International Standard ISO 4342

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION•МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ•ORGANISATION INTERNATIONALE DE NORMALISATION

Numerical control of machines — NC processor input — Basic part program reference language

Commande numérique des machines — Données d'entrée des processeurs CN — Langage de référence de base pour programme de pièce

First edition — 1985-12-15

UDC 681.323 : 621.9-52 : 681.3.06 **Ref. No. ISO 4342-1985 (E)**

Descriptors : machine tools, automation, data processing, numerical control, programming (computers), programming languages.

Price based on 278 pages

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work.

Draft International Standards adopted by the technical committees are circulated to the member bodies for approval before their acceptance as International Standards by the ISO Council. They are approved in accordance with ISO procedures requiring at least 75 % approval by the member bodies voting.

International Standard ISO 4342 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems*.

Users should note that all International Standards undergo revision from time to time and that any reference made herein to any other International Standard implies its latest edition, unless otherwise stated.

Printed in Switzerland

Contents

Annexes

This page intentionally left blank

Numerical control of machines — NC processor input — Basic part program reference language

0 Introduction

0.1 General features

0.1.1 A part program is an ordered set of instructions in a language and in a format required to cause the generation and commands for operations to be carried out under numerical control (NC). The instructions of the language are processed sequentially in two stages, processor and postprocessor. The processor stage is substantially independent of the numerically controlled machine, and the postprocessor stage takes account of the features of the machine and control system to be numerically controlled.

0.1.2 Numerical control is applied to many types of machine, but the language defined in this International Standard has been developed primarily for numerically controlled machine tools — hence the words "cutter" or "tool", and "workpiece" or "part", are used in the description of the language to indicate the working element and processed element respectively. Many of the reserved words of the language are also derived from metalworking terminology.

0.1.3 This International Standard is partly informative and partly definitive, with the intention of enabling unsophisticated users and potential users to discuss and define their requirements with suppliers. The formal syntax description of the language has been included to aid implementers.

0.1.4 The language described in this International Standard is a reference language (RL). It has been divided into logical pages according to the type of statement or facility, with the intention that each logical page will become a unit of further standardization work. It is also intended that each logical page is suitably identified as belonging to the core or a module of the total reference language.

0.1.5 The output from the processor (CLDATA — derived from the term "cutter location data"), which is also input to the postprocessor, is the subject of separate standardization documents. The definition of CLDATA effectively encompasses the part programming language necessary for using capabilities of the postprocessor, and the CLDATA standard ISO 3592 is the prime reference source for this part of the total language.

0.1.6 Some processors may interface with workshop technology processors. That part of the total language which encompasses the use of workshop technology will be described in an addendum to this International Standard.

0.2 Numerical control reference language

0.2.1 The numerical control (NC) reference language (RL) is a problem-oriented language developed for the machining of parts. It is similar to scientific programming languages, and contains many of the facilities and a large part of the computational ability of these; additionally the RL provides for both the description of shapes and commands for movement of the machine.

0.2.2 The RL is written in lines, and for the purpose of describing the RL, the level of communication assumed is the manuscript stage of programming. There is no special character for statement termination, but there is a special character to signify that a statement continues from one line to the next. This concept permits the description of the RL itself to be distinguished from the rules for representation of the RL on various media such as punched cards or paper tape. The rules for representing the RL on punched cards are given in annex A. The syntax description of the reference language is given in annex B.

0.2.3 The RL is a symbolic language; that is, an entity may be assigned to a symbolic name and the entity referenced later in the part program by that symbolic name. For other than arithmetical values a symbolic name may not normally be reassigned.

0.2.4 Unlike scientific programming languages there are no explicit or implicit type associations in the RL. The type of entity is determined by the type of statement in which the entity is defined.

0.2.5 The arithmetical type of entity has always a real value, that is, it is an approximate representation of a real number. It has an integer part and a fractional part and can only represent a certain number of the most significant digits of the number, depending on the implementation.

0.2.6 A significant feature of the RL is the "nesting" facility, by which an entity may be defined in parenthesis instead of using a symbolic name.

1 Scope and field of application

This International Standard defines a higher-level symbolic part-programming language which is processed by a digital computer to produce a NC machine program.

The language has been developed primarily for numerically controlled machine tools. It is a reference language, which means that it is not necessarily intended that the whole of the language be implemented. It is expected that parts, or subsets, of the language will be implemented to suit particular circumstances.

2 References

ISO 646, *Information processing — ISO 7-bit coded character set for information processing interchange.*

ISO 841, *Numerical control of machines — Axis and motion nomenclature.*

ISO 3592, *Numerical control of machines — NC processor output — Logical structure (and major words).*

ISO 4343, *Numerical control of machines — NC processor output — Minor elements of 2000-type records (post-processor commands).*

3 Coordinate system

3.1 ISO 841 is the basis for defining the coordinate system of the RL.

3.2 The coordinate system is a right-handed rectangular Cartesian system, related to a workpiece mounted on a machine and aligned with the principal linear slideways of that machine. The positive direction of movement of a component of a machine is that which causes an increasing positive dimension on the workpiece.

3.3 In the RL, the reference axes of the coordinate system are X, Y and Z. These are used in the description of the workpiece and it is assumed that the workpiece is stationary, with the tool or cutter moving relative to the workpiece coordinate system whether or not this is true for the actual NC machine tool operation.

3.4 When specifying angles of planes the positive direction is counterclockwise and the reference axis is as follows :

Plane	Reference axis
XY	X
YZ	Y
ZX	Z

3.5 The positive direction of angle is counterclockwise from the reference axis.

3.6 Angles are expressed in degrees and decimal fractions of a degree.

3.7 The output from the processor (CLDATA) uses the same conventions as the RL, and the output coordinates refer to a reference point on a cutter (usually the centre of the tip) relative to the workpiece coordinate system used in the part program.

4 Language structure

4.1 General comments

4.1.1 General semantics

Digits and letters are used to create unsigned numbers and keywords which in conjunction with characters and special characters may be used to create identifiers, labels and literal character strings. Any valid combination, if existing, may be used to construct a statement and a number of statements arranged in a specific order constitutes a part program.

4.1.2 Sub-contents

For

1) letters, see 4.2;
2) digits, see 4.3;
3) special characters, see 4.4;
4) characters, see 4.5;
5) symbol for literal delimiter, see 4.6;
6) literal character string, see 4.7;
7) unsigned numbers, see 4.8;
8) keywords, see 4.9;
9) simple indentifiers, see 4.10;
10) identifiers, see 4.11;
11) labels, see 4.12;
12) statements, see 4.13;
13) nesting, see 4.14;
14) part program, see 4.15.

4.1.3 Limitations

None.

4.1.4 Syntax

< language structure > : : = { $^{k}_{l}$ [< letters > < digits > < special characters > < characters >
< symbol for literal delimiter > < literal character string > < unsigned numbers >
< keywords > < simple identifiers > < labels > < statements > < nesting >
< part program >] }

4.2 Letters

4.2.1 Semantics

Letters have no individual meaning, being used for forming keywords, simple identifiers, character strings or labels.

4.2.2 Limitations

None.

4.2.3 Syntax

<letter> : : = A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

4.3 Digits

4.3.1 Semantics

Digits have no individual meaning, being used for forming simple identifiers, unsigned numbers, character strings or labels.

4.3.2 Limitations

None.

4.3.3 Syntax

< digit > : : = 0|1|2|3|4|5|6|7|8|9

4.4 Special characters

4.4.1 Semantics

Special characters are used as operators for building up arithmetic expressions and as punctuation marks (or separators) in statements. When special characters are used in literal character strings they are treated as characters with no syntactical significance.

+ – * / ↑	arithmetic operators (see 5.2).
)	closing parenthesis, used as a statement label separator (see 4.11) or in connection with opening parenthesis for nesting.
(	opening parenthesis, used with closing parenthesis in subscripting or nesting, or function arguments.
.	decimal point.
=	equals, used for assigning an entity to a name.
/	slant, used as a separator between a major keyword and the remainder of a statement.
,	comma, used as a separator between elements of a statement.
:	statement label separator (see 4.11).
$	dollar or other currency character used to couple statements and delimit the start of a comment field.
;	semi-colon, used as a separator between statements.
'	apostrophe, used for delimitation of a character string.

The space character has no significance except in character strings.

4.4.2 Limitations

None.

4.4.3 Syntax

< special character > : : = – | . | + | * | / | , | = | (|) | $ | ↑ | ; | : | '

NOTE — $ is given as an example of a national currency character.

4.5 Characters

4.5.1 Semantics

A character is a letter, digit or special character or other valid character.

4.5.2 Limitations

None.

4.5.3 Syntax

< character > : : = < letter > | < digit > | < special character > | < other valid character >

NOTE — Other valid characters have no significance within the language but are nevertheless considered as valid input. These characters are not otherwise defined in this International Standard. They should be manageable by the specific implementation and be selected from the character set defined by ISO 646 and ISO 840.

4.6 Symbol for literal delimiter

4.6.1 Semantics

The apostrophe is used at the beginning and at the end of a literal character string to indicate the extent of the literal string field.

4.6.2 Example

PARTNO/'VALVE HOUSING'

4.6.3 Limitations

None.

4.6.4 Syntax

<symbol for literal delimiter> : : = '

4.7 Literal character string

4.7.1 Semantics

A literal character string may be used in statements, for listing text, or in the postprocessor statements such as INSERT for passing special information through to the postprocessor on the CLDATA. The set of characters permissible is not limited to the set of letters, digits, and special characters defined in this International Standard. Within a literal character string, any special characters are treated simply as characters without syntactical significance.

4.7.2 Example

P1 = POINT/0,0,0 'COMPONENT DATUM'

4.7.3 Limitations

None.

4.7.4 Syntax

< literal character string > : : = < symbol for literal delimiter > $_{0}^{k}$ [< character >] < symbol for literal delimiter >

NOTES

1 The syntax of a literal character string implies that the empty string is allowed.

2 Space characters are significant.

3 A remark (comment) following a single or double currency character need not be a delimited character string.

4 A literal character string not terminated by a closing apostrophe prior to an arbitrary line limit (for example the card column 73 in annex A) is continued on the next line without the need for a currency character.

5 An apostrophe is represented by two apostrophes in a character string delimited by apostrophes.

4.8 Unsigned numbers

4.8.1 Semantics

Numbers have their usual meaning, being built up of decimal digits and may have a decimal point. If no decimal point is included in the number, it is assumed to appear after the rightmost digit.

4.8.2 Examples

47711

58.

3.14

4.8.3 Limitations

There is no defined limit to the number of digits within a number, the limit being implementation dependent. No distinctions are made between integer and real numbers since all are used internally in the real mode. In every case where an integer value is required (for example in a subscript) the fractional part of the real value is truncated.

Number representation within a computer is not necessarily exact. Therefore, approximations are used where necessary to achieve the effect of exact operations. These approximations are computer dependent.

4.8.4 Syntax

< unsigned number > : : = $_1^k$ [< digit >] [.] | $_1^k$ [< digit >] . $_1^k$ [< digit >]

4.9 Keywords

4.9.1 Semantics

Keywords have a fixed meaning within the language. They may be regarded as entries in a vocabulary list.

The set of keywords is not fixed but may be enlarged in future revisions of this International Standard.

Keywords do not have the role of delimiters, in contrast to some scientific programming languages for example, as two adjacent keywords have to be separated from each other by a special character.

Keywords are divided into two classes, the major keywords which define the type of the statement, and the minor keywords which give additional information. These classes can be divided into two subclasses each : the processor and the postprocessor keywords. Postprocessor keywords are listed and defined in ISO 4343. An alphabetical list of all keywords is given in annex D.

4.9.2 Examples

POINT

INTOF

4.9.3 Limitations

Keywords consist only of letters and have at least two letters.

4.9.4 Syntax

< keyword > : : = $_{2}^{6}$ [< letter >]

4.10 Simple identifiers

4.10.1 Semantics

Simple identifiers have no inherent meaning and are used only as specificators (see 8.1.4), labels or arithmetic symbols.

4.10.2 Examples

P2

L1

BOBBY1

4.10.3 Limitations

A simple identifier shall not be redefined within a part program except when used as an arithmetic symbol. It is recommended that processors should not accept keywords as simple identifiers. A simple identifier, unless used as a synonym, of more than one letter should contain a digit.

4.10.4 Syntax

$$\langle \text{simple identifier} \rangle ::= \langle \text{letter} \rangle \, {}_{0}^{5}[\langle \text{letter} \rangle \mid \langle \text{digit} \rangle]$$

4.11 Identifiers

4.11.1 Semantics

An identifier is similar to a simple identifier but normally carries a subscript.

4.11.2 Examples

P2(1)

P2(5)

4.11.3 Limitations

An identifier shall not be redefined within a part program except when used as an arithmetic symbol.

4.11.4 Syntax

< identifier > : : = < simple identifier > $_0^1$ [(< scalar >)]

4.12 Labels

4.12.1 Semantics

A part programmer may attach a label to a statement for statement identification. The label may be referred to in a conditional or unconditional transfer statement or a geometric transfer statement.

4.12.2 Example

A1) GOTO/P1

or alternatively A1 : GOTO/P1

where A1 is the label.

4.12.3 Limitations

None.

4.12.4 Syntax

< label > : : = $_{1}^{6}$ [< digit >] | < simple identifier >

4.12.5 Cross-reference (see annex D)

IF

JUMPTO

TRANTO

4.13 Statements

4.13.1 Semantics

A statement is a complete instruction or unit of information comparable with a sentence in a natural language. A statement consists of a single keyword or a syntactically defined sequence of elements separated by special characters. These elements are keywords, simple identifiers, unsigned numbers and character strings.

There are several types of statement, grouped for convenience of description into the following five main clauses of this International Standard : clause 5 arithmetic, clause 6 program definition, clause 7 program execution, clause 8 geometrical definition and clause 9 geometric execution.

Statements may be labelled or unlabelled. Wherever a keyword is allowed within a statement, it may be replaced by a previously defined synonym.

Everywhere within a macro where a keyword, a simple identifier, or unsigned number, is allowed, it may be replaced by a declared macro parameter name.

4.13.2 Limitations

None.

4.13.3 Syntax

< statement > : : = $^{1}_{0}$ [< label > [) | :]] < unlabelled statement >

< unlabelled statement > : : = < arithmetic statement > | < program definition statement > |
< program control statement > | < geometrical definition statement > |
< geometrical execution statement >

4.14 Nesting

4.14.1 Semantics

Any unlabelled arithmetic statement or unlabelled geometrical definition statement may be nested.

The statement is written between the opening and closing parentheses. Wherever a simple identifier may appear, it is possible to nest the full statement with a simple identifier which may be referenced later in the part program, or to nest the statement without a simple identifier or equals character. The three alternatives

a) simple identifier,

b) nested statement without simple identifier,

c) nested statement with simple identifier,

are included in the term "specificator". For example, where an entity of the type "circle" is valid, the term "circle specificator" is used.

4.14.2 Example

The statements

PT4 = POINT/3,6

PT6 = POINT/8,9

L2 = LINE/PT4, PT6

may be written as

L2 = LINE/(PT4 = POINT/3,6), (POINT/8,9)

4.14.3 Limitations

None.

4.15 Part program

4.15.1 Semantics

A part program is a logically complete and ordered sequence of statements and the result of processing a part program is to produce a correspondingly complete and ordered machine control program.

A part program consists of a part program identification statement, a sequence of statements and a part program termination statement.

4.15.2 Limitations

None.

4.15.3 Syntax

< part program > : : = < part program identification statement > $_{0}^{k}$ [< statement >]
< part program termination statement >

5 Arithmetic statements

5.1 General comments

5.1.1 General semantics

The arithmetic features of the language allow a wide range of algebraic expressions to be evaluated within a part program. In general, the operators used in part program arithmetic are those used in conventional algebra, the exception being the algebraic multiplication × which is replaced by *.

Conventional algebra allows a description to occupy more than one line. Since this is not conveniently converted into suitable computer input, this is not allowed in the part-program language.

For example :

$$A = \frac{1 + 2 + 3}{4}$$

allowable in algebra has to be converted to :

$A = (1 + 2 + 3)/4$

before use in the part-program language. For the same reason it is found necessary to introduce an exponential operator (** or ↑) into the part-program language so that

$A = 2^3$

is written as

A = 2**3 or A = 2 ↑ 3

Throughout, normal mathematical limitations apply, for example in using SQRT (A), A shall have a positive value or in using ASIN (B), B shall have a value lying between ± 1.

The character = is used in arithmetic statements, but may not have the meaning "equals" which it has in algebra. In part program arithmetic this character should be read as "is replaced by" or "is given the value of". It is therefore permissible to write $A = A + 1$ which means that one is to be added to the value of A, and A now given the value of the result.

Normal algebraic operation priorities apply. For example, in the statement A = 2 + 3*4, the * operation will be performed before the + operation. Brackets can be used in arithmetic statements and as in algebra may alter the sequence in which operations are carried out. If, for example, the above statement is written as A = (2 + 3)*4 the + operation is performed before the * operation.

The implied multiplication feature of algebra is not allowed. For example $A = 5(B + 2)$ which is permissible in algebra shall be written as A = 5*(B + 2).

