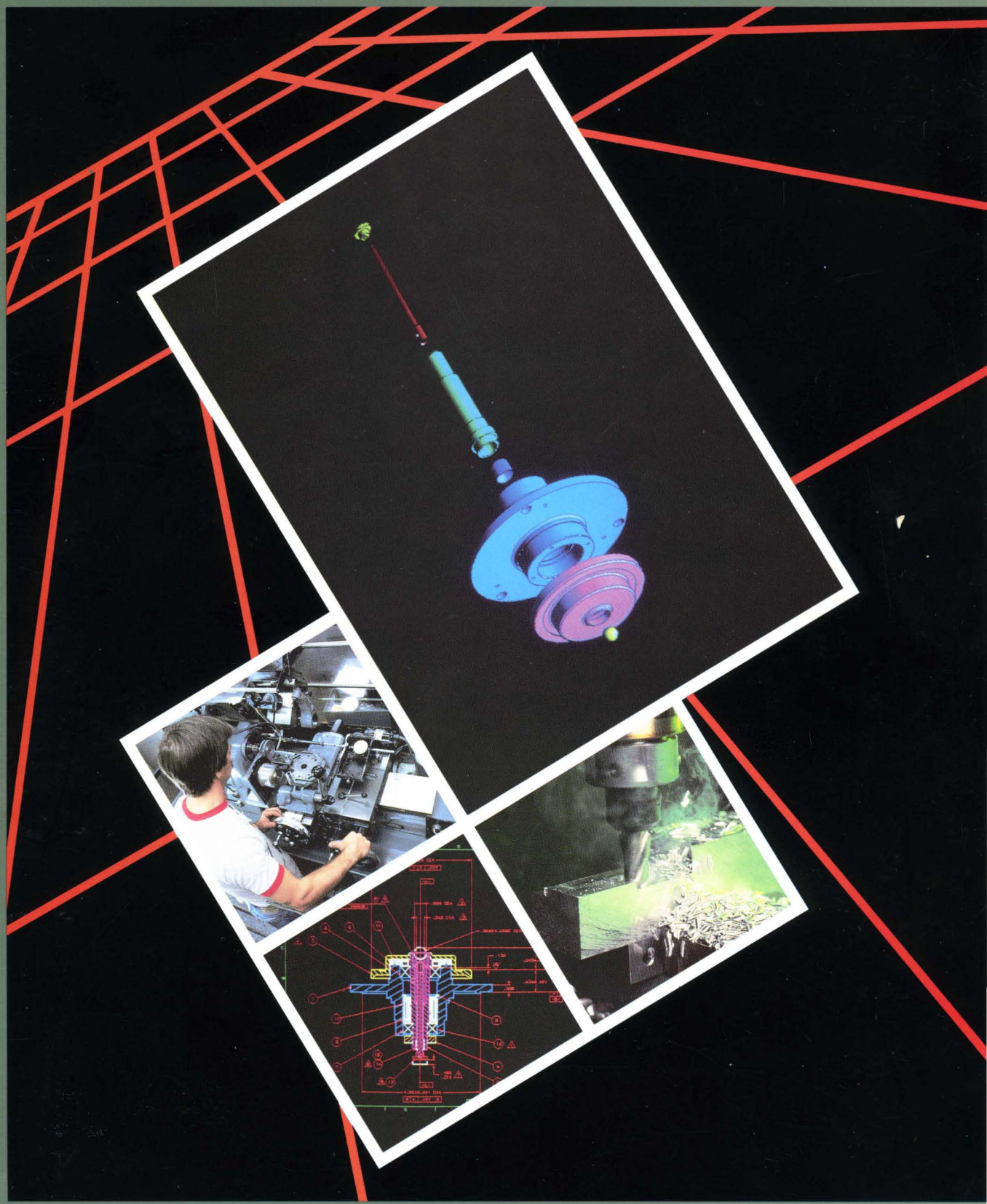
ICEM GPL

for NOS

CONTROL DATA



Reference

60462520

ICEM GPL
for NOS

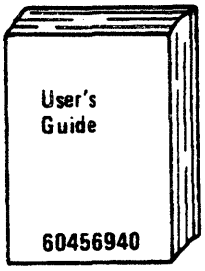
Reference

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features and parameters.

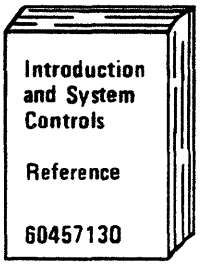
Publication Number 60462520

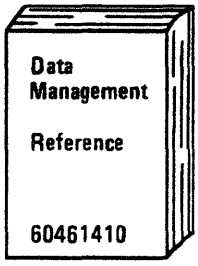
Related Manuals

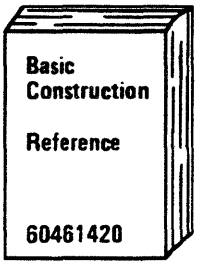
Background:

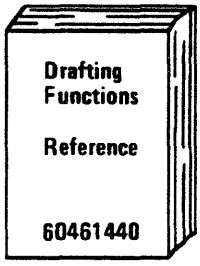


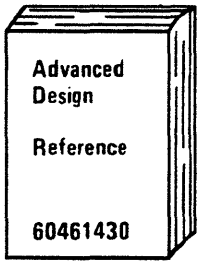
Manual Set:

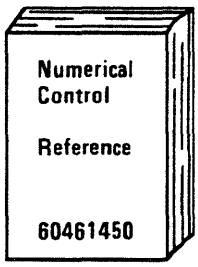


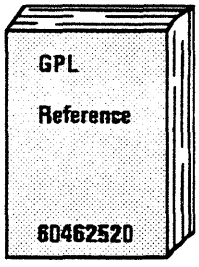






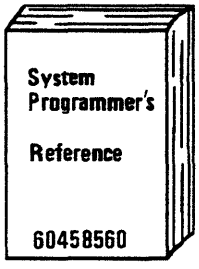






Additional References:

60457140



Manual History

Revision C documents GPL for ICEM DDN Version 1.62, printed in January 1987.

This revision includes the new commands MSTRNG and MSFILE for handling menu strings, enhancements to TLPATH, and miscellaneous technical and editorial changes. For clearer understanding, three commands were moved from chapter 6 to chapters where their operation is more functional. BLANKE and UNBLNK were moved to chapter 9; MODIFY was moved to chapter 14. Technical changes are indicated by change bars; change bars are not used on pages that have only editorial changes.

Revision Level	System Version	Date
A	1.57	May 1985
B	1.6	December 1985
C	1.62	January 1987

Contents

Figures

About This Manual

This manual describes GPL, the graphics programming language of ICEM DDN. ICEM DDN is the CONTROL DATA® Integrated Computer-aided Engineering and Manufacturing Design/Drafting/Numerical Control (ICEM DDN) software system.

Audience

This manual is a reference source for design engineers and drafting personnel who have had initial training in the use of the ICEM DDN system. It is not intended to be a tutorial guide to ICEM DDN. New users should refer to the ICEM Design/Drafting User's Guide for a step-by-step introduction to the ICEM DDN system.

Organization

This manual is organized as follows:

Chapter	Description
1	Introduces the GPL programming language and briefly describes the statement elements, statement types, program file format, restrictions, and syntax of the language.
2	Describes the components of a GPL statement. It is a more detailed discussion of each of the statement elements.
3	Is a brief overview of the operation of each major word statement.
4	Describes the operation of branching and conditional statements.
5	Explains the use of modal statements.
6	Tells how to manage data information.
7	Explains how to manage part information.
8	Describes how to manage program information.
9	Shows how to control the display of geometry.
10	Explains how to create points.
11	Describes how to create lines.
12	Shows how to create arcs and circles.
13	Explains how to create two-dimensional curves.
14	Tells how to manipulate entities.
15	Describes how to create three-dimensional entities.
16	Shows how to create different surfaces.
17	Explains how to set the ANSI 1973 drafting modals.
18	Tells how to create ANSI 1973 drafting entities.

Chapter	Description
19	Explains how to set the ANSI 1982 drafting modals.
20	Tells how to create ANSI 1982 drafting entities.
21	Shows the use of numerical control functions.
22	Describes how to use interactive commands.
23	Describes GPL input and output.
24	Explains how a GPL program is compiled.
25	Describes the operation of menu 5.13 GPL.

This manual is part of the ICEM DDN manual set.

The ICEM Design/Drafting Introduction and System Controls manual gives an overview of the major ICEM DDN concepts and describes menus 1 through 4 of the main menu: modals and fonts, blank/unblank operations, delete operations, and the file/terminate sequence.

The ICEM Design/Drafting Data Management manual describes menus 5 through 8 of the main menu: special functions, data base management operations, input/output operations, and display control.

The ICEM Design/Drafting Basic Construction manual describes menus 9 through 14 of the main menu: point construction, line construction, arc construction, special curve construction, entity manipulation, and data verification.

The ICEM Design/Drafting Drafting Functions manual describes menus 16, 18, and 19: drafting functions, analysis, and SI/US resize.

The ICEM Advanced Design manual describes menu 15 ADVANCED DESIGN, which covers three-dimensional curves and surfaces.

The ICEM Numerical Control manual describes menu 17 NUMERICAL CONTROL, the numerical control programming part of ICEM DDN.

Conventions

In this manual, headings contain a series of numbers separated by periods. These numbers represent the selections available within the ICEM DDN menu hierarchy. The first number in the heading is the main menu choice, the second number is from the second-level menu, and so on. For example, menu choice 12.7.3 HEXAGON is from the third level of the menu hierarchy.

When the word *system* is used, it refers to the ICEM DDN software system. When the Network Operating System is referred to, it is called either NOS or the operating system.

All text that the system displays is printed in uppercase letters in a special typeface, for example:

```
PEN THICKNESS
1.ON
2.OFF
3.SET PEN/THICKNESS
```

Additional Related Publications

You can find related information in the following publications:

Manual Title	Publication Number
Network Products Interactive Facility Version 1 Reference Manual	60455250
Network Products Interactive Facility Version 1 User's Guide	60455260
NOS Version 1 Reference Manual, Volume 1	60435400
UNIPLOT Version 3 User's Guide/Reference Manual	60454730
Automatically Programmed Tooling System (APT IV)	17326900
XEDIT Version 3 Reference Manual	60455730
Graphics Terminal Assist Version 1 User's Guide/Reference Manual	60476100
NOS 2 Reference Set, Volume 1 Introduction to Interactive Usage	60459660
NOS 2 Reference Set, Volume 2 Guide to System Usage	60459670
NOS 2 Reference Set, Volume 3 System Commands	60459680
ICEM Schematics Reference Manual	60456540
ICEM User-Defined Tablet Overlay	60457650
ICEM Engineering Data Library Version 1 Reference Manual	60459740
ICEM Design/Drafting GRAPL Programming Language Manual	60461460

Ordering Manuals

Control Data manuals are available through Control Data sales offices or through Control Data Corporation Literature Distribution Services (308 North Dale Street, St. Paul, Minnesota 55103).

Submitting Comments

The last page of this manual is a comment sheet. Please use it to give us your opinion of the manual's usability, to suggest specific improvements, and to report technical or typographical errors. If the comment sheet has already been used, you can mail your comments to:

Control Data Corporation
Technology and Publications Division ARH219
4201 Lexington Avenue North
St. Paul, Minnesota 55126-6198

Please indicate whether or not you would like a written response.

Introduction 1

Introduction 1

With Graphics Programming Language (GPL), you can write programs that create and file ICEM DDN part drawings. GPL statements are generally similar in operation to features in interactive ICEM DDN. They are used to create entities, set modals, retrieve and store data base information, and perform other operations required to create and file drawings. GPL statement syntax is similar in many respects to Control Data's Automatically Programmed Tooling System (APT) programming language.

You can use GPL programs for a number of purposes. You can:

- Create your own customized menus.
- Write programs that create similar parts parametrically. Starting from one part, you can create a family of parts by running the same program using different values for the input variables. Refer to the example programs in appendix G.
- Create your own customized graphics process for a special application.
- Read and write to external data files.
- Execute FORTRAN binary code. Please note that the FORTRAN subroutines that GPL executes must be fully compiled.
- Make graphs and tables.
- Write customized tutorials.
- Calculate serial functions.
- Analyze data.
- Read information from the ICEM DDN database.

GPL is compiled externally from ICEM DDN (refer to chapter 24) and executed internally from ICEM DDN (refer to chapter 25).

A GPL program is a series of statements that performs the operations for creating a part drawing. Refer to appendix G for a complete program example.

The sequence of statements in the input file generally determines the sequence of operations to be performed. In addition, control statements in the GPL program can test for certain logical conditions, and the outcome of those tests directs the program to execute different program sequences.

The GRAPL-to-GPL Translator (GTGT) utility automatically translates GRAPL source programs to GPL source programs. A one-to-one correspondence between GRAPL and GPL does not exist; statements that cannot be translated are flagged with informative remarks in the GPL source. Refer to appendix F.

Statement Elements

A GPL statement consists of different elements. These elements are described in detail in chapter 2, Components of a GPL Statement. Briefly, a statement consists of the following elements:

Element	Description
Major words	Statement elements that name the GPL operation to be performed. A major word that requires modifying parameters is followed by a slash (/). The slash separates the major word from its parameters.
Minor words	Specially reserved modifier elements that follow a major word. Minor words indicate required information such as positional relationships, surface materials, geometrical dimensions, delta values, and other information. Appendix B lists and describes the minor words.
Constants	Numeric values that do not vary during program execution.
Entity names	Names given to a geometric definition. Entity names in GPL are local unless intentionally assigned to the data base. Such assigned names are then available elsewhere in ICEM DDN.
Variable names	Names given to any combination of mathematical variables, constants, function statements, and arithmetic expressions.
Statement labels	Optional integer fields, from 1 to 5 digits in length, which uniquely identify the GPL statement in which they appear. If present, the label appears as the first element in a GPL statement, and each label must be unique within a given program. At least one blank must appear between the label and the remainder of the statement. A statement label is used to identify the statement to receive control from a branching or conditional statement. Refer to chapter 4, Branching and Conditional Statements, for more information on this feature.
Arithmetic operators	The operations of addition, subtraction, exponentiation, multiplication, and division.

Element	Description
Logical operators	The conditional operations of equality, such as: • Less than • Less than or equal to • Equal to • Not equal to • Greater than or equal to • Greater than
Functions	Reserved keywords for standard trigonometric and arithmetic functions.
Punctuation	The use of symbols or spaces to delineate parts of GPL statements.
Character strings	Strings of one or more contiguous text characters enclosed in single quotes.
Lower case letters in text	Under NOS, lowercase letters may be input, output, and defined. For interactive input (via TEXT) the receiving character field must be made twice as large as the maximum number of characters expected. During execution of the GPL program, the user simply clears the keyboard LOCK key so that lowercase characters can be entered. The text that was entered can later be output via the NOTE statement. The definition of lowercase letters is somewhat more troublesome: first switch the terminal to ASCII, then edit your program that defines the strings (as constants or in DATA statements), then switch the terminal back to NORMAL, and finally replace all occurrences of '^' in the program by '['.
Text variable	A variable name assigned to a character string.

Statement Types

There are four types of GPL statements:

Type	Description
Major word statement	Describes an operation that the GPL program performs.
Assignment statement	Links a name to a major word statement or an arithmetic expression.
Branching statement	Interrupts the normal sequential execution of the statements in a program and transfers control to another statement in the program.
Conditional statement	Makes a logical decision. If the statement is true, the statement either assigns a variable or transfers control to another statement in the program. If false, the statement continues execution with the next statement following the conditional statement.

Major Word Statements

Major word statements are used for the following purposes:

Statement	Description
Data management	Used to obtain entity data. These commands include operations similar to some operations in menu 5 SPECIAL FUNCTIONS.
Entity definition	Defines an entity. This statement can also be an assignment statement. Entity definitions are equivalent to the types and forms found in the interactive version of ICEM DDN. Refer to the ICEM DDN System Programmer's Reference Manual for descriptions of the types and forms as they are in the data base. Refer to appendix C for a list of entity types.
Interactive commands	Provides for user interaction with the GPL program.
Modals and fonts	Controls system modals and fonts. These commands include operations similar to some operations in menus 1 MODALS AND FONTS and 16.1 DRAFTING MODALS.
Part management	Controls part management. This command includes the file operation similar to that in menu 4 FILE.
Program management	Controls program management.

Assignment Statements

Assignment statements are used for the following purposes:

Statement	Description
Entity names	Links a name to a major word entity definition statement.
Text variable names	Links a name to a literal constant.
Variable names	Links a name to a mathematical expression.

Branching Statements

Branching statements are used to unconditionally transfer control from one part of the program to another part of the program. There are two unconditional branching statements, the GOTO statement and the JUMPTO statement.

Conditional Statements

Conditional statements are used to test for logical conditions and then perform some action. The conditional statements are:

Type	Description
FOR statement	Defines a conditional loop.
IF with assignment	Tests a logical condition and assigns a variable if true.
IF with branching	Tests a logical condition and branches if true.
Computed GOTO	Branches from a numerical condition.

NOTE

For the advanced programmer, several system I/O commands are available. Refer to appendix E.

Program File Format

The format requirements of a program file are:

- The first word in a GPL statement must be one of the following:
  - Major word
  - Statement label
  - Entity name
  - Mathematical variable
- Variable assignment statements must precede any statements that reference those variables.
- A program statement can be written on more than one line. A dollar sign ($) is used to continue the statement to the next line. Statement elements should not be broken, with the exception of character strings, which may be broken.

  Examples:

Correct Use	Incorrect Use
`CIRCLE/CENTER,PT001,LARGE,$` `  TANTO,CIR001`	`CIRCLE/CENTER,PT0013,LARGE,TAN$` `  TO,CIR001`
`NOTE/PT345,ANGLE,0,'WEIGHT=33.5$` `  GRAMS FOR ALL BEARINGS'`	

- The last line of the program must contain a FINI or STOP statement (for a description of these statements, refer to Major Word Statements, earlier in this chapter).

NOTE

TELEX users should use the FINI statement rather than the STOP statement. TELEX interprets a STOP statement as a user break and aborts a program.

Restrictions

The following restrictions apply to a GPL program:

- The length of a GPL statement is limited to a maximum of 80 characters per line. The statement can be continued on subsequent lines by using the $ (dollar sign) character. The number of continuation lines is limited to 19.
- The number of variables stored in the run-time library (RTL) is limited to 510.
- The total number of characters for simple entity names and simple variables is limited to a maximum of 6 characters. For subscripted variables, the total of 6 characters includes the characters, but not the parentheses and subscript.
- The maximum length of all strings and symbolic names taken together in a string is 1600 characters.
- The DELETE statement can delete only by using entity names. The exception is the DELETE/POINTS statement which deletes all points whether they are named or unnamed. This is different from the interactive version of ICEM Design/Drafting where entities can be deleted by entity type.
- Use caution when using the same entity name for different definitions. A redefinition is accepted and the name is dropped from the previous entity.
- It takes more time to run a GPL program if any of the following are true:
  - PAINT/OFF is not used.
  - The GET statement is used.
  - The ASSIGN statement is used.
- Scientific notation is not available in GPL.
- Unlike FORTRAN, GPL does not differentiate between real and integer variables.
- All angles in entity definition statements must be in degrees.
- Angles can be in radians for internal calculations in the GPL program. Trigonometric functions are available for evaluating angles in radians.
- All angles, whether in degrees or radians, must be expressed in decimal fractions. Minutes and seconds are not allowed.

  Examples:

  20-1/2° is expressed in GPL as 20.5

  10°, 45 min is expressed in GPL as 10.75
- The result of an entity construction is a pointer to the ICEM DDN data base. Such a pointer cannot be used in arithmetic statements.

  Example:

  ```
  PT=POINT/10,10,50
  ```

  The point PT cannot be used as a variable in arithmetic statements.

- Expressions can be used as indices of arrays, as initial, step, and until values of a FOR statement, and as a value for the computed GOTO statement. Expressions can also appear in major word statements in place of real variables or constants. However, an expression that defines an index cannot use another index expression in that definition. In GPL, an index is a valid index expression if it contains more than one constant or variable. For example, X(I(J)) is a valid index expression; however, Y(1,2*J) is not valid.

- The order of declarative statements at the beginning of a program must be as follows:

  MAIN or PROC

  CONST

  COMMON

  ENTITY, REAL, CHAR[1]

  END COM

  ENTITY, REAL, CHAR[1]

  Data

  Executable statements

  FINI or RETURN

- Blanks outside of character strings are ignored if they precede or follow a separator (the comma).

- Blanks cannot be used within keywords, symbolic names, or constants.

- Blanks must be used as separators with the reserved words FOR, GOTO, TO, STEP, and UNTIL.

  For example:

Valid Use	Invalid Use
GOTO 10	GOTO10
GO TO 20	GOTO20
FOR I=X, STEP Y, UNTIL Z	FORI=X,STEPY UNTILZ

- Strings (enclosed by apostrophes) may contain all characters including apostrophes and $. However, as in standard FORTRAN, apostrophes within a string must appear as two consecutive apostrophes. To be recognized as part of the string, the $ must be followed by at least one non-blank character within the same line. If, on the other hand, the $ is the last non-blank character of a line, it is considered a continuation mark, and the string continues with the first non-blank character of the next line.

1. ENTITY, REAL, and CHAR can be in any order among themselves.

Syntax Conventions

In this manual the following conventions are used in defining allowable syntax for GPL statements:

- All angles used to define an entity are expressed in degrees.
- The positive angular direction is counterclockwise.
- Units of measure are either millimeters or inches.
  - All U.S. customary unit measurements are in inches.
  - All SI unit measurements are in millimeters.
- All major and minor words are uppercase (for example, LINE, TANTO, and CIRCLE).
- Geometric entities are referred to by their entity type in lowercase (for example, line, point, and circle).
- Other names are represented by meaningful lowercase pseudonyms (for example, xcoord, radius, and angle).
- Items listed vertically within parentheses indicate that you must choose one of the items.
- All items enclosed in brackets are optional.
- Items arranged vertically within brackets indicate that you may choose one of the items.
- [,...] in a statement format indicates that the preceding parameter can be continued as an optional list of items.

Example:

Program Statement	Explanation
`SAVE/variable[,...]`	Indicates that a list of variables can be saved in the UTF.

```
SAVE/A1,B1,C,DEFX
```

Creating Geometry with GPL

The following steps create geometry with GPL:

- A GPL program is written in a local file using the operating system text editor.
- The GPL compiler compiles the program using the local text file.
- ICEM DDN menu 5.13.3 executes this program and creates the part geometry in the part drawing. Refer to chapter 25. During execution, GPL may issue execution error messages (listed in appendix D).

Filing GPL Program Results with the Current Drawing

How you file the GPL program output depends on whether or not you include the FILE statement in the program input file.

If the FILE statement is omitted from the program, the output drawing is displayed on the screen as a part of the current work view.

If the current work view is not blank, the GPL output drawing is superimposed on the current contents of the part drawing.

To make the GPL output drawing a permanent part of the current part drawing, you must manually file the part, using the File/Exit menu operation 4.FILE CURRENT PART/EXIT ICEM DDN, before logging out of ICEM DDN.

If the GPL program contains a FILE statement, the part drawing is filed automatically as the current part and sheet number when the FILE statement is encountered.

Running a GPL Program

Using menu choice 5.13.3 RUN GPL PROGRAM, you can run your GPL program. When you enter 5.13.3, the system displays:

ENTER SIX CHARACTER NAME — Enter the name of the program to be run.

The system checks for a local, external library that contains the name of the program. If found, the program is executed immediately.

The program to be executed must be compiled and the resulting object code must be put on a local external file named GPLLIB. It is not put there automatically after compilation. The GPL overlay library named GOLIB, another local external file, is also required. Refer to figure 1-1.

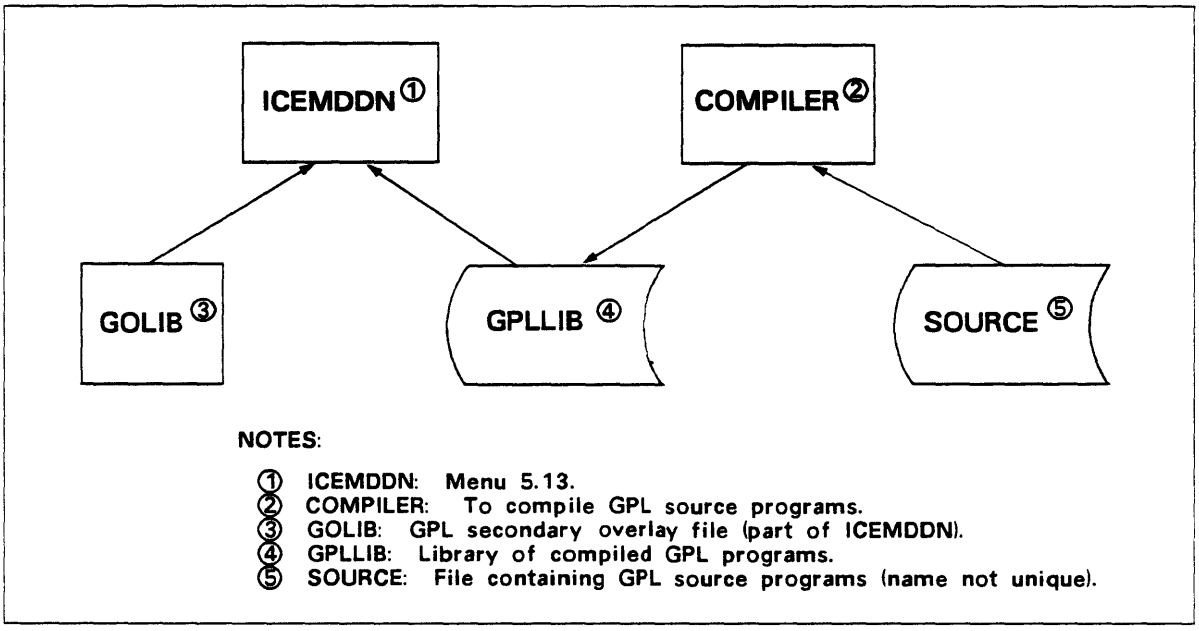


Figure 1-1. GPL File Structure

Using the PAUSE Statement

If the GPL program has a PAUSE statement, the program stops execution at the PAUSE statement. You can then perform other ICEM DDN operations. When you want to continue the program, select menu choice 5.13.2 CONTINUE GPL PROGRAM. The program resumes execution at the first line following the PAUSE command.

Components of a GPL Statement 2

Components of a GPL Statement 2

Major Words

Major words are the statement elements that name the GPL operation to be performed. A major word is followed by a slash (/) unless there are no modifier elements. Major words are reserved keywords, and may not be used as variable names.

Examples:

Program Statement	Explanation
`POINT/3,3,0`	The major word POINT defines a point at coordinates (3,3,0).
`PT001=POINT/4,4,0`	Point PT001 is defined at coordinates (4,4,0).
`CIR1=CIRCLE/CENTER,$` `  PT001,RADIUS,2.5`	Circle CIR1 is defined using point PT001 as the center with a radius of 2.5 units of measure.

Minor Words

Minor words are modifier elements that follow a major word and indicate information such as positional relationships, surface materials, geometrical dimensions, and delta values. Minor words are also reserved keywords, and may not be used as variable names.

Examples:

Program Statement	Explanation
`POINT/DELTA,PT001,2,4`	A point is created delta from point PT001, 2 units of measure in the x positive direction and 4 units of measure in the y positive direction.
`POINT/CENTER,CIR1`	A point is created at the center of circle CIR1.
`CIR1=CIRCLE/CENTER,PT001,$` `  RADIUS,2.5`	Circle CIR1 is created using point PT001 as the center with a radius of 2.5 units of measure.

Defined Symbols

You can define a symbol to represent a geometric entity or variable name. The symbol consists of up to 6 alphanumeric characters, starting with an alpha character. It may not be a GPL vocabulary word (that is, a reserved word). It may have subscripts up to three dimensions.

Geometric entity names should (must, if subscripted) be declared in an ENTITY statement. These entity names are defined by assignment (=) of major word statements which define the geometric entity, or by the interactive SELECT statement.

You can use variable names instead of constants or expressions. You can define a variable name in one of three ways: using an assignment statement, using one of the interactive commands such as a PARAMS or POS statement, or using the GET statement if the variable is defined in the run-time library (RTL). Such variables should (must, if subscripted) be declared in a REAL statement.

You can also use variable names to represent strings. These names are called text variables and must be declared in a CHAR statement. The CHAR statement must have at least one dimension which defines the maximum length in characters of the text variable. All text variables have two character counts: maximum and current. Both counts are preset to the number given in the CHAR statement.

While the maximum count does not change, the current count is redefined whenever the variable serves as receiving location (statements CONVER, DATE, TIME, TEXT, OBTAIN, MOVCHR, and READ). For MOVCHR, the current length of the target string is increased when necessary to include the last target character position of the move.

When the variable serves as the source (statements CLOSE, CMPCHR, DISPLA, EXEC, GET, MENU, MOVCHR, OPEN, PARAMS, PAUSE, POS, QUERY, SAVE, SELECT, STOP, TEXT, WRITE, and DRAFTING CREATION), only the current number of characters is available.

You can use the OBTAIN statement to retrieve the current length, and the SETCHL statement to set it (within the bounds of the maximum length).

Constants

Conventions

The following conventions apply to the use of numerical constants in GPL programs:

- Real constants are limited to 7 places to the left and 10 places to the right of the decimal point.
- The entire constant may not exceed a total of 15 decimal digits.
- The plus sign for positive numbers is optional.
- A decimal point need not be entered if the data has no fractional value.
- Preceding or trailing zeros need not be entered.
- Real or integer constants can be assigned to a symbolic name using the CONST statement. The symbolic name can then be used as real or integer constants.

Using the Assignment Statement to Create a Constant

You can create a constant by using an assignment statement. A constant is a name given to specific numerical value.

Format:

```
name=value
```

Parameter	Description
name	The name assigned to the constant.
value	The value of the constant.

Examples:

Program Statement	Explanation
`A=3.5`	A is created with a constant value of 3.5.
`B=450`	B is created with a constant value of 450.
`C=125*A+B-100`	C is created with a constant value of (125*A+B-100).

NOTE

Scientific notation is not available in the GPL language.

Named Entities

You can name an entity by using an assignment statement when creating the entity. Entity names are optional. Note that entity names defined in a GPL program are local (unlike entity names defined interactively under ICEM DDN menu 5.11 NAMED ENTITIES) and cannot be referenced interactively after execution of the GPL program which defined the name.

NOTE

Entity names in GPL are local unless specifically defined in the data base using the ASSIGN statement.

All ICEM DDN major and minor word names are reserved and may not be used as entity names.

Entity names can be simple or subscripted.

Statement format:

```
name=description
```

Parameter	Description
name	The name assigned to the entity.
description	The entity description. It is a major word statement defining a geometric entity.

Simple (Nonsubscripted) Entity Names

Simple entities are entity names assigned to one entity only. Simple entity names can be from 1 to 6 alphanumeric characters; the first character must be alphabetic.

Subscripted (Array) Entity Names

Subscripted (array) entities are a group of entities stored in the same array name. This array may be subscripted up to three dimensions. The location in the array is determined by a subscript index value enclosed in parentheses. A subscripted entity name consists of a 1- to 6-character alphanumeric name, beginning with an alphabetic character, followed by one, two, or three subscripts separated by commas and enclosed in parentheses.

A subscripted entity name must be dimensioned with an ENTITY statement in each program or subprogram that references the entity.

The following are examples of the ENTITY statement and valid subscripted entity names:

Program Statement	Explanation
`ENTITY/PT(10),L(200),ART(10)` ⋮	Point PT, line L, and arctangent ART are dimensioned as entity arrays.
`PT(10)=POINT/3,3,0`	Point PT(10) is created as the 10th element in the PT entity array.
`L(108)=LINE/PT3,PT4`	Line L(108) is created as the 108th element in the L entity array.

`ART(6)=CIRCLE/CENTER,PT(10)$`
`,TANTO,L(108)$`
`,GOANG,10,ENDANG,50`
	Arc ART(6) is created as the 6th element in the ART entity array.

`P(I,J,K)=POINT/I,J,K`	Point P(I,J,K) is created as the element in the three-dimensional array P at the location defined by the current values of I, J, and K.

Variables

A variable is an assignment statement that assigns any combination of variables, constants, functions, and arithmetic expressions to a name. Variables can be assigned either real or integer values, but internally, all variables are stored as real values.

Using the Assignment Statement to Name Variables

You can name a variable by using an assignment statement. A variable name is a name given to any combination of variables, constants, function statements, and arithmetic expressions.

Format:

```
name=value
```

Parameter	Description
name	The name assigned to the variable.
value	The value of the variable.

Simple (Nonsubscripted) Variables

Simple variables are variable names assigned to a single value. Simple variable names can be from 1 to 6 alphanumeric characters; the first character must be alphabetic.

Subscripted (Array) Variables

Subscripted (array) variables are a group of variables using the same name for an array of values. This array may be subscripted up to three dimensions. The location in the group is determined by a subscript enclosed in parentheses. A subscripted (array) variable name consists of a 1- to 6-character alphanumeric name that begins with an alphabetic character, which is followed by one, two, or three subscripts separated by commas and enclosed in parentheses.

- Subscripted variables must be dimensioned by a SIZE or REAL statement in each program that references the variable.
- Simple variables can be saved in the RTL using the SAVE statement.
- The number of all subscripted variable array elements is limited to a total of 2,097,152.
- The subscripted variable array is a one-, two-, or three-dimensional array.
- Subscripts can be simple variables, for example, ABCD(J).
- Major and minor word names are reserved and cannot be used as variable names.

The following are examples of correct and incorrect subscripted variable names:

Correct use:

Program Statement	Explanation
`ABCDEF=K+SIN(I)`	Uses 6 characters.
`ABCDE(6)=10.0`	Uses fewer than 6 characters not including the subscript.
`X(10,2,5)=Y(I,J,K)`	Uses three-dimensional subscripts.

Incorrect use:

Program Statement	Explanation
`ABCDEFG=47.1`	Uses 7 characters.
`POINT(3)=TOLDEF`	Uses the major word POINT.
`RADIUS(3)=R`	Uses the minor word RADIUS.
`7ABCDE=ABCDE`	Uses numeral as first character.

Creating Variables Using 5.2.1 VARIABLE CALCULATION

Using 5.2.1 VARIABLE CALCULATION, you can define a simple variable and assign a value to it. The variable name must be from 1 to 6 characters, the first of which must be an alphabetic character. The value can be specified as either a constant or an expression. After successful evaluation, the variable is automatically stored in the RTL. Any variables in the right-hand part of any statement in an expression must have been previously defined, and must exist in the RTL. Variables defined and saved with this menu choice are subsequently available to GPL programs using the GET statement.

Saving Variables in the RTL or the UTF

Variable values can be carried over from one GPL program to subsequent programs without having to repeat the assignment statement in each program. There are two variable storage areas in ICEM DDN:

- Run-Time Library (RTL)

  The RTL is used to store variables for the duration of an ICEM DDN session and is stored with the part. These variables are available to all programs executed during the session. At the end of a session, variable assignments stored in the RTL are saved only if you file the current part drawing. If you do not file the drawing, the variables are lost when you log out of ICEM DDN. When you retrieve the part drawing for a future session, variable assignments saved with a part drawing are automatically placed in the RTL. Simple variables are stored in the RTL only if they are specified in a SAVE statement (refer to the SAVE major word description later in this section) or defined interactively under menu 5.1 VARIABLE CALCULATION (refer to the ICEM Design/Drafting Data Management manual) or via data capture. The GET statement must be used to return variables from the RTL before use in a GPL program.

- User Technology File (UTF)

  You can globally save variable assignments stored in the RTL by moving them to the UTF using 5.3.2 MOVE VARIABLES FROM RTL TO UTF. You can retrieve variable assignments stored in the UTF for use in a GPL program executed at a later time. Programs, however, cannot access variable assignments directly from the UTF. For each subsequent session, variables stored in the UTF must be moved into the RTL for the session before the variables can be referenced in any program. UTF variables are moved to the RTL using menu operation 5.3.1 MOVE VARIABLES FROM UTF TO RTL. A variable assignment stored in the UTF remains as originally defined until that variable is redefined in a subsequent assignment statement and stored once again in the UTF.

Using the SAVE Statement

You must use the SAVE statement to save your simple variables before you use them in a subsequent program or subprogram. The SAVE statement reserves space in the RTL. For example, the following series of statements explains this process.

Program Statement	Explanation
`A=1`	A is equal to the value of 1.
`B=4`	B is equal to the value of 4.
`F=15`	F is equal to the value of 15.
`TAR=A+B`	TAR is equal to the value of A plus B or 1 plus 4.
`ABCD=TAR*F`	ABCD is equal to the value of TAR times F or 5 times 15.
`SAVE/A,B,F,TAR,ABCD`	This statement saves these variables in the RTL.

Using the REAL Statements

You must use the REAL statements to reserve storage space for your subscripted mathematical variables in the GPL program before you define them in the program. You must reserve this space before all statements in a given program except MAIN, REMARK, and other declarations.

These statements are used for the following purpose:

Statement	Purpose
REAL	Reserves space for calculated values.

Example:

Program Statement	Explanation
`REAL/A,B,C(5)`	Reserves space for variables A, B, and five spaces for variable C.

Using the CHAR and ENTITY Statements

The CHAR statement declares the names as text variables and reserves space by character length in the GPL program. The ENTITY statement declares the names as entities and reserves space in the GPL program for their pointers. Subscripted entity names must be dimensioned to the number of entities that are assigned.

These statements are used for the following purposes:

Statement	Purpose
CHAR	Reserves space for text variables.
ENTITY	Reserves space for entity data base pointers.

Example:

Program Statement	Explanation
`CHAR/CHR1(10)`	Declares text variable CHR1 and reserves space for 10 characters for CHR1.
`ENTITY/PT(10)`	Reserves space for 10 pointers named PT(1) through PT(10).
`ENTITY/CIR(2)`	Reserves space for 2 pointers named CIR(1) and CIR(2).

Using the GET Statement

You must use the GET statement in a subprogram to retrieve the values of variables if they are stored in the RTL. These values are only available if they were stored in the RTL using 5.2.1 VARIABLE CALCULATION or if they were stored in the RTL using a SAVE statement in a different program executed earlier.

Example:

Program Statement	Explanation
`GET/A,B,F,TAR,ABCD`	Retrieves the values of previously stored variables A, B, F, TAR, and ABCD.

Statement Labels

Statement labels are optional integer fields, from 1 to 5 digits in length, which uniquely identify the GPL statement in which the statement label appears. If present, the label appears as the first element in a GPL statement, and each label must be unique within a given program. At least one blank must appear between the label and the remainder of the statement.

- A statement label is used to identify the statement to receive control from a branching or conditional statement. Refer to chapter 4, Branching and Conditional Statements, for more information on this feature.
- Statement labels are not required on all GPL statements.
- One and only one statement label must exist somewhere for each transfer of control.
- Labels must be positive integers with 5 or fewer digits.

Example:

Program Statement	Explanation
`GOTO 250`	Branch unconditionally to statement label 250.
⋮	
`250 CONTIN`	Execution continues at the next statement.

Arithmetic Operators and Expressions

Arithmetic expressions use arithmetic operators to combine constants, variables, or functions in a sequence that can be reduced to a single arithmetic value.

The arithmetic operators available in GPL are:

+	Addition
-	Subtraction
**	Exponentiation
*	Multiplication
/	Division

Arithmetic expressions are evaluated from left to right in the following order of preference:

- Exponentiation
- Multiplication and division
- Addition and subtraction

Nested expressions (expressions within parentheses) are evaluated first, proceeding from the innermost expression to the outermost.

Example:

```
CALC=((3+7)/2-4)**3+5*6
```

Operation	Result
The innermost nest (3+7) is evaluated first.	(10/2-4)**3+5*6
Division (10/2) is the first operation to be performed in the second-level nest.	(5-4)**3+5*6
Subtraction (5-4) is performed after division in the second-level nest.	1**3+5*6
Exponentiation (1**3) is the highest-order operation.	1+5*6
Multiplication (5*6) is the next highest-order operation.	1+30
Addition (1+30) is the last operation. The final result is 31.	31

Functions

Functions are elements that provide a quick and simple means of performing certain commonly used arithmetic/trigonometric operations. They can be used in variable assignment statements. There are two types of functions: those that require a single argument and those that require several arguments.

Format:

```
function(argument(s))
```

Parameter	Description
function	The function name.
(argument(s))	The argument(s) for the function. The argument(s) can be variable(s).

The following functions are available in GPL:

Function[1]	Description
ABS or ABSF	Finds the value of a number without regard to sign (the absolute value).
ACOS	Finds the arc cosine. The answer is returned in radians.
ACOSF	Finds the arc cosine. The answer is returned in degrees.
ASIN	Finds the arc sine. The answer is returned in radians.
ASINF	Finds the arc sine. The answer is returned in degrees.
ATAN	Finds the arctangent given the tangent of an angle. The answer is returned in radians.
ATANF	Finds the arctangent given the tangent of an angle. The answer is returned in degrees.
COS	Finds the cosine of an angle. The angle must be expressed in radians.
COSF	Finds the cosine of an angle. The angle must be expressed in degrees.
COSH	Finds the hyperbolic cosine of an angle.
EXP or EXPF	Finds the value of e raised to a power.

1. Some functions have alternate forms, which have been provided for compatibility with Automated Programmed Tooling System (APT).

Function[2]	Description
GETBIT (var, pos, num)	Gets the bit string of length 'num' from variable 'var', starting at position 'pos'. **NOTE** For GETBIT and SETBIT, the bit position (ranging from 1 to 48 in NOS) is the lower end of the bit string in the variable (after it has been converted to integer internally).
LOG or LOGF	Finds the natural logarithm of a number (base e).
LOG10	Finds the common logarithm of a number (base 10).
MIN	Finds the minimum value of a series (up to 64 arguments).
MAX	Finds the maximum value of a series (up to 64 arguments).
ROUND	Rounds a real number to its nearest integer (0.5 rounds up).
SETBIT (var,pos,num,val)	Creates a new bit string by setting 'num' bits with the new bit representation of 'val' in variable 'var' starting at position 'pos'. The arguments remain unchanged. The new variable value is passed as the function value. **NOTE** For GETBIT and SETBIT, the bit position (ranging from 1 to 48 in NOS) is the lower end of the bit string in the variable (after it has been converted to integer internally).
SIGN	Transfers the sign of the second argument to the first argument (similar to the FORTRAN SIGN function).
SIN	Finds the sine of an angle. The angle must be expressed in radians.
SINF	Finds the sine of an angle. The angle must be expressed in degrees.
SINH	Finds the hyperbolic sine of an angle.
SQRT or SQRTF	Finds the square root of a number.
TRUNC	Truncates a real number to its integer part.
TAN	Finds the tangent of an angle. The angle must be expressed in radians.
TANH	Finds the hyperbolic tangent of an angle.

2. Some functions have alternate forms, which have been provided for compatibility with Automated Programmed Tooling System (APT).

Examples:

Program Statement	Explanation
`A=ATNF(0.5)`	The variable A is equal to 26.565°.
`A=ATAN(0.5)`	The variable A is equal to 0.4636 radians.
`PZ=L*SINF(45)`	The variable PZ is equal to L times the sine of 45°.
`SKL=SIN(K*L)`	The variable SKL is equal to the sine of K times L (K*L).
`PI=4*ATAN(1)`	The variable PI is equal to four times the arctangent of 1. The answer is returned in radians.
`B=SINF(ANGR)`	The variable B is equal to the sine of ANGR.
`C=L*COSF(ANG1)+$` `  K*SINF(ANG2)`	The variable C is equal to L times the cosine of angle ANG1 plus K times the sine of angle ANG2.
`IF(GETBIT(J,1,1).NE.0)$` `GOTO10`	If the bit string of length 1 from variable J at position 1 is not equal to 0 (if J is odd) go to 10.
`J=SETBIT(J,1,1,0)`	Creates a new bit string by setting 1 bit to 0 J at position 1 (round J to next lower even number).

Logical Operators

Logical operators available in GPL are the conditional operations of equality. They are used within the IF statement only. The periods are not part of the operators; they are required to separate the operators from the objects being compared.

Symbol	Description
.LE.	Less than or equal to
.LT.	Less than
.EQ.	Equal to
.NE.	Not equal to
.GE.	Greater than or equal to
.GT.	Greater than
.AND.	Logical and
.O,R.	Logical inclusive or

NOTE

.AND. and .OR. operators cannot be combined. A GPL program can use multiple .AND.s, or it can use multiple .OR.s, but it cannot use combined .AND.s and .OR.s.

Examples:

Program Statement	Explanation
`IF (STAT.EQ.-1) GOTO 40`	If the variable STAT is equal to -1, jump to the line labeled 40.

Punctuation Symbols

The following punctuation marks and other special symbols are used in GPL statements:

Symbol	Description
,	Separates modifiers in major word statements and also separates multiple subscripts.
.	Separates a logical operator from the two values to be compared or acts as a decimal point.
/	Separates major words from following modifiers.
=	Assigns a name to an entity, a constant, an expression, or another name.
$$	Indicates the end of a statement; comments can appear on the same line to the right of the double dollar signs ($$). This is equivalent to a REMARK statement if $$ appears in columns 1 and 2.
'	Encloses character strings.
( )	Indicates nested expressions within arithmetic expressions or encloses a subscript.
$	Indicates that the statement is continued on the next line. Character strings can be continued on a subsequent line also in this manner; other statement elements, such as minor words, entity names, literal constants, variable names, and so forth, cannot be continued from line to line.
Blank (spaces)	Generally, indicates spacing only. The compiler usually ignores blanks except in the GOTO and FOR statements. Blanks in character strings, which are used for display purposes are also significant.

Character Strings

A character string is a string of one or more contiguous text characters enclosed in single quotes. It is used in many interactive and drafting statements.

Example:

Program Statement	Explanation

`MENU/'CREATE THE CONNECTING CURVE',$`
`'LINE*ARC*SPLINE*',$`
`CHOICE,STAT`
	The character string CREATE THE CONNECTING CURVE is defined in the MENU command.

Text Variable

A variable name assigned to a character string. This variable name can be assigned using either the interactive TEXT command or the MOVCHR statement.

Examples:

Program Statement	Explanation
`MOVCHR/8,'CUT HERE',1,A,1`	Text variable A is defined with the text of CUT HERE.
`MOVCHR/5,'REF 1',1,B,1`	Text variable B is defined with the text of REF 1.
`MOVCHR/15,'CONSTANT$` `RADIUS',1,C,1`	Text variable C is defined with the text of CONSTANT RADIUS.
`TEXT/'ENTER TEXT',TXT,STAT`	The variable TXT is assigned to the alphanumeric string you enter.
`MOVCHR/16,'TOLERANCE=$` `0.0015',1,TXT,1`	The variable TXT is assigned using the MOVCHR statement.

NOTE

Text variables can be used in place of character strings in attributes, notes, labels, dimensions, and other statements.

GPL Vocabulary I (Major Words)

The following are GPL vocabulary words and must not be used as symbols:

1. Modals and Fonts

AHEAD	CDISPL	DORIG	PAINT	SELMOD	TXTORG
ANGCTL	CRES	DSCALE	PEN	SEQNO[3]	TXTOUT[3]
ANUNIT	CSET	DUAL	PREFIX	SLANT	TXTROT[3]
ARAUTO	CSIZE	FONT	RECOVR	SPATHS	VBORDS
ARIN	CURSOR	FRACT	REFRES	STATUS	VNAMES
AROUT	DECMAL	KEYIN	RESCAL	SYSDEC	VVECS
ARROW	DIMOF	LDDIAM	SECALN	TXTANG	WLINE
ATAIL	DIMORG	LEADER	SECVIS	TXTIN[3]	WPLANE[3]
AUTOD	DISDEF	LEVEL	SELENT	TXTJUS	ZSURF
BLANK	DISTOL	MATERL	SELGRP	TXTLOC	

2. Major Word Statements

ADIMEN	CMSRF	FILSRF	MODDFT	RDIMEN	SRFTEX
ARRAY	CONE	FIND[3]	MODIFY	READ	STRING
ASSIGN	CONVER	GCONIC	MOVCHR	REPANT	SYSTEM[3]
ATTRIB	CURARR	GEOTOL	MOVENT	REVSRF	TABCYL
BALOON	CYLNDR	GET	MSFILE	REWIND	TAPER
BEZIER	DATE	GROUP	MSTRNG	ROTATE[3]	TEMPLT
BEZSRF[3]	DATFEA	HEXDRN	NOTE	RTRIEV	TEXT
BLANKE	DATUM	HYPERB	OBTAIN	RULSRF	THIKNS
BULK	DDIMEN	HEXDRN	OFSRF[3]	SAVE	TIME
CDIMEN	DEFINE	IMF[3]	OPEN	SEARCH[3]	TLPATH
CHAMFR	DELETE	LABEL	PARABO	SECARR	TORUS
CHANGE	DEVSRF	LDIMEN	PARAMS	SECTON	TRANSL[3]
CHECK	DISPLA	LINE	PARTNO[3]	SELECT	TRIME
CIRCLE	ELLIPS	LPSOID	PLANE	SETCHL	UNBLNK
CLEAR	EVALC	MACCRV	POINT	SETGPG	USTRUC
CLINE	EVALS	MAGNFY	POS	SPCURV	VECTOR
CLOSE	EXEC	MAP	PROJEC	SPHERE	VIEW
CMPCHR	FILE	MENU	PTSET	SPLINE	WRITE
CMPCRV	FILLET	MIRROR	QUERY	SRCHD	ZOOM
CMPENT					

3. Program Control and Declaratives

CALL	DATA	ENDIF	GO[4]	PAUSE	STEP
CHAR	ELSE	ENTITY	GOTO	PROC	STOP
COMMON	ELSEIF	EOFI	IF	REAL	THEN
CONST	END	FINI	JUMPTO	REMARK	TO[4]
CONTIN	ENDCOM	FOR	MAIN	RETURN	UNTIL

3. Reserved for future use.

4. Reserved for compilation purposes.

GPL Vocabulary II (Minor, Modal, and Positional Words)

The following are GPL vocabulary words and must not be used as symbols:

ABOVE	CURVW	GLOBAL	NORMAL	RUBBER	UNIT
ACYCLC	CW	GOANG	NORPNT	SCALAR	UPATHS
ADD	CYCLIC	GPATRN	NORTH	SCALE	URIGHT
AFTER	DASHED	HANGLE	NORTRN	SECDIS	USER
ALL	DATMOD	HORIZ	NUMBER[5]	SECOND	USTART
ALUM	DATREF	INFIN	NUMBRX	SEQNUM	UTERM
ANGLE	DECIM	INPUT	NUMBRY	SHEET	UWRITE
ARCS	DEFVW	INSIDE	NVARY	SINGLE	VDEF
AREA	DEGREE	INSTNC	OFF	SLATE	VDIREC
AROUND	DEGTOL	INTOF	ON	SLOPE	VECSUM
ARRDO	DEL	IRON	ORIGIN	SMALL	VERTCL
ARRW	DELANG	JOG	OUT	SOLID	VPATHS
ASPCT	DELATR	LARGE	OUTPUT	SOUTH	WEST
ATANGL	DELTA	LATHE	PARAM	SPHRIC	WIDTH
ATNAME	DELTAX	LEAD	PARBLC	START	WITP
AUTO	DELTAY	LEADR	PARLA	STD	WITX
AUX	DELTAZ	LEFT	PARLEL	STDVW	XAXIS
BASIC	DETAIL	LENGTH	PARNOR	STEEL	XLARGE
BEARDS	DIFFER	LEVL	PARTNA	SUBAT	XMOVE
BEFORE	DIRECT	LIMIT	PATERN	SUPP1	XSMALL
BELOW	DISP2	LIN	PENNUM	SUPP2	XSTART
BLNK	DISTNC	LINES	PERPTO	SUPP11	XTROT
BORDER	DONT	LLEFT	PHANTM	SUPP22	XVALUE
BOTH	DOWNSP	LMC	PIERCE	SUPPB	XYMOVE
BRACKT	DRILL	LOCAL	PLASTC	SUPPB1	XYZMOV[5]
BRASS	EAST	LRIGHT	POINTS	SUPPB2	XZMOVE[5]
CCW	EDGE	MAGNES	POSITN	TAB1[5]	YAXIS
CENLIN	END	MARBLE	POSITV	TAB2[5]	YES
CENTER	ENDANG	MATRIX	POSN	TAB3[5]	YLARGE
CHAIN	ENTER	MEMBER	PTZ	TABLET	YMOVE
CHARST	ENTNAM	MIDDLE	RADANG	TANTO	YSMALL
CIRC	ENTPTR	MINUTE	RADIUS	TEXTD	YSTART
CIRCUM	ENTTY	MMC	RATIO	THICK	YTROT
COMPOS	ENTTYP	MODFY	RECT	THREAD	YVALUE
CONIC	EQUATR	MODVW	REF	TILTAN	YZMOVE[5]
CONTUR	ERRST	NAME	REGIN	TOLER	ZAXIS
COORD	FAST	NECK	REGOUT	TOLMOD	ZLARGE
COPPER	FILTER	NEGATV	RELAX	TOLREF	ZMOVE[5]
COPY	FINE	NFIXED	RFS	TOTAL	ZSMALL
CORNER	FIRST	NOAREA	RIGHT	TRIM	ZSTART
CREATE	FLAT	NONE	ROUGH	TYPIN	ZTROT
CRT	GCHAR	NOPROJ	RTHETA	UDIREC	ZVALUE
CURDEP	GLASS			ULEFT	
CURVE					

5. Reserved for future use.

Overview of Major Words 3

Overview of Major Words 3

Branching and Conditional Major Words

Use the following major words to test for logical conditions or to transfer control from one section of the program to another section:

Major Word	Description
GOTO	Branches unconditionally to a statement label.
Computed GOTO	Branches conditionally to a statement label.
IF	Branches conditionally or assigns a value.
THEN	Indicates that the following statements are executed only if the condition of the IF statement is true.
ELSE	Indicates that the following statements are executed only if the condition of the IF statement is false.
ELSEIF	Checks a secondary condition only if the first condition is false.
ENDIF	Ends an IF block.
FOR	Creates an iterative loop for repetitive operations.
EOFI	Terminates a FOR loop.
JUMPTO	Branches unconditionally to a statement label.

Modal Major Words

Use the following major words to set the modals for system operation:

Major Word	Description
BLANK	Blanks all subsequent entities.
CURSOR	Defines the input device for cursor control.
DISDEF	Displays subsequent entities only in the view of definition.
DISTOL	Sets the display tolerance.
FONT	Sets the line font for all subsequently created entities.
LEVEL	Assigns a level number to all subsequently created entities.
MSFILE	Changes the menu string file.
PAINT	Sets the drawing of entities on or off.
PEN	Sets the pen number.
RECOVR	Sets GPL recovery file on or off.
REFRES	Turns on the refresh buffer of the workstation.

Major Word	Description
RESCAL	Sets the RESCALE? prompt on or off.
SELENT	Selects the method of entity selection (single, chain, region in, or region out).
SELGRP	Turns on the ability to select single entities from a group.
SELMOD	Sets the method of entity selection on or off (sequence number, pointer, and entity names).
SPATHS	Sets the number of surface paths for a surface.
STATUS	Suppresses fatal error messages.
SYSDEC	Sets the system decimal places.
ZSURF	Sets the current transform coordinate depth (the zt-axis).

Entity and Character String Management Major Words

Use the following major words to manage the data base for entities:

Major Word	Description
ASSIGN	Assigns a name to an entity using its sequence number or its local GPL name.
ATTRIB	Attaches attributes to an entity.
CHECK	Tests for a legal pointer.
CLEAR	Clears the name of an entity.
CMPCHR	Compares two character strings.
CMPENT	Tests an array of pointers to get the entry point.
CONVER	Converts a real number to a character string or converts a character string to a real number.
DEFINE	Generates a set of entities.
DELETE	Deletes entities.
EVALC	Evaluates a curve for coordinates or parameters.
EVALS	Evaluates a surface for coordinates or parameters.
MOVCHR	Transfers a character string.
MOVENT	Transfers pointers.
OBTAIN	Extracts data from the TAB1, TAB2, or TAB3 data bases.
SETCHL	Sets the character length of a string.
TRIME	Trims an entity.

Variable Declaration, RTL I/O, and File Major Words

Use the following major words to declare text variables and entities, manage variables, and manage the part data base:

Major Word	Description
CHAR	Declares and reserves storage space for a text variable.
COMMON	Declares a common block for any following variables.
CONST	Assigns numerical value to a symbolic name.
DATA	Initializes a variable to a value.
ENDCOM	Terminates a common block.
ENTITY	Reserves storage space for entity data base pointers.
FILE	Catalogs the drawing for later modification or retrieval.
GET	Retrieves simple variables from the RTL.
REAL	Reserves storage space for real variables.
SAVE	Saves simple variables in the RTL.

Program Management Major Words

Use the following major words to manage the program:

Major Word	Description
CALL	Calls a GPL subroutine (refer to the PROC statement described below).
CONTIN	Continues program execution at the next GPL statement. Used as a labelled statement in structured programs.
DATE	Retrieves the current system date.
FINI	Indicates the end of a GPL program.
MAIN	Indicates the first statement in a GPL program.
MSTRNG	Executes a menu string from within a GPL program.
PAUSE	Halts the program run. The program can be resumed using menu 5.13.2 CONTINUE GPL PROGRAM.
PROC	Indicates the first statement in a GPL program used as a subroutine.
REMARK	Adds descriptive comments to the part program.
RETURN	Returns control from the subroutine to the calling program.

Major Word	Description
STOP	Terminates the execution of a GPL program. This is different from the FINI statement, which is always placed at the end of a GPL program.
TIME	Retrieves the current system time.

Display Control Major Words

Use the following major words to control the view display:

Major Word	Description
BLANKE	Blanks selected entities.
CHANGE	Changes current work view.
MAP	Maps transform coordinates from one view to another.
REPANT	Repaints the display.
UNBLNK	Unblanks an entity.
VBORDS	Sets view borders on or off.
VIEW	Creates a new view.
VNAMES	Sets view names on or off.
VVECS	Sets view vectors on or off.
ZOOM	Zooms a view.

Entity Definition Major Words

Use the following major words to create entities:

Major Word	Description
CIRCLE	Creates an arc or a circle.
CHAMFR	Creates a bevel near the intersection of two lines.
CMPCRV	Creates a composite curve.
CMSRF	Creates a curve mesh surface.
CONE	Creates a cone.
CYLNDR	Creates a cylinder.
DEVSRF	Creates a developable surface.
ELLIPS	Creates an ellipse.
FILLET	Creates a fillet near the intersection of two lines.
FILSRF	Creates a fillet surface.
GCONIC	Creates a general conic, that is, a circle, ellipse, hyperbola, or parabola.
HEXDRN	Creates a hexahedron.
HYPERB	Creates a hyperbola.
LINE	Creates a line.
LPSOID	Creates an ellipsoid.
MACCRV	Creates a machining/draft curve.
PARABO	Creates a parabola.
PLANE	Creates a plane.
POINT	Creates a unique point.
PTSET	Creates a point set.
REVSRF	Creates a surface of revolution.
RULSRF	Creates a ruled surface.
SPCURV	Creates the starting and ending conditions for a three-dimensional spline curve.

Major Word	Description
SPHERE	Creates a sphere.
SPLINE	Creates a spline through a series of points.
STRING	Creates a string figure.
TABCYL	Creates a tabulated cylinder.
TORUS	Creates a torus.
VECTOR	Creates a vector.

Entity Manipulation Major Words

Use the following major words to manipulate entities:

Major Word	Description
ARRAY	Copies a geometric entity in a rectangular or circular array.
GROUP	Creates a group of entities to be considered as a single logical unit (entity).
MIRROR	Creates a mirror (reversed) image of an entity or group of entities.
MODIFY	Modifies an entity.
PROJEC	Creates a three-dimensional figure by projecting a two-dimensional configuration onto a plane parallel to the original plane (ANSI 1973 and ANSI 1982).
RTRIEV	Retrieves a pattern.
TEMPLT	Creates a template.

Drafting Modal Major Words

Use the following major words to set the drafting modals for drafting operation:

Major Word	Description
AHEAD	Modifies the arrowhead length (ANSI 1973 and ANSI 1982).
ANGCTL	Controls the text angle (ANSI 1982).
ANUNIT	Sets the method for angular dimension representation (ANSI 1982).
ARAUTO	Determines the placement of arrows (ANSI 1982).
ARIN	Sets arrows inside witness lines for all subsequent dimensions (ANSI 1973 and ANSI 1982).
AROUT	Sets arrows outside witness lines for all subsequent dimensions (ANSI 1973 and ANSI 1982).
ARROW	Switches arrowhead alignment on or off (ANSI 1973 and ANSI 1982).
ATAIL	Controls the entry point of the tail location (ANSI 1982).
AUTOD	Controls auto-dimensioning (ANSI 1973 and ANSI 1982).
CDISPL	Sets character display ratios (ANSI 1973 and ANSI 1982).
CRES	Selects the output representation for standard set type characters (ANSI 1982).
CSET	Sets the character set used (ANSI 1973 and ANSI 1982).
CSIZE	Varies the character size (ANSI 1973 and ANSI 1982).
DECMAL	Sets the number of decimal places displayed in dimensioning for decimal numbers; changes the system from using integers to using real numbers (ANSI 1973 and ANSI 1982).
DIMOF	Sets the dimension offset distance (ANSI 1973 and ANSI 1982).
DIMORG	Sets which side of a generated dimension specifies the dimension origin (ANSI 1982).
DORIG	Sets the label and dimension origins (ANSI 1973).
DSCALE	Sets the drafting scale factor (ANSI 1973 and ANSI 1982).
DUAL	Sets dual dimensions on or off (ANSI 1973 and ANSI 1982).

Major Word	Description
FRACT	Changes the dimensioning display from using decimal numbers to using fractional (integer) numbers (ANSI 1973 and ANSI 1982).
KEYIN	Permits manual entry of dimension texts (ANSI 1973).
LDDIAM	Forces diameter symbol in linear dimensions (ANSI 1982).
LEADER	Determines the placement of the label leader with respect to the label text (ANSI 1982).
MATERL	Specifies the type of material for section lining (ANSI 1973 and ANSI 1982).
PREFIX	Changes the prefix character (ANSI 1982).
SECALN	Controls the alignment of section lining (ANSI 1982).
SECVIS	Determines whether to display section lining in all views or the view of definition only (ANSI 1982).
SLANT	Sets the slanted or vertical character set (ANSI 1973 and ANSI 1982).
TXTANG	Sets the text angle (ANSI 1973).
TXTJUS	Sets the text justification (ANSI 1973 and ANSI 1982).
TXTORG	Determines the method of indicating the position of text.
WLINE	Sets the display of witness lines for linear dimensions (ANSI 1973 and ANSI 1982).

The same word may operate slightly differently in ANSI 1973 and ANSI 1982. Check the appropriate chapter for complete details. ANSI 1973 statements are described in chapter 17 and ANSI 1982 statements are described in chapter 20.

Drafting Entity Definition Major Words

Use the following major words to create drafting entities:

Major Word	Description
ADIMEN	Creates angular dimensions (ANSI 1982).
BALOON	Draws a balloon with an arrow pointing to an entity (ANSI 1982).
CDIMEN	Creates a circular dimension label with a leader line (ANSI 1973 and ANSI 1982).
CLINE	Creates a centerline (ANSI 1973 and ANSI 1982).
CURARR	Draws an arrowhead at any position along an existing entity (ANSI 1982).
DATFEA	Displays a datum feature symbol and associated text within a feature frame symbol (ANSI 1982).
DATUM	Defines a datum target symbol (ANSI 1982).
DDIMEN	Creates a diametric dimension label (ANSI 1973 and ANSI 1982).
GEOTOL	Displays geometric tolerance or composite geometric tolerance symbols and associated text within a feature frame (ANSI 1982).
LABEL	Creates a label with leader line (ANSI 1973 and ANSI 1982).
LDIMEN	Creates a linear dimension label (ANSI 1973 and ANSI 1982).
MAGNFY	Produces a magnified drawing of a circular area (ANSI 1982).
MODDFT	Modifies certain drafting entities without having to redefine those entities (ANSI 1982).
NOTE	Creates a general note (ANSI 1973 and ANSI 1982).
RDIMEN	Creates an angular dimension with circular witness lines (ANSI 1973).
SECARR	Creates cross-section arrows (ANSI 1982).
SECTON	Automatically creates section lining lines for any closed figure (ANSI 1973 and ANSI 1982).
SRFTEX	Displays the standard basic symbol for surface texture (ANSI 1982).
TAPER	Creates a slope or taper dimension drawn to two lines (ANSI 1982).
THIKNS	Produces a dimension between two curves (ANSI 1982).

