C      3. SEE CHANGE NOTES THROUGHOUT THE CODE
C
C*****CHARACTER * 80 LINE1
C*****CHARACTER * 60 INFILE,OUTFIL
C
C      PROMPT AND GET FILE NAMES...
C
C      GOTO 1010
1000  WRITE(*,1900)
1010  WRITE(*,1910)
        WRITE(*,1920)
        READ(*,1930)INFILE
        WRITE(*,1940)
        READ(*,1930)OUTFIL
C
C      READ THE FIRST LINE...
C
C      OPEN(UNIT=10,FILE=INFILE,STATUS='OLD',ERR=1000)
C      READ(10,1950)LINE1
C      CLOSE(10)
C
C      CHECK TO SEE WHICH FORM THE INPUT IS IN...
C
C      ONLY A 'C'OMPRESS RECORD CAN OCCUR AT BEGINNING OF 'NEW' FILE
C      ONLY A 'S'TART RECORD CAN OCCUR AT BEGINNING OF 'OLD' FILE
C
C      IF (LINE1(73:73).EQ.'C') THEN
C          CALL OLDFRM(INFILE,OUTFIL)
C      ELSE IF (LINE1(73:73).EQ.'S') THEN
C          CALL NEWFRM(INFILE,OUTFIL)
C      ELSE
C          WRITE(*,1960)' File contains ILLEGAL record format'
C          CLOSE(10)
C          STOP
C      ENDIF
C      WRITE(*,1960)', '
C      WRITE(*,1960)', IGES conversion complete'
C      WRITE(*,1960)', '
C
C      FORMATS
C
1900  FORMAT(/1X,'Error in filename. Try again.')
1910  FORMAT(/1X,'*** IGES FILE CONVERSION PROGRAM ***')
1920  FORMAT($,' Enter input file name: ')
1930  FORMAT(A60)
1940  FORMAT($,' Enter output file name: ')
1950  FORMAT(A80)
```

## E. ASCII FORM CONVERSION UTILITY

```
1960 FORMAT(A)
C
END

*****
C
C      SUBROUTINE OLDFRM(INFILE,OUTFIL)
C
C      OLD FORM CONVERSION
C
C      PROGRAM ORIGANALLY WRITTEN BY J. M. SPAETH 7-24-84
C                      GENERAL ELECTRIC CORP. RE. & DEV.
C      RE-WRITTEN BY LEE KLEIN 9-20-84
C                      GENERAL DYNAMICS CAD/CAM POMONA DIV.
C      REVISED BY ROBERT COLSHER 22 AUG 1986
C                      IGES DATA ANALYSIS COMPANY
C
C      PURPOSE:
C          TO CONVERT NEW FORM OF IGES OUTPUT TO OLD FORM...
C
C      VARIABLE DECLARATIONS...
C
CHARACTER * (*) INFILE,OUTFIL
CHARACTER * 8 BLNK,INARR(20),NEWARR(20)
CHARACTER * 80 INLINE
CHARACTER * 160 OUTARR
INTEGER          ICNT1,ICNT2,ICNT3,IA,IB,IC
INTEGER          IJ,IK,IL,IT
C
C      INITIALIZE OUTARR TO BLANKS...
C
C      OUTARR(1:160)=' '
C
C      OPEN INPUT AND TEMP FILES...
C
OPEN(UNIT=1,FILE='FILE1.TMP',STATUS='NEW',CARRIAGECONTROL='LIST')
OPEN(UNIT=2,FILE='FILE2.TMP',STATUS='NEW',CARRIAGECONTROL='LIST')
OPEN(UNIT=3,FILE='FILE3.TMP',STATUS='NEW',CARRIAGECONTROL='LIST')
OPEN(UNIT=4,FILE='FILE4.TMP',STATUS='NEW',CARRIAGECONTROL='LIST')
OPEN(UNIT=9,FILE=OUTFIL,STATUS='NEW',CARRIAGECONTROL='LIST')
OPEN(UNIT=10,FILE=INFILE,STATUS='OLD')
C
C      INITIALIZE COUNTERS...
C
ICNT1 = -1
ICNT2 = 1
ICNT3 = 0
C
C      READ THE FILE AND SEPARATE INTO PARTS...
```

## E. ASCII FORM CONVERSION UTILITY

```
2000 READ(10,2900,END=2090)INLINE
      IF ((INLINE(73:73).EQ.'S').OR.(INLINE(73:73).EQ.'G')) GOTO 2010
      IF (INLINE(73:73).EQ.'T') GOTO 2020
      IF ((INLINE(1:1).EQ.'@').OR.(INLINE(1:1).EQ.'D')) GOTO 2040
C
C      IF IT IS AN C THAN DELETE IT OFF BY READING THE NEXT LINE
C
C      IF (INLINE(73:73).EQ.'C') GOTO 2000
C
C      PUT THE PARAMETER DATA INTO FILE4.TMP...
C
C      WRITE(4,2910) (INLINE(1:64),ICNT1,'P',ICNT2)
C      ICNT2 = ICNT2 + 1
C      GOTO 2000
C
C      WRITE HEADER LINES INTO FILE1.TMP...
C
2010 WRITE(1,2900)INLINE
      GOTO 2000
C
C      WRITE TERMINATION LINE INTO FILE2.TMP...
C
2020 WRITE(2,2900)INLINE
      GOTO 2000
C
C      WRITE DIRECTORY ENTRY LINES INTO FILE3.TMP...
C
2030 WRITE(3,2920) (OUTARR(1:8),ICNT2,OUTARR(17:80))
      WRITE(3,2900) OUTARR(81:160)
      ICNT1 = ICNT1 + 2
      GOTO 2000
C
C      REWRITE TO DE LINES IN THE NEW FORM...
C
C      GO THRU INLINE ONE CHAR. AT A TIME LOOKING FOR THE
C      DELIMETER (@, ,_)...
C
2040 IL = 1
2050 IF (IL.GT.80) GOTO 2000
      IF (INLINE(IL:IL).EQ.')') GOTO 2030
      IF (INLINE(IL:IL).NE.'@') GOTO 2080
C
C      DETERMINE IF THE FIELD IS ONE OR TWO CHARACTERS...
C
      IF (INLINE(IL+2:IL+2).EQ.'_') THEN
          IC=ICHAR(INLINE(IL+1:IL+1))-48
          IL=IL+2
      ELSE
          IA=ICHAR(INLINE(IL+1:IL+1))-48
          IB=ICHAR(INLINE(IL+2:IL+2))-48
```

## E. ASCII FORM CONVERSION UTILITY

```
IC=10*IA+IB
IL=IL+3
ENDIF
C
C      AT THIS POINT IC IS THE NUMBER OF THE RECORD FIELD BEING
C      PROCESSED, AND INLINE(IL)=USCORE
C
IT=0
IK=0
IJ=(IC-1)*8+1
C
C      RESET THE FIELD TO BE CHANGED TO ALL BLANKS IN ORDER TO CREATE
C      A COMPLETELY NEW FILED...
C
OUTARR(IJ:IJ+7)=' '
C
C      WE WILL NOW CONTINUE THRU THE LINE PICKING OFF THE CHAR.
C      OF THE RECORD FIELD ONE AT A TIME UNTIL A DELIMETER IS HIT...
C
2060 IK=IK+1
IF (INLINE(IL+IK:IL+IK).EQ.'@') GOTO 2070
IF (INLINE(IL+IK:IL+IK).EQ.' ') THEN
  IF (INLINE(IL+IK+1:IL+IK+1).EQ.' ') GOTO 2070
ENDIF
IF (INLINE(IL+IK:IL+IK).EQ.';') GOTO 2070
IT=IT+1
IJ=(IC-1)*8+IT
OUTARR(IJ:IJ)=INLINE(IL+IK:IL+IK)
IF (IC.EQ.1) THEN
  OUTARR(IJ+80:IJ+80)=INLINE(IL+IK:IL+IK)
ENDIF
GOTO 2060
2070 IL=IL+IT
2080 IL = IL + 1
GOTO 2050
C
C      REWIND ALL FILES BEFORE WE WRITE THEM TO OUTPUT...
C
2090 REWIND 1
REWIND 2
REWIND 3
REWIND 4
C
C      WRITE START AND GLOBAL RECORDS TO OUTPUT FILE...
C
2100 READ(1,2900,END=2110) INLINE
WRITE(9,2900) INLINE
GOTO 2100
C
```

## E. ASCII FORM CONVERSION UTILITY

```
C      WRITE THE DE RECORDS TO OUTPUT FILE.  THEY NOW BECOME RE-FORMATTED...
C
2110  READ(3,2930,END=2140) (INARR(I),I=1,10)
      READ(3,2930) (INARR(I),I=11,20)
      DO 2130 IP=1,20
          IF ((IP.NE.9).AND.(IP.NE.18)) GOTO 2120
          NEWARR(IP)=INARR(IP)
          GOTO 2130
C
C      CHANGE THOSE FIELDS THAT MUST BE RIGHT JUSTIFIED
C
2120  NEWARR(IP)=BLNK(INARR(IP))
2130  CONTINUE
      ICNT3=ICNT3+1
      WRITE(9,2940) ((NEWARR(J),J=1,9),'D',ICNT3)
      ICNT3=ICNT3+1
      WRITE(9,2940) ((NEWARR(J),J=11,19),'D',ICNT3)
      GOTO 2110
C
C      WRITE THE PD LINES TO THE OUTPUT FILE...
C
2140  READ(4,2900,END=2150) INLINE
      WRITE(9,2900) INLINE
      GOTO 2140
C
C      WRITE THE TERMINATE LINES TO THE OUTPUT FILE...
C
2150  READ(2,2900,END=2160) INLINE
      WRITE(9,2900) INLINE
      GOTO 2150
C
C      NOW CLOSE THE FILES AND DELETE THE TEMP ONES...
C
2160  CONTINUE
      CLOSE(UNIT=1,STATUS='DELETE')
      CLOSE(UNIT=2,STATUS='DELETE')
      CLOSE(UNIT=3,STATUS='DELETE')
      CLOSE(UNIT=4,STATUS='DELETE')
      CLOSE(9)
      CLOSE(10)
      RETURN
C
C      FORMATS
C
2900  FORMAT(A80)
2910  FORMAT(A64,1I8,1A1,1I7)
2920  FORMAT(A8,I8,A64)
2930  FORMAT(10A8)
2940  FORMAT(9A8,A,I7)
```

## E. ASCII FORM CONVERSION UTILITY

```
C
C      END
C*****
C
C      CHARACTER*(*) FUNCTION BLNK(BUF)
C
C      START FUNCTION BLNK HERE
C
C      WRITTEN BY P. R. KENNICOTT 9-29-83.
C                      GENERAL ELECTRIC CORP. RE. & DEV.
C      RE-WRITTEN BY LEE KLEIN 9-2-84.
C                      GENERAL DYNAMICS CAD/CAM POMONA DIV.
C      REVISED BY LEE KLEIN 8-7-86
C                      GENERAL DYNAMICS CAD/CAM POMONA DIV.
C      REVISED BY ROBERT COLSHER 22 AUG 1986
C                      IGES DATA ANALYSIS COMPANY
C
C      PURPOSE:
C          TO REMOVE BLANKS FROM END OF A CHARACTER STRING (RIGHT JUSTIFY)
C
C      INPUT:
C          BUF STRING WITH TRAILING BLANKS
C
C      OUTPUT:
C          BLNK STRING WITH TRAILING BLANKS REMOVED
C
C      METHOD:
C          FIND FIRST BLANK, THEN TRANSLATE OUTPUT STRING
C
C      RESTRICTIONS:
C          1. BUF <= 512 CHARACTERS.
C          2. LENGTHS OF BUF & BLNK MUST BE =.
C          3. FIRST CHARACTER MUST NOT BE BLANK OR NO CONVERSION.
C
C
C      VARIABLE DECLARATIONS...
C
C          CHARACTER*(*) BUF
C          INTEGER I
C          CHARACTER*512 IBUF
C
C          SET UP COUNTERS...
C
C          N=INDEX(BUF(1:), ' ') - 1
C          M=LEN(BUF)
C
C          CHECK FOR SIZE TOO BIG...
C
C          IF (M.GT.512) STOP 'Buffer too big at function BLNK'
C
```

## E. ASCII FORM CONVERSION UTILITY

```
C      CHECK FOR FIRST CHAR A BLANK...
C
C      IF (BUF(1:1).EQ.' ' .OR.BUF(M:M).NE.' ') THEN
C          BLNK=BUF
C          RETURN
C      ENDIF
C
C      OK PROCESS STRING...
C
C      DO 3000 I=1,N
3000  IBUF(M-I+1:M-I+1)=BUF(N-I+1:N-I+1)
      IBUF(1:M-N)=' '
      BLNK=IBUF
      RETURN
      END
*****
C      SUBROUTINE NEWFRM(INFILE,OUTFIL)
C
C      START NEW SUBROUTINE HERE
C
C      PROGRAM ORIGANALLY WRITTEN BY J. M. SPAETH 7-24-84
C                      GENERAL ELECTRIC CORP. RE. & DEV.
C      RE-WRITTEN BY LEE KLEIN 9-20-84
C                      GENERAL DYNAMICS CAD/CAM POMONA DIV.
C      REVISED BY LEE KLEIN 8-7-86
C                      GENERAL DYNAMICS CAD/CAM POMONA DIV.