The hierarchy of operational priorities is as follows :

Priority 1 ( )
2 **,↑
3 *,/
4 +,−

In the examples given later in this clause, the numeric values shown are for illustration only, and may be approximations to the real values.

5.1.2 Sub-contents

For

1) arithmetic operators, see 5.2;
2) arithmetic functions, see 5.3;
3) the algebraic functions, see 5.3.2;
4) the trigonometrical functions, see 5.3.3;
5) the exponential functions, see 5.3.4;
6) the vector functions, see 5.3.5;
7) the miscellaneous functions, see 5.3.6.

5.1.3 Limitations

None.

5.1.4 Syntax of arithmetic

< arithmetic symbol > : : = < identifier >

< adding operator > : : = + | –

< multiplying operator > : : = * | /

< exponential operator > : : = ** | ↑

< primary > : : = < unsigned number > | < arithmetic symbol > |
< scalar function > | (< arithmetic expression >)

< factor > : : = < primary > | < primary > < exponential operator > < primary >

< term > : : = < factor > | < term > < multiplying operator > < factor >

< arithmetic expression > : : = < term > | < adding operator > < term > |
< arithmetic expression > < adding operator > < term >

< scalar > : : = < primary > | < adding operator > < primary > |
(< arithmetic symbol > = < scalar >)

< arithmetic statement > : : = < arithmetic symbol > = < scalar >

< algebraic function > : : = [ABS | SQRT] (< scalar >)

< trigonometric function > : : = [SIN | COS | TAN | ASIN | ACOS | ATAN] (< scalar >) |
ANGL (< circle spec > , < point spec >) |
DIST (< point spec > , < point spec >) |

< exponential function > : : = [EXP | LOG] (< scalar >)

< vector function > : : = LNTH (< vector spec >) |
DOT (< vector spec > , < vector spec >)

< miscellaneous functions > : : = ANGL (< circle spec > , < point spec >) |
DIST (< point spec > , < point spec >) |
NUM (< pattern spec >)

< scalar function > : : = < trigonometric function > |
< algebraic function > |
< exponential function > |
< vector function > |
< miscellaneous function >

5.2 Arithmetic operators

5.2.1 Semantics

The adding operators + and − shall be used in either of the following ways :

a) As a unary operator, in which case the operator denotes the sign which has to be given to the following term :

+ denotes that the following term is to be positive;

− denotes that the following term is to be negative.

b) As a binary operator, in which case the operator, preceded by an arithmetic expression and followed by a term, denotes how the term is to be added to the arithmetic expression :

+ denotes that the term is to be added to the arithmetic expression;

− denotes that the term is to be subtracted from the arithmetic expression.

The multiplying operators * and / denote how the preceding term is to be multiplied by the following factor :

* the term is to be multiplied by the factor;

/ the term is to be divided by the factor.

The exponential operator ** or ↑ denotes that the preceding primary is to be raised to a power given by the following primary.

5.2.2 Examples

$A = B + C$	A contains the sum of B and C
$A = B * C$	A contains the result of B multiplied by C
$A = B ** C$	A contains the result of B to the power C

5.2.3 Limitations

None.

5.2.4 Syntax

< adding operator > : : = + | −

< multiplying operator > : : = * | /

< exponential operator > : : = ** | ↑

5.3 Arithmetic functions

5.3.1 General comments

NOTE — Throughout this clause the preferred form of function names has been used, i.e. SIN, ABS, etc. However, it is recognized that the older "F" forms, i.e. SINF, ABSF, etc., may, in some instances, remain in use for some time. The comments and observations made on the preferred forms apply equally well to the "F" versions.

5.3.1.1 Sub-contents

For

1) algebraic functions, see 5.3.2;
2) trigonometric functions, see 5.3.3;
3) exponential functions, see 5.3.4;
4) vector functions, see 5.3.5;
5) miscellaneous functions, see 5.3.6.

5.3.1.2 Syntax

```
< algebraic functions > : : = [ABS/SQRT] ( < scalar > )

< trigonometric functions > : : = [SIN/COS/TAN/ASIN/ACOS/ATAN] ( < scalar > )

< exponential functions > : : = [EXP/LOG] ( < scalar > )

< vector functions > : : = LNTH ( < vector spec > ) |
                           DOT ( < vector spec > , < vector spec > )

< miscellaneous functions > : : = ANGL ( < circle spec > , < point spec > ) |
                                  DIST ( < point spec > , < point spec > ) |
                                  NUM ( < pattern spec > )
```

5.3.2 The algebraic functions; ABS and SQRT

5.3.2.1 Semantics

The algebraic function designator denotes the type of arithmetic process to be performed using the following bracketed arithmetic expression.

ABS : the absolute value of the arithmetic expression is to be taken.

SQRT : the square root of the arithmetic expression is to be taken.

5.3.2.2 Examples

A = ABS (– 2); A contains the absolute value of (– 2), i.e. 2

A = SQRT (9); A contains the square root of (9), i.e. 3

5.3.2.3 Limitations

None.

5.3.2.4 Syntax

< algebraic function > : : = [ABS | SQRT] (< scalar >)

5.3.3 Trigonometric functions; SIN, COS, TAN, ASIN, ACOS, ATAN

5.3.3.1 Semantics

The trigonometric function designator denotes the type of trigonometric process to be performed on the following bracketed arithmetic expression :

SIN : the sine of the following arithmetic expression is to be taken.

COS : the cosine of the following arithmetic expression is to be taken.

TAN : the tangent of the following arithmetic expression is to be taken.

ASIN : the angle whose sine is given by the following arithmetic expression is to be taken.

ACOS : the angle whose cosine is given by the following arithmetic expression is to be taken.

ATAN : the angle whose tangent is given by the following arithmetic expression is to be taken.

In the case of the direct functions, the value of the arithmetic expression will be taken to represent degrees.

In the case of the inverse functions, the result given will be in degrees in the following ranges :

ASIN $-90 \leqslant A \leqslant 90$

ACOS $0 \leqslant A \leqslant 180$

ATAN $-90 \leqslant A \leqslant 90$

5.3.3.2 Examples

A = SIN(30)	A is given the value 0,5
A = COS(30)	A is given the value 0,866
A = TAN(45)	A is given the value 1,000
A = ASIN(0.5)	A is given the value 30
A = ASIN(– 0.5)	A is given the value – 30
A = ACOS(0.866)	A is given the value 30
A = ACOS(– 0.866)	A is given the value 150
A = ATAN(1)	A is given the value 45
A = ATAN(– 1)	A is given the value – 45

5.3.3.3 Limitations

None.

5.3.3.4 Syntax

< trigonometric function > : : = [SIN | COS | TAN | ASIN | ACOS | ATAN] (< scalar >)

5.3.4 The exponential functions; EXP and LOG

5.3.4.1 Semantics

The exponential function designator denotes the type of exponential process to be performed on the following bracketed arithmetic expression.

EXP : the value of e to a power given by the value of the following arithmetic expression is taken.

LOG : the natural logarithm of the following arithmetic expression is taken. In this case the expression must be positive.

5.3.4.2 Examples

A = EXP(2); *A* contains the value of e^2 i.e. 7,389 1

A = LOG(2); *A* contains the value of $\log_e 2$ i.e. 0,693 1

5.3.4.3 Limitations

None.

5.3.4.4 Syntax

< exponential function > : : = [EXP | LOG] (< scalar >)

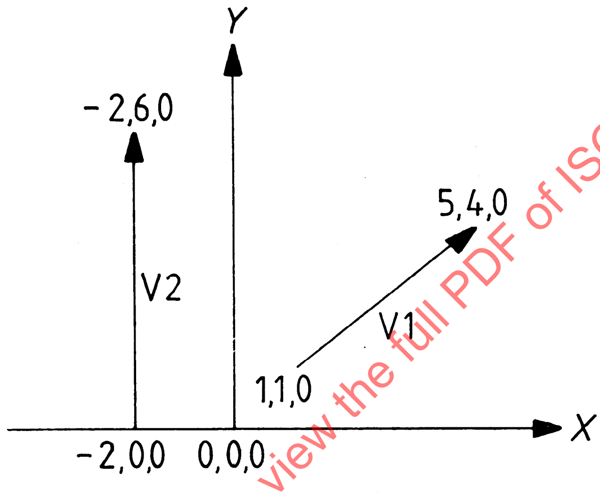
5.3.5 The vector functions; LNTH and DOT

5.3.5.1 Semantics

The LNTH vector function designator denotes that the length of the vector represented by the following bracketed vector specificator is to be calculated.

The DOT vector function designator denotes that the dot product of the vectors represented by the following bracketed pair of vector specificators is to be obtained. In algebraic terms, the result is (length of first vector) × (length of second vector) × COS (angle between the vectors), or diagrammatically it is the product of the length of one vector and the length of projection of the other vector on it.

5.3.5.2 Example



V1 = VECTOR /1, 1, 0, 5, 4, 0

V2 = VECTOR/ – 2, 0, 0, – 2, 6, 0

*A*1 = LNTH (V1) *A*1 contains the value 5

*A*2 = LNTH (V2) *A*2 contains the value 6

*A*3 = DOT (V1, V2) *A*3 contains the value 18, i.e. 6 × 5 × 3/5

Figure 1

5.3.5.3 Limitations

None.

5.3.5.4 Syntax

< vector function > : : = LNTH (< vector spec >) |
DOT (< vector spec > , < vector spec >)

5.3.6 The miscellaneous functions; ANGL, DIST, NUM

5.3.6.1 Semantics

The miscellaneous function designator denotes the type of process to be performed on the following bracketed miscellaneous function :

ANGL : the angle between the positive X-axis and the line joining the centre of the specified circle with the specified point.

DIST : the distance between the specified points.

NUM : the number of points in the specified pattern.

5.3.6.2 Limitations

None.

5.3.6.3 Syntax

```
< miscellaneous function > : : = ANGL ( < circle spec > , < point spec > ) |
                                 DIST ( < point spec > , < point spec > ) |
                                 NUM ( < pattern spec > )
```

6 Program definition statements

6.1 General comments

6.1.1 General semantics

Part program statements are normally processed in a sequential manner but it is sometimes desirable to repeat a sequence of statements. Program definition statements provide this facility by allowing symbols to be redefined, synonyms to be defined for keywords and altering the normal sequential processing of the part program statements.

6.1.2 Sub-contents

For

a) synonym statements, see 6.2;

b) reservation for subscripting, see 6.3;

c) definition of a macro, see 6.4.

6.1.3 Limitations

None.

6.1.4 Syntax

< program definition statement > : : = { < synonym declaration > < reserve declaration > < macro block > }

6.2 The synonym statement; SYN/

6.2.1 Semantics

The synonym statement is used in a part program where for some reason it is desirable to use an alternative form of spelling for a keyword.

The general form of the synonym-keyword pairs is

SYN/list of synonym-keyword pairs

The synonym given in a SYN statement will, where it appears subsequently in a part program, be taken to represent the keyword associated with it in the SYN/ statement and will be processed as such. Once a symbol has been defined as a synonym it cannot be redefined in any other context. A list of preferred synonyms is given in annex C.

The assignment of a synonym to a keyword does not preclude the subsequent use of a keyword as desired.

6.2.2 Examples

SYN/P,POINT,C,CIRCLE,TT,TANTO,L,LINE

this allows the statement

L1 = L/RIGHT,TT,C1,RIGHT,TT,C2

which is taken as

L1 = LINE/RIGHT,TANTO,C1,RIGHT,TANTO,C2

The following type of statement is not allowed because P was defined as synonym

P = POINT/0,1,2

6.2.3 Limitations

None.

6.2.4 Syntax

<synonym parameter list> : : = <simple identifier> , <keyword> |
<synonym parameter list> , <simple identifier> , <keyword>

<synonym declaration> : : = SYN/ <synonym parameter list>

6.3 Reservation for subscripting; RESERV/

6.3.1 Semantics

The RESERV statement is used to indicate that, within a part program, identifiers (by definition subscripted) may appear.

The general form of the statement is RESERV/ followed by a string of couplets the first part of each being the identifier to be subscripted and the second being the largest value of the subscript which may be given to that identifier.

The RESERV statement containing a particular identifier shall precede any use of an identifier.

6.3.2 Example

RESERV/P1,10,P2,7,PAT1,12

this allows the statement

P1(8) = POINT/INTOF,L1,L2

but would not allow

P2(8) = POINT/INTOF,L1,L2

6.3.3 Limitations

In certain implementations, the types of identifier which can be subscripted may be restricted.

An identifier which is in RESERV shall normally not be used without a subscript.

6.3.4 Syntax

<reserve parameter list> : : = <simple identifier> , <scalar> |
<reserve parameter list> , <simple identifier> , <scalar>

<reserve declaration> : : = RESERV/ <reserve parameter list>

6.4 Definition of a macro; MACRO/

6.4.1 Semantics

A MACRO definition is used to specify a group of part program statements that can subsequently be executed in a part program. The definition allows an identifier to be attached to the group for reference by subsequent CALLs.

A statement of the form

a) MAC1 = MACRO

or b) MAC2 = MACRO/A,B,C

or c) MAC3 = MACRO/A = 1.2, B = P1, C = 5

or d) MAC4 = MACRO/A = 1.2, B, C = 5

denotes the start of a MACRO sequence of part program statements. The MACRO statement may contain a parameter list [as in b), c) or d)] any item of which may be given a normal value [as in c)]. Parameters may be used anywhere within the macro and are only allocated values when the macro is called. Normal values are used only in the absence of an actual value in the call.

The end of the macro sequence of part program statements is denoted by the statement

TERMAC

6.4.2 Examples

```
MAC1 = MACRO
GOTO/P1
GODLTA/0.5
GODLTA/ – .5
TERMAC

MAC2 = MACRO/A, B
GOTO/A
GODLTA/B
GODLTA/ – B
TERMAC
```

6.4.3 Limitations

The following part program statements may not appear within a MACRO definition :

a) PARTNO
b) FINI
c) LOOPST
d) LOOPND
e) MACRO

In addition implementation restrictions may bar the use of SYN/ and RESERV/ within a macro.

6.4.4 Syntax

< normal macro item > : : = < keyword > | < simple identifier > | < unsigned number > | (< arithmetic expression >)

< macro item list > : : = < simple identifier > | < simple identifier > = < normal macro item > |
< macro item list > , < simple identifier > | < macro item list > ,
< simple identifier > = < normal macro item >

< macro assignment statement > : : = < simple identifier > = MACRO [/ < macro item list >]

< macro termination statement > : : = TERMAC

< macro block > : : = < macro assignment statement > $_0^k$ [< statement >] < macro termination statement >

6.4.5 Cross-reference (see annex D)

PARTNO

FINI

LOOPST

LOOPND

MACRO

CALL

7 Program execution statements

7.1 Part program control

7.1.1 General semantics

With the exception of "part program identification statement" and "part program termination statement" which have special places in a part program, the other "program execution statements" are used where control over things like input, output or program flow are desirable. These statements are grouped together as "program control statements".

7.1.2 Sub-contents

For

1) part program identification and termination, see 7.2;
2) part program identification statement, see 7.2.1;
3) part program termination statement, see 7.2.2;
4) machine and no-postprocessing statements, see 7.3;
5) machine statement, see 7.3.1;
6) no-postprocessing statement, see 7.3.2;
7) input/output statements, see 7.4;
8) print statement, see 7.4.1;
9) punch statement, see 7.4.2;
10) read statement, see 7.4.3;
11) cutter location print statement, see 7.4.4;
12) titles statement, see 7.4.5;
13) loop start and end statements and transfer statements, see 7.5;
14) loop start and end statement, see 7.5.1;
15) unconditional transfer statement, see 7.5.2;
16) conditional transfer statement, see 7.5.3;
17) geometric transfer statement, see 7.5.4;
18) copy statement and index specification, see 7.6;
19) copy statement, see 7.6.1;
20) index specification, see 7.6.2;
21) macro execution statement, see 7.7;
22) remark statement, see 7.8;
23) postprocessing print statement, see 7.9.

7.1.3 Limitations

None.

7.1.4 Syntax

< program execution statement > : : = { $_0^k$ [< part program identification statement > |
< part program termination statement > | < machine statement > |
< no-postprocessing statement > | < print statement > |
< punch statement > | < read statement > | < cutter location print statement > |
< titles statement > | < loop start and end statement > |
< unconditional transfer statement > | < conditional transfer statement > |
< geometric transfer statement > | < copy statement > | < index specification > |
< macro execution statement > | < remark statement > |
< postprocessing print statement >] }

7.2 Part program identification and termination

7.2.1 Part program identification statement

PARTNO/literal character string

7.2.1.1 Semantics

The PARTNO statement is used for part program identification.

7.2.1.2 Example

PARTNO/'VALVE HOUSING, DRAWING NO 72445'

7.2.1.3 Limitations

None.

7.2.1.4 Syntax

< part program identification statement > : : = PARTNO/ < literal character string > $_{0}^{u}$ [, < literal character string >]

7.2.2 Part program termination statement

FINI

7.2.2.1 Semantics

The FINI statement defines the physical end of a part program.

7.2.2.2 Limitations

None.

7.2.2.3 Syntax

< part program termination statement > : : = FINI

7.3 Machine and no-postprocessing statements

7.3.1 Machine statement

MACHIN/machine parameter list

7.3.1.1 Semantics

MACHIN statements allow specification of one or more postprocessors to be called. Each postprocessor is defined by a name and optionally an identifying number. Additional information to initialize postprocessor parameters may be given.