The same word may operate differently in ANSI 1973 and ANSI 1982. Check the appropriate chapter for complete details. ANSI 1973 statements are described in chapter 17 and ANSI 1982 statements are described in chapter 20.

Numerical Control Major Words

Use the following major words for numerical control operations.

Major Word	Description
SETGPG	Sets the N/C parameters in a generation parameter group (GPG).
TLPATH	Defines a toolpath from an existing entity, a list of entities or an entity array.

Interactive Command Major Words

Use the following major words for operations that request input from the person who is running the program:

Major Word	Description
DISPLA	Displays text and real variables.
MENU	Requests a selection from displayed menu choices during a GPL program run.
PARAMS	Requests values for a list of displayed data entry items (the values can then be used in the program).
POS	Requests a screen position selection using the graphics cursor.
QUERY	Requests a YES or NO to a displayed question.
SELECT	Request a selection from any entity displayed on the screen.
TEXT	Requests character string input.

Input/Output Major Words

Use the following major words for input/output operations:

Major Word	Description
BULK	Transfers bulk data.
CLOSE	Closes a previously opened file.
EXEC	Executes a FORTRAN subroutine.
OPEN	Opens a file.
READ	Inputs a file.
REWIND	Rewinds a file.
USTRUC	Outputs a UNISTRUC file.
WRITE	Outputs a file.

Branching and Conditional Statements 4

Branching and Conditional Statements 4

Branching and conditional statements control the execution of a GPL program.

Major Word	Description
GOTO	Either branches unconditionally or branches to specified statement labels (computed GOTO).
IF	Either branches conditionally or conditionally executes one or more statements.
THEN	Indicates that the following block of code to the next ELSE, ELSEIF, or ENDIF is executed only if the condition in the IF statement is true.
ELSE	Indicates that the following block of code to the next ENDIF is executed only if the condition on the IF statement is false.
ELSEIF	Checks a secondary condition only if the first condition is false.
ENDIF	Ends an IF block.
FOR	Creates an iterative loop for repetitive operations.
EOFI	Terminates a FOR loop.
JUMPTO	Branches unconditionally.

GOTO

The GOTO statement specifies an unconditional transfer of control.

Statement format:

```
GOTO statement label
```

Parameter	Description
statement label	The GPL statement at which the execution of the program should continue. It must be separated from the GOTO by a space.

Example:

Program Statement	Explanation
`GOTO 100`	Unconditionally transfers control to statement label 100.
`10 POINT/A,B`	This statement is ignored.
`100 A=A+1`	The program continues execution at this statement.

Computed GOTO

The computed GOTO branches to a label in a list of labels. The label indicated is specified by the value of a locator variable. If the locator is less than or greater than the number of labels given, the system displays:

```
RANGE ERROR IN COMPUTED GOTO
```

Statement format:

```
GOTO (label,...),locator
```

Parameter	Description
label	The list of statement labels. Labels must be separated by commas and enclosed in parentheses.
locator	The variable name whose value indicates the order of the label in the list of labels.

Example:

Program Statement	Explanation
`GOTO (25,35,45),X`	For this computed GOTO statement, if X is equal to 1, the program branches to statement label 25. If X is equal to 2, the program branches to statement label 35. And, if X is equal to 3, the program branches to statement label 45.

IF

The IF statement conditionally transfers control of the GPL program or conditionally executes one or more statements. The IF statement contains a logical condition to be evaluated as either true or false. If the logical expression is true, the conditional transfer or variable assignment is executed. If the logical expression is false, the conditional action is ignored, and execution continues with the next statement unless a condition statement block is specified by a THEN keyword. If the logical condition is false, and a statement block is specified by a THEN keyword, execution continues following the next ELSE, ELSEIF, or ENDIF block control keyword.

The condition expression can also be one or more logical expressions using the .OR. and .AND. operators. For example:

```
IF (A.LT.1.OR.A.GT.9) GOTO 9
IF (I.NE.1.AND.I.NE.5.AND.I.NE.11) THEN
```

The IF...THEN statement block can be formatted for easy readability in the following manner.

```
IF (condition) THEN
        Statements
ELSEIF (condition) THEN
        Statements
ELSE
        Statements
ENDIF
```

The following reserved words are used for control in the IF statement block:

Block Control Keywords	Description
THEN	Executes the following statements if the condition is true.
ELSEIF	Checks a secondary condition in an IF statement only if the first condition is false.
ELSE	Executes the following statements if none of the previous conditions are true.
ENDIF	Terminates a compound IF statement.

NOTE

.AND. and .OR. operators cannot be combined. A GPL program can use multiple .AND.s, or it can use multiple .OR.s, but it cannot use combined .AND.s and .OR.s.

Executing Conditionally

Statement format:

```
IF (value1.operator.value2) executable statement
```

Parameter	Description
value	The variables or constants to be compared in the condition. The condition must be enclosed in parentheses.
operator	The logical operation to be evaluated. The operator must have periods on both sides. The following keywords can be used as operators:

Logical Operator	Definition
LT	Less than
LE	Less than or equal to
EQ	Equal to
NE	Not equal to
GE	Greater than or equal to
GT	Greater than
AND	Logical and
OR	Logical inclusive or

The condition may be compound; that is, two or more logical conditions associated by AND or OR.

executable statement	Any executable statement (assignment, GOTO, or major word).

Examples:

Program Statement	Explanation
`QUAD=2`	Set the variable QUAD to have the value of 2.
`IF (ANG.LT.90) QUAD=1`	If the variable ANG is less than 90°, then set the variable QUAD to have the value of 1. This statement conditionally assigns a name to an expression.
`IF (QUAD.LT.2) GOTO 20`	If the variable QUAD is less than 2, then branch to statement label 20.

Program Statement	Explanation
`IF (I.EQ.1.AND.J.EQ.2) THEN`	If the compound condition is true (that is, I=1 and J=2), then the following block of statements is executed until the next ELSE, ELSEIF, or ENDIF.
`IF (STAT.NE.0) $` `  DISPLA/'STATUS', STAT`	If the variable STAT is not equal to 0, then execute the DISPLA statement.

FOR

The FOR statement and the EOFI statement are used to construct iterative loops. The FOR statement initiates the loop, and the EOFI statement terminates the loop.

The first time the loop is executed, the index variable has the initial value. Each time the loop is executed after that, the value of the index variable is changed by the step value until either:

- The index variable exceeds the final value (for a positive step value), or
- The index variable is exceeded by the value (for a negative value).

The program then terminates the loop and executes the next statement following the EOFI statement.

Statement format:

```
FOR index=initial value [,STEP step], UNTIL final value
```

Parameter	Description
index	Any simple variable name. It must be separated from the FOR word by a space. The index and the initial value must be joined by an equals sign.
initial value	The initial index value. It can be any integer value.
STEP	The minor word indicating that the next value is the step value of the statement.
step	The step, or increment, value. It can be any value; the default value is 1 if STEP is omitted.
UNTIL	The minor word indicating that the next value is the final value for the statement.
final value	The final value. It can be any value. If the final value is less than the initial value, and the step value is positive, the loop executes zero times.

EOFI

The EOFI statement is used to terminate an iterative loop initiated by a FOR statement.

Statement format:

```
EOFI
```

Example:

Program Statement	Explanation
`FOR I=2, STEP 2, UNTIL 10`	The FOR loop with variable I as an index. The loop is repeated with values for I of 2, 4, 6, 8, and 10, then the program continues with the statement after EOFI.
`J=I**2`	The variable J is set up to have the value of I squared. Each time through the loop, the current value of I is used.
`POINT/J,J`	A point is created at the coordinates specified by J, J. Each time through the loop, a new point is created using the expression I squared where the value of I is squared.
`EOFI`	The loop is ended. The loop creates points at the x- and y-coordinates of (4,4), (16,16), (36,36), (64,64), and (100,100).

JUMPTO

The JUMPTO statement branches unconditionally to the statement label indicated. Program execution continues at the specified statement label. This statement's function is identical to that of the GOTO statement.

Statement format:

```
JUMPTO/label
```

Parameter	Description
label	The number of the statement label.

Example:

Program Statement	Explanation
`JUMPTO/359` ⋮	The program branches to statement label 359.
`359 A=11`	The program resumes execution at statement label 359.

Modal Statements 5

Modal Statements 5

Modal statements set parameters for creating entities.

BLANK

The BLANK statement blanks all succeeding entities. The default value for BLANK is OFF.

Statement format:

```
BLANK/{ON }
      {OFF}
```

Parameter	Description
ON	Blanks all succeeding entities.
OFF	Does not blank all succeeding entities.

Example:

Program Statement	Explanation
`BLANK/OFF`	All succeeding entities are displayed as they are created.

CURSOR

The CURSOR statement determines the input device for graphics cursor control.

Statement format:

```
CURSOR/{CRT   }
       {TABLET}
```

Parameter	Description
CRT	The input device for graphics cursor control is the graphics terminal.
TABLET	The input device for graphics cursor control is the tablet and its stylus.

Example:

Program Statement	Explanation
`CURSOR/CRT`	The graphics terminal is the input device for the graphics cursor.

DISDEF

The DISDEF statement displays all subsequent entities in the view of definition only.

Statement format:

```
DISDEF/(ON  )
       (OFF )
```

Parameter	Description
ON	Displays all subsequent entities only in the view of definition.
OFF	Turns off view of definition display.

Example:

Program Statement	Explanation
`DISDEF/ON`	All subsequent entities are displayed only in the view of definition.

DISTOL

The DISTOL statement controls the display tolerance of curves. The display tolerance controls the pictorial representation of the straight line approximations of curves. This does not affect the mathematical representation of curves in the ICEM DDN data base.

Statement format:

```
DISTOL/tolerance
```

Parameter	Description
tolerance	The numerical value of the display tolerance. The display is coarser as the tolerance becomes larger with fewer approximating line segments used for the display. The display is finer as the tolerance becomes smaller with more line segments used for the display.

Example:

Program Statement	Explanation
`DISTOL/.001`	The display tolerance for all subsequent curves is 0.001.

FONT

The FONT statement defines the line font for subsequently defined lines and curves. The default font mode is normal weight and solid. Conics, splines, strings, labels, and dimensions are always normal weight and solid.

Statement format:

```
        ⎧SOLID  ⎫
FONT/   ⎪DASHED ⎪
        ⎪PHANTM ⎪
        ⎩CENLIN ⎭
```

Parameter	Description
SOLID	The minor word indicating solid lines continuously over the entire length of the entity.
DASHED	The minor word indicating dashed lines of equal segments separated by equal spaces. The segment is a nominal 3.16 mm (0.125 in) and the space is 0.79 mm (0.031 in). A dashed entity has a minimum of three parts, that is, two segments and a space. The dashed line is forced solid if the line length is less than 7.10 mm (0.281 in). A dashed circular line is segmented to generate approximate arc lengths of 3.16 mm (0.125 in) and spacing of 0.79 mm (0.031 in). A dashed arc always starts and ends with a segment. A dashed circle always has a segment centered about 0°.
PHANTM	The minor word indicating phantom lines made of a dashed line with a long segment, followed by two short segments. Spaces between all segments are equal. A phantom line starts and ends with a long segment. A phantom circle starts with a long segment centered at 0° and ends with short segments. The long segment is adjusted to assure proper closing. The segments are: • Long: 19.0 mm nominal (0.75 in) • Short: 3.16 mm (0.125 in) • Space: 0.79 mm (0.031 in)
CENLIN	The minor word indicating a centerline made up of a long segment followed by a short segment. Spacing between elements is equal.

Example:

Program Statement	Explanation
`FONT/DASHED`	All subsequent entities are displayed in a dashed line font.

LEVEL

The LEVEL statement changes the level number that you are using. Until the level is changed again, all subsequent entities are defined on this level. Level numbers 0 through 1023 are allowed.

Statement format:

```
LEVEL/level number
```

Parameter	Description
level number	The number of the level where you want to create new entities.

Example:

Program Statement	Explanation
LEVEL/3	All new entities are created on level number 3.

MSFILE

The MSFILE statement changes the menu string file from which your external menu strings are executed. All subsequent references to external menu strings, whether from the MSTRING statement or from a tablet, will use this file.

The default file name is MSTRING.

You also have the option of loading the original mstring filename into a GPL text variable. You can then have the GPL program reset the filename name back to the original filename before terminating the program.

Statement format:

```
MSFILE/(text variable)],ofn text var]
       ('mslfn'      )
```

Parameter	Description
text variable	The name of a text variable containing the file name that contains the menu strings that are accessed by subsequent MSTRNG statements. This variable should be dimensioned to 7 characters.
'mslfn'	The name of a local, external file (enclosed in single quotes) that contains the menu strings that are accessed by subsequent MSTRNG statements.
ofn text var	The name of a text variable that receives the original local mstring filename. This variable should be dimensioned to 7 characters.

The following error message is issued if the MSTRING file is not found or cannot be retrieved.

```
MENU STRING FILE CANNOT BE RETRIEVED
```

Example:

Program Statement	Explanation
`CHAR/OLFIL(7)`	Dimension the variable that receives the old menu string file to 7.
`MSFILE/'MSFL1',OLFIL`	Change the menu string file to 'MSFL1'. Save the name of the old file in OLFIL.
`MSTRNG/NAME,TLPATH`	Execute the menu string TLPATH which resides on the current menu string file (MSFL1).
`MSFILE/OLFIL`	Change the menu string file back to the original file.

PAINT

The PAINT statement turns on or off the display of entities at creation time. The default is ON.

Statement format:

```
PAINT/{ON }
      {OFF}
```

Parameter	Description
ON	Turns on the drawing of entities.
OFF	Turns off the drawing of entities. They are not drawn until a REPANT is done.

Example:

Program Statement	Explanation
`PAINT/ON`	All subsequent entities are displayed as they are created.

PEN

The PEN statement sets a new pen number for all subsequent entities. The pen number remains in effect until a new pen number is selected.

Statement format:

```
PEN/pen number
```

Parameter	Description
pen number	The number of the pen used in drawing subsequent entities. Pens are numbered from 0 to 16.

Example:

Program Statement	Explanation
`PEN/10`	All subsequent entities are drawn with pen number 10.

RECOVR

The RECOVR statement writes the GPL recovery file to file GPLREC when a FILE statement is encountered. GPLREC will contain any geometry created after the last FILE statement up to the FINI statement or a program abort.

Statement format:

```
RECOVR/(ON  )
       (OFF )
```

Parameter	Description
ON	Writes file GPLREC when a FILE statement is encountered.
OFF	Does not write file GPLREC when a FILE statement is encountered.

REFRES

The REFRES statement places all entities created in the workstation refresh buffer. If the workstation is not used, this statement is ignored.

Statement format:

```
REFRES/(ON  )
       (OFF )
```

Parameter	Description
ON	Places all entities created in the workstation refresh buffer.
OFF	The refresh mode is not used.

Example:

Program Statement	Explanation
REFRES/ON	The refresh buffer is used as storage for the created entities.

RESCAL

The RESCAL statement suppresses the RESCALE? message during the display of an entity if part of the entity is located outside the current view area.

Statement format:

```
RESCAL/{ON }
       {OFF}
```

Parameter	Description
ON	Displays the RESCALE? message if part of an entity is outside the current view area.
OFF	Does not display the RESCALE? message if part of an entity is outside the current view area.

Example:

Program Statement	Explanation
`RESCAL/ON`	The RESCALE? message is displayed if part of an entity is outside the current view area.

SELENT

The SELENT statement indicates the form of entity selection.

Statement format:

```
        {SINGLE}
        {CHAIN }
SELENT/{REGIN }
        {REGOUT}
        {USER  }
```

Parameter	Description
SINGLE	Selects single entities.
CHAIN	Selects chained entities.
REGIN	Selects all the entities within a region.
REGOUT	Selects all the entities outside a region.
USER	The user chooses the means of selection interactively, then selects.

Example:

Program Statement	Explanation
`SELENT/SINGLE`	Entities are selected singly.

SELGRP

The SELGRP statement allows selection of a group member or composite curve member.

Statement format:

```
SELGRP/{ON }
       {OFF}
```

Parameter	Description
ON	Indicates that an entity cannot be selected from a group or composite curve.
OFF	Indicates that an entity can be selected from a group or composite curve.

Example:

Program Statement	Explanation
SELMOD/ON	Entities cannot be selected from a group or composite curve. Only the group or composite curve can be selected.

SELMOD

The SELMOD statement sets the method of entity selection.

Statement format:

```
SELMOD/{ON }
       {OFF}
```

Parameter	Description
ON	Allows for selection of entities by methods other than screen position (that is, entity name, entity pointer, sequence number, and so forth).
OFF	Allows for selection of entities by screen position.

SPATHS

The SPATHS statement modifies the surface display path modals.

Statement format:

```
SPATHS/UPATHS,upaths,VPATHS,vpaths,UDIREC,upoints,VDIREC,vpoints
```

Parameter	Description
UPATHS	The minor word indicating the number of upaths.
upaths	The number of paths in the u direction.
VPATHS	The minor word indicating the number of vpaths.
vpaths	The number of paths in the v direction.
UDIREC	The minor word indicating the number of upoints.
upoints	The number of points per upath.
VDIREC	The minor word indicating the number of vpoints.
vpoints	The number of points per vpath.

Example:

Program Statement	Explanation
`SPATHS/UPATHS,7,VPATHS,11,$` `UDIREC,16,VDIREC,24`	All subsequent surfaces are created with 7 upaths, 11 vpaths, 16 upoints, and 24 vpoints.

STATUS

The STATUS statement suppresses fatal error messages during a GPL run. After entity construction statements, use the OBTAIN/ERRST command to check the error code for bad pointer errors. Bad pointer errors affect the creation of subsequent entities that use the bad pointers as references.

Statement format:

```
STATUS/(ON  )
       (OFF )
```

Parameter	Description
ON	Suppresses fatal error messages during a GPL run.
OFF	Displays fatal error messages during a GPL run.

Example:

Program Statement	Explanation
`STATUS/ON`	Fatal errors are not indicated during a GPL run.

SYSDEC

The SYSDEC statement controls the number of decimal places after the decimal point for all displayed real number input and output.

Statement format:

```
SYSDEC/number
```

Parameter	Description
number	The number of decimal places after the decimal point.

Example:

Program Statement	Explanation
`SYSDEC/6`	All real number input and output is displayed with 6 decimal places.

ZSURF

The ZSURF statement sets the current value of the Z plane so that only x- and y-coordinates are required.

Statement format:

```
ZSURF/depth
```

Parameter	Description
depth	The z depth or work plane where all subsequent entities are created.

Example:

Program Statement	Explanation
`ZSURF/1.5`	All subsequent two-dimensional entities are created at a z depth of 1.5 in.

Entity and Character String Statements 6

Entity and Character String Statements 6

Entity and character string statements perform different management functions. Several of these definitions are equivalent to types and forms found in menus 3 DELETE and 5 SPECIAL FUNCTIONS.

ASSIGN

The ASSIGN statement defines a name in the data base for an entity that is identified by its sequence number or its local GPL name. The name assigned can be either simple or subscripted. The sequence number can be expressed either as an integer value or by a variable name.

By Sequence Number

Statement format:

```
ASSIGN/SEQNUM,number,( 'name'        )[,status]
                     (text variable)
```

Parameter	Description
SEQNUM	The minor word indicating sequence number assignment.
number	The sequence number assigned to the entity.
'name'	The name in single quotes assigned to the entity. This name must not exceed 10 characters.
text variable	A text variable that contains the entity name.
status	A variable that contains the status of the assignment. If status is provided, it can have the following values:

Value	Description
0	The assignment was successful.
-1	Another entity already has that name.

If status is not provided and the entity name already exists, it is detached from its previous use and is assigned to this entity.

Example:

Program Statement	Explanation
`ASSIGN/SEQNUM,25,$` `  'ENT1',STAT`	Entity with sequence number 25 is given the name ENT1 in the text.

By Entity

Statement format:

```
ASSIGN/entity,( 'name'      )[,status]
              (text variable)
```

Parameter	Description
entity	The entity name.
'name'	The name in single quotes assigned to the entity.
text variable	A text variable that contains the name of the entity.
status	A variable that contains the status of the assignment.

Value	Description
0	The assignment was successful.
-1	Another entity already has that name.

If status is not provided and the entity name already exists, it is detached from its previous use and is assigned to this entity.

Example:

Program Statement	Explanation
`ASSIGN/SN(3),ENT2`	Entity SN(3) is given the name contained in the text variable ENT2.

ATTRIB

The ATTRIB statement attaches, copies, or deletes attributes from entities. Attributes consist of any textual information attached to an entity. Subattributes are combinations of text or numeric information.

The textual information can be entered directly using single quotes or indirectly using a text variable (refer to the TEXT command).

Creating and Associating Attributes and Subattributes with an Entity

Statement format:

```
ATTRIB/[entity[,...]  ],ATNAME,['attribute name'],SUBAT,[NAME,name[,...]          ]
       [LEVL,level    ]        [text variable    ]       [NUMBER,value[,...]      ]
                                                         [BOTH,name,value[,...]   ]
```

Parameter	Description
entity	The name or names of the entity assigned to this attribute. Names must be separated by commas. A maximum of 20 entities may be selected.
LEVL	The minor word indicating that attributes are to be assigned by level numbers. The attributes are assigned to all entities on that level.
level	The level number of the entities to be assigned this attribute.
ATNAME	The minor word indicating the attribute name.
'attribute name'	The attribute name enclosed in single quotes.
text variable	A text variable containing the attribute name.
SUBAT	The minor word indicating the subattribute names or numbers.
NAME	The minor word indicating the subattribute names.
name	The subattribute name or names assigned to the attribute can be either a text variable containing the name or the name itself enclosed in single quotes ('name'). Names must be separated by commas.
NUMBER	The minor word indicating the subattribute values.
value	The subattribute value or values assigned to the attribute. Values must be separated by commas.
BOTH	The minor word indicating both subattribute names and values. The names and values must be specified in the following order: name, value. Names and values must be separated by commas.

Examples:

Program Statement

```
ATTRIB/PT002,LN003,LN004$
     ,LN005,SP005,SP006$
     ,ATNAME,'CHANNEL'$
     ,SUBAT,BOTH$
     ,'WIDTH',12.0$
     ,'HEIGHT',16.0$
     ,'LENGTH',145.5$
```

Explanation

The attribute name 'CHANNEL' is assigned to the following entities: point PT002; lines LN003, LN004, and LN005; and splines SP005 and SP006.

In addition, the entities are assigned the following three subattributes of text and numerical data:

Text	Numerical Data
WIDTH	12.0
HEIGHT	16.0
LENGTH	145.5

Program Statement

```
ATTRIB/LEVL,104$
     ,ATNAME,XTIME
```

Explanation

All entities on level 104 have the attribute name stored in the variable XTIME.

The new attributes and subattributes are inserted at the end of the list of existing attributes and subattributes for that entity.

Deleting Attributes and Subattributes from an Entity

Statement format:

```
ATTRIB/DELATR,entity[,ATNAME,atrank[,...][,SUBAT,subrank,...]]
```

Parameter	Description
DELATR	The minor word indicating that attributes are to be deleted.
entity	The name of the entity from which the attributes are to be deleted.
ATNAME	The minor word indicating the numerical rank number of the attribute name. If ATNAME alone is specified, all attributes and subattributes are deleted.
atrank	The rank number of the attributes you want to delete. Ranks must be separated by commas. The rank refers to the numerical order of the attributes. If ATNAME is specified, only attributes indicated are deleted.
SUBAT	The minor word indicating the numerical rank number of the subattribute names or numbers. If omitted, all the subattributes of the indicated attribute are deleted.
subrank	The rank number or numbers of the subattributes to be deleted. Ranks must be separated by commas. The rank refers to the numerical order of the subattributes. If SUBAT is specified, only subattributes indicated are deleted.

If ATNAME is not specified, all attributes and subattributes associated with the given entity are deleted. If ATNAME, but not SUBAT is specified, the attribute and all associated subattributes are deleted. If both are specified, only subattributes are deleted.

Example:

Program Statement	Explanation

`ATTRIB/DELATR,PT003,PT006$`
`,PT122,ATNAME,3$`
`,SUBAT,5,6,7`
	Subattribute numbers 5, 6, and 7 are deleted from points PT003, PT006, and PT122.

Copying All Attributes and Subattributes from One Entity to Another

Statement format:

```
ATTRIB/COPY,entity1,entity2[,(BEFORE)]
                           [(AFTER )]
```

Parameter	Description
COPY	The minor word indicating that the attributes of the first entity are copied to the second entity.
entity1	The name of the entity whose attributes are copied.
entity2	The name of the entity that receives the copied attributes.
BEFORE	The minor word indicating that the copied attributes are to be placed before the attribute list of the second entity.
AFTER	The minor word indicating that the copied attributes are to be placed after the attribute list of the second entity.

If neither BEFORE nor AFTER is specified, AFTER is assumed.

Example:

Program Statement	Explanation
`ATTRIB/COPY,LN145$` `,LN322,BEFORE`	All of the attributes of entity LN145 are copied before the attribute list of entity LN322.

CHECK

The CHECK statement tests an entity for a legal ICEM DDN pointer.

Statement format:

```
CHECK/entity,result
```

Parameter	Description
entity	The name of the entity tested.
result	The name of the variable that receives the result of the test for a legal pointer. The result values are:

Result	Description
0	The entity has a legal pointer.
-1	The entity has an illegal pointer.

NOTE

Manipulating entities outside a GPL program and then entering the program using menu 5.13.2 CONTINUE GPL PROGRAM can produce unpredictable pointers.

Example:

Program Statement	Explanation
`CHECK/PT003,TEST` `IF (TEST.EQ.0) GO TO 200`	Entity PT003 is tested for a legal pointer and the variable TEST receives the result. If PT003 has a legal pointer, the program branches to statement 200.

CLEAR

The CLEAR statement removes the name associated with an entity from the data base.

Statement format:

```
CLEAR/{  'name'       }
      {text variable  }
```

Parameter	Description
'name'	The name of the entity in single quotes.
text variable	The name of a text variable that contains the name of the entity.

Example:

Program Statement	Explanation
`CLEAR/'PT51'`	Removes the name PT51 from the entity it is associated with in the data base.

CMPCHR

The CMPCHR statement compares one character string with another character string and returns a result indicating whether the two strings are equal.

Statement format:

```
CMPCHR/number,string1,chpos1,string2,chpos2,result
```

Parameter	Description
number	The number of characters to compare.
string1	The name of the first text variable or the first character string.
chpos1	The position of the first character to test in string1.
string2	The name of the second text variable or the second character string.
chpos2	The position of the first character to test in string2.
result	The name of the variable that receives the result of the test for the equality of the two character strings. The result values are:

Result	Description
0	The character strings are equal.
-1	The character strings are not equal.

Example:

Program Statement	Explanation
`CMPCHR/6,NOTE1,10,$` `NOTE2,31,TEST`	Text variable NOTE2 is tested for a 6-character string that matches a 6-character string in NOTE1. The variable TEST receives the result of the test.

Text Variable	Text
NOTE1	PART NO. 23-334.
NOTE2	O-RINGS ARE REQUIRED FOR PART 23-334.

For these two notes:

- The 6-character string in NOTE1 at character position 10 matches the 6-character string in NOTE2 at character position 31.
- The result value is 0.

CMPENT

The CMPENT statement compares one entity with an array of entity pointers to get the appropriate index into the entity array.

Statement format:

```
CMPENT/NUMBER,number,array,entity,result
```

Parameter	Description
NUMBER	The minor word for indicating that the value is the number of array elements.
number	The number of array elements.
array	The name of the array of entity pointers.
entity	The name of the entity checked.
result	The name of the variable that receives the result of checking the entity against the pointer array. The result values are:

Result	Description
-1	The array does not contain the entity.
>0	The resulting index number.

Example:

Program Statement	Explanation
`CMPENT/NUMBER,10,PT,$` `  PT3,TEST`	Entity pointer array PT is tested for entity PT3 and the variable TEST receives the result. If the array does not contain a pointer for PT3, the result value is -1.

CONVER

The CONVER statement either converts a real number into a character string or converts a character string into a real number. The maximum length of the character string is 10 characters.

Converting a Real Number into a Character String

Statement format:

```
CONVER/[POSITV],decimal places,NUMBER,variable,text variable
       [NEGATV]
```

Parameter	Description
POSITV	The minor word indicating that the characters in the string are to be right-justified and may have leading blanks.
NEGATV	The minor word indicating that the characters in the string are to be left-justified.
decimal places	The number of decimal places after the decimal point in the real number.
NUMBER	The minor word indicating that the following variable contains a real number that is to be converted to a character string and placed in the specified text variable.
variable	The name of the variable that contains the real number.
text variable	The name of the text variable that receives the converted number.

Example:

Program Statement	Explanation
`CONVER/POSITV,4,NUMBER$` `,L,LENGTH`	Real variable L is converted into a character string in text variable LENGTH. The number is stored in the character string positively to the value of four decimal places.

Converting a Character String into a Real Number

Statement format:

```
CONVER/NUMBER,text variable,variable,status
```

Parameter	Description
NUMBER	The minor word indicating that the following text variable is to be converted into a real number and placed in the specified variable.
text variable	The name of the text variable that contains the character string.
variable	The name of the variable that receives the converted real number.
status	The name of the variable that receives the status of the conversion. The status values are:

Status	Description
0	The real number was converted correctly.
-1	The character string contained illegal characters.

Example:

Program Statement	Explanation
`CONVER/NUMBER,LENGTH,$` `  L,TEST`	Text variable LENGTH is converted to real variable L. The variable TEST receives the status.

DEFINE

The DEFINE statement generates a number of points, lines, or arcs with one statement. The coordinate array size must be a multiple of 3 (points), 6 (lines), or 9 (arcs) times the number of entities. If an entity array is specified, the array size must equal the number of entities given.

Statement format:

```
        ⎧POINTS⎫
DEFINE/⎨LINES ⎬,number,coord array[,entity array]
        ⎩ARCS  ⎭
```

Parameter	Description
POINTS	The minor word for defining a number of points. Each three values in the coordinate array are the transform coordinates of a point.
LINES	The minor word for defining a number of lines. Each six values in the coordinate array are the transform coordinates of the endpoints of a line.
ARCS	The minor word for defining a number of arcs. Each nine values in the coordinate array are the transform coordinates of three points that define an arc (the arc extends from the first point through the second point and ends at the third point).
number	The number of entities defined.
coord array	The name of a coordinate array.
entity array	The name of an entity array that receives the newly defined entities.

Example:

Program Statement	Explanation
`REAL/PNTCOR(12)`	Reserve space for 12 elements of PNTCOR.
`ENTITY/PNTENT(4)`	Receive space for 4 entities in PNTENT.
`DATA/PNTCOR,1,2,0,3,4,0$` `  5,6,0,7,8,0`	Initialize PNTCOR with 4 sets of point coordinates.
`DEFINE/POINTS,4,PNTCOR(1)$` `  PNTENT(1)`	Define 4 points in the PNTENT array from the coordinates stored in PNTCOR. PNTENT(1) has the coordinates (1,2,0); PNTENT(2) has the coordinates (3,4,0); and so forth.

DELETE

The DELETE statement deletes entities not required for subsequent definitions.

Deleting Single Entities

Statement format:

```
DELETE/name[,...]
```

Parameter	Description
name	The name or names of the entities that are to be deleted.

Example:

Program Statement	Explanation
`DELETE/PT1,LN1,CIR4`	Point PT1, line LN1, and circle CIR4 are deleted.

Deleting from an Entity Array

Statement format:

```
DELETE/NUMBER,count,entity array
```

Parameter	Description
NUMBER	The minor word indicating that a number of entities are to be deleted from an entity array.
count	The number of entities to be deleted from the entity array.
entity array	The name of the entity array.

Example:

Program Statement	Explanation
`DELETE/NUMBER,10,ET(1)`	The first 10 elements of entity array ET are deleted.

Deleting All Points in a Single Operation

Statement format:

```
DELETE/POINTS
```

Parameter	Description
POINTS	The minor word indicating that all points are to be deleted from the part drawing.

NOTE

We do not recommend using this statement if points have been used in the definition of other entities.

EVALC

The EVALC statement evaluates a curve and returns either the coordinates of a point associated with a specified parameter or the value of a parameter for a specified point on the curve.

Parameter Returns Point

Statement format:

```
EVALC/PARAM,entity,parameter,{MODVW|DEFVW|CURVW},array1,array2,status
```

Parameter	Description
PARAM	The minor word indicating that point coordinates are to be returned from a specified parameter.
entity	The name of the entity to be evaluated. Lines, arcs, conics, splines, three-dimensional splines, bezier curves, and machining curves are allowable entity types.
parameter	The parameter specified. Refer to the System Programmer's Reference Manual for allowable parameters.
MODVW	The minor word indicating that the coordinates of the point are to be returned in model space.
DEFVW	The minor word indicating that the coordinates of the point are to be returned in the definition space of the indicated entity.
CURVW	The minor word indicating that the coordinates of the point are to be returned in the current workspace.
array1	The name of an array that receives the three-dimensional point coordinates.
array2	The name of an array that receives the first derivative of the curve.
status	The name of a variable that receives the status of the evaluation. The status values are:

Value	Description
0	The operation was successful.
1	The parameter was outside the defined interval of the curve.
-1	The entity selected was an illegal entity type.

Point Returns Parameter

Statement format:

```
EVALC/POINTS,entity,array,{MODVW|DEFVW|CURVW},parameter,status
```

Parameter	Description
POINTS	The minor word indicating that a parameter is to be returned from a specified point.
entity	The name of the entity to be evaluated. Lines, arcs, conics, splines, three-dimensional splines, bezier curves, and machining curves are allowable entity types.
array	The name of an array that contains the three-dimensional point coordinates.
MODVW	The minor word indicating that the coordinates of the point are given in model space.
DEFVW	The minor word indicating that the coordinates of the point are given in definition space of the indicated entity.
CURVW	The minor word indicating that the coordinates of the point are given in the current workspace.
parameter	The name of a variable that receives the parameter returned. Refer to the System Programmer's Reference Manual for allowable parameters.
status	The name of a variable that receives the status of the evaluation. The status values are:

Value	Description
0	The operation was successful.
-1	The entity selected was an illegal entity type.
-2	The point was not on the curve.

EVALS

The EVALS statement evaluates a surface and returns either the coordinates of a point associated with specified parameters or the values of the parameters of a specified point.

Parameter Returns Point

Statement format:

```
EVALS/PARAM,entity,param array,(MODVW|DEFVW|CURVW),array1,array2,array3,array4,status
```

Parameter	Description
PARAM	The minor word indicating that u and v point coordinates are to be returned from a specified parameter.
entity	The name of the entity to be evaluated. All surfaces are allowed.
param array	The name of an array that contains the u and v parameters specified. The parameter usually has a value between 0 and 1 (refer to the System Programmer's Reference Manual). This number may not always be a value between 0 and 1. For example, curve mesh surface has a value between 0 and the number of fixed curves minus 1.
MODVW	The minor word indicating that the coordinates of the point are to be returned in model space.
DEFVW	The minor word indicating that the coordinates of the point are to be returned in the definition space of the indicated entity.
CURVW	The minor word indicating that the coordinates of the point are to be returned in the current workspace.
array1	The name of an array that receives the three-dimensional point coordinates.
array2	The name of an array that receives the unitized normal of the curve.
array3	The name of an array that receives the first derivative of the curve in the u direction.
array4	The name of an array that receives the first derivative of the curve in the v direction.

Parameter	Description
status	The name of a variable that receives the status of the evaluation. The status values are:

Value	Description
0	The operation was successful.
1	The parameter was outside the defined interval of the curve.
-1	The entity selected was an illegal entity type.

Point Returns Parameter

Statement format:

```
EVALS/POINTS,entity,array,(MODVW|DEFVW|CURVW),param,[arr1[,arr2[,arr3[,arr4]]]],status
```

Parameter	Description
POINTS	The minor word indicating that a parameter is to be returned from a specified point.
entity	The name of the entity to be evaluated.
array	The name of an array that contains the three-dimensional point coordinates.
MODVW	The minor word indicating that the coordinates of the point are given in model space.
DEFVW	The minor word indicating that the coordinates of the point are given in the definition space of the indicated entity.
CURVW	The minor word indicating that the coordinates of the point are given in the current workspace.
param	The name of the array that receives the u and v parameters returned. The parameter usually has a value between 0 and 1. Refer to the System Programmer's Reference Manual. This number may not always be a value between 0 and 1. For example, curve mesh surface has a value between 0 and the number of fixed curves minus 1.
arr1	The name of the array that receives the three-dimensional point coordinates.
arr2	The name of the array that receives the unitized normal of the curve.
arr3	The name of the array that receives the first derivative of the curve in the u direction.
arr4	The name of the array that receives the first derivative of the curve in the v direction.
status	The name of the variable that receives the status of the evaluation. It has the following values:

Value	Description
0	The operation was successful.
-1	The entity selected was an illegal entity type.
-2	The point was not on the curve.

MOVCHR

The MOVCHR statement creates a text variable by moving characters from one text variable or character string to another text variable.

Statement format:

```
MOVCHR/nchr,{textvar1 },chpos1,textvar2,chpos2
            {'text'   }
```

Parameter	Description
nchr	The number of characters to be moved.
textvar1	The name of the text variable containing the text to be moved.
'text'	The character string to be moved enclosed in single quotes.
chpos1	The position of the first character to be moved.
textvar2	The name of the text variable receiving the text.
chpos2	The position at which to place the first moved character.

MOVENT

The MOVENT statement moves entities from one array to another array.

Statement format:

```
MOVENT/number,entity array1,entity array2
```

Parameter	Description
number	The number of entities to be moved.
entity array1	The name of the source entity array.
entity array2	The name of the destination entity array.

OBTAIN

The OBTAIN statement obtains or retrieves information from the database. The information is extracted from the ICEM DDN common arrays. The information obtained is assigned to variables specified in the command for later use in the GPL program.

The common arrays from which data can be obtained are TAB1 (master entity list), TAB2 (entity-dependent integer and character string data), TAB3 (entity-dependent real data, TAB4 (View Information), and TAB5 (entity names and pointers). Refer to the System Programmer's Reference Manual for more information on the contents of the common arrays. Also, see appendix E for more information on database I/O.

There are 15 forms of this statement.

Coordinates

Statement format:

```
OBTAIN/COORD,entity,array
```

Parameter	Description
COORD	The minor word for obtaining coordinate data.
entity	The entity for which data is obtained. The entity types allowed are points, lines, circles, and all drafting entities.
array	The name of an array in which to store the data. The size of the array (the number of words required) is dependent upon the entity type selected. The number of elements for each entity is:

Entity Type	Number of Elements	Explanation
Points	3	Points require three elements per entity to store the coordinates of each point.
Lines	6	Lines require six elements per entity to store the coordinates of each line.
Circles	6	Circles require six elements per entity to store the start and end angles in radians (2), the coordinates of the center point (3), and the radius (1).
Drafting entities	2	Drafting entities return the text origin (xt and yt).

NOTE

The coordinates retrieved will be model space coordinates.

Example:

Program Statements	Explanation
`REAL/ARR(3)`	Sets up an array for the point coordinates.
`LN1=LINE/100,100,0$` `,200,200,0`	Creates the first line.
`LN2=LINE/100,200,0$` `,200,100,0`	Creates the second line.
`PT1=POINT/INTOF,XLARGE$` `,LN1,LN2`	Finds the point of intersection.
`OBTAIN/COORD,PT1,ARR(1)`	Obtains the point coordinates and store them in array ARR.

Number of Characters

Statement format:

```
OBTAIN/NUMBER,{text variable}  ,variable
              {  'text'     }
```

Parameter	Description
NUMBER	The minor word for obtaining the number of characters in a text string.
text variable	The name of the text variable to be checked for the number of characters.
'text'	The text string enclosed in single quotes to be checked for the number of characters.
variable	The name of the variable that receives the number of characters.

Example:

The following statement extracts the number of characters from the string given.

```
OBTAIN/NUMBER,'NUMBER OF CHARACTERS IN THIS STRING',XN
```

The value is returned to the variable XN.

View Pointer

Statement format:

```
OBTAIN/STDVW,view number,view pointer,status
```

Parameter	Description
STDVW	The minor word for obtaining a standard view.
view number	The view number, which must be less than or equal to the number of views defined in ICEM DDN.
view pointer	The name of the variable to contain the resulting view pointer. The variable must have been declared in a previous ENTITY statement.
status	The name of the variable that receives the status of this operation. The status values are:

Value	Description
0	The operation was successful.
-1	The view number given does not exist.

Example:

Program Statement	Explanation
`OBTAIN/STDVW,4,VP1,ST`	The view pointer for view 4 is obtained and placed in variable VP1, and the status of the operation is placed in variable ST.

Definition View Pointer

Statement format:

```
OBTAIN/DEFVW,entity,view pointer
```

Parameter	Description
DEFVW	The minor word for obtaining the definition view pointer of an entity.
entity	The name of the entity for which the view pointer is to be obtained.
view pointer	The name of the variable to contain the resulting view pointer. The variable must have been declared in a previous ENTITY statement.

Example:

Program Statements	Explanation
`PT1=POINT/100,100,10` `OBTAIN/DEFVW,PT1,DV`	Entity PT1 is defined. The view pointer for entity PT1 is obtained and placed in variable DV.

Current View Pointer

Statement format:

```
OBTAIN/CURVW,view pointer
```

Parameter	Description
CURVW	The minor word for obtaining the current view pointer.
view pointer	The name of the variable to contain the resulting view pointer. The variable must have been declared in a previous ENTITY statement.

Entity Name

Statement format:

```
OBTAIN/ENTNAM,entity,text variable,status
```

Parameter	Description
ENTNAM	The minor word for obtaining the ICEM DDN entity name of an entity from the data base. The data base entity name must have been defined interactively or by the ASSIGN statement, or else status will return a value of -1.
entity	The local, GPL name of the entity for which the ICEM DDN entity name is to be obtained.
text variable	The name of the text variable to receive the ICEM DDN entity name (must be 10 characters in length).
status	The name of the variable that receives the status of this operation. The status values are:

Value	Description
0	The operation was successful.
-1	The entity name was not found.

Example:

Program Statements	Explanation
`CHAR/CHR1(10)`	Sets up a text variable for the entity name.
`ENTITY/PT1`	Defines entity PT1.
`REAL/S`	Defines the status variable.
`SELECT/'Indicates Point',` `PT1,S`	Selects the ICEM DDN entity and stores its pointer in PT1.
`OBTAIN/ENTNAM,PT1$` `,CHR1,S`	Obtains the entity name for PT1 and stores it in CHR1.

Entity Pointer

There are two forms of the OBTAIN/ENTPTR statement. The first form is:

Statement format:

```
OBTAIN/ENTPTR,{seq,number      },entity,status
              {text variable   }
              {   'text'       }
```

Parameter	Description
ENTPTR	The minor word for obtaining the entity pointer of an entity by way of its name defined in the data base or its sequence number.
seq,number	The sequence number of the entity for which the program is searching.
text variable	The text variable that contains the data base name of the entity whose pointer is being obtained. This text variable must be declared as 10 characters in length.
'text'	The data base name, enclosed in single quotes, of the entity whose pointer is being obtained.
entity	The name of the entity that receives the pointer.
status	The name of the variable that receives the status of this operation. The status values are:

Value	Description
0	The operation was successful.
-1	The entity name was not found.

Example:

Program Statements	Explanation
`ENTITY/PT1`	Defines the entity.
`REAL/S`	Defines the status variable.
`OBTAIN/ENTPTR,'POINT1'$` `,PT1,S`	Obtains the entity pointer for the ICEM DDN entity with the assigned name 'POINT1' and gives it to entity PT1.

The second form of the OBTAIN/ENTPTR statement is:

Statement format:

```
OBTAIN/ENTPTR,[selection criteria,]NUMBER,number,array,var
```

Parameter	Description
ENTPTR	The minor word for obtaining the entity pointer of selected entities.
selection criteria	The criteria by which entities are selected. They can be any one or more of the following keywords.
	PENNUM — Selects by pen number.
	LEVNUM — Selects by level number.
	COLNUM — Selects by color number.
	FONNUM — Selects by font number.
	ENTTYP — Selects by entity type.
	Each criteria can be entered with up to ten numbers. For example:
	PENNUM 1,2,3,4,5,6,7
	COLNUM 3,9,16
NUMBER	The minor word indicating the total number of entities to be selected.
number	The total number of entities to be selected.
array	The name of the array into which the entity pointers are stored.
var	The name of a variable containing the actual number of entity pointers retrieved.

Example:

The following statement accumulates all entities of types 1, 2, or 3 (points, lines, and arcs) provided they use pen Ø.

```
OBTAIN/ENTPTR,ENTTYP,1,2,3,PENNUM,0,NUMBER,20,E20,ST
```

Parameters of an Entity

Statement format:

```
OBTAIN/PARAM,entity,array
```

Parameter	Description
PARAM	The minor word for obtaining the parameters of an entity.
entity	The name of the entity for which the parameters are to be obtained.
array	The name of the array in which the data is to be stored. The size of the array (the number of words required) is dependent upon the entity type selected. The number of elements for each entity is:

Entity Type	Number of Elements	Explanation
Curves	2	Curves require two elements per entity to store the start and end parameters of each curve.
Surfaces	4	Surfaces require four elements per entity to store the UMIN, VMIN, UMAX, and VMAX parameters.

Example:

Program Statements	Explanation
`REAL/ARR(2)`	Defines the array for the parameters.
`LN=LINE/10,10,20,20`	Creates the line.
`OBTAIN/PARAM,LN,ARR(1)`	Obtains the parameters of LN and stores them in array ARR(1).

Latest Error Number

Statement format:

```
OBTAIN/ERRST,error status [,line number]
```

Parameter	Description
ERRST	The minor word for obtaining the latest error number that occurred during the current GPL run. This error check works only if the STATUS modal has been previously declared as on.
error status	The name of the variable to receive the error number.
line number	The name of the variable to receive the line number in which the error occurred.

Entity Type and Form

Statement format:

```
OBTAIN/ENTTYP,entity,type number[,form number]
```

Parameter	Description
ENTTYP	The minor word for obtaining the entity type and form number of an entity.
entity	The name of the entity for which the type and form number are to be obtained.
type number	The name of the variable to receive the type number of the entity.
form number	The name of the variable to receive the form number of the entity.

Part Name and Sheet Number

Statement format:

```
OBTAIN/PARTNA,text variable,sheet number
```

Parameter	Description
PARTNA	The minor word for obtaining the part name and sheet number of the current part.
text variable	The name of the text variable to receive the part name of the current part. The text variable must be sized to receive at least 70 characters.
sheet number	The name of the real variable to receive the sheet number of the current part.

Members of a Group

Statement format:

```
OBTAIN/MEMBER,entity,number,entity array,actual number
```

Parameter	Description
MEMBER	The minor word for obtaining the members of a group.
entity	The name of the entity (group or composite curve) for which the members of the group are to be obtained.
number	The size (dimension) of the entity array.
entity array	The name of the entity array to receive the members of the group.
actual number	The number of members actually in the group.

Character String

Statement format:

```
OBTAIN/CHARST,entity,text variable,status
```

Parameter	Description
CHARST	The minor word for obtaining the character string associated with a dimensioning entity (LDIMEN, CDIMEN, DDIMEN, RDIMEN, LABEL, and NOTE).
entity	The name of the entity for which the character string is to be obtained.
text variable	The name of the text variable to receive the character string.
status	The name of the variable to receive the number of characters in the dimensioning entity.

Sequence Number

Statement format:

```
OBTAIN/SEQNUM,entity,variable
```

Parameter	Description
SEQNUM	The minor word for obtaining the sequence number of an entity.
entity	The name of the entity for which the sequence number is to be obtained.
variable	The name of the variable to receive the sequence number of the entity.

Attribute and Subattributes

Statement format:

```
OBTAIN/ATNAME,entity,rank1,variable1,text variable1

        [                 (NAME,text variable2[,v1]          )]
SUBAT,rank2,[             (NUMBER,array[,v2]                 )]
        [                 (BOTH,text variable2,array[,v1,v2] )]
```

Parameter	Description
ATNAME	The minor word for obtaining the attributes of an entity.
entity	The name of the entity for which the attributes are to be obtained.
rank1	The rank order of the attribute to be obtained.
variable1	The name of the variable to receive the number of attributes.
text variable1	The name of the text variable to receive the name of the attribute.
SUBAT	The minor word for obtaining the subattributes of an entity.
rank2	Either: • The rank order of the subattribute to be obtained, if NAME or BOTH are requested, or • The number of values to be obtained in the array, if NUMBER is requested.
NAME	The minor word for obtaining the name of a subattribute.
NUMBER	The minor word for obtaining the value of a subattribute.
BOTH	The minor word for obtaining both the name and number of a subattribute.
text variable2	The name of the text variable to receive the subattribute name.
array	The name of the array variable to receive the subattribute value.
v1	The name of the variable to receive the number of subattribute names.
v2	The name of the variable to receive the number of subattribute numbers.

Pen, Level, Color, and Font Numbers

Statement Format:

```
        ⎧PENNUM⎫
OBTAIN/ ⎪LEVNUM⎪ ,entity,number
        ⎪COLNUM⎪
        ⎩FONNUM⎭
```

Parameter	Description
PENNUM LEVNUM COLNUM FONNUM	The minor word indicating a pen, level, color, or font number.
entity	The name of the entity for which the attributes are obtained.
number	The number of the pen, level, color, or font.

Pen (0-15) and level (0-1023) numbers default to 0 or are assigned by the user. Color and font have preassigned default numbers as follows:

Font Number	Font
0	Solid
1	Dashed
2	Phantom
3	Centerline

Color Number	Color
0	White
1	Red
2	Green
3	Blue-cyan
4	Magenta
5	Yellow
6	Orange
7	Light orange
8	Light green
9	Dark cyan
10	Green-blue
11	Dark red
12	Light purple
13	Yellow-green
14	Dark pink

SETCHL

The SETCHL statement changes the available length of a character string within the limits of the maximum declared length. Refer to Defined Symbols in chapter 2 for a discussion of maximum and current length.

Statement format:

```
SETCHL/textvar,value
```

Parameter	Description
textvar	The name of the character string whose available length is to be changed.
value	The new available length of the character string.

TRIME

The TRIME statement trims the entity at the intersection of another entity.

Statement format:

```
TRIME/curve,intersect,xts,yts[,zts]
```

Parameter	Description
curve	The name of the entity to be trimmed.
intersect	The name of the intersecting entity.
xts, yts, zts	The transform coordinates that indicate which end is to be trimmed.

Variable Declaration, RTL I/O, and File Statements 7

Variable Declaration, RTL I/O, and File Statements 7

Variable declaration, RTL I/O, and file statements perform different management functions. Several of these definitions are equivalent to functions available under menus 4 FILE and 5 SPECIAL FUNCTIONS.

CHAR

The CHAR statement declares a text variable or array. The name of a text variable can be referenced in any statement that allows text variables. Character arrays may have up to two dimensions besides the length declaration.

Statement format:

```
CHAR/name(nchar[,size][,size]),...
```

Parameter	Description
name	The name or names of the text variables.
nchar	The maximum number of characters required for the text variable (expressed as an integer).
size	The number of elements in the array's dimension. The number of characters and the dimension sizes must be enclosed in parentheses. A two-dimensional array can be specified.

Examples:

Program Statement	Explanation
`CHAR/CHR1(10),CHR2(60)`	Text variable CHR1 is set for a maximum of 10 characters. Text variable CHR2 is set for a maximum of 60 characters.
`CHAR/C1(15,5)`	Text variable C1 is set for a maximum of 15 characters in each of its 5 elements.

COMMON

The COMMON statement defines a common block of data that can be used by the GPL programs. This must be declared after the MAIN or PROC statements and before the first executable statement. The common block can only be initialized using the DATA statement in a MAIN program.

The common block is defined by the modal words COMMON and ENDCOM. All declarations between these words have their addresses in the common block.

In order to use the data declared in common procedure, the same declarations from COMMON to ENDCOM (except DATA statements) must appear at the start of both the main program and the procedure. For larger programs, common decks (blocks) can be created and maintained by using the NOS utility MODIFY. This is a recommended method for large GPL programs to ensure identical specification of data between various GPL programs and procedures.

Statement format:

```
COMMON
    declaration 1
    declaration 2
        :
    declaration n
ENDCOM
```

Parameter	Description
COMMON	Indicates the start of a common block of declarations.
ENDCOM	Indicates the end of the common block.

Example:

Program Statement	Explanation

`COMMON`
`  REAL/VAL1,VAL2,STATS(10)`
`  ENTITY/PT5(10)`
`ENDCOM`
	The real variables VAL1 and VAL2, real array STATS(10), and entity array PT5(10) are assigned addresses in a common block.

CONST

The CONST statement assigns numerical values to symbolic names. The symbolic name can be used as a real or integer constant. (Note that GPL stores integers internally as floating point numbers.) This is useful when declaring arrays and specifying the limits of a FOR loop.

Statement format:

```
CONST/name1=value1[,name2=value2,...]
```

Parameter	Description
name	The symbolic name for a constant.
value	The value that is assigned to the symbolic name. This can be a simple expression consisting of constants (symbolic or real). The expression must not contain parentheses and must be developable from left to right (that is, 8**2*y/4+2-6 is correcxt, but x+y/2 is incorrect). Only exponentiation of integers below 524288 (2**19) is allowed.

Example:

Program Statement	Explanation
`CONST/ROWS=10,COLUMS=15`	The value 10 is assigned to ROWS, and the value 15 is assigned to COLUMS.

DATA

The DATA statement initializes real, integer, or character variables or arrays. The DATA statement comes after all declarative statements and before any executable code. If more than one DATA statement is used to declare the elements of the same array, the indices should be entered in ascending order. The replication factor duplicates the same value for the number of elements specified. The replication factor can only be used with arrays.

Statement format:

```
DATA/name[(index)][,repl*]value1[,value2,...]
```

Parameter	Description
name	The name of the real, integer, or character variable that is to be initialized.
(index)	The ordinal number of the array element. The index must be enclosed in parentheses.
repl	The number of replications.
*	An operator indicating that the next value is duplicated for the number of elements specified.
value1,value2,...	The values to be assigned to the variable or array. The first value is assigned to the first element of the array; the second value is assigned to the second element; and so forth.

Example:

Program Statement	Explanation
`DATA/VAL,4.321`	The variable VAL is initialized to 4.321.
`DATA/VALS(2),4*1.00,3.14`	The second, third, fourth, and fifth elements of variable VALS are initialized to 1.00. The sixth element is initialized to 3.14.

NOTE

A maximum of 640 data statements are allowed with a total of 3000 individual data initializations. Only one array or simple variable may be initialized per DATA statement.

ENTITY

The ENTITY statement reserves storage for local, simple and subscripted entity names created during the GPL run. Once the entity is defined, its name can be referenced in any subsequent statement within the same routine that allows entity names. The maximum size of one array is 65,534. Entity arrays can have up to three dimensions.

Statement format:

```
ENTITY/name[(size1[,size2[,size3]])],...
```

Parameter	Description
name	The name of the entity being sized.
(size1,size2,size3)	The sizes of the dimensions of the entity array (expressed in integers). At least one size must be given. The size parameters must be enclosed in parentheses.

Example:

Program Statements	Explanation
`ENTITY/PT(10)`	An entity array named PT is sized for 10 elements.
`PT(1)=POINT/100,100,0`	The first entity defined is assigned to the first position in the entity array.
`PT(2)=POINT/PT(1),$` `  RTHETA,20,45`	The second entity defined is assigned to the second position in the entity array.

FILE

The FILE statement stores the drawing in the ICEM DDN database (TAPE3) for later retrieval and/or modification. All variables saved in the RTL are filed with the drawing. The drawing is filed under the current part name and sheet number.

If the modal RECOVR is on, each FILE statement updates the recovery file (GPLREC). If the recovery file is saved, you can continue processing the GPL program after the last FILE statement (even though ICEM DDN has been terminated by a fatal error) by using menu 5.13.1 RECOVER LAST FILE.

Statement format:

```
FILE
```

Example:

Program Statement	Explanation
FILE	The drawing you are currently working on is filed to TAPE3.

GET

The GET statement retrieves simple variables from the RTL.

Statement format:

```
      ⎧variable   ⎫⎡ ⎧variable   ⎫⎤
GET/  ⎨var='name' ⎬⎢,⎨var='name' ⎬⎥[,...]
      ⎩var=textvar⎭⎣ ⎩var=textvar⎭⎦
```

Parameter	Description
variable	The name or names of variables stored in the RTL. Names must be separated by commas.
var='name'	A variable can be retrieved from the RTL and renamed using an equals sign followed by the RTL name enclosed in single quotes. The name is restricted to 6 characters.
var=textvar	A variable can be retrieved from the RTL and renamed using an equals sign followed by a text variable that contains the RTL name (6 characters maximum). The text variable is not enclosed in single quotes.

NOTE

At least one option (variable, var='name', or var=text) must be specified.

Examples:

Program Statement	Explanation
`GET/A,B,F,TAB,ABCD`	Variables A, B, F, TAB, and ABCD are retrieved from the RTL.
`GET/SCT='TTL'`	The variable TTL is retrieved from the RTL and renamed SCT.
`GET/NT1=PIN`	The variable whose name is contained in PIN is retrieved from the RTL and renamed NT1.

REAL

The REAL declaration statement reserves storage space for calculated real variables. The size of a real array is limited to 1,000,000 in the compiler, although this limit may exceed that imposed by the memory and disk space of the computer during execution. Real arrays can have up to three dimensions.

Statement format:

```
REAL/variable[(size1[,size2[,size3]])],...
```

Parameter	Description
variable	The name of the real variable that is declared.
(size1,size2,size3)	The sizes of the dimensions of the real variable array (expressed in integers). At least one size must be given. The size parameters must be enclosed in parentheses.

Example:

Program Statement	Explanation
`REAL/A,B(10),C,D(10)`	Real variables A and C are declared. Real variable arrays B and D are sized as 10 elements.

NOTE

As in FORTRAN, GPL allows the user to overflow an array during execution. This may lead to undesirable results in other sections of the program. Be careful to set your dimensions to the maximum subscript used.

SAVE

The SAVE statement saves simple variables in the RTL. Saved simple variables can be listed under 5.3.4 LIST RUN TIME LIBRARY VARIABLES (refer to the ICEM Design/Drafting Data Management reference manual).

Statement format:

```
SAVE/{variable  }[,{variable  }][,...]
     {var='name'}  {var='name'}
     {var=textvar} {var=textvar}
```

Parameter	Description
variable	The name or names of the variables to be saved in the RTL. The names must be separated by commas.
var='name'	A variable can be saved in the RTL and renamed using an equals sign followed by the new name enclosed in single quotes. The name is restricted to 6 characters.
var=textvar	A variable can be saved in the RTL and renamed using an equals sign followed by a text variable that contains the new name (6 characters maximum). The text variable is not enclosed in single quotes.

Example:

Program Statement	Explanation
`SAVE/A,B,F,TAR,ABCD`	Variables A, B, F, TAR, and ABCD are saved in the RTL.
`SAVE/SCT='TTL'`	The variable SCT is saved in the RTL as TTL.
`SAVE/NT1=PIN`	The variable NT1 is saved in the RTL as the name contained in the text variable PIN.

Program Management Statements 8

Program Management Statements 8

Program management statements perform various management functions.

CALL

The CALL statement executes the subroutine named in a PROC statement and passes variables from the main program to the subroutine. The variables passed must agree in order and number in both the main program and the subroutine although their names may be different.

Entire arrays can be passed by specifying the array name with a subscript of 1, but must also be declared in the subroutine.

The number of arguments passed is limited to 62.

Statement format:

```
CALL/name,variable,...
```

Parameter	Description
name	The 1- to 6-character name of the subroutine.
variable	The name of the variable that is passed to the subroutine.

Example:

Program Statement	Explanation
`CALL/TEST,A,B(10)`	The program branches to the subroutine named TEST and executes it. Variables A and B(10) are passed to the subroutine.

CONTIN

The CONTIN statement continues program execution at the next GPL statement. This can be used as a destination statement for branching. The CONTIN statement must be preceded by a label number of no more than five digits.

Statement format:

```
label number CONTIN
```

Parameter	Description
label number	The statement label number of the CONTIN statement.

Example:

Program Statement	Explanation

`IF (A.EQ.B) GOTO 10`
`A = A+1`
`  ⋮`
`10 CONTIN`
	Test if A equals B. If it does, jump to statement label 10. Execute the following statements only if the condition was false.

DATE

The DATE statement gets the current date from the operating system and stores it in a text variable. This text variable must be dimensioned by a CHAR statement prior to using the DATE statement. The data is stored in the form yy-mm-dd (yy is year, mm is month, and dd is day).

Statement format:

```
DATE/variable
```

Parameter	Description
variable	The name of the variable that receives the date from the operating system.

Example:

Program Statement	Explanation
`CHAR/DT(10)`	The variable that receives the date is dimensioned to 10 characters.
`DATE/DT`	The current date is stored in variable DT.

FINI

The FINI statement indicates the end of a GPL program or subroutine. If it is at the end of a subroutine, this statement returns control to the calling program. If it is at the end of a main program, this statement returns control to menu 5 SPECIAL FUNCTIONS.

Statement format:

```
FINI
```

Example:

Program Statement	Explanation
`FINI`	The program has finished executing.

MAIN

The MAIN statement is the first statement in a GPL program. The name of the program given in the MAIN statement can be called from ICEM DDN for execution.

Descriptive text can be passed with the program. The text appears in source listing headers and is attached to the object code in a system-dependent fashion. For NOS, the text takes the form of a PIDL table (7700 table).

In the internal representation, this user-defined information is preceded by the compilation date and time and the compiler release level. System Utilities may be used to list this information.

Statement format:

```
MAIN/name[,'text']
```

Parameter	Description
name	The 1- to 6-character name of the program.
'text'	Descriptive text (40 characters maximum) enclosed in quotes.

Example:

Program Statement	Explanation
`MAIN/GEAR1`	The first statement of a program named GEAR1.
`MAIN/PRACT1,'FIRST PROGRAM'`	The first statement of a program named PRACT1 with descriptive text FIRST PROGRAM.

MSTRNG

The MSTRNG statement executes an ICEM DDN menu string from within a GPL program. The menu string may either be defined explicitly within the GPL program or defined as a named menu string on the external menu string file.

If the menu string is specified in the GPL program, it can either be defined in a text variable or enclosed in single quotes on the MSTRNG statement line. The menu string syntax must be identical to tablet menu string syntax (see Tablet String Programming Format in the ICEM Design/Drafting Data Management reference manual). Specifically, menu choices must be separated by periods and the lowercase "t" used to enclose data or text. The string cannot exceed 127 characters in length. The menu string can also be in packed format as discussed under 7.13.1.4 STRING DISPLAY FORMAT.

If you specify the menu string from a local file, the menu string must first be defined in the local file declared by 1.16.4 CHANGE MSTRING FILE. The default name is MSTRING. The menu string file name can also be changed using the MSFILE statement.

The automatic return of an MSTRNG statement can be suppressed by using the NORTRN minor word. After using NORTRN, you must use 5.13.2 CONTINUE GPL PROGRAM to continue execution of the GPL program.