C      REVISED BY ROBERT COLSHER 22 AUG 1986
C                      IGES DATA ANALYSIS COMPANY
C
C      PURPOSE:
C          TO CONVERT OLD FORM OF IGES OUTPUT TO NEW FORM...
C
C      VARIABLE DECLARATIONS...
C
C      INTEGER PDRCD
C      CHARACTER * 80 INLINE
C      CHARACTER * (*) INFILE,OUTFIL
C
C      OPEN THE INPUT AND TEMP FILES...
C
C      OPEN(UNIT=10,FILE=INFILE,STATUS='OLD')
C      OPEN(UNIT=1,FILE='TEST.TMP',STATUS='NEW',CARRIAGECONTROL='LIST')
C      OPEN(UNIT=2,FILE='FILE2.TMP',STATUS='NEW',RECL=80,
+          ACCESS='DIRECT',FORM='FORMATTED')
C      OPEN(UNIT=3,FILE='FILE3.TMP',STATUS='NEW')
C      OPEN(UNIT=7,FILE='FILE5.TMP',STATUS='NEW')
C
C      WRITE THE HEADER WITH A "C" TO SHOW COMPRESSED ASCII FORM...
```

## E. ASCII FORM CONVERSION UTILITY

```
C          WRITE(1,4900)'C',1
C
C          SEPERATE THE PD AND DE RECORDS, WHILE WRITING G,S, &T LINES
C          TO THE OUTPUT FILE...
C
4000  READ(10,4910,END=4040) INLINE
      IF (INLINE(73:73).EQ.'D') GOTO 4010
      IF (INLINE(73:73).EQ.'P') GOTO 4020
      IF (INLINE(73:73).EQ.'T') GOTO 4030
C
C          WRITE HEADER LINES INTO A TEST.TMP...
C
C          WRITE (1,4910) INLINE
C          GOTO 4000
C
C          WRITE DIRECTORY LINES INTO FILE3.TMP...
C
4010  WRITE (3,4910) INLINE
      GOTO 4000
C
C          WRITE PARAMETER DATA INTO FILE2.TMP...
C
4020  READ (INLINE(74:80),4920) PDRCD
      WRITE (2,REC=PDRCD,FMT=4910) INLINE
      GOTO 4000
C
C          WRITE TERMINATE RECORD INTO FILE5.TMP...
C
4030  WRITE (7,4910) INLINE
C
4040  CALL XPD
      REWIND 7
C
4050  READ(7,4910,END=4060) INLINE
      WRITE (1,4910) INLINE
      GOTO 4050
4060  CALL CMPRES(OUTFIL)
C
C          CLOSE FILES AND DELETE TEMP ONES...
C
C          CLOSE(UNIT=1,STATUS='DELETE')
C          CLOSE(UNIT=2,STATUS='DELETE')
C          CLOSE(UNIT=3,STATUS='DELETE')
C          CLOSE(UNIT=4)
C          CLOSE(UNIT=7,STATUS='DELETE')
C          CLOSE(UNIT=10)
C          RETURN
C
C          FORMATS
C
```

## E. ASCII FORM CONVERSION UTILITY

```
4900 FORMAT(72X,A,I7)
4910 FORMAT(A80)
4920 FORMAT(I7)
C
END
*****
C
SUBROUTINE XPD
C
C PROGRAM ORIGINALLY WRITTEN BY J. M. SPAETH 7-24-84
C           GENERAL ELECTRIC CORP. RE. & DEV.
C REVISED BY LEE KLEIN 8-7-86
C           GENERAL DYNAMICS CAD/CAM POMONA DIV.
C REVISED BY ROBERT COLSHER 22 AUG 1986
C           IGES DATA ANALYSIS COMPANY
C
C PURPOSE:
C           TO TRANSFER ALL PD & DE RECORDS FORM TEMPORY FILES TO
C           OUTPUT FILE IN MERGED FORM...
C
C VARIABLE DECLARATIONS...
C
CHARACTER * 1   SCOLN/';/'
CHARACTER * 8   BLNK,LSTDAT(20),NEWDAT(20),NUDAT
CHARACTER * 80  PDLINE,DELIN1,DELIN2
CHARACTER * 160 NEWDE
INTEGER        LFLD(20),FLDNUM,FLDBEG,FLDEND
INTEGER        CHRPTR,NEWPTR,PDPTR,PDCNT
C
C REWIND THE FILES...
C
REWIND 3
C
INITIALIZE LAST DATA SO AS NOT TO EQUAL NEXT DATA...
C
DO 5000 FLDNUM=1,20
  LSTDAT(FLDNUM) = 'XXXXXXXX'
5000 CONTINUE
C
C GET NEW DE RECORD
C
5010 READ(3,5900,END=5100) DELIN1
  READ(3,5900) DELIN2
  READ(DELIN1(9:16),5910) PDPTR
  READ(DELIN2(25:32),5910) PDCNT
  DELIN1(73:73)='
  DELIN2(73:73)='
C
C CLEAR OUT THE DE RECORD BUFFER...
C
NEWDE(1:160) = , ,
```

## E. ASCII FORM CONVERSION UTILITY

```
C
C      GET THE DATA FROM EACH FIELD OF THE DE RECORD SET...
C
      FLDBEG = -7
      FLDEND = 0
      DO 5020 FLDNUM=1,10
          FLDBEG=FLDBEG + 8
          FLDEND=FLDEND + 8
          READ(DELIN1(FLDBEG:FLDEND),5920) NEWDAT(FLDNUM)
          READ(DELIN2(FLDBEG:FLDEND),5920) NEWDAT(FLDNUM+10)
5020  CONTINUE
C
C      FIELD 9 MUST BE ZERO FILLED
C
      DO 5030 I = 1,8
          IF (NEWDAT(9)(I:I).EQ.' ') NEWDAT(9)(I:I) = '0'
5030  CONTINUE
C
C      FIELD 18 MUST BE RIGHT JUSTIFIED...
C
      NEWDAT(18)=BLNK(NEWDAT(18))
C
C      DETERMINE THE LENGTH OF THE DATA WITHIN EACH FIELD
C
      DO 5050 FLDNUM=1,20
          DO 5040 I=1,8
              IF (NEWDAT(FLDNUM)(I:I).NE.' ') THEN
                  LFLD(FLDNUM)= 9 - I
                  GOTO 5050
              ENDIF
5040  CONTINUE
5050  CONTINUE
C
C      WRITE THE DE SEQUENCE NUMBER AT THE BEGINNING OF THE OUTPUT DE RECORD
C
      NUDAT=NEWDAT(10)
      ENCODE(LFLD(10)+1,5930,NEWDE(1:LFLD(10)+1)) NUDAT(9-LFLD(10):8)
      CHRPTR = LFLD(10) + 2
C
C      SEARCH NEW DE RECORD SET FOR CHANGED DATA; WHEN FOUND WRITE CHANGED
C      DATA TO OUTPUT DE RECORD...
C
      DO 5060 FLDNUM=1,20
C
C      SKIP FIELDS THAT NO LONGER NEED PROCESSING...
C
      IF ((FLDNUM.EQ. 2).OR.
+        (FLDNUM.EQ.10).OR.
+        (FLDNUM.EQ.11).OR.
+        (FLDNUM.EQ.20)) GOTO 5060
```

## E. ASCII FORM CONVERSION UTILITY

```
IF (NEWDAT(FLDNUM).NE.LSTDAT(FLDNUM)) THEN
  IF (FLDNUM.GT.9) THEN
    ENCODE(4,5940,NEWDE(CHR PTR:CHR PTR+3)) FLDNUM
    CHR PTR=CHR PTR+4
  ELSE
    ENCODE(3,5950,NEWDE(CHR PTR:CHR PTR+2)) FLDNUM
    CHR PTR=CHR PTR+3
  ENDIF
  IF (LFLD(FLDNUM).NE.0) THEN
    IF (FLDNUM.NE.9) THEN
      READ(NEWDAT(FLDNUM)(9-LFLD(FLDNUM):8),5960)
      +     NEWDE(CHR PTR:CHR PTR-1+LFLD(FLDNUM))
      CHR PTR=CHR PTR+LFLD(FLDNUM)
    ELSE
      C
      C   FIELD 9 IS A SPECIAL CASE...
      C
      C   READ(NEWDAT(9)(1:8),5960) NEWDE(CHR PTR:CHR PTR+7)
      C   CHR PTR=CHR PTR+8
      ENDIF
      ENDIF
      ENDIF
      C
      C   STORE DATA FROM CURRENT DE RECORD SET TO COMPARE WITH NEXT SET
      C
      C   LSTDAT(FLDNUM)=NEWDAT(FLDNUM)
5060  CONTINUE
      NEWDE(CHR PTR:CHR PTR) = SCOLN
      C
      C   IF OUTPUT DE RECORD > 80 CHAR'S, WRITE 2 LINES...
      C
      C   IF (CHR PTR.GT.80) THEN
      DO 5070 I=1,11
        IF (NEWDE(82-I:82-I).EQ.'Q') GOTO 5080
      5070  CONTINUE
      5080  WRITE(1,5970)NEWDE(1:81-I)
        NEWDE(1:80)=NEWDE(82-I:161-I)
        ENDIF
        WRITE(1,5900) NEWDE(1:80)
      C
      C   ERASE UNNECESSARY DATA FROM PD RECORD AND WRITE TO OUTPUT FILE;
      C
      DO 5090 IL=1,PDCNT
      READ (2,5900,REC=PD PTR) PDLIN E
      PD PTR = PD PTR+1
      PDLIN E(65:80) = ' '
      WRITE(1,5900) PDLIN E
      5090  PDLIN E(1:80) = ' '
      C
      C   END OF LOOP GET NEXT DE RECORD...
```

## E. ASCII FORM CONVERSION UTILITY

```
C
      GOTO 5010
5100  CONTINUE
      RETURN
C
C      FORMATS
C
5900  FORMAT(A80)
5910  FORMAT(I8)
5920  FORMAT(A8)
5930  FORMAT ('D',A)
5940  FORMAT('Q',I2,'_')
5950  FORMAT('Q',I1,'_')
5960  FORMAT(A)
5970  FORMAT(A<81-I>)
C
      END

*****
C
C      SUBROUTINE CMPRES(OUTFIL)
C
C      START OF SUBROUTINE
C
C      PROGRAM ORIGINALLY WRITTEN BY J. M. SPAETH 7-24-84
C                      GENERAL ELECTRIC CORP. RE. & DEV.
C      REVISED BY LEE KLEIN 8-7-86
C                      GENERAL DYNAMICS CAD/CAM POMONA DIV.
C      REVISED BY ROBERT COLSHER 22 AUG 1986
C                      IGES DATA ANALYSIS COMPANY
C
C      PURPOSE:
C          TO CLEAR AWAY ALL TRAILING BLANKS FROM THE OUTPUT FILE
C
C      VARIABLE DECLARATIONS...
C
CHARACTER * 80 TEXT
CHARACTER * (*) OUTFIL
INTEGER      LENGTH
C
C      REWIND THE INPUT FILE AND OPEN THE OUTPUT FILE
C
C      REWIND 1
OPEN (UNIT=4,NAME=OUTFIL,STATUS='NEW',CARRIAGECONTROL='LIST',
+      ERR=6000)
GOTO 6010
C
C      GETS HERE IF THERE IS AN ERROR IN THE OUTPUT FILE NAME...
C
```

## E. ASCII FORM CONVERSION UTILITY

```
6000 WRITE(*,6900) 'Error in OUTPUT file name. Output written to file',
+           ' IGES.OUT'
      OPEN (UNIT=4,NAME='IGES.OUT',STATUS='NEW',CARRIAGECONTROL='LIST')
C
C     READ RECORD LINES INTO BUFFER ONE AT A TIME...
C
6010 READ(1,6910,END=6999) TEXT
      LENGTH = 80
C
C     GO THRU EACH LINE DELETING TRAILING BLANKS
C
6020 IF (TEXT(LENGTH:LENGTH).NE.' ') GOTO 6030
      LENGTH=LENGTH-1
      IF (LENGTH.GT.1) GOTO 6020
C
C     WRITE PROCESSED LINES TO THE OUTPUT FILE
C
6030 WRITE(4,6920) TEXT(1:LENGTH)
C
      GOTO 6010
C
6999 CONTINUE
      RETURN
C
C     FORMATS
C
6900 FORMAT(1X,A,A)
6910 FORMAT(A80)
6920 FORMAT(A<LENGTH>)
C
      END
```

## **Appendix F.      Obsolete Entities**

### **F.1 General**

The addition of new entities and forms which greatly increase the capability for transfer of specific data constructs has given cause to deprecate other entities and forms published in previous versions of this Specification. A file conforming to this version of the Specification shall not contain deprecated or obsolete constructs, forms, or entities. The parameter lists for these entities and forms are included herein to provide for interpretation of files created under an earlier version. The new entities or forms which are valid for the previous forms are as follows:

<b>Obsolete Entity/Form</b>		<b>Valid Entity/Form</b>	
402/2	External Logical Reference File Index	402/12	External Reference File Index
402/6	View List	402/3	Views Visible
402/8	Signal String	402/18	Flow
402/10	Text Node	312	Text Display Template
402/11	Connect Node	132	Connect Point
406/4	Region Fill Property	230	Sectioned Area

In addition, Form 0 of the Conic Entity ([Type 104](#)) is deprecated, FC 0 for the General Note Entity [ECO630](#) ([Type 212](#)) is obsolete, and the use of the Single Parent Associativity ([Type 402, Form 9](#)) to create holes in bounded planar regions is deprecated.