7.3.1.2 Context dependency

Postprocessor execution is inhibited if a NOPOST statement is present anywhere in the part program.

7.3.1.3 Example

MACHIN/EXPOST, 0

7.3.1.4 Limitations

None.

7.3.1.5 Syntax

< machine statement > : : = MACHIN/ < simple identifier > $^{1}_{0}$ [, < postprocessor parameter list >]

7.3.1.6 Cross-reference (see annex D)

NOPOST

NOTE — The postprocessor parameter list is dependent on a particular postprocessor so that its formal syntax description is not contained within the reference language document.

7.3.2 No-postprocessing statement

NOPOST

7.3.2.1 Semantics

Postprocessing is inhibited even if one or more MACHIN statements are present.

7.3.2.2 Example

NOPOST

7.3.2.3 Limitations

None.

7.3.2.4 Syntax

< no postprocessing statement > : : = NOPOST

7.3.2.5 Cross-reference (see annex D)

MACHIN

7.4 Input/output statements

7.4.1 Print statement

PRINT/ON

PRINT/OFF

PRINT/0

PRINT/PAGE

PRINT/2, S_1, . . ., S_n

PRINT/NONAME, S_1, . . ., S_n

PRINT/3, S_1, . . ., S_n

PRINT/3, ALL

PRINT/S_1, . . ., S_n

7.4.1.1 Semantics

A PRINT statement causes internally stored information to be printed. PRINT/ON : this feature permits the programmer to obtain a printout of the canonical form immediately following the definition statement. The statement has this effect on all subsequent definitions, until countermanded by a PRINT/OFF statement.

A PRINT/0 or PRINT/PAGE statement causes the page to be ejected.

A PRINT/2, . . . or PRINT/NONAME, . . . statement causes printing of canonical form or scalar value of the quantities specified, without their identification.

A PRINT/. . . or PRINT/3, . . . statement causes printing of the canonical form or scalar value of the quantities specified, with their identification.

PRINT/ALL or PRINT/3, ALL; this feature has the same effect as the PRINT/. . . statement, except that it affects all available quantities.

The symbolic print type specification is preferred usage.

7.4.1.2 Example

PRINT/L1,C2,ANGL4,PT2

7.4.1.3 Limitations

None.

7.4.1.4 Syntax

< print statement > : : = PRINT/ < print parameter list >

< print parameter list > : : = [ON | OFF] | [0 | PAGE] | $^{1}_{0}$ [NONAME | 2 | 3], < identifier > $^{u}_{0}$ [, < identifier >] | [3,] ALL

7.4.2 Punch statement

PUNCH/n, $S_1, \ldots, S_n$

PUNCH/n, ALL

PUNCH/ON

PUNCH/OFF

PUNCH/$S_1, \ldots, S_n$

PUNCH/ALL

7.4.2.1 Semantics

A PUNCH statement causes internally stored information to be punched. n is implementation dependent and controls the form of the output. ALL specifies that all of form n is to be punched. $S_1, \ldots, S_n$ are the quantities to be punched.

ON causes punching to be started.

OFF stops all punching.

ALL causes all information to be punched.

7.4.2.2 Example

PUNCH/A,B,C

7.4.2.3 Limitations

None.

7.4.2.4 Syntax

< punch statement > : : = PUNCH/ < punch parameter list >

< punch parameter list > : : = [ON | OFF] $|_0^1$ [< digit >], < identifier > $_0^u$ [, < identifier >] $|_0^1$ [< digit >], ALL

7.4.2.5 Cross-reference (see annex D)

READ

7.4.3 Read statement

READ/1

READ/ALL

READ/1, S_1, . . ., S_n

7.4.3.1 Semantics

READ/1 instructs the computer program to read cards in the same code and format as produced by the PUNCH statement.

READ/1, S_1, S_2, S_3, . . ., S_n instructs the computer program to read the cards and to assign a new symbol to each canonical form or scalar value on the cards.

S_1, S_2, S_3, etc., are the new symbols.

7.4.3.2 Context dependency

The symbols S_1, S_2, S_3, etc., are the symbolic names assigned to the canonical forms or scalar values read from the cards. The cards are in the same code and format as produced by the PUNCH statement.

7.4.3.3 Limitations

With the READ/ALL or READ/1 without new symbols there shall be an implementation dependent read termination.

7.4.3.4 Syntax

< read statement > : : = READ/ < read parameter list >

< read parameter list > : : = [1 | ALL] | 1, < identifier > $_0^u$ [, < identifier >]

7.4.3.5 Cross-reference (see annex D)

PUNCH

7.4.4 Cutter location print statement

CLPRNT

CLPRNT/ON

CLPRNT/OFF

CLPRNT/OMIT, *n*

7.4.4.1 Semantics

CLPRNT calls for a print of all cutter location data.

CLPRNT/ON starts printing of cutter location data.

CLPRNT/OFF stops printing of cutter location data.

CLPRNT/OMIT, n prints first point then points $n + 1$, $2n + 1$, $3n + 1$, . . . and last point.

7.4.4.2 Context dependency

The CLPRNT statement operates so that the information produced during processing in the computer is printed.

7.4.4.3 Limitations

None.

7.4.4.4 Syntax

< cutter location print statement > : : = CLPRNT [/ < clprint parameter list >]

< clprint parameter list > : : = [ON | OFF | OMIT, $_{1}^{u}$ [< digit >]]

7.4.5 Titles statement

TITLES/literal character string

7.4.5.1 Semantics

The TITLES statement permits labelling of the printed output. The literal character string following TITLES in the statement is printed immediately at the time the statement is executed. The string starts at the left printing margin and extends across the paper up to the length of the string. The literal character string may be continued beyond a single input line. Successive strings are concatenated into a single string for printing.

7.4.5.2 Example

PRINT/PAGE

TITLES/'SCALARS S_1, S_2, S_3, S_4 ARE THE', $

'CALCULATED SPINDLE SPEEDS FOR STEEL'

PRINT/S_1, S_2, S_3, S_4

7.4.5.3 Limitations

If the literal character string is greater than the print width of the paper, the folding back of the necessary continuation line(s) is implementor dependent.

7.4.5.4 Syntax

< titles statement > : : = TITLES/ < literal character string > $^{u}_{0}$[, < literal character string >]

7.4.5.5 **Cross-reference** (see annex D)

PRINT

7.5 Loop start and end statements and transfer statements

7.5.1 Loop start and end statements

LOOPST

LOOPND

7.5.1.1 Semantics

LOOPST and LOOPND define the bounds (extent) of the group of statements which is a loop. The statements between LOOPST and LOOPND comprise the loop. LOOPST and LOOPND are implementation dependent.

Within a loop the normal sequential execution of the part program may be altered by transfer statements (IF, JUMPTO). Branching to LOOPND is permitted but not to LOOPST.

7.5.1.2 Context dependency

The MACRO statement implies LOOPST, and TERMAC implies LOOPND, i.e. a MACRO as a whole has by definition all properties of a loop.

7.5.1.3 Examples

```
        J = 1
        LOOPST
A)      . . .
        . . .
        IF (J - 100) B,C,C
B)      J = J + 1
        JUMPTO/A
C)      LOOPND
```

7.5.1.4 Limitations

LOOPST and LOOPND shall always be used as a pair.

7.5.1.5 Syntax

< start of a loop > : : = LOOPST

< end of a loop > : : = LOOPND

7.5.1.6 Cross-reference (see annex D)

IF
JUMPTO
MACRO
TERMAC

7.5.2 Unconditional transfer statement

JUMPTO/label

7.5.2.1 Semantics

Normal sequential execution of the part program is interrupted and control continues at the statement with the label referenced by the JUMPTO statement.

7.5.2.2 Examples

```
      LOOPST
      . . .
A)    . . .
      . . .
      JUMPTO/A
      . . .
      LOOPND
      LOOPST
      . . .
A)    . . .
      . . .
      IF (VALUE – 2) A,B,C
      . . .
C)    . . .
      . . .
B)    LOOPND
```

7.5.2.3 Limitations

If LOOPST and LOOPND are used, then an unconditional transfer command and the statement associated with the label referred to by this command shall occur within the same loop. In a macro, an unconditional transfer shall be to a statement within that macro.

7.5.2.4 Syntax

< unconditional transfer statement > : : = JUMPTO/ < label >

7.5.2.5 Cross-reference (see annex B)

Label

7.5.3 Conditional transfer statement

IF (n) label1, label2, label3

7.5.3.1 Semantics

Normal sequential execution of a part program may be altered by discrimination on the value of an arithmetic expression or single scalar quantity (n) relative to zero; control is transferred to the statement with the first label if the value is less than zero, to the second label if it is zero and to the third label if it is greater than zero. Branching forward and backward is permitted.

7.5.3.2 Limitations

See 7.5.2.

7.5.3.3 Syntax

< conditional transfer statement > : : = IF (< arithmetic expression >) < label > , < label > , < label >

7.5.3.4 Cross-reference

Label Arithmetic expression	(see annex B)
LOOPST LOOPND MACRO TERMAC	(see annex D)

7.5.4 Geometric transfer statement

TRANTO/label

7.5.4.1 Semantics

Control is transferred to the statement with the specified label.

7.5.4.2 Context dependency

The TRANTO statement is intended for use in conjunction with multiple check surface motion statements.

7.5.4.3 Examples

```
    TRANTO/A
    . . . . . .
    . . . . . .
A)  . . . . . .
```

7.5.4.4 Limitations

Only motion and post processor statements may be used as TRANTO aims.

7.5.4.5 Syntax

< geometric transfer statement > : : = TRANTO/ < label >

7.5.4.6 Cross-reference

Label (see annex B)

Multiple check surface (see 9.3)

7.6 COPY statement and index specification

7.6.1 COPY statement

COPY/i,SAME,*n*

COPY/i,TRANSL,*x*,*y*,*z*,*n*
XYROT

COPY/i,YZROT,angle,*n*
ZXROT

COPY/i,MODIFY,smat,*n*

7.6.1.1 Semantics

The COPY statement generates *n* copies of the copy block defined by the (INDEX/i — INDEX/i, NOMORE) or the (INDEX/i — COPY/i, . . .) pair.

The cutter location points are transformed according to the modifier list :

SAME	(no transformation)
TRANSL,*x*,*y*,*z*	(translation of axes)
XYROT,angle	(rotation in *XY*-plane)
YZROT,angle	(rotation in *YZ*-plane)
ZXROT,angle	(rotation in *ZX*-plane)
MODIFY, symbol of a matrix	(general transformation).

7.6.1.2 Context dependency

A copy block may be imbedded within the range of a TRACUT or vice versa. In either case the TRACUT statements transform the original cutter location points only; the TRACUT feature is not cumulative.

7.6.1.3 Example

```
. . . .
INDEX/7
. . . .
GO TO/x,y,z
. . . .
COPY/7, MODIFY,MATR,2
. . . .
COPY/7, TRANSL,x1,y1,z1,1
. . . .
J = 5
INDEX/J
. . . .
INDEX/J,NOMORE
. . . .
COPY/J,XYROT,90,3
. . . .
COPY/7,SAME,2
. . . .
```

7.6.1.4 Limitations

COPY loops shall not be permitted to overlap but they can be nested.

7.6.1.5 Syntax

< copy statement > : : = COPY/ < copy parameter list >

< copy parameter list > : : = < label > ,[SAME | TRANSL, < scalar > , < scalar > , < scalar > |
[XYROT | YZROT | ZXROT], < scalar > | MODIFY, < matrix spec >], < scalar >

7.6.1.6 Cross-reference (see annex D)

INDEX
TRACUT
MATRIX

7.6.2 Index specification

INDEX/i

INDEX/i, NOMORE

7.6.2.1 Semantics

INDEX/i and INDEX/i, NOMORE define the bounds (extent) of a group of statements which can be referenced by a COPY statement. This group is named a copy loop.

7.6.2.2 Context dependency

INDEX/i, NOMORE can be replaced by one or more COPY/i, . . . statements. In this case each COPY statement defines a copy loop with a common starting point.

7.6.2.3 Example

See 7.6.1.

7.6.2.4 Limitations

Copy loops shall not be permitted to overlap, but they can be nested. The same index value cannot be used in more than one pair of INDEX specificators or one set of INDEX-COPY statements.

7.6.2.5 Syntax

< index specification > : : = INDEX/ < label > $^{1}_{0}$ [, NOMORE]

7.6.2.6 Cross-reference (see annex D)

COPY

7.7 Macro execution statement

CALL/name

CALL/name, macro parameter list

7.7.1 Semantics

The CALL statement executes a MACRO defined in the part program. A MACRO with the assigned symbolic name is referenced and the effects are as if the macro CALL is replaced by the MACRO sequence (statements between MACRO and TERMAC). The formal parameters within the MACRO sequence are replaced by the normal parameters specified in the MACRO statement except where superseded by actual parameters from the list of the MACRO CALL statement. The statements in the MACRO sequence are executed.

The CALL statement may be used to call a MACRO defined outside the part program.

The CALL statement may be used to call a special program. In this case the list format is defined by the requirements of the special program.

7.7.2 Context dependency

The actual parameters of a CALL apply only during the resulting single execution and in no way alter the stored MACRO.

A CALL can be included anywhere in the part program even in a loop (between LOOPST and LOOPND) and in a MACRO sequence. The latter has the effect of nested MACROs.

7.7.3 Example

```
. . .
. . .
MACNAM = MACRO/A, B = 10, PL, H, P
. . .
. . .
CALL/BSURF, . . . (list of paramaters)
. . .
. . .
TERMAC
. . .
. . .
PL5 = PLANE/ . . .
H5 = arithmetic expression
. . .
. . .
CALL/MACNAM, A =5, PL = PL5, H = H5, P = POINT
```

7.7.4 Limitations

In a CALL statement within a MACRO definition, the formal parameter name may not be the same as any parameter names in the MACRO being defined.

7.7.5 Syntax

< macro execution statement > : : = CALL/ < simple identifier > $_0^1$ [, < formal macro assignment list >]

< formal macro assignment list > : : = < formal macro item > $_0^n$ [, < formal macro item >]

< formal macro item > : : = < simple identifier > = < normal macro item >

7.8 Remark statement

REMARK/literal character string

7.8.1 Semantics

This statement is printed in the verification listing of the part program, and the processor takes no other action.

7.8.2 Example

REMARK/'GEOMETRICAL DEFINITIONS FOLLOW'

7.8.3 Syntax

< remark statement > : : = REMARK/ < literal character string >

7.8.4 Limitations

Where the literal character string is continued on a second line, the line images are printed as input and not concatenated on a print line.

7.9 Postprocessing print statement

PPRINT/literal character string

7.9.1 Semantics

This statement is transferred via the CLDATA to the postprocessor to enable a comment to be added to the listing produced by the postprocessor.

7.9.2 Example

PPRINT/'TOOL CHANGE'

7.9.3 Limitations

None.

7.9.4 Syntax

< postprocessor print statement > : : = PPRINT/ < literal character string > $_0^u$ [, < literal character string >]

8 Geometrical definition statements

8.1 General comments

8.1.1 Limitations

Only in special cases are limitations given for the definition and use of geometry. A syntactically valid statement may represent a geometrically nonsensical situation, and it is not possible to cover all such situations or the treatment of them by processors.

A simple example of this concerns the scalar in circle definition statements giving the radius of the circle; if the scalar is a nested arithmetic expression it may have a negative value (geometrically meaningless) and this may be rejected by some processors while others take the absolute value.

8.1.2 Coordinate system

x, y and z in geometric definition statements refer to distances in the basic coordinate system (see clause 3), but local or secondary reference systems may also be used (see 8.2.1).

8.1.3 Frequently used modifiers

8.1.3.1 For the unambiguous definition of a geometrical element it is necessary in many definitions to use modifiers :

NEGX	IN	TANTO	XLARGE	RIGHT	LARGE	CLW	PERPTO	XYPLAN
NEGY	OUT	INTOF	XSMALL	LEFT	SMALL	CCLW	PARLEL	YZPLAN
NEGZ			YLARGE				ATANGL	ZXPLAN
POSX			YSMALL				SLOPE	
POSY			ZLARGE					
POSZ			ZSMALL					

8.1.3.2 The use of IN or OUT before a symbol specifies that the element being defined lies entirely inside or outside, respectively, of the associated element.

8.1.3.3 The use of XLARGE, etc., specifies the position of the element being defined, relative to the associated element occurring next. XLARGE, for example, would indicate that the element being defined has larger values in x than another possible solution. In many cases there is a choice of modifier to give the desired solution; for example, unless a line is parallel to one of the coordinate axes then one of the pair XLARGE, XSMALL would indicate the same side of the line as one of the pair YLARGE, YSMALL.

8.1.3.4 The modifiers RIGHT and LEFT are used to choose between two possible solutions, the viewing direction being from the first associated element to the second.

8.1.3.5 The modifiers POSX or NEGX specify a direction in x which is positive or negative respectively; similarly for POSY and NEGY in the y-direction, and POSZ and NEGZ in the z-direction.

8.1.3.6 The modifiers LARGE and SMALL are used to choose between two possible solutions, the words being used in their natural sense.