NOTE

The menu string to be executed will begin at the Special Functions menu. Therefore, you will usually want to begin each menu string with "f" to return to the main menu.

Alternate Screen Position Input (the I key) is available within your menu string.

Statement format:

```
        ┌'menu string'                  ┐
MSTRNG/ │menu string text variable      │[,NORTN]
        │NAME,'msname'                  │
        │NAME,msname text var           ┘
```

Parameter	Description
'menu string'	The explicit menu string enclosed in single quotes.
menu string text variable	The name of a text variable that contains the menu string.
NAME	The minor word for retrieving the named mstring file from the local file specified by the MSFILE modal.
'msname'	The name of a menu string (enclosed in single quotes) that resides on the local file specified by the MSFILE modal.

Parameter	Description
msname text var	The name of a text variable containing the name of a menu string that resides on the local file specified by the MSFILE modal.
NORTN	The minor word for suppressing automatic return from the execution of an MSTRNG statement to the next statement following the MSTRNG statement in the executing GPL program. If you use this minor word, you must use 5.13.2 CONTINUE GPL PROGRAM to resume execution at the next statement following the MSTRNG statement in the executing GPL program.

The following error messages can occur from incorrect use of the MSTRNG statement.

```
SYNTAX ERROR WITHIN MENU STRING

ERROR FOUND WHILE READING MSTRING FILE

MENU STRING NAME NOT ON FILE
```

Examples:

Program Statement	Explanation
`MSTRNG/'F.6.6.1.3.t]t.t]t'`	Lists to the screen all entities by type.
`MSTRNG/NAME,TP1`	Executes menu string TP1 residing on the current menu string file.
`MSTRNG/'F.6.6.1.3.t]t',NORTRN`	Lists entity types without returning to the next GPL statement.

PAUSE

The PAUSE statement interrupts the GPL program and returns you to the GPL menu. The program can be restarted using two different menus. Menu 5.13.2 CONTINUE GPL PROGRAM begins execution at the first statement after the PAUSE that caused the interrupt. Menu 5.13.3 RUN GPL PROGRAM restarts the program at its beginning.

Statement format:

```
PAUSE/['text']
```

Parameter	Description
'text'	The text to be displayed when the program returns to the GPL menu. It must be enclosed in single quotes (10 characters maximum).

Example:

Program Statement	Explanation
`PAUSE/'WAITING'`	The current GPL program is interrupted and waits until either 5.13.2 CONTINUE GPL PROGRAM or 5.13.3 RUN GPL PROGRAM is executed.

PROC

The PROC statement names the first statement of a subroutine, and can receive variables from the main program. The variables passed must agree in order and number in both the main program and the subroutine. Arrays can be passed by specifying the array name without the subscript, but must be declared in both the main program and the subroutine. Subscripts are not allowed in the argument list.

This statement is executed when a CALL statement with the name of the PROC statement is encountered in the main program. The name of the subroutine must be unique to the GPL library.

Descriptive text can be passed with the program. The text appears in source listing headers and is attached to the object code in a system-dependent fashion. For NOS, the text takes the form of a PIDL table (7700 table).

In the internal representation, this user-defined information is preceded by the compilation date and time and the compiler release level. System Utilities can be used to list this information.

Statement format:

```
PROC/name,variable,...[,'text']
```

Parameter	Description
name	The 1- to 6-character name of the subroutine.
variable	The name of a variable passed from the main program to the subroutine.
'text'	Descriptive text (40 characters maximum) enclosed in quotes.

Example:

Program Statement	Explanation
`PROC/TEST,A,B`	The first statement of a subroutine named TEST with the variables named A and B passed from the main program.
`PROC/COORDS,x,y,z$` `,'FIND CLOSEST POINT'`	The first statement of a subroutine named COORDS with variables x, y, and z passed from the main program. The descriptive text is FIND CLOSEST POINT.

REMARK

The REMARK statement adds descriptive comments to the part program. This information is not used, but is included for program readability. If more than one line is required for the comment, each line must start with a REMARK command. The REMARK statement must appear at the leftmost position of the program line.

Comments can also be added to any statement by placing them after $$. A statement that starts with $$ is equivalent to a REMARK command.

Statement format:

```
REMARK/...descriptive comment...
```

Example:

Program Statement	Explanation
REMARK/THE FOLLOWING REMARK/PROGRAM CREATES REMARK/A WHEEL BEARING	Descriptive comments preceding a program that creates the geometry for a wheel bearing.

RETURN

The RETURN statement can occur anywhere in a subroutine except as the first or last statement. It returns control to the statement immediately after the CALL statement that invoked the subroutine. (A FINI statement terminates the subroutine.)

Statement format:

```
RETURN
```

Example:

Program Statement	Explanation
RETURN	The subroutine ends and execution starts at the statement immediately after the CALL statement that invoked the subroutine.

STOP

The STOP statement terminates the GPL program. This command returns control to menu 5 SPECIAL FUNCTIONS.

Statement format:

```
STOP/['text']
```

Parameter	Description
'text'	The text to be displayed when the program returns to the GPL menu. It must be enclosed in single quotes (10 characters maximum).

NOTE

This statement should not be used under TELEX. The program must always have a FINI statement to end the program, even if you use a STOP statement to stop execution.

TIME

The TIME statement returns the current time as hh:mm:ss (hh is hours, mm is minutes, and ss is seconds). The text variable must be dimensioned by a CHAR statement prior to using the TIME statement.

Statement format:

```
TIME/variable
```

Parameter	Description
variable	The name of the variable to receive the current time.

Example:

Program Statement	Explanation
`CHAR/TM(10)`	The variable that receives the time is dimensioned to 10 characters.
`TIME/TM`	The current time is stored in variable TM.

Display Control Statements 9

Display Control Statements 9

Display control statements perform display control functions. Several of these definitions are equivalent to types and forms found in menu 8 DISPLAY CONTROL.

BLANKE

The BLANKE statement blanks specified entities.

Blank Specified Entities

Statement format:

```
BLANKE/entity[,...]
```

Parameter	Description
entity	The names of the entities to be blanked. This list can contain a maximum of 64 entities.

Example:

Program Statement	Explanation
`BLANKE/PT1,LN2,CIR3`	Entities PT1, LN2, and CIR3 are blanked.

Blank an Entity Array

Statement format:

```
BLANKE/NUMBER,count,entity array
```

Parameter	Description
NUMBER	A minor word indicating that a number of entities are to be blanked.
count	The number of entities to be blanked in the entity array. This number must not exceed 250.
entity array	The array of entities to be blanked.

Example:

Program Statement	Explanation
`BLANKE/NUMBER,3,E(1)`	The three entities E(1), E(2), and E(3) are blanked.

Blank All Except Specified Entities

Statement format:

```
BLANKE/ALL[,entity[,...]]
```

Parameter	Description
ALL	The minor word indicating that all entities are to be blanked except those in the optional parameter list.
entity	The names of the entities not to be blanked.

Example:

Program Statement	Explanation
`BLANKE/ALL,PT27,PT28,PT29`	All entities are blanked except PT27, PT28, and PT29.

Blank All of a Level, Color, or Pen

Statement format:

```
BLANKE[LEVNUM|COLNUM|PENNUM],number[,entity[,...]]
```

Parameter	Description
LEVNUM	The minor word indicating that all entities on the given level number are to be blanked.
number	The number of the level, color, or pen of which entities are to be blanked.
COLNUM	The minor word indicating that all entities of the given color number are to be blanked.
PENNUM	The minor word indicating that all entities of the given pen number are to be blanked.
entity	The names of the entities not to be blanked.

Example:

Program Statement	Explanation
`BLANKE/LEVL,3,PT50,PT51`	All entities on level 3 are blanked except PT50 and PT51.

CHANGE

The CHANGE statement changes the current work view, which must have been defined previously using the VIEW statement or obtained as a standard view using the OBTAIN statement. A maximum of 12 views can be used simultaneously.

Multiple Views

Statement format:

```
CHANGE/v1[,v2[,...]][,NORPNT]
```

Parameter	Description
v1, v2...	A listing of view pointers. The first view is defined as the current work view. A maximum of 12 views can be used simultaneously.
NORPNT	The minor word for not repainting the display when the work view is changed.

Two Views

Statement format:

```
CHANGE/(VERTCL),v1,v2[,NORPNT]
       (HORIZ )
```

Parameter	Description
VERTCL	The minor word for changing the top and bottom views.
HORIZ	The minor word for changing the right and left views.
v1	The pointers of the view defined as the current work view. This is the top view for VERTCL and the left view for HORIZ.
v2	The pointers of the second view. This is the bottom view for VERTCL and the right view for HORIZ.
NORPNT	The minor word for not repainting the display when the work view is changed.

Changing Current Work View

Statement format:

```
CHANGE/CURVW,v1,status
```

Parameter	Description
CURVW	The minor word for changing any displayed view as the current work view.
v1	The view pointer entity defining the desired current work view.
status	The name of a variable that receives the status of the CHANGE operation. The status values are:

Value	Description
0	The operation was successful.
-1	The view given is not on the screen.

MAP

The MAP statement maps a three-dimensional coordinate array from one view to another view.

Statement format:

```
MAP/vp1,array1,vp2,array2
```

Parameter	Description
vp1	The pointer of the view that sends the data.
array1	The name of the array that sends the data.
vp2	The pointer of the view that receives the data.
array2	The name of the array receives the data.

REPANT

The REPANT statement repaints the screen. Deleted entities disappear from the display if you are using a storage tube terminal. Deleted and blanked entities are not displayed by the REPANT statement.

Statement format:

```
REPANT
```

Example:

Program Statement	Explanation
REPANT	The screen display of the part drawing is repainted.

UNBLNK

The UNBLNK statement unblanks blanked entities.

Specified Entities

Statement format:

```
UNBLNK/entity[,...]
```

Parameter	Description
entity	The name of the entity to unblank. The total number of entities may not exceed 64.

Entities from an Entity Array

Statement format:

```
UNBLNK/NUMBER,count,entity array
```

Parameter	Description
NUMBER	The minor word for unblanking an entity from an array.
count	The number of entities to be unblanked from the array. The total number of entities cannot exceed 250.
entity array	The name of an entity array.

All Except Specified Entities

Statement format:

```
UNBLNK/ALL[,entity,...]
```

Parameter	Description
ALL	The minor word for unblanking all except selected entities.
entity	The name of the entity not to unblank. The total number of entities cannot exceed 63.

On a Level

Statement format:

```
UNBLNK/LEVL,level number[,entity,...]
```

Parameter	Description
LEVL	The minor word for unblanking an entity from a level.
level number	The number of the level from which to unblank.
entity	The name of the entity that is left blanked. The total number of entities cannot exceed 62.

VBORDS

The VBORDS statement turns on or off the display of view borders.

Statement format:

```
VBORDS/(ON )[,view...]
       (OFF)
```

Parameter	Description
ON	Turns on view border display.
OFF	Turns off view border display.
view...	An optional list of views to receive borders. If omitted, all standard view borders are turned on.

Example:

Program Statement	Explanation
VBORDS/ON	All subsequent view borders are displayed.

VIEW

The VIEW statement creates a new view. The resulting view pointer can be used in the CHANGE statement to change the current work view and should be declared by an ENTITY statement. There are three definition forms: auxiliary view by rotation angle, auxiliary view parallel to a plane defined by three points, and view by matrix.

Auxiliary View by Rotation Angle

Statement format:

```
VIEW/AUX[,XTROT,xt angle][,YTROT,yt angle][,ZTROT,zt angle][,ORIGIN,xt,yt,zt]
```

Parameter	Description
AUX	The minor word for creating a new auxiliary view.
XTROT	The minor word for indicating rotation in the XT plane.
xt angle	The xt angle in degrees.
YTROT	The minor word for indicating rotation in the YT plane.
yt angle	The yt angle in degrees.
ZTROT	The minor word for indicating rotation in the ZT plane.
zt angle	The zt angle in degrees.
ORIGIN	The minor word for indicating the origin of the view.
xt,yt,zt	The xt-, yt-, and zt-coordinates of the view origin.

NOTE

At least one minor word must be present after AUX.

Auxiliary View Parallel to a Plane Defined by Three Points

Statement format:

```
VIEW/AUX,point1,point2,point3[,ORIGIN,xt,yt,zt]
```

Parameter	Description
AUX	The minor word for creating a new auxiliary view.
point1, point2	Point1 and point2 define the edge of the plane parallel to the x-axis.
point2, point3	Point2 and point3 define the edge of the plane parallel to the y-axis.
ORIGIN	The minor word for indicating the view origin.
xt,yt,zt	The xt-, yt-, and zt-coordinates of the view origin.

View by Matrix

Statement format:

```
VIEW/MATRIX,a11,a21,a31,...,a33,a14,a24,a34
```

Parameter	Description
MATRIX	The minor word for creating a view using a matrix.
a11 through a34	The values of the elements of a rotation matrix. The rotation matrix is in the form: a11 a21 a31 a12 a22 a32 a13 a23 a33 a14 a24 a34 The matrix must be orthonormal and $\lvert \det \rvert = \pm 1$.

VNAMES

The VNAMES statement turns on or off the view names display.

Statement format:

```
VNAMES/{ON }[,view...]
       {OFF}
```

Parameter	Description
ON	Turns on view name display.
OFF	Turns off view name display.
view...	An optional list of views to receive names. If omitted, all standard views are named.

Example:

Program Statement	Explanation
VNAMES/ON	All subsequent view names are displayed.

VVECS

The VVECS statement turns on or off the display of coordinate view vectors.

Statement format:

```
VVECS/{ON }[,view...]
      {OFF}
```

Parameter	Description
ON	Turns on coordinate view vector display.
OFF	Turns off coordinate view vector display.
view...	An optional list of views to receive vectors. If omitted, all standard view vectors are turned on.

Example:

Program Statement	Explanation
VVECS/ON	All subsequent coordinate view vectors are displayed.

WAIT

The WAIT statement pauses the program so that the display of the drawing has time to catch up with the program.

Statement format:

```
WAIT
```

ZOOM

The ZOOM statement zooms in and out of the part drawing.

Auto MAXMIN

Statement format:

```
ZOOM/MAXMIN
```

Parameter	Description
MAXMIN	The minor word for indicating automatic maximums and minimums. The whole part drawing is automatically displayed on the screen.

Diagonal Points

Statement format:

```
ZOOM/COORD,{point}  ,{point}
           {xt,yt}  ,{xt,yt}
```

Parameter	Description
COORD	The minor word for zooming to the region defined by two corner points.
point	The name of a corner point.
xt,yt	The xt- and yt-coordinates of a corner point.

Scale Factor with Origin or Center

Statement format:

```
ZOOM/SCALE,scale[,(ORIGIN),(point)]
               (CENTER) (xt,yt)
```

Parameter	Description
SCALE	The minor word for zooming to a specified scale factor.
scale	The scale factor.
ORIGIN	The minor word for using the origin of the drawing as a base for zooming.
CENTER	The minor word for using the center of the drawing as a base for zooming.
point	The name of the origin or center point.
xt,yt	The xt- and yt-coordinates of the origin or center point.

POINT Statements 10

POINT Statements 10

POINT

POINT statements create points in the part drawing. A point is a unique spatial position defined by coordinates, lines, angles, arcs, and intersections of two curves. Point definitions are equivalent to forms found in menu 9 POINT.

NOTE

The DEFINE statement can be used as an alternate way of producing a large number of points.

Coordinates

Statement format:

```
POINT/xcoord,ycoord[,zcoord]
```

Parameter	Description
xcoord	The x-coordinate of the point being created.
ycoord	The y-coordinate of the point being created.
zcoord	The z-coordinate of the point being created. If zcoord is omitted, it is equal to the current depth value.

Examples:

Program Statement	Explanation
`PT1=POINT/2,1,1.3`	Point PT1 is created at coordinates x=2, y=1, and z=1.3.
`PT2=POINT/3.1,1`	Point PT2 is created at coordinates x=3.1, y=1, and z equals the current depth value.
`PT3=POINT/X,Y,Z`	Point PT3 is created at coordinates defined by variables X, Y, and Z.

Polar Coordinates

Statement format:

```
POINT/point,RTHETA,radius,angle
```

Parameter	Description
point	The name of the reference point.
RTHETA	The minor word for defining a point using polar coordinates.
radius	The radius in units of measure from the reference point to where you want to create the point.
angle	The positive angle from the horizontal axis (xt-axis) of the reference point to where you want to create the point.

Example:

Program Statement	Explanation
`P345=POINT/PT2,RTHETA,1.5,45`	Point P345 is created using polar coordinates at a radius of 1.5 units of measure and at an angle of 45° from point PT2.

Delta Coordinates

Statement format:

```
POINT/point,DELTA,deltax,deltay,deltaz
```

Parameter	Description
point	The name of the reference point.
DELTA	The minor word for defining a point using delta coordinates.
deltax	The delta x-coordinate of the point being created.
deltay	The delta y-coordinate of the point being created.
deltaz	The delta z-coordinate of the point being created.

Example:

Program Statement	Explanation
`VDF=POINT/PT1,DELTA,2,1,0`	Point VDF is created at delta coordinates x=2, y=1, and z=0 from point PT1.

Vectored Point

Statement format:

```
POINT/point,{XSMALL}
            {XLARGE},distance,line
            {YSMALL}
            {YLARGE}
```

Parameter	Description
point	The name of the reference point.
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate in which direction the point is created. Refer to figure 10-1.
distance	The distance in units of measure from the reference point along an imaginary line parallel to the reference line.
line	The name of the reference line.

Example:

Program Statement	Explanation
`PT5=POINT/PT1,XLARGE,2,LIN1`	Point PT5 is created in the x positive direction from point PT1 at a distance of 2 units of measure parallel to line LIN1.

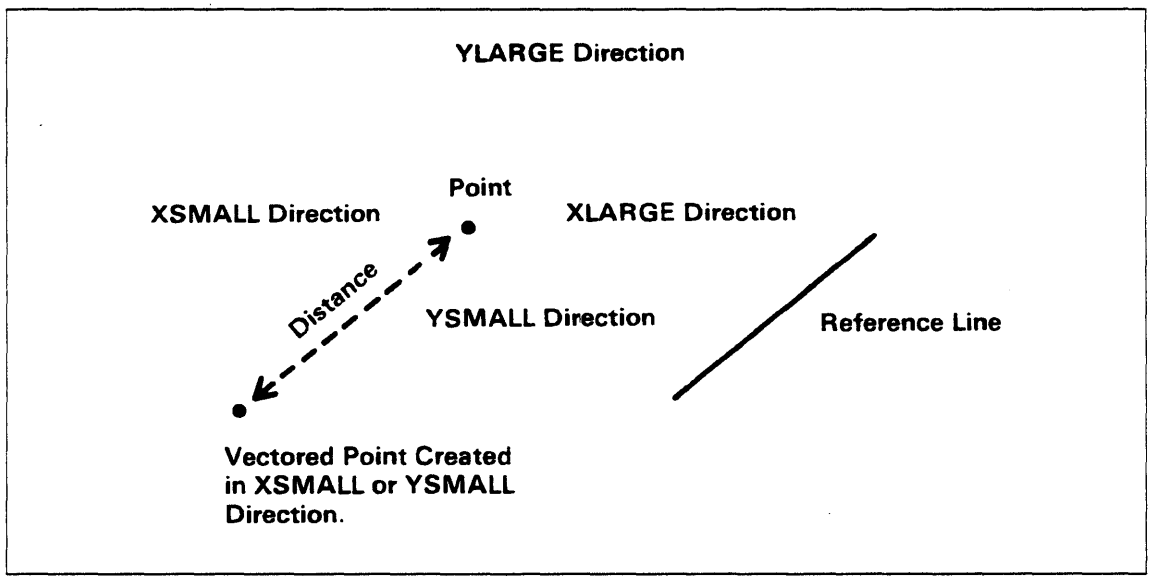


Figure 10-1. Vectored Point

Center of a Circle

Statement format:

```
POINT/CENTER,circle
```

Parameter	Description
CENTER	The minor word for defining a point as the center of a circle.
circle	The name of the circle.

Example:

Program Statement	Explanation
`P4578=POINT/CENTER,C789`	Point P4578 is created in the center of circle C789.

On a Circle

Statement format:

```
POINT/circle,ATANGL,angle
```

Parameter	Description
circle	The name of the reference circle.
ATANGL	The minor word for indicating the angle.
angle	The positive angle from the horizontal axis (xt-axis) along the reference circle at which you want to create the point.

Example:

Program Statement	Explanation
`P29=POINT/CIR32,ATANGL,35.2`	Point P29 is created at a positive angle of 35.2° from the horizontal axis along the circumference of circle CIR32.

Curve Endpoint

Statement format:

```
POINT/END, XSMALL
           XLARGE
           YSMALL
           YLARGE , curve
```

Parameter	Description
END	The minor word for defining the point on the end of a curve.
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate in which direction the point is created. Refer to figure 10-2.
curve	The name of the reference curve.

Example:

Program Statement	Explanation
`PT104=POINT/END,YLARGE,C99`	Point PT104 is created at the end of circle C99 in the y positive direction.

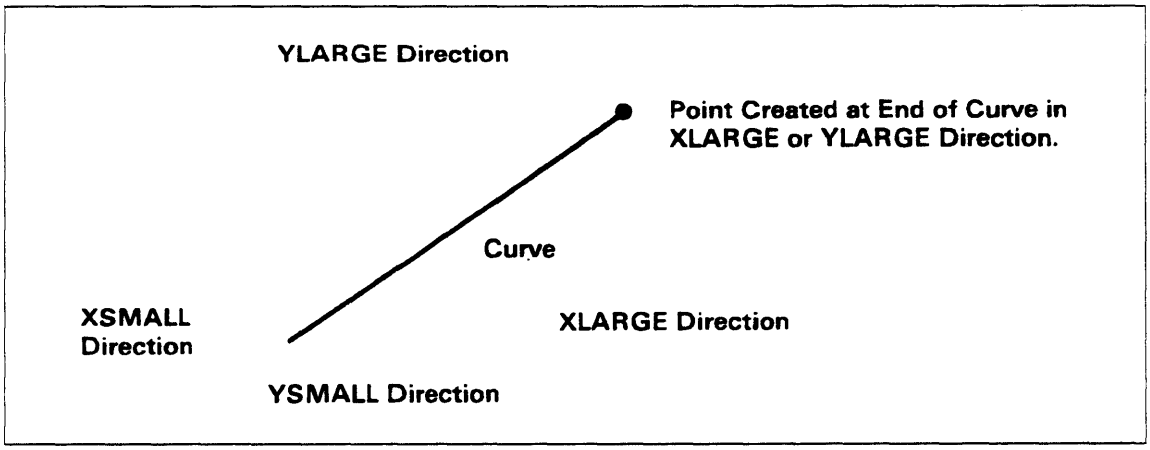


Figure 10-2. Point on the End of a Curve

Intersection of Two Curves

Statement format:

```
                  [ [ xt,yt[,zt] ] ]
POINT/INTOF       [ [ XSMALL     ] ]
                [ , [ XLARGE     ] ] ,curve1,curve2
                  [ [ YSMALL     ] ]
                  [ [ YLARGE     ] ]
```

Parameter	Description
INTOF	The minor word for defining the point at the true intersection of two reference curves (as opposed to the current depth intersection).
xt,yt,zt	The coordinates of a location at or near the intersection of the two reference curves. It is recommended that this coordinate method be used since the positional form may give an incorrect result if either the first entity is off or nearly off the screen or if PAINT is off.
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The minor words XSMALL, XLARGE, YSMALL, and YLARGE indicate which of several possible intersections is specified. Refer to figure 10-3. A direction indicator is not required if only one intersection is possible, as in the case of the intersection of two lines.
curve1	The name of the first reference curve.
curve2	The name of the second reference curve. The second curve indicates which intersection point, if there is more than one intersection.

Example:

Program Statement	Explanation
`P24=POINT/INTOF,L29,L30`	Point P24 is created at the intersection of line L29 and line L30.
`PT10B=POINT/INTOF,XLARGE$` `L42,C40B`	Point PT10B is created at the intersection of line L42 and circle C40B in the x positive direction.

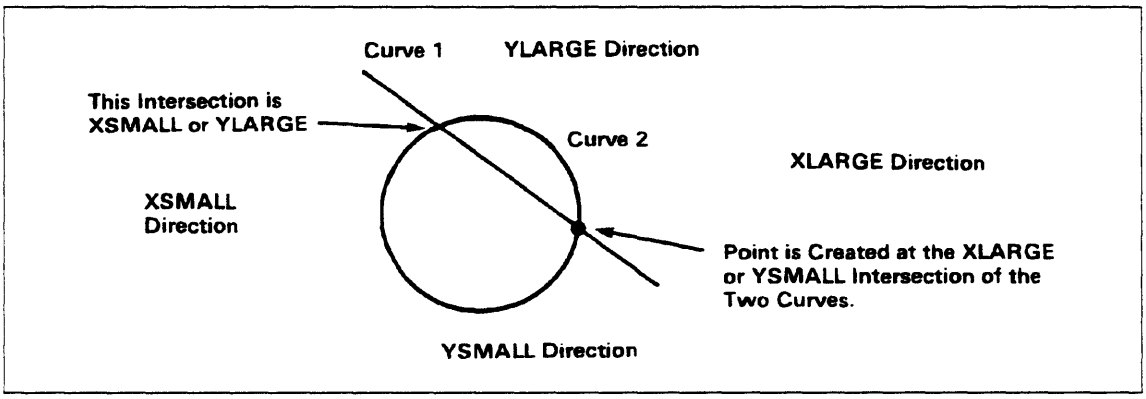


Figure 10-3. Point at the Intersection of Two Curves

On a Line at an X, Y, or Z Value

Statement format:

```
                 ⎧XVALUE,xcoord⎫
POINT/line,      ⎨YVALUE,ycoord⎬
                 ⎩ZVALUE,zcoord⎭
```

Parameter	Description
line	The name of the reference line.
XVALUE	The minor word for creating a point along the x-axis of the line.
YVALUE	The minor word for creating a point along the y-axis of the line.
ZVALUE	The minor word for creating a point along the z-axis of the line.
xcoord	The x-coordinate of the point along the x-axis.
ycoord	The y-coordinate of the point along the y-axis.
zcoord	The z-coordinate of the point along the z-axis.

Example:

Program Statement	Explanation
`POINT/LN1,XVALUE,5.0`	A point is created along the x-axis of line LN1 at a distance of 5 units.

Projection of a Point on a Curve

Statement format:

```
POINT/NOPROJ,point,curve
```

Parameter	Description
NOPROJ	The minor word for projecting a point on a curve.
point	The name of the point that is to be projected.
curve	The name of the curve on which the point is to be projected. Lines, arcs, conics, splines, point sets, three-dimensional splines, and machining curves are legal entities for this operation.

Example:

Program Statement	Explanation
`POINT/NOPROJ,X22,C34`	Point X22 is projected on curve C34.

At a Bearing and Distance

Statement format:

```
POINT/BEARDS,point,{NORTH},{EAST},angle,distance
                   {SOUTH} {WEST}
```

Parameter	Description
BEARDS	The minor word for creating a point at a bearing and distance from a reference point.
point	The name of the point from which the bearing and distance are to be taken.
NORTH	The minor word for a north bearing.
SOUTH	The minor word for a south bearing.
EAST	The minor word for an east bearing.
WEST	The minor word for a west bearing.
angle	The angle from the reference point to the new point.
distance	The distance from the reference point to the new point.

Example:

Program Statement	Explanation
`POINT/BEARDS,PT45,$` `  NORTH,EAST,45,1.0`	A point is created at a bearing of 45° east of north of the reference point and a distance of 1 unit of measure.

On a Curve at a Parameter

Statement format:

```
POINT/CURVE,curve,parameter
```

Parameter	Description
CURVE	The minor word for creating a point along a curve at a parameter.
curve	The name of the reference curve.
parameter	The parameter along the curve at which to create the point. The parameter must lie between the starting and ending parameters of the curve.

Example:

Program Statement	Explanation
`POINT/CURVE,CRV1,0.35`	A point is created along curve CRV1 at a parameter of 0.35 from the start end of the curve.

Surface Normal

Statement format:

```
POINT/NORMAL,point,surface
```

Parameter	Description
NORMAL	The minor word for creating a surface normal point.
point	The name of the base point.
surface	The name of the surface.

Example:

Program Statement	Explanation
`POINT/NORMAL,Q3,SRF1`	A surface normal point is created on surface SRF1 from the base point Q3.

Pierce

Statement format:

```
POINT/PIERCE,point,surface,vector
```

Parameter	Description
PIERCE	The minor word for creating a pierce point.
point	The name of the base point.
surface	The name of a drive surface.
vector	The name of a vector. The surface is pierced by a line parallel to the vector. The pierce point is created as close as possible to the base point.

Example:

Program Statement	Explanation
`POINT/PIERCE,Q5,SF2,V3`	A pierce point is created at the intersection of surface SF2 and a line that is parallel to the vector V3 and passes through the base point Q5.

Spherical Point

Statement format:

```
POINT/SPHRIC,radius,anglez,anglex
```

Parameter	Description
SPHRIC	The minor word for creating a spherical point.
radius	The radius from the origin (0,0,0) to the point in units of measure.
anglez	The angle of the point from the z-axis.
anglex	The angle of the point from the x-axis.

Example:

Program Statement	Explanation
`POINT/SPHRIC,5,45,30`	A spherical point is created at a radius of 5 units of measure from the origin, 45° from the z-axis, and 30° from the x-axis.

LINE Statements 11

LINE Statements 11

LINE statements create lines in a part drawing. Line definitions are equivalent to forms found in menu 10 LINE.

CHAMFR (Bevel)

Statement format:

```
CHAMFR/ XSMALL  ,line1, XSMALL  ,line2,distance,angle
        XLARGE          XLARGE
        YSMALL          YSMALL
        YLARGE          YLARGE

[,TRIM[, XSMALL  ,line1][, XSMALL  ,line2]]
       XLARGE            XLARGE
       YSMALL            YSMALL
       YLARGE            YLARGE
```

Parameter	Description
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate in which direction the line is to be created.
	The ends of the two lines defined by the large and small indicators define in which quadrant the chamfer is constructed. The four quadrants are defined by the intersection of the two lines.
line1	Either the name of the first line to be used for the chamfer or the name of the line to be trimmed.
line2	Either the name of the second line to be used for the chamfer or the name of the line to be trimmed.
distance	The distance of the chamfer in units of measure from the intersection along line1.
angle	The angle of line1 to the chamfer.
TRIM	The minor word for indicating lines are to be trimmed.

Examples:

Program Statement	Explanation

`LN3=CHAMFR/XSMALL,LN1,$`
`YSMALL,LN2,.5,45,$`
`TRIM,XLARGE,LN1,$`
`YLARGE,LN2`
	Line LN3 is created as a chamfer between the x negative end of line LN1 and the y negative end of line LN2. The start of the chamfer is distant from the intersection by 0.5 units of measure. The angle of chamfer is 45°. The x positive end of line LN1 and the y positive end of line LN2 are trimmed.

`LN4=CHAMFR/XSMALL,LN1,$`
`YSMALL,LN2,DS,ANGL,$`
`TRIM,XLARGE,LN1,$`
`YLARGE,LN2`
	This is the same as line 3 except that the distance and the angle of chamfer are the variables DS for the distance and ANGL for the angle.

Figure 11-1 shows a chamfer. The chamfer is created in the quadrant defined by the XSMALL end of line 1 and the YSMALL end of line 2. The four quadrants are defined by the intersection of the two lines.

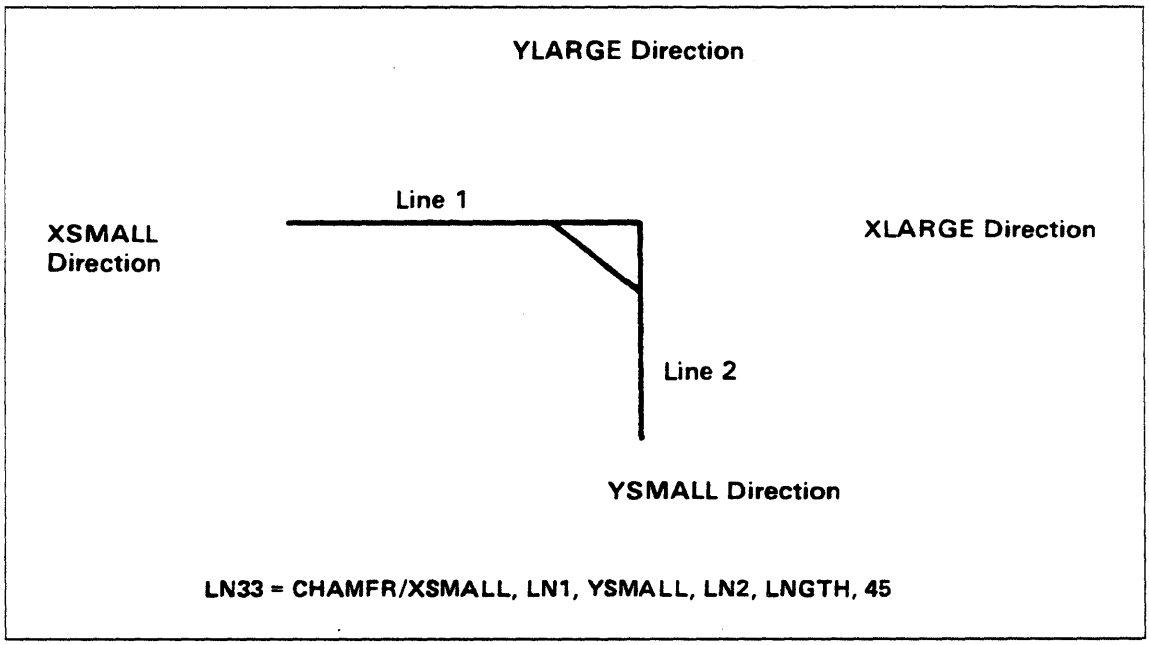


Figure 11-1. Chamfer

LINE

LINE statements create lines in the part drawing. A line is a curve defining the shortest distance between two points.

NOTE

The DEFINE statement can be used as an alternate way of producing a large number of lines.

Coordinates

Statement format:

```
LINE/x1,y1[,z1],x2,y2[,z2]
```

Parameter	Description
x1,y1,z1	The x-, y-, and z-coordinates of one end of the line being created.
x2,y2,z2	The x-, y-, and z-coordinates of the second end of the line being created.

If z1 and z2 are omitted, they are equal to the current depth value.

Example:

Program Statement	Explanation
`LIN23=LINE/2,1,3,4,1.5,2`	Line LIN23 is created between coordinates x=2, y=1, and z=3, and coordinates x=4, y=1.5, and z=2.
`LIN24=LINE/X1,Y1,Z1,X2,Y2,Z2`	Line LIN24 is created between coordinates defined by variables X1, Y1, and Z1, and coordinates defined by variables X2, Y2, and Z2.

Two Points

Statement format:

```
LINE/point1,point2
```

Parameter	Description
point1	The name of the point to be used as one end of the line being created.
point2	The name of the point to be used as the other end of the line being created.

Example:

Program Statement	Explanation
`LIN32=LINE/PT1,PT2`	Line LIN32 is created between point PT1 and point PT2.

Tangent to Two Curves

Statement format:

```
LINE/TANTO, { XSMALL } , curve1, { XSMALL } , curve2
            { XLARGE }           { XLARGE }
            { YSMALL }           { YSMALL }
            { YLARGE }           { YLARGE }
```

Parameter	Description
TANTO	The minor word indicating that a line is to be created tangentially from the specified side of curve1 to the specified side of curve2.
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate in which direction the line is to be created. For example, in figure 11-2, a line is drawn from the YLARGE side of curve1 to the YSMALL side of curve2.
curve1	The name of the first curve.
curve2	The name of the second curve.

Example:

Program Statement	Explanation
`LIN10=LINE/TANTO,XSMALL,$` `  CUR1,YSMALL,CUR2`	Line LIN10 is created as a tangent from the XSMALL side of curve CUR1 to the YSMALL side of curve CUR2.
`LIN12=LINE/TANTO,XLARGE,$` `  CUR1,YLARGE,CUR2`	Line LIN12 is created as a tangent from the XLARGE side of curve CUR1 to the YLARGE side of curve CUR2.

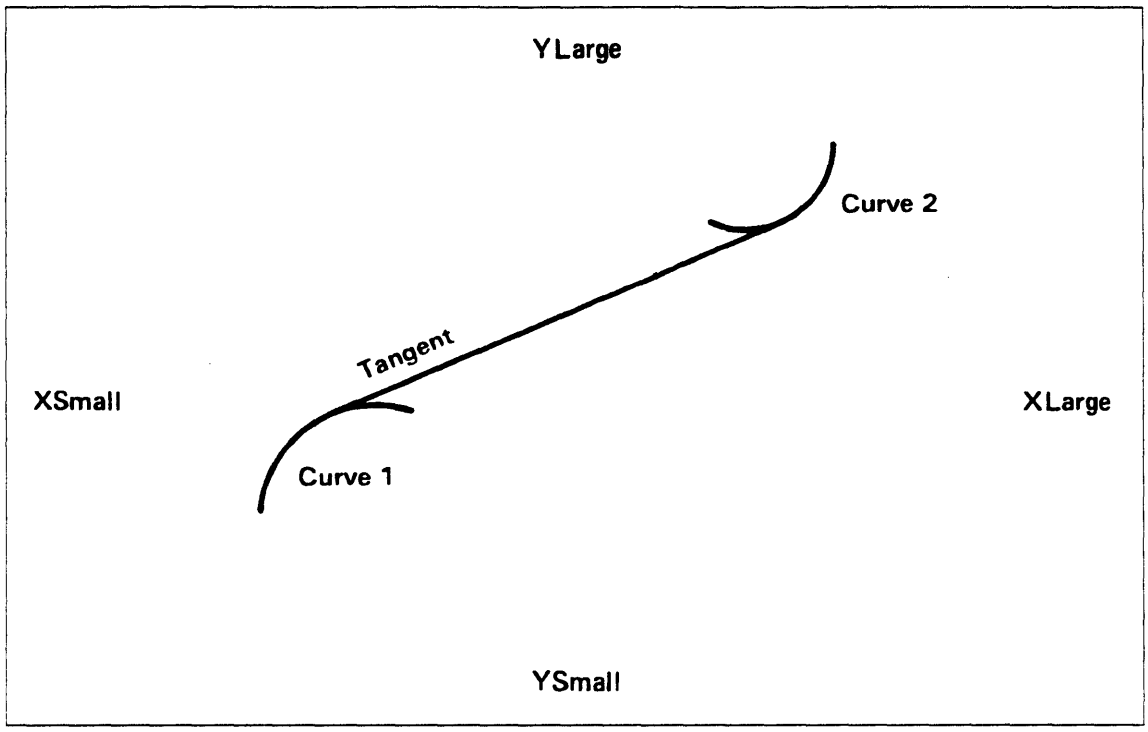


Figure 11-2. Tangent Indicators

Figure 11-3 is an example of a tangent line.

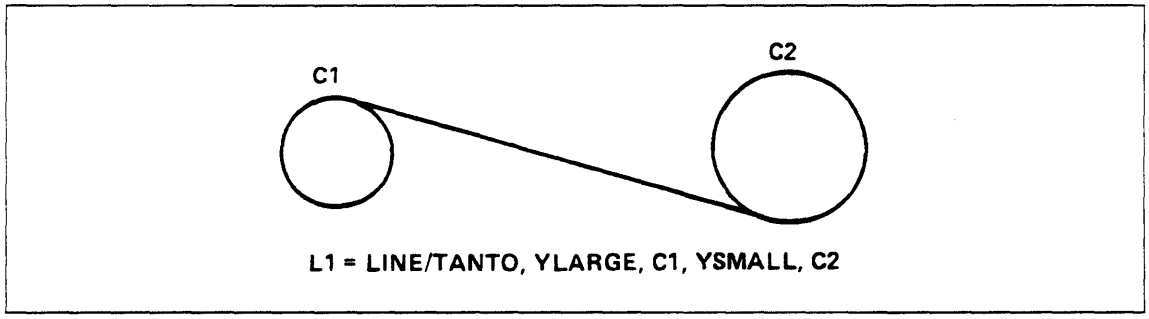


Figure 11-3. Line Tangent to Two Curves

Vertical or Horizontal Through a Point

Statement format:

```
LINE/point,(VERTCL)
           (HORIZ )
```

Parameter	Description
point	The name of the point that you want to draw a line through.
VERTCL	The minor word indicating that the line is to be drawn parallel to the vertical axis.
HORIZ	The minor word indicating that the line is to be drawn parallel to the horizontal axis.

Example:

Program Statement	Explanation
`LIN15=LINE/PT1,VERTCL`	Line LIN15 is created vertically through point PT1.

Through a Point and Tangent to a Curve

Statement format:

```
LINE/point,TANTO,(XSMALL),curve
                 (XLARGE)
                 (YSMALL)
                 (YLARGE)
```

Parameter	Description
point	The variable name of the point through which you want to draw a line.
TANTO	The minor word indicating that a line is to be created tangential to the curve.
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate in which direction the line is to be created. For example, in figure 11-4, a line is drawn from the point to the XSMALL side of the curve.
curve	The name of the curve.

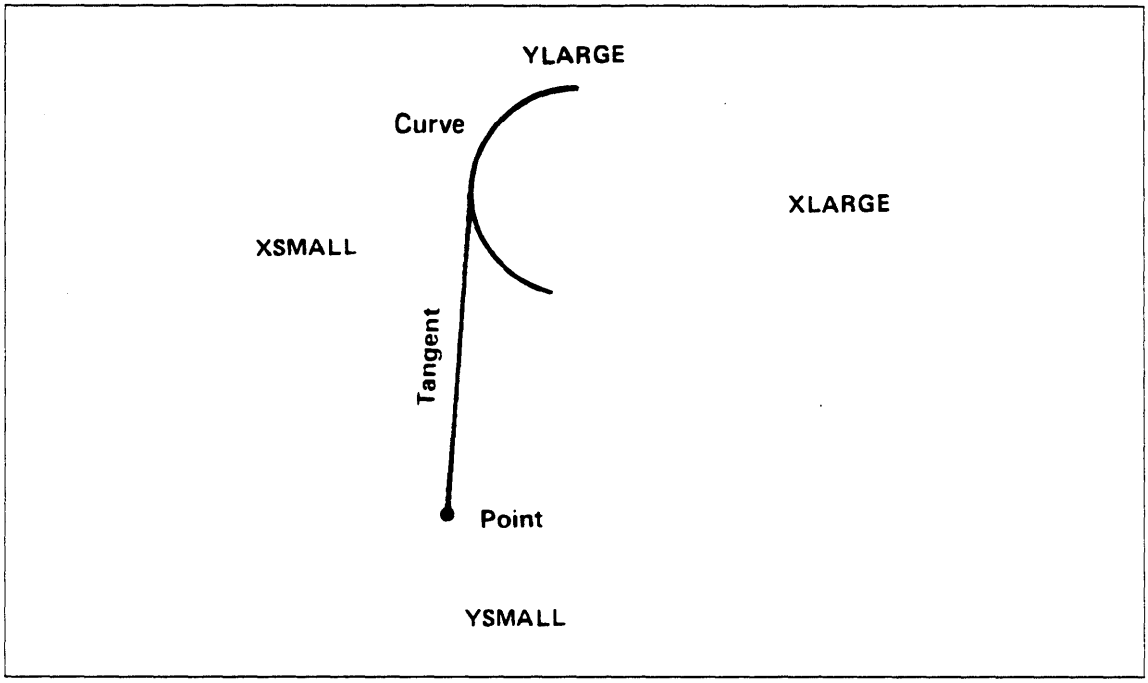


Figure 11-4. Line Through a Point and Tangent to a Curve

Example:

Program Statement	Explanation
`LIN18=LINE/PT1,TANTO$` `,XSMALL,CUR1`	Line LIN18 is created through point PT1 and tangent to the XSMALL side of curve CUR1.

From a Point for a Distance at an Angle to an Existing Line

Statement format:

```
LINE/point,ATANGL,angle,line,DISTNC,distance
```

Parameter	Description
point	The name of the point from which the line is to be drawn.
ATANGL	The minor word for indicating the angle.
angle	The positive angle from an imaginary line parallel to the reference line. Refer to figure 11-5.
line	The name of the reference line.
DISTNC	The minor word for indicating the length of the new line.
distance	The length of the new line in units of measure.

Example:

Program Statement	Explanation
`LIN2=LINE/PT1,ATANGL,15,$` `LIN1,DISTNC,8`	Line LIN2 is created from point PT1 at an angle of 15° to line LIN1 for a distance of 8 units of measure.

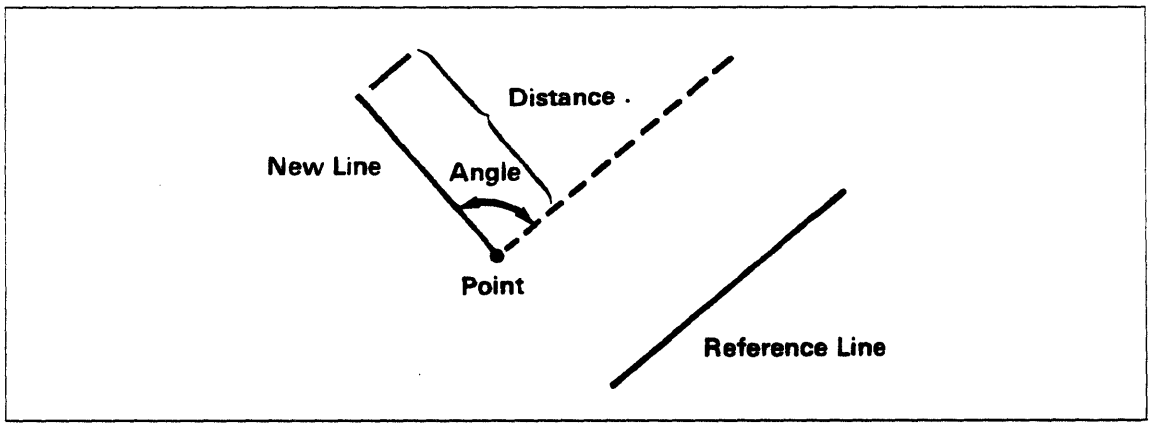


Figure 11-5. From a Point at an Angle

Through a Point and Parallel to a Line

Statement format:

```
LINE/point,PARLEL,line
```

Parameter	Description
point	The name of the point through which the line is to be drawn.
PARLEL	The minor word indicating that a line is to be created through the reference point and parallel to the reference line.
line	The name of the reference line.

Example:

Program Statement	Explanation
`LIN2=LINE/PT1,PARLEL,LIN1`	Line LIN2 is created through point PT1 and parallel to line LIN1.

Through a Point and Perpendicular to a Line

Statement format:

```
LINE/point,PERPTO,line
```

Parameter	Description
point	The name of the point through which the line is to be drawn.
PERPTO	The minor word indicating that a line is to be created through the reference point and perpendicular to the reference line.
line	The name of the reference line.

Example:

Program Statement	Explanation
`LIN2=LINE/PT1,PERPTO,LIN1`	Line LIN2 is created through point PT1 and perpendicular to line LIN1.

Offset Distance Parallel to a Line

Statement format:

```
LINE/PARLEL,line,(XSMALL)
                 (XLARGE),offset
                 (YSMALL)
                 (YLARGE)
```

Parameter	Description
PARLEL	The minor word indicating that a line is to be created parallel to and at an offset distance from the reference line.
line	The name of the reference line.
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate on which side of the reference line the line is to be created.
offset	The offset distance of the new line from the reference line (in units of measure).

Example:

Program Statement	Explanation
`LIN2B=LINE/PARLEL,LIN1,$` `  XLARGE,.5`	Line LIN2B is created parallel to and on the x positive side of line LIN1 at an offset distance of 0.5 units of measure.

Tangent to a Curve and Parallel to a Line

Statement format:

```
LINE/TANTO, { XSMALL } ,curve,PARLEL,line
            { XLARGE }
            { YSMALL }
            { YLARGE }
```

Parameter	Description
TANTO	The minor word indicating that a line is to be created tangent to a curve.
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate on which side of the curve the line is to be created.
curve	The name of the reference curve.
PARLEL	The minor word indicating that a line is to be created parallel to a reference line.
line	The name of the reference line.

Example:

Program Statement	Explanation
L308=LINE/TANTO,$ XLARGE,CUR1,$ PARLEL,LIN2	Line L308 is created tangent to curve CUR1 on the x positive side of the curve and parallel to line LIN2.

Tangent to a Curve and Perpendicular to a Line

Statement format:

```
LINE/TANTO, { XSMALL } ,curve,PERPTO,line
            { XLARGE }
            { YSMALL }
            { YLARGE }
```

Parameter	Description
TANTO	The minor word indicating that a line is to be created tangent to a curve. Refer to figure 11-6.
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate on which side of the curve the line is to be created.
curve	The name of the reference curve.
PERPTO	The minor word indicating that a line is to be created perpendicular to a reference line.
line	The name of the reference line.

Example:

Program Statement	Explanation
L309=LINE/TANTO,$ YLARGE,CUR1,$ PERPTO,LIN2	Line L309 is created tangent to curve CUR1 on the y positive side of the curve and perpendicular to line LIN2.

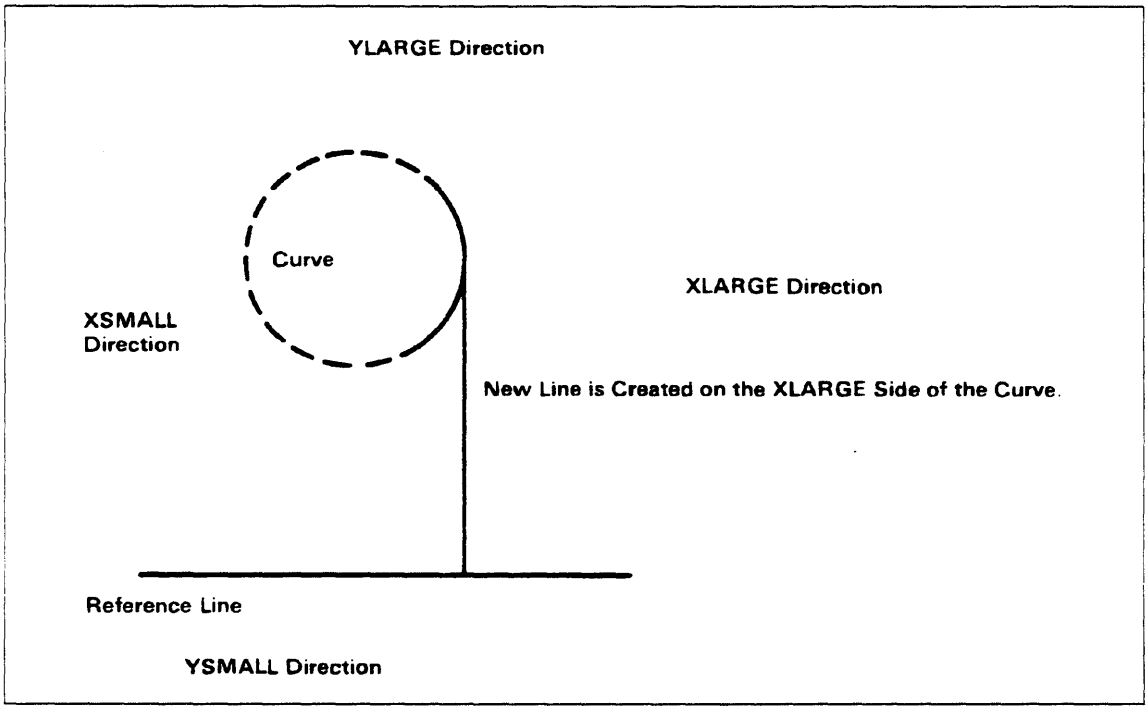


Figure 11-6. Tangent to a Curve and Perpendicular to a Line

Join Two Curve Endpoints

Statement format:

```
                 ( XSMALL )              ( XSMALL )
LINE/END, ( XLARGE ) ,curve1, ( XLARGE ) ,curve2
                 ( YSMALL )              ( YSMALL )
                 ( YLARGE )              ( YLARGE )
```

Parameter	Description
END	The minor word indicating that a line is to be created from the end of curve1 to the end of curve2.
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate which ends of curve1 and curve2 are to be joined by the new line.
curve1, curve2	The names of the curves being connected.

Examples:

Program Statement	Explanation
`CDV=LINE/END,YSMALL,LN1,$` `  XLARGE,LN2`	Line CDV is created from the y negative end of line LN1 to the x positive end of line LN2.
`CDZ=LINE/END,YLARGE,LN1,$` `  XSMALL,LN2`	Line CDZ is created from the y positive end of line LN1 to the x negative end of line LN2.

Infinite Line

Statement format:

```
LINE/INFIN,line
```

Parameter	Description
INFIN	The minor word indicating that a line is to be modified to an infinite line.
line	The name of the line being modified.

Example:

Program Statement	Explanation
`LINE/INFIN,L2020`	Line L2020 is modified to infinite status.

Axis

Statement format:

```
LINE/{XAXIS}
     {YAXIS}
```

Parameter	Description
XAXIS	The minor word indicating that the line is to be defined as the x-axis.
YAXIS	The minor word indicating that the line is to be defined as the y-axis.

Example:

Program Statement	Explanation
`FGH=LINE/XAXIS`	Line FGH is created as the x-axis.

ARC/CIRCLE Statements 12

ARC/CIRCLE Statements 12

ARC/CIRCLE statements create arcs and circles in the part drawing. Arc definitions are equivalent to forms found in menu 11 ARC/CIRCLE/FILLET.

NOTE

The DEFINE statement can be used as an alternative method of producing a large number of circles.

CIRCLE

A circle is the locus of points that are a constant distance (the radius) from a fixed point (the center). An arc is a portion of a circle. The default starting angle is 0.0°, and the default ending angle is 360.0°.

Center Coordinates and Radius

Statement format:

```
CIRCLE/CENTER,xcoord,ycoord,zcoord,RADIUS,radius[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
CENTER	The minor word for indicating the center of the circle.
xcoord	The x-coordinate of the center of the circle.
ycoord	The y-coordinate of the center of the circle.
zcoord	The z-coordinate of the center of the circle.
RADIUS	The minor word for indicating the radius of the circle.
radius	The radius of the circle in units of measure.
GOANG	The minor word for indicating the starting angle of the circle. The starting angle is determined relative to the x positive axis.
goang	The starting angle of the circle in degrees. The default is 0°.
ENDANG	The minor word for indicating the ending angle of the circle.
endang	The ending angle of the circle in degrees. The default is 360°.

Example:

Program Statement	Explanation
`CIR1=CIRCLE/CENTER,2,0,0,$` `  RADIUS,0.5`	Circle CIR1 is created using coordinates x=2, y=0, and z=0 as the center of the circle. The radius is 0.5 units of measure.

Center Point and Radius

Statement format:

```
CIRCLE/CENTER,point,RADIUS,radius[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
CENTER	The minor word for indicating the center of the circle.
point	The name of the center point.
RADIUS	The minor word for indicating the radius of the circle.
radius	The radius of the circle in units of measure.
GOANG	The minor word for indicating the starting angle of the circle. The starting angle is determined relative to the x positive axis.
goang	The starting angle of the circle in degrees. The default is 0°.
ENDANG	The minor word for indicating the ending angle of the circle.
endang	The ending angle of the circle in degrees. The default is 360°.

Example:

Program Statement	Explanation

`A122=CIRCLE/CENTER,PT1,$`
`RADIUS,0.5,$`
`GOANG,45,ENDANG,195`
	Arc A122 is created using point PT1 as the center of the circle that defines the arc. The radius is 0.5 units of measure. The arc starts at 45° from the x positive axis and ends at 195°.

Center Point and Tangent Line

Statement format:

```
CIRCLE/CENTER,point,TANTO,line[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
CENTER	The minor word for indicating the center of the circle.
point	The name of the center point.
TANTO	The minor word indicating that a circle is to be created tangent to a line.
line	The name of the reference line.
GOANG	The minor word for indicating the starting angle of the circle.
goang	The starting angle of the circle in degrees. The default is 0°.
ENDANG	The minor word for indicating the ending angle of the circle.
endang	The ending angle of the circle in degrees. The default is 360°.

Example:

Program Statement	Explanation
`C129=CIRCLE/CENTER,PT1,$` `  TANTO,LIN1`	Circle C129 is created using point PT1 as the center. The circumference is tangent to line LIN1.

Center Point and Tangent Circle

Statement format:

```
CIRCLE/CENTER,point,TANTO,{LARGE},circle[,GOANG,goang][,ENDANG,endang]
                          {SMALL}
```

Parameter	Description
CENTER	The minor word for indicating the center of the circle.
point	The name of the center point.
TANTO	The minor word indicating that a tangent circle is to be used as a reference.
LARGE	The minor word indicating that the larger of two possible circles is to be drawn.
SMALL	The minor word indicating that the smaller of two possible circles is to be drawn.
circle	The name of the reference circle.
GOANG	The minor word for indicating the starting angle of the circle.
goang	The starting angle of the circle in degrees. The default is 0°.
ENDANG	The minor word for indicating the ending angle of the circle.
endang	The ending angle of the circle in degrees. The default is 360°.

Example:

Program Statement	Explanation
`C154=CIRCLE/CENTER,PT1,$` `  TANTO,LARGE,A448`	Circle C154 is created using point PT1 as the center. It is the largest of two possible circles tangent to circle A448.

Center Point and Point on Circumference

Statement format:

```
CIRCLE/CENTER,point1,point2[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
CENTER	The minor word for indicating the coordinate values for the center of the circle.
point1	The name of the center point.
point2	The name of the circumference point that determines the radius.
GOANG	The minor word for indicating the starting angle of the circle.
goang	The starting angle of the circle in degrees. The default is 0°.
ENDANG	The minor word for indicating the ending angle of the circle.
endang	The ending angle of the circle in degrees. The default is 360°.

Example:

Program Statement	Explanation
`BQ12=CIRCLE/CENTER,PT1,PT2,$` `ENDANG,135`	Circle BQ12 is created using point PT1 as the center and point PT2 as a reference point for the radius. The arc starts at 0° from the x positive axis and ends at 135°.

Three Points on the Circumference

Statement format:

```
CIRCLE/(xt1,yt1,zt1,xt2,yt2,zt2,xt3,yt3,zt3)
       (      point,point,point            )
```

Parameter	Description
xt, yt, zt	The coordinate values of three circumference points.
point	The entity names of three circumference points.

The arc is defined through all three points. The endpoints are the first and last points specified. The actual direction of definition depends on the orientation of previously defined views.

Example:

Program Statement	Explanation
`A323=CIRCLE/PT1,PT2,PT3`	Arc A323 is created using points PT1, PT2, and PT3 as three points on the circumference.

Existing Arc

Statement format:

```
CIRCLE/circle[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
circle	The name of a reference circle.
GOANG	The minor word for indicating the starting angle of the circle.
goang	The starting angle of the circle in degrees. The default is 0°.
ENDANG	The minor word for indicating the ending angle of the circle.
endang	The ending angle of the circle in degrees. The default is 360°.

Example:

Program Statement	Explanation
`DB22=CIRCLE/DB10,$` `  GOANG,30,ENDANG,135`	Arc DB22 is created using arc DB10 as a reference. Arc DB22 starts at 30° from the x positive axis and ends at 135°.

Inscribed in Three Lines

Statement format:

```
CIRCLE/line,line,line[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
line	The entity names of the three reference lines. The arc is defined moving from the first line to the last line in a counterclockwise direction.
GOANG	The minor word for indicating the starting angle of the circle.
goang	The starting angle of the circle in degrees. The default is 0°.
ENDANG	The minor word for indicating the ending angle of the circle.
endang	The ending angle of the circle in degrees. The default is 360°.

Example:

Program Statement	Explanation
`XX67=CIRCLE/LN1,LN2,LN3`	Circle XX67 is created within the boundaries described by lines LN1, LN2, and LN3.