### **F.2 Obsolete General Note FC 0**

FC 0 specifies an obsolete symbol font for the General Note Entity ([Type 212](#)) and should not be used. It is included here ([see Figure F1](#)) for reference in processing files written in accordance with Version 1.0 of the Specification.

## F.2 OBSOLETE GENERAL NOTE FC ZERO

0	☒	10	↓	20		30	0
1	÷	11	→	21	!	31	1
2	≤	12	←	22	"	32	2
3	≥	13	ϕ	23	#	33	3
4	△	14	θ	24	\$	34	4
5	✓	15	γ	25	%	35	5
6	×	16	ψ	26	&	36	6
7	≡	17	ω	27	,	37	7
8	≠	18	λ	28	(	38	8
9	ʃ	19	α	29	)	39	9
A	▷	1A	δ	2A	*	3A	:
B	∨	1B	μ	2B	+	3B	;
C	∧	1C	π	2C	,	3C	<
D	≈	1D	—	2D	-	3D	=
E	Σ	1E	±	2E	.	3E	>
F	↑	1F	°	2F	/	3F	?

Figure F1. Obsolete General Note Font specified by FC 0

40	⌚	50	P	60	'	70	(P)
41	A	51	Q	61	↙	71	⌚
42	B	52	R	62	㊥	72	◎
43	C	53	S	63	□	73	⌚
44	D	54	T	64	○	74	□
45	E	55	U	65	○	75	○
46	F	56	V	66	//	76	△
47	G	57	W	67	∅	77	◊
48	H	58	X	68	↗	78	↑
49	I	59	Y	69	=	79	☒
4A	J	5A	Z	6A	⊕	7A	Y
4B	K	5B	[	6B	∩	7B	{
4C	L	5C	\	6C	⊥	7C	
4D	M	5D	]	6D	Ⓜ	7D	}
4E	N	5E	^	6E	∅	7E	~
4F	O	5F	-	6F	○	7F	z

Figure F1. Obsolete General Note Font specified by FC 0 (Continued)

## F.3 OBSOLETE USE OF SINGLE PARENT ASSOCIATIVITY

### F.3 Obsolete Use of Single Parent Associativity

Use of the Single Parent Associativity has been deprecated. This functionality shall be implemented ECO618 using the Trimmed (Parametric) Surface Entity ([Type 144](#)) or the Bounded Surface Entity ([Type 143](#)).

The following is the obsolete description that was present in previous versions of the Specification.

The case of a bounded portion of a fixed plane minus some portion(s) of that plane is 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 file and has a backpointer to the associativity structure. The child plane entity will have a subordinate entity switch class of 01 (Physically Dependent).

## F.4 EXTERNAL LOGICAL REFERENCE FILE INDEX (TYPE 402, FORM 2)

### F.4 External Logical Reference File Index (Type 402, Form 2)

The External Logical Reference File Index Entity appears in one file which contains references from another file. It contains a list of the symbolic names used by the referencing files and the DE pointers to the corresponding definitions within the referenced file. See Section 3.6.4 and the External Reference Entity (Type 416) for more detail.

#### DEFINITION

<u>Index</u>	<u>Set Value</u>	<u>Meaning</u>
1	1	One class (externally referenced entities)
2	2	Back pointers not required
3	2	Unordered list of entries in a class
4	2	Number of items in an entry
5	2	First item is a value (External Reference Entity symbolic name)
6	1	Second item is a pointer (internal entity DE pointer)

#### DESCRIPTION

##### Directory Entry

**Entity Type Number: 402**

**Form Number: 2**

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	N	Integer	Number of index entries
2	NAME1	String	First External Reference Entity symbolic name
3	PTR1	Pointer	Pointer to the DE of the first internal entity
:	:	:	
2 * N	NAMEN	String	Last External Reference Entity symbolic name
1+2*N	PTRN	Pointer	Pointer to the DE of the last internal entity

Additional pointers as required (see Section 2.2.4.5.2).

## F.5 VIEW LIST ASSOCIATIVITY (TYPE 402, FORM 6)

### F.5 View List Associativity (Type 402, Form 6)

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 entries) 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

<u>Index</u>	<u>Set Value</u>	<u>Meaning</u>
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

#### DESCRIPTION

##### Directory Entry

**Entity Type Number: 402**  
**Form Number: 6**

#### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	1	Integer	Single entry in first class
2	N1	Integer	Number of entities in second class
3	DEV	Pointer	Pointer to the DE of the View Entity
4	DE1	Pointer	Pointer to the DE of the first entity visible in view specified in Parameter 3
:	:	:	
3+N1	DEN1	Pointer	Pointer to the DE of the last entity visible

Additional pointers as required (see Section 2.2.4.5.2).

## F.6 SIGNAL STRING ASSOCIATIVITY (TYPE 402, FORM 8)

### F.6 Signal String Associativity (Type 402, Form 8)

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 a 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 Curve Entity (Type 102), Copious Data Entity (Type 106, Forms 11 or 12), or any of the entities which maybe members of Composite Curve Entity.

#### DEFINITION

<u>Index</u>	<u>Set Value</u>	<u>Meaning</u>
1	4	Four classes
		Class 1 (Signal Names)
2	2	Back pointers not required
3	1	Ordered
4	1	One item per entry
5	2	Item is value
		Class 2 (Connections)
6	1	Back pointers required
7	2	Unordered
8	1	One item per entry
9	1	Pointer to Connect Node
		Class 3 (Schematic)
10	1	Back pointers required
11	1	Ordered
12	1	One item per entry
13	1	Pointer to geometry
		Class 4 (Physical Layout)
14	1	Back pointers required
15	1	Ordered
16	1	One item per entry
17	1	Pointer to geometry

## F.6 SIGNAL STRING ASSOCIATIVITY (TYPE 402, FORM 8)

### DESCRIPTION

#### Directory Entry

Entity Type Number: 402

Form Number: 8

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
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 physical signal string
5	SIG1	String	Signal name
:	:	:	
4+NS	SIGNS	String	Signal name
5+NS	PC1	Pointer	Pointer to the DE of the first Connect Node Entity
:	:	:	
4+NS+N1	PCN1	Pointer	Pointer to the DE of the last Connect Node Entity
5+NS+N1	P S 1	Pointer	Pointer to the DE of the first entity in schematic logical signal string
:	:	:	
4+NS+N1	PSN2	Pointer	Pointer to the DE of the last entity in schematic logical signal string
+N2			
5+NS+N1	PP1	Pointer	Pointer to the DE of the first entity in physical signal string
+N2			
:	:	:	
4+NS+N1	PPN3	Pointer	Pointer to the DE of the last entity in physical signal string
+N2+N3			

Additional pointers as required (see Section 2.2.4.5.2).

**F.7 Text Node Associativity (Type 402, Form 10)**

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 multiply 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 exist.

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 one or more General Notes attached 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 override 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 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**

<b>Index</b>	<b>Set Value</b>	<b>Meaning</b>
1	2	Two classes
		Class 1 (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)
		Class 2 (Text Description)
6	2	Back pointers not required
7	1	Ordered class
8	7	Seven items/entry
9	2	Box width
10	2	Box height
11	3	Font code characteristic
12	2	Slant angle
13	2	Rotation angle
14	2	Mirror flag
15	2	Rotate internal text flag

**DESCRIPTION****Directory Entry**

**Entity Type Number: 402**  
**Form Number: 10**

**Parameter Data**

<b>Index</b>	<b>Name</b>	<b>Type</b>	<b>Description</b>
1	NP	Integer	Number of geometry pointers
2	NTD	Integer	Number of Text Descriptions (NTD=1)
3	GP1	Pointer	Pointer to the DE of the point entity (original location)
4	GP2	Pointer	Pointer to the DE of the instance point entity (first instance)
:	:	:	
NP+2	GPNP	Pointer	Pointer to the DE of the instance point entity (NP- 1 instance)
NP+3	WT	Real	Box width
NP+4	HT	Real	
NP+5	FC	Integer	Font code characteristic (default = 1) or pointer
NP+6	SL	Real	Slant angle of text in radians. $\pi/2$ is the value for no slant angle and is the default value.
NP+7	A	Real	Rotation angle in radians for text.
NP+8	M	Integer	Mirror flag (0=no mirror, 1=YT mirror axis, 2=XT mirror axis.)
		Integer	Rotate internal text flag (0=text horizontal, 1=text vertical)

Additional pointers as required (see Section 2.2.4.5.2).

## F.8 CONNECT NODE ASSOCIATIVITY (TYPE 402, FORM 11)

### F.8 Connect Node Associativity (Type 402, Form 11)

The purpose of the Connect Node is to imply a logical connection between 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 two-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

<u>Index</u>	<u>Set</u>	<u>Value</u>	<u>Meaning</u>
1		2	Two classes
			Class 1 (Geometry Pointers)
2		1	Back pointers required
4		1	One item per entry
5		1	Pointer (to Point Entity)
			Class 2 (Connection entities)
6		2	Back pointers not required
7		2	Unordered class
8		1	One item per entry
9		2	Item is value

## F.8 CONNECT NODE ASSOCIATIVITY (TYPE 402, FORM 11)

### DESCRIPTION

#### Directory Entry

Entity Type Number: 402

Form Number: 11

### Parameter Data

<u>Index</u>	<u>Name</u>	<u>Type</u>	<u>Description</u>
1	NC	Integer	Number of pointers (to points)
2	NP	Integer	Number of entries in second class
3	PT1	Pointer	Pointer to the DE of the defining point entity (original location)
4	PT2	Pointer	Pointer to the DE of the instance point entity (first instance)
:	:	:	
NC+2	PTNC	Pointer	Pointer to the DE of the last instance point entity (NC - 1 instance)
NC+3	DT1	Data	First data entry
:	:	:	
NC+NP+2	DTNP	Data	Last data entry

Additional pointers as required (see Section 2.2.4.5.2).

## F.9 REGION FILL PROPERTY (TYPE 406, FORM 4)

### F.9 Region Fill Property (Type 406, Form 4)

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.

#### DESCRIPTION

##### Directory Entry

Entity Type Number: 406

Form Number: 4

#### Parameter Data

Index	Name	Type	Description
1	NP	Integer	Number of property values (NP=2)
2	FC	Integer	Fill code: 0=solid fill 1=unfill ( <i>i.e.</i> , a gap in solid fill) 2=meshed fill  Use of Fill Code = 2 indicates that an associativity is used to link the fill area with its fill mesh description. Using the associativity will allow the implementation of this obsolete method. The recommended method of mesh fill is to use the <a href="#">Type 230 Sectioned Area Entity</a> .
3	0	Pointer	Obsolete. Note: a previous erroneous implementation of this parameter was as a pointer to the DE of a Section Entity defining linear segments of meshed fill. This previous implementation would be indicated by a non-zero value.

Additional pointers as required ([see Section 2.2.4.5.2](#)).

## Appendix G. Parallel Projections from Perspective Views

For those CAD systems that support only parallel projections we recommend using the view reference point, the view up vector, and the view plane normal to construct an analogous view transformation matrix. The process for constructing a suitable transformation matrix is as follows.

1. Perform a translation so that the view reference point becomes the origin. ECO630
2. Perform a rotation so that the view plane normal becomes the positive Z-axis. ECO630
3. Perform a rotation so that the projection of the view up vector onto the view plane becomes the positive Y-axis. ECO630

The 4x4 transformation matrix for translating the view reference point to the origin is:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -VRP_x & -VRP_y & -VRP_z & 1 \end{bmatrix}$$

A rotation matrix can be constructed that transforms the view plane normal to the positive Z-axis ECO630 and the projected view up vector to the positive Y-axis.

Let normalized view plane normal be

$$r_z = \frac{VPN}{\|VPN\|} = Z'$$

Let cross product of the view up vector with the view plane normal be

$$r_x = \frac{VUP \times VPN}{\|VUP \times VPN\|} = X'$$

Let cross product of Z' with X' be

$$r_y = r_z \times r_x = Y'.$$

Then the resulting rotation matrix for constructing the view coordinate system is:

$$R = \begin{bmatrix} r_{1x} & r_{1y} & r_{1z} & 0 \\ r_{2x} & r_{2y} & r_{2z} & 0 \\ r_{3x} & r_{3y} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## **G. PARALLEL PROJECTIONS FROM PERSPECTIVE VIEWS**

The final transformation matrix is formed by multiplying the translational matrix by the rotational matrix. That is, the general transformation matrix used for creating a parallel view based on the original perspective view parameters is:

$$T \times R.$$

## Appendix H. Deprecated Binary Form

The formats defined in [Section 2.2](#) and [Section 2.3](#), referred to collectively as the ASCII Form, have character oriented record lines. This Appendix describes a deprecated bit stream binary representation of data used as an alternative format to the ASCII Form. The binary representation of data, including ASCII characters, is organized in multiples of 8-bit bytes.

This data is transportable by user selected communication protocols with the data treated as “transparent” or bit stream data. All entity parameterizations and data organization are otherwise identical to the ASCII Form.

### H.1 Constants

The following constants need to be represented in the Binary Form:

- Integer numbers
- Real numbers
- String constants
- 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.