8.1.3.7 The modifiers CLW and CCLW are used to specify a clockwise sense or counterclockwise (anticlockwise) sense respectively.

8.1.3.8 The modifier TANTO is used to indicate that the associated geometric elements are tangential and the modifier INTOF is preceded by a number which specifies the required interaction.

8.1.3.9 The modifier PERPTO is used to specify that the required geometric element is perpendicular to the associated geometric element; PARLEL is used to specify that the required geometric element is parallel to the associated geometric element; ATANGL is used to specify that the required geometric element is at a slope to the associated geometric element and SLOPE is used to specify that the required geometric element is at a slope to the associated geometric element.

8.1.3.10 XYPLAN is used to specify that the required geometric element lies in the XY plane, similarly for YZPLAN in the YZ plane and ZXPLAN in the ZX plane.

8.1.4 Specificator

The term "specificator" or "spec" is used to refer to the identifier which names a particular geometric quantity, or to a nested geometrical definition.

For example if a point is defined by its coordinates 2, 5 as

P1 = POINT/2,5

then, P1 is the "point specificator".

The following terms are used :

a) point specificator;

b) pattern specificator;

c) line specificator;

d) plane specificator;

e) vector specificator;

f) circle specificator;

g) cylinder specificator;

h) sphere specificator;

j) cone specificator;

k) ellipse specificator;

m) hyperbola specificator;

n) lofted conic specificator;

p) general conic specificator;

q) quadric specificator;

r) tabulated cylinder specificator;

s) matrix specificator;

t) ruled surface specificator.

A collective term "surface specificator" is also used. This is defined as :

< surface specificator > : : = < line spec > | < circle spec > | < plane spec > | < cylinder spec > | < sphere spec > |
< cone spec > | < ellipse spec > | < hyperbola spec > | < lofted conic spec > |
< general conic spec > | < quadric spec > | < tabulated cylinder spec > |
< ruled surface spec >

Some definitions use one of the alternatives, "point spec" or the x and y coordinates of the point. This is termed "point option".

< point option > : : = < point spec > | < scalar > , < scalar >

In pattern definitions, the term "pattern data" is used and this is defined as

< pattern data > : : = < pattern data item > $_0^k$ [, < pattern data >]

< pattern data item > : : = < scalar > | < scalar > , AT , < scalar >

< conic spec > : : = < ellipse spec > | < hyperbola spec > | < lofted conic spec > | < general conic spec >

8.2 Declarations of reference system

8.2.1 Declaration of a local reference system

REFSYS/reference system parameter list

8.2.1.1 Semantics

All elements defined in a part program are referred to the basic coordinate system, but a local reference system which is a three-axis Cartesian coordinate system rotated and/or translated with respect to the basic system may be used. The start of such a reference system is denoted by a REFSYS statement and it operates on subsequent geometric definitions until terminated by another REFSYS statement. All types of statements are allowable within a local reference system. Only geometric definitions are affected, the results of processing being canonical forms in the basic coordinate system.

8.2.1.2 Sub-contents

For the

1) declaration of a local reference system by a specified transformation matrix, see 8.2.2;

2) end of local reference system declaration, see 8.2.3.

8.2.1.3 Limitations

CLDATA information containing coordinates produced within a local reference system may be in either the local reference system or the basic coordinate system depending upon the implementor. Matrices and tabulated cylinders are not transformed from the coordinate system in which defined if referenced in another coordinate system.

8.2.1.4 Syntax

< reference system declaration > : : = REFSYS/ < reference system parameter list >

8.2.2 Declaration of a local reference system by a specified transformation matrix

REFSYS/matrix

8.2.2.1 Semantics

The matrix is a previously defined transformation matrix which transforms geometric definitions from the local reference system to the basic reference system. It replaces any prior reference system.

8.2.2.2 Limitations

None.

8.2.2.3 Syntax

< reference system parameter list > : : = < matrix spec >

8.2.3 End of local reference system declaration

REFSYS/NOMORE

8.2.3.1 Semantics

This statement terminates the operation of a local reference system.

8.2.3.2 Limitations

None.

8.2.3.3 Syntax

< reference system parameter list > : : = NOMORE

8.3 Declarations of *z*-surface

8.3.1 Declaration of *z*-surface

ZSURF/*z*-surface parameter list

8.3.1.1 Semantics

When a point is defined by the intersection of two geometric elements or by its *x*- and *y*-coordinates, or when a pattern of points is defined, the unspecified *z*-coordinate has a value of zero unless a ZSURF statement defining a non-zero value is in effect.

By means of a ZSURF declaration a *z*-coordinate value is associated with subsequent point definitions which do not have an explicit *z*-coordinate value, and with subsequent pattern definitions.

8.3.1.2 Sub-contents

For the ZSURF declaration by

1) a value for the *z*-coordinate, see 8.3.2;
2) a plane, see 8.3.3;
3) a cylinder, see 8.3.4;
4) a sphere, see 8.3.5;
5) a cone, see 8.3.6;
6) a quadric, see 8.3.7.

8.3.1.3 Limitation

The *z*-coordinate is undefined if the named surface is not intersected by a vector that passes through the point and is parallel to the *Z*-axis.

8.3.1.4 Syntax

< *z*-surface declaration > : : = ZSURF/ < *z*-surface parameter list >

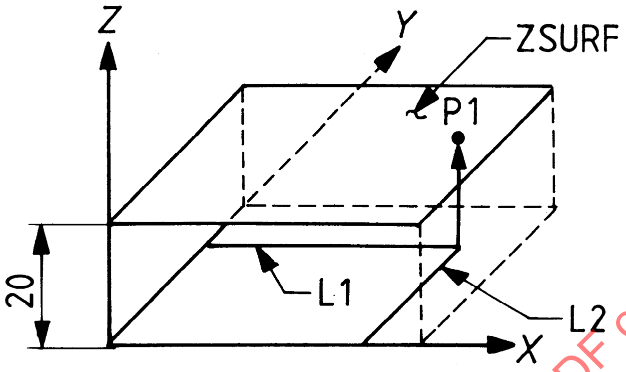
8.3.2 Declaration of z-surface parallel to the XY-plane by its height or distance from the XY-plane

ZSURF/height

8.3.2.1 Semantics

The height is the z-coordinate value associated with certain subsequent geometric definitions.

8.3.2.2 Example



ZSURF/20

P1 = POINT/INTOF, L1, L2

Figure 2

8.3.2.3 Limitations

None.

8.3.2.4 Syntax

< z-surface parameter list > : : = < scalar >

8.3.3 Declaration of z-surface by a specified non-vertical plane

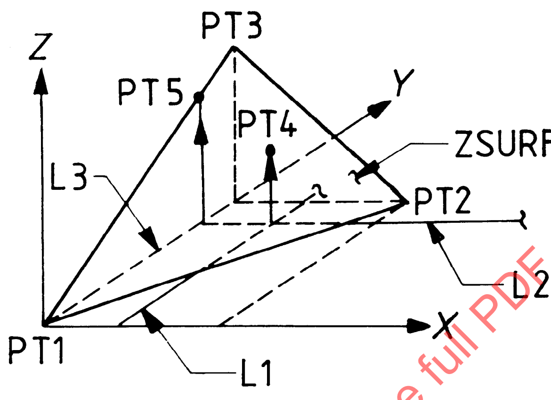
ZSURF/plane

8.3.3.1 Semantics

Plane is the symbol for a plane. If the plane is parallel to the XY-plane then the z-coordinate value associated with certain subsequent geometric definitions is the distance from the XY-plane.

If the plane is not parallel to the XY-plane, the implied z-coordinate value is not constant but varies with x- and y-coordinate values.

8.3.3.2 Example



PL1 = PLANE/PT1, PT2, PT3
ZSURF/PL1

PT4 = POINT/INTOF, L1, L2

PT5 = POINT/INTOF, L2, L3

Figure 3

8.3.3.3 Limitations

None.

8.3.3.4 Syntax

< z-surface parameter list > : : = < plane spec >

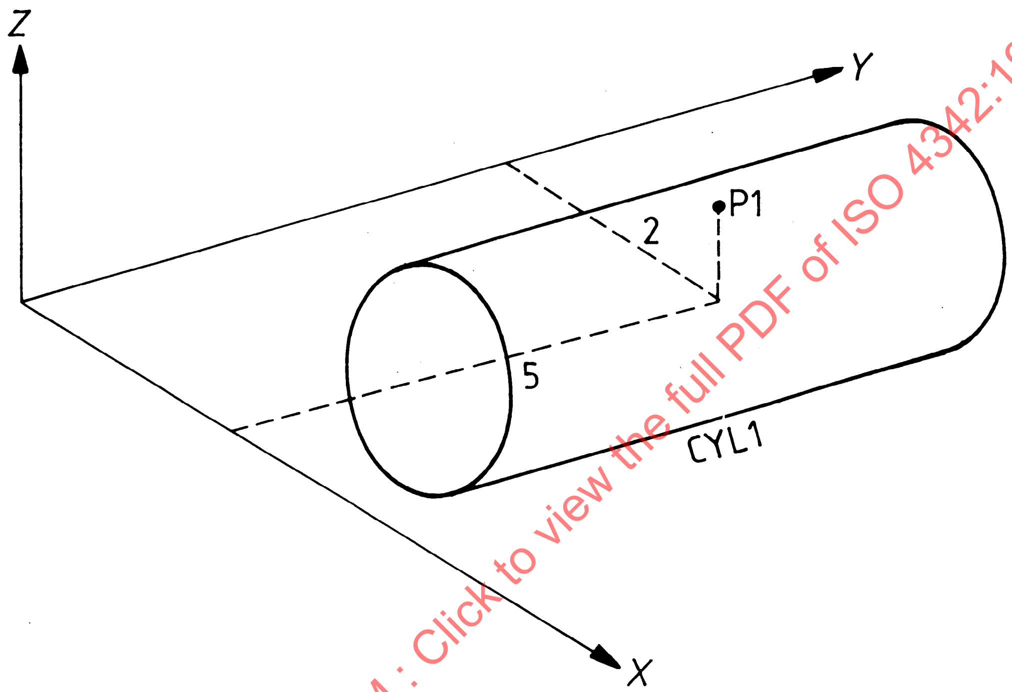
8.3.4 Declaration of a *z*-surface by a specified non-vertical cylinder

ZSURF/cylinder ,ZLARGE
,ZSMALL

8.3.4.1 Semantics

Cylinder is the symbol for a cylinder. ZLARGE and ZSMALL are used to distinguish between two possible solutions. If neither is given then ZLARGE is assumed.

8.3.4.2 Example



ZSURF/CYL1, ZLARGE

P1 = POINT/2, 5

Figure 4

8.3.4.3 Limitations

None.

8.3.4.4 Syntax

< *z*-surface parameter list > : : = < cylinder spec > $^{1}_{0}$ [,ZLARGE | ,ZSMALL]

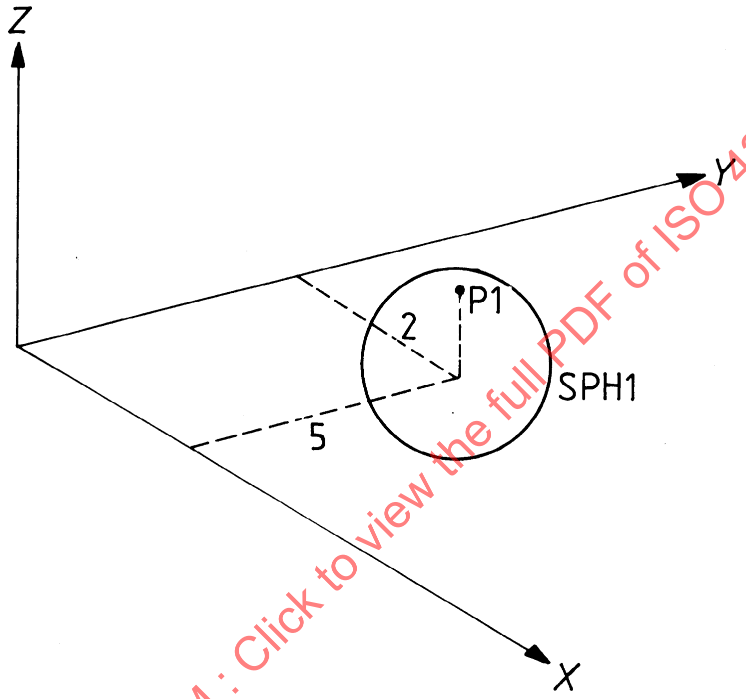
8.3.5 Declaration of a z-surface by a specified sphere

ZSURF/sphere ,ZLARGE
,ZSMALL

8.3.5.1 Semantics

Sphere is the symbol for a sphere. ZLARGE and ZSMALL are used to distinguish between two possible solutions. If neither is given then ZLARGE is assumed.

8.3.5.2 Example



ZSURF/SPH1, ZLARGE

P1 = POINT/2, 5

Figure 5

8.3.5.3 Limitations

None.

8.3.5.4 Syntax

< z-surface parameter list > : : = < sphere spec > $^{1}_{0}$ [,ZLARGE | ,ZSMALL]

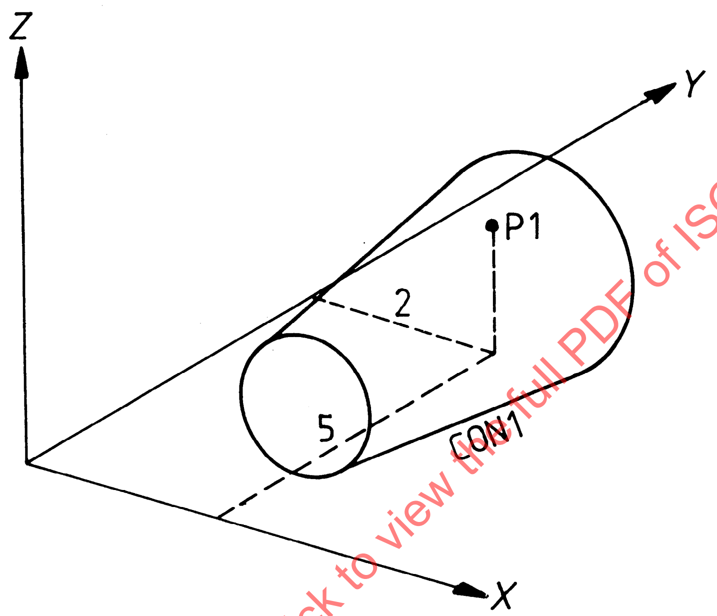
8.3.6 Declaration of a *z*-surface by a specified cone

ZSURF/cone ,ZLARGE
,ZSMALL

8.3.6.1 Semantics

Cone is the symbol for a cone. ZLARGE and ZSMALL are used to distinguish between two possible solutions. If neither is given then ZLARGE is assumed.

8.3.6.2 Example



ZSURF/CON1, ZLARGE

P1 = POINT/2, 5

Figure 6

8.3.6.3 Limitations

None.

8.3.6.4 Syntax

< *z*-surface parameter list > : : = < cone spec > $_0^1$ [,ZLARGE | ,ZSMALL]

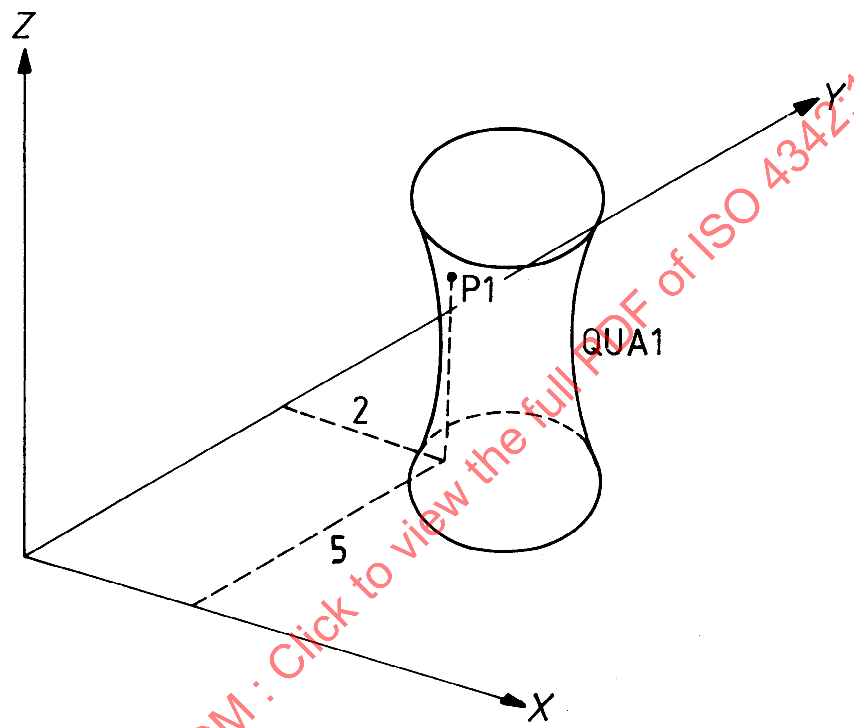
8.3.7 Declaration of a *z*-surface by a specified quadric

ZSURF/quadric ,ZLARGE
,ZSMALL

8.3.7.1 Semantics

Quadric is the symbol for a quadric. ZLARGE and ZSMALL are used to distinguish between two possible solutions. If neither is given then ZLARGE is assumed.