Normal to View

Statement format:

```
CIRCLE/NORMAL,point1,point2[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
NORMAL	The minor word for generating a circle normal to view.
point1	The name of the center point of the circle.
point2	The name of a circumference point of the circle.
GOANG	The minor word for indicating the starting angle of the circle.
goang	The starting angle of the circle in degrees. The default is 0°.
ENDANG	The minor word for indicating the ending angle of the circle.
endang	The ending angle of the circle in degrees. The default is 360°.

Example:

Program Statement	Explanation
`DB22=CIRCLE/NORMAL,PT1,PT2,$` `GOANG,30,ENDANG,135`	Arc DB22 is created normal to view using PT1 as the center point and PT2 as the circumference point. Arc DB22 starts at 30° from the x positive axis and ends at 135°.

FILLET

A fillet is an arc tangential to any two geometric entities: point, line, circle, conic, or spline. The fillet is generated counterclockwise from the first entity to the second entity. There are two methods of indicating the fillet center:

- Specifying the coordinate of the center point.
- Indicating the center point using positional words.

The preferred method is to indicate the coordinates, as this avoids ambiguity and locates the center point factor.

The other method locates the center point by specifying the end points with positional words. The center point is the halfway point between the end points. In cases where this halfway point lands on one of the entities, this method fails, and you must specify coordinates.

No Trim

Statement format:

```
        ⎧XSMALL⎫                ⎧XSMALL⎫
FILLET/⎪XLARGE⎪,entity1,⎪XLARGE⎪,entity2,RADIUS,radius
        ⎪YSMALL⎪                ⎪YSMALL⎪
        ⎩YLARGE⎭                ⎩YLARGE⎭
```

Parameter	Description
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate in which quadrant the center point of the fillet is to be created.
entity	The names of the entities between which the fillet is to be created. The entities can be points, lines, circles, conics, or splines.
RADIUS	The minor word for indicating the fillet radius.
radius	The fillet radius in units of measure.

Example:

Program Statement	Explanation
`F123=FILLET/XSMALL,S1,XLARGE,$` `  L1,RADIUS,0.5`	Fillet F123 is created between spline S1 and line L1. The center is located in the x negative direction from spline S1 and the x positive direction from line L1. The fillet radius is 0.5 units of measure.

Figure 12-1 shows a fillet. The fillet is created in the quadrant defined by the XSMALL end of entity 1 and the YSMALL end of entity 2. The four quadrants are defined by the intersection of the two lines.

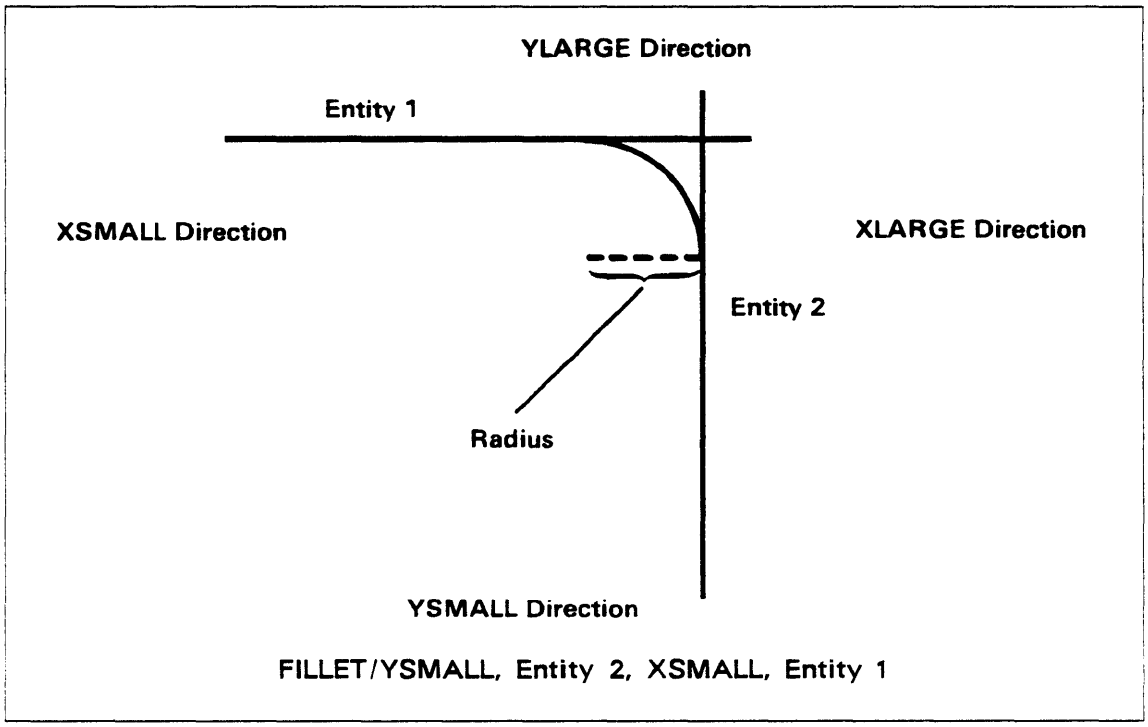


Figure 12-1. Fillet

Auto Trim

Statement format:

```
                 [XSMALL]                [XSMALL]
FILLET/          [XLARGE] ,entity1,      [XLARGE] ,entity2,RADIUS,radius,TRIM
                 [YSMALL]                [YSMALL]
                 [YLARGE]                [YLARGE]
```

Parameter	Description
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate in which quadrant the center point of the fillet is to be created.
entity	The names of the entities between which the fillet is to be created. The entities can be points, lines, circles, conics, or splines.
RADIUS	The minor word for indicating the fillet radius.
radius	The fillet radius in units of measure.
TRIM	The minor word indicating that both entities are to be trimmed or extended.

Program Statement	Explanation
`F124=FILLET/XSMALL,S1,XLARGE,$` `  L1,RADIUS,0.5,TRIM`	Fillet 124 is created between spline S1 and line L1. The center is located in the x negative direction from spline S1. The fillet radius is 0.5 units of measure, and it is automatically trimmed.

Figure 12-2 is an example of a fillet with automatic trim.

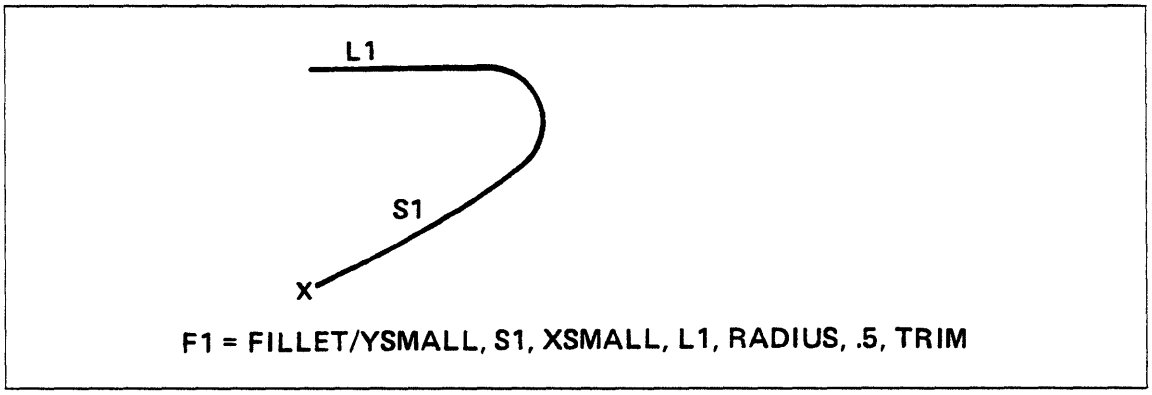
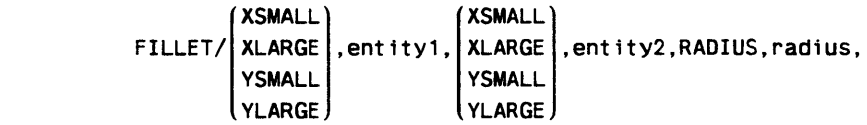
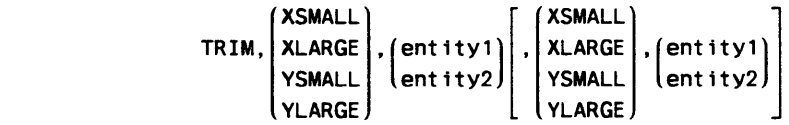


Figure 12-2. Fillet with Automatic Trim

Selective Trim

Statement format:





Parameter	Description
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate in which quadrant the point is to be created.
entity	The names of the entities between which the fillet is to be created. The entities can be points, lines, circles, conics, or splines.
RADIUS	The minor word for indicating the fillet radius.
radius	The fillet radius in units of measure.
TRIM	The minor word indicating that either or both entities are to be trimmed or extended.

Fillet Center by Coordinates

Statement format:

```
FILLET/xcoord,ycoord,entity1,entity2,RADIUS,radius,
```

```
[ TRIM[ , (XSMALL / XLARGE / YSMALL / YLARGE) , (entity1 / entity2) ][ , (XSMALL / XLARGE / YSMALL / YLARGE) , (entity1 / entity2) ] ]
```

Parameter	Description
xcoord	The x-coordinate of the center of the fillet.
ycoord	The y-coordinate of the center of the fillet.
entity	The names of the entities between which the fillet is created. The entities can be points, lines, circles, conics, or splines.
RADIUS	The minor word indicating the fillet radius.
radius	The fillet radius in units of measure.
TRIM	The minor word indicating that either or both entities are trimmed or extended.
XSMALL	The minor word for the x negative direction.
XLARGE	The minor word for the x positive direction.
YSMALL	The minor word for the y negative direction.
YLARGE	The minor word for the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate in which quadrant the center point of the fillet is created.

Two-Dimensional Curve Statements 13

Two-Dimensional Curve Statements 13

Two-dimensional curve statements create curves in one plane in the part drawing. Two-dimensional curve definitions are equivalent to types and forms found in menu 12 OTHER CURVES.

ELLIPS

An ellipse is a plane curve that is the locus of a point, which moves so that the sum of the distances from the point to two other points in the plane is a constant.

Statement format:

```
ELLIPS/CENTER,point,half major,half minor,axis angle[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
CENTER	The minor word for indicating the center of the ellipse.
point	The name of the center point.
half major	One-half the length of the major axis in units of measure.
half minor	One-half the length of the minor axis in units of measure.
axis angle	The angle the major axis makes with the x-axis.
GOANG	The minor word for indicating the starting angle.
goang	The starting angle of the circumference of the ellipse in degrees. The default is 0°.
ENDANG	The minor word for indicating the ending angle.
endang	The ending angle of the circumference of the ellipse in degrees. The default is 360°.

Example:

Program Statement	Explanation
`EL1=ELLIPS/CENTER,PT1,$` `.87,.44,45,GOANG,45`	An ellipse is created with the point PT1 as the center. The major axis is 1.74 (2 x 0.87) units of measure. The minor axis is 0.88 (2 x 0.44) units of measure. The angle of the major axis to the x-axis is 45°. The ellipse starts at 45° from the major axis and ends at 360°.

GCONIC

A general conic is a plane curve represented by the equation:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

The circle, ellipse, hyperbola, and parabola can be defined in this way.

Statement format:

```
GCONIC/coeff A,coeff B,coeff C,coeff D,coeff E,coeff F
```

Parameter	Description
coeff	The coefficients of the quadratic equation for conics.

Example:

Program Statement	Explanation
`GC1=GCONIC/.66,0.0,.93,$` `-6.0,-6.8,8.56`	General conic GC1 is created with the following coefficients: A=0.66, B=0.0, C=0.93, D=-6.0, E=-6.8, and F=8.56.

HYPERB

A hyperbola is a plane curve that is the locus of a point, which moves so that the difference between the distances from the point to two other points in the plane is a constant. The curve is composed of two parts. All values of x for the hyperbola must be positive (before rotation). The hyperbola is defined between −45° and +45°.

Statement format:

```
HYPERB/CENTER,point,half transverse,half conjugate,angle
```

Parameter	Description
CENTER	The minor word for indicating the vertex of the hyperbola.
point	The name of the vertex.
half transverse	One-half the length of the transverse axis of the hyperbola in units of measure.
half conjugate	One-half the length of the conjugate axis of the hyperbola in units of measure.
angle	The angle between the transverse axis and the x-axis.

Example:

Program Statement	Explanation
`HYP1=HYPERB/CENTER,PT1,$` `.9,.5,0`	Hyperbola HYP1 is created with point PT1 as the vertex. The transverse axis is 1.8 (2 x 0.9) units of measure. The conjugate axis is 1.0 (2 x 0.5) units of measure. The hyperbola is not rotated in relation to the x-axis.

PARABO

A parabola is the locus of a point, which moves so that its distance from a fixed point is equal to its distance from a fixed line not through the point.

Statement format:

```
PARABO/CENTER,point,length,angle,y maximum,y minimum
```

Parameter	Description
CENTER	The minor word for indicating the vertex of the parabola.
point	The name of the vertex.
length	The distance from the vertex to the focus of the parabola in units of measure.
angle	The angle between the principal axis and the x-axis.
y maximum	The y maximum bound for the parabola.
y minimum	The y minimum bound for the parabola.

Example:

Program Statement	Explanation
`PAR1=PARABO/CENTER,PT1,$` `1.5,270,1,-1`	Parabola PAR1 is created with point PT1 as the vertex. The distance from the vertex to the focus of the parabola is 1.5 units of measure. The angle of the principal axis to the x-axis is 270°. In other words, the parabola is rotated 270°. The y maximum bound is +1, and the y minimum bound is −1.

Figure 13-1 is an example of a parabola.

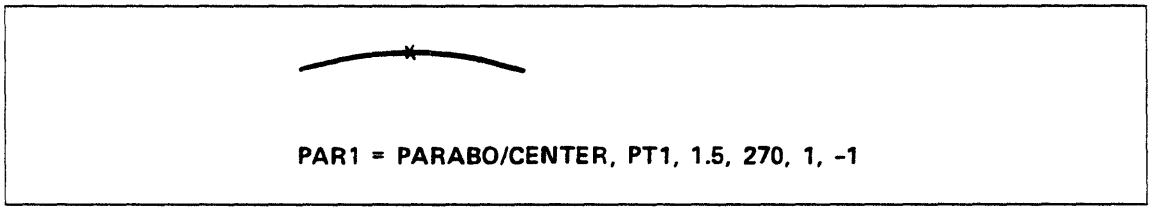


Figure 13-1. Parabola

PTSET

A point set is a curve defined by the intersection of a surface and an infinite plane. The curve is defined in the view of the slicing plane. The starting point should be near an edge. Refer to menu 15.1.2 SURFACE EDGE CURVE in the ICEM Advanced Design manual.

Minimum Number of Points to Approximate a Surface-to-Plane Intersection

Statement format:

```
PTSET/surface,plane[,NUMBER,number pts],START,point
```

Parameter	Description
surface	The name of the reference surface.
plane	The name of the reference plane.
NUMBER	The minor word for indicating the minimum number of intersection points to be used to approximate the curve. The number of intersection points is translated into a tolerance using the following scheme:

Points	Tolerance
<20	0.2540 mm (0.0100 in)
>20	0.1270 mm (0.0050 in)
>30	0.0254 mm (0.0010 in)
>50	0.0127 mm (0.0005 in)

number pts	The minimum number of intersection points to be used to approximate the curve.
START	The minor word for indicating the starting point of the intersection curve.
point	The name of the starting point of the intersection curve.

Points on a Surface-Plane Intersection Curve Satisfying the Specified Tolerance

Statement format:

```
PTSET/surface,plane[,tolerance,step size],START,point
```

Parameter	Description
surface	The name of the reference surface.
plane	The name of the reference plane.
tolerance	The tolerance factor for approximating the intersection curve.
step size	The step size, which is the approximate distance between each point on the intersection curve.
START	The minor word for indicating the starting point of the intersection curve.
point	The name of the starting point of the intersection curve.

SPLINE

A spline is a free-form curve generated from a series of ordered points. The slope and curvature are contiguous at each given point of the spline.

All points or point coordinates are projected into transform space before processing. The depth coordinate is determined by ZSURF if the points are not coplanar. A minimum of three points and a maximum of 42 points can be specified. Refer to menu 12.1 SPLINE in the ICEM Design/Drafting Basic Construction manual for further information.

The spline start or end condition can be either circular (default) or parabolic. For circular start or end conditions, specified slopes are optional.

The optional parabolic start and end conditions provide for a third degree starting or ending parabolic segment in the spline. If parabolic conditions are not specified, the end segments are assumed to be circular.

The spline tolerance option specifies the maximum discontinuity in the curvature of the spline. Acceptable values are positive and greater than or equal to 0.000001. Small values slow the creation of the spline, and depending on the particular points, may prevent the spline from being created.

The optional circular start and end slopes force the angle of tangency of the spline at the first or last points. If the start and end slopes are not specified, a circular end condition is assumed.

The point movement option provides for a minimum of strain energy at each point. If the point movement option is used, the spline points are adjusted by the specified tolerance. The spline constrains to the first and last points, but is within tolerance of the intermediate points.

The spline can be defined by the following methods:

- Specifying existing points or coordinates individually
- Polar from origin
- Specifying the number of points with a point array

Points, Coordinates, or Polar from Origin

Statement format:

```
SPLINE/(point                    )[,SLOPE,slope]  (point                    )[,...]
       (xcoord,ycoord            )[ PARBLC      ],(xcoord,ycoord            )
       (RADANG,radius,radang     )                (RADANG,radius,radang     )

[,SLOPE,slope][,(XYMOVE)                          ][,TOLER,spline tol value]
[ PARBLC     ][,(XMOVE ),point adj value          ]
               (YMOVE )
```

Parameter	Description
point	The name of the spline's start point.
xcoord	The x-coordinate of the spline's start point.
ycoord	The y-coordinate of the spline's start point.
RADANG	The minor word for specifying the points of the spline from the origin (0,0) using the radius and angle, a polar method.
radius	The radius from the origin to a reference point in units of measure.
radang	The angle of a reference point to the horizontal axis in degrees.
SLOPE	The minor word for indicating a slope angle for the starting or ending point.
slope	The tangent angle at the starting or ending point.
PARBLC	The minor word indicating that the first or last segment of the spline is a parabola.
XYMOVE, XMOVE, YMOVE	The minor words indicating the direction of point adjustment value. XYMOVE allows the movement constraint in both the xt and yt directions. XMOVE allows movement only in the xt direction; YMOVE allows movement only in the yt direction. Use movement constraint cautiously and only with very small values. Avoid movement constraint unless it is absolutely necessary.
point adj value	The amount of allowable point movement to be used in approximating the curve.
TOLER	The minor word for indicating the spline tolerance.
spline tol value	The maximum amount of discontinuity in the curvature of the spline.

Example:

Program Statement	Explanation
`S1=SPLINE/1,1,SLOPE,30,$` `2.68,2,4,4,XMOVE,.005`	Spline S1 is created starting at coordinates (1,1) with a slope angle of 30°. Additional points are at coordinates (2.68,2) and (4,4). The point adjustment allowed is 0.005 units of measure in the xt direction only.

Figure 13-2 shows the spline created in this example.

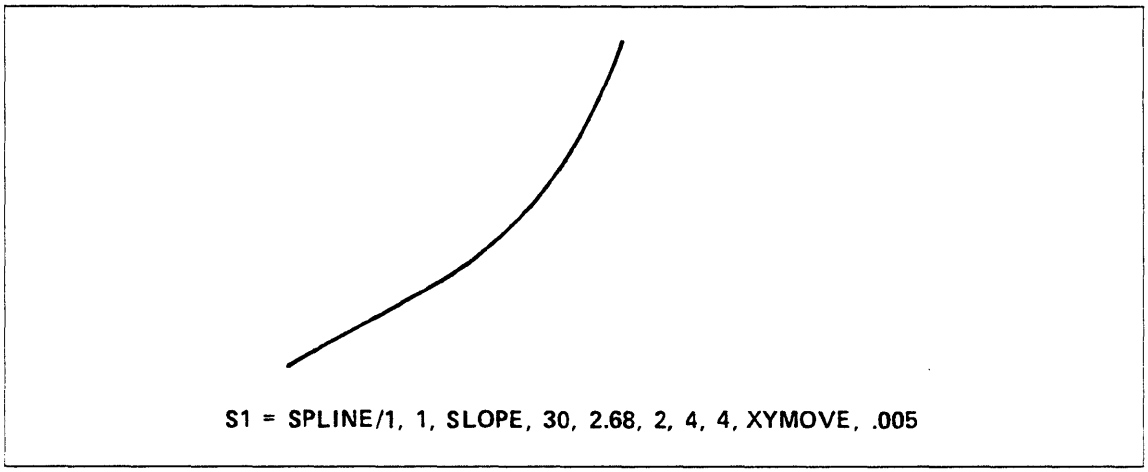


Figure 13-2. Spline

Number of Points

Statement format:

```
SPLINE/point,NUMBER,number,{point array}
                           {coord array}
```

Parameter	Description
point	The name of the starting point for the spline.
NUMBER	The minor word for specifying a number of points in a point or coordinate array.
number	The number of points in the point array, or the number of points the coordinate array defines.
point array	The name of the point array.
coord array	The name of the coordinate array.

Example:

Program Statement	Description
`REAL/CORDS(6)`	Allocate space for the real array, CORDS.
`ENTITY/PNTS(3),SPT`	Define and allocate space for the entities, PNTS and SPT.
`DATA/CORDS,2,4,3,9,4,16`	Initialize the CORDS array.
`SPT = POINT/0,0,0`	Define the point SPT.
`SPLINE/SPT,NUMBER,3,CORDS(3)`	Create a spline starting at SPT and using the 3 sets of coordinates in CORDS.

`PNTS(1) = POINT/5,25,0`
`PNTS(2) = POINT/6,36,0`
`PNTS(3) = POINT/7,49,0`
	Define the points.

`SPLINE/SPT,NUMBER,3,PNTS(1)`	Creat a spline starting at SPT and using the 3 points in PNTS.

STRING

The STRING statement creates a string figure, a series of points and/or arcs linked to form a single geometric entity. Each string segment starts at the end of the previous segment. Arcs are tangent to the end of the previous segment.

Strings are defined by:

- A start point and segments defined as arcs or lines. After defining a start point, you can define any number of lines and arcs in any order.
- A start point and segments defined by a number of points from a point or coordinate array.

Arcs or Lines

Statement format:

```
STRING/start definition,{line endpoint definition}
                        {arc definition          }

     [,{arc definition          }[,...]]
     [ {line endpoint definition}      ]
```

Parameter	Description

start definition	The start point is defined by either the name of a point or the x- and y-coordinates of a point, as follows:
	`{point          }`
	`{xcoord,ycoord}`

Parameter	Description
point	The name of the start point.
xcoord,ycoord	The x-coordinate and y-coordinate of the start point.

Parameter	Description
line endpoint definition	The line endpoint is defined by either the name of a line endpoint, the x- and y-coordinates of the line endpoint, or delta values from the previous point, as follows:

```
(point                    )
(xcoord,ycoord            )
(DELTA,deltax,deltay      ),...
(DELTAX, deltax           )
(DELTAY, deltay           )
```

Parameter	Description
point	The name of a point.
xcoord,ycoord	The x-coordinate and y-coordinate of the line endpoint.
DELTA	The minor word indicating that the next field of information is the delta x and y values from the last defined point.
DELTAX	The minor word indicating that the next field is the deltax value from the last defined point.
DELTAY	The minor word indicating that the next field is the deltay value from the last defined point.
deltax	The delta x distance from the previous point.
deltay	The delta y distance from the previous point.

If either deltax or deltay is 0, it can be omitted by enclosing the changing delta value in parentheses.

Parameter	Description
arc definition	The arc endpoint is defined by the clockwise or counterclockwise radius and angle from the previous point, as follows:

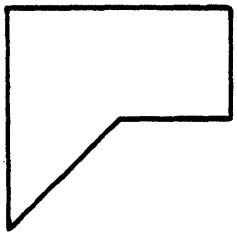
```
[ [CW  ],radius,angle ],...
[ [CCW ]              ]
```

Parameter	Description
CW	The minor word for indicating the clockwise radius and angle of the next point. The arc center is automatically generated, and the arc starts at the last defined point in a clockwise direction.
CCW	The minor word for indicating the counterclockwise radius and angle of the next point. The arc center is automatically generated, and the arc starts at the last defined point in a counterclockwise direction.
radius	The radius of the next point.
angle	The positive angle from the horizontal axis in a counterclockwise (CCW) or clockwise (CW) direction to the next point.

Examples:

Program Statement	Explanation
`STR3=STRING/PT1,PT2,PT3,PT4`	String STR3 is defined using PT1 as the start point. Lines are drawn from PT1 to PT2, from PT2 to PT3, and from PT3 to PT4.
`STR4=STRING/3,3,3,4,$` `4,4,3,3`	A one-unit box is created using coordinates (3,3) as the start point. Lines are drawn from (3,3) to (3,4), from (3,4) to (4,4), and from (4,4) to (3,3).
`STR5=STRING/PT1,DELTA,.5,.6,$` `DELTA,.3,.4,`	String STR5 is defined using PT1 as the start point. Lines are drawn from PT1 to delta coordinates (.5,.6) and from delta coordinates (.5,.6) to delta coordinates (.3,.4).
`STR6=STRING/PT1,CW,$` `5,45`	String STR6 is an arc of 45° clockwise from PT1 with a radius of 5 units of measure.

Figure 13-3 is an example of a string.



STR1 = STRING/2, 2, 2, 4, 4, 4, 4, 3, 3, 3, 2, 2

Figure 13-3. String

By Coordinate Array

Statement format:

```
STRING/(point          ),NUMBER,number,(point array)
       (xcoord,ycoord)                 (coord array)
```

Parameter	Description
point	The name of the start point for the spline.
xcoord,ycoord	The x-coordinate and y-coordinate of the start point.
NUMBER	The minor word for specifying a number of points in a point or coordinate array.
number	A number of points (coordinates) in the entity array.
point array	The name of a point array.
coord array	The name of a coordinate array.

Example:

Program Statement	Explanation
`STR9=STRING/PT1,NUMBER,16,$` `APNT`	String STR9 is defined using the coordinates of 16 points stored in array APNT. Lines are drawn between the points starting from PT1 to the last point coordinate in APNT.

Entity Manipulation Statements 14

Entity Manipulation Statements 14

Entity manipulation statements are used to manipulate entities by array, group, mirror, template, projection, and retrieval. Entities are manipulated using definitions that are equivalent to types and forms found in menu 13 ENTITY MANIPULATION in the ICEM Design/Drafting Basic Construction manual.

ARRAY

The ARRAY statement reproduces geometric entities as displayed rectangular or circular arrays.

Rectangular Array Defined by the Number of Rows and Columns

Statement format:

```
ARRAY/name,RECT[,{          point          }][,DELTAX,deltax][,DELTAY,deltay]
                 {XSTART,xstart,YSTART,ystart}

      [,NUMBRX,numbrx][,NUMBRY,numbry][,TILTAN,angle]

      [,{ARRDO},element column number,element row number,...]
        {DONT }
```

Parameter	Description
name	The name of the entity to be reproduced as a displayed rectangular array.
RECT	The minor word indicating a rectangular array.
point	The name of the center point. If the center point is omitted, 0,0 is assumed.
XSTART	The minor word for indicating the x-coordinate of the center point. If the center point is omitted, 0,0 is assumed.
xstart	The x-coordinate of the center point.
YSTART	The minor word for indicating the y-coordinate of the center point. If the center point is omitted, 0,0 is assumed.
ystart	The y-coordinate of the center point.
DELTAX	The minor word for indicating the delta x value of the distance between array elements.
deltax	The delta x distance between array elements in units of measure.
DELTAY	The minor word for indicating the delta y value of the distance between array elements.

Parameter	Description
deltay	The delta y distance between array elements in units of measure.
NUMBRX	The minor word for indicating the number of column array elements to be duplicated in the x direction. The x direction is horizontal to the right before rotation.
numbrx	The number of column array elements to be duplicated in the x direction. This is the number of columns created.
NUMBRY	The minor word for indicating the number of row array elements duplicated in the y direction. The y direction is vertically up before rotation.
numbry	The number of row array elements duplicated in the x direction. This is the number of rows created.
TILTAN	The minor word for indicating the positive angle from the horizontal axis at which the array is created. The angle can be between 0° and 90°.
angle	The positive angle from the horizontal axis where the array is to be created.
ARRDO	The minor word for indicating which row and column array element to generate. Elements must be specified by column and row pairs.
DONT	The minor word for indicating which row and column array element not to generate. Elements must be specified by column and row pairs.
element column number	The number of the rectangular array column containing the element.
element row number	The number of the rectangular array row containing the element.

If the array created is to be a line, the entry for DELTAX, DELTAY, NUMBRX, or NUMBRY can be omitted entirely. An array is a line if DELTAX or DELTAY is 0, or if NUMBRX or NUMBRY is 1.

The maximum size of NUMBRX times NUMBRY is 100 minus the number of ARRDOs or DONTs given. The default TILTAN is 0.0.

ARRDO is used when only specific entities are to be generated. Each entity is listed by a pair of numbers. For example, ARRDO,3,2 means do the entity in the third column from the left, second row from the bottom.

DONT is used when specific entities are to be omitted.

Example:

Program Statement	Explanation

`AR12=ARRAY/GRP1,RECT,$`
`DELTAX,1.75,$`
`DELTAY,.75,NUMBRX,4,$`
`NUMBRY,3,DONT,2,2,3,2`
	Array AR12 is a rectangular array of group GRP1. The x distance between elements is 1.75 units of measure, and y distance is 0.75 units of measure. Four columns of group GRP1 are created in the x direction, and three rows of the group are created in the y direction. The following array elements are excluded: the element in column 2, row 2 and the element in column 3, row 2.

Circular Array Defined by the Number of Copies

Statement format:

```
ARRAY/name,CIRC[,(        point        )],RADIUS,radius[,GOANG,goang]
               [ (XSTART,xstart,YSTART,ystart)]

,NUMBER,copies[,ENDANG,endang][,(ARRDO),element number,...]
                               [ (DONT) ]
```

Parameter	Description
name	The name of the entity to be reproduced as a displayed circular array.
CIRC	The minor word indicating a circular array.
point	The name of the center point. If the center point is omitted, 0,0 is assumed.
XSTART	The minor word for indicating the x-coordinate of the center point. If the center point is omitted, 0,0 is assumed.
xstart	The x-coordinate of the center point.
YSTART	The minor word for indicating the y-coordinate of the center point. If the center point is omitted, 0,0 is assumed.
ystart	The y-coordinate of the center point.
RADIUS	The minor word for indicating the radius of the circular array.
radius	The radius of the circular array in units of measure.
GOANG	The minor word for indicating the starting angle of the circular array. The default is 0.
goang	The starting angle of the circular array in degrees.
NUMBER	The minor word for indicating the number of copies of the entity. The number of copies determines the angle between copies. For example, for a circular array from 0° to 360° with 6 copies, each copy is placed 60° from the previous copy.
copies	The number of copies of the entity.
ENDANG	The minor word for indicating the ending angle of the circular array.
endang	The ending angle of the circular array in degrees.
ARRDO	The minor word for indicating which element to generate.
DONT	The minor word for indicating which element not to generate.
element number	The number of the circular array element.

For circular arrays, if the center point is omitted, 0,0 is assumed. If the starting angle is omitted, 0° is assumed. If the ending angle is omitted, 360° is assumed. The maximum number of duplications is 100 minus the number of ARRDOs or DONTs given.

ARRDO is used when only specific entities are to be generated. Each copy is indicated by a number. The first entity duplicated is 1, and, counting counterclockwise, the next copy is 2, the next is 3, and so forth. For example, ARRDO,2,5 means the second and fifth copies are to be generated.

DONT is used when specific sets of entities are to be omitted.

Example:

Program Statement	Explanation

`AR24=ARRAY/GRP1,CIRC,PT1,$`
`  RADIUS,.75,NUMBER,8,$`
`  DONT,6,7,8`
	Array AR24 is a circular array with point PT1 as the center point of the array. Group GRP1 is the element reproduced. The radius of the elements from the center point is 0.75 units of measure. There are eight elements in the array, and they are spaced 45° apart (360/8=45). Elements 6, 7, and 8 are excluded.

Circular Array Defined by the Number of Copies and the Delta Angle Between Copies

Statement format:

```
ARRAY/name,CIRC[,{          point          }],RADIUS,radius[,GOANG,goang]
                 {XSTART,xstart,YSTART,ystart}

      ,NUMBER,copies,DELANG,delang[,{ARRDO},element number,...]
                                    {DONT }
```

Parameter	Description
name	The name of the entity to be reproduced as a displayed circular array.
CIRC	The minor word indicating a circular array.
point	The name of the center point. If the center point is omitted, 0,0 is assumed.
XSTART	The minor word for indicating the x-coordinate of the center point. If the center point is omitted, 0,0 is assumed.
xstart	The x-coordinate of the center point.
YSTART	The minor word for indicating the y-coordinate of the center point. If the center point is omitted, 0,0 is assumed.
ystart	The y-coordinate of the center point.
RADIUS	The minor word for indicating the radius of the circular array.
radius	The radius of the circular array in units of measure.

Parameter	Description
GOANG	The minor word for indicating the starting angle of the circular array. The default is 0°.
goang	The starting angle of the circular array in degrees.
NUMBER	The minor word for indicating the number of copies of the entity.
copies	The number of copies of the entity.
DELANG	The minor word for indicating the delta angle between copies. The delta angle determines the angle between copies. For example, a delta angle of 30° places each copy 30° from the previous copy.
delang	The delta angle between copies.
ARRDO	The minor word for indicating which element to generate.
DONT	The minor word for indicating which element not to generate.
element number	The number of the circular array element.

Example:

Program Statement	Explanation

`AR33=ARRAY/CIR1,CIRC,$`
`  RADIUS,.75,NUMBER,6,$`
`  DELANG,30`
	Array AR33 is a circular array with coordinates (0,0) as the center point of the array. Circle CIR1 is the element duplicated. The radius of the elements from the center point is 0.75 units of measure. There are six elements in the array, and they are spaced 30° apart. The starting angle is 0° (the default).

Circular Array Defined by the Delta Angle Between Copies

This statement produces copies at every delta angle from the starting angle to the ending angle. In other words, the number of copies is determined by the starting and ending angles and the delta angle between copies.

(start angle - end angle) / delta angle between copies = number of copies

Statement format:

```
ARRAY/name,CIRC[,{          point          }][,RADIUS,radius[,GOANG,goang]
                 {XSTART,xstart,YSTART,ystart}]

,DELANG,delang[,ENDANG,endang][,{ARRDO},element number,...]
                                {DONT }
```

Parameter	Description
name	The name of the entity to be reproduced as a displayed circular array.
CIRC	The minor word indicating a circular array.
point	The name of the center point. If omitted, 0,0 is assumed.
XSTART	The minor word for indicating the x-coordinate of the center point.
xstart	The x-coordinate of the center point.
YSTART	The minor word for indicating the y-coordinate of the center point.
ystart	The y-coordinate of the center point.
RADIUS	The minor word for indicating the radius of a circular array.
radius	The radius of the circular array in units of measure.
GOANG	The minor word for indicating the starting angle of the circular array. The default is 0.
goang	The starting angle of the circular array.
DELANG	The minor word for indicating the delta angle between copies. The delta angle determines the angle between copies. For example, a delta angle of 30° places each copy 30° from the previous copy.
delang	The delta angle between copies.

Parameter	Description
ENDANG	The minor word for indicating the ending angle of the circular array. The default is 360°.
endang	The ending angle of the circular array.
ARRDO	The minor word for indicating which entity to generate.
DONT	The minor word for indicating which entity not to generate.
element number	The number of the circular array element.

Example:

Program Statement	Explanation

`PAT1=ARRAY/PT1,CIRC,PT2,$`
`  RADIUS,.75,GOANG,15,$`
`  DELANG,45,ENDANG,195`
	Array PAT1 is a circular array with PT2 as the center point. Point PT1 is the element reproduced. The radius of the elements from the center point is 0.75 units of measure. The first element of the array is created 15° from the x positive axis. The last element of the array is at 195°. There are four elements in the array. The number of elements is determined by the equation:
	
	(ENDANG - GOANG) / DELANG
	
	In this example:
	
	(195° - 15°) / 45° = 4 elements
	
	The four elements are spaced 45° apart.

GROUP

The GROUP statement gathers all the components of a configuration into a logical unit, which is then treated as an entity. Entity types point, line, arc, conic, spline, and group are allowed. Groups can contain up to seven levels. No group can contain more than 240 entities.

By Entity Names

Statement format:

```
GROUP/name[,...]
```

Parameter	Description
name	The names of the entities to be defined as a group.

Example:

Program Statement	Explanation
`GRP1=GROUP/PT1,AR3,LN5`	Group GRP1 is composed of point PT1, array AR3, and line LN5.
`GRP24=GROUP/CIR1,CIR2,$` `L1,L2,L3,L4,L5,L6,L7,L8`	Group GRP24 is composed of circles CIR1 and CIR2 and lines L1, L2, L3, L4, L5, L6, L7, and L8.

By Entity Array

Statement format:

```
GROUP/NUMBER,number,entity array
```

Parameter	Description
NUMBER	The minor word for specifying a number of entities from an entity array.
number	The number of entities from the entity array.
entity array	The name of the entity array.

NOTE

Individual entities cannot be manipulated if they are in a group. The GROUP status must first be deleted (DELETE/group name).

MIRROR

The MIRROR statement copies a number of entities reflected about an axis. Entities can be entered using entity names or using an entity array. This axis can be the x-axis, y-axis, or any defined line. If more than one entity is reflected, the entities are defined as a group. A maximum of 240 entities can be mirrored. Entity types point, line, arc, conic, spline, and point set are allowed. A maximum of 64 entity names can be explicitly listed. If more are required, an entity array should be used.

By Entity Names

Statement format:

```
        ⎧line  ⎫
MIRROR/⎨XAXIS ⎬,entity[,...]
        ⎩YAXIS ⎭
```

Parameter	Description
line	The line to be the axis of mirroring.
XAXIS	The x-axis to be the axis of mirroring.
YAXIS	The y-axis to be the axis of mirroring.
entity	The name of the entity to be mirrored.

Example:

Program Statement	Explanation
`PLT2=MIRROR/YAXIS,LN1,$` `  LN2,LN3,LN4`	A mirrored entity PLT2 is created by reflecting entities LN1, LN2, LN3, and LN4 about the y-axis.

By Entity Array

Statement format:

```
        ⎧line  ⎫
MIRROR/⎨XAXIS ⎬,NUMBER,number,entity array
        ⎩YAXIS ⎭
```

Parameter	Description
line	The line to be the axis of mirroring.
XAXIS	The x-axis to be the axis of mirroring.
YAXIS	The y-axis to be the axis of mirroring.
NUMBER	The minor word for specifying a number of entities from an entity array.
number	The number of entities from the entity array.
entity array	The name of the entity array.

MODIFY

The MODIFY statement can modify the following items:

- The coordinates of a point, line, or drafting entity
- An entity pen, level, color, or font number
- The radius, start or end angle, or center of a circle

Point and Line Modifications

Statement format:

```
MODIFY/entity,COORD,number,array
```

Parameter	Description
entity	The name of the entity to be modified (point or line).
COORD	The minor word for modifying coordinates.
number	The number of coordinates to be modified.
array	The name of the array in which the data is to be stored. The size of the array (the number of words required) is dependent upon the entity type selected. The number of elements for each entity is:

Entity Type	Number of Elements	Explanation
Points	3	Points require three elements per entity to store the coordinates of each point.
Lines	6	Lines require six elements per entity to store the coordinates of each line.

NOTE

For point, line, and circle entities, you can use this statement only if the entities were created with the first definition form. For drafting entities, you can use this statement only in the same view in which those entities were created.

Pen, Level, Color, and Font Modifications

Statement format:

```
                  [PENNUM]
MODIFY/entity,    [LEVNUM] ,number
                  [COLNUM]
                  [FONNUM]
```

Parameter	Description
entity	The name of the entity being modified.
PENNUM	The minor word for modifying the pen number.
LEVNUM	The minor word for modifying the level number.
COLNUM	The minor word for modifying the color number.
FONNUM	The minor word for modifying the font number.
number	The new value for the pen, level, font, or color.

Pen (0-15) and level (0-1023) numbers default to 0 or are assigned by the user. Color and font have preassigned default numbers as follows:

Font Number	Font
0	Solid
1	Dashed
2	Phantom
3	Centerline

Color Number	Color
0	White
1	Red
2	Geeen
3	Blue-cyan
4	Magenta
5	Yellow
6	Orange
7	Light orange
8	Light green
9	Dark Cyan
10	Green-blue
11	Dark red
12	Light purple
13	Yellow-green
14	Dark pink

Circle Modifications

Statement format:

```
MODIFY/circle [CENTER,x value,y value,z value][,RADIUS,radius]
[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
circle	The name of the circle to be modified.
CENTER	The minor word for modifying the center.
x value	The new x value.
y value	The new y value.
z value	The new z value.
RADIUS	The minor word for modifying the radius.
radius	The radius in units of measure.
GOANG	The minor word for the starting angle.
goang	The starting angle in degrees.
ENDANG	The minor word for the ending angle.
endang	The ending angle in degrees.

NOTE

You must specify at least one of the options (RADIUS, GOANG, or ENDANG).

PROJEC

The PROJEC statement projects a two-dimensional geometric configuration normal to the work plane. The projected configuration is automatically duplicated at the delta depth specified, and connecting lines between the endpoints are created. The result of this statement will be a group if an entity name is assigned to the statement. Allowable projection entities are lines, arcs, general conics, two-dimensional splines, and three-dimensional splines.

Defining by Single Entities

Statement format:

```
PROJEC/entity1,entity2,...,entityn,delta depth
```

Parameter	Description
entity1, entity2, ⋮ entityn	The names of the entities to be projected. n can be: $0 < n \leq 64$
delta depth	The delta depth projected normal to the plane of definition.

Example:

Program Statement	Explanation
`PROJEC/LINA,LINB,LINC,3`	An entity is created and projected normal to the plane of definition using entities LINA, LINB, and LINC for a depth of 3 units of measure.

The following message occurs if the number of entities exceeds 64.

```
ONLY 64 ENTITIES ALLOWED
```

The following message occurs if the number of entities is less than or equal to zero.

```
AT LEAST ONE ENTITY IS REQUIRED
```

Defining with an Array of Entities

Statement format:

```
PROJEC/NUMBER,number,entity array,delta depth
```

Parameter	Description
NUMBER	The minor word for defining a projection by an array of entities.
number	The number of entities from the array.
entity array	The name of an array of entity pointers.
delta depth	The delta depth projected normal to the plane of definition.

If the number of entities exceeds 64 or is less than or equal to 0, one of the following error messages is displayed:

```
ONLY 64 ENTITIES ALLOWED

AT LEAST ONE ENTITY IS REQUIRED
```

Example:

Program Statement	Explanation
`PROJEC/NUMBER,22,BOX(1),2`	A projected entity is created using 22 entities in array BOX for a depth of 2 units of measure.

RTRIEV

The RTRIEV statement retrieves a pattern from the local pattern library (PATERN). A pattern is defined as a group if the minor word YES is used or the pattern is named (PA=RTRIEV).

Retrieving a Two-Dimensional Pattern

Statement format:

```
RTRIEV/PATERN,('name' | text variable),(point | xcoord,ycoord,zcoord)

    [,SCALE,scale][,ANGLE,angle][,YES],status
```

Parameter	Description
PATERN	The minor word for retrieving a pattern from the local pattern library.
'name'	The name of the pattern enclosed in single quotes (64 characters maximum).
text variable	The name of the text variable that contains the name of the pattern (64 characters maximum).
point	The name of the point that locates the pattern in the part drawing.
xcoord	The x-coordinate of the point that locates the pattern in the part drawing.
ycoord	The y-coordinate of the point that locates the pattern in the part drawing.
zcoord	The z-coordinate of the point that locates the pattern in the part drawing.
SCALE	The minor word for indicating the scale factor of the pattern.
scale	The scale factor of the pattern. The default is 1.0.
ANGLE	The minor word for indicating the rotation angle of the pattern.
angle	The rotation angle of the pattern. The default is 0.0°.
YES	The minor word that defines the pattern as a group.

Parameter	Description
status	The name of the variable that is to receive the retrieval status.

Value	Description
0	Retrieval was completed.
-1	The pattern named was not found.
-2	Not applicable to two-dimensional patterns.
-3	Some or all entities could not be grouped.

Retrieving a Three-Dimensional Pattern

Statement format:

```
RTRIEV/PATERN,{ 'name'        },{ point                 },
              {text variable  } {xcoord,ycoord,zcoord   }

    COORD,point1,point2,point3[,SCALE,scale][,ANGLE,angle][,YES],status
```

Parameter	Description
PATERN	The minor word for retrieving a pattern from the local pattern library.
'name'	The name of the pattern enclosed in single quotes (64 characters maximum).
text variable	The name of the text variable that contains the name of the pattern (64 characters maximum).
point	The name of the point that locates the pattern in the part drawing.
xcoord	The x-coordinate of the point that locates the pattern in the part drawing.
ycoord	The y-coordinate of the point that locates the pattern in the part drawing.
zcoord	The z-coordinate of the point that locates the pattern in the part drawing.
COORD	The minor word for locating a pattern using three noncollinear points.
point1,point2,point3	The names of the three noncollinear points that locate the pattern.
SCALE	The minor word for indicating the scale factor of the pattern.
scale	The scale factor of the pattern. The default is 1.0.
YES	The minor word that defines the pattern as a group.
ANGLE	The minor word for indicating the rotation angle of the pattern.
angle	The rotation angle of the pattern. The default is 0.0°.

Parameter	Description
status	The name of the variable that is to receive the retrieval status.

Value	Description
0	Retrieval was completed.
-1	The pattern named was not found.
-2	The points given are collinear (three-dimensional only).
-3	Some or all entities could not be grouped.
-4	The maximum number of views was exceeded.
-5	At least one new view was created.

TEMPLT

The TEMPLT statement creates a master and instance template. The template is then stored in the UTF. The template entities are defined:

- In the UTF as a UTF template.
- As a master template in the current part.
- As a template instance in the current part.

The templates are defined by using entity names or an entity array, or by creating an instance. Allowable entities are points, lines, arcs, and pointsets.

NOTE

If a pointer is returned by the statement, it is an instance pointer.

By Entity Names

Statement format:

```
TEMPLT/CREATE,'template name',name[,...][,status]
```

Parameter	Description
CREATE	The minor word indicating template creation.
'template name'	The name to be given to the UTF template. The UTF name is carried with both the master template and the template instance.
name	The names of the entities that are to make up the template.
status	The name of a variable that is to receive the status of the template operation.

Value	Description
0	No error was found.
1	The CREATE option overwrote a master of the same name.

Example:

Program Statement	Explanation

`TEM1=TEMPLT/CREATE,$`
`'GROMMET',LN004,LN005,$`
`CR006,SP009,LN012,LN020`
	Template TEM1 is created with the name GROMMET. It is made up of lines LN004 and LN005, circle CR006, spline SP009, and lines LN012 and LN020.

By Entity Arrays

Statement format:

```
TEMPLT/CREATE,'template name',NUMBER,number,entity array[,status]
```

Parameter	Description
CREATE	The minor word indicating template creation.
'template name'	The name to be given to the UTF template. The UTF name is carried with both the master template and the template instance.
NUMBER	The minor word for specifying a number of entities from an entity array.
number	The number of entities from the entity array.
entity array	The name of the entity array.
status	The name of a variable that is to receive the status of the template operation.

Value	Description
0	No error was found.
1	The CREATE option overwrote a master of the same name.

As An Instance

Statement format:

```
                                    ⎧NORTH,WEST⎫
                                    ⎪SOUTH,WEST⎪
TEMPLT/INSTNC,'template name',      ⎪CENTER    ⎪,{ point    }
                                    ⎪NORTH,EAST⎪  {xt,yt,zt}
                                    ⎩SOUTH,EAST⎭

    [,SCALE,scale][,ANGLE,angle][,status]
```

Parameter	Description
INSTNC	The minor word indicating template instance.
'template name'	The name to be given to the UTF template. The UTF name is carried with both the master template and the template instance.
NORTH,WEST	The minor words for locating the instance in the upper left of the origin point.
SOUTH,WEST	The minor words for locating the instance in the lower left of the origin point.
CENTER	The minor words for locating the instance in the center of the origin point.
NORTH,EAST	The minor words for locating the instance in the upper right of the origin point.
SOUTH,EAST	The minor words for locating the instance in the lower right of the origin point.
point	The name of the origin point.
xt,yt,zt	The xt-, yt-, and zt-coordinates of the origin point.
SCALE	The minor word for indicating the scale factor of the pattern.
scale	The scale factor of the pattern. The default is 1.0.
ANGLE	The minor word for indicating the rotation angle of the pattern.
angle	The rotation angle of the pattern. The default is 0.0°.
status	The name of the variable to receive the status of the template operation.

Value	Description
0	No error was found.
2	No master was found for TEMPLT/INSTNC.
3	Unlinked master was found for TEMPLT/INSTNC.

Three-Dimensional Curve Statements 15

Three-Dimensional Curve Statements 15

CMPCRV

The CMPCRV statement creates a composite curve, that is, an ordered set of contiguous curves (lines, arcs, conics, and splines). The limit of curves allowed in a composite curve is 100.

Statement format:

```
CMPCRV/( curve[,...]                    )
       (NUMBER,number,curve array       )
```

Parameter	Description
curve	The names of the curves to be used in creating a composite curve.
NUMBER	The minor word for specifying a number of curves from a curve array.
number	The number of curves from the curve array.
curve array	The name of the curve array.

Example:

Program Statement	Explanation
`CCUR1=CMPCRV/L445,C245,$` `SPL9,L124`	Composite curve CCUR1 is created using line L445, circle C245, spline SPL9, and line L124.

MACCRV

The MACCRV statement creates a machining curve, that is, a three-dimensional curve implied by a set of three-dimensional points. The default tolerance is 0.1 regardless of the unit of measure used. The user input tolerance must be greater than 0.

Surface Edge Curve

Statement format:

```
MACCRV/EDGE,surface,point[,NUMBER,approx pts]
```

Parameter	Description
EDGE	The minor word for defining by the surface edge curve.
surface	The name of the surface whose edge is intersected.
point	The name of the point indicating the edge.
NUMBER	The minor word indicating the minimum number of intersection points to be used to approximate the curve.
approx pts	The number of intersection points.

Surface Intersection Curve

Statement format:

```
MACCRV/INTOF,drive surface,check surface[,tolerance,step],{{NORMAL},point}
                                                          {{PIERCE}
                                                          {     EDGE  }
```

Parameter	Description
INTOF	The minor word indicating the definition of an intersection curve.
drive surface	The name of the surface to be used to create the machining curve.
check surface	The name of the surface at which the machining curve is to be created.
tolerance	The tolerance factor in units of measure. The number of intersection points is translated into a tolerance using the following scheme:

Points	Tolerance
<20	0.2540 mm (0.0100 in)
>20	0.1270 mm (0.0050 in)
>30	0.0254 mm (0.0010 in)
>50	0.0127 mm (0.0005 in)

step	The step size in units of measure.
NORMAL	The minor word for definition using a normal point.
PIERCE	The minor word for definition using a pierce point.
point	The name of the starting point of the intersection curve.
EDGE	The minor word for definition using an edge intersection. If the result is more than one machining curve, a group is defined. The member of that group can be obtained using the OBTAIN/MEMBER statement.

Draft Curve to a Surface

Statement format:

```
MACCRV/( curve[,...]                  ),surface[,ANGLE,angle,point][,tolerance]
       (NUMBER,number,curve array)
```

Parameter	Description
curve	The name or names of the curves that are to be used to create the machining curves.
NUMBER	The minor word for specifying a number of curves from a curve array.
number	The number of curves from the curve array.
curve array	The name of the curve array.
surface	The name of the surface at which the machining curve is to be created.
ANGLE	The minor word for indicating the projection angle at which the machining/draft curve is to be created.
angle	The projection angle of the machining curve in degrees and fractions of a degree. The angle is measured from the curve projected normal to the surface to that angle. The default angle is 0.0°. The bounds for angle input are 0.0° up to but not including 90°.
point	The name of the starting point of the draft curve.
tolerance	The tolerance factor of the curve in units of measure.

Draft Curve to a Depth

Statement format:

```
MACCRV/( curve[,...]          ),CURDEP,depth,ANGLE,angle,point[,tolerance]
       (NUMBER,number,curve array)
```

Parameter	Description
curve	The name or names of the curves that are to create the machining curves.
NUMBER	The minor word for specifying a number of curves from a curve array.
number	The number of curves from the curve array.
curve array	The name of the curve array.
CURDEP	The minor word for creating a curve at a depth.
depth	The depth at which the machine curve is created in units of measure.
ANGLE	The minor word for indicating the projection angle at which the machining/draft curve is created.
angle	The projection angle of the machining curve in degrees and fractions of a degree. The angle is measured from the curve projected normal to the surface to that angle. The default angle is 0.0°. The bounds for angle input are 0.0° up to but not including 90°.
point	The name of the starting point of the draft curve.
tolerance	The tolerance factor of the curve in units of measure.

SPCURV

The SPCURV statement creates a three-dimensional spline curve, that is, a three-dimensional curve that passes through a number of specified points. If you do not specify a start or end condition, the condition is defaulted to relaxed.

Global and Local Methods

Statement format:

```
                 {        point[,...]          } [  {{TANTO,line1}[ {TANTO,line2}]}]
SPCURV/GLOBAL,   {number,{entity array     }   } [, {{RELAX      }[,{RELAX      }]}]
                 {       {coordinate array }   } [  {          CYCLIC             }]
                                                 [  {          ACYCLC             }]
```

Parameter	Description
GLOBAL	The minor word for defining a spline curve with the global method, which is the second continuous derivative form.
point	The names of the points that are to define the spline curve.
number	The number of entities in the entity array.
entity array	The name of the entity array.
coordinate array	The name of the coordinate array.
TANTO	The minor word for defining the start or end condition of the spline tangent to a given line. The direction of the given lines is used to calculate the start and end conditions.
line	The name of the line.
RELAX	The minor word that defines the start or end condition of the spline as relaxed.
CYCLIC	The minor word indicating that the endpoint has the identical slope of the start point.
ACYCLC	The minor word indicating that the endpoint has the opposite slope of the start point.

Filter Method

Statement format:

```
SPCURV/FILTER,number of points,{ point array     }tolerance[,output tolerance]
                               {coordinate array}
```

Parameter	Description
FILTER	The minor word for defining a spline curve with the filter method.
number of points	The number of points in the point or coordinate array. The maximum number of points is 1000.
point array	The name of the point array.
coordinate array	The name of the coordinate array. The coordinates must be given in model space.
tolerance	The tolerance of the curve.
output tolerance	The output tolerance of the curve.

VECTOR

The VECTOR statement creates a vector, that is, single-line segment that has direction and magnitude in three-dimensional space.

Coordinates

Statement format:

```
VECTOR/xcoord1,ycoord1,zcoord1,xcoord2,ycoord2,zcoord2
```

Parameter	Description
xcoord1	The x-coordinate of the vector start point.
ycoord1	The y-coordinate of the vector start point.
zcoord1	The z-coordinate of the vector start point.
xcoord2	The x-coordinate of the vector endpoint.
ycoord2	The y-coordinate of the vector endpoint.
zcoord2	The z-coordinate of the vector endpoint.

Example:

Program Statement	Explanation
`V344=VECTOR/0,0,1,2,2,3.5`	Vector V344 is created with coordinates x=0, y=0, and z=1 as the start of the vector, and x=2, y=2, and z=3.5 as the end of the vector.

Two Points

Statement format:

```
VECTOR/POINTS,point1,point2
```

Parameter	Description
POINTS	The minor word indicating definition by two points.
point1	The name of the first vector point.
point2	The name of the second vector point.

Example:

Program Statement	Explanation
`VEC1=VECTOR/POINTS,PT1,PT2`	Vector VEC1 is created with point PT1 as the start of the vector and point PT2 as the end of the vector.

Scalar Times Existing Vector

Statement format:

```
VECTOR/vector,SCALAR,scalar
```

Parameter	Description
vector	The name of the existing vector.
SCALAR	The minor word indicating definition by scalar times existing vector.
scalar	The scalar of the vector in inches or millimeters.

Example:

Program Statement	Explanation
V501=VECTOR/VEC1,SCALAR,2	Vector V501 is created as the product of the reference vector, VEC1, and a scalar of 2.

Cross Product of Two Vectors

Statement format:

```
VECTOR/vector1,vector2
```

Parameter	Description
vector1	The name of the first vector.
vector2	The name of the second vector.

Example:

Program Statement	Explanation
V225=VECTOR/VEC1,V344	Vector V225 is created as the cross product of vector VEC1 and V344.

Unit Normal of Existing Vector

Statement format:

```
VECTOR/UNIT,vector
```

Parameter	Description
UNIT	The minor word indicating the definition of the unit normal of an existing vector.
vector	The name of the reference vector.

Example:

Program Statement	Explanation
VEC2=VECTOR/UNIT,VEC1	Vector VEC2 is created as a unit normal vector of vector VEC1.

Surface Unit Normal

Statement format:

```
VECTOR/NORMAL,surface,point
```

Parameter	Description
NORMAL	The minor word indicating the surface unit normal vector.
surface	The name of a surface.
point	The name of a point on the surface.

Through a Point at a Given Length and Angle

Statement format:

```
VECTOR/point,ANGLE,angle,LENGTH,length
```

Parameter	Description
point	The name of the point.
ANGLE	The minor word for indicating the angle of the vector from the horizontal axis.
angle	The angle of the vector from the horizontal axis in degrees and fractions of degrees.
LENGTH	The minor word for indicating the length of the vector from the reference point.
length	The length of the vector in units of measure.

Example:

Program Statement	Explanation
`V905=VECTOR/PT2,ANGLE,30,$` `  LENGTH,1`	Vector V905 is created from point PT2 at an angle of 30° to the horizontal axis for a distance of 1 unit of measure.

Intersection of Two Planes

Statement format:

```
VECTOR/INTOF,plane1,plane2
```

Parameter	Description
INTOF	The minor word indicating that the vector is to be defined as the intersection of two nonparallel planes. The order in which the planes are selected determines the direction of the vector.
plane1	The name of the first plane.
plane2	The name of the second plane.

Example:

Program Statement	Explanation
`V405=VECTOR/INTOF,PL22,PL23`	Vector V405 is created at the intersection of planes PL22 and PL23.

Sum of Two Vectors

Statement format:

```
VECTOR/VECSUM,vector1,vector2
```

Parameter	Description
VECSUM	The minor word indicating that a vector is to be defined as the sum of two vectors.
vector1	The name of the first vector.
vector2	The name of the second vector.

Example:

Program Statement	Explanation
`VEC3=VECTOR/VECSUM,VEC1,VEC2`	Vector VEC3 is created as the vector sum of vectors VEC1 and VEC2.

Difference of Two Vectors

Statement format:

```
VECTOR/DIFFER,vector1,vector2
```

Parameter	Description
DIFFER	The minor word indicating a vector defined as the difference of two vectors.
vector1	The name of the first vector.
vector2	The name of the second vector.

Example:

Program Statement	Explanation
`V299=VECTOR/DIFFER,VEC2,VEC3`	Vector V299 is created as the vector difference of vectors VEC2 and VEC3.

Through a Point at an Angle with a Line or Vector at a Length

Statement format:

```
VECTOR/point,vector,ATANGL,angle,LENGTH,length
```

Parameter	Description
point	The name of the reference point.
vector	The name of the reference line or vector.
ATANGL	The minor word for indicating the angle of the vector to the reference line or vector.
angle	The angle of the vector in degrees or fractions of degrees.
LENGTH	The minor word for indicating the length of the vector.
length	The length of the vector in units of measure.

Example:

Program Statement	Explanation
`V32=VECTOR/P240,V344,$` `  ATANGL,30,LENGTH,10`	Vector V32 is created with point P240 as the tail of the new vector at an angle of 30° to vector V344. The new is 10 units of measure in length.

Surface Statements 16

Surface Statements 16

Surface statements create three-dimensional surfaces. Surface definitions are equivalent to types and forms found in menu 15 EXTENDED GEOMETRY in the ICEM Design/Drafting Advanced Design manual.

CMSRF

The CMSRF statement creates a curve mesh surface, that is, a generalized surface that meshes a family of variable curves with a family of opposing (but not necessarily orthogonal) fixed curves. No more than 30 fixed curves (nf) and 30 variable curves (nv) can be specified. The number of fixed curves times the number of variable curves is limited to 341.

Statement format:

```
CMSRF/NFIXED,nf,{fcurve[,...]},NVARY,nv,{vcurve[,...]}
                {fcurve array}         {vcurve array}
```

Parameter	Description
NFIXED	The minor word for indicating fixed curves.
fcurve	The names of the fixed curves (maximum is 30).
fcurve array	The name of the fixed curve array.
nf	The number of fixed curves from the array (maximum is 30).
NVARY	The minor word for indicating variable curves.
vcurve	The names of the variable curves (maximum is 30).
vcurve array	The name of the variable curve array.
nv	The number of variable curves from the array (maximum is 30).

Example:

Program Statement	Explanation

`CSF44=CMSRF/NFIXED,2,$`
`CRV1,CRV2,$`
`NVARY,2,CRV3,CRV4`
	Curve mesh surface CSF44 is created. The surface is bounded by two fixed curves, CRV1 and CRV2, and by two variable curves, CRV3 and CRV4.

CONE

The CONE statement creates a cone, that is, a surface generated by revolving a line about an axis that lies in the same plane.

Line

Statement format:

```
CONE/line,HANGLE,half-angle
```

Parameter	Description
line	The name of a line representing the cone axis.
HANGLE	The minor word for indicating the half-angle of the cone. The half-angle is the angle of the surface to the axis of the cone.
half-angle	The half-angle in degrees and fractions of degrees.

Example:

Program Statement	Explanation
`CN43=CONE/L344,HANGLE,25`	Cone CN43 is created around line L344 with a half-angle of 25°.

Coordinates

Statement format:

```
CONE/xcoord1,ycoord1,zcoord1,xcoord2,ycoord2,zcoord2,HANGLE,half-angle
```

Parameter	Description
xcoord1	The x-coordinate of the first cone axis endpoint.
ycoord1	The y-coordinate of the first cone axis endpoint.
zcoord1	The z-coordinate of the first cone axis endpoint.
xcoord2	The x-coordinate of the second cone axis endpoint.
ycoord2	The y-coordinate of the second cone axis endpoint.
zcoord2	The z-coordinate of the second cone axis endpoint.
HANGLE	The minor word for indicating the half-angle of the cone. The half-angle is the angle of the surface to the centerline of the cone.
half-angle	The half-angle in degrees and fractions of degrees.

Example:

Program Statement	Explanation
`CN52=CONE/3,4,20,12,4,20,$` `HANGLE,15`	Cone CN52 is created around a line that extends from coordinates (3,4,20) to coordinates (12,4,20) with a half-angle of 15°.

Two Points

Statement format:

```
CONE/POINTS,point1,point2,HANGLE,half-angle
```

Parameter	Description
POINTS	The minor word for indicating two endpoints of the cone axis.
point1	The name of the first cone axis point.
point2	The name of the second cone axis point.
HANGLE	The minor word for indicating the half-angle of the cone. The half-angle is the angle of the surface to the axis of the cone.
half-angle	The half-angle in degrees and fractions of degrees.

Example:

Program Statement	Explanation
`CN63=CONE/POINTS,P33,P39,$` `  HANGLE,30`	Cone CN63 is created around an axis that extends from point P33 to point P39 with a half-angle of 30°.

Point and Delta Coordinates

Statement format:

```
CONE/point,DELTA,deltax,deltay,deltaz,HANGLE,half-angle
```

Parameter	Description
point	The name of the reference point.
DELTA	The minor word for indicating the delta coordinates.
deltax	The delta x-coordinate of the axis endpoint.
deltay	The delta y-coordinate of the axis endpoint.
deltaz	The delta z-coordinate of the axis endpoint.
HANGLE	The minor word indicating the half-angle of the cone. The half-angle is the angle of the surface to the axis of the cone.
half-angle	The half-angle in degrees and fractions of degrees.

Example:

Program Statement	Explanation

`CN89=CONE/P132,$`
`  DELTA,0,12,24,$`
`  HANGLE,12.5`
	Cone CN89 is created around an axis that extends from point P132 to delta coordinates (0,12,24) with a half-angle of 12.5°.

CYLNDR

The CYLNDR statement creates a cylinder, that is, a surface generated by revolving a line about a parallel axis.

Line

Statement format:

```
CYLNDR/line,RADIUS,radius[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
line	The name of the line representing the axis of the cylinder.
RADIUS	The minor word for indicating the cylinder radius.
radius	The radius in units of measure.
GOANG	The minor word for indicating the start angle of the cylindrical surface.
goang	The start angle of the surface in degrees or fractions of degrees. The default is 0°.
ENDANG	The minor word for indicating the end angle of the cylindrical surface.
endang	The end angle of the surface in degrees or fractions of degrees. The default is 360°.

Example:

Program Statement	Explanation

`CY23=CYLNDR/L342,$`
`  RADIUS,4.25,$`
`  GOANG,180,ENDANG,270`
	Cylinder CY23 is created around line L342 at a radius of 4.25 units of measure. The start angle is 180°, and the end angle is 270°.

Coordinates

Statement format:

```
CYLNDR/xcoord1,ycoord1,zcoord1,xcoord2,ycoord2,zcoord2,RADIUS,radius
  [,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
xcoord1	The x-coordinate of the first cylinder axis endpoint.
ycoord1	The y-coordinate of the first cylinder axis endpoint.
zcoord1	The z-coordinate of the first cylinder axis endpoint.
xcoord2	The x-coordinate of the second cylinder axis endpoint.
ycoord2	The y-coordinate of the second cylinder axis endpoint.
zcoord2	The z-coordinate of the second cylinder axis endpoint.
RADIUS	The minor word for indicating the cylinder radius.
radius	The radius in units of measure.
GOANG	The minor word for indicating the start angle of the cylindrical surface.
goang	The start angle of the surface in degrees or fractions of degrees. The default is 0°.
ENDANG	The minor word for indicating the end angle of the cylindrical surface.
endang	The end angle of the surface in degrees or fractions of degrees. The default is 360°.

Example:

Program Statement	Explanation

`CY46=CYLNDR/10,5,15,3,9,7.5,$`
`  RADIUS,39.5,$`
`  GOANG,90,ENDANG,180`
	Cylinder CY46 is created around an axis that extends from coordinates (10,5,15) to coordinates (3,9,7.5) at a radius of 39.5 units of measure. The start angle is 90° and the end angle is 180°.

Two Points

Statement format:

```
CYLNDR/POINTS,point1,point2,RADIUS,radius[,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
POINTS	The minor word for indicating the two cylinder axis endpoints.
point1	The name of the first cylinder axis endpoint.
point2	The name of the second cylinder axis endpoint.
RADIUS	The minor word for indicating the cylinder radius.
radius	The radius of the cylinder in units of measure.
GOANG	The minor word for indicating the start angle of the cylindrical surface.
goang	The start angle of the surface in degrees or fractions of degrees. The default is 0°.
ENDANG	The minor word for indicating the end angle of the cylindrical surface.
endang	The end angle of the surface in degrees or fractions of degrees. The default is 360°.

Example:

Program Statement	Explanation

`CY103=CYLNDR/P49,P64,$`
`  RADIUS,24.75,$`
`  GOANG,90,ENDANG,180`
	Cylinder CY103 is created around an axis that extends from point P49 to point P64 at a radius of 24.75 units of measure. The start angle is 90° and the end angle is 180°.