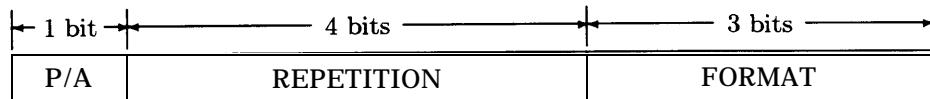
The repetition portion of the control byte is unsigned and biased by 1 so that the true quantity of numbers to which the repetition field applies is one more than the unsigned value of the field.

The format of the control byte is shown in [Figure H1](#).

**H.1.1 Integer Numbers.** 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 H2](#).

Two lengths,  $I$ , of integer data can be selected by the system which generates the file.

The length of single precision data is  $I_s$  and the length of double precision data is  $I_d$ , defined in [Section H.2.1](#).

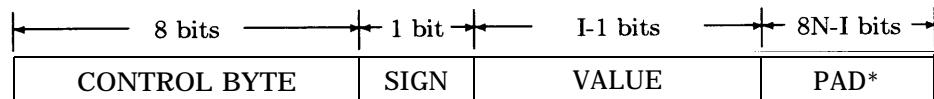


P/A = 0 If only the first of a set of repeated values is physically present  
 = 1 If all expected values are physically present

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

Figure H1. Format of the Control Byte Used in the Binary Form



\*The PAD of zeroes from 1 to 7 bits is included only if the length I of the integer number is not a multiple of 8 bits

Figure H2. Format of an Integer Number in the Binary Form

**H.1.2 Real Numbers.** The structure of a real number shall be a sign bit followed by a biased exponent value of NX bits which is a power of 2 and a binary fraction of NF bits. (NX and NF are defined in [Section H.2.1](#).) 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 indicates 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  $B=2^{**}(NX-1)$ .

The fraction field consists of NF bits interpreted as the low order bits of a normalized (NF+1)-bit fraction part, F. The fraction lies between 0.5 (inclusive) and 1.0 (exclusive). Since the most significant bit of a normalized fraction is always 1, it is not explicitly represented.

Numbers with a nonzero biased exponent have a value given by:

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

The structure of a real number is shown in [Figure H3](#).

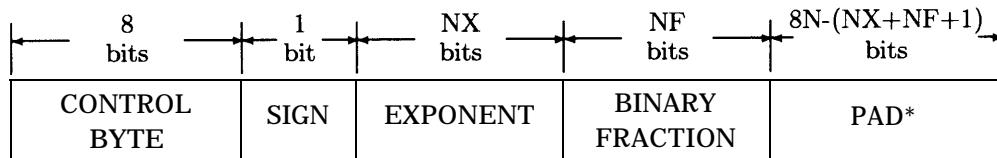
Two lengths of real data can be selected by specifying the length of each exponent (NX) and the length of each fractional portion (NF).

**H.1.3 String Constants.** Following the control byte will be a character count with a length of  $I_s$ , defined in [Section H.2.1](#). Where the character count exceeds the capability of an  $I_s$  length integer, the 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 string constant is shown in [Figure H4](#).

**H.1.4 Pointers.** 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 context. For example, a defining matrix value of zero would imply that the identity rotation matrix and zero translation vector are used. This case might also be handled by using the control byte to indicate a default value.

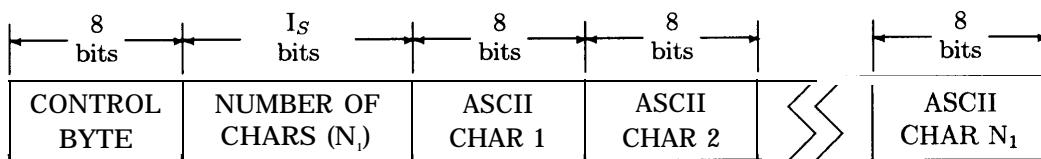
**H.1.5 Language Constants.** Language constants are the string constants of the Macro Definition Entity which, in the ASCII Form, are not preceded by nH and are terminated with a record delimiter. In the Binary Form, the format of language constants will be identical to string constants. Each language constant (Macro Statement) will be an individual string constant.



\*The PAD of zeroes from 1 to 7 bits is included only if the length  $NX+NF+1$  of the floating point number is not a multiple of 8 bits

Figure H3. Format of a Real Number in the Binary Form

For  $N_1 > 0$



For  $N_1 < 0, \dots, N_K < 0, N_R > 0$

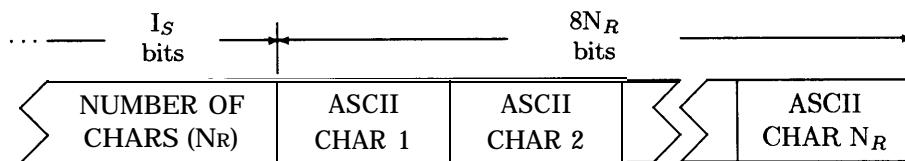
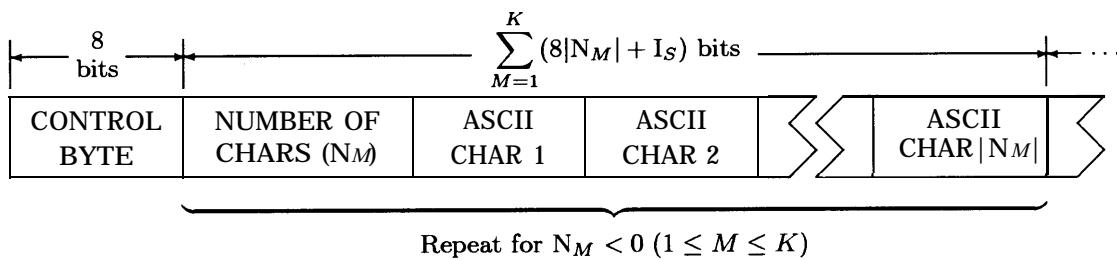


Figure H4. Structure of a String Constant in the Binary Form

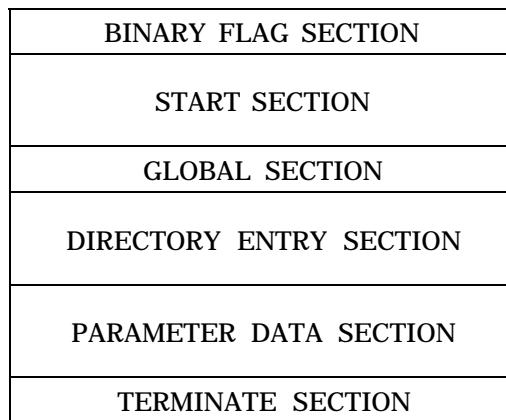


Figure H5. General File Structure in the Binary Form

## **H.2 File Structure**

The general file structure is shown in [Figure H5](#) and comprises the following six sections:

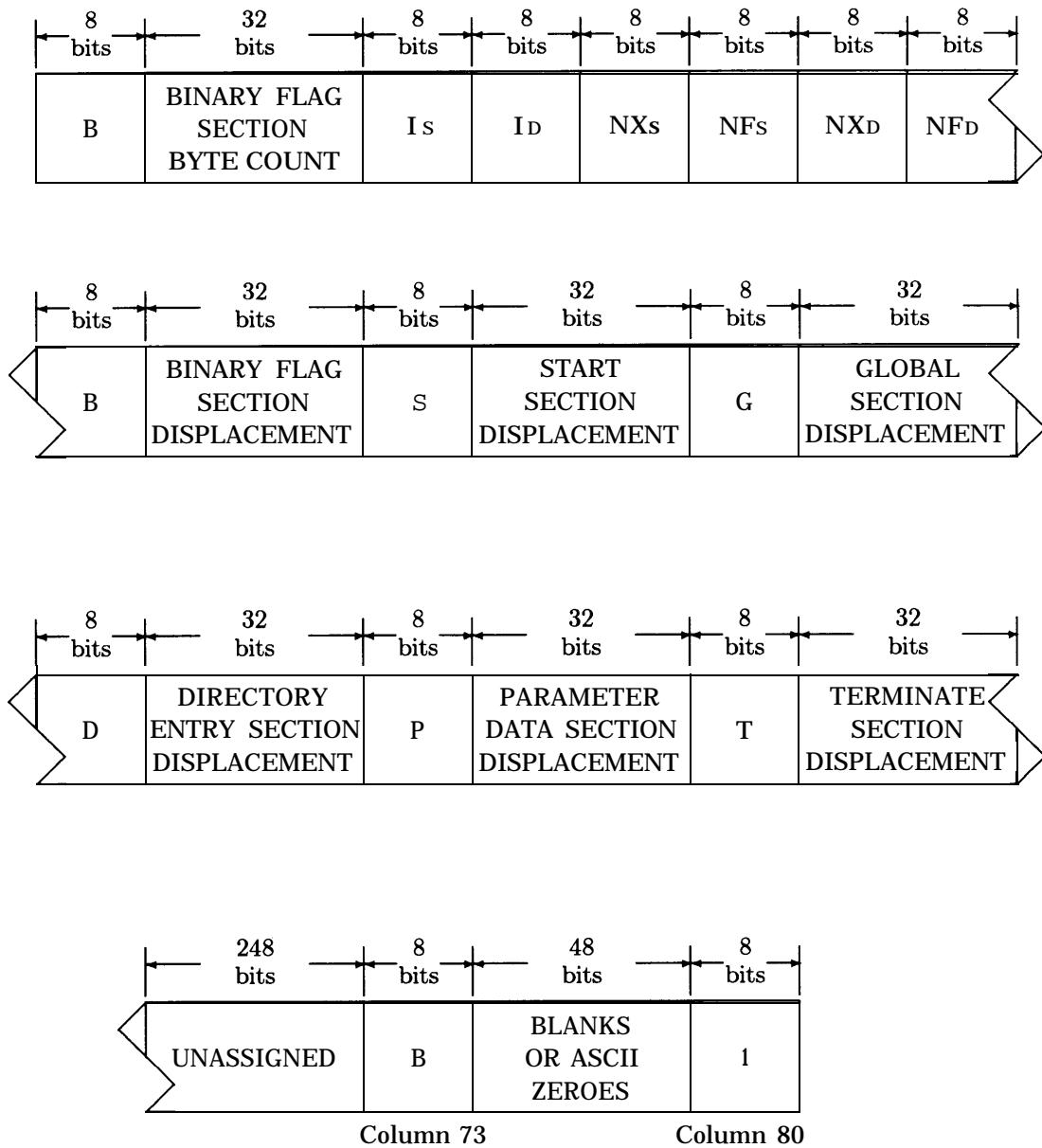
- Binary Flag Section
- Start Section
- Global Section
- Directory Entry Section
- Parameter Data Section
- Terminate Section

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

Following the Terminate Section of the 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.

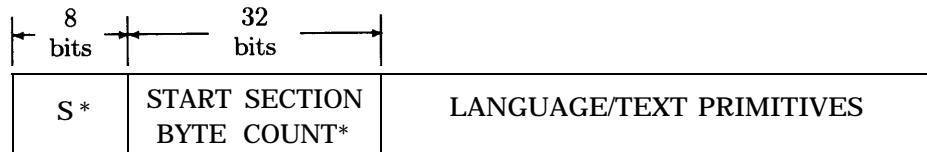
**H.2.1 Binary Flag Section.** The format of the Binary Flag Section is shown in [Figure H6](#). The Binary Flag Section contains a letter code indicating that the file is in Binary Form and also contains information required by a postprocessor to decode the file. (In previous versions of this Specification, this section was called the Binary Information Section.) The Binary Flag Section comprises the following data items, all of which are integers unless otherwise specified:

- Binary Flag Section identifier consisting of the ASCII letter B.
- Binary Flag Section byte count. This byte count (a 32 bit unsigned integer) 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 real primitives.
- Length  $NF_s$  of binary fraction of single precision real primitives.
- Length  $NX_d$  of exponent of double precision real primitives.
- Length  $NF_d$  of binary fraction of double precision real primitives.
- ASCII letter B.
- Binary Flag Section displacement. This is the byte count of the total length of the Binary Flag Section including all null padding characters. This length is the actual length from the initial B of the Binary Flag 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.
- ASCII letter D.
- Directory Entry Section displacement. This is the byte count of the total length of the Directory Entry 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.



NOTE: No fields in the Binary Flag Section have control bytes

Figure H6. Format of the Binary Flag Section in the Binary Form



\*These fields do not have control bytes

Figure H7. Format of the Start Section in the Binary Form

- 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 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 Flag Section are similar in format to the section identification of the ASCII Form 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 (or compressed ASCII if a C). If it contains a B, it is binary.

**H.2.2 Start Section.** The format of the start section is shown in [Figure H7](#). It comprises 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 the ASCII Form. There is no required physical correspondence between the ASCII Form and language/text primitives. One language/text primitive may contain the equivalent of several complete or partial ASCII records. 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.

**H.2.3 Global Section.** The format of the Global Section is shown in [Figure H8](#). The Global Section comprises the following data items:

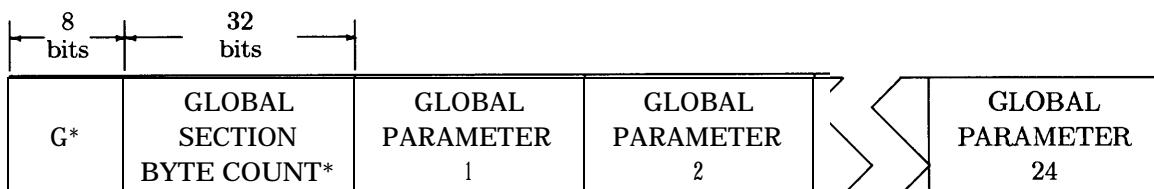
- Global Section identifier consisting of the ASCII letter G.