8.3.7.2 Example



ZSURF/QUA1, ZLARGE

P1 = POINT/2, 5

Figure 7

8.3.7.3 Limitations

None.

8.3.7.4 Syntax

< *z*-surface parameter list > : : = < quadric spec > $^{1}_{0}$ [,ZLARGE | ,ZSMALL]

8.4 Definitions of a point

8.4.1 Definition of a single point

POINT/point parameter list

8.4.1.1 Semantics

The point has three coordinates x, y, z.

When the z-coordinate is not explicitly given, the program computes it from the current ZSURF.

When no explicit ZSURF statement is given, the XY plane is assumed.

8.4.1.2 Context dependency

ZSURF except when the point is defined by its three coordinates or as the intersection of three planes.

8.4.1.3 Sub-contents

For the definition of a point

1) by its rectangular coordinates, see 8.4.2;
2) as intersection of two lines, see 8.4.3;
3) as intersection of a line and a circle, see 8.4.4;
4) as intersection of two circles, see 8.4.5;
5) by its polar coordinates in the coordinate plane mentioned in the statement, see 8.4.6;
6) on a circle at an angle with the X-axis, see 8.4.7;
7) as the nth point of a pattern, see 8.4.8;
8) as the centre of a circle, see 8.4.9;
9) as the intersection of a line and a conic, see 8.4.10;
10) as intersection of a line and a tabulated cylinder near a given point, see 8.4.11;
11) as the intersection of three planes, see 8.4.12;
12) at increments in x, y and z from a previously defined point, see 8.4.13;
13) by its polar coordinates relative to a previously defined point, see 8.4.14;
14) on a line with its x- or y-coordinate given, see 8.4.15;
15) at a distance along a tabulated cylinder from another point, see 8.4.16.

8.4.1.4 Syntax

<point definition statement> : : = <identifier> = POINT/ <point parameter list>

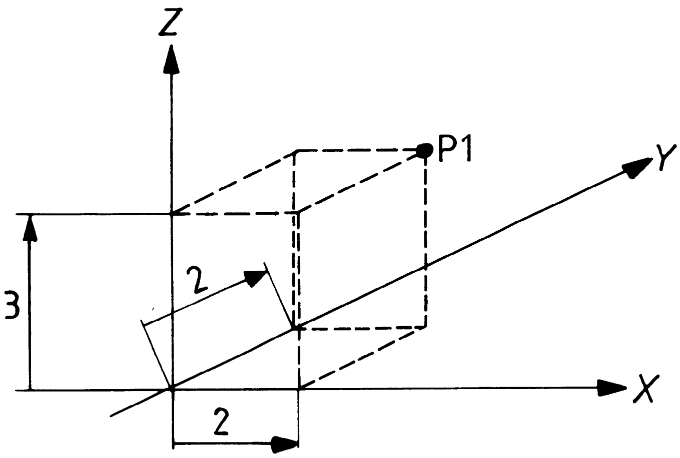
8.4.2 Definition of a point by its rectangular coordinates

POINT/*x*, *y*, *z*

8.4.2.1 Semantics

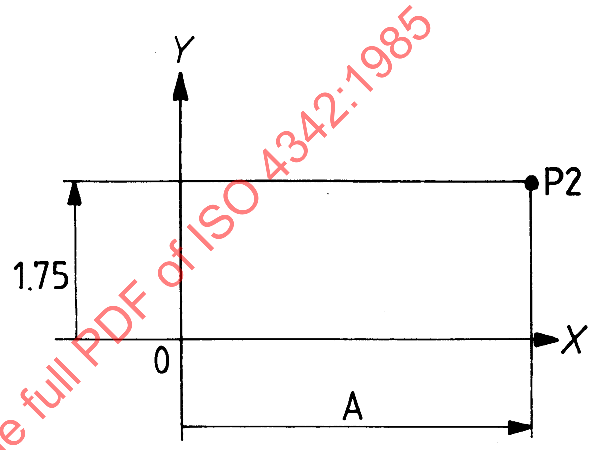
x, *y*, *z* represent the coordinates along the *X*-axis, *Y*-axis, *Z*-axis respectively.

8.4.2.2 Example



P1 = POINT/2, 2, 3

Figure 8a)



A = 2.25
P2 = POINT/A, 1.75

Figure 8b)

8.4.2.3 Limitations

None.

8.4.2.4 Syntax

< point parameter list > ::= < scalar > , < scalar > $_0^1$ [, < scalar >]

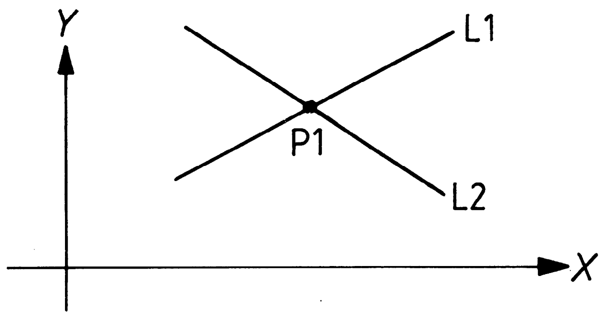
8.4.3 Definition of a point as intersection of two lines

POINT/INTOF, line1, line2

8.4.3.1 Semantics

Line1 and line2 are the symbols for two lines.

8.4.3.2 Example



P1 = POINT/INTOF, L1, L2

Figure 9

8.4.3.3 Limitations

The two lines shall be different and cannot be parallel.

8.4.3.4 Syntax

<point parameter list> : : = INTOF, <line spec>, <line spec>

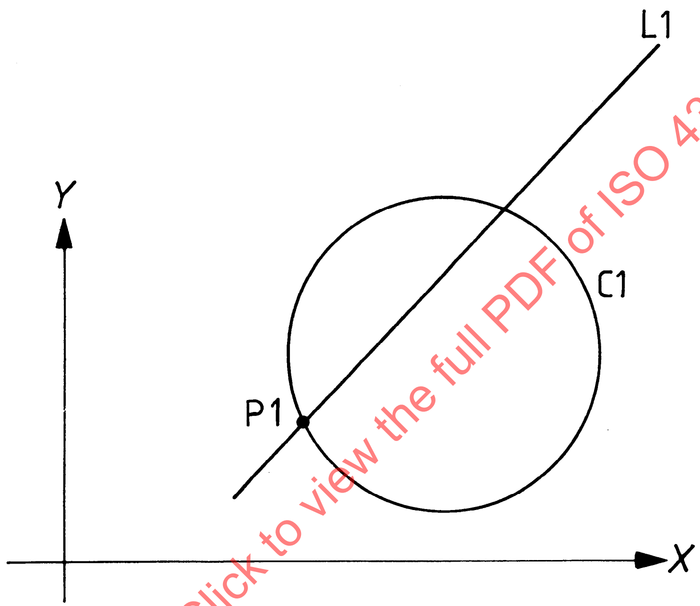
8.4.4 Definition of a point as intersection of a line and a circle

POINT/ XLARGE | XSMALL | YLARGE | YSMALL , INTOF, line, circle

8.4.4.1 Semantics

Line and circle are symbols for a line and a circle respectively.

8.4.4.2 Example



P1 = POINT/XSMALL, INTOF, L1, C1

Figure 10

8.4.4.3 Limitations

The line and the circle shall intersect or be tangential.

When the line is parallel to a coordinate axis, only the modificators associated with this axis shall be used; for example YLARGE, YSMALL shall not be used when the line is parallel to the X-axis.

8.4.4.4 Syntax

< point parameter list > : : = |XLARGE|XSMALL|YLARGE|YSMALL|, INTOF, < line spec > , < circle spec >

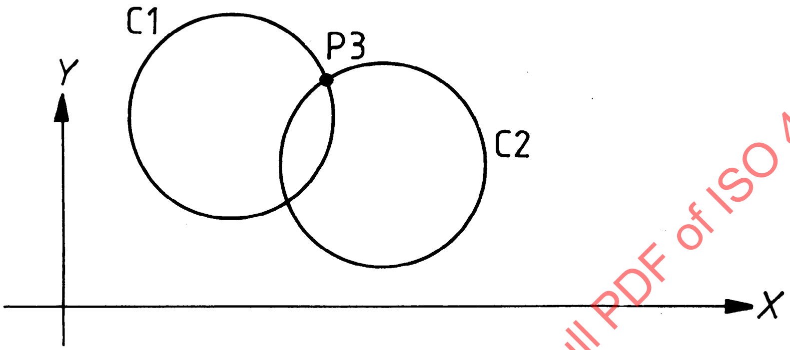
8.4.5 Definition of a point as intersection of two circles

```
        XLARGE
POINT/  XSMALL  , INTOF, circle1, circle2
        YLARGE
        YSMALL
```

8.4.5.1 Semantics

Circle1 and circle2 are the symbols for two circles.

8.4.5.2 Example



P3 = POINT/YLARGE, INTOF, C1, C2

Figure 11

8.4.5.3 Limitations

The two circles shall intersect or be tangential.

When the line defined by the centres of the two circles is parallel to a coordinate axis, only the modificators associated with the other axis may be used : for example YLARGE or YSMALL when the two centres have the same y-coordinate.

8.4.5.4 Syntax

< point parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL], INTOF, < circle spec > , < circle spec >

8.4.6 Definition of a point by its polar coordinates in the coordinate plane mentioned in the statement

```
                XYPLAN
POINT/RTHETA,   YZPLAN, radius, angle
                ZXPLAN
```

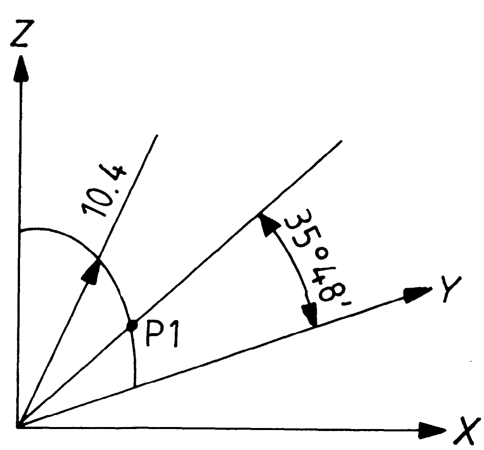
8.4.6.1 Semantics

RTHETA specifies that the radius is given before the angle.

Radius represents the vector radius.

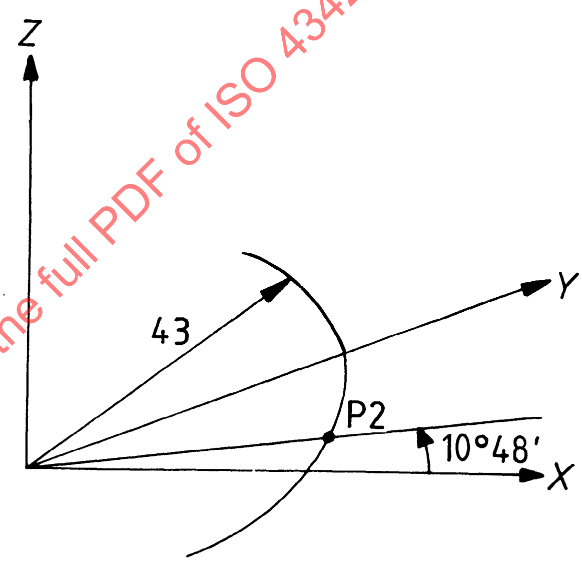
Angle represents the algebraic angle from the first referenced axis to the vector radius.

8.4.6.2 Example



P1 = POINT/RTHETA, YZPLAN, 10.4, 35.8

Figure 12a)



P2 = POINT/RTHETA, XYPLAN, 43, 10.8

Figure 12b)

8.4.6.3 Limitations

The radius may not have a negative value.

8.4.6.4 Syntax

< point parameter list > : : = RTHETA, [XYPLAN | YZPLAN | ZXPLAN], < scalar > , < scalar >

8.4.7 Definition of a point on a circle at an angle with the *X*-axis

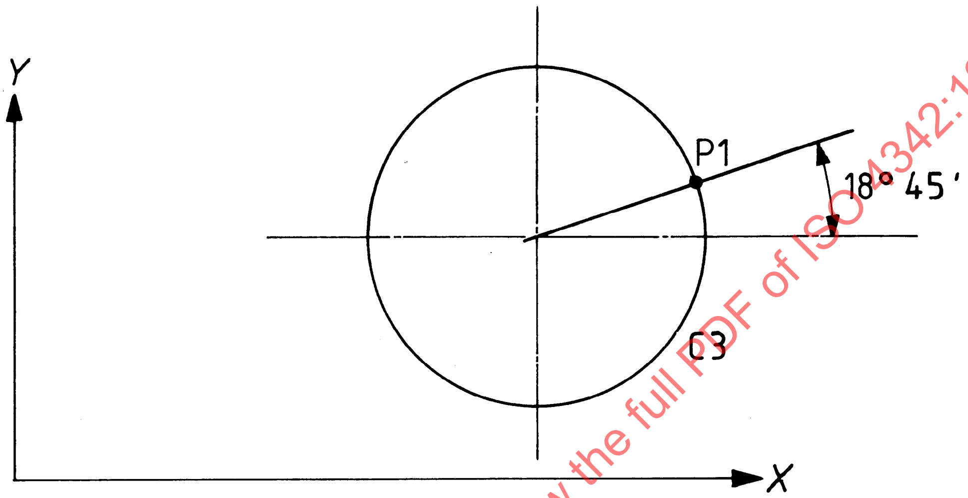
POINT/circle, ATANGL, angle

8.4.7.1 Semantics

Circle is the symbol for a circle.

Angle represents the algebraic angle from the *X*-axis to the radius ending at the point.

8.4.7.2 Example



P1 = POINT/C3, ATANGL, 18.75

Figure 13

8.4.7.3 Limitations

None.

8.4.7.4 Syntax

< point parameter list > : : = < circle spec > , ATANGL, < scalar >

8.4.8 Definition of a point as the *n*th point of a pattern

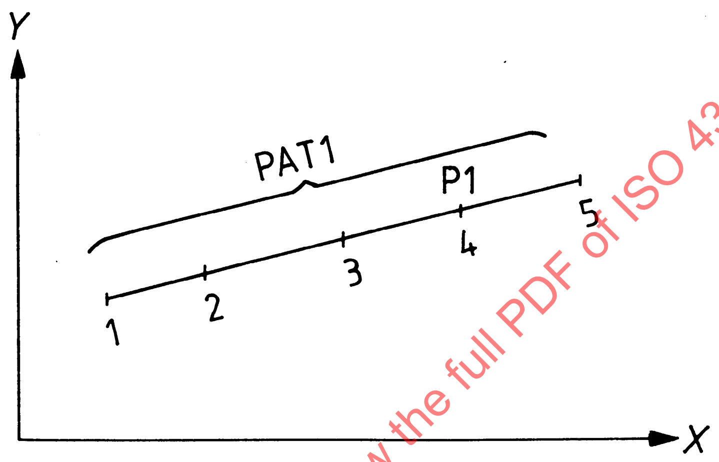
POINT/pattern, point number

8.4.8.1 Semantics

Pattern is the symbol for a pattern of points.

Number represents the sequence number of the point in the pattern.

8.4.8.2 Example



P1 = POINT/PAT1, 4

Figure 14

8.4.8.3 Limitations

The value of the point number shall result in an integer; an integer whose value shall not be greater than the total number of points in the pattern nor less than one.

8.4.8.4 Syntax

< point parameter list > : : = < pattern spec > , < unsigned number >

8.4.9 Definition of a point as the centre of a circle

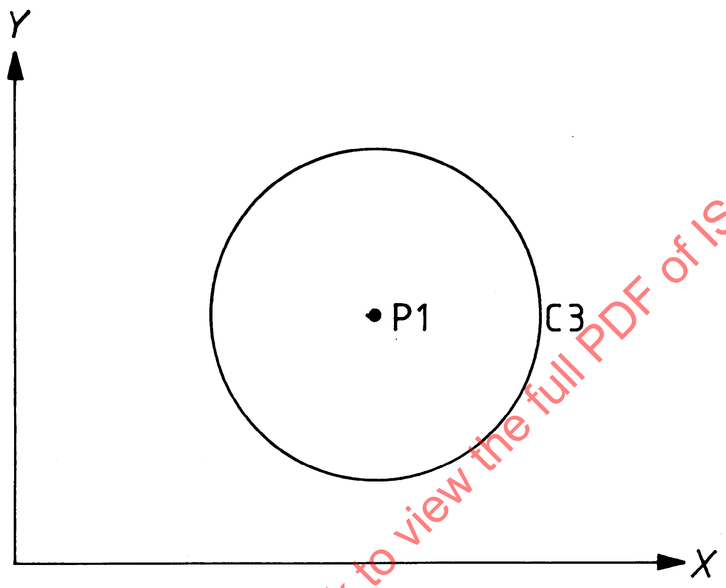
POINT/CENTER, circle

8.4.9.1 Semantics

CENTER specifies that the center of the associated circle is required.

Circle is the symbol for a circle.

8.4.9.2 Example



P1 = POINT/CENTER, C3

Figure 15

8.4.9.3 Limitations

None.