Point and Delta Coordinates

Statement format:

```
CYLNDR/point,DELTA,deltax,deltay,deltaz,RADIUS,radius
  [,GOANG,goang][,ENDANG,endang]
```

Parameter	Description
point	The name of the base reference point.
DELTA	The minor word for indicating the delta coordinates.
deltax	The delta x-coordinate of the axis endpoint.
deltay	The delta y-coordinate of the axis endpoint.
deltaz	The delta z-coordinate of the axis endpoint.
RADIUS	The minor word for indicating the cylinder radius.
radius	The radius of the cylinder in units of measure.
GOANG	The minor word for indicating the start angle of the cylindrical surface.
goang	The start angle of the surface in degrees or fractions of degrees. The default is 0°.
ENDANG	The minor word for indicating the end angle of the cylindrical surface.
endang	The end angle of the surface in degrees or fractions of degrees. The default is 360°.

Example:

Program Statement	Explanation

`CYL6=CYLNDR/P49,3,2,0,$`
`  RADIUS,10,$`
`  GOANG,90,ENDANG,180`
	Cylinder CYL6 is created around an axis that extends from point P49 to delta coordinates (3,2,0) at a radius distance of 10 units of measure. The start angle is 90° and the end angle is 180°.

DEVSRF

The DEVSRF statement creates a developable surface, that is, a special ruled surface whose corresponding points on the two generating curves have the same tangent value. Refer to menu 15.2.5 in the ICEM Design/Drafting Advanced Design manual for a complete description.

Statement format:

```
        ⎧XSMALL⎫                ⎧XSMALL⎫
DEVSRF/ ⎪XLARGE⎪ ,curve1, ⎪XLARGE⎪ ,curve2[,DEGTOL,degtol]
        ⎪YSMALL⎪                ⎪YSMALL⎪
        ⎪YLARGE⎪                ⎪YLARGE⎪
        ⎪ZSMALL⎪                ⎪ZSMALL⎪
        ⎩ZLARGE⎭                ⎩ZLARGE⎭
```

Parameter	Description
XSMALL	The minor word indicating the x negative direction.
XLARGE	The minor word indicating the x positive direction.
YSMALL	The minor word indicating the y negative direction.
YLARGE	The minor word indicating the y positive direction.
ZSMALL	The minor word indicating the z negative direction.
ZLARGE	The minor word indicating the z positive direction.
	The preceding minor words indicate which ends of curve1 and curve2 are to be connected.
curve1	The name of the first curve.
curve2	The name of the second curve.
DEGTOL	The minor word for indicating the tolerance factor of the surface.
degtol	The tolerance factor of the surface in degrees.

Example:

Program Statement	Explanation
`DV22=DEVSRF/XLARGE,ARC6,$` `  XLARGE,ARC7,DEGTOL,.005`	Developable surface DV22 is created by connecting the x positive end of arc ARC6 to the x positive end of arc ARC7 (the two opposite ends are also connected) with a surface tolerance of 0.005°.

FILSRF

The FILSRF statement creates a fillet surface, that is, a surface that provides a constant radius transition between two surfaces. Both surfaces cannot be infinite planes. The fillet surface follows the intersection of the two surfaces. For best results, the surfaces should not intersect along an edge.

The tolerance is used to determine the intersection path. Any tolerance greater than or equal to 0.005 mm (0.0002 in) is acceptable.

The starting point is used for two purposes. First, it is used to determine which quadrant formed by the two surfaces should contain the fillet. The point, therefore, should be slightly offset from each surface and within both boundaries. Second, it is used to define where the fillet should start. If an edge intersection of one surface with the other exists within one radius of the starting point, the fillet starts at this edge point. Otherwise, it starts at the closest surface intersection point to the starting point.

Statement format:

```
FILSRF/surface1,surface2,RADIUS,radius,tolerance,START,point
```

Parameter	Description
surface1	The name of the first surface at which the fillet is to be created.
surface2	The name of the second surface at which the fillet is to be created.
RADIUS	The minor word for indicating the radius of the fillet surface.
radius	The radius of the fillet surface in units of measure.
tolerance	The tolerance in degrees. The tolerance factor is used to approximate the intersection curve. The default tolerance is 0.127 mm (0.005 in). The tolerance must be greater than or equal to 0.005 mm (0.0002 in).
START	The minor word for indicating the start point of the fillet surface.
point	The name of the start point of the fillet surface.

Example:

Program Statement	Explanation

`FS1=FILSRF/S1,S2,$`
`RADIUS,1.5,$`
`0.05,START,P1`
	Fillet surface FS1 is created using surfaces S1 and S2. The fillet surface has a radius of 1.5 units of measure and a tolerance of 0.05°. The fillet surface starts at point P1.

HEXDRN

The HEXDRN statement creates a hexahedron, that is, a solid defined by orthogonal planes.

Coordinates

Statement format:

```
HEXDRN/xcoord,ycoord,zcoord,DELTAX,deltax,DELTAY,deltay,DELTAZ,deltaz
```

Parameter	Description
xcoord	The x-coordinate of the hexahedron center point.
ycoord	The y-coordinate of the hexahedron center point.
zcoord	The z-coordinate of the hexahedron center point.
DELTAX	The minor word for indicating the delta x-coordinate.
deltax	The delta x-coordinate to the center of the hexahedron face in inches or millimeters.
DELTAY	The minor word for indicating the delta y-coordinate.
deltay	The delta y-coordinate to the center of the hexahedron face in inches or millimeters.
DELTAZ	The minor word for indicating the delta z-coordinate.
deltaz	The delta z-coordinate to the center of the hexahedron face in inches or millimeters.

Example:

Program Statement	Explanation

`HEX22=HEXDRN/3.25,5,0,$`
`DELTAX,12.25,DELTAY,5,$`
`DELTAZ,0`
	Hexahedron HEX22 is created with the center point at coordinates (3.25,5,0) and the center of each face at delta coordinates (12.25,5,0).

Point

Statement format:

```
HEXDRN/point,DELTAX,deltax,DELTAY,deltay,DELTAZ,deltaz
```

Parameter	Description
point	The name of the hexahedron center point.
DELTAX	The minor word for indicating the delta x-coordinate.
deltax	The delta x-coordinate to the center of the hexahedron face in inches or millimeters.
DELTAY	The minor word for indicating the delta y-coordinate.
deltay	The delta y-coordinate to the center of the hexahedron face in inches or millimeters.
DELTAZ	The minor word for indicating the delta z-coordinate.
deltaz	The delta z-coordinate to the center of the hexahedron face in inches or millimeters.

LPSOID

The LPSOID statement creates an ellipsoid, that is, a solid defined by an ellipse rotated about its major axis.

Statement format:

```
LPSOID/ellipse
```

Parameter	Description
ellipse	The name of the ellipse used to create the ellipsoid.

Example:

Program Statement	Explanation
`LP30=LPSOID/ELP44`	Ellipsoid LP30 is created by rotating ellipse ELP44 about its major axis.

PLANE

The PLANE statement creates a plane, that is, a right-angled flat surface in three-dimensional space.

Coefficients

Statement format:

```
PLANE/a,b,c,d,CENTER,center,CORNER,corner
```

Parameter	Description
a	The first coefficient of the planar equation $Ax+By+Cz+D=0$.
b	The second coefficient of the planar equation $Ax+By+Cz+D=0$.
c	The third coefficient of the planar equation $Ax+By+Cz+D=0$.
d	The fourth coefficient of the planar equation $Ax+By+Cz+D=0$.
CENTER	The minor word for indicating the center of the plane.
center	The name of the center point of the plane.
CORNER	The minor word for indicating the corner of the plane.
corner	The name of the corner point of the plane.

Example:

Program Statement	Explanation
`PL33=PLANE/4,5,9,10,$` `  CENTER,P29,CORNER,P30`	Plane PL33 is created using the planar equation where $A=4$, $B=5$, $C=9$, and $D=10$. The center point is P29 and the corner point is P30.

Three Noncollinear Points

Statement format:

```
PLANE/POINTS,point1,point2,point3,CENTER,center,CORNER,corner
```

Parameter	Description
POINTS	The minor word indicating a plane defined by points.
point1,point2, point3	The names of the points defining the plane.
CENTER	The minor word for indicating the center of the plane.
center	The name of the center point of the plane.
CORNER	The minor word for indicating the corner of the plane.
corner	The name of the corner point of the plane.

Example:

Program Statement	Explanation

`PL34=PLANE/POINTS,P55,$`
`P60,P65,$`
`CENTER,P31,CORNER,P32`
	Plane PL34 is created using points P55, P60, and P65 to define the plane. The center point is P31 and the corner point is P32.

Through a Point and Parallel to a Plane

Statement format:

```
PLANE/point,PARLEL,plane,CENTER,center,CORNER,corner
```

Parameter	Description
point	The name of the reference point.
PARLEL	The minor word indicating a plane parallel to a plane.
plane	The name of the reference plane.
CENTER	The minor word for indicating the center of the plane.
center	The name of the center point of the plane.
CORNER	The minor word for indicating the corner of the plane.
corner	The name of the corner point of the plane.

Example:

Program Statement	Explanation
`PL45=PLANE/P39,PARLEL,PL61,$` `CENTER,P109,CORNER,P110`	Plane PL45 is created parallel to PL61 through point P39. The center point is P109 and the corner point is P110.

Parallel to a Plane at a Distance

Statement format:

```
PLANE/PARLEL,plane,DISTNC,distance,CENTER,center,CORNER,corner
```

Parameter	Description
PARLEL	The minor word indicating a plane parallel to a plane.
plane	The name of the reference plane.
DISTNC	The minor word for indicating the distance from the reference plane.
distance	The distance from the reference plane in units of measure.
CENTER	The minor word for indicating the center of the plane.
center	The name of the center point of the plane.
CORNER	The minor word for indicating the corner of the plane.
corner	The name of the corner point of the plane.

Example:

Program Statement	Explanation

`PL46=PLANE/PARLEL,PL62,$`
`DISTNC,5.25$`
`CENTER,P109,CORNER,P110`
	Plane PL46 is created parallel to PL62 at a distance of 5.25 units of measure. The center point is P109 and the corner point is P110.

Through a Point and Perpendicular to an Existing Vector

Statement format:

```
PLANE/point,PERPTO,vector,CENTER,center,CORNER,corner
```

Parameter	Description
point	The name of the reference point.
PERPTO	The minor word indicating perpendicular to an existing vector.
vector	The name of the reference vector.
CENTER	The minor word for indicating the center of the plane.
center	The name of the center point of the plane.
CORNER	The minor word for indicating the corner of the plane.
corner	The name of the corner point of the plane.

Example:

Program Statement	Explanation
`PL47=PLANE/P38,PERPTO,V62,$` `CENTER,P109,CORNER,P110`	Plane PL47 is created through point P38 and perpendicular to vector V62. The center point is P109 and the corner point is P110.

Two Points and Perpendicular to a Plane

Statement format:

```
PLANE/POINTS,point1,point2,PERPTO,plane,CENTER,center,CORNER,corner
```

Parameter	Description
POINTS	The minor word indicating a plane defined by points.
point1,point2	The names of the points defining the plane.
PERPTO	The minor word indicating perpendicular to a plane.
plane	The name of the reference plane.
CENTER	The minor word for indicating the center of the plane.
center	The name of the center point of the plane.
CORNER	The minor word for indicating the corner of the plane.
corner	The name of the corner point of the plane.

Example:

Program Statement	Explanation

`PL48=PLANE/POINTS,P38,$`
`P62,PERPTO,PL23,$`
`CENTER,P109,CORNER,P110`
	Plane PL48 is created through points P38 and P62 and perpendicular to plane PL23. The center point is P109 and the corner point is P110.

Point and Perpendicular to Two Planes

Statement format:

```
PLANE/PERPTO,plane1,plane2,point,CENTER,center,CORNER,corner
```

Parameter	Description
PERPTO	The minor word indicating perpendicular to two planes.
plane1,plane2	The names of the planes defining the perpendicularity.
point	The name of the reference point.
CENTER	The minor word for indicating the center of the plane.
center	The name of the center point of the plane.
CORNER	The minor word for indicating the corner of the plane.
corner	The name of the corner point of the plane.

Example:

Program Statement	Explanation

`PL49=PLANE/PERPTO,PL48,$`
`PL50,P38,CENTER,P109,$`
`CORNER,P110`
	Plane PL49 is created perpendicular to planes PL48 and PL50 through point P38. The center point is P109 and the corner point is P110.

Two Lines

Statement format:

```
PLANE/line,line,CENTER,center,CORNER,corner
```

Parameter	Description
line	The names of the lines defining the plane.
CENTER	The minor word for indicating the center of the plane.
center	The name of the center point of the plane.
CORNER	The minor word for indicating the corner of the plane.
corner	The name of the corner point of the plane.

Example:

Program Statement	Explanation
`PL50=PLANE/L35,L36,$` `CENTER,P44,CORNER,P55`	Plane PL50 is created between lines L35 and L36. The center point is P44 and the corner point is P55.

REVSRF

The REVSRF statement creates a surface of revolution, that is, a surface generated by revolving a two-dimensional curve about a line at specified angles.

Statement format:

```
                  (XSMALL )
                  (XLARGE )
REVSRF/curve,     (YSMALL ) ,line[,GOANG,goang][,ENDANG,endang]
                  (YLARGE )
                  (ZSMALL )
                  (ZLARGE )
```

Parameter	Description
curve	The name of the curve that creates the surface of revolution.
XSMALL	The minor word indicating the x negative direction.
XLARGE	The minor word indicating the x positive direction.
YSMALL	The minor word indicating the y negative direction.
YLARGE	The minor word indicating the y positive direction.
ZSMALL	The minor word indicating the z negative direction.
ZLARGE	The minor word indicating the z positive direction.
	The preceding minor words indicate which end of the curve to use as a reference in determining the start and end angles.
line	The name of the line representing the axis at revolution.
GOANG	The minor word for indicating the start angle of the surface of revolution.
goang	The start angle of the surface in degrees and fractions of degrees. The default is 0°.
ENDANG	The minor word for indicating the end angle of the surface of revolution.
endang	The end angle of the surface in degrees and fractions of degrees. The default is 360°.

Example:

Program Statement	

`RS45=REVSRF/ARC3,XSMALL,$`
`  LINE2,GOANG,220,$`
`  ENDANG,270`
	A surface of revolution is created by revolving arc ARC3 around line LINE2. The starting angle of the surface is 220° and the ending angle is 270°. The angles are determined using the x negative end of ARC3.

RULSRF

The RULSRF statement creates a ruled surface, that is, a surface generated by sweeping straight lines from the ends of one curve to the ends of another curve.

Statement format:

```
        (XSMALL)          (XSMALL)
        (XLARGE)          (XLARGE)
RULSRF/(YSMALL),curve1,   (YSMALL),curve2
        (YLARGE)          (YLARGE)
        (ZSMALL)          (ZSMALL)
        (ZLARGE)          (ZLARGE)
```

Parameter	Description
XSMALL	The minor word indicating the x negative direction.
XLARGE	The minor word indicating the x positive direction.
YSMALL	The minor word indicating the y negative direction.
YLARGE	The minor word indicating the y positive direction.
ZSMALL	The minor word indicating the z negative direction.
ZLARGE	The minor word indicating the z positive direction.
	The preceding minor words indicate which ends of curve1 and curve2 are to be connected.
curve1	The name of the first curve.
curve2	The name of the second curve.

Example:

Program Statement	Explanation
`RLSF3=RULSRF/YSMALL,ARC9,$` `  XLARGE,ARC10`	Ruled surface RLSF3 is created between the y negative end of arc ARC9 and the x positive end of arc ARC10.

SPHERE

The SPHERE statement creates a sphere, that is, a surface defined by all points at a specified distance from a preset point.

Coordinates and Radius

Statement format:

```
SPHERE/xcoord,ycoord,zcoord,RADIUS[,radius[,EQUATR[,GOANG,goang]
    [,ENDANG,endang]][,CIRCUM[,GOANG,goang][,ENDANG,endang]]
```

Parameter	Description
xcoord	The x-coordinate of the sphere center point.
ycoord	The y-coordinate of the sphere center point.
zcoord	The z-coordinate of the sphere center point.
RADIUS	The minor word for indicating the sphere radius.
radius	The sphere radius in units of measure.
EQUATR	The minor word for indicating the equator of the sphere. The default goang for the equator is 0°. The default endang is 180°.
GOANG	The minor word for indicating the start angle of the equator or circumference.
goang	The start angle in degrees.
ENDANG	The minor word for indicating the circumference and end angle of the equator or circumference.
endang	The end angle in degrees.
CIRCUM	The minor word for indicating the sphere circumference. The default goang for the circumference is 0°. The default endang is 360°.

Example:

Program Statement	Explanation
`SPH4=SPHERE/3,3,9,RADIUS,10`	Sphere SPH4 is created with the center point at coordinates (3,3,9) and a radius of 10 units of measure. The equator and circumference start at 0° and end at 360° (the defaults).

Point and Radius

Statement format:

```
SPHERE/point,RADIUS,radius[,EQUATR[,GOANG,goang][,ENDANG,endang]]

      [,CIRCUM[,GOANG,goang][,ENDANG,endang]]
```

Parameter	Description
point	The name of the reference center point.
RADIUS	The minor word for indicating the sphere radius.
radius	The radius in units of measure.
EQUATR	The minor word for indicating the equator of the sphere. The default goang for the equator is 0°. The default endang is 180°.
GOANG	The start angle of the equator or circumference.
goang	The start angle in degrees.
ENDANG	The end angle of the equator or circumference.
endang	The end angle in degrees.
CIRCUM	The minor word for indicating the sphere circumference. The default goang for the circumference is 0°. The default endang is 360°.

Example:

Program Statement	Explanation

`SPH5=SPHERE/P338,RADIUS,10,$`
`EQUATR,GOANG,120,$`
`ENDANG,170,$`
`CIRCUM,GOANG,180,$`
`ENDANG,270`
	Sphere SPH5 is created with point P338 as the center point and a radius of 10 units of measure. The equator starts at 120° and ends at 170°. The circumference starts at 180° and ends at 270°.

TABCYL

The TABCYL statement creates a tabulated cylinder, that is, a surface generated by moving a curve in the direction of a line or vector.

Statement format:

```
                  ( XSMALL )
                  | XLARGE |
TABCYL/curve,     | YSMALL | ,line,PARAM,param1,param2
                  | YLARGE |
                  | ZSMALL |
                  ( ZLARGE )
```

Parameter	Description
curve	The name of the curve that generates the surface (generatrix).
XSMALL	The minor word indicating the x negative direction.
XLARGE	The minor word indicating the x positive direction.
YSMALL	The minor word indicating the y negative direction.
YLARGE	The minor word indicating the y positive direction.
ZSMALL	The minor word indicating the z negative direction.
ZLARGE	The minor word indicating the z positive direction.
	The preceding minor words indicate which end of the reference line the surface starts at.
line	The name of the direction line.
PARAM	The minor word for indicating the parameter values of the length of the surface created.
param1	The starting distance in units of measure from the curve at which the surface is created.
param2	The ending distance in units of measure from the curve at which the surface is created.

Example:

Program Statement	Explanation

`TC93=TABCYL/ARC4,$`
`  XSMALL,LINE8,$`
`  PARAM,5,25`
	Tabulated cylinder TC93 is created by moving curve ARC4 along line LINE8 from the x negative end of LINE8 to the x positive end. The surface is created 5 units of measure away from ARC4 to 25 units of measure away from ARC4.

TORUS

The TORUS statement creates a torus, that is, a surface generated by revolving a circle about a line.

Coordinates and Radii

Statement format:

```
TORUS/xcoord,ycoord,zcoord,RADII,inrad,outrad
```

Parameter	Description
xcoord	The x-coordinate of the torus center point.
ycoord	The y-coordinate of the torus center point.
zcoord	The z-coordinate of the torus center point.
RADII	The minor word for indicating the torus radii.
inrad	The inside radius in units of measure from the center point to the inner side of the torus.
outrad	The outside radius in units of measure from the center point to the outer side of the torus.

The difference of the inrad from the outrad determines the diameter of the circle that creates the torus.

Example:

Program Statement	Explanation
`TOR67=TORUS/30,9,12,$` `  RADII,5,15`	Torus TOR67 is created with the center point at coordinates (30,9,12). The inside radius of the torus is 5 units of measure, and the outside radius of the torus is 15 units of measure.

Point and Radii

Statement format:

```
TORUS/point,RADII,inrad,outrad
```

Parameter	Description
point	The name of the center point of the torus.
RADII	The minor word for indicating the radii of the torus.
inrad	The inside radius in units of measure from the center point to the inner side of the torus.
outrad	The outside radius in units of measure from the center point to the outer side of the torus.

Example:

Program Statement	Explanation
`TOR68=TORUS/PT15,$` `  RADII,6,12`	Torus TOR68 is created with point PT15 as the center point. The inside radius of the torus is 6 units of measure, and the outside radius of the torus is 12 units of measure.

ANSI 1973 Drafting Modal Statements 17

ANSI 1973 Drafting Modal Statements 17

This chapter describes the drafting modal statements that work according to the 1973 ANSI standard.

Drafting modal statements set various parameters for the creation of drafting entities. Drafting modal definitions are equivalent to types and forms found in menu 16.1 DRAFTING MODALS.

AHEAD

This modal statement modifies the arrowhead length.

Statement format:

```
AHEAD/length
```

Parameter	Description
length	The length of the arrowhead in units of measure. The width is one-third the length. This changes all subsequent label and dimension arrowheads.

Example:

Program Statement	Explanation
`AHEAD/2.1`	All subsequent arrowheads are created with a length of 2.1 and a width of 0.7 units of measure.

ARIN

This modal statement defines subsequent dimension arrows to be inside the witness lines. This applies only to dimensions that are outside the witness lines.

Statement format:

```
ARIN
```

Example:

Program Statement	Explanation
`ARIN`	All subsequent arrows are created inside the witness lines.

AROUT

This modal statement defines subsequent dimension arrows to be outside the witness lines and pointing toward the witness lines. This applies only to dimensions that are inside the witness lines.

Statement format:

```
AROUT
```

Example:

Program Statement	Explanation
AROUT	All subsequent arrows are created outside the witness lines.

ARROW

This modal statement switches on or off the arrowhead alignment of dimensions in linear dimensions. This modal affects linear dimensions in interactive ICEM DDN only. For arrowhead alignment, refer to chapter 18 for the use of the YES modifier in the LDIMEN statement.

Statement format:

```
ARROW/{ON }
      {OFF}
```

Parameter	Description
ON	Switches on arrowhead alignment.
OFF	Switches off arrowhead alignment.

AUTOD

This modal statement controls automatic dimensioning. The dimensions for linear, angular, radial, and diametric dimensions are automatically calculated. This is the default mode, and can only be used in the interactive mode to switch off KEYIN.

Statement format:

```
AUTOD
```

Example:

Program Statement	Explanation
AUTOD	All subsequent dimensions are created automatically.

NOTE

This modal applies to interactive use only.

CDISPL

This modal statement controls character display ratios. Any or all of these ratios can be modified as desired. At least one minor word is required.

Statement format:

```
CDISPL/[DELTA,delta][,ASPCT,aspect][,DOWNSP,downspace]
```

Parameter	Description
DELTA	The minor word for indicating the delta ratio. The character height multiplied by the delta ratio determines the distance between the starts of characters.
delta	The delta ratio expressed as a value or variable.
ASPCT	The minor word for indicating the aspect ratio. The aspect ratio is the ratio of character width to character height.
aspect	The aspect ratio expressed as a value or variable.
DOWNSP	The minor word for indicating the downspace ratio. The downspace ratio, DOWNSP, multiplied by the character height determines the distance between lines of text.
downspace	The downspace ratio expressed as a factor.

Example:

Program Statement	Explanation
`CDISPL/DELTA,1.3,DOWNSP,.9`	All subsequent characters are displayed. The distance between the start of one character and the start of another is 1.3 times the character height. The distance between one line and another is 0.9 times the character height.

CSET

This modal statement selects character set type. The type can be fine, fast, standard, or user-defined. Refer to menu 5.4 USER-DEFINED SYMBOLS in the ICEM Design/Drafting Data Management manual for the user-defined character and symbol definition.

Statement format:

```
        ⎧FINE                 ⎫
CSET/   ⎪FAST                 ⎪
        ⎪STD                  ⎪
        ⎩USER,character set   ⎭
```

Parameter	Description
FINE	The minor word indicating the use of the fine character set type.
FAST	The minor word indicating the use of the coarse character set type.
STD	The minor word indicating the use of the standard character set type.
USER	The minor word indicating the use of a user-defined character set.
character set	The name of the user-defined character set. It can be a string or text variable of up to 4 characters.

Example:

Program Statement	Explanation
`CSET/FAST`	All subsequent characters are in the fast character set.

CSIZE

This modal statement varies the character size. It remains in effect until a new CSIZE instruction is entered. The default is the character size of interactive ICEM DDN.

Statement format:

```
CSIZE/height
```

Parameter	Description
height	The height of the characters. The default is 2.5 mm (0.1 in).

Example:

Program Statement	Explanation
`CSIZE/.25`	The character size for all subsequent characters is 0.25 units of measure.

DECMAL

This modal statement sets the number of decimal places displayed in a dimension. If the current mode is fractional, it is changed to the decimal mode.

Statement format:

```
DECMAL/number
```

Parameter	Description
number	The number of decimal places, which can be from 0 through 6.

Example:

Program Statement	Explanation
`DECMAL/3`	All subsequent dimensions are displayed with three decimal places.

DIMOF

This modal statement changes the dimension offset distances.

Statement format:

```
DIMOF/[TEXTD,text offset][,WITP,witness offset][,WITX,witness extension]
```

Parameter	Description
TEXTD	The minor word for indicating the distance from the text of a dimension to its dimension lines.
text offset	The text offset distance in units of measure.
WITP	The minor word for indicating the distance a witness line is offset from the endpoint of the entity dimensioned.
witness offset	The witness offset distance in units of measure.
WITX	The minor word for indicating the distance a witness line extends past a dimension line.
witness extension	The witness extension distance in units of measure.

Example:

Program Statement	Explanation
`DIMOF/TEXTD,.2,$` `  WITP,.2,WITX,.4`	All subsequent dimensions are created at an offset distance of 0.2 units of measure from the dimension line. Witness lines extend from 0.2 units of measure from the end of the entity dimensioned to 0.4 units of measure beyond the dimension line.

NOTE

At least one option is required.

DORIG

This modal statement controls label and dimension origin. It determines the location on the screen where the lower left corner of the first character of a label or dimension is to be placed. Refer to 16.1.9 LABEL AND DIMEN ORIGIN in the ICEM Design/ Drafting Functions manual to see how this modal is applied. DELTA, AUTO, and PARLA affect drafting entities created in GPL. The other two modes affect entities created in interactive ICEM DDN only.

Statement format:

```
        (POSN  )
        (TYPIN )
DORIG/  (DELTA )
        (AUTO  )
        (PARLA )
```

Parameter	Description
POSN	The minor word indicating that the label and dimension origin is supplied directly by the user by screen position when the statement is executed.
TYPIN	The minor word indicating that the label and dimension origin is supplied directly by the user by a data entry when the statement is executed.
DELTA	The minor word indicating that the label and dimension origin is a delta distance from a point, label, or dimension.
AUTO	The minor word indicating that the label and dimension origin is automatically centered for linear dimensions, or it joins the leader line and leader line extension of a circular dimension of the center of a selected arc. The label and dimension origin is the lower left corner of the first character of labels and dimensions.
PARLA	The minor word indicating that the dimension is parallel to a line or arc.

Example:

Program Statement	Explanation
`DORIG/DELTA`	All subsequent label and dimension origins are created a delta distance away from a specified point, label, or dimension.

DSCALE

This modal statement sets the drafting scale factor. The drafting scale factor is the output scale of the arrowhead size of all labels and dimensions.

DSCALE scales the following dimension parameters:

- The distance from the text to the dimension line.
- The distance the witness line is offset from the reference point.
- The distance the witness line extends past the dimension line.
- The spacing and dash size for the centerline.
- The preset balloon radius.
- Arrowhead length.

Statement format:

```
DSCALE/factor
```

Parameter	Description
factor	The drafting scale factor. The default is 1.

Example:

Program Statement	Explanation
`DSCALE/.8`	The drafting scale factor is changed to 0.8 for all subsequent drafting entities.

DUAL

This modal statement sets dual dimensioning. If dual dimensioning is off, the standard dimensions are created. If dual dimensioning is on, both U.S. customary unit and SI unit dimensions are created. This modal affects drafting entities created in interactive ICEM DDN only.

Statement format:

```
DUAL/{ON  }
     {OFF }
```

Parameter	Description
ON	Sets dual dimensioning on, and both U.S. customary unit and SI unit dimensions are used.
OFF	Sets dual dimensioning off, and standard dimensions are used.

Example:

Program Statement	Explanation
`DUAL/ON`	Dual dimensioning is set for all subsequent dimensions.

FRACT

This modal statement changes the mode from decimal to fraction.

Statement format:

```
FRACT
```

Example:

Program Statement	Explanation
FRACT	All subsequent dimensions are displayed in fractions.

KEYIN

This modal statement is used to enter dimensions directly into a dimension (refer to 16.1.5 KEY-IN DIMENS in the ICEM Design/Drafting Drafting Functions manual for the application). In interactive mode only, this modal reverses the affect of AUTOD.

Statement format:

```
KEYIN
```

Example:

Program Statement	Explanation
KEYIN	The user enters the dimension text directly into the label specified in the dimension statement.

NOTE

This modal applies to interactive use only.

MATERL

This modal statement automatically creates the symbolic representation of different materials, that is, the section lines that correspond to the material type selected for any closed configuration.

Statement format:

```
          ( IRON   )
          ( STEEL  )
          ( BRASS  )
          ( COPPER )
          ( RUBBER )
MATERL/   ( REFRCT )
          ( MARBLE )
          ( SLATE  )
          ( LEAD   )
          ( MAGNES )
          ( ALUM   )
```

Parameter	Description
IRON	The minor word indicating the material type for iron.
STEEL	The minor word indicating the material type for steel.
BRASS	The minor word indicating the material type for bronze, brass, and copper.
COPPER	The minor word indicating the material type for bronze, brass, and copper.
RUBBER	The minor word indicating the material type for rubber and plastic.
REFRCT	The minor word indicating the material type for refractory material.
MARBLE	The minor word indicating the material type for marble, slate, and glass.
SLATE	The minor word indicating the material type for marble, slate, and glass.
LEAD	The minor word indicating the material type for lead, zinc, and babbitt.
MAGNES	The minor word indicating the material type for magnesium, aluminum, and aluminum alloys.
ALUM	The minor word indicating the material type for magnesium, aluminum, and aluminum alloys.

Example:

Program Statement	Explanation
MATERL/STEEL	Sectioning lines for the next SECTON statement use the material type for steel.

SLANT

This modal statement sets the character slant.

Statement format:

```
SLANT/(ON  )
      (OFF )
```

Parameter	Description
ON	Indicates the characters are to be slanted.
OFF	Indicates the characters are to be vertical.

Example:

Program Statement	Explanation
`SLANT/ON`	All subsequent characters are slanted.

TXTANG

The TXTANG statement selects the type of units used in the dimensioning of angles (created by RDIMEN statements).

Statement format:

```
        (DECIM  )
TXTANG/(DEGREE )
       (MINUTE )
       (SECOND )
```

Parameter	Description
DECIM	All subsequent angle dimensions are in decimal units.
DEGREE	All subsequent angle dimensions are in degrees only.
MINUTE	All subsequent angle dimensions are in degrees and minutes.
SECOND	All subsequent angle dimensions are in degrees, minutes, and seconds.

Example:

Program Statement	Explanation
`TXTANG/DEGREE`	All subsequent angle dimensions are in degrees only.

TXTJUS

The TXTJUS statement specifies which form of text justification is used for subsequent dimensioning statements.

Statement format:

```
        ⎧LEFT  ⎫
TXTJUS/⎨CENTER⎬
        ⎩RIGHT ⎭
```

Parameter	Description
LEFT	Text is left-justified. The origin point of the text is the left corner of the last line of text.
CENTER	Text is centered. The origin point of the text is the center of all lines of text.
RIGHT	Text is right-justified. The origin point of the text is the right corner of the last line of text.

Example:

Program Statement	Explanation
TXTJUS/CENTER	For all dimension text, each line of text is centered about the origin point. The origin point is located at the center of all lines of text.

WLINE

This modal statement sets witness line control for linear dimensions.

Statement format:

```
        (DISP2  )
        (SUPP1  )
        (SUPP2  )
WLINE/  (SUPPB  )
        (SUPP11 )
        (SUPP22 )
        (SUPPB1 )
        (SUPPB2 )
```

Parameter	Description
DISP2	Indicates no suppression (displays both witness lines). This is the default condition.
SUPP1	Suppresses the first witness line (suppresses the display of the witness line associated with the first selected entity).
SUPP2	Suppresses the second witness line (suppresses the display of the witness line associated with the second selected entity).
SUPPB	Suppresses both witness lines (displays no witness lines in this dimension).
SUPP11	Suppresses the display of the witness and dimension line of the first selected entity.
SUPP22	Suppresses the display of the witness and dimension line of the second selected entity.
SUPPB1	Suppresses the display of both witness lines and the dimension line of the first selected entity.
SUPPB2	Suppresses the display of both witness lines and the dimension line of the second selected entity.

Example:

Program Statement	Explanation
`WLINE/DISP2`	All subsequent witness lines are displayed. This statement shuts off one of the suppression conditions.

ANSI 1973 Dimensioning and Other Statements 18

ANSI 1973 Dimensioning and Other Statements 18

This chapter describes the dimensioning and other statements that work according to the 1973 ANSI standard. These statements imitate the operations found in menu 16 DRAFTING.

These drafting statements define entities used for drafting purposes such as dimensions, labels, notes, and sectioning. Drafting entity definitions are equivalent to the types and forms found in menu 16 DRAFTING.

CDIMEN

The CDIMEN statement defines a circular (radius) dimension. This statement dimensions the radius of the arc. A circular dimension is generated with a leader line. The leader line can originate either from the center of the arc pointing at the arc or from outside the arc pointing at the arc. If it originates from outside the arc, it must point so that an extension of the leader would pass through the center of the arc. The origin of the text is positioned as specified by the TXTJUS modal.

Statement format:

```
CDIMEN/{START},{INSIDE},arc name[,{point }][],deltax,deltay[,{   'text'    }][]
       {END  } {OUT   }          [{entity}]                  [{text variable}]
```

Parameter	Description
START	The minor word indicating that the leader origin is at the start of the text.
END	The minor word indicating that the leader origin is at the end of the text.
INSIDE	The minor word indicating that the arrowhead is to be placed on the inside of the arc pointing to the dimension.
OUT	The minor word indicating that the arrowhead is to be placed on the outside of the arc pointing to the center of the circle.
arc name	The name of the arc being dimensioned.
point	The name of the point to be the origin of the dimension text.
entity	The name of the dimensioning entity to be the origin of the dimension text.
deltax	The delta x value either from the center of the arc being dimensioned or from the point or dimensioning entity to the origin of the dimension text.
deltay	The delta y value from the center of the arc being dimensioned or from the point or dimensioning entity to the origin of the dimension text.

Parameter	Description
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of the text variable containing the text.

Examples:

Program Statement	Explanation
`CD1=CDIMEN/START,OUT,$` `  CIR1,3,2`	Circular dimension CD1 is created for circle CIR1. The leader origin is at the start of the text. The arrowhead of the leader is placed on the outside of the circle pointing into the center of the circle. The text is placed at delta coordinates (3,2) from the center of the circle.
`CD2=CDIMEN/END,INSIDE,$` `  CIR2,PT1,0,0`	Circular dimension CD2 is created for circle CIR2. The leader origin is at the end of the text. The arrowhead of the leader is placed on the inside of the circle pointing away from the center of the circle. Text is placed at delta coordinates (0,0) from point PT1. In other words, it uses the actual coordinates of point PT1.

CLINE

The CLINE statement defines a centerline.

Two Points

Statement format:

```
CLINE/point1,point2
```

Parameter	Description
point1	The name of the first point that defines the centerline.
point2	The name of the second point that defines the centerline.

A dash is placed between two lines which extend through the two points for 2.5 mm (0.1 in).

Example:

Program Statement	Explanation
CLN1=CLINE/PT1,PT2	Centerline CLN1 is created between points PT1 and PT2.

Multiple Points

Statement format:

```
CLINE/point1,point2,point3,...,pointn
```

Parameter	Description
point	The names of the points that define the centerline. Up to 12 points can be entered.

A centerline is created through the points, and dashes are centered on all the points other than the endpoints.

Example:

Program Statement	Explanation
SEC2=CLINE/PT1,PT2,PT3	Centerline SEC2 is created from point PT1 through point PT2 to point PT3.

Circle

Statement format:

```
CLINE/circle
```

Parameter	Description
circle	The name of the circle for which a crossed centerline is created.

A crossed centerline is automatically generated for one circle. Two linear centerlines are generated, one horizontal and one vertical, with the dashes crossing at the center of the circle and the endpoints extending 2.5 mm (0.1 in) outside the circle.

Example:

Program Statement	Explanation
`CLN3=CLINE/CIR1`	A crossed centerline CLN3 is created through the center of circle CIR1.

Circles with Collinear Centers

Statement format:

```
CLINE/circle1,circle2,...,circlen
```

Parameter	Description
circle	The names of the circles that define the centerline. Up to six circles can be entered.

Crossed centerlines are automatically generated for each circle in a line of circles that have collinear centers. The centerline places a dash at the centers of the circles, with the endpoints of the centerline extending 2.5 mm (0.1 in) outside the end circles. Each circle also has a crossed normal centerline with dashes crossing at the center of the circle and the endpoints extending 2.5 mm (0.1 in) outside the circle.

Example:

Program Statement	Explanation
`SEC4=CLINE/CIR1,CIR2,$` `CIR3,CIR4`	Centerline SEC4 is created from circle CIR1 through circles CIR2 and CIR3 to circle CIR4.

Bolt Circle

Statement format:

```
CLINE/circular array
```

Parameter	Description
circular array	The name of a circular array.

A centerline is created along the center of each element of a circular array. Cross centerlines are automatically generated for each element of the array. The cross centerlines follow the rules given under Circles with Collinear Centers.

Legal components of the circular array include points and circles.

DDIMEN

The DDIMEN statement defines a diameter dimension. Diameter dimensions are generated so that dimension lines lie along a diameter of the circle. The direction is determined by the direction of a line from the circle center to the center of the dimension text. The origin text of the dimension is positioned as specified by the TXTJUS modal.

Statement format:

```
DDIMEN/arc name[,(point )],deltax,deltay[,(   'text'     )]
               [ (entity)]                [(text variable)]
```

Parameter	Description
arc name	The name of the arc being diametrically dimensioned.
point	The name of the point to be the origin of the dimension text.
entity	The name of the dimensioning entity to be the origin of the dimension text.
deltax	The delta x value either from the center of the arc being dimensioned, or from the point or dimensioning entity to the origin of the dimension text.
deltay	The delta y value either from the center of the arc being dimensioned, or from the point or dimensioning entity to the origin of the dimension text.
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of the text variable containing the text.

Example:

Program Statement	Explanation
DD1=DDIMEN/C11,6.25,3.5	Diameter dimension DD1 is created for circle C11. The origin point of the dimension text is at delta coordinates (6.25, 3.5) from the center of circle C11.

LABEL

The LABEL statement constructs labels. A general label consists of a string of characters with an associated leader line generated from the label to an entity on the drawing. The leader includes a tail, which is a short horizontal bar extending from the beginning or end of the text. If the minor word END is specified, the tail is placed to the right of the text.

The vertical position of the leader line tail is determined by the modal statement LEADER. The leader touches the first text line by default. The leader line automatically terminates at the entity through one of the following options:

- Coordinate Method. Coordinates at or near the desired leader line termination point are entered. These coordinates need not be directly on the entity, as the leader line automatically intersects itself precisely with the entity. If the entity is a point, the leader terminates at the point.
- Slope Method. The angle of the leader line is entered. The leader line is generated to the entity indicated at the angle specified. The leader line automatically terminates at the intersection point on the entity.
- Midpoint Method. If no leader line termination control is used, the leader line automatically ends at the midpoint of the entity.

The text of the label is positioned by locating the origin (TXTJUS modal statement) either with the absolute coordinates or at a delta distance from either another dimensioning entity or a point. The text angle of the label can be varied by entering ANGLE and the text angle in degrees. The string of characters that form the text of the label can be replaced by a text variable.

Statement format:

```
LABEL/(START),((xt,yt),deltax,deltay),entity labeled
      (END  )  (point )
               (entity)

      [,(SLOPE,slope  )][,ANGLE,angle][,('text'       )]
        (termxt,termyt)]               [ (text variable)]
```

Parameter	Description
START	The minor word indicating that the leader origin is at the start of the text.
END	The minor word indicating that the leader origin is at the end of the text.
xt,yt	The absolute coordinates of the label text origin.
point	The name of the point to be the origin of the label text.
entity	The name of the dimensioning entity to be the origin of the label text.

Parameter	Description
deltax	The delta x value from the point or the dimensioning entity to the origin of the dimension text.
deltay	The delta y value from the point or the dimensioning entity to the origin of the dimension text.
entity labeled	The name of the entity being labeled.
SLOPE	The minor word for indicating the angle of the leader from the positive x-axis. The leader line is generated to the entity indicated at the angle specified. The horizontal line length from the label center is fixed at 3.17 mm (0.125 in). The leader line automatically terminates at the point at which it intersects the entity.
slope	The angle of the leader line from the positive x-axis.
termxt, termyt	The coordinates at or near the leader line termination point.
ANGLE	The minor word for indicating the text angle of the label.
angle	The text angle in degrees.
'text'	The text of the label.
text variable	The name of the text variable containing the text for the label.

Example:

Program Statement	Explanation
`LAB1=LABEL/START,2.1,3.7,LN2,$` `  'chamfer .0625 x 45 degrees'`	Label LAB1 is created for line LN2 at coordinates (2.1,3.7). The leader starts at the beginning of the text. The text is: chamfer .0625 x 45 degrees.

LDIMEN

The LDIMEN statement defines linear dimensions which are horizontal, vertical, or parallel. The dimension text for horizontal dimensions is generated parallel to the x-axis with vertical witness lines. Dimension lines for vertical dimensions are generated parallel to the y-axis with horizontal witness lines.

Parallel dimensions are generated relative either to a given line, or to the apparent line that exists between any two points, and produce dimension lines parallel to the line. The witness lines for parallel dimensions are perpendicular to the dimension lines, the text is horizontal, and the dimension lines pass through its geometric center. The following two types of parallel dimensioning are available:

Dimension	Description
PARLEL	Generates a dimension relative either to a given line or to the apparent line that exists between two entity endpoints.
PARNOR	Generates a dimension to two parallel lines.

When selecting entities to dimension, includes a position indicator after the entity. The exceptions are:

- If the entity is a point.
- If only one entity is dimensioned (the entity's endpoints are used).

To align the arrowheads with the arrowheads of the previous dimension, use the YES parameter.

The dimension origin of the text can be entered in the following ways:

Origin	Description
Coordinates	Imagine an arc drawn counterclockwise from the first entity endpoint through the dimension origin to the second entity endpoint. The first point of this arc is the one from which a delta xt value and delta yt value finds the dimension origin.
Delta from a point or dimensioning entity	A point or dimensioning entity is specified and delta x and delta y points or coordinates are given from that point or entity.

The definition form for this statement is:

dimension type,entity selection[,arrowhead alignment],dimension origin[,key-in text]

Statement format:

```
                                 point                              point
          HORIZ                         XSMALL                             XSMALL
LDIMEN/   VERTCL  ,   entity   [ ,      XLARGE  ]  [ ,   entity   [ ,      YLARGE  ] ]
          PARLEL                        YSMALL                             YSMALL
          PARNOR                        YLARGE                             YLARGE

[,YES][,origin],deltaxt,deltayt[,  'text'       ]
                                   text variable
```

Parameter	Description
HORIZ	The minor word indicating a horizontal dimension.
VERTCL	The minor word indicating a vertical dimension.
PARLEL	The minor word indicating a parallel dimension relative to a given line.
PARNOR	The minor word indicating a parallel dimension normal to two given parallel lines.
point	The names of points being dimensioned.
entity	The names of entities being dimensioned.
XSMALL	The minor word indicating the x negative direction.
XLARGE	The minor word indicating the x positive direction.
YSMALL	The minor word indicating the y negative direction.
YLARGE	The minor word indicating the y positive direction.
	The XSMALL, XLARGE, YSMALL, and YLARGE minor words indicate which end of the entity to dimension.
YES	The minor word for aligning the arrowheads with the arrowheads of the previous dimension.
origin	The name of a reference point or entity to be the origin of the dimension.
deltaxt, deltayt	The delta xt- and yt-coordinates from the first entity endpoint or the reference point or entity.
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of the text variable containing the text.

Examples:

Program Statement	Explanation
`LD1=LDIMEN/HORIZ,$` `  LN1,3.15,2.5`	Horizontal linear dimension LD1 is created for line LN1. The dimension text starts at delta coordinates (3.15, 2.5) from line LN1.
`LD2=LDIMEN/VERTCL,LN2,.5,1`	Vertical linear dimension LD1 is created for line LN1. The dimension text starts at coordinates (.5,1).

`LD3=LDIMEN/PARLEL,$`
`  LN1,XLARGE,$`
`  LN3,XLARGE,PT1,0,0`
	Parallel linear dimension LD3 is created between line LN1 and line LN3. The dimension text starts a point PT1. The dimension is drawn from the x positive end of line LN1 to the x positive end of line LN2.

NOTE

The NOTE statement enters notes to a part drawing. General notes consist of contiguous strings of characters including blanks. The note is entered as an alphanumeric string and can contain up to 200 characters. The \ (backslash) starts a new line immediately below the previous line.

A general note is positioned by locating the origin (TXTJUS modal) using either a point or dimensioning entity and delta coordinates or by specifying xt- and yt-coordinates.

Statement format:

```
NOTE/( xt,yt                    ),( (POSITV),line  ),(   'text'       )
     ( (point ),deltax,deltay   ),( (NEGATV)       ),( text variable )
     ( (entity)                 ),( (CW    ),arc   ),( real variable )
                                   ( (CCW   )       )
                                   ( (ANGLE),angle   )
```

Parameter	Description
xt,yt	The absolute coordinates of the note origin.
point	The name of a point used as the origin of the note text.
entity	The name of a dimensioning entity used as the origin of the note text.
deltax	The delta x value from the point or entity to the origin of the note text.
deltay	The delta y value from the point or entity to the origin of the note text.
POSITV	The minor word for positioning the note text parallel to a given line in the positive direction from the x-axis. If the line is vertical, the text is displayed sideways and upwards.
line	The name of the reference line.
NEGATV	The minor word for positioning the note text parallel to a given line in the negative direction from the x-axis. If the line is vertical, the text is displayed sideways and downwards.

Parameter	Description
CW	The minor word for positioning the note text parallel to a given arc in a clockwise direction.
arc	The name of the reference arc.
CCW	The minor word for positioning the note text parallel to a given arc in a clockwise direction.
ANGLE	The minor word for the text angle of the label.
angle	The text angle in degrees.
'text'	The text of the note. The text must be enclosed in single quotes.
text variable	The name of a text variable containing the text.
real variable	The name of a real variable. The variable is converted to a text string and truncated. The current numeric mode must be DECMAL.

Examples:

Program Statement	Explanation
`NT1=NOTE/1.5,2.75,ANGLE,0,$` `  'THIS IS A GENERAL NOTE'`	Note NT1 is created at coordinates (1.5,2.75).
`NT2=NOTE/PT1,0,0,ANGLE,0,$` `  'THIS IS A GENERAL NOTE'`	Note NT2 is created at delta coordinates (0,0) from point PT1. The note is created at the coordinates of point PT1, but the delta coordinates are still required for this statement.

RDIMEN

The RDIMEN statement defines an angular dimension between two lines. Circular dimension lines are created. The radius of the witness line is automatically generated. The automatically generated text is the angle included by the lines measured from the first line entered. The text is in degrees and fractions thereof. The text is generated as specified by the TXTANG modal.

The text is positioned by locating the origin (TXTJUS modal) using either the xt- and yt-coordinates which are delta from the intersection of the two lines or by specifying the delta coordinates from a point or dimensioning entity.

Statement format:

```
RDIMEN/(XSMALL|XLARGE|YSMALL|YLARGE),line,(XSMALL|XLARGE|YSMALL|YLARGE),line

[,(point|entity),deltax,deltay][,('text'|text variable)]
```

Parameter	Description
XSMALL	The minor word indicating the x negative direction.
XLARGE	The minor word indicating the x positive direction.
YSMALL	The minor word indicating the y negative direction.
YLARGE	The minor word indicating the y positive direction.
	The preceding minor words indicate which ends of the entities to dimension.
line	The names of the entities to be dimensioned.
point	The name of a point used as the origin of the dimension text.
entity	The name of a dimensioning entity used as the origin of the dimension text.
deltax	The delta x value from the intersection of the two generating lines or from the point or drafting entity to the origin of the dimension text.
deltay	The delta y value from the intersection of the two generating lines or from the point or drafting entity to the origin of the dimension text.
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of a text variable containing the text.

Example:

Program Statement	Explanation
`RD1=RDIMEN/XLARGE,LN1,$` `  YLARGE,LN2,PT1,0,0`	Angular dimension RD1 is created from the x positive end of line LN1 to the y negative end of line LN2. The dimension text starts at point PT1.

SECTON

The SECTON statement sections parts. Sectioning automatically generates section lines for any closed (apparent to view) configuration. Islands are automatically recognized if they are in the list of entities or the entity array.

Statement format:

```
SECTON/{      entity[,...]       }[,SECDIS,distance][,ANGLE,angle]
       {NUMBER,number,entity array}
```

Parameter	Description
entity	The names of the entities sectioned. Entities allowed are lines, arcs, general conics, and splines.
NUMBER	The minor word for specifying a number of entities from an entity array.
number	The number of entities in the entity array.
entity array	The name of an entity array.
SECDIS	The minor word indicating the distance between sectioning lines. The spacing between the lines is adjustable, as is the angle at which the lines are generated. If the distance between lines is omitted, 2.5 mm (0.10 in) is assumed.
distance	The distance between the sectioning lines in units of measure.
ANGLE	The minor word indicating the angle of the sectioning lines. If the angle for lines is omitted, 45° is assumed. The angle for a set of section lines can be varied between 0° and 180°.
angle	The angle of the sectioning lines in positive degrees from the x-axis.

Examples:

Program Statement	Explanation

`MATERL/IRON`
`SEC1=SECTON/LN1,LN10,LN9,$`
`LN11,LN8,LN7,LN13,LN12,$`
`SECDIS,.25`
	Section SEC1 is created using the section lining symbol of iron. The distance between lines is modified to 0.25 units of measure.

`MATERL/STEEL`
`SEC2=SECTON/LN1,CIR1,CIR2,$`
`LN2,LN9,LN10,ANGLE,135`
	Section SEC2 is created using the section lining symbol of steel. The angle of the lines is modified to 135° from the x positive axis.

`MATERL/BRASS`
`SEC3=SECTON/LN3,CIR3,LN4,$`
`LN5,LN7,LN8,ANGLE,135`
	Section SEC3 is created using the section lining symbol of brass. The angle of the lines is modified to 135° from the x positive axis.

`MATERL/ALUM`
`SEC4=SECTON/LN7,LN6,LN12,$`
`LN13,SECDIS,.175,$`
`ANGLE,30`
	Section SEC3 is created using the section lining symbol of aluminum. The distance between lines is modified to 0.175 units of measure. The angle is modified to 30° from the x positive axis.

ANSI 1982 Drafting Modal Statements 19

ANSI 1982 Drafting Modal Statements 19

This chapter describes the drafting modal statements that work according to the 1982 ANSI standard. These statements imitate the operations found in menu 16.1 DRAFTING MODALS.

The same statement described elsewhere in this manual, works only with the 1973 ANSI standard. For example, AHEAD in this chapter applies to the 1982 ANSI standard. AHEAD as described in chapter 17 DRAFTING MODALS in this manual applies to the 1973 ANSI standard. In many of these statements, the commands appear to act exactly the same, however, there may be subtle differences. You should therefore be careful as you use these statements. Also, some commands described in this chapter do not work at all with the 1973 standard. For example, ANGCTL is not implemented in the 1973 ANSI standard.

AHEAD

This modal statement sets the length of arrowheads and dimension origin circles. The width of arrowheads is one-third the length. The length is preset to 0.15 inches (3.81 mm). If the value entered is less than or equal to zero, the following error message is displayed:

```
VALUE MUST BE GREATER THAN ZERO
```

Changing this modal affects only dimensions and arrowheads subsequently created.

Statement format:

```
AHEAD/length
```

Parameter	Description
length	The length of the arrowhead in units of measure.

Example:

Program Statement	Explanation
`AHEAD/.3`	The length for all subsequent arrowheads is set to 0.3 units of measure. The width of the arrowhead is set to 0.1 units of measure.

ANGCTL

This modal statement controls the angle at which the text of a note or label is written. The default value for ANGCTL is HORIZ.

Statement format:

```
        (HORIZ )
ANGCTL/(ENTER )
        (PARLEL)
        (TOTAL )
```

Parameter	Description
HORIZ	The minor word for setting the text horizontally to the screen.
ENTER	The minor word for displaying a menu that lets the user choose the method of setting the text angle.
PARLEL	The minor word for writing a note parallel to a line or arc.
TOTAL	The minor word for writing notes at a user-specified angle or parallel to a line or arc.

NOTE

ANGCTL affects only notes and labels created interactively.

Example:

Program Statement	Explanation
`ANGCTL/PARLEL`	All subsequent text is written parallel to a line or arc.

ANUNIT

This modal statement sets the method in which the angle in angular dimensions is represented. The default value of ANUNIT is DECIM.

Statement format:

```
        ⎧DECIM  ⎫
ANUNIT/⎪DEGREE ⎪
        ⎪MINUTE ⎪
        ⎩SECOND ⎭
```

Parameter	Description
DECIM	The minor word for expressing the angle in decimals.
DEGREE	The minor word for expressing the angle in degrees.
MINUTE	The minor word for expressing the angle in degrees and minutes.
SECOND	The minor word for expressing the angle in degrees, minutes, and seconds.

Example:

Program Statement	Explanation
`ANUNIT/DEGREE`	All subsequent angular dimensions are written in degrees.

ARAUTO

This modal statement automatically determines the placement of arrows according to the placement of text and the available space. ARAUTO cancels the effect of ARIN and AROUT. Existing dimensions are not affected.

Statement format:

```
ARAUTO
```

Example:

Program Statement	Explanation
`ARAUTO`	All subsequent dimension arrows are placed according to the placement of the text and the available space.

ARIN

This modal statement generates dimensions with the arrows inside (between) the entities or extension lines to which the dimensions extend. ARIN cancels the effect of ARAUTO and AROUT. Existing dimensions are not affected.

Statement format:

```
ARIN
```

Example:

Program Statement	Explanation
ARIN	All subsequent arrows are placed inside the extension lines.

AROUT

This modal statement generates dimensions with the arrows outside the entities or extension lines to which the dimensions extend. AROUT cancels the effect of ARAUTO and ARIN. Existing dimensions are not affected.

Statement format:

```
AROUT
```

Example:

Program Statement	Explanation
AROUT	All subsequent arrows are placed outside the extension lines.

ARROW

This modal statement creates vertical or horizontal dimensions that are aligned with previously created dimensions of the same type. The default value of ARROW is OFF. Changing this modal statement affects only dimensions subsequently created.

Statement format:

```
ARROW/{ON }
      {OFF}
```

Parameter	Description
ON	Switches on arrowhead alignment.
OFF	Switches off arrowhead alignment.

NOTE

ARROW affects only dimensions created interactively. Use the YES parameter of LDIMEN for aligning dimensions in GPL.

Example:

Program Statement	Explanation
`ARROW/ON`	Switches on arrowhead alignment for arrowheads created interactively.

ATAIL

This modal statement controls the entry point of the tail location while creating or modifying the origin of drafting entities with tails.

The default value of ATAIL is ON.

Statement format:

```
ATAIL/{ON }
      {OFF}
```

Parameter	Description
ON	Sets the tail location automatically.
OFF	Does not set the tail location automatically.

Example:

Program Statement	Explanation
`ATAIL/ON`	The tail location is set automatically.

AUTOD

This modal statement automatically calculates or prompts for dimensions for linear, angular, radius, and diameter dimensions. This statement affects only those dimensions created interactively.

Dimensions created in GPL are always automatically calculated unless a dimension text is specified in the GPL statement.

The default value of AUTOD is ON.

Statement format:

```
AUTOD/{ON }
      {OFF}
```

Parameter	Description
ON	Dimension text is automatically generated.
OFF	Prompts to enter dimension text manually.

Example:

Program Statement	Explanation
AUTOD/ON	Dimension text is automatically generated and not entered interactively.

CDISPL

This modal statement modifies the ratios of character display. This modal statement affects only subsequently created dimensions and does not change existing dimensions.

Statement format:

```
CDISPL/[DELTA,char space][,ASPCT,aspect][,DOWNSP,line space][,TOLER,tolerance]
```

Parameter	Description
DELTA	The minor word indicating the spacing between the centers of consecutive characters.
char space	The value multiplied by the character height that gives the spacing between the centers of consecutive characters. This value is preset to 1.1.
ASPCT	The minor word indicating the aspect ratio, which is the ratio between character width and character height.
aspect	The value of the aspect ratio. This value is preset to 1.
DOWNSP	The minor word indicating the downspace ratio, which is the line spacing value multiplied by the character height that gives the distance between the bottom of lines of text.
line space	The value of the downspace ratio. This value is preset to 1.5.
TOLER	The minor word indicating the tolerance ratio (the ratio between the character size for tolerance and fraction characters and the character size for main characters in dimensions).
tolerance	The value of the tolerance ratio. This value is preset to 1.

If the value entered for any of these options is less than or equal to zero, the following error message is displayed:

```
VALUE MUST BE GREATER THAN ZERO
```

NOTE

At least one minor word is required.

Example:

The following statement sets the character display ratios. The spacing between characters is set to a ratio of 1.5 times the character height. The character width is set to a ratio of 0.9 of the character height. The spacing between lines is set to a ratio of 2 times the character height.

```
CDISPL/DELTA,1.5,ASPCT,.9,DOWNSP,2
```

CRES

This modal statement selects a method of output representation for standard set type characters. This modal statement affects all existing standard characters generated. Lowercase characters appear as uppercase in FAST mode. The default value for CRES is FINE.

Statement format:

```
CRES/(FINE)
     (FAST)
```

Parameter	Description
FINE	The minor word indicating the fine character set type.
FAST	The minor word indicating the coarse character set type.

Example:

Program Statement	Explanation
`CRES/FINE`	The character set type is set to the FINE representation mode.

CSET

This modal statement selects the character set to be used in defining text. This modal statement does not affect the text in any existing dimensions, labels, or notes.

If you try to access a user-defined character set and the character set is not found, the system displays:

```
CHARACTER SET NOT FOUND IN UTF
```

If the character variable for the character set name is more than four characters long, the system displays:

```
CHARACTER SET NAME TOO LONG
```

Standard characters are used for any characters not defined in the character set.

Statement format:

```
CSET/(STD
      (USER,character set name)
```

Parameter	Description
STD	The minor word indicating the standard character set.
USER	The minor word indicating a user-defined character set.
character set name	The name of a user-defined character set (maximum 4 characters in length).

Example:

Program Statement	Explanation
`CSET/USER,'MYCH'`	The user-defined character set MYCH is selected.

CSIZE

This modal statement specifies the height of the characters that are written by the dimension, label, and note operations. Existing dimensions are not affected. This value is preset to 0.125 inches (3.18 mm).

If the value entered is less than or equal to zero, the following error message is displayed.

```
VALUE MUST BE GREATER THAN ZERO
```

Statement format:

```
CSIZE/character height
```

Parameter	Description
character height	The height of the characters.

Example:

Program Statement	Explanation
`CSIZE/.25`	The height of all subsequent characters is set to 0.25 units of measure.

DECMAL

This modal statement sets the number of decimal places in the display of dimensions. It can also be used to return to the decimal mode from the fraction mode. If dual dimensioning has been set, the user can set the number of decimal places to be used in the regular units text and/or the alternate units text. The number of decimal places for regular dimensions is preset to 4 (9 maximum). The number of decimal places for alternate dimensions is preset to 3 (6 maximum). This modal statement does not affect existing dimensions.

Statement format:

```
DECMAL/[regular unit decimal places][,SECOND,alternate unit decimal places]
```

Parameter	Description
regular unit decimal places	The value of the regular unit decimal places.
SECOND	The minor word indicating alternate unit decimal places.
alternate unit decimal places	The value of the alternate unit decimal places.

Example:

Program Statement	Explanation
`DECMAL/3,SECOND,4`	The regular unit decimal places is set to 3 and the alternate unit decimal places is set to 4.

NOTE

No parameters are required to return to decimal mode from fraction mode. The number of decimal places for regular units text and alternate units text are the previous values set.

DIMOF

This modal statement sets the dimension offset distances.

If the value entered for any of these options is less than or equal to zero, the following error message is displayed:

```
VARIABLE OR CONSTANT MUST BE GREATER THAN ZERO
```

Statement format:

```
DIMOF/[TEXTD,text offset][,WITP,witness offset][,WITX,witness extension]
```

Parameter	Description
TEXTD	The minor word indicating the distance from the text to the dimension lines.
text offset	The value of the distance from the text to the dimension lines, which is preset to 0.1 inch (2.54 mm). This parameter affects only newly created dimensions.
WITP	The minor word indicating the distance from an extension line to the entity from which it extends.
witness offset	The value of the distance from an extension line to the entity from which it extends, which is preset to 0.0625 inch (1.59 mm). This parameter affects only newly created dimensions.
WITX	The minor word indicating the distance that an extension line extends beyond a dimension line.
witness extension	The value of the distance that an extension line extends beyond a dimension line, which is preset to 0.125 inch (3.17 mm). This parameter affects only newly created dimensions.

Example:

Program Statement	Explanation
`DIMOF/TEXTD,.3,WITP,$` `.2,WITX,.4`	The text is set 0.3 units from the dimension line. Witness lines start at 0.2 units from the entity, and they end at 0.4 units beyond the entity.

NOTE

At least one minor word is required.

DIMORG

This modal statement sets which side, if any, of a generated dimension specifies a dimension origin. A dimension origin is graphically represented as a circle instead of an arrowhead at the end of a dimension line. The diameter of the dimension origin circle is determined by the arrowhead length. This modal statement applies to horizontal, vertical, parallel, and thickness dimensions. The default value of DIMORG is NONE.

Statement format:

```
        ⎧FIRST  ⎫
DIMORG/⎨SECOND ⎬
        ⎩NONE   ⎭
```

Parameter	Description
FIRST	Indicates the first side selected as the dimension origin.
SECOND	Indicates the second side selected as the dimension origin.
NONE	Indicates that neither side is selected as the dimension origin.

Example:

Program Statement	Explanation
`DIMORG/FIRST`	The first side selected is the dimension origin.

DSCALE

This modal statement sets the scale factor used in generating dimensions, labels, and notes. The following are affected:

- The character size.
- The arrowhead size of all dimensions and labels.
- The distance from the text of a dimension to the dimension line.
- The distance that an extension line is offset from a reference point.
- The distance that the extension line extends past the dimension line.
- The spacing and dash size for centerlines.
- The preset balloon and datum target symbol radius.

This statement affects all newly created dimensions. If the drafting scale is halved, the above drafting modal statements are doubled. Conversely, if the drafting scale is doubled, the above drafting modal statements are halved.

Alternately, a ratio to the current draft scale may be entered which will proportionally affect the drafting scale factor.

Statement format:

```
DSCALE/(drafting scale factor)
       (     RATIO,ratio     )
```

Parameter	Description
drafting scale factor	The drafting scale factor. The value is preset to 1.
RATIO	The minor word indicating a ratio to the current drafting scale.
ratio	The ratio expressed as a value.

Example:

Program Statement	Explanation
`DSCALE/.8`	The drafting scale factor is set to 0.8 for all subsequent entities.