- Global Section byte count. This byte count excludes the 5 bytes required for the Global Section identifier and the section byte count. This byte count also excludes any null padding characters.
- 24 global parameters.

Control bytes apply to only the 24 global parameters.

The global parameters have the same sequence and meaning as the ASCII Form Global Parameters with the exception that Global Parameters 1 (parameter delimiter character), 2 (record delimiter), 7 (number of bits for integer representation), 8 (single precision magnitude), 9 (single precision significance), 10 (double precision magnitude), and 11 (double precision significance) shall be ignored in binary form. The Binary Flag Section shall supersede these global parameters.

**H.2.4 Directory Entry Section.** The format of the Directory Entry Section is shown in [Figure H9](#). The Directory Entry Section comprises 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 the length in bytes including control bytes, of the subsequent 16 data fields
    - entity type number
    - parameter data
    - structure
    - line font pattern
    - level
    - view



\*These fields do not have control bytes

Figure H8. Format of the Global Section in the Binary Form

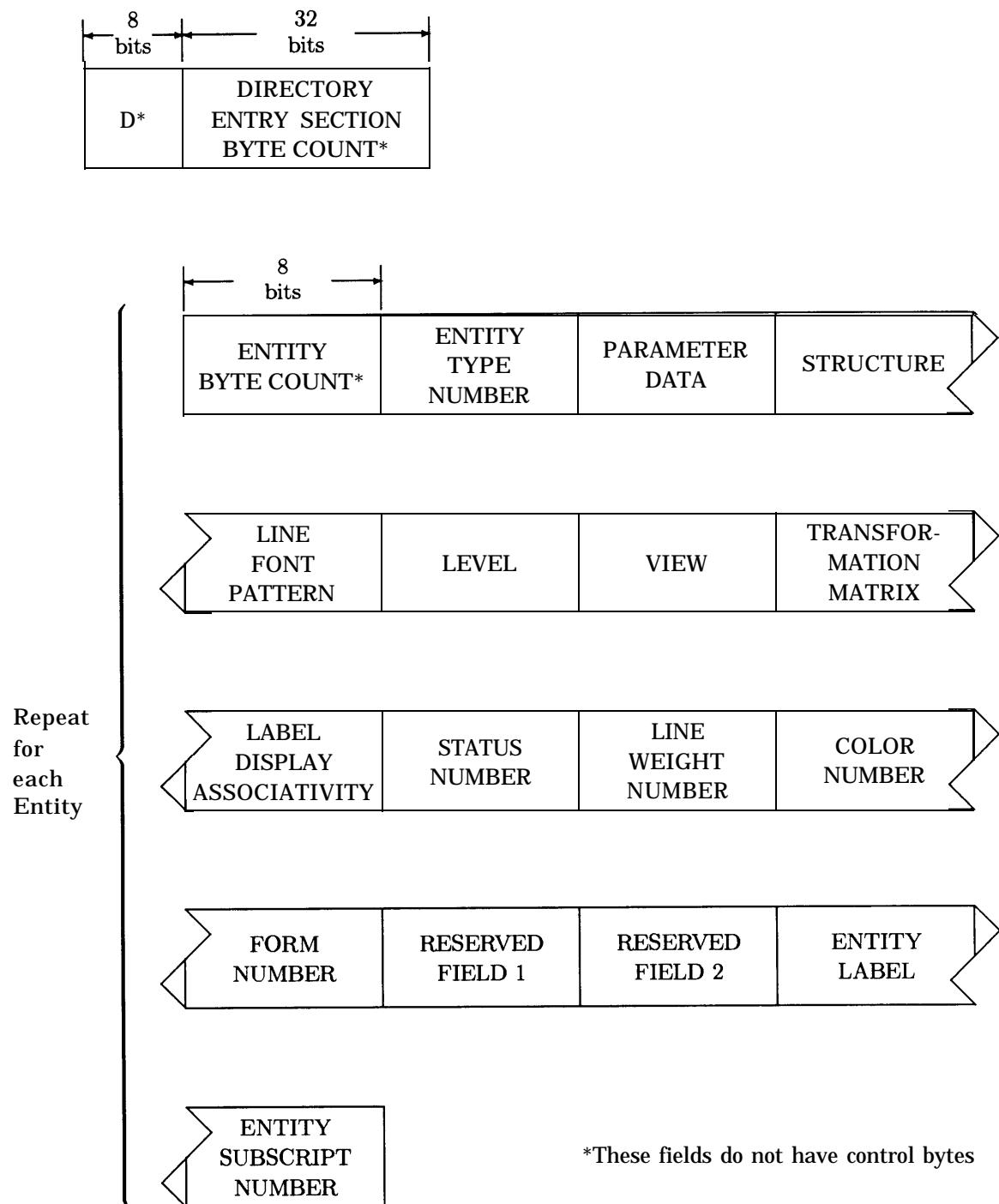
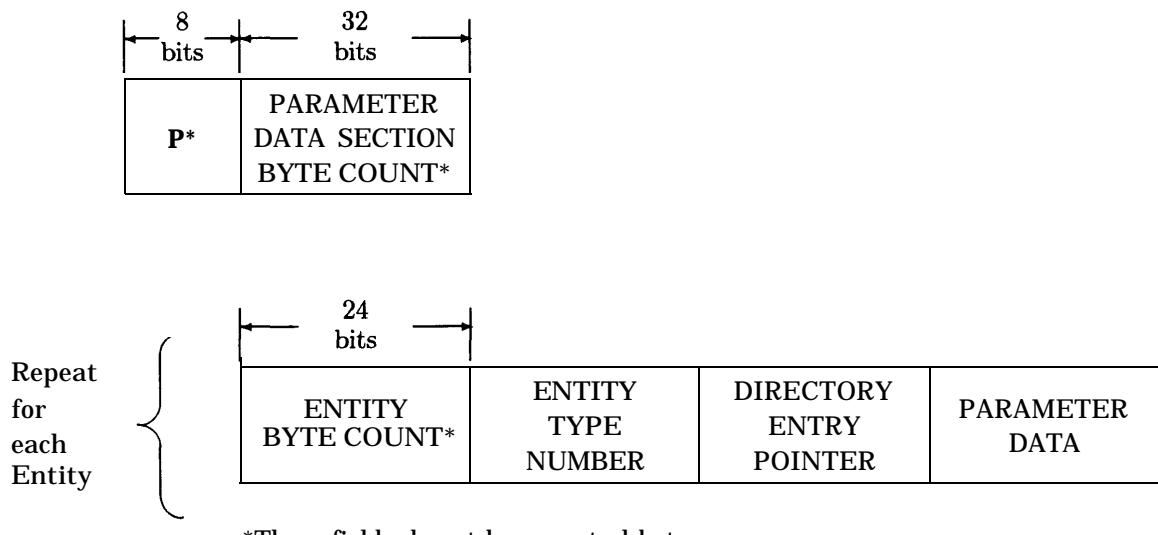


Figure H9. Format of the Directory Entry (DE) Section in the Binary Form



\*These fields do not have control bytes

Figure H10. Format of the Parameter Data (PD) Section in the Binary Form

- transformation matrix
- label display associativity
- status number
- line weight number
- color number
- form number
- reserved field 1
- reserved field 2
- entity label
- entity subscript number

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 fields in the ASCII Form. Within a single file, the length of the 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.

**H.2.5 Parameter Data Section.** The format of the Parameter Data Section is shown in [Figure H10](#). The Parameter Data Section comprises 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.

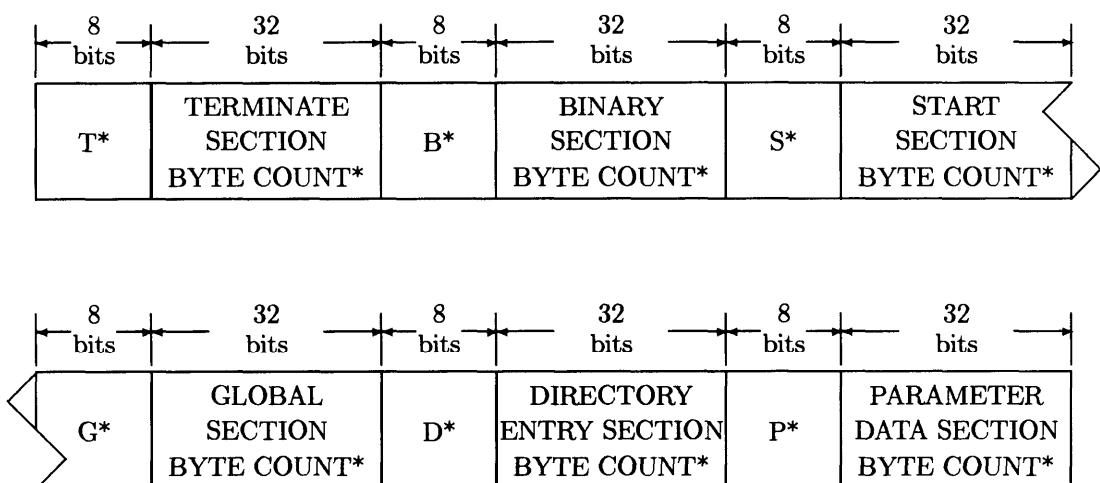
- For each Parameter Data entry, the following data fields are required:
  - entity byte count, which is composed of the lengths, including control bytes, of all subsequent data fields for this entity
  - entity type
  - Directory Entry pointer (relative to Directory Entry section)
  - Parameter Data.

Control bytes apply only to the entity type, Directory Entry pointer and Parameter Data fields.

The Parameter Data entry fields, except for the entity byte count, are identical to and have the same sequence as the ASCII Form.

**H.2.6 Terminate Section.** The format of the Terminate Section is shown in [Figure H11](#). The Terminate Section comprises 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 Flag 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.



\*These fields do not have control bytes

Figure H11. Format of the Terminate Section in the Binary Form

## **Appendix I. Manifold Solid B-Rep Objects**

The boundary representation of a Manifold Solid B-Rep Object (MSBO) utilizes a graph of edges ECO630 and vertices embedded in a connected, oriented, bounded, closed 2-manifold surface called a shell. The embedded graph divides the surface into arcwise-connected areas known as faces. The edges and vertices, therefore, form the boundaries of the faces. The embedded graph may be disconnected. Since the graph is labeled, each entity in the graph has a unique identity.

When a tunnel is drilled through a three-dimensional volume, the corresponding operation on the two-dimensional surface which is the boundary of the volume is *adding a handle*. This can be thought of as cutting out two disks and connecting their boundaries with a cylindrical tube. For example, adding a handle to a sphere produces a torus. Adding a second handle gives a double torus, etc. The number of handles in a surface is the genus, denoted  $H$ .

### **Euler Relations**

Various equalities and inequalities relating topological properties of entities are derived from the ECO630 invariance of a number known as the Euler characteristic. Typically these may be used as checks on the integrity of the topological structure. A violation of an Euler condition signals an impossible MSBO. Systems may perform a validity check on manifold solid boundary representations of objects using the following form of the Euler formula:

$$V - E + 2F - L - 2(S-H) = 0$$

where  $V, E, F, L, S$  are the numbers of distinct vertices, edges, faces, loops, and shells.  $H$  is the sum of the genera of the shells. If the value for  $H$  is not known, the above equation may be transformed into an inequality which can be used as a necessary condition for the validity of the MSBO.

The following figures attempt to illustrate the conceptual representation of the MSBO. [Figure I1](#) ECO630 illustrates how the model handles the cylinder with planar capping surfaces. [Figure I2](#) illustrates the sphere's representation, and the Euler formula for the sphere. [Figure I3](#) illustrates the Euler formula for the torus.