8.4.9.4 Syntax

< point parameter list > : : = CENTER, < circle spec >

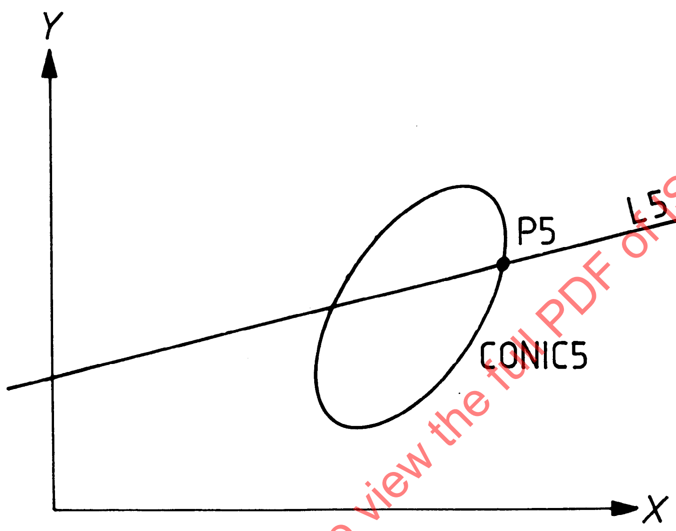
8.4.10 Definition of a point as the intersection of a line and a conic

```
        XLARGE
POINT/  XSMALL  , INTOF, line, conic
        YLARGE
        YSMALL
```

8.4.10.1 Semantics

Line and conic are symbols for a line and a conic respectively.

8.4.10.2 Example



P5 = POINT/XLARGE, INTOF, L5, CONIC5

Figure 16

8.4.10.3 Limitations

The line and the conic shall intersect or be tangential. When the line is parallel to a coordinate axis, only the modifiers associated with this axis may be used, for example only XLARGE, XSMALL may be used when the line is parallel to the X-axis.

8.4.10.4 Syntax

< point parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL], INTOF, < line spec > , <conic spec >

8.4.11 Definition of a point as intersection of a line and a tabulated cylinder, near a given point

POINT/INTOF, line, tabcyl, point

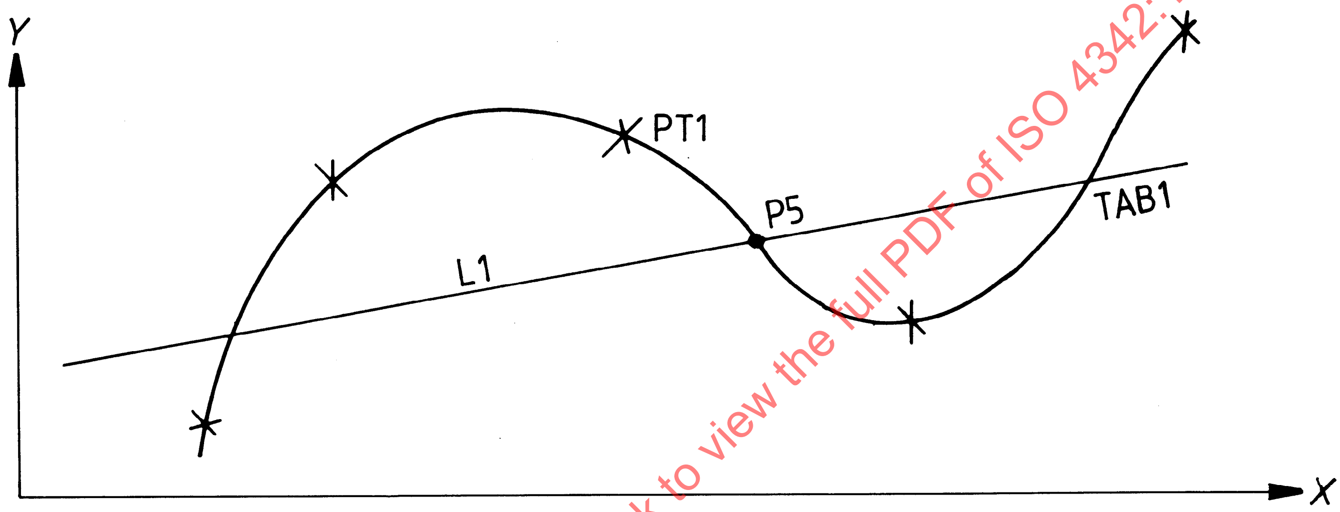
8.4.11.1 Semantics

Line is a symbol for a line.

Tabcyl is the symbol for a tabulated cylinder perpendicular to the XY-plane.

Point is the symbol for a point near the point being defined. This nearby point is needed to determine the proper point in case of multiple intersections.

8.4.11.2 Example



P5 = POINT/INTOF, L1, TAB1, PT1

Figure 17

8.4.11.3 Limitations

Multiple intersections shall not be allowed between the line and the tabcyl segment selected from the near point depending on implementation.

The line and tabcyl shall intersect or be tangential.

8.4.11.4 Syntax

< point parameter list > : : = INTOF, < line spec > , < tabulated cylinder spec > , < point spec >

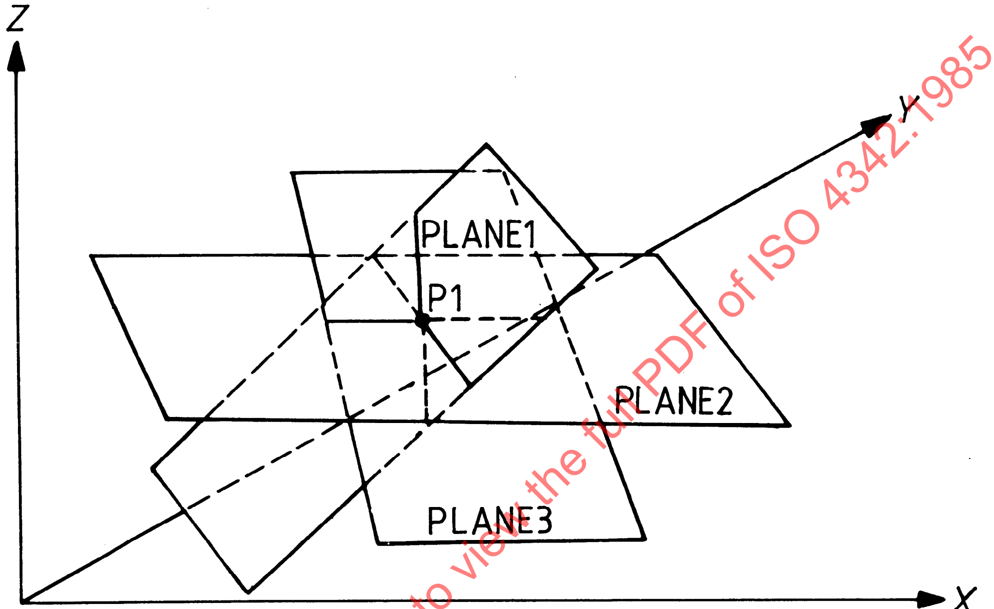
8.4.12 Definition of a point as the intersection of three planes

POINT/INTOF, plane1, plane2, plane3

8.4.12.1 Semantics

Plane1, plane2, plane3 are the symbols for the three planes.

8.4.12.2 Example



P1 = POINT/INTOF, PLANE1, PLANE2, PLANE3

Figure 18

8.4.12.3 Limitations

Any two planes shall be different and cannot be parallel.

8.4.12.4 Syntax

< point parameter list > : : = INTOF, < plane spec > , < plane spec > , < plane spec >

8.4.13 Definition of a point at x, y and z increments from a previously defined point

POINT/point, DELTA, increment1, increment2, increment3

8.4.13.1 Semantics

Point is the symbol for a defined point.

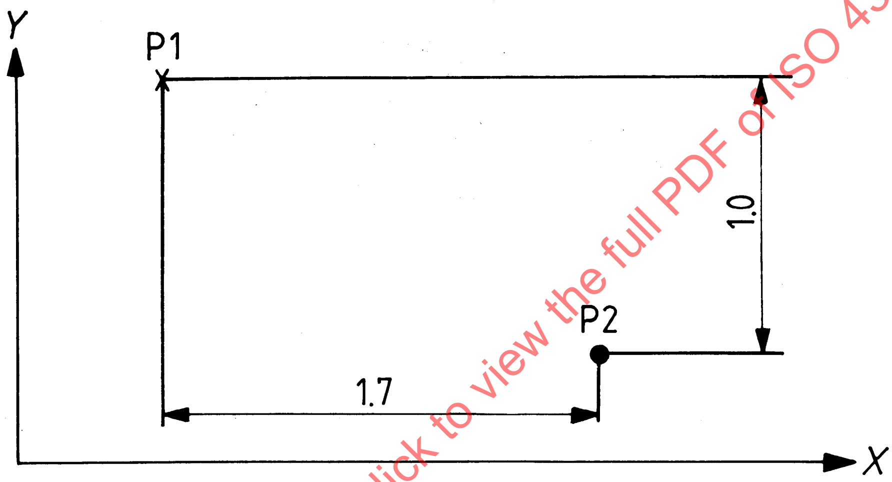
DELTA specifies that the following values are incremental values from the specified point :

a) Increment1 is the x increment.

b) Increment2 is the y increment.

c) Increment3 is the z increment. If omitted, a zero z increment is assumed.

8.4.13.2 Example



P2 = POINT/P1, DELTA, 1.7, − 1.0

Figure 19

8.4.13.3 Limitations

None.

8.4.13.4 Syntax

< point parameter list > : : = < point spec > , DELTA, < scalar > , < scalar > $^{1}_{0}$ [, < scalar >]

8.4.14 Definition of a point at a radial distance and at an angle from a previously defined point

POINT/point, THETAR, angle, distance

POINT/point, RTHETA, distance, angle

8.4.14.1 Semantics

Point is the symbol for a point.

THETAR specifies that the angle is given before the radial distance.

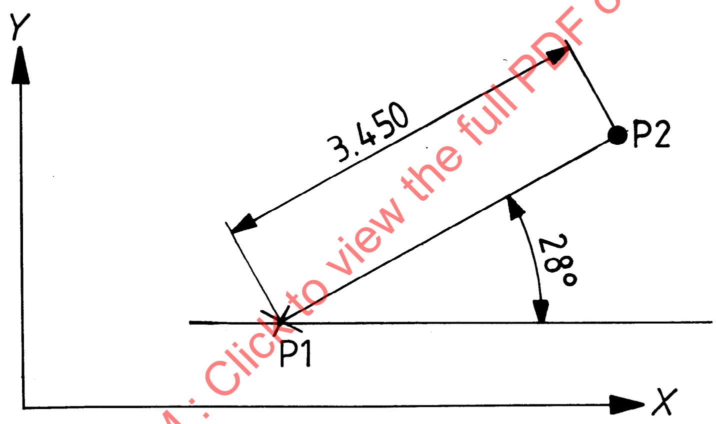
Angle is measured from a line parallel to the X-axis passing through the point to the radial line.

Distance is the length of the radial line.

RTHETA specifies that the radial distance is given before the angle.

ZSURF applies for the z-coordinate.

8.4.14.2 Example



P2 = POINT/P1, THETAR, 28, 3.45

P2 = POINT/P1, RTHETA, 3.45, 28

Figure 20

8.4.14.3 Limitations

The radial distance specified shall be positive.

8.4.14.4 Syntax

< point parameter list > : : = < point spec > ,[THETAR | RTHETA], < scalar > , < scalar >

8.4.15 Definition of a point on a line and with a given *x*- or *y*-coordinate value

POINT/line , XCOORD / YCOORD , number

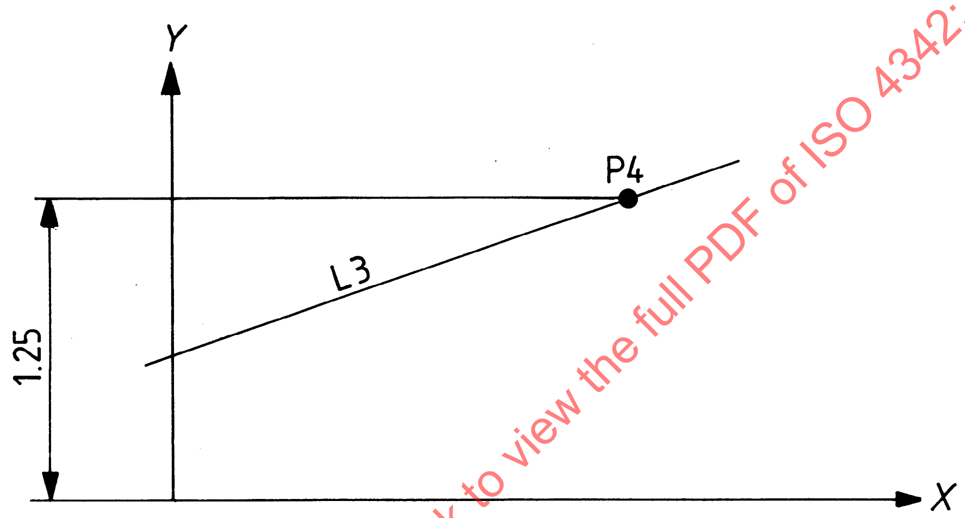
8.4.15.1 Semantics

Line is the symbol for a line.

XCOORD specifies that the following number is an *x*-coordinate value.

YCOORD specifies that the following number is a *y*-coordinate value.

8.4.15.2 Example



P4 = POINT/L3, YCOORD, 1.25

Figure 21

8.4.15.3 Limitations

None.

8.4.15.4 Syntax

< point parameter list > : : = < line spec > ,[XCOORD | YCOORD], < scalar >

8.4.16 Definition of a point at a given distance along a tabulated cylinder from a point on the tabulated cylinder

```
                XLARGE
POINT/point,    XSMALL  , tabcyl, distance
                YLARGE
                YSMALL
```

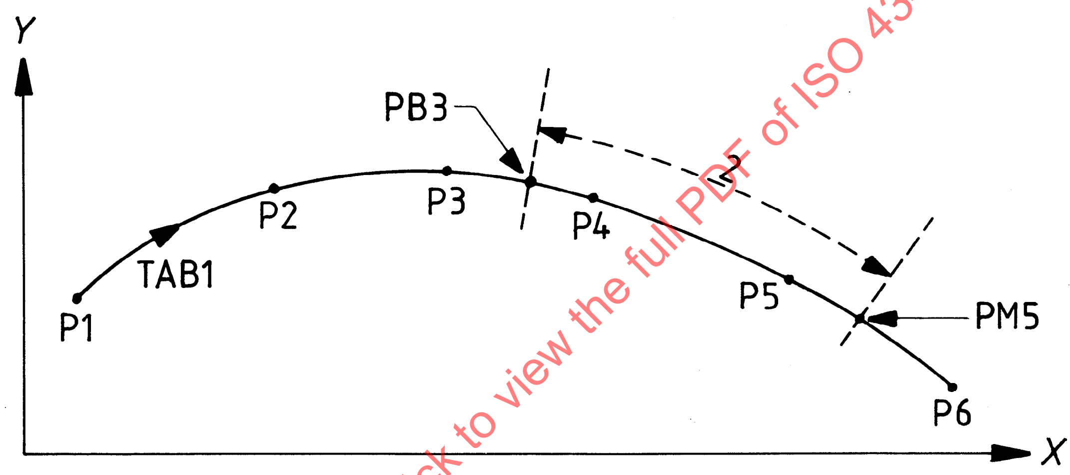
8.4.16.1 Semantics

Point is the symbol for the given point on the tabulated cylinder.

Tabcyl is the symbol for the tabulated cylinder perpendicular to the XY-plane.

Distance is the distance along the tabulated cylinder between the given point and the required point.

8.4.16.2 Example



PB3 = POINT/PM5, XSMALL, TAB1, 2

Figure 22

8.4.16.3 Limitations

Only positive distances are acceptable.

8.4.16.4 Syntax

< point parameter list > : : = < point spec > ,[XLARGE | XSMALL | YLARGE | YSMALL],
< tabulated cylinder spec > , < scalar >

8.5 Definitions of a pattern of points

8.5.1 Definition of a pattern of points

PATERN/pattern parameter list

8.5.1.1 Semantics

A pattern is a sequence of points, with each point having a sequence number implied by the method of definition.

8.5.1.2 Context dependency

ZSURF (see 8.3)
POINT (see 8.4)

8.5.1.3 Frequently used modifiers

LINEAR specifies that the pattern of points be along a straight line.

ARC specifies that the pattern of points be along a circular arc.

GRID specifies that the pattern of points forms a grid or parallelogram pattern.

8.5.1.4 Sub-contents

8.5.1.4.1 For pattern modification, see 8.5.2.

8.5.1.4.2 For the definition of a linear pattern

a) by starting and ending points and the total number of points in the pattern, see 8.5.3;
b) by the starting point, a vector for direction and spacing and the total number of points in the pattern, see 8.5.4;
c) by the starting point, a vector for direction, and increments in the direction of the vector, see 8.5.5;
d) by the starting point, a vector for direction, and a specified number of points in the direction of the vector, see 8.5.6;
e) by the starting point, an angle for direction, and a specified number of points in this direction, see 8.5.7.