DUAL

This modal statement determines whether the dimension operations are in both SI and U.S. customary units or in SI or U.S. customary units as set for the particular part.

The default value of DUAL is OFF.

Statement format:

```
        ⎧OFF   ⎫
DUAL/   ⎪BOTH  ⎪
        ⎪BRACKT⎪
        ⎩POSN  ⎭
```

Parameter	Description
OFF	The minor word indicating no dual dimensioning.
BOTH	The minor word for producing dual dimensions with the alternate dimensions inside square brackets and a line separating upper and lower text.
BRACKT	The minor word for producing dual dimensions with the alternate dimensions inside square brackets and no line separating upper and lower text.
POSN	The minor word for producing dual dimensions with a line separating upper and lower text and no brackets around the alternate dimensions.

Example:

Program Statement	Explanation
DUAL/BOTH	Dimensions are written using both SI and U.S. customary units.

FRACT

This modal statement sets the fraction mode for automatically generated dimension text. The system writes dimensions as common or mixed fractions displayed with an accuracy of up to 1/64.

Statement format:

```
FRACT
```

Example:

Program Statement	Explanation
FRACT	All subsequent dimensions are written in fractions.

LDDIAM

This modal statement controls the automatic insertion of a diameter symbol with the text automatically generated for certain dimensions. This symbol applies to horizontal, vertical, parallel, and thickness dimensions. Note that automatically generated diameter dimensions always have diameter symbols.

The preset value of LDDIAM is OFF.

Statement format:

```
LDDIAM/(ON  )
       (OFF )
```

Parameter	Description
ON	Indicates all subsequent linear dimension have a diameter symbol as the first character of the text.
OFF	Turns off the forced diameter symbol.

Example:

Program Statement	Explanation
`LDDIAM/ON`	The diameter symbol is inserted in the dimension text.

LEADER

This modal statement determines the placement of the label leader with respect to the label text. Existing labels are not affected.

The default value of LEADER is FIRST.

Statement format:

```
LEADER/(FIRST  )
       (MIDDLE )
```

Parameter	Description
FIRST	The label leader line starts from the first text line.
MIDDLE	The label leader line starts from the middle text line.

Example:

Program Statement	Explanation
`LEADER/FIRST`	The leader line starts from the first line of the text.

MATERL

This modal statement selects the type of section lining.

The default value of MATERL is IRON.

Statement format:

```
MATERL/material type
```

Parameter	Description
material type	The material type, which is one of the following minor words.

Minor Word	Description
IRON	Iron
STEEL	Steel
BRASS or COPPER	Bronze, brass, copper
RUBBER or PLASTC	Rubber, plastic
REFRCT	Refractory material
MARBLE, SLATE or GLASS	Marble, slate, glass
LEAD	Zinc, lead, babbitt
MAGNES or ALUM	Magnesium, aluminum, aluminum alloys

Example:

Program Statement	Explanation
MATERL/STEEL	The symbol for steel is used for subsequent section lining.

PREFIX

This modal statement changes the character used to prefix a special symbol character set. Entering the prefix character followed by certain preassigned characters allows entry of drafting symbols into notes, labels, and dimensions.

The reverse slant (\) is the default prefix character.

Statement format:

```
PREFIX/'character'
```

Parameter	Description
'character'	The prefix character, which is a string or character variable of 1 character enclosed in single quotes.

Example:

Program Statement	Explanation
`PREFIX/'*'`	The prefix character is changed to the asterisk symbol *.

SECALN

This modal statement controls the alignment of new section lining to existing section lining.

The default value of SECALN is OFF.

Statement format:

```
SECALN/[ON ]
       [OFF]
```

Parameter	Description
ON	Subsequent section lining is aligned with the previous section lining.
OFF	Subsequent section lining is not aligned with the previous section lining.

NOTE

SECALN affects only section lining created interactively. Use the YES,entity parameters with SECTON for aligning section lining in GPL.

Example:

Program Statement	Explanation
`SECALN/ON`	Subsequent section lining is aligned with previous section lining.

SECVIS

This modal statement determines whether to display section lining in all views or only in the view in which the section lining is defined.

The system default display is in the view of definition only.

This modal statement affects subsequently generated section lining.

Statement format:

```
SECVIS/{VDEF}
       {ALL }
```

Parameter	Description
VDEF	Section lining is displayed in the view of definition only.
ALL	Section lining is displayed in all views.

Example:

Program Statement	Explanation
SECVIS/VDEF	Section lining is displayed in the view of definition only.

SLANT

This modal statement selects either vertically oriented characters or slanted characters. This modal statement does not affect existing text. The default value of SLANT is OFF.

Statement format:

```
SLANT/{ON }
      {OFF}
```

Parameter	Description
ON	Indicates the characters are slanted.
OFF	Indicates the characters are vertical.

Example:

Program Statement	Explanation
SLANT/ON	All subsequent characters are slanted.

TXTJUS

This modal statement specifies the text origin (in relation to the text) and the text justification. This modal statement does not affect existing text.

The default value of TXTJUS is LEFT.

Statement format:

```
        ( LEFT   )
TXTJUS/ ( CENTER )
        ( RIGHT  )
```

Parameter	Description
LEFT	Text is left justified. The origin point of the text is the lower left corner of the first character.
CENTER	Text is centered. The origin point of the text is the center of the full text.
RIGHT	Text is right justified. The origin point of the text is the lower right corner of the last character of the first line.

Example:

Program Statement	Explanation
`TXTJUS/RIGHT`	All subsequent text is right-justified and the origin is the lower right corner of the last character of the first line.

TXTORG

This modal statement presets the method of indicating the position of text.

The default value for TXTORG is POSN.

Statement format:

```
        ( POSN  )
TXTORG/ | ENTER |
        | DELTA |
        ( AUTO  )
```

Parameter	Description
POSN	In interactive mode: use the graphics cursor to indicate a screen position. In GPL: the x and y values specify absolute coordinates.
ENTER	In interactive mode: enter the coordinates of a position. In GPL: the x and y values specify absolute coordinates.
DELTA	In interactive mode: enter horizontal and vertical displacements from an existing point, note, label, or dimension. In GPL: the x and y values specify a delta distance from a given point or predefined location (dependent on the type of dimension).
AUTO	The minor word to automatically center horizontal, vertical, and parallel dimensions; to join the leader line extension of a circular dimension at the center of an arc. Same effect as ENTER for angular and diameter dimensions.

NOTE

TXTORG does not affect notes and labels created in GPL.

Example:

Program Statement	Explanation
`TXTORG/DELTA`	All subsequent text origins are a delta displacement from another location.

WLINE

This modal statement controls the generation of extension lines and dimension lines. The extension line extends from an entity; it is the line to which a dimension is measured. A dimension line is the arrow from the text to the extension line; it is the line that shows the dimension being measured.

The default value of WLINE is DISP2.

Statement format:

```
        (DISP2  )
        (SUPP1  )
        (SUPP2  )
WLINE/  (SUPPB  )
        (SUPP11 )
        (SUPP22 )
        (SUPPB1 )
        (SUPPB2 )
```

Parameter	Description
DISP2	The minor word for generating dimension and extension lines to both entities.
SUPP1	The minor word for suppressing the extension line that extends from the first entity selected.
SUPP2	The minor word for suppressing the extension line that extends from the second entity selected.
SUPPB	The minor word for suppressing both extension lines.
SUPP11	The minor word for suppressing the dimension arrow and associated extension line for the first entity selected.
SUPP22	The minor word for suppressing the dimension arrow and associated extension line for the second entity selected.
SUPPB1	The minor word for suppressing both extension lines and the dimension arrow for the first entity selected.
SUPPB2	The minor word for suppressing both extension lines and the dimension arrow for the second entity selected.

ANSI 1982 Dimensioning and Other Statements 20

ANSI 1982 Dimensioning and Other Statements 20

This chapter describes the dimensioning and other statements that work according to the 1982 ANSI standard. These statements imitate the operations found in menu 16 DRAFTING.

The same statement described elsewhere in this manual works only with the 1973 ANSI standard. For example, CDIMEN in this chapter applies to the 1982 ANSI standard. CDIMEN as described in chapter 18, ANSI 1973 Dimensioning and Other Statements, applies to the 1973 ANSI standard.

Also, some commands described in this chapter do not work at all with the 1973 standard. For example, DATUM is not implemented in the 1973 ANSI standard.

ADIMEN

The ADIMEN statement creates angular dimensions. Angular dimensions are generated so as to result in circular dimension lines with the radius automatically determined. The automatically generated label is the angle included by the lines measured in a counterclockwise direction from the first line entered. The label is generated as specified by the AUNITS modal statement. The text of the dimension is positioned by locating the origin (TXTJUS modal statement) with either absolute coordinates or as a delta distance from either another dimensioning entity or a point. If the TXTORG/DELTA modal statement is set, specifying only coordinates will position the origin a delta distance from the intersection of the two generating lines.

Statement format:

```
        ⎛XLARGE⎞                ⎛XLARGE⎞                 ⎛entity,xt,yt⎞ [,⎛'text'        ⎞]
ADIMEN/ ⎜YLARGE⎟ ,line 1, ⎜YLARGE⎟ ,line 2, ⎜  xt,yt    ⎟ [ ⎝text variable⎠]
        ⎜XSMALL⎟                ⎜XSMALL⎟                 ⎝              ⎠
        ⎝YSMALL⎠                ⎝YSMALL⎠
```

Parameter	Description
XLARGE YLARGE XSMALL YSMALL	The minor words for defining a location in the x positive, y positive, x negative, and y negative directions.
line 1,line 2	The names of the two lines that are dimensioned.
entity,xt,yt	The name of an entity and the xt- and yt-coordinates for the text origin of the text of the dimension.
xt,yt	The xt- and yt-coordinates for the text origin of the text of the dimension.
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of a text variable containing the text.

Example:

The following statement creates an angular dimension starting at the x positive end of LIN1 and ending at the y positive end of LIN2. The text origin is set at the absolute coordinates, x = −0.5 and y = 0.75.

```
AD1=ADIMEN/XLARGE,LIN1,YLARGE,LIN2, 0.5,0.75
```

NOTE

ADIMEN replaces the RDIMEN statement of the 1973 ANSI standard.

BALOON

The BALOON statement draws a balloon with an arrow pointing to an entity in the current drawing. A detail number and a sheet number can be displayed in the balloon.

The arrow from the balloon will touch the entity chosen at the parameter supplied after the entity name. Allowable entities are points, lines, arcs, conics and splines. If no radius is given, the last system default of (initially 0.5 in 12.7 mm) is used for the radius of the balloon. The center of the balloon is positioned with either absolute coordinates or as a delta distance from another dimensioning entity or point.

Statement format:

```
BALOON/entity,parameter[,RADIUS,radius],{ entity,xt,yt } ,DETAIL,detail number
                                        { xt,yt     }

    [,SHEET,sheet number]
```

Parameter	Description
entity	The name of the entity to which a balloon is attached.
parameter	The parameter along the entity at which the balloon is attached.
RADIUS	The minor word indicating the radius of the balloon.
radius	The radius of the balloon in units of measure.
entity,xt,yt	The name of an entity and the xt- and yt-coordinates for the text origin of the text of the dimension.
xt,yt	The xt- and yt-coordinates for the text origin of the text of the dimension.
DETAIL	The minor word indicating the detail number of the balloon.
detail number	The detail number which is a string or character variable of up to nine characters.
SHEET	The minor word indicating the sheet number of the balloon.
sheet number	The sheet number which is a string or character variable of up to nine characters.

Example:

The following statement creates a balloon attached at 0.25 units of measure along LN1. The balloon has a radius of 0.75 units of measure. The center of the balloon is set at delta coordinates from PT1 of x = 2 and y = 2. The detail number is 1.

```
BALOON/LN1,.25,RADIUS,.75,PT1,2,2,DETAIL,'1'
```

CDIMEN

The CDIMEN statement creates circular dimensions. A circular dimension label is generated with a leader line which can originate from the center of the arc pointing at the arc or can originate outside the arc pointing at the arc such that an extension or leader would pass through the center of the arc.

The text of the dimension is positioned by locating the origin (TXTJUS modal statement) with either absolute coordinates or as a delta distance from either another dimensioning entity or a point. If the TXTORG/DELTA modal statement is set, specifying only coordinates will position the origin a delta distance from the center of the circle being dimensioned.

The leader line includes a tail, which is a short horizontal bar extending from the beginning or end of the text. If the minor word START is specified, the tail is placed to the left of the text. If the minor word END is specified, the tail is placed to the right of the text.

If the modal ATAIL is set ON, the modifiers START and END are ignored and the leader origin is set automatically.

Statement format:

```
CDIMEN/{START},arc,{entity,xt,yt}[,{'text'        }]
       {END  }     {     xt,yt  }  [ {text variable}]
```

Parameter	Description
START	The minor word indicating the leader origin at the start of text.
END	The minor word indicating the leader origin at the end of text.
arc	The name of the arc being dimensioned.
entity,xt,yt	The name of an entity and the xt- and yt-coordinates for the text origin of the text of the dimension.
xt,yt	The xt- and yt-coordinates for the text origin of the text of the dimension.
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of a text variable containing the text.

Examples:

The following statement creates a circular dimension CD1 circle CIR1. The leader starts at the start of text. The text origin is set at absolute coordinates, x=3 and y=2.

```
CD1=CDIMEN/START,CIR1,3,2
```

The following statement creates a circular dimension CD2 for circle CIR2. The leader starts at the end of text. The text origin is set at delta coordinates, x=-2 and y=1.25 from point PT1.

```
CD2=CDIMEN/END,CIR2,PT1, 2,1.25
```

CLINE

The CLINE statement constructs centerlines.

Linear - Multiple Points

The following format creates a centerline between a series of up to 12 points. The effect will be a line, through the points, in which dashes are placed exactly at the midpoints.

If the points entered are not linear, the system displays:

```
CLINE POINTS NOT COINCIDENT
```

Statement format:

```
CLINE/point1,point2,...,pointn
```

Parameter	Description
point1 through pointn	The names of the points through which the centerline is defined.

Example:

Program Statement	Explanation
`CLN1=CLINE/PNT1,PNT2`	Centerline CLN1 is created through points PNT1 and PNT2.

Circular - Multiple Circles

The following format creates a centerline between a series of one to six circles. Crossed centerlines are automatically generated for each circle of a line of circles with colinear centers. The centerline through the centers places a dash at each center with the endpoints extending 0.1 in (2.54 mm) outside the end circles. Each circle also has a centerline generated normal to the line through all of the centers.

Statement format:

```
CLINE/circle1,circle2,...,circle
```

Parameter	Description
circle1 through circlen	The names of the circles through which the centerline is defined.

Example:

Program Statement	Explanation
`CLN2=CLINE/CIR2,CIR3,$` `  CIR4,CIRC5`	Centerline CLN2 is created through circles CIR2, CIR3, CIR4, and CIRC5.

Bolt Circle - Circular Array

The following format creates a centerline through a circular array. A centerline is created along the center of each element of a circular array. Crossed centerlines are automatically generated for each element just as in the multi-circle case.

Statement format:

```
CLINE/circular array
```

Parameter	Description
circular array	The name of the circular array.

Examples:

Program Statement	Explanation
CLN3=CLINE/AR	Centerline CLN3 is created through circular array AR.

CURARR

The CURARR statement draws an arrowhead at any position along an existing curve. Arrowheads created by this operation are considered triangles by the system and can be used the same way that a triangle is used.

Arrowhead at Curve End

The following format places the arrowhead at the indicated curve end. Valid entities are lines, arcs, conics, splines, composite curves, pointsets, three-dimensional splines, and machining curves.

Statement format:

```
        ⎧XLARGE⎫
CURARR/ ⎪YLARGE⎪ ,entity
        ⎪XSMALL⎪
        ⎩YSMALL⎭
```

Parameter	Description
XLARGE YLARGE XSMALL YSMALL	The minor words for defining a location in the x positive, x negative, y positive, and y negative directions.
entity	The name of the entity.

Example:

Program Statement	Explanation
CURARR/XLARGE,ARC1	An arrowhead is set at the x positive end of ARC1.

Arrowhead on Curve at a Parameter

The following format places the arrowhead along a two-dimensional curve at a specified parameter. The position indicator after the entity determines the direction along the curve at which the arrowhead is to point. Valid entities are lines, arcs, conics, and splines.

Statement format:

```
CURARR/PARAM,parameter,entity,{XLARGE}
                              {YLARGE}
                              {XSMALL}
                              {YSMALL}
```

Parameter	Description
PARAM	The minor word indicating a parameter along a curve.
parameter	The parameter expressed as a ratio along the curve.
entity	The name of the entity.
XLARGE YLARGE XSMALL YSMALL	The minor words for defining a location in the x positive, x negative, y positive, and y negative directions.

Example:

Program Statement	Explanation
`CURARR/PARAM,.75,LN1,XLARGE`	An arrowhead is set on curve LN1 at a ratio of 0.75 along the curve. The arrowhead points in the x positive direction.

DATFEA

The DATFEA statement displays a datum feature symbol and associated text within a feature frame on the drawing. If a connecting leader is to be produced, a datum reference entity must be given. Points, lines, arcs, splines, conics, and dimensions are allowable entities. The connection to the entity will be at a given parameter on the entity.

The connection line will join the feature frame at the midpoint of the left edge of the frame, the midpoint of the right edge of the frame, the lower left corner of the frame, the lower right corner of the frame, the upper left corner of the frame, or the upper right corner of the frame. An arrowhead can be drawn at the end of the connecting leader if desired.

The lower left corner of the datum feature symbol can be positioned with absolute coordinates, with an existing point, or above or below existing text of a note, label, dimension, or feature frame.

Statement format:

```
DATFEA/[ LEADR,entity,parameter,(LEFT|RIGHT|ULEFT|URIGHT|LLEFT|LRIGHT),(JOG|DIRECT|MODFY)[,ARRW],
       (entity,(ABOVE|BELOW)|xt,yt|point),('text'|text variable)
       ]
```

Parameter	Description
LEADR	The minor word for producing a leader line.
entity	The name of the entity to which the datum feature is attached.
parameter	A parameter expressed as a ratio at which the leader line is attached to the entity.
LEFT	The minor word for attaching the leader to the midpoint of the left side of the feature frame.
RIGHT	The minor word for attaching the leader to the midpoint of the left side of the feature frame.
ULEFT	The minor word for attaching the leader to the upper left corner of the feature frame.
URIGHT	The minor word for attaching the leader to the upper right corner of the feature frame.

Parameter	Description
LLEFT	The minor word for attaching the leader to the lower left corner of the feature frame.
LRIGHT	The minor word for attaching the leader to the lower right corner of the feature frame.
JOG	The minor word indicating that a horizontal line is drawn that is two times the text dimension distance followed by a line directly connected to the datum.
DIRECT	The minor word indicating a straight line is drawn from the datum to the feature frame.
MODFY	The minor word indicating that the position of the frame may be modified so that a horizontal or vertical line can be drawn.
ARRW	The minor word to produce an arrowhead at the end of the leader.
xt,yt	The absolute coordinates where the lower left corner of the datum feature symbol is placed.
point	The name of a point where the lower left corner of the datum feature symbol is placed.
ABOVE	The minor word indicating the datum feature symbol goes above existing text.
BELOW	The minor word indicating the datum feature symbol goes below existing text.
'text'	The datum feature that is a string of 1 character enclosed in single quotes.
text variable	The datum feature that is a text variable of 1 or 2 characters.

Example:

Program Statement	Explanation
`DATFEA/LEADR,PT1,LEFT,$` `  MODFY,2,2'A'`	A datum feature symbol is attached to PT1. The leader is attached to the midpoint of the lower left side of the feature frame. The lower left corner is set at coordinates, x=2 and y=2. The datum feature symbol is A.

DATUM

The DATUM statement defines a datum target symbol. The symbol is composed of arcs, pointsets, section lining, and notes all joined into one group entity.

The datum target may be specified by a point, line, circular area, or an existing entity. A reference number must be given to be placed in the lower half of the datum symbol circle. The target circle center can be positioned by locating the origin with absolute coordinates. If no radius is supplied, the default of 0.5 in (12.7 mm) is used.

The pen number used in drawing the datum target may be set differently than the current pen. However, the target symbol (circle, notes, and leader line) is always defined using the current pen number. The target pen number can be changed only for point with or without area, line, or circle target.

Point without Area

If a point without area is the target, a point must be given at which the target should be placed.

Statement format:

```
DATUM/NOAREA[,radius][,PENNUM,pen number],point,xt,yt,{'text'       }
                                                      {text variable}
```

Parameter	Description
NOAREA	The minor word indicating a datum target symbol without an area.
radius	The radius of the datum target symbol.
PENNUM	The minor word indicating the datum target pen number.
pen number	The pen number used to create the datum target.
point	The name of the point at which the datum target is placed.
xt,yt	The absolute coordinates of the datum target symbol.
'text'	The target identifier that is a string of 9 characters or less enclosed in single quotes.
text variable	The target identifier that is a text variable of 9 characters or less.

Example:

The following statement creates a datum target of a large X. The datum target symbol has a radius of 0.75 units. The target symbol is drawn with the number 2 pen. The target symbol is placed at point PT1. The target identifier is placed at absolute coordinates of x=3 and y=4. The target identifier is C.

```
DATUM/NOAREA,.75,PENNUM,2,PT1,3,4,'C'
```

Point with Area

If a point with area is the target, a point must be given at which the target should be placed and a diameter of the represented area to be written in the note above the line in the target symbol also must be given.

Statement format:

```
DATUM/AREA[,radius][,PENNUM,pen number],point,diameter,

    xt,yt,('text'       )
          (text variable)
```

Parameter	Description
AREA	The minor word for a datum target symbol with an area.
radius	The radius of the datum target symbol.
PENNUM	The minor word for the datum target pen number.
pen number	The pen number used to create the datum target.
point	The name of the point at which the datum target is placed.
diameter	The diameter of the represented area.
xt,yt	The absolute coordinates of the datum target symbol.
'text'	The target identifier that is a string of 9 characters or less enclosed in single quotes.
text variable	The target identifier that is a text variable of 9 characters or less.

Example:

The following statement creates a datum target of a large X. The datum target symbol has a radius of 0.25 units. The target symbol is placed at point PT1. The diameter of the area represented is 2.5 units. The target identifier is placed at absolute coordinates of x=3 and y=4. The target identifier is B.

```
DATUM/AREA,.25,PT1,2.5,3,4,'B'
```

Line

If a line is the target, points through which the line will be drawn must be given. A pointset is drawn in the phantom font through the two points and the extension-dimension distance is extended beyond the points. A parameter must be given to position the point on the line at which the target leader points.

Statement format:

```
DATUM/LIN[,radius][,PENNUM,pen number],point1,point2,parameter,

    xt,yt,('text'       )
          (text variable)
```

Parameter	Description
LIN	The minor word indicating a datum target symbol as a line.
radius	The radius of the datum target symbol.
PENNUM	The minor word indicating the datum target pen number.
pen number	The pen number used to create the datum target.
point1 point2	The names of the two endpoints of the line.
parameter	The parameter along the line at which the leader line is attached.
'text'	The target identifier that is a string of 9 characters or less enclosed in single quotes.
text variable	The target identifier that is a text variable of 9 characters or less.

Example:

The following statement creates a datum target of a line. The datum target symbol has a radius of 0.75 units. The datum target (the line) is drawn in a number 1 pen. The line is drawn between points PT23 and PT24. The leader is attached at a ratio of .25 along the line. The target identifier is placed at absolute coordinates of x=1 and y=1. The target identifier is contained in text variable TXT1.

```
DATUM/LIN,.75,PENNUM,1,PT23,PT24,.25,1,1,TXT1
```

Circle

If a circle is the target, the target circle center point must be supplied and the diameter of the target circle must be given. The angle of sectioning and the distance between section lines may be given otherwise the system default values will be used. The diameter value will be written in the upper half of the symbol circle.

Statement format:

```
DATUM/CIRC[,radius][,PENNUM,pen number],point,diameter[,ANGLE,angle]

    [,SECDIS,distance],xt,yt,{'text'      }
                             {text variable}
```

Parameter	Description
CIRC	The minor word indicating a datum target symbol as a section lined phantom circle.
radius	The radius of the datum target symbol.
PENNUM	The minor word indicating the datum target pen number.
pen number	The pen number used to create the datum target.
point	The name of a point used at the target circle center.
diameter	The diameter of the target circle.
ANGLE	The minor word indicating the angle of the section lines.
angle	The angle of the section lines.
SECDIS	The minor word indicating the distance between section lines.
distance	The distance between section lines in units of measure.
xt,yt	The absolute coordinates of the datum target symbol.
'text'	The target identifier that is a string of 9 characters or less enclosed in single quotes.
text variable	The target identifier that is a text variable of 9 characters or less.

Example:

The following statement creates a datum target of a section lined phantom circle. The datum target symbol has a default radius of 0.5 units. The datum target (the section lined phantom circle) is drawn in a number 4 pen. The target circle center is point PT12. The diameter of the target circle is 1.5 units. The angle of the section lines is 20 degrees. The distance between the section lines is 0.2 units. The target identifier is placed at absolute coordinates of x=4 and y=6. The target identifier is contained in text variable TXT2.

```
DATUM/CIRC,PENNUM,4,PT12,1.5,ANGLE,20,SECDIS,.2,4,6,TXT2
```

Existing Entity

If an existing entity is the target, an entity name and a parameter must be given. The parameter determines the point on the entity at which the leader points. Valid entities are points, lines, arcs, conics, and splines.

Statement format:

```
DATUM/ENTTY[,radius],entity,parameter,xt,yt,('text'
                                            (text variable)
```

Parameter	Description
ENTTY	The minor word indicating the datum target symbol as an existing entity.
radius	The radius of the datum target symbol.
entity	The name of an existing entity.
parameter	The parameter along the entity at which the leader line is attached.
xt,yt	The absolute coordinates of the datum target symbol.
'text'	The target identifier that is a string of 9 characters or less enclosed in single quotes.
text variable	The target identifier that is a text variable of 9 characters or less.

Example:

The following statement creates a datum target of an existing entity. The datum target symbol has a radius of 0.5 units (because the radius is not given, the system default is used). The parameter on the line where the leader line starts is 0.35. The leader line points at the start end of LN1. The target identifier is placed at absolute coordinates of x=5 and y=7. The target identifier is contained in text variable TXT2(1).

```
DATUM/ENTTY,LN1,.35,5,7,'D'
```

DDIMEN

The DDIMEN statement creates diameter dimensions. Diameter dimensions are generated so that dimension lines lie along a diameter of the circle. The direction is determined by the direction of a line from the circle center to the origin of the label (TXTJUS modal statement.)

The text of the dimension is positioned by locating the origin (TXTJUS modal statement) with either absolute coordinates or as a delta distance from either another dimensioning entity or a point. If the TXTORG/DELTA modal statement is set, specifying only coordinates will position the origin a delta distance from the center of the circle being dimensioned.

Statement format:

```
DDIMEN/arc,{entity,xt,yt}[,{'text'        }]
           {  xt,yt      }[,{text variable}]
```

Parameter	Description
arc	The name of the arc being diametrically dimensioned.
entity,xt,yt	The name of an entity and the xt- and yt-coordinates for the text origin of the text of the dimension.
xt,yt	the xt- and yt-coordinates for the text origin of the text of the dimension.
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of a text variable containing the text.

Example:

Program Statement	Explanation
`DDS=DDIMEN/CR1,6.25,3.5`	Diametric dimension DDS is created using CR1. The dimension is placed at absolute coordinates x=6.25 and y=3.5.

GEOTOL

The GEOTOL statement displays geometric tolerance or composite geometric tolerance symbols and associated text within a feature frame on the drawing.

If the frame is to be connected to an entity with a connecting leader, a datum reference entity must be given. Points, lines, arcs, conics, splines, and dimensions are allowable entities. The connection to the entity will be at a given parameter on the entity.

The connection line will join the feature frame at the midpoint of the left edge of the frame, the midpoint of the right edge of the frame, the lower left corner of the frame, the lower right corner of the frame, the upper left corner of the frame, or the upper right corner of the frame. An arrowhead may be drawn at the end of the connecting leader if desired.

The lower left corner of the feature frame symbol can be positioned by locating the origin with absolute coordinates, with an existing point, or above or below existing text of a note, label, dimension, or feature frame.

A geometric characteristic and symbol may be selected which will be drawn in the leftmost section of the frame.

If Profile of a Surface is chosen for the geometric characteristic and a leader line is desired, a symbol may be attached to the leader line signifying that the profile applies to surfaces all around the part. Using the minor word, AROUND, accomplishes this.

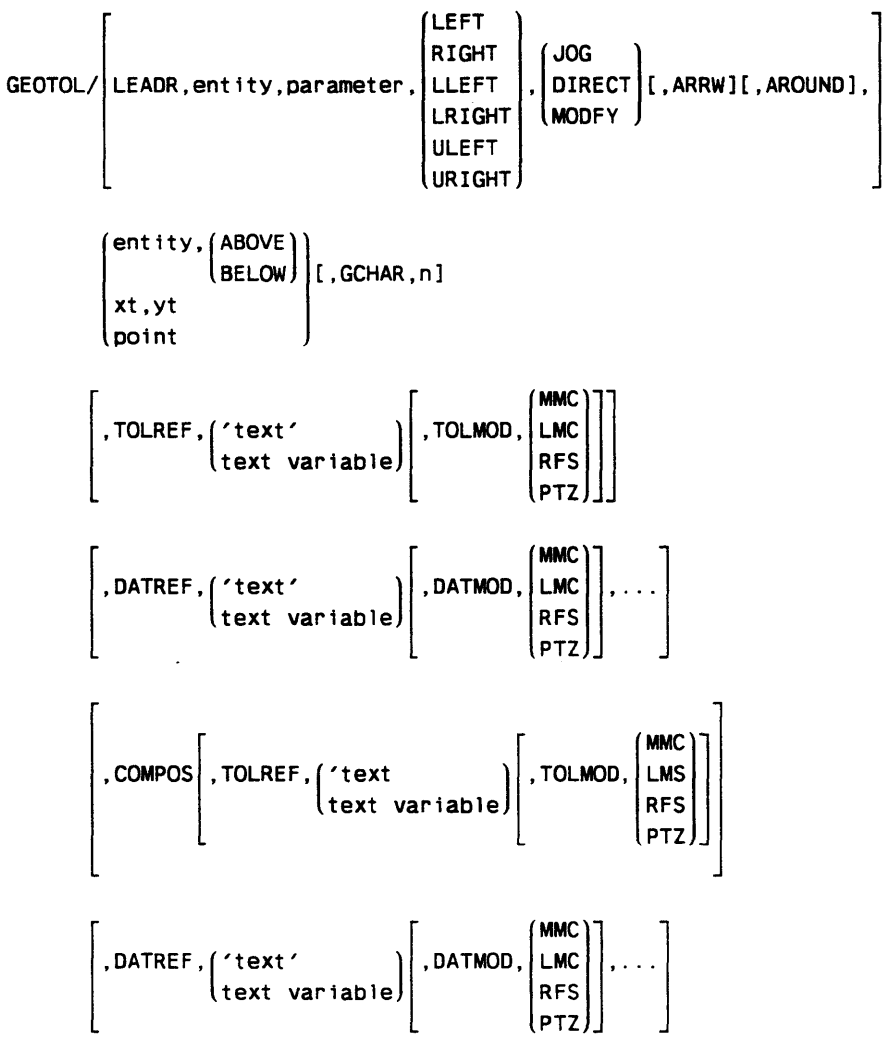
Tolerance and datum references can be entered. A total of five references may be entered with not more than one tolerance reference and not more than five datum references.

The modifiers are as follows:

- MMC - Maximum material condition.
- LMC - Least material condition.
- RFS - Regardless of feature size.
- PTZ - Projected tolerance zone.

If the minor word, COMPOS, is specified, the first datum references and tolerances will be entered in the top frame, and the datum references and tolerances following COMPOS will be entered in the bottom frame. Again, a total of five references may be entered in the bottom frame with not more than one tolerance reference and not more than five datum references. The modifiers are the same as above.

Statement format:



Parameter	Description
LEADR	The minor word for producing a leader line.
entity	The name of the entity to which a geometric tolerance is attached.
parameter	The parameter along the entity at which the geometric tolerance is attached.
LEFT	The minor word for attaching the leader to the midpoint of the left side of the feature frame.
RIGHT	The minor word for attaching the leader to the midpoint of the right side of the feature frame.
ULEFT	The minor word for attaching the leader to the upper left corner of the feature frame.
URIGHT	The minor word for attaching the leader to the upper right corner of the feature frame.

Parameter	Description
LLEFT	The minor word for attaching the leader to the lower left corner of the feature frame.
LRIGHT	The minor word for attaching the leader to the lower right corner of the feature frame.
JOG	The minor word indicating a horizontal line that is two times the text dimension distance followed by a line directly connected to the datum.
DIRECT	The minor word indicating a straight line from the datum to the feature frame.
MODFY	The minor word indicating that the position of the frame may be modified so that a horizontal or vertical line can be drawn.
ARRW	The minor word to produce an arrowhead at the end of the leader.
AROUND	The minor word indicating that the geometric characteristic is PROFILE and that the symbol indicating this is attached to the leader line.
entity	The name of the entity to which the geometric tolerance is attached.
ABOVE	The minor word indicating that the feature control frame is placed above an existing note, label, dimension or feature control frame.
BELOW	The minor word indicating that the feature control frame is placed below an existing note, label, dimension or feature control frame.
xt,yt	The absolute coordinates of the location of the lower left corner of the feature frame.
point	The name of a point used as the location of the lower left corner of the feature frame.
GCHAR	The minor word indicating the geometric characteristic.

Parameter	Description
n	The number of geometric characteristic. It is one of the following numbers.

Number	Characteristic
1	Straightness
2	Flatness
3	Circularity
4	Cylindricity
5	Profile of a line
6	Profile of a surface
7	Angularity
8	Perpendicularity
9	Parallelism
10	Position
11	Concentricity
12	Circular Runout
13	Total Runout

Parameter	Description
TOLREF	The minor word indicating the tolerance reference.
'text'	The text of the tolerance or datum reference enclosed in single quotes. The text can be a maximum of 20 characters.
text variable	The name of a variable that contains the text of the tolerance or datum reference. The text can be a maximum of 20 characters.
TOLMOD	The minor word indicating the tolerance modifier.
MMC	The minor word indicating maximum material condition (MMC) for a tolerance or datum reference.
LMC	The minor word indicating least material condition (LMC) for a tolerance or datum reference.
RFS	The minor word indicating regardless of feature size (RFS) for a tolerance or datum reference.
PTZ	The minor word indicating projected tolerance zone (PTZ) for a tolerance or datum reference.
DATREF	The minor word indicating a datum reference.
DATMOD	The minor word indicating a datum modifier.
COMPOS	The minor word indicating that all references and modifiers before this word are entered in the top frame, and all references and modifiers after this word are entered in the bottom frame.

Examples:

The following statement creates a geometric tolerance symbol for entity NT1.

```
GEOTOL/NT1,BELOW,GCHAR,2,DATREF,'ABC',DATMOD,MMC
```

The feature control frame is placed below the existing entity. The geometric characteristic is 2, indicating flatness. The datum reference is ABC and the datum modifier is MMC.

The following statement creates a geometric tolerance symbol with a leader line for entity LN1.

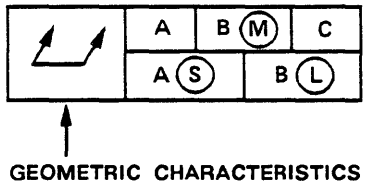
```
GEOTOL/LEADR,LN1,.5,LEFT,MODFY,ARRW,AROUND,$
     PT1,GCHAR,6,TOLREF,TXT1,TOLMOD,MMC,$
     DATREF,TXT2,DATMOD,MMC,COMPOS,DATREF,$
     TXT3,DATMOD,LMC,DATREF,TXT4
```

The leader is attached to the entity at a ratio of 0.5 along the entity and attached to the left side of the feature frame. The position of the feature frame can be modified so that either a horizontal or vertical line can be drawn. The leader has an arrowhead. The geometric characteristic is PROFILE and that symbol is used.

The geometric characteristic is 6 indicating the profile of a surface. The text for the tolerance reference is contained in text variable TXT1, and the tolerance modifier is MMC. The datum reference is contained in text variable TXT2, and the datum modifier is MMC. All this information is placed in the top frame. The following information is placed in the bottom frame: a datum reference that is contained in text variable TXT3, a datum modifier LMC, and a datum reference that is contained in text variable TXT4.

The geometric characteristic is in both the top and bottom frames.

For example:



NOTE

At least one characteristic or reference is required for geometric tolerances. For composite geometric tolerances, one characteristic and at least one reference on the top and one reference on the bottom are required.

LABEL

The LABEL statement constructs labels. A general label consists of a string of characters with an associated leader line generated from the label to an entity on the drawing. The leader includes a tail, which is a short horizontal bar extending from the beginning or end of the text. If the minor word START is specified, the tail is placed to the left of the text. If the minor word END is specified, the tail is placed to the right of the text. If modal statement ATAIL has been set to ON, the tail is placed automatically.

The vertical position of the leader line tail is determined by the modal statement LEADER. The leader touches the first text line by default. The leader line automatically terminates at the entity through one of the following options:

- Coordinate method - Coordinates at or near the desired leader line termination point may be entered. These coordinates need not be directly on the entity, as the leader line will automatically intersect itself precisely with the entity. If the entity is a point, the leader will terminate at the point.
- Slope method - The angle of the leader line is entered. The leader line will be generated to the entity indicated at the angle specified. The leader line automatically terminates at the intersection point on the entity.
- Parameter method - A parameter is entered. The leader line automatically terminates at the parameter entered for the entity.
- Midpoint method - If no leader line termination control is used, the leader line will automatically end at the midpoint of the entity.

The text of the label is positioned by locating the origin (TXTJUS modal statement) with either the absolute coordinates, or a delta distance from either another dimensioning entity or a point. The text angle of the label may be varied by entering ANGLE and the text angle in degrees. The string of characters which form the text of the label may be replaced by a text variable.

Statement format:

```
LABEL/{START}{entity,xt,yt}{entity
      {END  }{  xt1,yt1   }

      [,[{SLOPE,slope}][,ANGLE,angle],{'text'       }
        [{   xt,yt    }]              {text variable}
        [{parameter   }]
```

Parameter	Description
START	The minor word indicating the leader origin at the start of text.
END	The minor word indicating the leader origin at the end of text.
entity,xt,yt	The name of an entity and the xt- and yt-coordinates for the text origin of the text of the dimension.
xt1,yt1	The xt- and yt-coordinates for the text origin of the text of the dimension.
SLOPE	The minor word indicating the angle of the leader from the positive x-axis.

Parameter	Description
slope	The angle of the slope of the leader.
xt,yt	The absolute coordinates of the point at or near the point on the entity to which the leader is attached.
parameter	A parameter expressed as a ratio at which the leader is attached.
ANGLE	The minor word indicating the angle of the dimension text.
angle	The angle of the dimension text.
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of a text variable containing the text.

Example:

The following statement creates a label LAB1. The leader line starts at the beginning of text. The text origin is at absolute coordinates (2,4). The entity that is labeled is line LN2.

```
LAB1=LABEL/START,2,4,LN2,'CHAMFER .625 X 45 DEGREES'
```

LDIMEN

The LDIMEN statement creates linear dimensions. Linear dimensions are horizontal, vertical, or parallel. Horizontal dimensions measure the horizontal distance between the ends of two entities. Vertical dimensions measure the vertical distance between the ends of two entities.

There are two types of parallel dimensions.

The regular kind, designated PARLEL, measures the two-dimensional distance between the ends of two entities. This type of parallel dimension is generated relative to a given line or the apparent line that would exist between these two entity endpoints.

The second type of parallel dimension measures the normal distance between two parallel lines. This type of parallel dimension is designated PARNOR. When selecting the entities to dimension, a position indicator must follow the entity. The two exceptions to this rule are first, if the entity is a point, no position indicator is needed and second, if only one entity is being dimensioned, then just the entity need be used, as the entity's endpoints are used for the dimension.

If it is desirable to align the arrowheads of this dimension with those of a previously defined dimension, a YES followed by the drafting entity that the new dimension is to be aligned with must be entered.

The text of the dimension is positioned by locating the origin (TXTJUS modal statement) with either absolute coordinates or as a delta distance from either another dimensioning entity or a point. If the TXTORG/DELTA modal statement is set,the coordinates used are not relative to the origin 0,0. Rather, they are relative to one of the two entity endpoints which form the dimension.

Consider a counterclockwise arc drawn through three points, the entity endpoints and the text origin point, such that the text origin point is the second or middle point. The first point of this arc is the one from which a delta xt value and delta yt value (the input coordinates) are used to find the dimension origin.

The text of the dimension may be either a string of characters or a text variable if it is being entered manually.

Definition form:

```
Dimension type,entity selection[,arrowhead alignment],
     dimension origin[,dimension text]
```

Statement format:

```
         (HORIZ )  (              point,point                   )
         |VERTCL|  |entity,position ind,entity,position ind     |
LDIMEN/  |PARLEL|, |     entity,position indicator,point        |
         |PARNOR|  |     point,entity,position indicator        |
         (      )  (              entity                        )

      [,YES,ref entity],(entity,xt,yt)[,('text         )]
                        (     xt,yt   ) [ (text variable)]
```

Parameter	Description
HORIZ	The minor word for a horizontal dimension.
VERTCL	The minor word for a vertical dimension.
PARLEL	The minor word for a parallel dimension relative to a given line.
PARNOR	The minor word for a parallel dimension normal to two given parallel lines.
point,point	The names of the points that are dimensioned.
entity	The name of the entity that is dimensioned.
pos ind	The position indicator showing at which end of an entity the dimension is taken. It can be any of the following minor words.

Word	Description
XSMALL	Indicates the x negative end.
XLARGE	Indicates the x positive end.
YSMALL	Indicates the y negative end.
YLARGE	Indicates the y positive end.

Parameter	Description
YES	The minor word for aligning the arrowheads.
ref entity	The name of an entity used to align the arrow heads.
entity,xt,yt	The name of an entity and the xt- and yt-coordinates for the text origin of the text of the dimension.
xt,yt	the xt- and yt-coordinates for the text origin of the text of the dimension.
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of a text variable containing the text.

Examples:

The following statement creates a horizontal linear dimension for LN1. The text origin is at absolute coordinates (−1.5,1.75).

```
LD1=LDIMEN/HORIZ,LN1,-1.5,1.75
```

The following statement creates a horizontal linear dimension between the XLARGE end of LN1 to PT7. The arrowheads are aligned to the reference entity (YES). The text origin is at absolute coordinates (−2,2). The reference entity is ENT3.

```
LD2=LDIMEN/HORIZ,LN1,XLARGE,PT7,YES,ENT3,-2,2
```

The following statement creates a vertical linear dimension between PT7 and the YSMALL end of SPL3. The text origin is at a delta distance (5,−3) from LD2.

```
LD3=LDIMEN/VERTCL,PT7,SPL3,YSMALL,LD2,5,-3
```

MAGNFY

The MAGNFY statement produces a magnified drawing of a circular area. The entities that can be magnified are points, lines, arcs, conics, splines, pointsets, centerlines, and section lining. Automatic dimensioning performed on magnified entities reflects the size of the original, unmagnified entities.

The area to be magnified may be within an existing circle or within an area defined by a center and a radius. In the latter case, a circle is created around the area to be magnified.

The magnified reproduction may be drawn within an existing circle, within an area defined by a center and a radius, or within an area defined by a center and a magnification size.

For the last two options, a border is drawn around the magnified drawing if the minor word BORDER is specified.

Statement format:

```
MAGNFY/{CIRC,circle1 }, {CIRC,circle2                          }
       {xt,yt,radius }  {  xt,yt,{RADIUS,radius}[,BORDER]      }
                                 {ratio         }               }
```

Parameter	Description
CIRC	The minor word indicating a circle parameter.
circle1	The name of a circle defining the area to magnify.
xt,yt	The absolute coordinates of a circle center.
RADIUS	The minor word indicating a radius parameter.
radius	The radius in units of measure.
circle2	The name of a circle defining the magnified reproduction.
BORDER	The minor word for drawing a border around the reproduction.
ratio	The magnification size of the the reproduction.

Example:

The following statement reproduces and magnifies the area defined in ARC1 and places the center of the magnified reproduction at absolute coordinates x=2.5 and y=3. The magnification ratio is 2. A border is drawn around the magnified drawing.

```
MAGNFY/CIRC,ARC1,2.5,3,2,BORDER
```

MODDFT

The MODDFT statement modifies certain drafting entities without having to redefine those entities.

BASIC

Using the minor word BASIC, a rectangle can be added or deleted around the text in a dimension.

Statement format:

```
MODDFT/BASIC,{ADD},dimension entity
             {DEL}
```

Parameter	Description
BASIC	The minor word for adding or deleting a rectangle around text in a dimension.
ADD	The minor word to add a rectangle around the dimension text.
DEL	The minor word to delete a rectangle from around the dimension text.
dimension entity	The name of the dimensioning entity at which the text is altered.

Example:

The following statement creates a linear dimension LD1.

```
LD1=LDIMEN/VERTCL,LN1,PT1,3,1
```

The next statement modifies LD1 by adding a rectangle around the dimension text.

```
MODDFT/BASIC,ADD,LD1
```

REF

Using the minor word REF, you can add or delete parentheses to or from the text in a dimension.

Statement format:

```
MODDFT/REF,{ADD},dimension entity
           {DEL}
```

Parameter	Description
REF	The minor word for adding or deleting parentheses from the dimension text.
ADD	The minor word to add parentheses around the dimension text.
DEL	The minor word to delete parentheses from around the dimension text.
dimension entity	The name of the dimensioning entity at which the text is altered.

Example:

The following statement creates a linear dimension LD2.

```
LD2=LDIMEN/HORIZ,PT1,PT2,3,5
```

The next statement adds parentheses around the dimension text of LD2.

```
MODDFT/REF,ADD,LD2
```

TOLER

Using the minor word TOLER, you can add a tolerance to the text in a dimension. The tolerance can be entered in any of the following formats:

+tolerance 1	A positive value is entered as the top part of the tolerance.
-tolerance 1	A negative value is entered as the top part of the tolerance.
+tolerance 1, -tolerance 2	The tolerances are written one above the other immediately following the text in the dimension.
±tolerance 1	A positive/negative value is entered as the bottom part of the tolerance.

The tolerances are entered as characters but stored as real numbers in the database. If any characters following the plus or minus sign are not numerals, the following error message will be displayed:

```
CONVERSION ERROR OCCURRED
```

Tolerance characters are displayed with the height determined by the tolerance ratio (CDISPL/TOLER,n).

If limits are chosen instead of a tolerance, the upper limit is the dimension and the positive value, and the lower limit is the remainder after subtracting the negative value from the dimension.

NOTE

CDISPL/TOLER,n (tolerance ratio) does not affect the ± tolerance.

Statement format:

```
MODDFT/TOLER,dimension entity,'tolerance 1',[,'tolerance 2'][,LIMIT]
```

Parameter	Description
TOLER	The minor word for adding a tolerance value to a dimension text.
dimension entity	The name of the dimensioning entity at which the text is altered.
'tolerance 1'	The first tolerance value enclosed in quotes. It is entered as the top part of the tolerance.
'tolerance 2'	The second tolerance value enclosed in quotes. It is entered as the bottom part of the tolerance.
LIMIT	The minor word for adding limits to the dimension text.

Example:

Program Statement	Explanation
MODDFT/TOLER,LD3,'.05',' .05'	A tolerance of ±0.05 (character strings) is added after dimension LD3.

NOTE

The NOTE statement creates general notes. General notes consist of continuous strings of characters. The note is entered as a string of characters and may contain up to 200 characters. The character string may be replaced by a text variable or a real variable.

A general note is positioned by locating the origin (TXTJUS modal statement) with either the absolute coordinates or as a delta distance from either another dimensioning entity or a point.

The text of the note may be varied by entering ANGLE and the text angle in degrees. If a real variable is given instead of a text variable, the real number is truncated and converted into a text string.

The note may be parallel to a line by entering the line and the direction with respect to the Xt-axis. POSITV is for the positive Xt direction and NEGATV is for the negative Xt direction. For a vertical line, POSITV will cause the text to be drawn upwards.

The note may be parallel to an arc by entering the arc and the direction of rotation of the text. CW is for clockwise and CCW is for counterclockwise.

Statement format:

```
                              {{POSITV},line}
                              {{NEGATV}     }   {'text'       }
NOTE/{entity,xt,yt},          {{CW    },arc }  ,{text variable}
     {    xt,yt   }          ,{{CCW   }     }   {real variable}
                              {ANGLE,angle  }
```

Parameter	Description
entity,xt,yt	The name of an entity and the xt- and yt-coordinates for the text origin of the text of the note.
xt,yt	The xt- and yt-coordinates for the text origin of the text of the note.
POSITV	The minor word for positioning the note parallel to a reference line in the positive direction from the x-axis.
line	The name of the reference line.
NEGATV	The minor word for positioning the note parallel to a reference line in the negative direction from the x-axis.
CW	The minor word for positioning the text parallel to a given arc in the clockwise direction.
arc	The name of the reference arc.
CCW	The minor word for positioning the text parallel to a given arc in the counterclockwise direction.

Parameter	Description
ANGLE	The minor word for the angle of the dimension text.
angle	The angle of the dimension text.
'text'	The text of the note. The text must be enclosed in single quotes.
text variable	The name of a text variable containing the text.
real variable	The name of a real variable that is converted to text.

Examples:

The following statement creates a note at absolute coordinates (5,5) parallel to line LN01 with the text GENERAL NOTE.

```
NT1=NOTE/5,5,POSITV,LN01,'GENERAL NOTE'
```

The following statements prompt for text that is entered in text variable TXT, and if a] is entered, note NT2 is placed at a delta distance (0,-2.5) from note NT1. The text angle is 10 degrees and the text is retrieved from text variable TXT. If something other than a] is entered, the program goes to statement label 50.

```
TEXT/'ENTER STRING',TXT,STAT
IF(STAT.EQ.0)GOTO 50
NT2=NOTE/NT1,0, 2.5,ANGLE,10,TXT
```

SECARR

The SECARR statement creates cross-section arrows for declaring details or cutouts for drafting.

The following data must be supplied in order to create a section arrow:

- THICK - The thickness of the section arrow's main body.
- LENGTH - The length of the section arrowhead.
- WIDTH - The width of the section arrowhead.
- NECK - The minimum distance between an arrowhead and the first jog of the main body.
- LEVL - The level on which the section arrows are to be defined. The default is the current level. (optional)

The arrowhead may be placed at the start of the dimension, at the end of the dimension, at both ends of the dimension, or may be eliminated entirely.

A section arrow is defined as a point set curve and can be section lined in a separate process.

Statement format:

```
SECARR/THICK,n1,LENGTH,n2,WIDTH,n3,NECK,n4

[,LEVL,n],{START}  ,{xt1,yt1,xt2,yt2,...,xtn,ytn      }
         {END   }  {NUMBER,number,coordinate array}
         {BOTH  }
         {NONE  }
```

Parameter	Description
THICK	The minor word indicating the thickness of the section arrow's main body.
n1,n2,n3,n4	The value for the parameters: THICK, LENGTH, WIDTH, NECK or LEVL.
LENGTH	The minor word indicating the length of the section arrowhead.
WIDTH	The minor word indicating the width of the section arrowhead.
NECK	The minor word indicating the minimum distance between an arrowhead and the first jog of the main body.
LEVL	The minor word indicating the level on which the section arrows are to be defined. The default is the current level. (Optional)

Parameter	Description
START	The minor word for placing an arrowhead at the start of the dimension.
END	The minor word for placing an arrowhead at the end of the dimension.
BOTH	The minor word for placing an arrowhead at both the start and the end of the dimension.
NONE	The minor word indicating no section arrowheads.
xt,yt	The absolute coordinates of a series of points. There must be between 4 and 50 absolute coordinates.
NUMBER	The minor word indicating the number of entities in an array.
number	The number of elements in the array. It is a value of at least 4 and not more than 50.
coordinate array	The name of a real coordinate array.

Example:

The following statement creates a section arrow with a thickness of 0.7 units of measure, a length of 2 units, a width of 3 units, and a neck of 2 units. The section arrow is assigned to level 1. Arrows are placed at both ends of the entity. The jog positions are at absolute coordinates (3,3) and (4,7).

```
SECARR/THICK,.7,LENGTH,2,WIDTH,3,NECK,2,LEVL,1,BOTH,3,3,4,7
```

SECTON

The SECTON statement automatically generates section lines in the work plane for any closed configuration. The spacing between the lines is adjustable as is the angle at which the lines are generated. If the distance between lines is omitted, 0.25 inch (6.35 mm) is assumed. If the angle for lines is omitted, 45 degrees is assumed. The angle for a set of section lines can be varied between 0 degrees and 180 degrees. The boundary lines can consist of lines, arcs, general conics, and splines. Islands are automatically recognized if they are contained in the list of entities or entity array.

With the YES parameter, selection of a previously defined section lining entity to align the new section lining with is allowed.

Statement format:

```
SECTON/{NUMBER,number,entity array }[,SECDIS,distance][,ANGLE,angle]
       {entity1,entity2,...,entityn}

    [,YES,ref entity]
```

Parameter	Description
NUMBER	The minor word indicating the number of entities in an array.
number	The number of entities in the array. It is a value of at least 2 and not more than 50.
entity array	The name of a entity array.
entity	The names of a series of entities.
SECDIS	The minor word indicating the distance between section lines.
distance	The distance between section lines in units of measure.
ANGLE	The minor word indicating the angle of the section lines.
angle	The angle of the section lines.
YES	The minor word for aligning the section lines to previous section lines.
ref entity	The name of an entity used to align the section lines.

Examples:

The following statements create section lines.

First, the material is set to brass.

```
MATERL/BRASS
```

Then, the section areas are defined.

The first section area is defined by LN3, CIR3, LIN4, LIN5, LIN7, and LIN8. The section lines are drawn at an angle of 135 degrees.

```
SEC1=SECTON/LIN3,CIR3,LIN4,LIN5,LIN7,LIN8,ANGLE,135
```

The second section area is defined by LIN7, LIN6, LIN2, and LIN13. The distance between section lines is 0.18 units of measure, and the lines are drawn at an angle of 30 degrees.

```
SEC2=SECTON/LIN7,LIN6,LIN2,LIN13,SECDIS,.18,ANGLE,30
```

SRFTEX

The SRFTEX statement displays the standard basic symbol for surface texture either displaced from an entity or entities and joined to the entities by arrows, or attached directly to the entity.

The value of Ra (roughness) can be in micrometers for SI units or microinches for U.S. customary units. When using SI units, the system displays the number of decimal places specified by DECMAL/n. When using U.S. customary units, Ra is displayed as a whole number. Entering 0 produces a symbol with no text.

The position where the arrow or arrows attach to the symbol base can be determined with absolute coordinates.

Basic Symbol Attached to Arrow

Using the minor words BASIC and ARRW, a surface texture symbol and its base can be displayed, displaced from an entity or entities or joined to entities with one or more arrows.

A parameter must be given with each two-dimensional curve to indicate the approximate position of the end point of the arrow on the curve. Up to ten entities and parameters can be specified.

The surface texture symbol can be displayed above a horizontal base or to the left of a vertical base. If the minor word LEFT is specified, the starting point of the arrow will be to the left of the symbol base if it is horizontal, or at the bottom of the symbol base if it is vertical. If the minor word RIGHT is specified, the starting point of the arrow will be to the right of the symbol base if it is horizontal, or at the top of the symbol base if it is vertical.

Statement format:

```
SRFTEX/BASIC,ARRW,roughness,entity1,parameter1,entity2,parameter2,...,

    entityn,parametern,{HORIZ },{LEFT },xt,yt
                       {VERTCL}{RIGHT}
```

Parameter	Description
BASIC	The minor word for adding a surface texture symbol.
ARRW	The minor word for adding arrows to the surface texture symbol.
roughness	The value of roughness (Ra).
entity	The names of a series of entities to which the surface texture symbol is attached. The number of entities can be less than or equal to 10.
parameter	A parameter expressed as a ratio at which the surface texture symbol is attached. The number of parameters must equal the number of entities given.
HORIZ	The minor word indicating a horizontal base.
VERTCL	The minor word indicating a vertical base.
LEFT	The minor word for placing the starting point of the arrow to the left of the entity for a horizontal base and below the entity for a vertical base.
RIGHT	The minor word for placing the starting point of the arrow to the right of the entity for a horizontal base and above the entity for a vertical base.
xt,yt	The absolute coordinates of the starting point of the arrow.

Example:

The following statement creates a surface texture symbol with arrows for CIR1, CIR2, and LN1 ending at the parameters given after each entity. The roughness factor (Ra) is 63. The symbol has a horizontal base and the starting point of the arrow is placed to the left of the entity at absolute coordinates (8,10).

```
SRFTEX/BASIC,ARRW,63,CIR1,3.14,CIR2,1.57,LN1,.5,HORIZ,LEFT,8,10
```

Basic Symbol Attached to Entity

Using the minor words BASIC and ENT, a symbol can be displayed directly attached to an entity. The symbol can be displayed horizontally above the entity, vertically to the left of the entity, or vertically to the right of the entity. A parameter must be given with the entity to determine the position of the symbol on the entity.

Statement format:

```
SRFTEX/BASIC,ENTTY,roughness,{ABOVE|RIGHT|LEFT},entity,parameter
```

Parameter	Description
BASIC	The minor word for adding a surface texture symbol.
ENTTY	The minor word for attaching a surface texture symbol to an entity.
roughness	The value of roughness (Ra).
ABOVE	The minor word for placing the symbol horizontally above the entity.
RIGHT	The minor word for placing the symbol vertically to the right of the entity.
LEFT	The minor word for placing the symbol vertically to the left of the entity.
entity	The name of an entity to which the surface texture symbol is attached.
parameter	A parameter expressed as a ratio at which the surface texture symbol is attached. The number of parameters can be less than or equal to zero.

Example:

The following creates a surface texture symbol attached to LN1. The roughness factor (Ra) is 63. The symbol is placed horizontally above the entity. The symbol is attached on the entity at a point that is a ratio of 0.5 along the entity.

```
SRFTEX/BASIC,ENTTY,63,ABOVE,LN1,.5
```

TAPER

The TAPER statement creates a slope or taper dimension drawn to two lines. Either a flat or conical taper dimension can be specified. The first line specified is opposite the side where the taper dimension will be placed. The second line specifies the line to which the dimension arrow points. To draw an arrow that points to the second line, give a parameter to position the arrow. The slope of the leader can be entered instead of a parameter. The leader angle is measured in a counterclockwise direction.

The text of the dimension is positioned by locating the origin (TXTJUS modal statement) with either absolute coordinates or as a delta distance from either another dimensioning entity or a point.

Statement format:

```
TAPER/(CONIC),line1,line2,(parameter),(entity,xt,yt)[,('text'        )]
      (FLAT )             (SLOPE,angle)( xt,yt      )[ (text variable)]
```

Parameter	Description
CONIC	The minor word indicating a conical taper dimension.
FLAT	The minor word indicating a flat taper dimension.
line1	The first line being dimensioned.
line2	The second line being dimensioned.
parameter	The parameter along the second line at which the arrow is drawn from the first line to the second line.
SLOPE	The minor word for specifying the slope method of drawing an arrow from the first line to the second line.
angle	The angle of the slope of the arrow.
entity,xt,yt	The name of an entity and the xt- and yt-coordinates for the text origin of the text of the dimension.
xt,yt	the xt- and yt-coordinates for the text origin of the text of the dimension.
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of a text variable containing the text.

Example:

The following statement creats a conical taper dimension between LN1 and LN2. The arrow is placed using slope method and an angle of 45 degrees. The text origin is set using PT1 and delta coordinates (0,1).

```
TAPER/CONIC,LN1,LN2,SLOPE,45,PT1,0,1
```

THIKNS

The THIKNS statement produces a thickness dimension between two curves. The dimension is taken from a point on the first curve to an imaginary point on the second curve. The point on the second curve is located by dropping an imaginary line from the point on the first curve normal to the second curve. The point at which the imaginary line intersects the second curve is the second point of the dimension. When selecting entities to be dimensioned, coordinates must be entered as close as possible to the point on the first curve where the measurement is to be made. A parameter on this curve may be entered instead of coordinates.

The text of the dimension is positioned by locating the origin (TXTJUS modal statement) with either absolute coordinates or as a delta distance from either another dimensioning entity or a point. If the TXTORG/DELTA modal statement is set, specifying only coordinates will position the origin a delta distance from the point on the first curve where the measurement was made.

Statement format:

```
THIKNS/entity1,(parameter),entity2,(entity,xt,yt)[,['text'       ]]
               (xt1,yt1   )          (   xt,yt  )  [(text variable)]
```

Parameter	Description
entity1	The name of the first entity being dimensioned.
parameter	The parameter expressed as a ratio along the first entity at which the dimension is taken.
xt1,yt1	The absolute coordinates at which the dimensioned is taken.
entity2	The name of the second entity being dimensioned.
entity,xt,yt	The name of an entity and the xt- and yt-coordinates for the text origin of the text of the dimension.
xt,yt	The xt- and yt-coordinates for the text origin of the text of the dimension.
'text'	The text of the dimension. The text must be enclosed in single quotes.
text variable	The name of a text variable containing the text.

Example:

The following statement creates a thickness dimension between SPL1 and CIR1. The dimension is taken at or near the absolute coordinates of x = 5 and y = 3 for SPL1 and the normal from this point to CIR1. The origin of the text is at coordinates (1,2). The dimension text is DISTANCE FROM SPLINE TO ARC.

```
THIKNS/SPL1,5,3,CIR1,1,2,'DISTANCE FROM SPLINE TO ARC'
```

Numerical Control Statements 21

Numerical Control Statements 21

This chapter describes statements that control numerical control functions. These statements are similar in operation to some functions found in menu 17 NUMERICAL CONTROL.

SETGPG

The SETGPG statement modifies the N/C parameters in the Generation Parameter Group (GPG) used during toolpath generation. The GPG parameters being changed:

- Must have been previously created, and
- Must be either in a local text file or attached to an existing toolpath.

All GPG parameters can be defined with this statement. Refer to the ICEM Numerical Control manual for more information.

Using a Local Text File GPG

Statement format:

```
                  ⎧CONTUR⎫
                  ⎪ROUGH ⎪
SETGPG/LATHE,     ⎪DRILL ⎪ ,('gpg name')[,('local file')]
                  ⎩THREAD⎭   ( gpgvar  )[,(   lfnvar   )]
```

Parameter	Description
LATHE	The minor word indicating lathe toolpath parameters.
CONTUR	The minor word indicating a lathe contouring GPG.
ROUGH	The minor word indicating a lathe roughing GPG.
DRILL	The minor word indicating a lathe drilling PGP.
THREAD	The minor word indicating a lathe threading GPG.
'gpg name'	The name of the GPG to retrieve from the local text file.
gpgvar	A text variable that contains the name of the GPG to retrieve from the local text file.
'local file'	The name of an external local text file. The name must be enclosed in quotes.
lfnvar	A text variable containing the name of an external local text file.

The following self-explanatory error messages can occur from improper use of the SETGPG statement and an external local file.

```
GPG FILE READ ERROR

GPG NOT COMPATIBLE WITH ENTITY TYPE

GENERATION PARAMETER GROUP NOT FOUND

GPG STATEMENT ERRORS, SEE FILE - GPGERR
```

Example:

Program Statement	Explanation
`SETGPG/LATHE,THREAD,'GPG2','THD'`	The thread generation parameter group is named GPG2 and is located in local file THD.

Using an Existing Toolpath GPG

Statement format:

```
SETGPG/LATHE,{CONTUR}
             {ROUGH }
             {DRILL }
             {THREAD},toolpath
```

Parameter	Description
LATHE	The minor word indicating lathe toolpath parameters.
CONTUR	The minor word indicating a lathe contouring GPG.
ROUGH	The minor word indicating a lathe roughing GPG.
DRILL	The minor word indicating a lathe drilling GPG.
THREAD	The minor word indicating a lathe threading GPG.
toolpath	The local GPL name of an existing toolpath entity.