## I. MANIFOLD SOLID B-REP OBJECTS

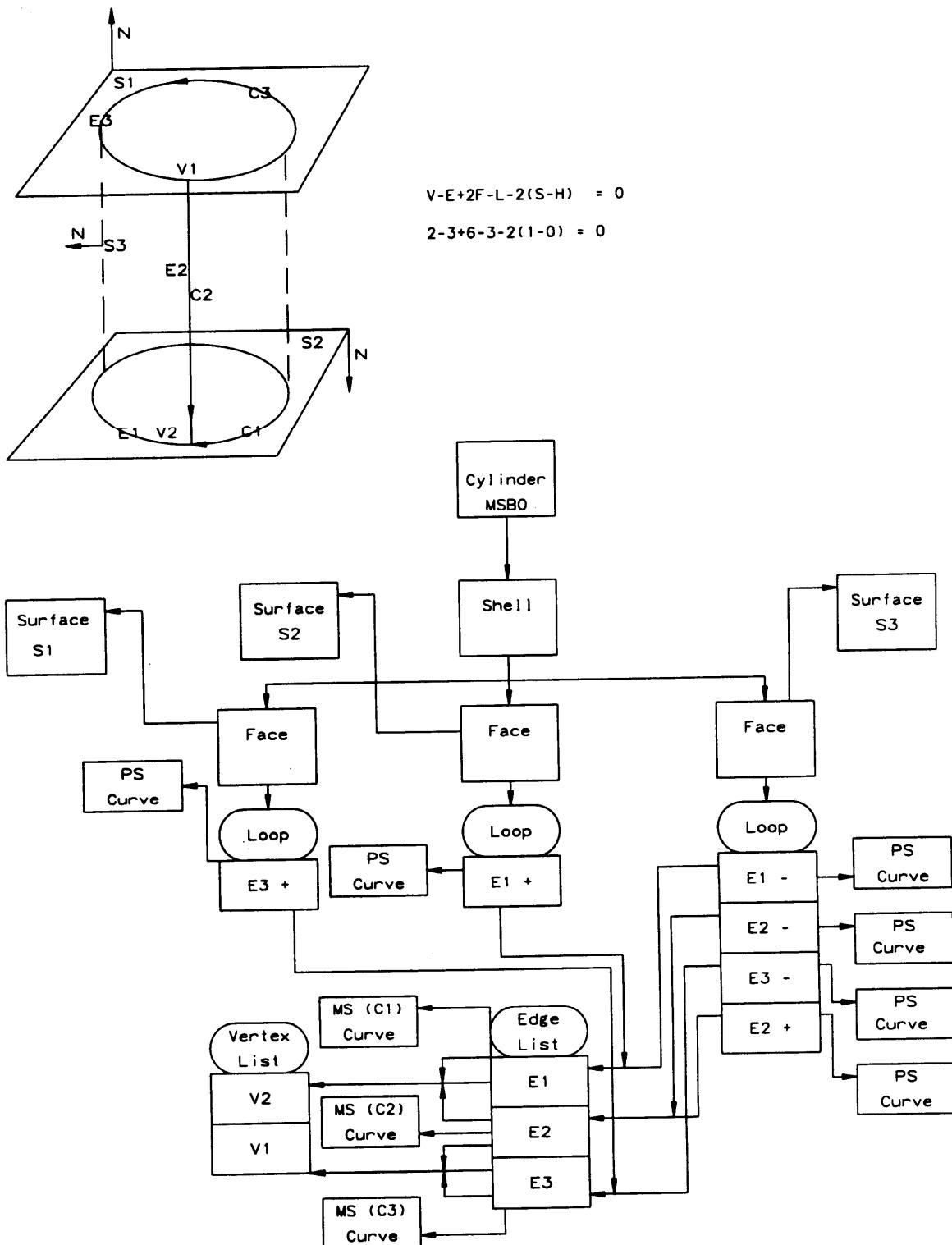


Figure I1. One possible MSBO and the Euler formula of a cylinder with capping planar surfaces.

## I. MANIFOLD SOLID B-REP OBJECTS

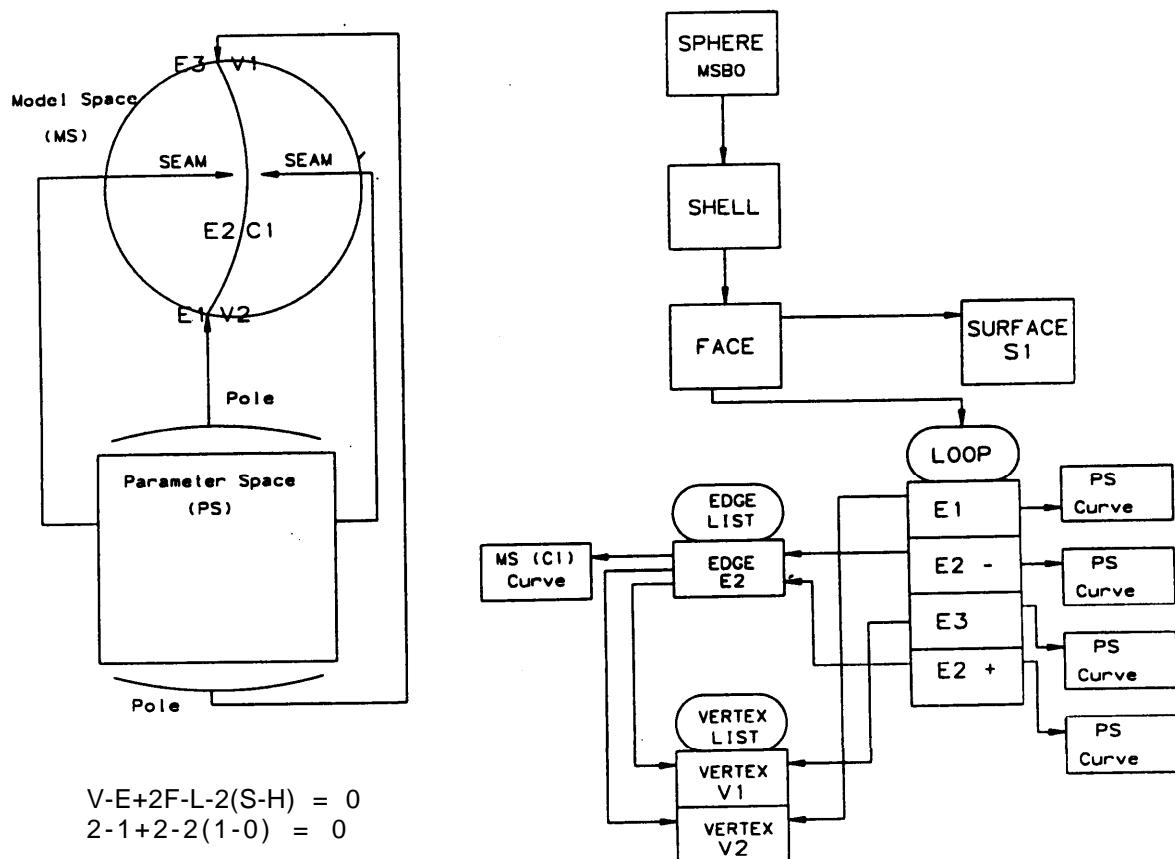
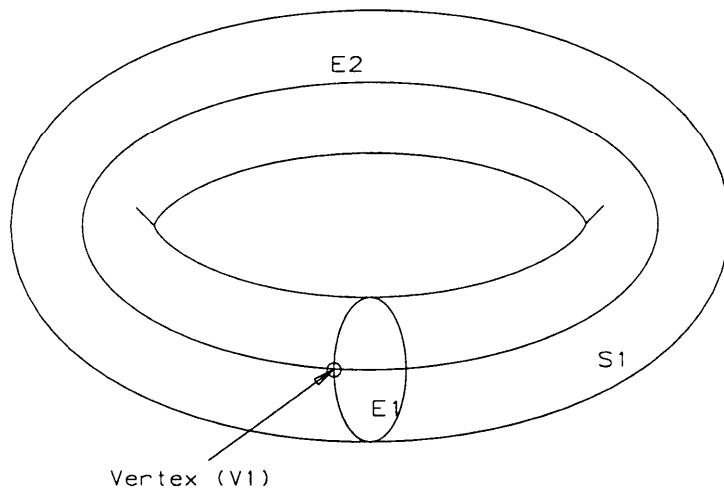


Figure I2. One possible MSBO representation and the Euler formula of a sphere.

## I. MANIFOLD SOLID B-REP OBJECTS



$$V - E + 2F - L - 2(S - H) = 0$$

$$1 - 2 + 2 - 1 - 2(1 - 1) = 0$$

Figure 13. Euler formula of a Torus.

## **Appendix J. List of References**

- [ANSI68] *Code for Information Interchange (X3.4 -1968)*, American National Standards Institute, 1968.
- [ANSI72] *Graphic Symbols for Railroad Maps and Profiles (Y32.7-1972)*, American National Standards Institute, 1972.
- [ANSI77] *Code for Information Interchange (X3.4-1977)*, American National Standards Institute, 1977.
- [ANSI78] *Programming Language FORTRAN (X3.9-1978)*, American National Standards Institute, 1978.
- [ANSI79] *Line Conventions and Lettering (Y14.2M-1979)*, American National Standards Institute, 1979.
- [ANSI79a] *Graphical Symbols for Pipe Fittings, Valves, and Piping (Z32.2.3-1979)*, American National Standards Institute, 1979.
- [ANSI81] *Digital Representation for Communication of Product Definition Data*, Parts 1, 2, and 3, (Y14.26M- 1981), American National Standards Institute, 1981. Permanently out of print.
- [ANSI82] *Dimensioning and Tolerancing, (Y14.5M- 1982)*, American National Standards Institute, 1982.
- [ANSI85] *Computer Graphics-Graphical Kernel System (GKS), Functional Description, (X3.124-1985)*, American National Standards, 1985.
- [ASME87] *Digital Representation for Communication of Product Definition Data*, (ASME Y14.26M- 1987), The American Society of Mechanical Engineers or the American National Standards Institute, 1987.
- [ASME89] *Digital Representation for Communication of Product Definition Data*, (ASME Y14.26M- 1989), The American Society of Mechanical Engineers or the American National Standards Institute, 1989.
- [CH84] Charles Hamilton, *A Guide to Printed Circuit Board Design*, Butterworths, 1984. ECO651
- [DEB078] deBoor, C., *A Practical Guide to Splines*, Springer-Verlag, 1978.
- [DOCA76] DoCarmo, M. P., *Differential Geometry of Curves and Surfaces*, Prentice Hall, 1976.
- [FAR188] Farin, G., *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, 1988.
- [FAUX79] Faux, I., and M. J. Pratt, *Computational Geometry for Design and Manufacture*, John Wiley and Sons, 1979.

## J. LIST OF REFERENCES

- [**FUHR81**] Fuhr, R and Smith, R, *Boeing Advanced Geometry Communications Requirements*, Presented to IGES Extensions and Repairs Committee August 13, 1981. ECO630
- [**GORD74**] Gordon, W. J. and R. F. Riesenfeld, "B-Spline Curves and Surfaces", published in Barnhill, R. E. and R. F. Riesenfeld, ed., *Computer Aided Geometric Design*, Academic Press, 1974.
- [**HILD76**] Hildebrand, F., *Advanced Calculus for Applications*, Prentice Hall, 1976.
- [**HON80**] Hon, R. W., and C. H. Sequin, *A Guide to LSI Implementation*, SSL 79-7, Xerox Palo Alto Research Center, January 1980.
- [**IGES95**] *Operating Procedures and Life Cycle Documentation for The Initial Graphics Exchange Specification* Unpublished; for copy, contact IGES/PDES Organization Administrative Office.
- [**IEEE75**] *Reference Designators for Electrical and Electronics Parts and Equipment*, (IEEE Std 200-1975), Institute of Electrical and Electronics Engineers, 1975.
- [**IEEE76**] *An American National Standard ASTM/IEEE Standard Metric Practice* (IEEE Std 268-1976), Institute of Electrical and Electronics Engineers, 1976.
- [**IEEE84**] *Standard Dictionary of Electrical and Electronics Terms*, (ANSI/IEEE Standard 100-1984), Institute of Electrical and Electronics Engineers, 1984.
- [**IEEE85**] *Standard for Binary Floating-Point Arithmetic* (ANSI/IEEE Std 754-1985), Institute of Electrical and Electronics Engineers, 1985.
- [**IEEE260**] *IEEE Standard Letter Symbols for Units of Measurement* (ANSI/IEEE Std 260), Institute of Electrical and Electronics Engineers, 1978.
- [**IITR68**] *APT Computer System Manual: Volume 2 - Subroutine Library*, Illinois Institute of Technology Research Institute, 1968.
- [**IPCT85**] *Terms and Definitions for Interconnecting and Packaging Electronic Circuits* (ANSI/IPC-T-50C), Institute for Interconnecting and Packaging Electronic Circuits, Revision C, March 1985.
- [**ISHM82**] *Hybrid Microcircuit Design Guide*, ISHM-1402/IPC-H-855 The International Society for Hybrid Microcircuits, Reston, Virginia, October 1982. ECO651
- [**ISO1073**] *Alphanumeric Character Sets for Optical Recognition - Part II: Character Set OCR-B - Shapes and Dimensions of the Printed Image*, (ISO1073/II), International Organization for Standardization, 1976.
- [**ISO7942**] *Information Processing, Graphical Kernel System (GKS), Functional Description*, (IS07942-1985), International Organization for Standardization, 1985.
- [**ISO8859**] *Information Processing-8-Bit Single-Byte Coded Graphic Character Sets-Part 1: Latin Alphabet No. 1*, International Organization for Standardization, 1987.
- [**JIS6226**] *Code of the Japanese Graphic Character Set for Information Interchange*, (JIS C 6226- 1983), Japan Institute for Standardization, 1983.
- [**JOBL78**] Joblove, G. H. and D. Greenberg, "Color Spaces for Computer Graphics", *SIGGRAPH Proceedings*, 1978.
- [**KAPL52**] Kaplan, W., *Advanced Calculus*, Addison-Wesley, 1952.

## J. LIST OF REFERENCES

- [MIL12] *Abbreviations for Use on Drawings, Specifications, Standards, and in Technical Documents* (MIL-STD-12D), U.S. Department of Defense, May 1981.
- [MIL133] *Parameters to be Controlled for the Specification of Microcircuits* (MIL-STD-1331), U.S. Department of Defense, August 1970.
- [MIL195] *General Specification for Semiconductors* (MIL-STD-19500G), U.S. Department of Defense, August 1987.
- [NBS80] *Initial Graphics Exchange Specification (IGES), Version 1.0*, NBSIR 80-1978 (R), U.S. National Bureau of Standards, 1980. Out of print.
- [NBS83] *Initial Graphics Exchange Specification (IGES), Version 2.0*, NBSIR 82-2631 (AF), U.S. National Bureau of Standards, 1982. Available from the National Technical Information Service (NTIS) as PB83-137448.