8.5.1.4.3 For the definition of a circular pattern

a) by a circle, starting and ending angles, direction, and the total number of points, see 8.5.8;
b) by a circle, a starting angle, direction, and the total number of equally spaced points along the circle, see 8.5.9;
c) by a circle, a starting angle, direction, and angular increments along the circle, see 8.5.10;
d) by a circle, a starting angle, direction, and a specified number of points along the circle, see 8.5.11.

8.5.1.4.4 For the definition of a parallelogram pattern

a) by two linear patterns, see 8.5.12;
b) by a pattern, a vector for spacing and direction, and the total number of rows, see 8.5.13;
c) by a pattern, a vector for direction, and increments in the direction of the vector, see 8.5.14;
d) by a pattern, a vector for direction, and a specified number of points in the direction of the vector, see 8.5.15.

8.5.1.4.5 For the definition of combined patterns

a) by two patterns (TRAFO), see 8.5.16;
b) by points and patterns (RANDOM), see 8.5.17;
c) by a pattern and a mirroring line (MIRROR), see 8.5.18;
d) by a linear pattern and a circular pattern (CIRCUL), see 8.5.19.

8.5.1.5 Syntax

< pattern definition statement > : : = < identifier > = PATERN/ < pattern parameter list > $^{1}_{0}$ [,INVERS]

8.5.2 Pattern modification

8.5.2.1 Semantics

The implied ordering of the points in a pattern may be reversed through use of the modifier INVERS.

Whenever a pattern symbol is used as a modifier, it may be followed by either of the keywords OMIT or RETAIN followed by the sequence numbers of points in the pattern — number1 . . . number*n*. This causes the pattern to be modified only in the statement in which the pattern symbol is used : OMIT causes the numbered points to be excluded and RETAIN causes them to be included. Where more than two sequential point numbers are given, the modifier THRU may be used between the first numbered point and the last.

8.5.2.2 Example

Compare with the example in 8.5.3.

PAT3 = PATERN/LINEAR, P1, P2, 7, INVERS

P2 is the first point and P1 the seventh point in the sequence.

. . . pattern,	OMIT RETAIN	, number1, number2, ..., number*n*
. . . pattern,	OMIT RETAIN	, number1, THRU, number*n*

Compare with the example in 8.5.18.

PAT3 = PATERN/MIRROR, L1, PA2, OMIT, 2

This results in the second point of pattern PA2 being ignored in the calculation of PAT3.

8.5.2.3 Limitations

None.

8.5.2.4 Syntax

< modified pattern > : : = < pattern spec > $_0^1$ [,[OMIT | RETAIN], < scalar > $_0^k$ [[,THRU], < scalar >]] $_0^1$ [,INVERS]

8.5.3 Definition of a pattern by the starting and ending points and the total number of points in the pattern

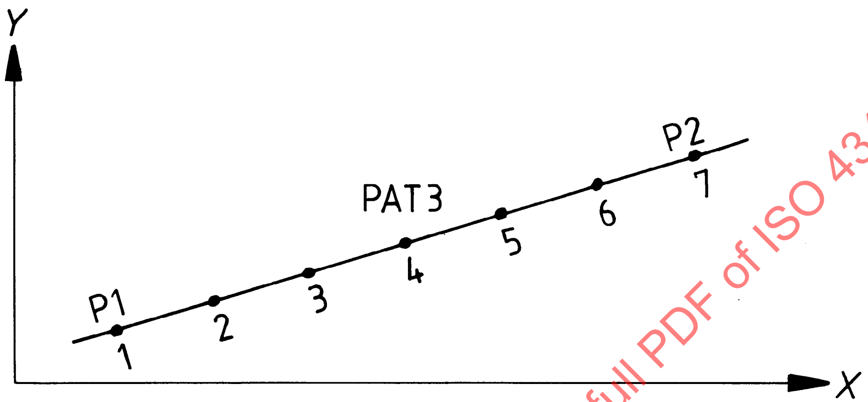
PATERN/LINEAR, point1, point2, number

8.5.3.1 Semantics

Point1 and point2 are the symbols for the starting and ending points.

Number specifies the total number of equally spaced points.

8.5.3.2 Example



PAT3 = PATERN/LINEAR, P1, P2, 7

Figure 23

8.5.3.3 Limitations

None.

8.5.3.4 Syntax

< pattern parameter list > : : = LINEAR, < point spec > , < point spec > , < scalar >

8.5.4 Definition of a pattern by the starting point, a vector for direction and spacing, and the total number of points in the pattern

PATERN/LINEAR, point, vector, number

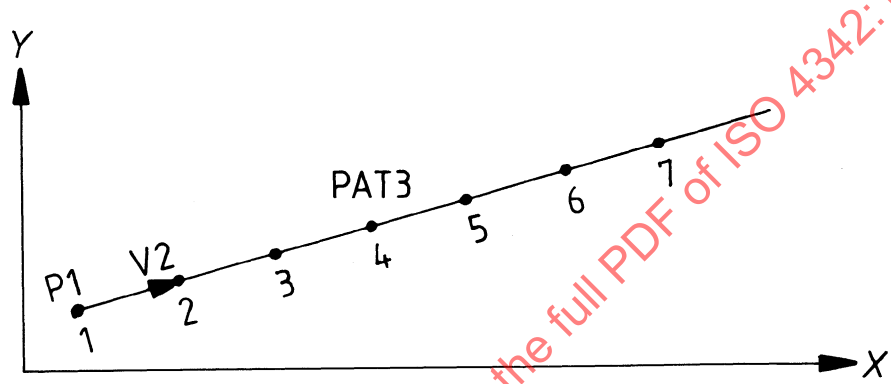
8.5.4.1 Semantics

Point is the symbol for the starting point of the pattern.

Vector is the symbol for a vector which gives the direction and spacing of the points.

Number specifies the total number of equally spaced points.

8.5.4.2 Example



PAT3 = PATERN/LINEAR, P1, V2, 7

Figure 24

8.5.4.3 Limitations

None.

8.5.4.4 Syntax

<pattern parameter list> : : = LINEAR, <point spec> , <vector spec> , <scalar>

8.5.5 Definition of a pattern by the starting point, a vector for direction, and increments in the direction of the vector

PATERN/LINEAR, point, vector, INCR, increment1, . . ., increment*n*

8.5.5.1 Semantics

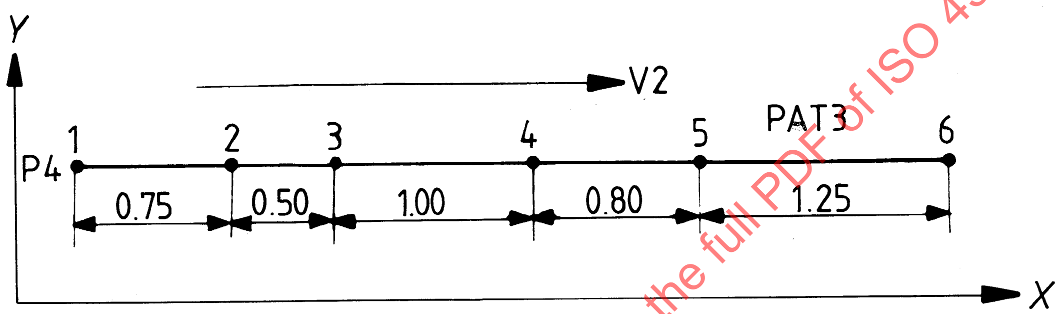
Point is the symbol for the starting point of the pattern.

Vector is the symbol for a vector which gives the direction of the pattern.

INCR specifies that the following values are incremental.

Increment1 specifies the distance from the first point, increment*n* the distance from the previous point, etc.

8.5.5.2 Example



PAT3 = PATERN/LINEAR, P4, V2, INCR, 0.75, 0.5, 1, 0.8, 1.25

Figure 25

8.5.5.3 Limitations

The increment value shall not be negative.

8.5.5.4 Syntax

< pattern parameter list > : : = LINEAR, < point spec > , < vector spec > , INCR $_1^n$ [, < scalar >]

8.5.6 Definition of a pattern by the starting point, a vector for direction and a specified number of points in the direction of the vector

PATERN/LINEAR, point, vector, INCR, number1, AT, increment1, . . ., number*n*, AT, increment*n*

8.5.6.1 Semantics

Point is the symbol for the starting point of a pattern.

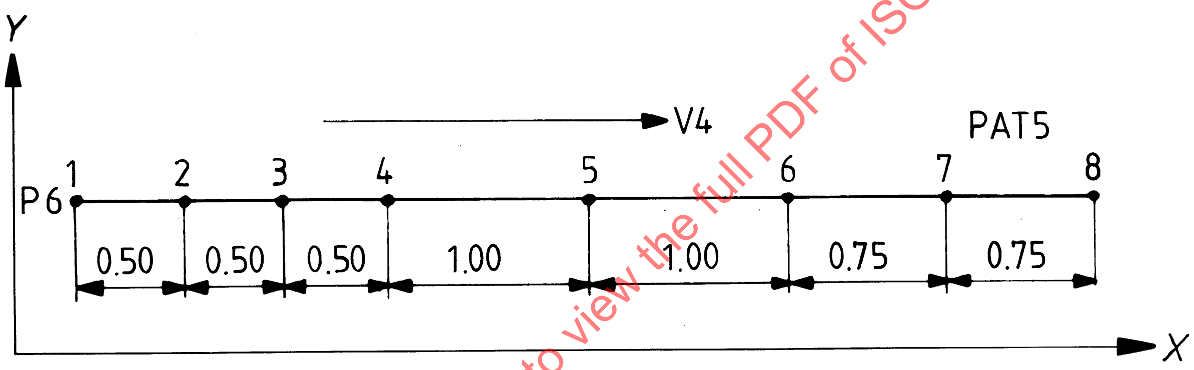
Vector is the symbol for a vector which gives the direction of the pattern.

INCR specifies that the following values are incremental.

Number1, AT, increment1 specifies the number of points at their incremental distances apart.

Number*n*, AT, increment*n* specifies a number of further points at their incremental distances apart.

8.5.6.2 Example



PAT5 = PATERN/LINEAR, P6, V4, INCR, 3, AT, 0.5, 2, AT, 1, 2, AT, 0.75

Figure 26

8.5.6.3 Limitations

The increment value shall not be negative.

8.5.6.4 Syntax

< pattern parameter list > : : = LINEAR, < point spec > , < vector spec > , INCR, < pattern data >

8.5.7 Definition of a pattern by the starting point, an angle for direction, and a specified number of points in this direction

PATERN/LINEAR, point, ATANGL, angle, INCR, number1, AT, increment1, . . ., number*n*, AT, increment*n*

8.5.7.1 Semantics

Point is the symbol for the starting point.

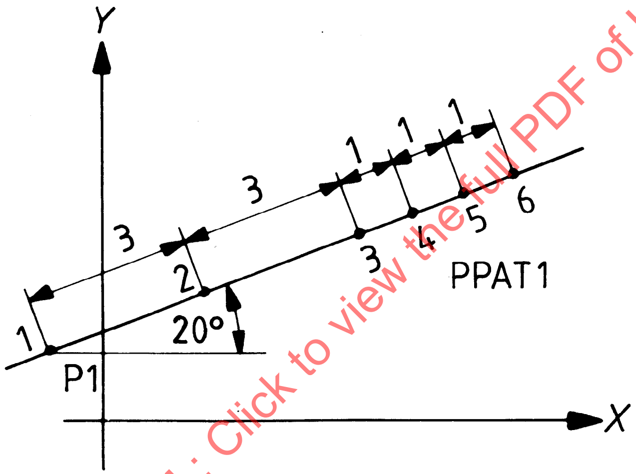
ATANGL, angle specifies the angle between the positive *X*-axis and a line through the starting point.

INCR specifies that the following values are incremental.

Number1, AT, increment1 specifies the number of points at their incremental distances apart.

Number*n*, AT, increment*n* specifies the number of further points at their incremental distances apart.

8.5.7.2 Example



PPAT1 = PATERN/LINEAR, P1, ATANGL, 20, INCR, 2, AT, 3, 3, AT, 1

Figure 27

8.5.7.3 Limitations

The increment value shall not be negative.

8.5.7.4 Syntax

< pattern parameter list > : : = LINEAR, < point spec > , ATANGL, < scalar > , INCR, < pattern data >

8.5.8 Definition of a pattern by a circle, starting and ending angles, direction and the total number of points

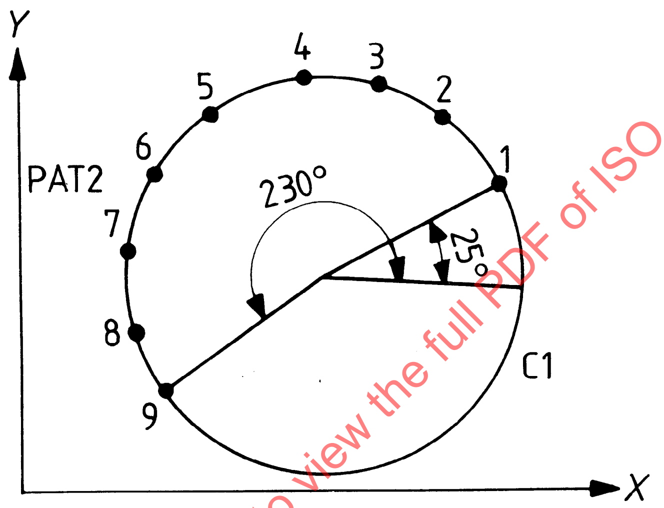
PATERN/ARC, circle, angle1, angle2, CLW / CCLW , number

8.5.8.1 Semantics

Circle is the symbol for a circle. Angle1 specifies the starting angle and angle2 the ending angle; both are measured from the positive *X*-axis to an imaginary line through the centre of the circle and the first or last location on the circle.

Number specifies the total number of equally spaced points.

8.5.8.2 Example



PAT2 = PATERN/ARC, C1, 25, 230, CCLW, 9

Figure 28

8.5.8.3 Limitations

None.

8.5.8.4 Syntax

< pattern parameter list > : : = ARC, < circle spec > , < scalar > , < scalar > ,[CLW | CCLW], < scalar >

8.5.9 Definition of a pattern by a circle, starting angle, direction, and the total number of equally spaced points along the circle

PATERN/ARC, circle, angle CLW / CCLW , number

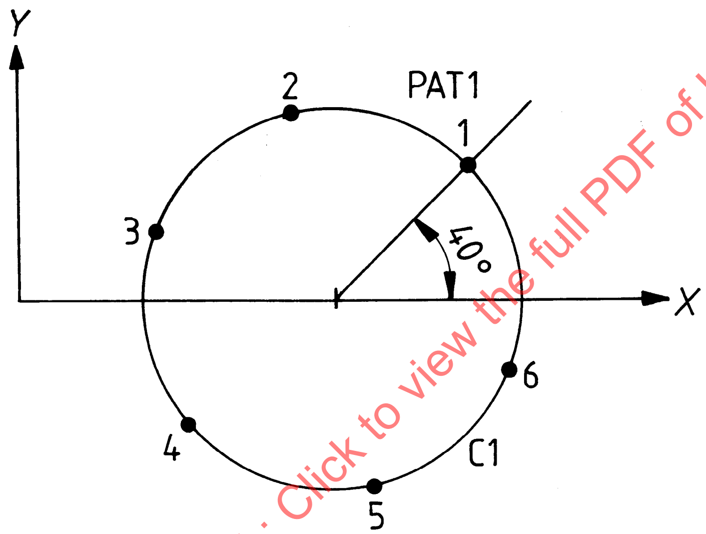
8.5.9.1 Semantics

Circle is the symbol for the circle.

Angle specifies the angle between the positive *X*-axis and an imaginary line through the centre of the circle and the starting position on the circle.

Number specifies the total number of equally spaced points.

8.5.9.2 Example



PAT1 = PATERN/ARC, C1, 40, CCLW, 6

Figure 29

8.5.9.3 Limitations

None.

8.5.9.4 Syntax

< pattern parameter list > : : = ARC, < circle spec > , < scalar > ,[CLW | CCLW], < scalar >

8.5.10 Definition of a pattern by a circle, a starting angle, direction, and angular increments along the circle

PATERN/ARC, circle, angle, CLW
CCLW , INCR, increment1, . . ., increment$n$

8.5.10.1 Semantics

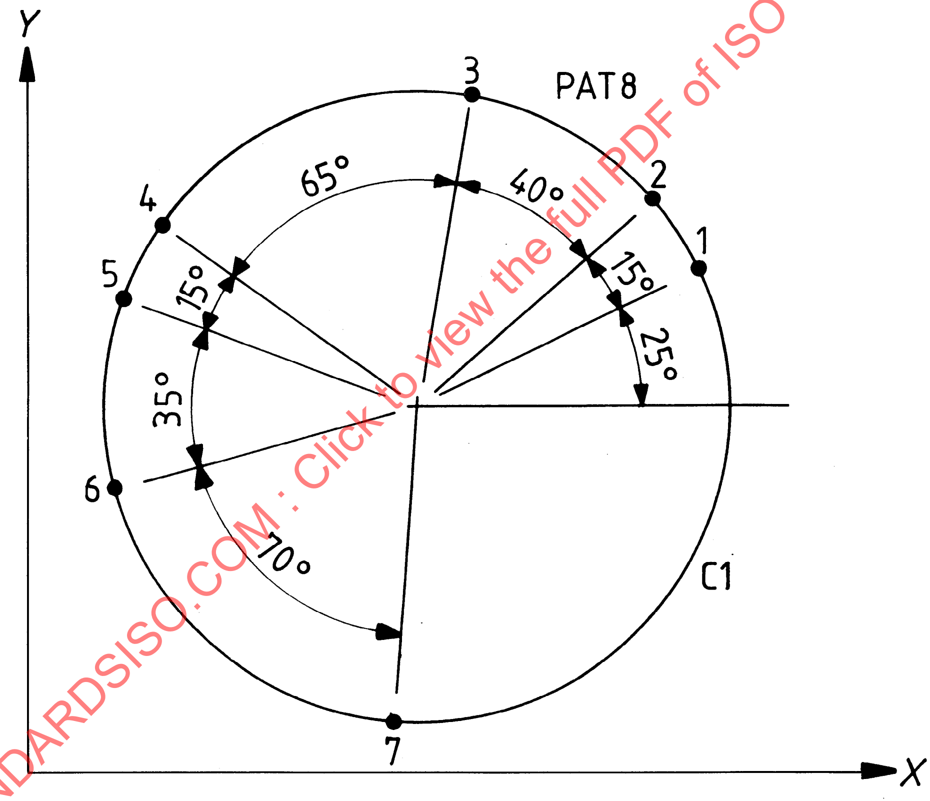
Circle specifies the symbol for the circle.