The following self-explanatory error messages can occur from improper use of the SETGPG statement using an existing toolpath.

```
ENTITY NOT FOUND

INCOMPATIBLE ENTITY USED IN DEFINITION
```

Example:

Program Statement	Explanation
`ENTITY/T143`	Define entity T143.
`REAL/S`	Define status variable.
`OBTAIN/ENTPTR,'TL143'$` `,T143,S`	Obtain the pointer of existing toolpath TL143 and assign it to T143.
`SETGPG/LATHE,DRILL,T143`	Replace the current drilling GPG with the GPG from toolpath TL143.

TLPATH

The TLPATH statement provides the ability to define toolpath entities for lathe contouring, roughing, drilling, or threading operations. Contouring, roughing, and threading toolpaths are generated along part geometry (specified as a list of entities). Drilling toolpaths do not require any geometric entities. Each toolpath reflects the current GPG settings for that operation. Interactive prompts are not allowed with this statement. Therefore, any GPG settings of PROMPT are changed to default values.

The following self-explanatory error messages can occur from improper use of the TLPATH statement.

```
BLANK ENTITY NOT FOUND

CONTOUR ENTITY NOT FOUND

NO CLOSED BOUNDARIES COULD BE FOUND INDETERMINATE

ENTITY NOT FOUND

INCOMPATIBLE ENTITY USED IN DEFINITION

DEFINITION START

INVALID NUMBER SUPPLIED

GPG NOT COMPATIBLE WITH ENTITY TYPE
```

CONTUR

Statement format:

```
tpname=TLPATH/LATHE,CONTUR,(ent1,ent2,...,entn       )[,START,(xt,yt )]
                           (NUMBER,number,ent array)[       (point)]
```

Parameter	Description
tpname	The name assigned to the defined toolpath.
LATHE	The minor word indicating lathe toolpath parameters.
CONTUR	The minor word indicating a lathe contouring toolpath.
ent1 through entn	A list of entities used to create a toolpath. The entities can be lines, arcs, conics, two-dimensional and three-dimensional splines, bezier curves, point sets, and machine curves. The limit of the total number of entities is 64.
NUMBER	The minor word indicating the number of entities in the entity array.
number	The number of entities in the entity array. The maximum number of entities is 100.
ent array	The name of an entity array.
START	The minor word indicating the starting entity selection point.
xt,yt	A space location expressed in transform coordinates indicating the starting entity selection point.
point	The name of a point indicating the starting entity selection point.

Example:

The following statement creates a lathe contour toolpath using 21 entities contained in the entity array AR33. The new toolpath is named T322.

```
T322=TLPATH/LATHE,CONTUR,NUMBER,21,AR33(1)
```

ROUGH

Statement format:

```
tpname=TLPATH/LATHE,ROUGH

BLNK(ent1,ent2,...,entn
     (NUMBER,number,ent array)

CONTUR(ent1,ent2,...,entn
       (NUMBER,number,ent array)
```

Parameter	Description
tpname	The name assigned to the defined toolpath.
LATHE	The minor word indicating lathe toolpath parameters.
ROUGH	The minor word indicating a lathe roughing toolpath.
BLNK	The minor word indicating the blank (original material boundary) entities.
ent1 through entn	A list of entities used to create a toolpath. The entities can be lines, arcs, conics, two-dimensional and three-dimensional splines, bezier curves, point sets, and machine curves. The limit of the total number of entities is 64.
NUMBER	The minor word indicating the number of entities in the entity array.
number	The number of entities in the entity array. The maximum number of entities is 100.
ent array	The name of an entity array.
CONTUR	The minor word indicating a lathe contouring toolpath.

DRILL

Statement format:

```
tpname=TLPATH/LATHE,DRILL
```

Parameter	Description
tpname	The name assigned to the defined toolpath.
LATHE	The minor word indicating lathe toolpath parameters.
DRILL	The minor word indicating a lathe drilling toolpath.

Example:

The following statement creates a lathe drilling toolpath named TP134. All the parameters needed to create the toolpath must be specified in the GPG.

```
TP134=TLPATH/LATHE,DRILL
```

THREAD

Statement format:

```
tpname=TLPATH/LATHE,THREAD,ent1[,START,(xt,yt)]
                                      (point)
```

Parameter	Description
tpname	The name assigned to the defined toolpath.
LATHE	The minor word indicating lathe toolpath parameters.
THREAD	The minor word indicating a lathe threading toolpath.
entl	The entity used to create the toolpath.
START	The minor word indicating the starting entity selection point.
xt,yt	A space location expressed in transform coordinates indicating the starting entity selection point.
point	The name of a point indicating the starting entity selection point.

Example:

The following statement creates a lathe threading toolpath using the entity LN52 with the starting entity selection location at point PT10. The new toolpath is named T413.

```
T413=TLPATH/LATHE,THREAD,LN52,START,PT10
```

Interactive Statements 22

Interactive Statements 22

Interactive statements display text, variables, prompts, and menus. Except for DISPLA, these statements require input from the person running the GPL programs.

DISPLA

The DISPLA statement displays text and real variables during an interactive GPL program.

Statement format:

```
DISPLA/( 'text'       )[,variable,...]
       (text variable )
```

Parameter	Description
'text'	The displayed text in single quotes. The number of characters can be less than or equal to 200.
text variable	The name of a text variable containing the text to be displayed.
variable	The name of a real variable containing the numerical value to be displayed. Up to 64 real variables can be listed in one statement.

Example:

Program Statement	Explanation
`DISPLA/'X,Y,Z VALUES'$` `,X,Y,Z`	When this statement is executed, ICEM DDN displays the message X,Y,Z VALUES and the actual values of the real variables X, Y, and Z.

NOTE

There must be a field in the text area. For example, if you used

```
DISPLA/'',x,y,2
```

the empty field in the single quotes would cause unpredictable results.

MENU

The MENU statement displays a predefined menu header and menu choices, and requires an entry to the displayed menu.

Statement format:

```
      (menu head var ),('menu item1*menu item2*...*'          ),choice,status
MENU/('menu head'    ) ( menu item var1* menu item var2*...*)
```

Parameter	Description
menu head var	The name of a variable containing the text of the menu header (≤ 40 characters).
'menu head'	The displayed text of the menu header. The text must be enclosed in single quotes and cannot contain more than 40 characters.
menu item	The displayed text of the menu choice. The complete list of menu choices must be enclosed in single quotes. Each menu choice, including the last one, must be followed with an asterisk. The maximum number of choices is 19 with a maximum of 40 characters per choice.
*	A delimiter that separates the menu items and indicates the end of menu item listing.
menu item var	The name of a variable containing the text of the menu choice. Each menu choice, including the last one, must be followed with an asterisk. The maximum number of choices is 19 with a maximum of 40 characters per choice.
choice	The name of a variable that receives the menu choice entered. The choice is numerically equal to the following values:

Value	Description
1	Indicates the first menu item.
2	Indicates the second menu item.
3 ⋮	Indicates the third menu item.
n	Indicates the nth menu item.

status — The name of a variable that receives the entry status. The numerical values of the status variable are:

Value	Description
-1	Indicates a valid menu choice.
0	Indicates a [.
1	Indicates a].
2	Indicates a ? (help).

Examples:

Program Statement	Explanation

`MENU/'ENTITY CONSTRUCTION',$`
`'POINT*LINE*ARC*SPLINE*',$`
`ISW,STAT`
	A menu is created with a header of ENTITY CONSTRUCTION and four choices, POINT, LINE, ARC, and SPLINE. The number of the menu choice is stored in the variable ISW, and the status is stored in the variable STAT.

`IF (STAT.GE.0) GOTO 500`
`IF (ISW.EQ.1 ) GOTO 100`
`IF (ISW.EQ.2 ) GOTO 200`
	Input from the person using the program determines the setting of ISW and STAT. The results can be used to branch from the regular sequential execution of the program.

PARAMS

The PARAMS statement provides the capability of modifying default data values. The statement displays a list of variables and their corresponding default values. The user may accept those values or enter new values.

Statement format:

```
                                        (prompt array          )
PARAMS/'head',number of elements,[-],('prompt1','prompt2',...),real array,status
```

Parameter	Description
'head'	The displayed text of the data entry header. The text must be enclosed in single quotes and cannot contain more than 40 characters.
number of elements	The number of data entry elements. If this number is preceded by a minus sign, the PARAMS statement is for data display only.
prompt array	The name of a text array containing the prompts for the corresponding elements (1 to 10 characters in length).
'prompt'	The displayed text for the element. The text must be enclosed in single quotes and is from 1 to 10 characters in length. The equals sign is provided in the display and should not be included in the text.
real array	A subscripted variable array which contains the real values input by the user. This array must have been set up in a REAL statement. If only one variable location is required, a simple (nonsubscripted) variable can be used. Preassigned values, if any, must be entered in this array.
status	The name of a variable for the entry status. The numerical values of the status variable are:

Value	Description
-2	Indicates a] without data entry.
0	Indicates a [.
-1	Indicates valid values followed by a].

Examples:

Program Statement	Explanation

`PARAMS/'POINT COORDS',3,$`
`'XVAL','YVAL','ZVAL',$`
`RVAL(1),STAT`
	An interactive prompt is created with a display header that says POINT COORDS. Three real values are requested, 1.XVAL, 2.YVAL, and 3.ZVAL. The real values are stored in an array RVAL. The status is stored in STAT.

`IF (STAT.EQ. 2) GOTO 300`
`PT1=POINT/RVAL(1),$`
`RVAL(2),RVAL(3)`
	If STAT equals -2 (a carriage return), then the program branches to the statement labeled 300. Otherwise, point PT1 is created using the x-, y-, and z-coordinates stored in RVAL(1), RVAL(2), and RVAL(3).

`PARAMS/'REVSRF ANGLES',2,$`
`'GOANG',$`
`'ENDANG',$`
`RVAL(1),STAT`
	An interactive prompt is created with a display header that says REVSRF ANGLES. Two preassigned real values are displayed. These values are stored in RVAL. The status is stored in STAT.

`SR1=REVSRF/CR1,LN1,XLARGE,$`
`GOANG,RVAL(1),$`
`ENDANG,RVAL(2)`
	A surface of revolution is created using curve CR1 and line LN1. The starting and ending angles of the surfaceare the values stored in RVAL(1) and RVAL(2).

POS

The POS statement accepts a screen position selection using the graphics cursor and reads the coordinates of this screen position.

Statement format:

```
POS/'prompt',xvalue,yvalue,zvalue,status
```

Parameter	Description
'prompt'	The displayed text of the data entry prompt. The text must be enclosed in single quotes and cannot contain more than 40 characters.
xvalue	The name of a variable that receives the value of the x-coordinate of the screen position indicated.
yvalue	The name of a variable that receives the value of the y-coordinate of the screen position indicated.
zvalue	The name of a variable that receives the value of the z-coordinate of the screen position indicated. The z-coordinate of the screen position is the current depth.
status	The name of a variable for the entry status. The possible numerical values of the status variable are:

Value	Description
-1	Indicates a C (entity select) key or the space bar.
0	Indicates a [.
1	Indicates a].
2	Indicates a ? (help).

Example:

Program Statement	Explanation
`POS/'INDICATE LOCATION',$` `  XVAL,YVAL,ZVAL,STAT`	The interactive prompt INDICATE LOCATION is created. The x-, y-, and z-coordinates of the location indicated are stored in variables XVAL, YVAL, and ZVAL. The status is stored in STAT.
`IF (STAT.GE.0) GOTO 2000` `PT1=POINT/XVAL,YVAL,ZVAL`	If STAT is greater than or equal to zero,], [or ?, then the program jumps to the statement labeled 2000. If not, a point is created using the x-, y-, and z-coordinates stored in XVAL, YVAL, and ZVAL.

QUERY

The QUERY statement requires a YES or NO answer to a displayed question.

Statement format:

```
QUERY/'prompt',response[,status]
```

Parameter	Description
'prompt'	The displayed text of the query prompt. The text must be enclosed in single quotes and cannot contain more than 40 characters.
response	The name of a variable that receives the answer. This variable is set to 1 if Y or YES is entered; it is set to 2 if N or NO is entered. It is set to 0 for any other entry.
status	The name of a variable for the entry status. The numerical values of the status variable are:

Value	Description
-1	Indicates a valid response.
0	Indicates a [.
1	Indicates a].
2	Indicates a ? (help).

Example:

Program Statement	Explanation
`QUERY/'DO YOU WANT TO DELETE$ ALL POINTS?',IANS,ISW`	An interactive prompt that says DO YOU WANT TO DELETE ALL POINTS? is displayed. The YES or NO response is stored in IANS, and the status is stored in ISW.
`IF (ISW.NE. 1) GOTO 50`	If no choice is made, the program jumps to the statement labeled 50.
`IF (IANS.NE.1) GOTO 50`	If N is entered, the program jumps to the statement labeled 50.
`DELETE/POINTS`	Otherwise, all points are deleted.
`50 STOP`	

SELECT

The SELECT statement displays a message and requires an entity selection from the entities displayed on the screen. The method of entity selection (single, chain, region in, region out, graphics cursor, pointer, name, and so forth) is controlled by the entity selection modals. These are set with the SELENT and SELMOD statements. The defaults are single entity by graphics cursor.

One Entity

Statement format:

```
SELECT/'prompt',entity,status[,coord array]
```

Parameter	Description
'prompt'	The displayed text of the select prompt. The text must be enclosed in single quotes and may not contain more than 40 characters.
entity	The name to be given to the entity selected.
status	The name of a variable for the entry status. The possible numerical values of the status variable are:
coord array	The name of a coordinate array in which the coordinates of the graphics cursor are stored.

Value	Description
-1	Indicates that an entity was selected.
0	Indicates a [.
1	Indicates a].
2	Indicates a ? (help).

Example:

Program Statement	Explanation
`SELECT/'INDICATE CURVE',$` `  CV1,STATE,CRV(1)`	The interactive prompt INDICATE CURVE is displayed. The curve selected is named CV1, and the status is stored in STATE. The coordinates of the graphics cursor are stored in array CRV.

NOTE

There is a limit of 250 entities selectable with each SELECT statement.

A Number of Entities of a Specific Type

Statement format:

```
SELECT/'prompt',mask,maxc,list,count,status
```

Parameter	Description
'prompt'	The displayed text of the prompt. The text must be enclosed in single quotes and may not contain more than 40 characters.
mask	A bit selection mask of the form: `2**entity-type1+2**entity-type2+...+2**entity-typen` where the following are frequently used entity types. (Refer to Appendix C in the ICEMDDN System Programmer's Reference Manual for detailed information.)

Type	Entity
1	Point
2	Line
3	Circle
4	Conic
5	Spline
6	Composite Curve
10	Machining Curve
11	String
12	Rectangular Array
13	Circular Array
15	Group
32	Linear Dimension
33	Circular Dimension
34	General Label
35	Diameter Dimension
36	Angular Dimension
37	General Note
38	Centerline
39	Section Lining

For example, to select points and splines, the mask is calculated as follows:

```
MASK=2**1 + 2**5
SELECT/'SELECT POINTS AND SPLINES ONLY',MASK,3,$
  LIST(1),CNT,ST
```

The mask is set for points and splines. The number of entities allowed (maxc) is 3. The entities are stored in entity array LIST. The variable CNT contains the number of entities selected by you. The variable ST contains the status of the selection operation.

Parameter	Description
maxc	The maximum number of entity selections allowed. This number may not exceed 250 entities.
list	An entity array for storing entities. As in the previous example, the array LIST contains one entity in each element of the array. That is, LIST(1) is the first entity, LIST(2) is the second entity, and so on. The variable name chosen can be used anywhere that an entity name can be used.
count	A variable that contains the number of entities selected for this operation.
status	The name of a variable for the entry status. The possible numerical values of the status variable are:

Value	Description
-1	Indicates that entities were selected.
0	Indicates a [.
1	Indicates a].
2	Indicates a ? (help).

NOTE

There is also an extended mask capability for selecting entity types whose flag bits exceed the size of the mantissa of a single precision real (OS dependent). The extended mask consists of a real array, usually of length 4, whereby the first element contains the negative of the number of elements following (typically -3). The following elements (that is, mask(2) through mask(4)) contain 16 bit slices of the total mask. Mask(2) can select entity types 1-15 (0 not used), mask(3) can select entity types 16-31, and mask(4) can select entity types 32-47. For example, to select linear (type 32) and circular (type 33) dimensions, the following mask is calculated:

```
MASK(4) = 2**0 + 2**1
```

```
      0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15

(2)   0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
(3)  16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
(4)  32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
```

TEXT

The TEXT statement displays a message and requires character input. The TEXT statement assigns a variable name to the characters entered. A text variable defined by a TEXT statement can be used, for example, as the text of an attribute, note, label, or dimension. This is done by referencing the variable name in an ATTRIB, NOTE, LABEL, or LDIMEN statement, respectively.

The text entered can be up to 200 characters. The displayed prompt appears before the text is entered. A back slash (\) entered anywhere in the text causes a down space (line feed) in subsequent displays of the entered text. Terminate text entry by entering a second carriage return, an operation complete, or an operation reject. Characters entered can be uppercase and lowercase, and do not have to be enclosed in single quotes.

Statement format:

```
TEXT/'prompt',name,status
```

Parameter	Description
'prompt'	The displayed text of the text prompt. The text for prompt must be enclosed in single quotes and can be up to 40 characters in length.
name	The name of a variable that receives the text entered. The name can be either a simple or a subscripted variable name. The variable must be dimensioned by a CHAR statement.
status	The name of a variable for the entry status. The possible numerical values of the status variable are:

Value	Description
-1	Indicates valid text input.
0	Indicates a [.
1	Indicates a].

Example:

Program Statement	Explanation
`TEXT/'ENTER NOTE',TXT1,STAT`	An interactive prompt is displayed that says ENTER NOTE. The text entered is assigned the variable name TXT1, and the status is stored in STAT.
`IF (STAT.EQ.0) GOTO 200`	If [is entered, the program jumps to the statement labeled 200.
`NT1=NOTE/5,5,ANGLE,0,TXT1`	A note is created at coordinates (5,5) using TXT1.

GPL Input and Output 23

GPL Input and Output 23

Conventions and Restrictions

The following limitations and conventions apply to the use of files in GPL input and output:

- A file must be opened before it is used, and a file must be closed after it is used. The OPEN command opens the file and the CLOSE command closes the file.
- The PAUSE, STOP, and FINI statements automatically close all open files.
- REWIND works only with sequential access. It does not work for word addressable files.
- Input/output statements are more efficient if they are grouped together.

File Formats

The following types of files are supported for input/output in GPL:

Type	Description
INPUT, OUTPUT	Formatted files, coded sequential input/output. These files are formatted as free-formatted text records. They are defined as follows: • Only four files can be in use simultaneously in a GPL program. • A record is defined as a maximum of 128 characters. • All numbers are treated as decimal numbers. A decimal point is automatically placed after the last digit of an integer. • In output files, the decimal point is automatically placed as specified in the SYSDEC statement. • Legal separators are the blank or the comma. • Leading blanks are ignored. • The single quote (') defines the beginning and ending of a string of characters. If quotes are needed in a string of characters, two consecutive quotes are used.
BININ, BINOUT	Unformatted, binary sequential input/output. These files are not converted when they are transferred. For bulk data, binary input/output has significant performance increases as compared to coded input/output. Only four files can be in use simultaneously in a GPL program.
BINDIR	Binary direct access word addressable. Direct access input/output is virtual in that it uses page buffers. Physical READ occurs when the file address is not found in a buffer. Physical WRITE occurs when all the buffers are full, or when a CLOSE statement is executed. Only two direct access files can be open simultaneously.

NOTE

BININ or BINDIR files created externally to GPL must be unformatted and should not have a header record. The following NOS statement

```
FILE, filename, RT=U, BT=C
```

should immediately precede the file's creation.

Fixed Format Input/Output

When a file is opened for input or output, text items can be read or written in fixed rather than in free format.

In fixed format, the length of a text item is no longer determined by the current length of the character variable in the I/O statement. To distinguish fixed from free format, the count must be negative on the I/O statement.

On input, the length is the maximum length specified in the CHAR declaration.

On output, the length is the current length, which may be smaller than the maximum, depending on previous use of the variable. (The current length can be set by the SETCHL statement.)

To use fixed format for input of real numbers, the following must apply:

1. The line must be read into a text variable.
2. The part of the line that contains the number must be moved into a different field.
3. This field must be converted into a real variable (CONVER).

On output this must be done in reverse order.

Example:

```
REAL/RC80, ST
CHAR/FILE7(7),C80(80), L80TMP(10)
DATA/FILE7, 'FILE7'
OPEN/FILE7,INPUT,ST
READ/FILE7,ST,-1,C80
MOVCHR/8, C80, 1, L80TMP, 1
CONVER/NUMBER, L80TMP, RC80, ST
$$ RC80 will contain the 8 digit real number from FILE7
```

BULK

The BULK statement transfers data into or out of either the current part's run-time library (RTL) or the global user technology file (UTF). This allows big application programs a fast means of recovery without using additional files.

Statement format:

```
BULK/[GLOBAL],name,[INPUT ],count,[entity array],status
     [LOCAL ]     [OUTPUT]      [real array   ]
                                [text array   ]
```

Parameter	Description
GLOBAL	The minor word for addressing the UTF.
LOCAL	The minor word for addressing the current part.
name	A text variable or character string containing the name of the record addressed. It must be a unique character string (10 characters maximum) that identifies the record.
INPUT	The minor word for handling the bulk data file as an input file.
OUTPUT	The minor word for handling the bulk data file as an output file.
count	The number of words or strings transferred. It can have the following values:

Value	Description
1024	The maximum number of real data words that can be transferred.
512	The maximum number of entity data words that can be transferred.
0	Zero is a special control number. If count equals 0 and the file is INPUT, the system checks if a record is present and, if found, sets the status to 0. If count equals 0 and the file is OUTPUT, the compiler checks if a record is present and deletes it.

entity array	The entity array name. Data is input or output depending on whether INPUT or OUTPUT is used.
real array	The name of a real data array. Data is input or output depending on whether INPUT or OUTPUT is used.

Parameter	Description
text array	The text array name. Data is input or output depending on whether INPUT or OUTPUT is used.
status	The variable name that contains the operation status. It can have the following values:

Value	Description
-1	Truncates the record because the record exceeds the value given in the count variable.
-2	Indicates the record named not found on INPUT.
-3	Indicates the wrong type of record found on INPUT. That is, the type of array or text variable did not match the type of record found.
0	Indicates the output operation was successful, or if the count equals 0, the input operation was successful.
>0	Indicates the number of items that were read in.

NOTE

Only experienced programmers should use this statement, as the new data base entities may overwrite the old data base entities with the same name.

Use GLOBAL with care. The overwriting or deleting a record that was not the last record created is very slow.

CLOSE

The CLOSE statement closes a file that was previously opened with an OPEN command.

The CLOSE statement writes an end-of-file (EOF) at the current position for files that were previously opened as BINOUT or OUTPUT.

Statement format:

```
CLOSE/( 'file name' ),status
      (text variable)
```

Parameter	Description
'file name'	The name of a local file enclosed in single quotes (7 characters maximum).
text variable	The name of a text variable that contains a string of characters that names the local file (7 characters maximum).
status	A variable that contains the condition of the file used for input/output. The condition is expressed as one of the following values:

Value	Description
0	The close operation was successful.
-1	The file was not opened.

OPEN

The OPEN statement opens a file for input or output. The file is positioned at the beginning-of-information (BOI).

Statement format:

```
OPEN/{ 'file name' }, { BININ  }
     { text variable}  { BINOUT }
                       { BINDIR } ,status
                       { INPUT  }
                       { OUTPUT }
```

Parameter	Description
'file name'	The name of a local file enclosed in single quotes (7 characters maximum).
text variable	The name of a text variable that contains a string of characters that names the local file (7 characters maximum).
BININ	The file is used for input. The file is formatted as a binary sequential file.
BINOUT	The file is used for output. The file is formatted as a binary sequential file.
BINDIR	The file is used for input or output. The file is formatted as a binary direct access word addressable file.
INPUT	The file or text variable is used for input. The file is formatted as a coded sequential file.
OUTPUT	The file or text variable is used for output. The file is formatted as a coded sequential file.
status	A variable that contains the condition of the file used for input/output. The condition is expressed as one of the following values:

Value	Description
0	The OPEN operation was successful.
-1	The file is already opened.
-2	There are too many open files.
-3	File is empty.
>0	The highest word address (BINDIR file only).

READ

The READ statement reads character strings or numbers according to their variable types. The number of variables is limited to a maximum of 60 variables. The reading operation continues until the count specified is reached regardless of line terminators. If a new line is read and no more input is required (that is, no more variables are requested), the rest of the line is ignored.

Sequential Access

This form is for both INPUT and BININ type files.

Statement format:

```
READ/('file name' ),status,count,(  array    )
     (text variable)              (variable,...)
```

Parameter	Description
'file name'	The name of a local file enclosed in single quotes (7 characters maximum).
text variable	The name of a text variable that contains a string of characters that names the local file (7 characters maximum).
status	A variable that indicates the condition of the file being read after executing the READ statement. The condition is expressed as one of the following values:

Value	Description
0	The read operation was successful.
1	EOF marker is encountered.
-1	The file is not open.
-2	The file is opened for output.
-3	A conversion error has occurred.
-4	Illegal count.

count	Either the number of items (CHAR or REAL) in the array or the number of variables listed. A negative count indicates fixed format.
array	The name of an array to be read. It is recommended that the count for the array does not exceed 4000 for performance reasons.
variable	A listing of the variables to be read (REAL or CHAR).

Direct Access

This form is for use only with BINDIR type files. The READ statement has the additional parameter, address.

Statement format:

```
READ/{'file name'  },status,count,address,{  array     }
     {text variable}                      {variable,...}
```

Parameter	Description
'file name'	The name of a local file enclosed in single quotes (7 characters maximum).
text variable	The name of a text variable that contains a string of characters that names the local file (7 characters maximum).
status	A variable that contains the condition of the file used for input. The condition is expressed as one of the following values:

Value	Description
>0	The read operation was successful (indicates the next word address).
-1	The file is not open.
-2	The file is opened for output.
-3	A conversion error has occurred.
-4	Illegal count.

count	Either the number of items (CHAR or REAL) in the array or the number of variables listed. A negative count indicates fixed format.
address	The address of the first word of the file read.
array	The name of an array to be read. It is recommended that the count for the array does not exceed 4000 for performance reasons.
variable	A listing of the variables to be read (REAL or CHAR).

REWIND

The REWIND statement rewinds a sequential file.

Statement format:

```
REWIND/( 'file name' ),status
       (text variable)
```

Parameter	Description
'file name'	The name of a local file enclosed in single quotes (7 characters maximum).
text variable	The name of a text variable that contains a string of characters that names the local file (7 characters maximum).
status	A variable that contains the condition of the file used for input/output. The condition is expressed as one of the following values:

Value	Description
0	The rewind operation was successful.
-1	The file does not exist.

USTRUC

The USTRUC statement creates a file which can be used as input to the UNISTRUC program.

USTART Parameter

The USTRUC/USTART statement generates a DEFBLK card in the UNISTRUC program.

Statement format:

```
USTRUC/USTART,n
```

Parameter	Description
USTART	The minor word for generating a DEFBLK card in a UNISTRUC program.
n	The block number of the DEFBLK card.

UWRITE Parameter

The USTRUC/UWRITE statement defines the number of points per curve and the number of entities to be transferred to the UNISTRUC program. If lines are specified, only two points are generated for each line.

Statement format:

```
USTRUC/UWRITE,np,ne,entity array
```

Parameter	Description
UWRITE	The minor word for transferring entities to the UNISTRUC program.
np	The number of points to be generated per curve.
ne	The number of entities in the entity array.
entity array	The name of the entity array to be transferred.

UTERM Parameter

The USTRUC/UTERM statement flushes the data buffers and writes an end-of-record (EOR) to the UNISTRUC program. The resulting file is not rewound.

Statement format:

```
USTRUC/UTERM
```

Parameter	Description
UTERM	The minor word for flushing the data buffers and writing an EOR to a UNISTRUC program.

WRITE

The WRITE statement writes the specified variables to the file indicated. For coded files, the SYSDEC statement determines the number of decimal places for REAL variables. A maximum of 128 characters per line are written to the output file. A new line is written per each 128 characters.

Sequential Access

This form is for both OUTPUT and BINOUT type files.

Statement format:

```
WRITE/('file name' ),status,count,(  array     )
      (text variable)              (variable,...)
```

Parameter	Description
'file name'	The name of a local file enclosed in single quotes (7 characters maximum).
text variable	The name of a text variable that contains a string of characters that names the local file (7 characters maximum).
status	A variable that indicates the condition of the file being written after executing the WRITE statement. The condition is expressed as one of the following values:

Value	Description
0	The write operation was successful.
-1	The file is not open.
-2	The file is opened for input.
-3	A conversion error has occurred.
-4	Illegal count.
>0	In binary mode, this is the next write address.

count	Either the number of items (CHAR or REAL) in the array or the number of variables listed. A negative count indicates fixed format.
array	The name of an array to be written.
variable	A listing of the variables to be written. Character strings can be written out by enclosing them in single quotes in the variable listing.

Direct Access

This form is for use only with BINDIR type files. The WRITE statement has the additional parameter, address.

Statement format:

```
WRITE/[ 'file name' ],status,count,address,[ array     ]
      [text variable]                      [variable,...]
```

Parameter	Description
'file name'	The name of a local file enclosed in single quotes (7 characters maximum).
text variable	The name of a text variable that contains a string of characters that names the local file (7 characters maximum).
status	A variable that contains the condition of the file used for output. The condition is expressed as one of the following values:

Value	Description
>0	The write operation was successful (this is the next write address).
-1	The file is not open.
-2	The file is opened for input.
-3	A conversion error has occurred.
-4	Illegal count.

count	Either the number of items (CHAR or REAL) in the array or the number of variables listed. A negative count indicates fixed format.
address	The address of the first word of the file at which to write.
array	The name of an array to be written.
variable	A listing of the variables to be written.

Using the EXEC Statement

The EXEC statement executes a subroutine written in FORTRAN. Only variables of type REAL can be passed as parameters. The subroutine must be in a library (default is named UCLIB) and must be of the type CAP (capsule).

Statement format:

```
EXEC/('FTN name'|ftn variable)[,('lib name'|lib variable)], number,array
```

Parameter	Description
'FTN name'	The name of the FORTRAN subroutine enclosed in single quotes.
ftn variable	The name of a variable that contains a string of characters that names the FORTRAN subroutine.
'lib name'	The name of the library enclosed in single quotes (default is named UCLIB).
lib variable	The name of a text variable containing a string of characters that names the library.
number	The number of parameters in the parameter array.
array	The name of a real array that contains the parameters to be passed to the FORTRAN subroutine.

The following limitations must be observed:

- The capsule library must be created by the NOS utility GTR with the D parameter, or by using LIBEDIT if you replace a capsule in an existing library or use the BUILD directive. The following procedure is the kind used to generate a capsule library from a subroutine. The subroutine is called SUB123 on local file COMPILE.

```
FTN5,I,L=LIST.
CAPSULE,SUB123.
LOAD,LGO.
NOGO.
GTR,ABS,UCLIB,D.CAP/*
```

- The subroutine cannot use any FORTRAN input/output statements.
- The subroutine called must start with the following two statements:

```
SUBROUTINE name (N,A)
DIMENSION A(N)
```

  The subroutine can then call other subroutines within the same capsule. The total size of the capsule including array A is limited to 40,000 octal (16,384 decimal) words. This is a maximum size, but for reasons of efficiency, a capsule size of 20,000 octal (8,192 decimal) with an array size of 10,000 octal (4,096 decimal) should not be exceeded.

Compiling a GPL Program 24

Compiling a GPL Program 24

GPL Control Statement

The GPL control statement, which is entered from NOS, calls the compiler. Opening and closing parentheses enclose the parameters. The parentheses are special symbols, which are used to indicate the start and end of the parameters. You may use commas instead of parentheses. These parentheses must be included in the control statement. The parameters in the control statement are order independent. Parameters cannot be repeated in a control statement. If the statement is entered in interactive mode, a carriage return is sufficient to end the statement.

Statement format:

```
GPL(I=input,L=list,B=objfil,E=errfil,LO,EL,ET,PS,IL)
```

Parameter	Description
I=input	The file name (1 to 7 characters) that contains the source input text.

Parameter	Description
I omitted	Default: The file name is INPUT.
I alone	Special case: The file name is COMPILE.

Parameter	Description
L=list	The file name (1 to 7 characters) that contains listable compiler output.

Parameter	Description
L omitted	Default: The file name is OUTPUT.
L alone	Special case: The file name is LIST.
L = 0	Special case: Suppress listable compiler output. Only error statistics for the programs compiled are listed on the file named OUTPUT.

Parameter	Description
B=objfil	The file name (1 to 7 characters) that contains the object code output.

Parameter	Description
B omitted	Default: The file name is GPLOBJ.
B alone	Special case: The file name is LGO.
B=0	Special case: Suppress object code output.

Parameter	Description
E=errfil	The file name (1 to 7 characters) that contains the faulty statements and error messages.

Parameter	Description
E omitted	Default: suppress error message file.
E alone	Special case: the file name is ERRS.
E=0	Special case: same as E omitted.

LO — List Options (simply concatenated, any sequence). S stands for source listing, R for reference map. (For maintenance, O and Q display object code proper and the complete object code file, respectively, and Z, P, A, M, and Dn (with n=1 to 4) permit intermediate debug output for various components of the compiler.

Parameter	Description
LO omitted	Default: S, source listing only.
LO alone	Special case: SR, source listing and reference map only.

EL — Level of error messages to be put on output file; T means all, F means FATAL ones only.

Parameter	Description
EL omitted	Default: T, all error messages.
EL only	Special case: F, FATAL error messages only.

ET — Error Terminate Flag; if present, the compiler terminates on a fatal error.

PS — The page size in number of lines of the output listing.

Parameter	Description
PS omitted	Default: page size is 60 lines.
PS only	Special case: Page size is 40 lines.

IL — The source input line length to be scanned.

Parameter	Description
IL omitted	Default: line length is 80 columns.
IL only	Special case: line length is 72 columns.

When the compiler terminates, it passes information at the control level as to how successfully it ran. This information is system dependent.

The CCL error flag (EF) is set in the following manner:

Flag	Description
0	No errors.
1	Trivial errors (warnings).
2	Fatal errors.
3	Catastrophic errors (compilation terminates, for example, because of table overflow).

CCL register R1 is set to the number of programs compiled successfully.

If there are no catastrophic errors, and no fatal errors with EL=F specified as an execution parameter, the system displays:

```
GPL COMPLETED:  xx TRIVIAL, yy FATAL ERRORS; zz PROGRAMS COMPILED
```

If there are catastrophic errors, or fatal errors with EL=F set, the system displays:

```
GPL ABORTED:  xx TRIVIAL, yy FATAL ERRORS; zz PROGRAMS COMPILED
```

On all systems, this message is also put at the end of either list output.

Library Format Under NOS

The source to be compiled must be available in a local, sequential file. Use the NOS utility MODIFY both to:

- Maintain the source of larger program packages.
- Select one or more of the decks used for compilation under control of a procedure.

The object code delivered by the compiler must be put into a standard format library before ICEM DDN can execute it. Achieve this by using the NOS utilities GTR or LIBEDIT.

Example	Explanation
`GET,FN.`	Get the GPL source program.
`ATTACH,GPL.`	Attach the GPL compiler.
`GPL,I=FN,L=GPLLIST.`	Compile program FN with listing put on GPLLIST.
`ATTACH,GPLLIB/M=W.`	Attach GPL library.
`LIBEDIT,B=GPLOBJ,P=GPLLIB,Z./*BUILD GPLD`	Place compiled programs on GPL library.
`REWIND,*.`	
`COPY,NEW,GPLLIB.`	

NOTE

ICEM DDN Version 1.6 contains a new version of the GPL compiler. Unlike the old compiler, which kept source and object code together in a library in a special format, the new compiler doesn't do any library handling. Newly compiled programs do not work on versions of ICEM DDN created prior to version 1.6. Therefore, do not insert the object code from the new compiler into libraries created prior to version 1.6.

Menu 5.13 GPL 25

5.13 GPL

With this choice, you can execute precompiled GPL programs. These programs are custom applications ranging from family of parts generators to interactive tutorials. These programs must be contained in a local GPL library.

The menu for this section is:

```
GPL
1.RECOVER LAST FILE
2.CONTINUE GPL PROGRAM
3.RUN GPL PROGRAM
4.LIST GPL NAMES
5.CHANGE LIBRARY NAME
```

The following sections describe the choices in this menu.

5.13.1 Recover Last File

With this choice, you can continue program execution following a FILE command in a GPL program (refer to chapter 7 for a description of the FILE command). Execution continues at the next statement following the last FILE command.

5.13.2 Continue GPL Program

With this choice, you can continue program execution following a PAUSE statement in the GPL program. Choosing this menu item if there is no PAUSE statement in the current GPL program returns you to 5 SPECIAL FUNCTIONS.

5.13.3 Run GPL Program

With this choice, you can execute the GPL program.

`ENTER SIX CHARACTER NAME`	Enter the name of the GPL program to run. This name can be 1 to 6 characters long.

5.13.4 List GPL Names

With this choice, you can display the names of GPL programs and subprograms which are contained in a local GPL library.

5.13.5 Change Library Name

With this choice, you can change the name of the local GPL library to use in your program. The default name of this library is GPLLIB.

`CURRENT GPL LIBRARY IS name` `ENTER NEW GPL LIBRARY NAME`	Enter the name of the local GPL library that you want to access. The name of the library can be 1 to 7 characters long.

Appendixes

Glossary A

A

Alphanumeric

The letters of the alphabet (A through Z) and the digits (0 through 9).

Angular Dimension

The entity type showing the angular distance between two lines.

ANSI

American National Standards Institute. English (U.S. customary) dimensions are specified in feet and/or inches. Metric dimensions are specified in millimeters.

Approximation

A process of computing a curve that interpolates the given point set.

APT

The Automated Programmed Tooling system, a language for defining part geometry and numerically controlled toolpaths.

Arithmetic Operator

Any of these operations: addition, subtraction, exponentiation, multiplication, and division.

Aspect Ratio

The ratio between the character width and the character height in dimensions.

Assignment Statement

A statement that assigns a name to an entity or a mathematical expression.

Attention Indicator

A small oval that the system displays on an entity to indicate that the entity has been selected.

Attribute Text

User-assigned alphanumeric text associated with an entity. Attribute text is used to identify and describe an entity.

Automatic Dimensioning

The system capability that calculates the distance or angle of a dimension and automatically writes the text for that distance or angle in the dimension.

Auxiliary View

A view that can be user-defined. The standard views (top, bottom, right, left, front, back, isometric, and auxiliary) are defined by the system.

B

Blanking

A process applied to entities in the current part where the entities remain in the part, but are not displayed on the screen. Contrast with Deleting.

C

CCLW

Counterclockwise.

CCW

Counterclockwise.

Centerline

1. A font type.
2. A drafting entity shown only in the view of definition, used to refer to center locations.

Chain

The process of selecting a series of contiguous curves.

Chamfer

A straight line that joins two lines, arcs, or curves. It is generally used to change a sharp corner into a blunt one. Contrast with Fillet.

Character String

One or more contiguous text characters enclosed in single quotes.

Circular Array

Copies of an object created at equally spaced intervals along the perimeter of an imaginary circle.

CLW

Clockwise.

Conditional Statement

A statement that tests for a logical decision and then performs an action based on the results of that decision.

Conic Section

One of a group of plane curves, including the circle, ellipse, hyperbola, and parabola.

Constant

A numerical value that is used without variation throughout the program.

Contiguous Curves

Adjacent curves that share a common endpoint or that have endpoints within a short distance of each other.

Coordinate Axes

The lines that form a right-handed, three-dimensional frame of reference. The model axes (x, y, and z) and the transform axes (xt, yt, and zt) are referenced from a shared origin point in which x = 0, y = 0, and z = 0.

Coordinate Space

A space created by the position (rotation and translation) of the coordinate axes. There are two types of coordinate space: model and transform.

Coordinates

The position of a point in relation to the x-, y-, and z- or xt-, yt-, and zt-axes.

Curve

A line, arc, conic, or two-dimensional spline. In some cases, it also includes points, three-dimensional splines, strings, composite curves, machining curves, and Bezier curves.

CW

Clockwise.

D

Data Base

An integrated set of files, tables, arrays, and other data structures.

Data Capture

The system capability to save as a variable either a default parameter value or a previously entered parameter value displayed on the screen.

Datum Feature Symbol

One or two characters in a rectangular frame that identifies a part feature.

Datum Target

A specific point, line, or area identified on a drawing by a datum target symbol. It establishes datum planes commonly used for manufacturing or inspection repeatability.

Default Value

A value used by the system if] is entered in response to a data entry prompt. It is also the initial value assigned to a modal until it is changed.

Definition Form

Each entity in ICEM DDN is defined with a specific entity type and form. The types and forms are described in the ICEM Design/Drafting System Programmer's Reference Manual.

Deleting

The removal of entities from the data base. Contrast with Blanking.

Depth

The current value of the workplane along the z- or zt-axis of the workspace.

Dimension

An entity used to show a geometric characteristic of a part, such as the diameter, length, angle, or center distance.

Dimension Lines

Arrows from dimension text to extension lines that indicate the dimension being measured.

DIN

Deutsche Industrie Norme (German industry standards).

Direct Access File

Permanent file that can be attached directly to a job without any local copy being generated. All changes to this file are made to the file itself.

Dormant Entity

An entity created by the system for the purpose of defining another entity. Dormant entities cannot be displayed or manipulated. All dormant entities have sequence numbers.

Downspace Ratio

The distance from the bottom of one line of dimension text to the bottom of the next line.

Dual Dimensioning

The system capability to produce dimensions containing both SI and U.S. customary units.

E

Entity

The representation of a geometric construction in the ICEM DDN data base. Examples are points, lines, arcs, and spheres.

Entity Name

A name given to a geometric definition.

Entity Type Numbers

System-assigned code numbers used to identify specific entity types such as line, group, and copious data.

EOF

End of file.

EOR

End of record.

Extension Lines

Lines extending from entities between which dimensions are given. The distance measured is indicated.

F

Feature

A specific component of a part such as a notch, the sides of a hole, or the flat side.

Feature Frame

The entity type used for datum feature, geometric tolerance, or composite geometric tolerance symbols.

Fillet

An arc that joins two lines, arcs, or curves. It begins and ends at a point of tangency on each of the two lines.

Font

The method used to represent a line or curve in a display. Examples of fonts are solid, dashed, phantom, and centerline.

Function Control Key/Square

A keyboard/tablet character that has a special function in ICEM DDN such as menu and entity selection.

Functions

Commonly used arithmetic/trignometric operations, such as absolute value, sine, and cosine.

G

Graphics Cursor

The cursor symbol used to locate or define entities by screen position. The graphics cursor can be crosshairs or some other cursor symbol.

Group

A set of entities defined and treated as a unit such as a balloon or a surface texture entity.

H

Hidden Lines

Line segments that are obscured from view in display of a three-dimensional object.

I

ICEM DDN

Control Data's integrated computer-aided engineering and manufacturing software application for design, drafting, and numerical control.

ICEMDDN

The name of the direct access file that contains ICEM DDN.

Interpolation

A process of computing a curve that satisfies all the conditions imposed; for example, a curve that passes through all given points.

L

Label

Text and a leader with an arrow from the text to an entity on the drawing.

Leader

A line leading from a dimension value or an explanatory note to a selected entity.

Level Management

A method of assigning entities to different levels for management purposes.

Level Number

A number assigned to a collection of entities.

Local File

Any file that is currently associated with a job.

Logical Operators

The conditional operations of equality, such as less than, less than or equal to, equal to, greater than or equal to, or, greater than.

M

Major Word

A statement element that names the operation to be performed.

Major Word Statement

A statement that describes a GPL operation.

Menu

A list of options used to perform operations in ICEM DDN.

Message

Text written by the system to display information, errors, or warnings. See also Prompt.

Minor Word

A modifier element that follows a major word.

Mirror

An operation that enables you to produce a copy of a set of entities in the part by rotating them 180 degrees around a selected line.

Modal

A user-assigned status or value that controls the operation of ICEM DDN. Examples are modals for setting system decimal places and for activating the grid.

Model Coordinate Space

The space created by the position of the x-, y-, and z-axes. This space is displayed as the front view, view 1. See Transform Coordinate Space.

N

N/C

Numerical control.

Normal

Perpendicular.

NOS

Network Operating System.

Note

An entity that contains graphics text for drawings.

O

Object Code

The executable code of the GPL program.

Operating System

The set of system programs that controls the execution of user programs and commands.

Operation Complete (])

A keyboard/tablet operation that completes the current operation.

Operation Reject ([)

A keyboard/tablet operation that rejects the previous operation. The system returns to the preceding prompt.

P

Parameter

A variable whose values determine the operation or characteristics of a system.

Part Name

A series of alphanumeric characters that identify a part or drawing used in conjunction with a sheet number.

PATTERN

Default primary pattern library.

Pattern

A named set of entities that can be selected and saved in a pattern library. They can be retrieved and copied into a part at a specified point.

Permanent File

A file saved by the system between terminal sessions.

Point

A geometric construction located by x-, y-, and z-coordinates.

Point-Set Curve

A piecewise linear or polynomial approximation to an actual curve. It is a series of points stored as transform coordinates.

Pointer Number

The number assigned to an entity that describes the location in the data base of that entity's parameters.

Polar Line

A line extending from an existing point in a specified direction for a specified distance.

Polar Point

A point at a specified distance and angle from a selected base point.

Program File

The text version of the program. See also Source Code.

Projection

An operation for copying entities at a specified distance from the original entities. The projected entities are connected to the original entities with lines.

Prompt

A screen display requiring user action. See also Message.

PRU

Physical record unit.

Punctuation

Common punctuation marks, such as the comma and the period, and other special symbols used to delineate the format of a statement.

R

Rectangular Array

Copies of an object at equally spaced intervals, arranged in rows and columns.

Region Select

A process of selecting all the entities within or outside of a specified region.

Rescale

To change the scale factor and redraw a part using the new scale so that all entities in the part are displayed.

Rotate

To revolve a construction about an axis.

RTL

See Run-Time Library.

Run-Time Library (RTL)

The storage area for all variables defined by data capture, variable calculation, and program execution.

S

Scalar

A quantity that has magnitude only. Contrast with Vector.

Scale Factor

Ratio of the current display with respect to the data base.

Section Lining

An entity type defining patterned representations for material types including iron, steel, brass/copper, rubber/plastic, refractory, glass/slate, lead, or aluminum/magnesium. Section lining was formerly called cross hatching.

Sequence Number

A unique sequential number associated with each entity.

Sheet Number

A subdivision of a part name. Sheets can be numbered from 0 to 9,999 with a part name.

SI

Systeme International d'Unites (international system of units). Dimensions are calculated in metric units. Lengths have units of millimeters. Areas have units of millimeters squared unless otherwise specified.

Single Select

A process of selecting individual entities one at a time.

Source Code

The text version of the program. See also Program File.

Spacing

The distance between the centers of consecutive characters.

Spline

A curve that passes through and remains smooth between data points.

Standard Views

The system-generated coordinate systems from which the user can view a display of the part surfaces from these view points: front, back, top, bottom, right, left, isometric, and auxiliary.

Statement Label

A unique integer identifier for a statement.

String

A single entity that is displayed as a connected set of lines and arcs. A string, however, is not lines and arcs and cannot be used in place of lines and arcs.

Subscript

An array identifier that uniquely identifies an element's position within an array.

Surface Normal Point

A point on a surface located at the intersection of the surface and a surface normal vector.

Surface of Revolution

Surface produced by the rotation of a curve around an axis through a specified angular displacement.

Surface Parameters

A pair of numbers (u,v) that specify positions on a surface.

Surface Pierce Point

The intersection between a surface and a line parallel to a vector and passing through a selected base point.

System

The ICEM DDN software system.

T

Tabulated Cylinder

Surface generated by the translation of a curve along a specified direction with upper and lower limits on the distance of translation.

TAPE3

A local file on which parts are filed and from which parts are retrieved.

Template

A named set of geometric entities that have two forms, masters and instances. A master is the shape to which the instances conform.

Temporary File

A nonpermanent file associated with a job only during job processing.

Text

Alphanumeric characters and symbols used in notes, labels, and dimensions.

Text Variable

A string of one or more characters assigned a variable name.

Tolerance

The range of variation allowed in maintaining a specified dimension.

Tolerance Ratio

The ratio between the character size for tolerance characters and the character size for main characters in dimensions.

Transform Coordinate Space

The space created by the position of the xt-, yt-, and zt-axes. This space is displayed in any view other than view 1. See Model Coordinate Space.

Transformation of Coordinates

The mapping of the xt-, yt-, and zt-coordinates into the x-, y-, and z-coordinates.

Translate

To relocate entities on the screen display by assigning them different origin coordinates.

Trimming

To delete or extend a segment of an entity to a prescribed boundary.

U

U Path

A curve on a surface in one of two parametric directions. See V Path.

Unblank

A procedure for making visible entities that have been blanked by the user.

Units of Measurement

Acceptable units of measurement for GPL are either millimeters, inches, or inches/feet.

User Technology File (UTF)

The storage area for various products of 5 SPECIAL FUNCTIONS including defined character sets and level tables.

V

V Path

A curve on a surface in one of two parametric directions. See U Path.

Variable Calculation

An operation in 5.2 GRAPL that calculates the value of an arithmetic expression and assigns that value to a user-specified variable.

Variable Name

A name given to a mathematical expression.

Vector

A quantity that has magnitude and direction. It is commonly represented by a line segment with directionality. The line's length represents the magnitude and its orientation in space represents the direction. Contrast with Scalar.

View
A display of coordinate space.

W

Work Plane
A specific plane in workspace on which two-dimensional entities are constructed.

Work View
The view in which screen input is accepted.

Workspace
The space in which entity construction and specification occurs.

Z

Zoom
To enlarge or decrease the size of the display proportionately.

Minor, Modal, and Positional Relationship Words B

Word	Description
ABOVE	Places the dimension text above the dimension line (SRFTEX statement minor word). Places the feature frame above existing note, label, or feature frame (DATFEA and GEOTOL statement minor word).
ACYCLC	Makes the start and end conditions anticyclic; that is, in opposite directions (SPCURV statement minor word).
ADD	Adds a rectangle or parentheses (MODDFT statement minor word).
AFTER	Copies the attributes of an entity after the existing attributes of another entity (ATTRIB statement minor word).
ALL	Blanks or unblanks all entities (BLANKE and UNBLANK statement minor word). Indicates section lining is displayed in all views (SECVIS statement model word).
ALUM	Specifies aluminum material in section-lining (MATERL statement minor word).
ANGLE	Specifies the angle in statements where an angle is required.
ARCS	Defines arcs (DEFINE statement minor word).
AREA	Indicates a datum target symbol with an area (DATUM statement minor word).
AROUND	Indicates that the geometric characteristic is PROFILE and that the symbol indicating this is attached to the leader line (GEOTOL statement minor word).
ARRDO	Defines only specific holes, points, or sets of entities (ARRAY statement minor word).
ARRW	Produces an arrowhead (DATFEA, SRFTEX, and GEOTOL minor word).
ASPCT	Sets the aspect ratio for characters; that is, the ratio between character width and character height (CDISPL statement minor word).
ATANGL	Specifies the angle between points or lines. Measures the angle from the positive x-axis (POINT statement minor word). Measures the angle from the given line (LINE statement minor word).
ATNAME	Adds an attribute name to an entity (ATTRIBUTE statement minor word). Retrieves an attribute name from an entity (OBTAIN statement minor word).
AUTO	Specifies an automatic origin for dimensioning (DORIG statement minor word).

Word	Description
AUX	Creates an auxiliary view (VIEW statement minor word).
BASIC	Adds or deletes rectangle around dimension text (MODDFT statement minor word). Adds a surface texture symbol (SRFTEX statement minor word).
BEARDS	Defines a point at a bearing and distance (POINT statement minor word).
BEFORE	Copies the attributes of one entity before the existing attributes of another entity (ATTRIB statement minor word).
BELOW	Places the feature frame below existing note, label, or feature frame (DATFEA and GEOTOL statement minor word).
BINDIR	Declares a binary direct access word addressable file (OPEN statement minor word.
BININ	Declares a binary sequential input file (OPEN statement minor word).
BINOUT	Declares a binary sequential output file (OPEN statement minor word).
BLNK	Indicates the blank (original material boundary) entities (SETGPG and TLPATH statement minor word).
BORDER	Draws a border around the reproduced, magnified section of drawing (MAGNFY statement minor word).
BOTH	Adds subattribute names and numbers to an entity (ATTRIB statement minor word). Retrieves the subattribute name and number (OBTAIN statement minor word). Produces dual dimensions (DUAL statement modal word). Places arrows at both ends of a dimension (SECARR statement minor word).
BRACKT	Produces dual dimensions with alternate dimensions in brackets with no line separating upper and lower text (DUAL statement modal word).
BRASS	Specifies BRASS material in section-lining (MATERL statement minor word).
CCW	Indicates the counterclockwise arc direction (STRING statement minor word). Aligns a note in the counterclockwise direction parallel to a given arc (NOTE statement minor word).
CENLIN	Defines all subsequent curves in the centerline mode until a new font is entered (FONT statement minor word).
CENTER	Defines a circle center (POINT and CIRCLE statement minor word). Defines a point center for plane (PLANE, ELLIPS, HYPERB, and PARABO statement minor word). Locates a template instance from a given origin (TEMPLT statement minor word). Generates a centered character string (TXTJUS statement minor word). Defines the center of a drawing (ZOOM statement minor word).

Word	Description
CHAIN	Selects entities in a chain (SELENT statement minor word).
CHARST	Retrieves the character string of a dimensioning entity (OBTAIN statement minor word).
CIRC	Defines a circle (MAGNFY and DATFEA statement minor word). Defines a circular array (ARRAY statement minor word).
CIRCUM	Defines the circumference of a sphere (SPHERE statement minor word).
COMPOS	Indicates that all references and modifiers before this word are entered in the top frame, and all references and modifiers after this word are entered in the bottom frame (GEOTOL statement minor word).
CONIC	Indicates a conic taper dimension (TAPER statement minor word).
CONTUR	Indicates a lathe contouring toolpath (SETGPG and TLPATH statement minor word).
COORD	Retrieves coordinates (OBTAIN statement minor word). Positions a 3-D pattern using a local coordinate system (RTRIEV statement minor word). Modifies coordinates (MODIFY statement minor word). Zooms to a region defined by two corner points (ZOOM statement minor word).
COPPER	Specifies copper material in section-lining (MATERL statement minor word).
COPY	Copies attributes from one entity to another entity (ATTRIB statement minor word).
CORNER	Indicates the X, Y of a corner of the plane (PLANE statment minor word).
CREATE	Creates a master (TMPLT statement minor word).
CRT	Designates the CRT as the input device (CURSOR statement minor word).
CURVE	Specifies a point on a curve (POINT statement minor word).
CURVW	Retrieves the current work view pointer (OBTAIN statement minor word). Changes the current work view (CHANGE statement minor word).
CW	Indicates the clockwise arc direction (STRING statement minor word). Aligns a note in the clockwise direction parallel to a given arc (NOTE statement minor word).
CYCLIC	Makes the start and end conditions cyclic (SPCURV statement minor word); that is, the same direction.

Word	Description
DASHED	Defines all subsequent curves in the dashed mode until a new font is entered (FONT statement minor word).
DATMOD	Indicates a datum modifier (GEOTOL statement minor word).
DATREF	Indicates a datum reference (GEOTOL statement minor word).
DECIM	Indicates that the angle text of a radial dimension is decimal (TXTANG and ANUNIT statement minor word).
DEFVW	Retrieves the definition view of an entity (OBTAIN statement minor word).
DEGREE	Indicates that the angle text of a radial dimension is in degrees only (TXTANG statement minor word) (ANUNIT statement minor word in ANSI 82).
DEGTOL	Indicates the degree of tolerance (DEVSRF statement minor word).
DEL	Deletes a rectangle or parenthesis (MODDFT statement minor word).
DELANG	Specifies the delta angle for all statements which require incremental angular information.
DELATR	Deletes an attribute (ATTRIB statement minor word).
DELTA	Defines a delta distance from a given origin (POINT, TXTORG, CDISPL, and DORIG statement minor word).
DELTAX	Specifies the delta distance from an x-coordinate for all statements which require incremental x-coordinate information.
DELTAY	Specifies the delta distance from a y-coordinate for all statements which require incremental y-coordinate information.
DELTAZ	Specifies the delta distance from a z-coordinate for all statements which require incremental Z information.
DETAIL	Indicates the detail number of the balloon (BALOON statement minor word).
DIFFER	Indicates the difference of two vectors (VECTOR statement minor word).
DIRECT	Indicates a straight line from the datum to the feature frame (DATFEA and GEOTOL minor word).
DISP2	Displays both witness lines of a linear dimension (WLINE statement minor word).
DISTNC	Specifies the distance or length of a line (parallel PLANE statement minor word and LINE statement minor word).

Word	Description
DONT	Suppresses specific holes, points, or sets of entities (ARRAY statement minor word).
DOWNSP	Specifies the downspace ratio for characters; that is, the distance between lines of text (CDISPL statement minor word).
DRILL	Indicates a lathe drilling toolpath (SETGPG and TLPATH statement minor word).
EAST	Indicates an east bearing (bearing/distance POINT statement minor word). Locates a template instance from a given origin (TEMPLT statement minor word).
EDGE	Defines a machine curve as a surface edge curve (MACCRV statement minor word).
END	Indicates that the leader starts at the end of the label (CDIMEN and LABEL statement minor word). Joins two curve endpoints (LINE statement minor word). Defines a point at the end of a curve (POINT statement minor word). Places arrowhead at end of dimension (SECARR statement minor word).
ENDANG	Specifies the ending angle for any statement in which a terminating angle value is required.
ENTER	Indicates that the user chooses the method of angle control (ANGCTL statement minor word), or the method of text origin (TXTORG statement minor word).
ENTNAM	Retrieves the name of an entity (OBTAIN statement minor word).
ENTPTR	Retrieves the pointer of a named entity (OBTAIN statement minor word).
ENTTY	Indicates a datum target symbol as an existing entity (DATUM statement minor word). Attaches a surface texture symbol to an enitity (SRFTEX statement minor word).
ENTTYP	Retrieves the type and subtype of an entity (OBTAIN statement minor word).
ERRST	Retrieves the error and line number if an error was encountered and STATUS is ON (OBTAIN statement minor word).
FAST	Sets the character set to fast (CSET statement minor word) (CRES statement minor word in ANSI 82).
FILTER	Indicates that the filter method is used to calculate the spline coefficients (SPCURV statement minor word).
FINE	Sets the character set to fine (CSET minor word) (CRES statement minor word in ANSI 82).

Word	Description
FIRST	Indicate that the leader starts from the first text line (LEADER statement modal word). Indicates the first side selected as the dimension origin (DIMORG statement modal word).
FLAT	Indicates a flat taper dimension (TAPER statement minor word).
GCHAR	Indicates geometric characteristic (GEOTOL statement minor word).
GLASS	Specifies glass material in section lining (MATERL statement minor word in ANSI 1982).
GLOBAL	Indicates that the second derivative continuous method is used to calculate the spline coefficients (SPCURV statement minor word). Addresses the UTF (BULK statement minor word).
GOANG	Specifies the start angle for any statement in which the initial angle value is required.
GPATRN	Reads a pattern from the global pattern library (RTRIEV statement minor word).
HANGLE	Indicates the half-angle of a cone (CONE statement minor word).
HORIZ	Defines a horizontal line through a point (LINE statement minor word). Generates the dimension parallel to the x-axis (LDIMEN and ANGCTL statement minor word). Changes the view to the left or right view (CHANGE statement minor word). Sets a horizontal base for a surface texture symbol (SRFTEK statement minor word).
INFIN	Toggles on the infinite status of a line (LINE statement minor word).
INPUT	Declares the file as an input file (OPEN and BULK statement minor word).
INSIDE	Indicates the dimension line to be generated inside of the arc or circle (CDIMEN statement minor word).
INSTNC	Creates an instance (TEMPLT statement minor word).
INTOF	Indicates the intersection of two curves. Defines a point which is the intersection of two plane curves (POINT statement minor word). Defines a machine curve as surface intersection curve (MACCRV statement minor word).
IRON	Specifies iron material in section-lining (MATERL statement minor word).
JOG	Indicates a horizontal line that is two times the text dimension distance followed by a line directly connected to the datum (DATFEA and GEOTOL minor word).

Word	Description
LARGE	Defines a circle in which a center and tangent circle are provided (CIRCLE statement minor word). Clarifies the ambiguity in any entity definition where using either LARGE or SMALL defines the entity more clearly.
LATHE	Indicates a lathe toolpath (SETGPG and TLPATH statement minor word).
LEAD	Specifies lead material in section-lining (MATERL statement minor word).
LEADR	Indicates a leader line (DATFEA and GEOTOL statement minor word).
LEFT	Indicates that the text origin of a dimension is to be the lower left corner of the text (TXTJUS statement minor word). Places the arrow for a surface texture symbol (SRFTEX statement minor word). Attaches the leader to the midpoint of the left side of the feature frame (DATFEA and GEOTOL statement minor word).
LENGTH	Indicates the length of the section arrowhead (SECARR statement minor word). Indicates the length of the vector (VECTOR statement minor word).
LEVL	Assigns an attribute to all entities of a specified level (ATTRIB statement minor word). Blanks or unblanks all entities of a specified level (BLANKE or UNBLNK statement minor word). Defines the level of a section arrow (SECARR statement minor word).
LIMIT	Adds limits to dimension text (MODDFT statement minor word).
LIN	Indicates a datum target symbol as a line (DATUM statement minor word).
LINES	Creates lines (DEFINE statement minor word).
LLEFT	Attaches the leader to the lower left corner of the feature frame (GEOTOL statement minor word).
LMC	Indicates least material condition (GEOTOL statement minor word).
LOCAL	Indicates that the local method defined by Pochob/Renner is used to calculate the coefficients of the spline (SPCURV statement minor word). Addresses the current part (BULK statement minor word).
LRIGHT	Attaches the leader to the lower right corner of the feature frame (GEOTOL statement minor word).
MAGNES	Specifies magnesium material in section-lining (MATERL statement minor word).
MARBLE	Specifies marble material in section-lining (MATERL statement minor word).

Word	Description
MATRIX	Creates an auxiliary view from the given matrix elements (VIEW statement minor word).
MAXMIN	Sets the view to automatic maximum and minimums (ZOOM statement minor word).
MEMBER	Retrieves the members (entities) which form a group or composite curve (OBTAIN statement minor word).
MIDDLE	Places the text of a dimensioning entity between the dimension line (TXTLOC statement minor word). The leader line starts from the middle text line (LEADER statement modal word in ANSI 82).
MINUTE	Indicates that the angle text of a radial dimension is in degrees and minutes (TXTANG statement modal word) (ANUNIT statement modal word in ANSI 82).
MMC	Indicates maximum material condition (GEOTOL statement minor word).
MODFY	Indicates that the position of the frame may be modified so that a horizontal or vertical line can be drawn (DATFEA and GEOTOL statement minor word).
MODVW	Indicates that the point coordinates are given in modal space (EVALC or EVALS statement minor word).
NAME	Defines or extracts subattribute name(s) from an entity (ATTRIB and OBTAIN statement minor word). Retrieves the named mstring file (MSTRNG statement minor word).
NECK	Indicates the minimum distance between the arrowhead and the first jog of the main body (SECARR statement minor word).
NEGATV	Establishes in note parallel to a line a negative x-direction (NOTE statement minor word). Left-justifies the generated character string (CONVER statement minor word).
NFIXED	Indicates number of fixed curves for curve mesh surface (CMSRF statement minor word).
NOAREA	Indicates a datum target symbol without an area (DATUM statement minor word).
NONE	Indicates that neither side is selected as the origin (DIMORG statement modal word). Indicates that no section arrows are drawn (SECARR statement minor word).
NOPROJ	Indicates a normal projection of a point on a curve (POINT statement minor word).
NORMAL	Specifies a point normal to a surface (POINT and MACCRV statement minor word). Defines a circle normal to the work view (CIRCLE statement minor word).