- [NBS86] *Initial Graphics Exchange Specification (IGES), Version 3.0*, NBSIR 86-3359, U.S. National Bureau of Standards, 1986. Available from the National Technical Information Service (NTIS) as PB86-199759.
- [NBS88] *Initial Graphics Exchange Specification (IGES), Version 4.0*, NBSIR 88-3813, U.S. National Bureau of Standards, 1988. Available from the National Technical Information Service (NTIS) as PB88-235452.
- [NIST90] *Initial Graphics Exchange Specification (IGES), Version 5.0*, NISTIR 4412, U. S. National Institute of Standards and Technology, 1990.
- [ROGE76] Rogers, D. F. and J. A. Adams, *Mathematical Elements for Computer Graphics*, McGraw-Hill, 1976.
- [SMIT78] Smith, A. R., "Color Gamut Transformation Pairs", *Computer Graphics*, 1978.
- [SPICE] Nagel, L. W., *SPICE2: A Computer Program to Simulate Semiconductor Circuits*, Electronics Research Laboratory Report No. ERL-M520, University of California, 9 May 1975.
- [THOM60] Thomas, G., *Calculus and Analytic Geometry*, Addison-Wesley, 1960.
- [TIL080] Tilove, R. B., and Requicha, A. A. G., "Closure of Boolean Operations on Geometric Entities", *Computer Aided Design*, Vol. 12, No. 5, September 1980.
- [USPR091] *Initial Graphics Exchange Specification (IGES), Version 5.1*, USPRO/IPO, September, 1991.
- [USPR093] *Digital Representation for Communication of Product Definition Data*, USPRO/ IPO-100 IGES 5.2, U. S. Product Data Association, 1993

## **Appendix K. 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.

### **ARCWISE CONNECTED**

A set is arcwise connected if given two points in the set it is possible to join the two points with a curve such that all points of the curve are in the set.

### **ASSEMBLY ([IEEE75])**

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 PREDEFINED 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 ANNOTATION**

In electrical engineering, the practice of changing the unique identifier for components noted by symbols on a schematic to match those assigned actual components when the circuit is packaged.

**BACK POINTER**

A pointer in the parameter data section of an entity pointing to an associativity instance of which it is a member.

**BASIC PART ([IEEE75])**

One piece, or two or more pieces joined together, which are not normally subject to disassembly without destruction of designed use.

**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 sub-sequence of the knot sequence used to specify parametric spline curves. ECO630

**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 B](#) 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, non-intersecting curve which has been divided by two distinct points lying on the curve. ECO630

### 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 representation of a component may be a collection of entities, associativities, and properties.

### COMPOSITE CURVE

A connected curve which is formed by concatenating one or more curve segments.

### CONIC ARC ENTITY

A geometric entity which is a finite connected portion of an ellipse, a parabola, or a hyperbola.

### CONNECT POINT ENTITY

A geometric entity giving the XYZ location and other information (*e.g.*, text labels) of a point of connection. May be independent or subordinate to a Network Subfigure Definition and/or Instance. Used for netlist information.

### CONNECTED CURVE

A curve such that for any two points P1 and P2, one can travel from P1 to P2 without leaving the curve.

### CONNECTED GRAPH

A graph is connected if there is a path between any two vertices.

### 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. As the weights must all be 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 Appendix B](#) for more details.

### COONS PATCH

A surface obtained by transfinite interpolation of boundary curves. Typically the surface is a bicubic polynomial spline.

### COPIOUS DATA ENTITY

A geometric entity sometimes used as an annotation entity, containing arrays of tuples of real numbers to which a specific meaning has been assigned. Each 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.

## K. GLOSSARY

### 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.

### DIRECTORY ENTRY SECTION

The section of an exchange file, consisting of fixed field data items, that forms 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.

### DRILLED HOLE PROPERTY

A predefined property that assigns the physical attribute of a hole that can be made by a drill. May be used in electrical applications to 1) define a via from one printed circuit board, PCB, layer to another, 2) define a plated via hole, and 3) give the first physical drill diameter and/or the finished hole diameters. It is usually attached to a point, circle, subfigure definition, or subfigure instance.

### EDGE VERTEX

A method of geometric modeling in which a two- or three-dimensional object is represented by ECO630 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.

## K. GLOSSARY

### 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.

### EXTERNAL REFERENCE ENTITY

A mechanism for referencing definitions which do not reside in the same exchange file as the instances of those definitions.

### FACE BOUNDARY

Within the context of an MSBO, it is a curve along which the face is joined to another.

### 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.

### FLASH ENTITY

A geometric entity used for photo-plotting apertures and other filled areas. May be used for representing metallic conductive material on a printed circuit board such as pads and traces. Also, may be used in integrated circuit (IC) chip masks.

### FLEXIBLE PRINTED CIRCUIT

An arrangement of printed circuit and components utilizing flexible base materials with or without flexible cover layers.

### FLOW ASSOCIATIVITY

A predefined associativity that represents a flow path. In electrical applications such as schematics and physical descriptions for Printed Wiring Boards, PWB, Printed Circuit Boards, PCB, PCB assemblies, ICs, etc., it presents a common electrical signal (*e.g.*, voltage). In piping applications, it represents a flow path between only one source and sink, but branching is allowed to other Flow Associativities. It provides netlist information for a single flow.

### FONT CHARACTERISTIC

An integer which is used to identify a text font. Font characteristic numbers may be positive which indicate a 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.

## K. GLOSSARY

### 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.

### GENUS

The number of handles in a surface.

### GEOMETRIC

Having to do with the shape information (points, curves, surfaces, and volumes), necessary to represent some object.

### GLOBAL SECTION

The section of an exchange file consisting of general information describing the file, the file generator (preprocessor), and information needed by the file reader (postprocessor).

### GRAPH

A set of vertices and edges which join pairs of vertices (not necessarily distinct). Vertices which are joined by one edge are adjacent. There may be multiple edges connecting the same two vertices.

### GRID

The set of  $(u, v)$  where  $u$  and  $v$  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, shielding, or heat sinking.

### GROUP ASSOCIATIVITY

A predefined associativity for forming any collection of entities.

### HANDLE

When a tunnel is drilled through a three-dimensional volume, the corresponding operation on the two-dimensional surface which is the boundary of the volume is *adding a handle*. This can be thought of as cutting out two disks and connecting their boundaries with a cylindrical tube. For example, adding a handle to a sphere produces a torus.

### 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.

## K. GLOSSARY

HYBRID MICROCIRCUIT ASSEMBLY (HMA)	ECO649 ECO630
An LEP in which components are electrically interconnected on an insulating substrate on which conductors and/or resistors have been previously deposited. The HMA is further classified as either a thin film, thick film, or green tape based on the process used to fabricate the interconnect layers.	
INSTANCE	
A particular occurrence of some item or relationship. Several instances may reference the same item.	
INTEGRATED CIRCUIT (IC)	ECO649
An LEP in which components are electrically interconnected on an insulating substrate through a photolithographic process. A similar process may be used to modify the local substrate properties to create the components.	
KNOT SEQUENCE	
A nondecreasing sequence of real numbers used to specify parametric spline curves.	
LABEL DISPLAY ASSOCIATIVITY	
A predefined 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.	
LAYERED ELECTRICAL PRODUCT (LEP)	ECO649
A specially processed insulating substrate, consisting of one or more layers, which electrically interconnects components that may be mounted on or within the substrate. The LEP is a generic term, which encompass integrated circuits (IC), printed wiring assemblies (PWA), hybrid microcircuit assemblies (HMA), and others.	
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.	
LEVEL FUNCTION PROPERTY	
A predefine property that assigns an “application data base defined functionality” to a level. This property may stand alone ( <i>e.g.</i> , DE status is independent), that is no other entity points to it. Also, see the level field in directory entry.	
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.	

**LINE WIDENING PROPERTY**

A predefined property that overrides the line weight given in the directory entry of an entity by providing a physical value for the actual width. May be used in electrical applications to describe metallization on a printed circuit board such as traces and off board connections. Also, see the FLASH and SECTION entities and the Region Fill property.

**LINEAR DIMENSION**

An annotation entity used to represent a distance between two locations.

**LINEAR PATH ENTITY**

A geometric entity that defines a collection of linear segments that form a path. Also, see [Copious Data Entity Forms 11 and 12](#).

**LIST INDEX**

The index into the list of generic components of the list starting at 1, *i.e.*, List Index 1 refers to the first component of the list.

**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 about an axis.

**MODEL**

A particular collection of data in an exchange file that describes a product.

**MODEL SPACE**

A right-handed three-dimensional Cartesian coordinate space in which the product is represented.

**NEGATIVE BOUNDED PLANAR PORTION**

A hole.

**NETWORK SUBFIGURE DEFINITION ENTITY**

A structure entity used to define a schematic symbol, component or pipe in electrical and piping applications. Shall be used whenever associated Connect Point Entities need to be instanced with the Network Subfigure Instance Entity. For physical components, it may have subordinate entities (copious data, simple closed planar curve, subfigure definition or instance, *etc.*) that have attached a Region Restriction Property giving design rules for auto routing a Printed Circuit Board, PCB. Also, 2-D component outlines and 3-D physical descriptions may be defined.

### NETWORK SUBFIGURE INSTANCE ENTITY

A structure entity used to specify an occurrence of a schematic symbol, component or pipe in electrical and piping applications. It has associated “instanced” Connect Point Entities that give the XYZ model space point of connections. Used in netlist information and part lists.

### NODAL DISPLACEMENT and ROTATIONAL ENTITY

This entity is used to communicate finite element post processing data. It contains the node identifier, original node coordinates, and incremental displacements and rotations for each node for each load case.

### NODAL LOAD/CONSTRAINT ENTITY

An entity used in a finite element model to apply a force, moment, or other loading or constraints at a specific node.

### NODE

A point in space used to define a finite element topology.

### NULL ENTITY

The Null Entity ([Type 0](#)) is intended to be ignored by a processor. A processor should skip over all DE and PD data associated with this entity.

**NULL STRING** The null string is an empty string parameter. This value is valid for any entity ECO630 whose PD section contains a string parameter. Example: For specifying a null string within a General Note Entity ([Type 212](#)), the number of characters parameter (NC) shall be zero, and the Z depth parameter (ZS) shall be followed by two parameter delimiters.

### 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 exchange file consisting of specific geometric or annotative information about the entities or pointers to related entities.

### PARAMETERIZED SURFACE

$S(u, v)$  is a parametric representation of a surface if it meets the following criteria:

The untrimmed domain of  $S(u, v)$  is a rectangle,  $D$ , consisting of those points  $(u, v)$  such that  $a \leq u \leq b$  and  $c \leq v \leq d$  for given constant  $a, b, c$ , and  $d$  with  $a < b$  and  $c < d$ .

The mapping  $S = S(u, v) = (x(u, v), y(u, v), z(u, v))$  is defined for each ordered pair  $(u, v)$  in  $D$ .

It is one-to-one in the interior (but not necessarily on the boundary) of  $D$ .

It has continuous normal vectors at every point of  $D$  except those which map to poles.

### 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 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.

**PART NUMBER PROPERTY**

A predefined property that provides one or more text strings giving one to four distinct part numbers (Generic, MIL-STD, Vendor, and/or Internal) to an entity representing a physical part. May be used in electrical, piping or other applications. Usually, it is attached to a subfigure definition and/or instance that represents the part. May be used for part lists.

**PATCH**

A surface represented by parametric functions of two parameters which can be viewed as blendings of four given boundary curves.

**PATH**

If  $V_0$  and  $V_n$  are vertices of a graph then  $(V_0..V_k..V_n)$  is a path if  $V_k$  and  $V_{k+1}$  ( $k = 0, n - 1$ ) are adjacent.

**PIN NUMBER PROPERTY**

A predefined property that provides a text string giving a component pin number value to an entity representing an electrical component. Also, see the CONNECT POINT Entity.

**PLANE ENTITY**

A geometric entity consisting of all or a portion of a plane.

**PLATED-THROUGH HOLE ([IPCT85])**

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 DIMENSION ENTITY**

An annotation entity consisting of a leader, text, and an optional circle or hexagon enclosing the text.

**POINT ENTITY**

A geometric entity which has no size but possesses a location in space.

**POINTER**

A number that indicates the location of an entity within an exchange file.

**POLE**

Let  $P$  be a point in  $R3$ . Then  $P$  is a pole of the surface defined by the mapping  $S(u, v)$  if any of the following are true:

- $P = S(a, v)$  for all  $v$  such that  $c \leq v \leq d$ ,
- $P = S(b, v)$  for all  $v$  such that  $c \leq v \leq d$ ,
- $P = S(u, c)$  for all  $u$  such that  $a \leq u \leq b$ ,
- $P = S(u, d)$  for all  $u$  such that  $a \leq u \leq b$

**POSITIVE BOUNDED PLANAR PORTION**

The top of a peg.

**POSTPROCESSOR**

A program which translates an exchange file of product definition data from the form defined by this Specification into the data base form of a specific CAD/CAM system.

**PREDEFINED ASSOCIATIVITIES**

Associativities which are defined within this standard.