Word	Description
NORPNT	Indicates no repaint during a view change (CHANGE statement minor word).
NORTN	Suppresses automatic return from execution of MSTRNG (MSTRNG statement minor word).
NORTH	Indicates a north bearing (bearing/distance POINT statement minor word). Locates a template instance from a given origin (TEMPLT statement minor word).
NUMBER	Defines a number. For example, a number of points is defined (entity definition minor word). Defines or extracts subattribute number(s) from an entity (ATTRIB statement minor word). Converts a real number into a character string (CONVER statement minor word). Deletes a number of entities from an entity array (DELETE statement minor word). Retrieves the number of characters in a character string (OBTAIN statement minor word). Unblanks a number of entities (UNBLANK statement minor word). Defines an entity array (PROJEC, SECTON, TLPATH, and SECARR statement minor word).
NUMBRX	Indicates number of times an entity or group of entities is to be repeated in the x-direction (ARRAY statement minor word).
NUMBRY	Indicates number of times an entity or group of entities is to be repeated in the y-direction (ARRAY statement minor word).
NVARY	Indicates number of variable curves for curve mesh surface (CMSRF statement minor word).
OFF	Turns a modal statement off.
ON	Turns a modal statement on.
ORIGIN	Indicates the origin of the auxiliary view (VIEW statement minor word). Defines the origin of a drawing (ZOOM statement minor word).
OUT	Generates the dimension line outside of the circle or arc, pointing toward the arc (CDIMEN statement minor word).
OUTPUT	Declares a file as an output file (OPEN and BULK statement minor word).
PARAM	Indicates the start and end parameters of a direction line (TABCYL statement minor word). Indicates the parameters of an entity are to be returned (OBTAIN statement minor word). Returns point coordinates from a specified parameter (EVALC and CURARR statement minor word). Returns u and v point coordinates from a specified parameter (EVALS statement minor word).

Word	Description
PARLA	Indicates that the witness line is parallel to the corresponding line or circle (dimension statement minor word).
PARLEL	Defines a line parallel to an existing line (LINE statement minor word). Generates the desired dimension parallel to the given line or the line that would exist between the given points (LDIMEN and ANGCTL statement minor word).
PARNOR	Generates the desired dimension normal to a set of parallel lines (LDIMEN statement minor word).
PARTNA	Retrieves the current part and sheet number (OBTAIN statement minor word).
PATERN	Retrieves a pattern from the local pattern library (RTRIEV statement minor word).
PENNUM	Indicates the datum target pen number (DATUM statement minor word).
PERPTO	Defines a line perpendicular to a given line (LINE statement minor word). Specifies a plane perpendicular to another plane or a vector (PLANE statement minor word).
PHANTM	Defines all subsequent curves as phantom until a new font is entered (FONT statement minor word).
PIERCE	Defines a surface pierce point (POINT and MACCRV statement minor word).
PLASTC	Specifies plastic material in section lining (MATERL statement minor word in ANSI 82).
POINTS	Creates points (DEFINE statement minor word). Deletes all points (DELETE statement minor word). Returns the parameter of a specified point (EVALC and EVALS statement minor word). Indicates plane definition using three edge points (PLANE statement minor word).
POSITV	Establishes a positive x-direction for a note parallel to a line (NOTE statement minor word). Generates a character string right-justified in a 10-character array with leading blanks (CONVER statement minor word).
POSN	Indicates the origin position. The origin coordinates are supplied directly by the user's graphic input (screen position) (DORIG and TXTORG statement minor word). Produces a dual dimension with a line separating the upper and lower dimensions (DUAL statement minor word).

Word	Description
PTZ	Indicates projected tolerence zone (GEOTOL statement minor word).
RADANG	Defines a spline point using a radius and an angle (SPLINE statement minor word).
RADII	Indicates the inside and outside torus radius (TORUS statement minor word).
RADIUS	Defines the circle radius (entity definition, BALOON, and MAGNFY statement minor word).
RATIO	Indicates the ratio to the current drafting scale (DSCALE statement modal word).
RECT	Defines a rectangular array (ARRAY statement minor word).
REF	Adds or deletes parentheses around dimension text (MODDFT statement minor word).
REGIN	Selects all entities which lie within a region (SELENT statement minor word).
REGOUT	Selects all entities which lie outside a region (SELENT statement minor word).
RELAX	Indicates the relaxed start or end conditions (SPCURV statement minor word).
RFS	Indicates regardless of feature size (GEOTOL statement minor word).
RIGHT	Indicates the text origin of a dimension to be the lower right end of the text (TXTJUS, DATFEA, GEOTOL, and SRFTEX statement minor word).
ROUGH	Indicates lathe roughing toolpath (SETGPG and TLPATH statement minor word).
RTHETA	Defines the polar coordinates of a point (POINT statement minor word).
RUBBER	Specifies rubber material in section-lining (MATERL statement minor word).
SCALAR	Defines a scalar multiple (entity definition statement minor word).
SCALE	Defines a scale factor for drawing a pattern (RTRIEV statement minor word). Zooms to a specified scale (ZOOM statement minor word).
SECDIS	Defines the distance between section lines (SECTON and DATUM statement minor word).

Word	Description
SECOND	Indicates that the second side is selected as the dimension origin (DIMORG statement modal word). Indicates that the angle text of a radial dimension is in degrees, minutes, and seconds (TXTANG and DECMAL statement minor word) (ANUNIT statement minor word).
SEQNUM	Retrieves the sequence number of an entity (OBTAIN statement minor word). Names the entity with that sequence number (ASSIGN statement minor word).
SHEET	Indicates the sheet number of the balloon (BALOON statement minor word).
SINGLE	Selects single entities (SELENT statement minor word).
SLATE	Specifies SLATE material in section-lining (MATERL statement minor word).
SLOPE	Supplies a slope at either a spline start or spline end (SPLINE statement minor word). Indicates the angle of the leader line (LABEL and TAPER statement minor word).
SMALL	Defines a circle in which the center and tangent circle are provided (CIRCLE statement minor word). Clarifies the ambiguity in any entity definition where using either LARGE or SMALL defines the entity more clearly.
SOLID	Defines all subsequent curves as solid lines (FONT statement minor word).
SOUTH	Indicates a south bearing (bearing/distance POINT statement minor word). Locates a template instance from a given origin (TEMPLT statement minor word).
SPHRIC	Defines spherical coordinates (POINT statement minor word).
START	Indicates that the leader line starts at the start of the label (CDIMEN and LABEL statement minor word). Indicates the start point for the start of the intersection (PTSET statement minor word). Indicates the start point of the intersection curve (FILSRF statement minor word). Indicates that the start point of the leader line is at the beginning of the dimension (SECARR statement minor word).
STD	Sets the character set to standard (CSET statement minor word).
STDVW	Retrieves the view pointer of a standard view (OBTAIN statement minor word).
STEEL	Specifies steel material in section-lining (MATERL statement minor word).

Word	Description
SUBAT	Defines or deletes a subattribute of an entity (ATTRIB statement minor word). Retrieves a subattribute from an entity (OBTAIN statement minor word).
SUPPB	Suppresses the display of both witness lines (dimension and WLINE statement minor word).
SUPPB1	Suppresses the display of both witness lines and the first dimension line (dimension and WLINE statement minor word).
SUPPB2	Suppresses the display of both witness lines and the second dimension line (dimension and WLINE statement minor word).
SUPP1	Suppresses the display of the first witness line (dimension and WLINE statement minor word).
SUPP2	Suppresses the display of the second witness line (dimension and WLINE statement minor word).
SUPP11	Suppresses the display of the first dimension line and the first witness line (dimension and WLINE statement minor word).
SUPP22	Suppresses the display of the second dimension line and the second witness line (dimension and WLINE statement minor word).
TABLET	Designates the tablet as the input device (CURSOR statement minor word).
TANTO	Defines a line where the line is tangent to circles, or through a point and tangent to a circle (LINE statement minor word).
TEXTD	Sets the distance from the text of a dimension to its dimension lines (DIMOF statement minor word).
THICK	Indicates the thickness of the section arrow's main body (SECARR statement minor word).
THREAD	Indicates a lathe threading toolpath (SETGPG and TLPATH statement minor word).
TILTAN	Defines a rectangular array at an angle with the horizontal axis (ARRAY statement minor word).
TOLER	Specifies the tolerance ratio (CDISPL statement minor word).
TOLMOD	Indicates the tolerance modifier (GEOTOL statement minor word).
TOLREF	Indicates the tolerance reference (GEOTOL statement minor word).

Word	Description
TOTAL	Writes notes at a user-specified angle parallel to a line or arc.
TRIM	Trims the lines which bound the fillet or chamfer at the fillet tangent points, or chamfer section point (FILLET and CHAMFR statement minor word).
TYPIN	Indicates the user enters origin coordinates (DORIG statement minor word).
UDIREC	Sets the number of points per U-path for surface display (SPATHS statement minor word).
ULEFT	Attaches a leader to the upper left corner of the feature frame (GEOTOL statement minor word).
UNIT	Sets the unit normal vector (VECTOR statement minor word).
UPATHS	Sets the number of U-paths for surface display (SPATHS statement minor word).
URIGHT	Attaches the leader to the upper right corner of the feature frame (GEOTOL statement minor word).
USER	Sets the character set to user defined (CSET statement minor word). Sets entity selection to user input (SELENT statement minor word).
USTART	Writes a new DEFBLK card (USTRUC statement minor word).
UTERM	Terminates a UNISTRUC program (USTRUC statement minor word).
UWRITE	Writes entity data (USTRUC statement minor word).
VDEF	Indicates section lining is displayed in the view of definition only (SECVIS statement modal word).
VDIREC	Sets the number of points per V-path for surface display (SPATHS statement minor word).
VECSUM	Adds two vectors (VECTOR statement minor word).
VERTCL	Defines a vertical line through a point (LINE statement minor word). Generates the desired dimension parallel to the y-axis (LDIMEN statement minor word). Changes the view to the top or bottom view (CHANGE statement minor word). Indicates a vertical base (SRPTEX statement minor word).
VPATHS	Indicates number of V-paths for surface display (SPATHS statement minor word).

Word	Description
WEST	Indicates a west bearing (bearing/distance POINT statement minor word). Locates a template instance from a given origin (TEMPLT statement minor word).
WIDTH	Indicates the width of the section arrowhead (SECARR statement minor word).
WITP	Indicates that the distance of a witness line is offset from the endpoint of the entity at which it is pointing (DIMOF statement minor word).
WITX	Indicates that the distance of a witness line extends past the dimension line (DIMOF statement minor word).
XAXIS	Defines an x-axis (LINE statement minor word). Specifies the mirror axis (MIRROR statement minor word).
XLARGE	Clarifies in any ambiguous entity definition that the desired point has the larger of two possible x-coordinate values.
XMOVE	Allows point movement in the xt direction only (SPLINE statement minor word).
XSMALL	Clarifies in any ambiguous entity definition that the desired point has the smaller of two possible x-coordinate values.
XSTART	Defines the initial x-value in entity definition statements.
XTROT	Performs a rotation about the x-axis to create an auxiliary view (VIEW statement minor word).
XYMOVE	Allows point movement in the xt and yt directions (SPLINE statement minor word).
XYVALUE	Defines an x-value on a line (POINT statement minor word).
YAXIS	Defines a y-axis (LINE statement minor word). Defines the mirror axis (MIRROR statement minor word).
YES	Aligns the present linear dimension arrowheads with those of the previous or user-selected dimension (LDIMEN statement minor word) (SECTON statement minor word in ANSI 82). Defines a group (RTRIEV statement minor word).
YLARGE	Clarifies in any ambiguous entity definition that the desired point has the larger of two possible y-coordinate values.
YMOVE	Allows point movement in yt direction only (SPLINE statement minor word).
YSMALL	Clarifies in any ambiguous entity definition that the desired point has the smaller of two possible y-coordinate values.

Word	Description
YSTART	Defines the initial y value in entity definition statements.
YTROT	Performs a rotation about the y-axis to create an auxiliary view (VIEW statement minor word).
YVALUE	Defines a y value on a line (POINT statement minor word).
ZAXIS	Defines a z-axis in entity definition statements.
ZLARGE	Indicates the desired point has the larger of two (2) possible z-coordinate values for surface definition statements.
ZSMALL	Indicates the desired point has the smaller of two (2) possible z-coordinate values for surface definition statements.
ZSTART	Defines the initial z value in entity definition statements.
ZTROT	Performs a rotation about the z-axis to create an auxiliary view (VIEW statement minor word).
ZVALUE	Defines a z-value on a line (POINT statement minor word).

ICEM DDN Entity Types C

The following lists the entity types that can be used in GPL.

Type	Description
1	Point
2	Line
3	Arc/circle/fillet
4	General conic
5	2-D spline
6	Composite curve
7	Vector
8	Point set
9	3-D spline
10	Machining curve
11	String
12	Rectangular array
13	Circular array
14	Copious data
15	Group
16	Real variable
18	Plane
19	Surface of revolution
20	Tabulated cylinder
21	Ruled/developable surface
22	Curve mesh surface
23	Fillet surface
26	Bezier surface
28	Offset surface
29	Composite surface
30	Curve driven surface
31	Bezier curve
32	Linear dimension
33	Circular dimension
34	General label
35	Diameter dimension
36	Angular (radial) dimension
37	General note
38	Centerline
39	Cross hatching
40	True position symbol
41	Template
45	Systems GPG, Inserts, and NC modal
46	Toolpath
47	Composite toolpath
48	Toolpath GPG and inserts
49	Tool
50	Systems entity

GPL Execution Error Messages D

The following messages can occur during execution.

Message	Description
1	Error in read program file
2	Error in read data file
3	Error in write data file
4	Illegal key
5	Illegal major word
6	Illegal position word
7	Illegal modal or minor word
8	Illegal control byte
9	Illegal data file address
10	Undefined entity
11	Illegal entity type
12	Variable used as entity
13	Entity used as variable
14	Illegal variable or constant
15	Illegal entity subtype
16	End of statement not found
17	Math error: no unique solution
18	Subroutine not in library
19	Number of parameters inconsistent
20	Variable not found in RTL
21	Variable table overflow
22	Modal word missing
23	DO/DONT index out of bounds
24	Invalid conic
25	MIRROR line length too small
26	Too few or too many entities
27	Too few or too many spline points
28	Character set not found in UTF
29	Illegal scale factor
30	SQRT called with negative argument
31	No intersection point found
32	Distance too small
33	Angle out of range
34	Area not uniquely defined
35	Boundary not closed
36	Boundary too complex
37	Maximum iteration count exceeded
38	Division by nearly zero
39	Entities must be of same type
40	Curves not contiguous
41	The planes are parallel
42	No edge point found to start
43	Both lines have zero length
44	Lines are parallel
45	Plane definition is not unique
46	Axis length too small
47	Directrix length too small

Message	Description
48	Character set name too long
49	CLINE points not coincident
50	No plane between the lines selected
51	Distance too large for FILLET
52	Bad count for data file I/O
53	Internal error: cannot find any PTE (GLRDF)
54	Buffer conflict in GLFLDF
55	Curve(s) is/are outside of view
56	Line is parallel to coordinate
57	Text center is outside limit
58	Membership count is too high
59	Statement not yet implemented
60	View matrix is not orthonormal
61	Data input for SPCURV incorrect
62	Two consecutive points are coincident
63	Tangent for SPCURV is zero
64	PATTERN disk read error
65	Sequence number not found
66	Error on file MESHFIL
67	Too many views selected
68	Could not match curvatures
69	Cannot file part
70	Range error in computed GOTO
71	Function argument out of range
72	Too many attribute data
73	Vector has zero length
74	Entity name already exists in RTL
75	Overlay or capsule library search fails
76	Overlay or capsule too large
77	Overlay or capsule cannot be linked
78	Modify in wrong view
79	Tried to read past EOF
80	No intersection point found (TOL)
81	TMPLT: Master undefined our unlinked
82	TMPLT: Master already defined
83	Cannot define machining draft curve
84	Tolerance too small for draft curve
85	Angle for draft curve out of range
86	Register denotation out of range
87	Character variable too short
88	Character variable too long
89	Cannot restore buffer from cache
90	Error indicator return from ICEM DDN
91	Too many fixed curves (CMSRF)
92	Too many variable curves (CMSRF)
93	Number of curves (F*V) exceeds limit
94	Surface could not be defined (DEVSRF)
95	Both surfaces are infinite planes
96	Array index is not positive
99	Drafting entity cannot be completed
100	Aligning with nonsimilar dimension
101	Illegal geometric characteristic
102	Only 64 entities allowed
103	At least one entity is required

Message	Description
104	Value must be greater than zero
105	Negative value not allowed
106	At least two points required
107	Conversion error occurred
108	Same circle for MAG and MAG area
109	Ten-entity limit attached to arrow
110	At least two JOG positions are required
111	Only 50 JOG positions are allowed
112	Y coord not supplied with X coord
113	Illegal number of datums and tolerances
114	Geometric characteristic required for composite
115	Group member illegal in composite curve
116	File not found
117	GPG not compatible with entity type
118	Generation parameter group not found
119	Definition start indeterminate
120	Invalid number supplied
121	Condition requiring prompt encountered
122	Entity not found
123	Incompatible entity used in definition
124	GPG statement errors, see file - GPGERR
125	GPG file read error
131	Sytax error within menu string
132	Error found while reading MSTRING file
133	Menu string name not on file
134	Menu string file cannot be retrieved
141	No closed boundary can be determined
142	No cut passes could be generated
143	Material to be removed outside boundary
144	Blank entity not found
145	Contour entity not found
151	GPLIS file index or header incorrect
152	Blanked entity is not selectable
154	Cannot find normal/pierce point

System I/O Commands E

CAUTION

The statements documented in this appendix are recommended for experienced users only. If you use the statements improperly, you could destroy the TAPE3 part file. Refer to the ICEM DDN System Programmer's Reference Manual for information about the ICEM DDN COMMON arrays.

EC(3) and EC(4) are not accessible by this command. A value of 0 will be returned without error if an attempt is made to access them.

The statement:

```
SEQNO/value
```

sets the next sequence number to be assigned by ICEM DDN in EC(27) to the specified value, with the restrictions that:

- Value must not be below the current content of EC(27).
- Value must not be above 2**16.

The statement:

```
SYSTEM/type,indicator,index,count,variable[,entity]
```

permits the transfer of data between the system COMMON arrays and GPL variables.

Parameter	Description
type	A number from 1 to 4 that defines the type of operation as follows: 1 Read from integer COMMON array 2 Read from real COMMON array 3 Write to integer COMMON array 4 Write to real COMMON array
indicator	A number from 1 to 20 (according to ICEM DDN menu 6.7.1 display) that indicates which COMMON array is to be addressed.
index	A number from 1 to the size of the addressed array that gives the starting index for the transfer.

Parameter	Description
count	The number of words to be transferred, if positive. If this quantity is negative, the number is assumed to be 1, and count is used as a mask to select certain bit fields, the contents of which are shifted to the right-justified position. For example: `count=-6` reads what is in bits 1 and 2 (note that 2**1 + 2**2=6) and shifts the contents down to positions 0 and 1. A negative count is valid only if type=1.
variable	The name of a GPL real variable, or if cnt/m is greater than 1, the array to receive or give the values transferred.
entity	GPL entity variable containing a valid entity pointer. This parameter is only required if type is 1 or 3 and indicator is 5 (meaning the EC array).

The statement:

```
OBTAIN/{TAB1}
       {TAB2},entity,index,count,{variable}
       {TAB3}                    {entity  }
```

obtains common data from TAB1, TAB2, or TAB3.

Count words are obtained from the indicated array, starting at the specified index for the defined entity; the count words are transferred to the variable.

If the indicator is TAB2, an entity (or entity array) may alternatively be specified as receiving field if the TAB2 content at the specified index consists of entity pointers.

GTGT: GRAPL-to-GPL Translator F

This utility will execute, external to ICEM DDN, via a typical control language program call from within a batch job or procedure, or interactively from the terminal.

Example:

```
GTGT.
```

GTGT will have the capability of substituting local file names for the input, output, and list files.

Example:

```
GTGT (I=INPF, O=OUTF, L=LISTF)
```

If these local file names are not supplied, as in the former example above, the default local file names GRAIOF, COMPILE, and GPLLIST will be assumed for input, output, and list respectively. (GRAIOF and COMPILE are also default input file names for GRAPL and GPL source, respectively, in ICEM DDN and the GPL compiler.)

The input file (default GRAIOF) may be a multi-record file. Each record of GRAIOF contains one GRAPL program preceded by a header (the program name) of from one to six alphanumeric characters. This header will be translated to a "PROC/progname" statement in GPL, where "progname" is the header, and written to the output file as line 1. Line 2 written to the output file will contain the REMARK: $$CAUTION- IF PROGRAM, CHANGE "PROC" TO "MAIN".

Some statements do not translate. In these cases informative remarks are written to the output file.

GPL Program Examples G

Flange Program

Figure G-1 depicts a flange, which is a standard part in the mechanical design environment. Usually, the dimensions are defined as variables, and in practice, the variables are replaced by dimensions that meet the total design criteria.

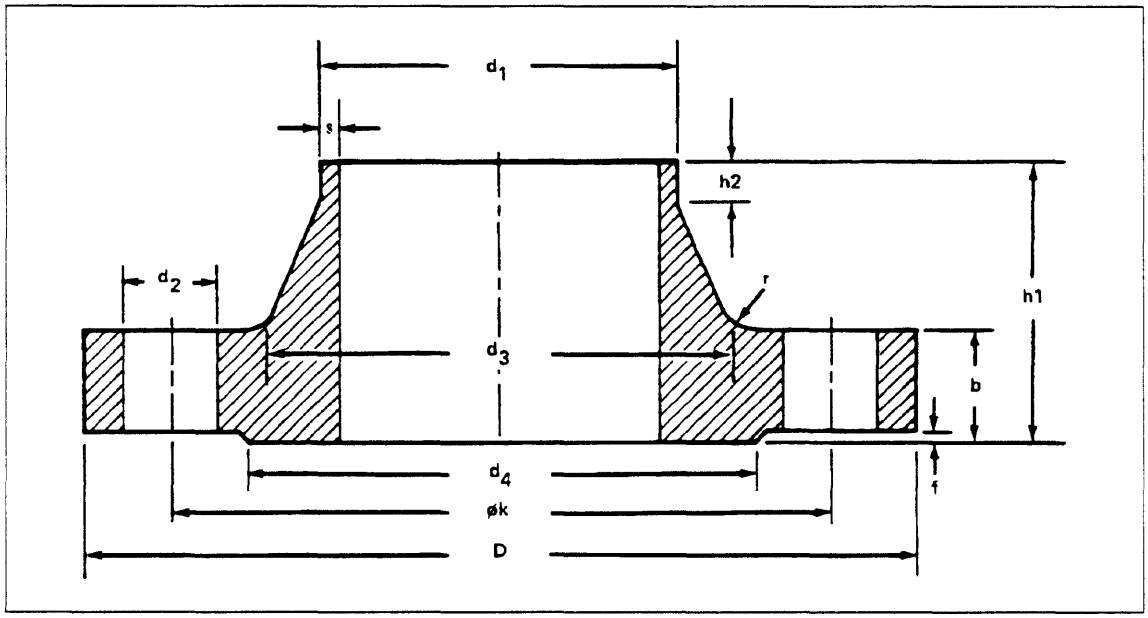


Figure G-1. Standard Welding Neck Flange

This example explains how the flange is defined using the GPL language. In this example, you can modify all of the dimensions in an interactive mode. The variables are input via the READ feature of GPL. The input file is defined as FLANSCH.

The program documentation explains the programming techniques used to define the part.

You are encouraged to study the enclosed program listing to gain a better understanding of GPL and the programming techniques used in this example.

Figure G-2 shows the layout used for FLANSCH.

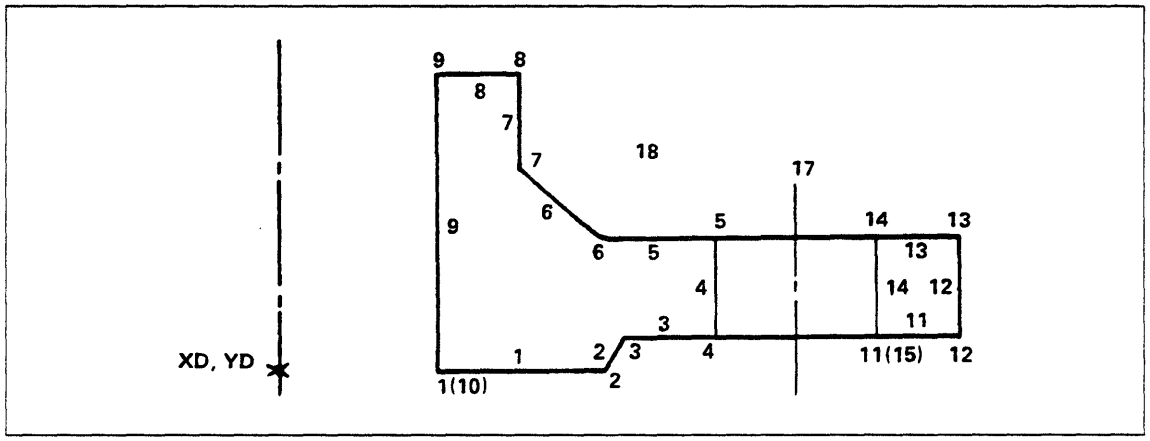


Figure G-2. Flange Layout

The definition of variables for figure G-2 follows.

```
X(1) = D1/2-S
Y(1) = 0
X(2) = D4/2
Y(2) = 0
X(3) = X(2)+F
Y(3) = F
X(4) = (K-D2)/2
Y(4) = F
X(5) = X(4)
Y(5) = B
X(6) = D3/2
Y(6) = B
X(7) = D1/2                    $$
Y(7) = H1-H2                   $$ MIRROR ABOUT Y-AXIS
X(8) = D1/2                    $$
Y(8) = H1                      FOR I=1, UNTIL 10
X(9) = X(1)                    X(I) =  X(I)
Y(9) = H1                      EOFI
X(10) = X(1)                   $$
Y(10) = Y(1)                   10 CONTIN
```

The following listing shows the input file FLANSCH.

200	219.1	340	24	295	62	235	5.9	10	16	268	3	22
250	273	395	26	350	68	285	6.3	12	16	320	3	22
300	323.9	445	26	400	68	344	7.1	12	16	370	4	22
350	355.6	505	26	460	68	385	7.1	12	16	430	4	22
400	406.4	565	26	515	72	440	7.1	12	16	482	4	26
500	508	670	28	620	75	542	7.1	12	16	586	4	26
600	610	780	28	725	80	642	7.1	12	18	885	5	30
700	711	895	30	840	80	745	8	12	18	800	5	30
800	813	1015	32	950	90	850	8	12	18	905	5	33
900	914	1115	34	1050	95	950	10	12	20	1005	5	33
1000	1016	1230	34	1160	95	1052	10	16	20	1110	5	36
1200	1220	1455	38	1380	115	1255	11	16	25	1350	5	39
1400	1420	1675	42	1590	120	1460	12	16	25	1535	5	42
1600	1620	1915	46	1820	130	1665	14	16	25	1760	5	48
1800	1820	2115	50	2020	140	1868	15	16	30	1960	5	48
2000	2020	2325	54	2230	150	2072	16	16	30	2170	5	48
2200	2220	2550	58	2440	160	2275	18	18	35	2370	6	56
2400	2420	2760	62	2650	170	2478	20	18	35	2570	6	56
2600	2620	2960	66	2850	180	2680	22	18	40	2780	6	56
2800	2820	3180	70	3070	190	2882	22	18	40	3000	6	56
3000	3020	3405	75	3290	200	3085	24	18	45	3210	6	62

The following is a listing of the program FLANGE. This program requires that the input data file FLANSCH is also attached.

```
MAIN/FLANGE
$$
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$ *            MAIN PROGRAM FOR FLANGE                *
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$
$$ *** MAIN PROGRAM FOR DRAWING AND DIMENSIONING
$$ *** A FLANGE UTILIZING THE READ FEATURE OF GPL
$$ *** TO GET THE DIMENSIONS FOR THE FLANGE TO
$$ *** BE CREATED.
$$
$$ *** LOCAL DECLARATIONS---
$$
REAL/E(2),ARR(3),STAT,XPOS,YPOS,ZPOS,ANS
$$
$$ *** ASK USER FOR NOMINAL DIAMETER AND ROTATION ANGLE
$$
1000 CONTIN
PARAMS/'*** PLEASE TYPE IN ***',2,'NOM. DIAM.','ANGLE',E(1),STAT
NENN    = E(1)
WINKEL  = E(2)
$$
$$ *** DETERMINE THE BASE COORDINATES OF THE FLANGE
$$
10  CALL/GRPOS,XPOS,YPOS,ZPOS
XD = XPOS
YD = YPOS
$$
$$ *** READ DIMENSIONS ACCORDING TO NOMINAL DIAMETER
$$
20 CALL/TABELE,NENN,D,D1,D2,D3,D4,S,K,H1,H2,F,B,R
$$
$$ *** DRAW THE FLANGE AND DISPLAY THE DIMENSIONS
$$
CALL/GEOMET,D,D1,D2,D3,D4,S,K,H1,H2,F,B,R,WINKEL,XD,YD
$$
$$ *** ALL DONE, ASK IF USER WANTS ANOTHER FLANGE
$$
QUERY/'ANOTHER FLANGE (Y/N)? ',ANS
IF ( ANS .EQ. 1. )      GO TO 1000
$$
$$ *** TERMINATE PROGRAM,  USER ANSWERED NO
$$
STOP/'ALL DONE'
FINI
```

TABELE Subprogram

The following is a listing of the subprogram TABELE.

```
PROC/TABELE,NENN,D,D1,D2,D3,D4,S,K,H1,H2,F,B,R
$$
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$ *            PROCEDURE TABELE                          *
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$
$$ *** PROCEDURE TO READ THE DIMENSIONS OF THE FLANGE
$$ *** GIVEN THE NOMINAL DIAMETER (NENN).
$$
REAL/NENN,NW,D,D1,D2,D3,D4,B,H1,H2,S,K,R,F
CHAR/LFN(7)
$$
$$ *** OPEN DATA FILE
$$
MOVCHR/7,'FLANSCH',1,LFN,1
OPEN/LFN,INPUT,STAT
$$
$$ *** CHECK FOR ERRORS
$$
IF ( STAT .NE. 0. )              GO TO 8000
$$
$$ *** SET COUNT OF VARIABLES TO READ
$$
COUNT = 13
$$
$$ *** READ DATA FILE UNTIL NOMINAL DIAMETER FOUND.
$$ *** IF EOF GO TO ERROR EXIT AND PRINT MESSAGE.
$$
1000 READ/LFN,STAT,COUNT,NW,D1,D,B,K,H1,D3,S,R,H2,$
                              D4,F,D2
$$
$$ *** CHECK FOR EOF
$$
IF ( STAT .EQ. 1. )              GO TO 8100
$$
$$ *** CHECK FOR OTHER ERROR (CONVERSION)
$$
IF ( STAT .NE. 0. )              GO TO 8200
$$
$$ *** CHECK IF VALUES FOUND
$$
IF ( NW .NE. NENN )              GO TO 1000
$$
```

```
$$ *** NOMINAL DIAMETER OF FLANGE FOUND IN DATA FILE
$$ *** CLOSE DATA FILE, AND RETURN TO CALLING
$$ *** PROGRAM.
$$
CLOSE/LFN,STAT
JUMPTO/9000
$$
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$ *           E R R O R   S E C T I O N                   *
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$
8000 DISPLA/'OPEN ERROR'
     CLOSE/LFN,STAT
     STOP/'OPEN ERROR'
8100 DISPLA/'NOMINAL DIAMETER NOT FOUND ON FILE'
     CLOSE/LFN,STAT
     STOP/'NOT FOUND'
8200 DISPLA/'CONVERSION ERROR/CORRECT DATA FILE !'
     CLOSE/LFN,STAT
     STOP/'CONV ERROR'
$$
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$ *           N O R M A L   E X I T                       *
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$
9000 RETURN
     FINI
```

GEOMET Subprogram

The following is a listing of the subprogram GEOMET.

```
PROC/GEOMET,D,D1,D2,D3,D4,S,K,H1,H2,F,B,R,WINKEL,XD,YD
$$
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$ *                PROCEDURE GEOMET                     *
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$
$$ *** PROCEDURE TO CALCULATE THE CONTOURS OF THE FLANGE
$$
REAL/X(40),Y(40),R
ENTITY/LN(40)
$$
$$ *** PRESET X AND Y ARRAYS
$$
X(1)  = D1/2-S
Y(1)  = 0
X(2)  = D4/2
Y(2)  = 0
X(3)  = X(2)+F
Y(3)  = F
X(4)  = (K-D2)/2
Y(4)  = F
X(5)  = X(4)
Y(5)  = B
X(6)  = D3/2
Y(6)  = B
X(7)  = D1/2
Y(7)  = H1-H2
X(8)  = D1/2
Y(8)  = H1
X(9)  = X(1)
Y(9)  = H1
X(10) = X(1)
Y(10) = Y(1)
X(11) = (K + D2) / 2
Y(11) = F
X(12) = D/2
Y(12) = F
X(13) = D/2
Y(13) = B
X(14) = X(11)
Y(14) = B
X(15) = X(11)
Y(15) = F
X(16) = K/2
Y(16) = 0.
X(17) = K/2
Y(17) = B + F
X(18) = D4/2
Y(18) = B + 6.*F
```

```
X(19) = 0.
Y(19) = -F
X(20) = 0.
Y(20) = H1 + F
$$
$$ *** MIRROR ABOUT Y-AXIS
$$
FOR I=1, UNTIL 18
M    = I + 20.
X(M) = -X(I)
Y(M) = Y(I)
EOFI
$$
$$ *** ROTATE COORDINATES ABOUT WINKEL AND
$$ *** TRANSLATE COORDINATES BY XD, YD .
$$
SF = SINF(WINKEL)
CF = COSF(WINKEL)
FOR I=1, UNTIL 38
RETT = X(I)
X(I) = X(I)*CF - Y(I)*SF + XD
Y(I) = RETT*SF + Y(I)*CF + YD
EOFI
$$
$$ *** SET PEN AND LEVEL AND DEFINE THE CONTOUR
$$
LEVEL/100
PEN/3
FOR I=1, UNTIL 14
IF ( I .EQ. 10. )              GO TO 10
J     = I + 1.
LN(I) = LINE/X(I),Y(I),X(J),Y(J)
M     = I + 20.
J     = M + 1.
LN(M) = LINE/X(M),Y(M),X(J),Y(J)
10 EOFI
$$
$$ *** DEFINE FILLETS
$$
CR1 = FILLET/X(18),Y(18),LN( 6),LN( 5),RADIUS,R,TRIM
CR2 = FILLET/X(38),Y(38),LN(25),LN(26),RADIUS,R,TRIM
```

```
$$
$$ *** CREATE CONNECTING LINES
$$
LINE/X( 5),Y( 5),X(14),Y(14)
LINE/X( 4),Y( 4),X(11),Y(11)
LINE/X(25),Y(25),X(34),Y(34)
LINE/X(24),Y(24),X(31),Y(31)
LINE/X( 9),Y( 9),X(29),Y(29)
LINE/X( 1),Y( 1),X(21),Y(21)
$$
$$ *** CREATE CENTERLINE AT LEVEL 200, PEN 0
$$
LEVEL/200
PEN/0
FONT/CENLIN
LN1617 = LINE/X(16),Y(16),X(17),Y(17)
LN3637 = LINE/X(36),Y(36),X(37),Y(37)
LN1920 = LINE/X(19),Y(19),X(20),Y(20)
$$
$$ *** RADIAL DIMENSIONING
$$
200 PT1 = POINT/X(38),Y(38)
CDIMEN/START,INSIDE,CR2,PT1,0,0
DELETE/PT1
$$
$$ *** ALL OTHER DIMENSIONING ARE DONE IN BEMASS
$$
CALL/BEMASS,D,D1,D2,D3,D4,S,K,H1,H2,F,B,R,WINKEL,XD,YD
$$
$$ *** CROSS HATCHING
$$
MATERL/STEEL
DIST = 4.75
WNK  = WINKEL + 45.
SECT1 = SECTON/LN(1),LN(2),LN(3),LN(4),LN(5),CR1,LN(6),LN(7),$
              LN(8),LN(9),SECDIS,DIST,ANGLE,WNK
SECT2 = SECTON/LN(11),LN(12),LN(13),LN(14),SECDIS,DIST,ANGLE,WNK
SECT3 = SECTON/LN(21),LN(22),LN(23),LN(24),LN(25),CR2,LN(26),$
       LN(27),LN(28),LN(29),SECDIS,DIST,ANGLE,WNK
SECT4 = SECTON/LN(31),LN(32),LN(33),LN(34),SECDIS,DIST,ANGLE,WNK
$$
$$ *** ALL DONE, RETURN TO CALLING PROGRAM
$$
RETURN
FINI
```

BEMASS Subprogram

The following is a listing of the subprogram BEMASS.

```
PROC/BEMASS,D,D1,D2,D3,D4,S,K,H1,H2,F,B,R,WINKEL,XD,YD
$$
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$ *           PROCEDURE FOR DIMENSIONING                    *
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$
$$ *** HORIZONTAL AND VERTICAL DIMENSIONING
$$
REAL/X(33),Y(33)
$$
$$ *** VARIABLE DESCRIPTIONS---
$$ ***          X(I)    , Y(I)      -   FIRST COORDINATE
$$ ***          X(I+1)  , Y(I+1)    -   SECOND COORDINATE
$$ ***          X(I+2)  , Y(I+2)    -   CENTER OF TEXT
$$
DIS1  = B
DIS2  = 1.5 * B
DIS3  = 2.0 * B
DIS4  = 2.5 * B
X(1)  = -D1/2
Y(1)  = H1
X(2)  = D1/2
Y(2)  = H1
X(3)  = 0
Y(3)  = H1 + DIS1
X(4)  = -D1/2
Y(4)  = H1
X(5)  = -D1/2 + S
Y(5)  = H1
X(6)  = X(5) + 3.*S
Y(6)  = H1 + DIS1
X(7)   = -K/2 -D2/2
Y(7)  = B
X(8)   = -K/2 + D2/2
Y(8)  = B
X(9)  = -K/2
Y(9)  = B + DIS1
X(10) = -D3/2
Y(10) = B
X(11) = D3/2
Y(11) = B
X(12) = 0.
Y(12) = -DIS1
X(13) = -D4/2
Y(13) = 0.
X(14) = D4/2
Y(14) = 0.
X(15) = 0.
Y(15) = -DIS2
X(16) = -K/2
Y(16) = 0.
X(17) = K/2
```

```
Y(17) = 0.
X(18) = 0.
Y(18) = -DIS3
X(19) = -D/2
Y(19) = 0.
X(20) = D/2
Y(20) = 0.
X(21) = 0.
Y(21) = -DIS4
X(22) = D1/2
Y(22) = H1 - H2
X(23) = D1/2
Y(23) = H1
X(24) = D1/2 + DIS1
Y(24) = H1 - H2/2
X(25) = D/2
Y(25) = 0.
X(26) = D/2
Y(26) = F
X(27) = D/2 + DIS1
Y(27) =  4.*F
X(28) = D/2
Y(28) = 0.
X(29) = D/2
Y(29) = B
X(30) = D/2 + DIS2
Y(30) = B/2
X(31) = D1/2
Y(31) = 0.
X(32) = D1/2
Y(32) = H1
X(33) = D/2 + DIS3
Y(33) = H1/2
$$
$$ *** ROTATE COORDINATES ABOUT WINKEL AND
$$ *** TRANSLATE COORDINATES BY XD, YD .
$$
SF = SINF(WINKEL)
CF = COSF(WINKEL)
FOR I=1, UNTIL 33
RETT = X(I)
X(I) = X(I)*CF - Y(I)*SF + XD
Y(I) = RETT*SF + Y(I)*CF + YD
EOFI
```

```
$$
$$ *** DIMENSIONING
$$
FOR I=1, STEP 3, UNTIL 31
J    = I + 1.
M    = I + 2.
XDIF = X(J) - X(I)
YDIF = Y(J) - Y(I)
DIFF = SQRT(XDIF*XDIF+YDIF*YDIF)
ARIN
IF ( DIFF .GT. 30. )            GO TO 10
AROUT
10 CONTIN
PT1 = POINT/X(I),Y(I)
PT2 = POINT/X(J),Y(J)
PT3 = POINT/X(M),Y(M)
LDIMEN/PARLEL,PT1,PT2,PT3,0,0
DELETE/PT1,PT2,PT3
EOFI
$$
$$ *** ALL DONE, RETURN TO CALLING PROGRAM
$$
        RETURN
        FINI
```

GRPOS Subprogram

The following is a listing of the subprogram GRPOS.

```
PROC/GRPOS,XPOS,YPOS,ZPOS
$$
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$ *                PROCEDURE GRPOS                      *
$$ * * * * * * * * * * * * * * * * * * * * * * * * * * * *
$$
$$ *** PROCEDURE TO GET THE BASE COORDINATES FOR
$$ *** THE FLANGE.
$$
$$      1. SELECT SCREEN POSITION
$$      2. KEYIN SCREEN COORDINATES
$$      3. SELECT EXISTING POINT
$$
REAL/KOOR(3),XPOS,YPOS,ZPOS,STAT,CNT,ISW
ENTITY/ENT(2)
$$
$$ *** GIVE USER THE CHOICE TO SELECT COORDINATES
$$
5 MENU/'** FLANGE BASE COORDINATES **','SCREEN POSITION*$
KEYIN COORDINATES*EXISTING POINT*',ISW,STAT
$$
$$ *** CHECK FOR REJECT
$$
IF ( STAT .GE. 0. )            GO TO 9999
IF ( ISW  .NE. 1. )            GO TO 20
$$
$$ *** SELECT SCREEN POSITION
$$
10  POS/'** INDICATE SCREEN POSITION **',XPOS,YPOS,ZPOS,STAT
IF ( STAT .GE. 0. )            GO TO 5
JUMPTO/9999
$$
$$ *** KEYIN COORDINATES
$$
20 IF ( ISW .NE. 2 )           GO TO 30
PARAMS/'**  ENTER COORDINATES **',3,'X-COORD','Y-COORD','Z-COORD',$
                                    KOOR(1),STAT
IF ( STAT .EQ. 0. )            GO TO 5
XPOS = KOOR(1)
YPOS = KOOR(2)
ZPOS = KOOR(3)
JUMPTO/9999
```

```
$$
$$ *** SELECT EXISTING POINT
$$
30 IF ( ISW .NE. 3. )          GO TO 5
SELECT/'** SELECT EXISTING POINT **',2,1,ENT(1),CNT,STAT
IF ( STAT .NE.  1. )           GO TO 5
$$
$$ *** GET COORDINATES OF POINT FROM DATABASE
$$
OBTAIN/COORD,ENT(1),KOOR(1)
XPOS = KOOR(1)
YPOS = KOOR(2)
ZPOS = KOOR(3)
 $$
 $$ *** ALL DONE, RETURN TO CALLING PROGRAM
 $$
 9999 RETURN
      FINI
```

Figure G-3 shows an example of a welding neck flange generated with a diameter of 200 units of measure.

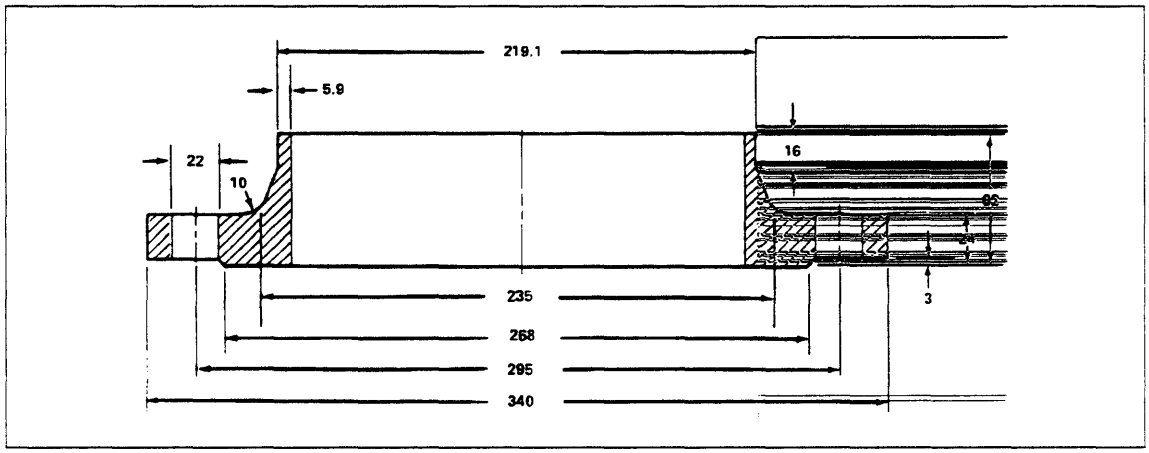


Figure G-3. First Variant of Welding Neck Flange Generated by GPL Nominal Diameter 200 Units of Measure

Figure G-4 shows an example of a welding neck flange generated with a diameter of 250 units of measure.

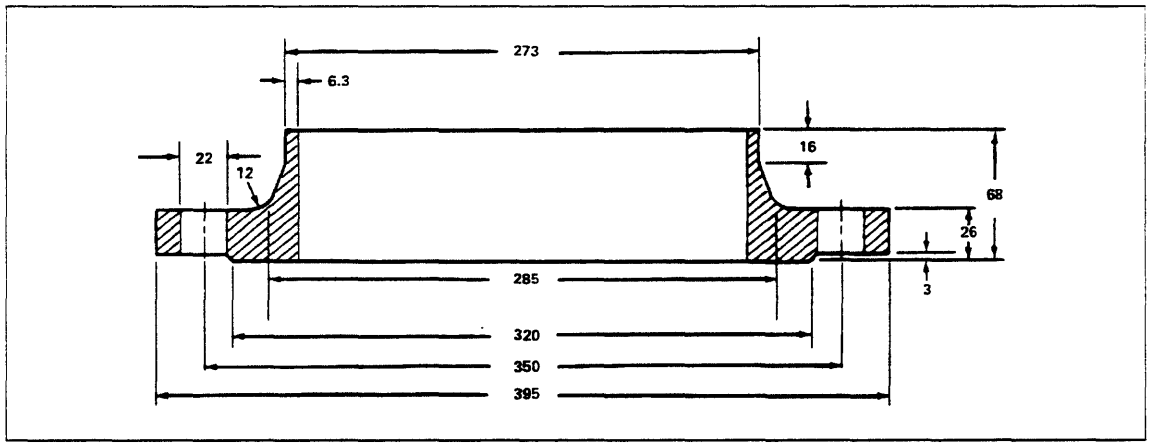


Figure G-4. Second Variant of Welding Neck Flange Generated by GPL Nominal Diameter 250 Units of Measure

Bushing Program

This example defines the automated design of a bushing (figure G-5). The program with which the bushing is generated is named BUCHSE. The program documentation is self-explanatory. The following is a listing of the program BUCHSE.

```
MAIN/BUCHSE
$$
$$ *** RESERVE STORAGE FOR VARIABLES
$$
REAL/VARI(8),COOR(3),IANS,STAT,XM,YM,ZM
$$
$$ *** DEFINE DEFAULT VALUES
$$
A    = 40.
B    = 16.
C    = 32.
D    = 39.
E    = 5.5
F    = 3.5
G    = 30.
THRD = 36.
$$
$$ *** PRESET ARRAY WITH DEFAULTS
$$
VARI(1) = A
VARI(2) = B
VARI(3) = C
VARI(4) = D
VARI(5) = E
VARI(6) = F
VARI(7) = G
VARI(8) = THRD
COOR(1) = 0.
COOR(2) = 0.
COOR(3) = 0.
$$
$$ *** ASK USER IF HE WANTS TO SEE AN EXAMPLE OF THE 'BUCHSE'
$$
5 QUERY/'** DO YOU WANT AN EXAMPLE (Y/N) ?',IANS
IF ( IANS .EQ. 1 )          GOTO 20
$$
$$ *** ASK USER FOR SCREEN LOCATION TO POSITION THE PATTERN
$$
POS/'** INDICATE POSITION OF BUCHSE ',XM,YM,ZM,STAT
IF ( STAT .GE. 0. )           GO TO 1000
$$
$$ *** ASK FOR 2 VIEWS (TOP/SIDE)
$$
QUERY/'** TWO VIEWS (Y/N) ?',IANS
IF ( IANS .NE. 1. )           GO TO 10
```

```
$$
$$ *** NOW RETRIEVE THE PATTERN FROM THE LOCAL PATTERN
$$ *** LIBRARY, PATTERN WANTED AS A GROUP (IND. BY YES)
$$
RTRIEV/PATERN,'BM2',XM,YM,ZM,YES,STAT
IF ( STAT .LT. 0. )            GO TO 1000
JUMPTO/20
$$
$$ *** ONLY ONE VIEW WANTED, SO GET OTHER PATTERN
$$
10 RTRIEV/PATERN,'BM1',XM,YM,ZM,YES,STAT
IF ( STAT .LT. 0. )            GO TO 1000
$$
$$ *** ASK USER IF HE WANTS TO SEE THE DEFAULT VALUES
$$
20 QUERY/'** LIST AND CHANGE DEFAULTS (Y/N) ?',IANS
IF ( IANS .NE. 1. )            GO TO 50
$$
$$ *** USER WANTS TO CHANGE THE DEFAULTS
$$
30 PARAMS/'LIST OF VARIABLES : ',8,'A','B','C','D','E','F','G',$
         'THRD',VARI(1),STAT
IF ( STAT .EQ. 0. )            GO TO 1000
$$
$$ *** THE  SELECTION  OF THE  SCREEN COORDINATES  TO
$$ *** POSITION THE PATTERN IS DONE BY PROCEDURE 'SEL'.
$$
COOR1 = COOR(1)
COOR2 = COOR(2)
COOR3 = COOR(3)
BEEND = 0
CALL/SEL,COOR1,COOR2,COOR3,BEEND
$$
$$ *** IF NOT ANOTHER 'BUCHSE' WANTED, EXIT PROGRAM
$$
IF ( BEEND .EQ. 1. )           GO TO 1000
$$
XB = COOR1
YB = COOR2
ZB = COOR3
$$
$$ *** SET PEN NUMBER 3  AND LINE STYLE TO SOLID
$$
PEN/3
FONT/SOLID
A1   = VARI(1)/2.
YB1  = YB+A1
```

```
$$
$$ *** CREATE LINES
$$
LIN1     = LINE/XB,YB,ZB,XB,YB1,ZB
XB1      = XB+VARI(5)
LIN2     = LINE/XB,YB1,ZB,XB1,YB1,ZB
YB2      = YB1-((VARI(1)/2)-(VARI(3)/2))
LIN3     = LINE/XB1,YB1,ZB,XB1,YB2,ZB
XB2      = XB1 + VARI(6)
LIN4     = LINE/XB1,YB2,ZB,XB2,YB2,ZB
WINKEL   = 90 - VARI(7)
VARI(7)  = WINKEL
WINKEL   = VARI(7)
TANWIN   = SINF(WINKEL)/COSF(WINKEL)
XHILF    = (VARI(8)-VARI(3))*0.5/TANWIN
XB3      = XB2 + XHILF
YB3      = YB + VARI(8)/2.
LIN5     = LINE/XB22,YB2,ZB,XB3,YB3,ZB
XB4      = XB + VARI(4)
LIN6     = LINE/XB3,YB3,ZB,XB4,YB3,ZB
LIN7     = LINE/XB4,YB3,ZB,XB4,YB,ZB
BHILF    = YB + VARI(2)/2.
LIN8     = LINE/XB,BHILF,ZB,XB4,BHILF,ZB
PEN/1
FONT/DASHED
LIN9     = LINE/PARLEL,LIN6,YSMALL,1.0
FONT/CENLIN
XP1      = XB - 10.
XP2      = XB + VARI(4) + 10.
LIN10    = LINE/XP1,YB,ZB,XP2,YB,ZB
PEN/3
FONT/SOLID
YYB1     = YB - A1
LIN11    = LINE/XB,YB,ZB,YYB1,ZB
LIN12    = LINE/XB,YYB1,ZB,XB1,YYB1,ZB
YYB2     = YYB1 + VARI(1)/2 - VARI(3)/2
LIN13    = LINE/XB1,YYB1,ZB,XB1,YYB2,ZB
LIN14    = LINE/XB1,YYB2,ZB,XB2,YYB2,ZB
YYB3     = YB - VARI(8)/2
LIN15    = LINE/XB2,YYB2,ZB,XB3,YYB3,ZB
LIN16    = LINE/XB3,YYB3,ZB,XB4,YYB3,ZB
LIN17    = LINE/XB4,YYB3,ZB,XB4,YB,ZB
BBHILF   = YB - VARI(2)/2
LINE18   = LINE/XB,BBHILF,ZB,XB4,BBHILF,ZB
FONT/DASHED
PEN/1
LIN19    = LINE/PARLEL,LIN16,YLARGE,1.0
```

```
$$
$$ *** ASK USER IF HE WANTS SIDE-VIEW TOO
$$
QUERY/'** SIDE-VIEW DESIRED (Y/N) ?',IANS
IF ( IANS .NE. 1. )            GO TO 400
PAINT/OFF
XBS      = XB - ( 45 + VARI(1)*1.5 )
PT10     = POINT/XBS,YB,ZB
PAINT/ON
FONT/CENLIN
PEN/0
YBS1     = YB + VARI(1)*1.5
YBS2     = YB - VARI(1)*1.5
LIN20    = LINE/XBS,YBS,ZB,XBS,YBS2,ZB
XBS1     = XBS + VARI(1)*1.5
XBS2     = XBS - VARI(1)*1.5
LIN21    = LINE/XBS1,YB,ZB,XBS2,YB,ZB
FONT/SOLID
PEN/3
RR       = VARI(1)/2.
CIR10    = CIRCLE/CENTER,PT10,RADIUS,RR
RR       = VARI(2)/2.
CIR11    = CIRCLE/CENTER,PT10,RADIUS,RR
RR       = VARI(8)/2.
CIR12    = CIRCLE/CENTER,PT10,RADIUS,RR
PEN/1
RR       = VARI(8)/2. - 1.
CIR13    = CIRCLE/CENTER,PT10,RADIUS,RR,GOANG,60,ENDANG,390
FONT/DASHED
RR       = VARI(3)/2.
CIR14    = CIRCLE/CENTER,PT10,RADIUS,RR
DELETE/PT10
400 CONTIN
$$
$$ *** ASK USER IF HE WANTS CROSS-HATCHING AND DIMENIONING
$$
QUERY/'** X-HATCH AND DIMENSIONING (Y/N) ?',IANS
IF ( IANS .NE. 1. )            GO TO 600
$$
$$ *** CROSS-HATCHING SECTION
$$
500 CONTIN
PEN/0
SEC1     = SECTON/LIN1,LIN2,LIN3,LIN4,LIN5,LIN6,LIN7,LIN8
SEC2     = SECTON/LIN11,LIN12,LIN13,LIN14,LIN15,LIN16,LIN17,LIN18
```

```
$$
$$ *** DIMENSIONING SECTION
$$
ARROW/ON
AUTOD
DSCALE/1.25
ARIN
DD      = VARI(4)/2.
LDIMEN/HORIZ,LIN1,YLARGE,LIN7,YLARGE,DD,30
AROUT
DD      = VARI(5)/2. + 22.
LDIMEN/HORIZ,LIN1,YLARGE,LIN3,YLARGE,DD,15.
DD      = VARI(2)/2.
LDIMEN/VERTCL,LIN18,XLARGE,LIN8,XLARGE,30.,DD
ARIN
DD      = VARI(8)/2.
LDIMEN/VERTCL,LIN16,XLARGE,LIN6,XLARGE,40.,DD
DD      = VARI(3)/2.
LDIMEN/VERTCL,LIN14,XSMALL,LIN4,XSMALL, 35.,DD
DD      = VARI(1)/2.
LDIMEN/VERTCL,LIN12,XSMALL,LIN2,XSMALL, 40.,DD
DD      = VARI(1)/2. + 5.
EE      = -DD
AROUT
LDIMEN//HORIZ,LIN14, 15.,EE
PAINT/OFF
PT100   = POINT/INTOF,XLARGE,LIN14,LIN15
DD      = VARI(1)/2.
LIN20   = LINE/PT100,ATANGL,90.,LIN14,DISTNC,DD
PT101   = POINT/PT100,RTHETA,DD,240.
PAINT/ON
ARIN
WW      = -DD
RD1     = RDIMEN/YSMALL,LIN20,YSMALL,LIN15,PT101,0,WW
DELETE/PT100,PT101,LIN20
$$
$$ *** ASK USER IF HE WANTS ANOTHER "BUCHSE"
$$
600 CONTIN
QUERY/'** ANOTHER "BUCHSE" (Y/N) ?',IANS
IF ( IANS .EQ. 1. )            GO TO 5
 $$
 $$ *** TERMINATE PROGRAM
 $$
 1000 CONTIN
 FINI
```

Figure G-5 shows a bushing generated by program BUCHSE.

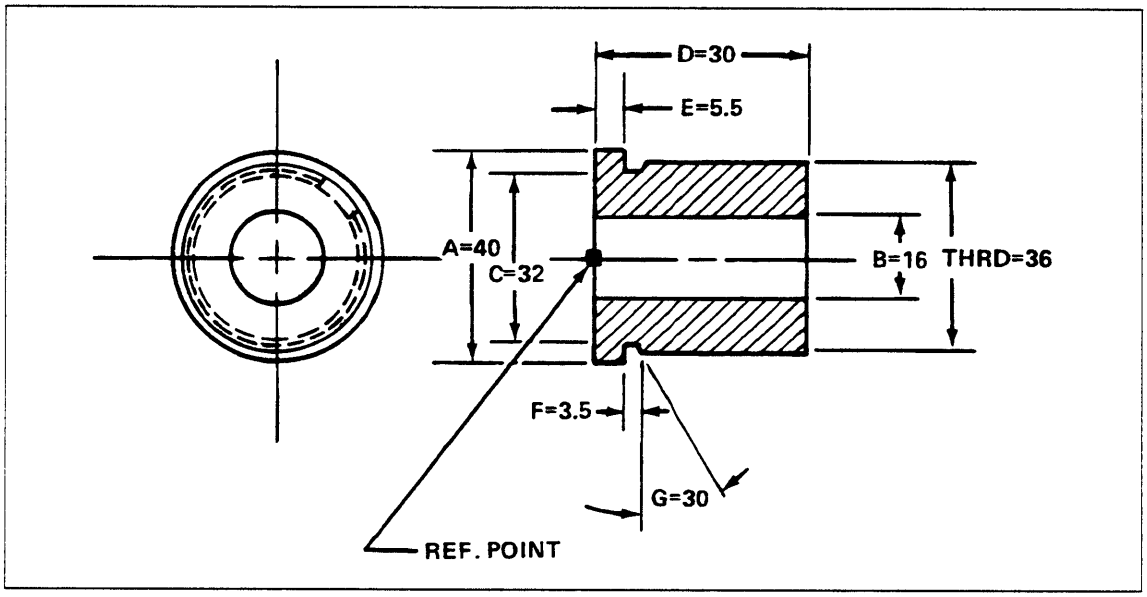


Figure G-5. Default Bushing Generated by BUCHSE

Figure G-6 shows a bushing generated with different dimensions.

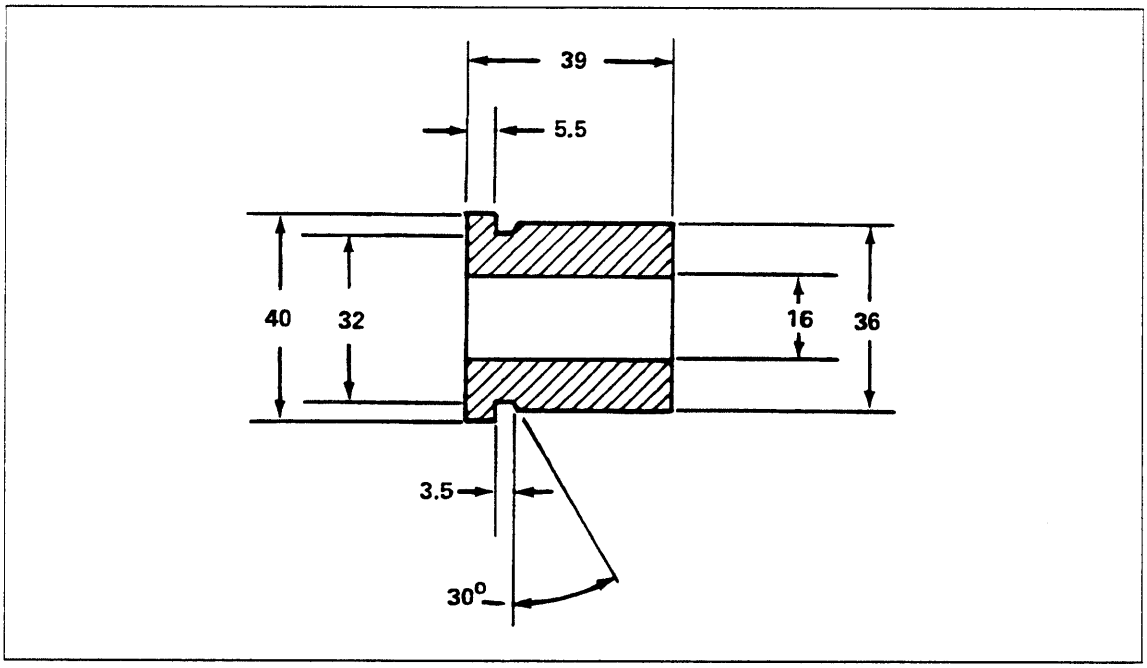


Figure G-6. Bushing Variant

Index

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

We value your comments on this manual. While writing it, we made some assumptions about who would use it and how it would be used. Your comments will help us improve this manual. Please take a few minutes to reply.

Who are you?

- ☐ Manager
- ☐ Systems analyst or programmer
- ☐ Applications programmer
- ☐ Operator
- ☐ Other _____

How do you use this manual?

- ☐ As an overview
- ☐ To learn the product or system
- ☐ For comprehensive reference
- ☐ For quick look-up

What programming languages do you use? _____

_____

How do you like this manual? Check those questions that apply.

Yes	Somewhat	No	
☐	☐	☐	Is the manual easy to read (print size, page layout, and so on)?
☐	☐	☐	Is it easy to understand?
☐	☐	☐	Does it tell you what you need to know about the topic?
☐	☐	☐	Is the order of topics logical?
☐	☐	☐	Are there enough examples?
☐	☐	☐	Are the examples helpful? (☐ Too simple? ☐ Too complex?)
☐	☐	☐	Is the technical information accurate?
☐	☐	☐	Can you easily find what you want?
☐	☐	☐	Do the illustrations help you?

Comments? If applicable, note page and paragraph. Use other side if needed.

Would you like a reply? ☐ Yes ☐ No

From:

Name _____ Company _____

Address _____ Date _____

_____ Phone _____

_____

Please send program listing and output if applicable to your comment.

Comments (continued from other side)

Please fold on dotted line;
seal edges with tape only.

FOLD

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
First-Class Mail Permit No. 8241 Minneapolis, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA
Technology & Publications Division
ARH219
4201 N. Lexington Avenue
Arden Hills, MN 55126-6198

Command Index

The following indexes the GPL major words by page numbers.