**PREPROCESSOR**

A program which translates a file of product definition data from the data base form of a specific CAD/CAM system into the form defined by this Specification.

**PRINTED BOARD ([IPCT85])**

The general term for completely processed printed circuit or printed wiring configurations. It includes rigid or flexible, single, double, or multilayer boards.

**PRINTED CIRCUIT ([IPCT85])**

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 ([IPCT85])**

A part manufactured from rigid base material upon which a completely processed printed circuit has been formed.

**PRINTED WIRING ([IPCT85])**

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.

**PRINTED WIRING ASSEMBLY (PWA)**

An assembly (an LEP) of packaged parts used as components placed on a printed wiring board (PWB). The conductive patterns formed on the common base provide the majority of connections among the packaged components. A PWB is distinguished from a printed circuit board by having no printed components formed on the common base. The resulting assembly is an electronic circuit designed to meet some of an electronic system's requirements.

**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 multiplied by only the corresponding 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 multiplied by the corresponding weight.

**REFERENCE DESIGNATOR PROPERTY**

A predefined property that provides a text string giving a component reference designator value to an entity representing an electrical component. Also, see the Network Subfigure entity.

**REGION**

The bounded area enclosed by a closed curve or a combination of curves.

**REGION FILL PROPERTY**

A predefined property that is used to solid fill or unfill (nested) a closed area. May be used for cross-section material representations (*i. e.*, concrete, steel, *etc.*) and artwork. Also, see the SECTION entity. Also, see the FLASH and SECTION entities and the Line Widening property.

**REGION RESTRICTION PROPERTY**

A predefined property that provides design rules in electrical applications. Especially, region restrictions regarding Printed Circuit Board, PCB routing rules for prohibiting or permitting vias and traces under component outlines and placement of components on the printed circuit board.

**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.

**SEAM**

Let C be a curve in R3. Then C is a seam of the surface defined by S(u,v) if it is the image in model space of

$$\begin{aligned}C(v) &= S(a, v) \text{ for all } v \text{ such that } c \leq v \leq d \text{ and} \\C(v) &= S(b, v) \text{ for all } v \text{ such that } c \leq v \leq d\end{aligned}$$

or

$$\begin{aligned}C(u) &= S(u, c) \text{ for all } u \text{ such that } a \leq u \leq b \text{ and} \\C(u) &= S(u, d) \text{ for all } u \text{ such that } a \leq u \leq b\end{aligned}$$

## K. GLOSSARY

### SECTION DISPLAY SYMBOL

An arrangement of fonted straight lines in a repetitive planar pattern at a specified spacing and angle.

### SECTION ENTITY

A pattern used to distinguish a closed region in a diagram. It is represented as a form of the copious data entity.

### SIMPLE CLOSED CURVE

Informally, a simple closed curve is a curve that may be obtained as follows: (1) join together the two ends of an infinitely thin string of appropriate nonzero finite length; (2) situate the resulting loop so that it does not intersect itself.

### SINGLE PARENT ASSOCIATIVITY

A predefined associativity that provides logical grouping of a single parent entity to its many children entities.

### SPLINE

A piecewise continuous polynomial.

### START SECTION

The section of an exchange file containing a human-readable file prologue.

### SUBASSEMBLY ([IEEE75])

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 ([IEEE75])

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.

### TABULAR DATA PROPERTY

The tabular data property provides a structure to accommodate point form data. The basic structure is a two-dimensional array containing data list for dependent and independent variable.

## K. GLOSSARY

### TABULATED CYLINDER ENTITY

A geometric entity which is a surface generated by moving a line segment called the generatrix parallel to itself along a space curve called the directrix.

### TERMINATE SECTION

The final section of an exchange file, indicating the sizes of each of the preceding file sections.

### TEXT DISPLAY TEMPLATE ENTITY

An annotation entity used to define the display location of a text string. In electrical applications, it gives a “relative” mode of location dependent on a connect point for pin number and/or pin function of components (*e.g.*, Integrated Circuit, IC, chip pins). For electrical schematic symbols and physical components, it may give the display location of the reference designator text and/or part number.

### 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 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 ([IEEE75])

A major building block for a set or system, consisting of a combination of basic parts, sub-assemblies, 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 ([IPCT85])

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 nongeometry information.

### VIEWING BOX

The clipping box used to define a view.

### WEIGHT

A positive 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 [Appendix B](#) for details.

## **K. GLOSSARY**

### **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.

ECO630

## **Appendix L. Index of Entities**

‡Entities with this symbol have not been tested. See Section 1.9.

Angular Dimension Entity (Type 202) . . . . .	218
Associativity Definition Entity (Type 302) . . . . .	289
Associativity Group Type Property (Form 23)‡ . . . . .	460
Associativity Instance Entity (Type 402) . . . . .	374
Attribute Table Definition Entity (Type 322) . . . . .	335
Attribute Table Instance Entity (Type 422) . . . . .	510
Basic Dimension Property (Form 31)‡ . . . . .	478
Block Entity (Type 150) . . . . .	174
Boolean Tree Entity (Type 180) . . . . .	192
Boundary Entity (Type 141) . . . . .	157
Bounded Surface Entity (Type 143) . . . . .	165
Centerline Entity (Type 106, Forms 20-21) . . . . .	79
Circular Arc Entity (Type 100) . . . . .	64
Circular Array Subfigure Instance Entity (Type 414) . . . . .	501
Closure Property (Type 406, Form 36)‡ . . . . .	485
Color Definition Entity (Type 314) . . . . .	329
Composite Curve Entity (Type 102) . . . . .	67
Conic Arc Entity (Type 104) . . . . .	71
Connect Point Entity (Type 132) . . . . .	131
Copious Data Entity (Type 106, Forms 1-3) . . . . .	75
Curve Dimension Entity (Type 204)‡ . . . . .	220
Curve on a Parametric Surface Entity (Type 142) . . . . .	162
Definition Levels Property (Form 1) . . . . .	416
Diameter Dimension Entity (Type 206) . . . . .	222
Dimension Display Data Property (Form 30)‡ . . . . .	475
Dimension Tolerance Property (Form 29)‡ . . . . .	472
Dimension Units Property (Form 28)‡ . . . . .	470
Diminished Geometry Associativity (Type 402, Form 13) . . . . .	386
Diminished Geometry Associativity (Type 402, Form 21)‡ . . . . .	403
Diminished Entity (Type 123)‡ . . . . .	112
Drawing Entity (Type 404) . . . . .	409
Drawing Sheet Approval Property (Type 406, Form 32)‡ . . . . .	480
Drawing Sheet ID Property (Type 406, Form 33)‡ . . . . .	481
Drawing Size Property (Form 16) . . . . .	449
Drawing Units Property (Form 17) . . . . .	450
Drilled Hole Property (Form 6) . . . . .	422
Edge Entity (Type 504)‡ . . . . .	516
Edge List Entity (Type 504, Form 1) . . . . .	516
Element Results Entity (Type 148)‡ . . . . .	171

## L. INDEX OF ENTITIES

Ellipsoid Entity (Type 168).....	190
Entity Label Display Associativity (Type 402, Form 5).....	381
External Reference Entity (Type 416).....	503
External Reference File Index Associativity (Form 12).....	385
External Reference File List Property (Form 12).....	445
Face Entity (Type 510)‡.....	520
Finite Element Entity (Type 136).....	137
Flag Note Entity (Type 208).....	224
Flash Entity (Type 125).....	120
Flow Associativity (Form 18).....	393
Flow Line Specification Property (Form 14).....	447
General Label Entity (Type 210).....	227
General Note Entity (Type 212).....	229
General Symbol Entity (Type 228).....	272
Generic Data Property (Form 27)‡.....	468
Group Associativity (Type 402, Form 1).....	376
Group Without Back Pointers Associativity (Form 7).....	383
Hierarchy Property (Form 10).....	426
Highlight Property (Form 20)‡.....	457
Intercharacter Spacing Property (Form 18)‡.....	451
Leader (Arrow) Entity (Type 214).....	257
LEP Artwork Stackup Property (Form 25)‡.....	465
LEP Drilled Hole Property (Form 26)‡.....	466
Level Function Property (Form 3).....	419
Level to LEP Layer Map Property (Form 24)‡.....	462
Line Entity (Type 110, Form 0).....	90
Line Font Definition Entity (Type 304).....	292
Line Font Property (Form 19)‡.....	452
Line Widening Property (Form 5).....	420
Linear Dimension Entity (Type 216).....	262
Linear Path Entity (Type 106, Forms 11-13).....	77
Loop Entity (Type 508)‡.....	518
MACRO Definition Entity (Type 306)‡.....	298
MACRO Instance Entity‡ .....	312
Manifold Solid B-Rep Object Entity (Type 186) ‡.....	197
Name Property (Form 15).....	448
Network Subfigure Definition Entity (Type 320).....	333
Network Subfigure Instance Entity (Type 420).....	508
New General Note Entity (Type 213)‡.....	246
Nodal Displacement and Rotation Entity (Type 138).....	153
Nodal Load/Constraint Entity (Type 418).....	506
Nodal Results Entity (Type 146)‡.....	168
Node Entity (Type 134).....	134
Nominal Size Property (Form 13).....	446
Null Entity (Type 0).....	63
Offset Curve Entity (Type 130).....	129
Offset Surface Entity (Type 140).....	155
Ordered Group with Back Pointers Associativity (Form 14).....	389
Ordered Group, no Back Pointers Associativity (Form 15).....	390
Ordinate Dimension Entity (Type 218).....	264
Overscore Property (Type 406, Form 35)‡ .....	483

## L. INDEX OF ENTITIES

Parametric Spline Curve Entity (Type 112) . . . . .	94
Parametric Spline Surface Entity (Type 114) . . . . .	98
Part Number Property (Form 9) . . . . .	425
Perspective View Entity (Type 410, Form 1)‡ . . . . .	496
Pick Property (Form 21)‡ . . . . .	458
Pin Number Property (Form 8) . . . . .	424
Piping Flow Associativity (Type 402, Form 20)‡ . . . . .	399
Planar Associativity (Type 402, Form 16) . . . . .	391
Plane Entity (Type 108) . . . . .	87
Plane Surface Entity (Type 190) ‡ . . . . .	203
Point Dimension Entity (Type 220) . . . . .	267
Point Entity (Type 116) . . . . .	102
Pre-defined Associativities . . . . .	374
Property Entity (Type 406) . . . . .	415
Radius Dimension Entity (Type 222) . . . . .	269
Rational B-Spline Curve Entity (Type 126) . . . . .	123
Rational B-Spline Surface Entity (Type 128) . . . . .	126
Rectangular Array Subfigure Instance Entity (Type 412) . . . . .	499
Reference Designator Property (Form 7) . . . . .	423
Region Restriction Property (Form 2) . . . . .	417
Right Angular Wedge Entity (Type 152) . . . . .	176
Right Circular Cone Frustum Entity (Type 156) . . . . .	180
Right Circular Conical Surface Entity (Type 194)‡ . . . . .	209
Right Circular Cylinder Entity (Type 154) . . . . .	178
Right Circular Cylindrical Surface Entity (Type 192)‡ . . . . .	206
Ruled Surface Entity (Type 118) . . . . .	104
Section Entity (Type 106, Forms 31-38) . . . . .	81
Sectioned Area Entity (Type 230) . . . . .	275
Segmented Views Visible Associativity (Type 402, Form 19)‡ . . . . .	397
Selected Component Entity (Type 182)‡ . . . . .	195
Simple Closed Planar Curve Entity (Type 106, Form 63) . . . . .	86
Single Parent Associativity (Type 402, Form 9) . . . . .	384
Singular Subfigure Instance Entity (Type 408) . . . . .	488
Solid Assembly Entity (Type 184) . . . . .	196
Solid Instance Entity (Type 430) . . . . .	512
Solid of Linear Extrusion Entity (Type 164) . . . . .	188
Solid of Revolution Entity (Type 162) . . . . .	186
Sphere Entity (Type 158) . . . . .	182
Spherical Surface Entity (Type 196)‡ . . . . .	212
Subfigure Definition Entity (Type 308) . . . . .	322
Surface of Revolution Entity (Type 120) . . . . .	107
Tabular Data Property (Form 11) . . . . .	427
Tabulated Cylinder Entity (Type 122) . . . . .	110
Text Display Template Entity (Type 312) . . . . .	327
Text Font Definition Entity (Type 310) . . . . .	323
Toroidal Surface Entity (Type 198) ‡ . . . . .	215
Torus Entity (Type 160) . . . . .	184
Transformation Matrix Entity (Type 124) . . . . .	113
Trimmed(Parametric) Surface Entity (Type 144) . . . . .	166
Underscore Property (Type 406, Form 34)‡ . . . . .	482
Uniform Rectangular Grid Property (Form 22)‡ . . . . .	459

## L. INDEX OF ENTITIES

Units Data Entity ( <a href="#">Type316</a> )‡ .....	330
Vertex Entity ( <a href="#">Type 502</a> )‡ .....	514
Vertex List Entity ( <a href="#">Type 502, Form 1</a> ) .....	514
View Entity ( <a href="#">Type 410</a> ) .....	490
Views Visible Associativity ( <a href="#">Type 402, Form 3</a> ) .....	377
Views Visible, Color, Line Weight Associativity ( <a href="#">Form 4</a> ) .....	379
Witness Line Entity ( <a href="#">Type 106, Form 40</a> ) .....	84