Angle specifies the starting angle from the positive X-axis to an imaginary line through the centre of the circle and the first location on the circle.

INCR specifies that the following values are incremental.

Increment1 specifies the angular distance from the first point, increment$n$ the angular distance from the previous point.

8.5.10.2 Example



PAT8 = PATERN/ARC, C1, 25, CCLW, INCR, 15, 40, 65, 15, 35, 70

Figure 30

8.5.10.3 Limitations

None.

8.5.10.4 Syntax

< pattern parameter list > : : = ARC, < circle spec > , < scalar > ,[CLW | CCLW], INCR, < pattern data >

8.5.11 Definition of a pattern by a circle, a starting angle, direction, and a specified number of points along the circle

PATERN/ARC, circle, angle, CLW / CCLW , INCR, number1, AT, increment1, . . ., number*n*, AT, increment*n*

8.5.11.1 Semantics

Circle is the symbol for a circle.

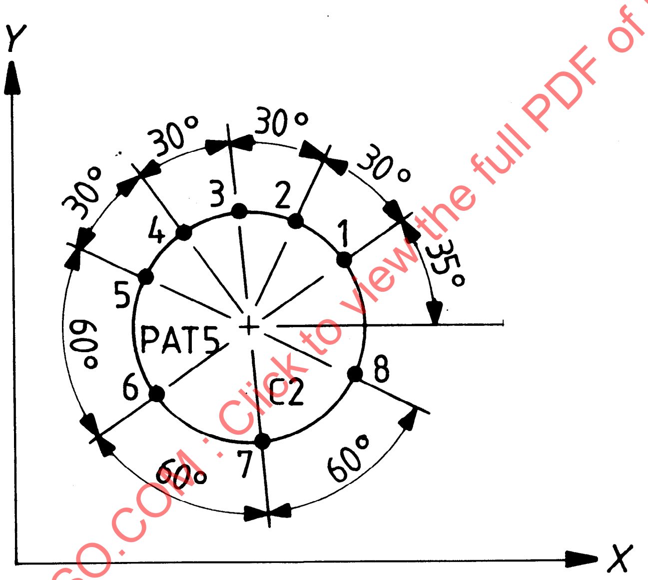
Angle specifies the starting angle from the positive X-axis to an imaginary line through the centre of the circle and the first location on the circle.

INCR specifies that the following values are incremental.

Number1, AT, increment1 specifies the number of points at their incremental values in degrees apart.

Number*n*, AT, increment*n* specifies the number of points at their incremental values in degrees apart.

8.5.11.2 Example



PAT5 = PATERN/ARC, C2, 35, CCLW, INCR, 4, AT, 30, 3, AT, 60

Figure 31

8.5.11.3 Limitations

None.

8.5.11.4 Syntax

< pattern parameter list > : : ARC, < circle spec > , < scalar > ,[CLW | CCLW], INCR, < pattern data >

8.5.12 Definition of a pattern by two linear patterns

PATERN/GRID, pattern1, pattern2

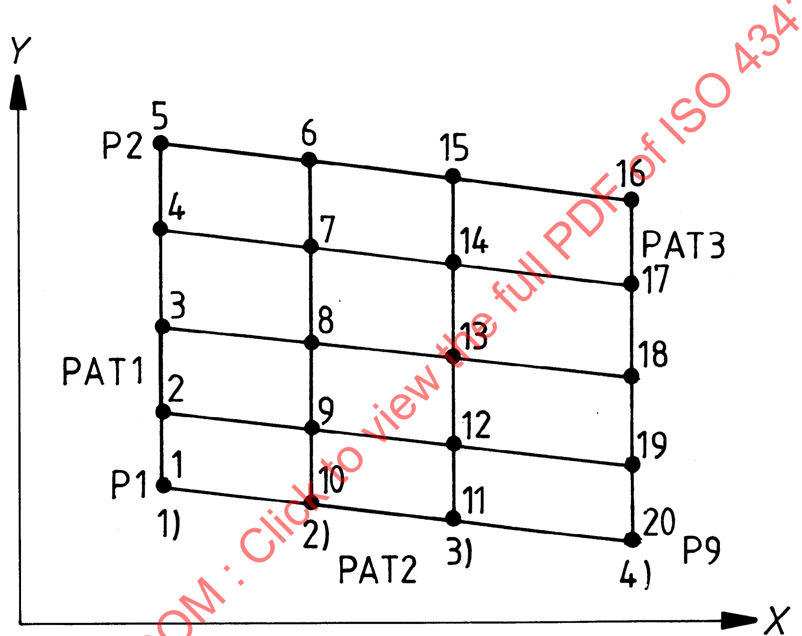
8.5.12.1 Semantics

Pattern1 and pattern2 are the symbols for two linear patterns.

The pattern is formed in such a way that a second pattern is started at every point of the first pattern. The resulting pattern is thus a parallelogram pattern.

The locations in the resulting pattern have a sequence which is determined by the order of the locations in the first pattern used in the GRID pattern. The first row is in the order of the first pattern, the second row is in the reverse order, etc.

8.5.12.2 Example



PAT1 = PATERN/LINEAR, P1, P2, 5

PAT2 = PATERN/LINEAR, P1, P9, 4

PAT3 = PATERN/GRID, PAT1 , PAT2

Figure 32

8.5.12.3 Limitations

Pattern1 and pattern2 shall be linear patterns.

8.5.12.4 Syntax

< pattern parameter list > : : = GRID, < linear pattern spec > , < linear pattern spec >

8.5.13 Definition of a pattern by a pattern, a vector for spacing and direction, and the total number of rows

PATERN/GRID, pattern, vector, number

8.5.13.1 Semantics

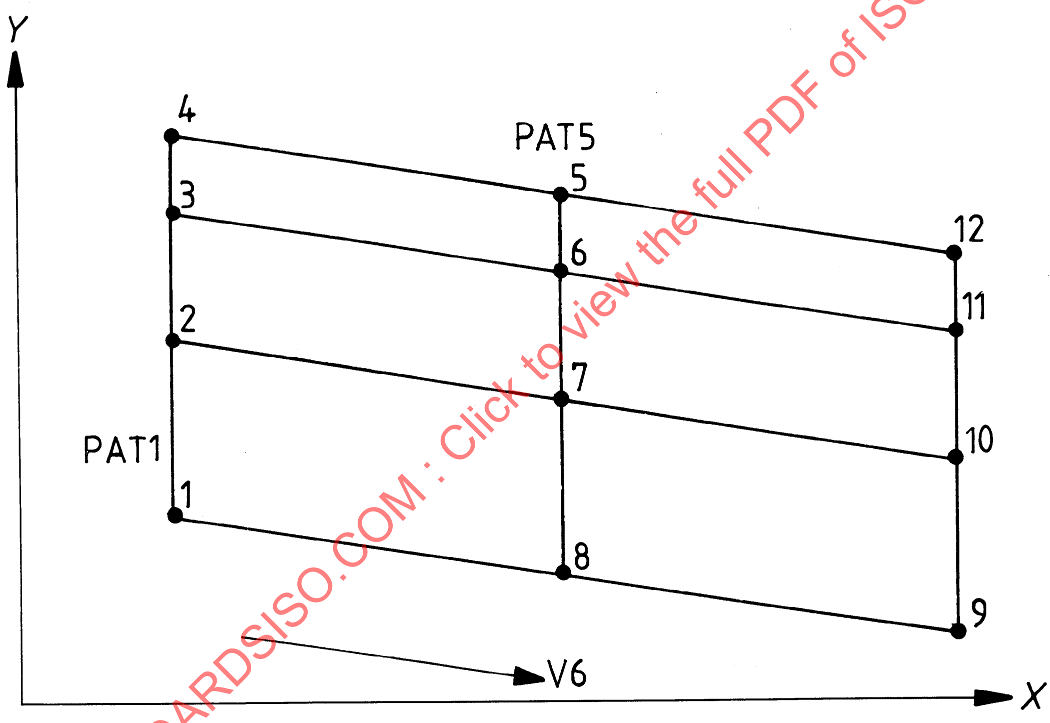
Pattern is the symbol for a linear pattern.

Vector is the symbol for a vector.

The pattern is formed in such a way that a vector is started at every point of the pattern. There then exists a second row of points that is parallel to that described in the pattern.

Number specifies the number of rows in the parallelogram pattern that are formed in this way. The locations in the resulting pattern have a sequence which is determined by the order of the locations in the pattern used in the GRID pattern. The first row is in the order of the first pattern, the second row is in the reverse order, etc.

8.5.13.2 Example



PAT5 = PATERN/GRID, PAT1, V6, 3

Figure 33

8.5.13.3 Limitations

The pattern shall be a linear pattern.

8.5.13.4 Syntax

< pattern parameter list > : : = GRID, < linear pattern spec > , < vector spec > , < scalar >

8.5.14 Definition of a pattern by a pattern, a vector for direction, and increments in the direction of the vector

PATERN/GRID, pattern, vector, INCR, increment1, . . ., increment*n*

8.5.14.1 Semantics

Pattern is the symbol for a linear pattern.

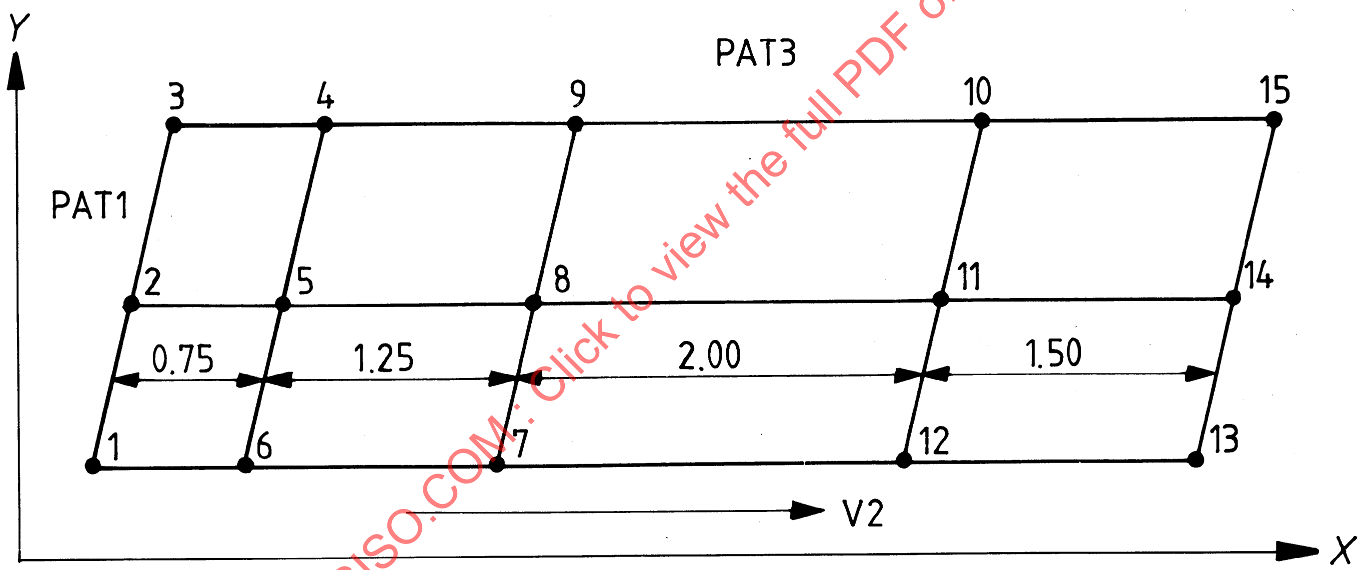
Vector is the symbol for a vector.

INCR specifies that the following values are incremental.

Increment1 is the distance from the first row, increment*n* from the previous row.

The pattern is formed in such a way that an increment is measured in the direction of the vector from every point of the pattern. There then exists a second row of points that is parallel to that described in the pattern. This is done once for every given increment. The locations in the resulting pattern have a sequence which is determined by the order of the locations in the first pattern used in the GRID pattern. The first row is in the order of the first pattern, the second row is in the reverse order, etc.

8.5.14.2 Example



PAT3 = PATERN/GRID, PAT1, V2, INCR, 0.75, 1.25, 2, 1.5

Figure 34

8.5.14.3 Limitations

The pattern shall be a linear pattern.

8.5.14.4 Syntax

< pattern parameter list > : : = GRID, < linear pattern spec > , < vector spec > , INCR, < pattern data >

8.5.15 Definition of a pattern by a pattern, a vector for direction, and a specified number of points in the direction of the vector

PATERN/GRID, pattern, vector, INCR, number1, AT, increment1, . . ., number*n*, AT, increment*n*

8.5.15.1 Semantics

Pattern is the symbol for a linear pattern.

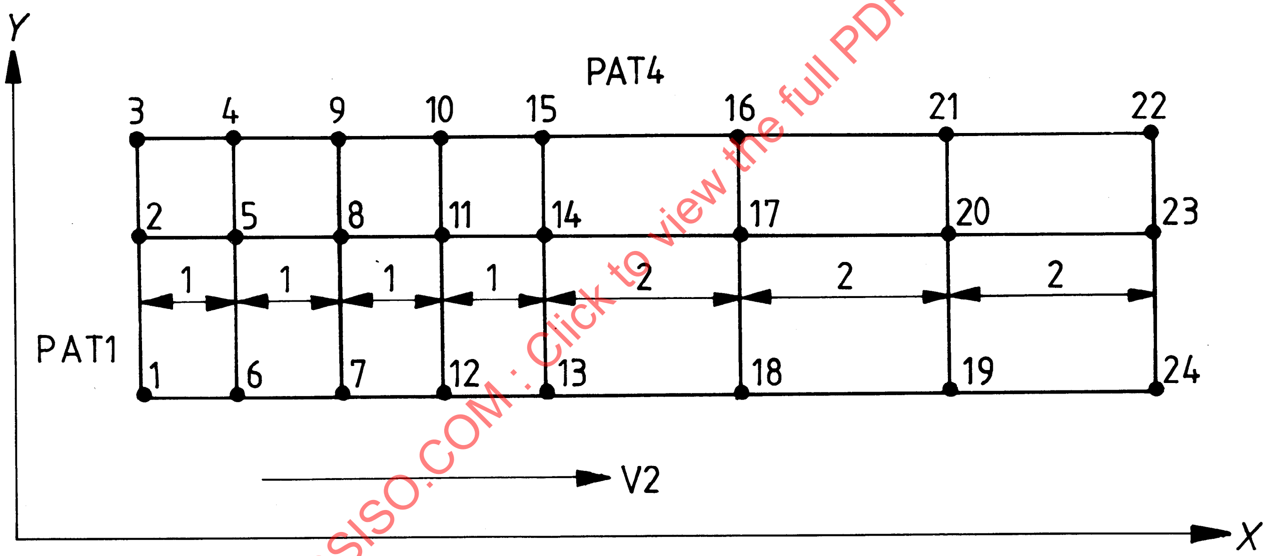
Vector is the symbol for a vector.

INCR specifies that the following values are incremental.

Number1 specifies the number of increments and increment1 the required incremental value, number*n* specifies the number of further increments and increment*n* their incremental value.

The pattern is formed in such a way that an increment is measured in the direction of the vector from every point of the pattern. There then exists a second row of points that is parallel to that described in the pattern. This is done once for every given increment. The locations in the resulting pattern have a sequence which is determined by the order of the locations in the first pattern used in the GRID pattern. The first row is in the order of the first pattern, the second row is in the reverse order, etc.

8.5.15.2 Example



PAT4 = PATERN/GRID, PAT1, V2, INCR, 4, AT, 1, 3, AT, 2

Figure 35

8.5.15.3 Limitations

The pattern shall be a linear pattern.

8.5.15.4 Syntax

< pattern parameter list > : : = GRID, < linear pattern spec > , < vector spec > , INCR, < pattern data >

8.5.16 Definition of a pattern by two patterns

PATERN/TRAFO, pattern1, pattern2, XYROT, angle

8.5.16.1 Semantics

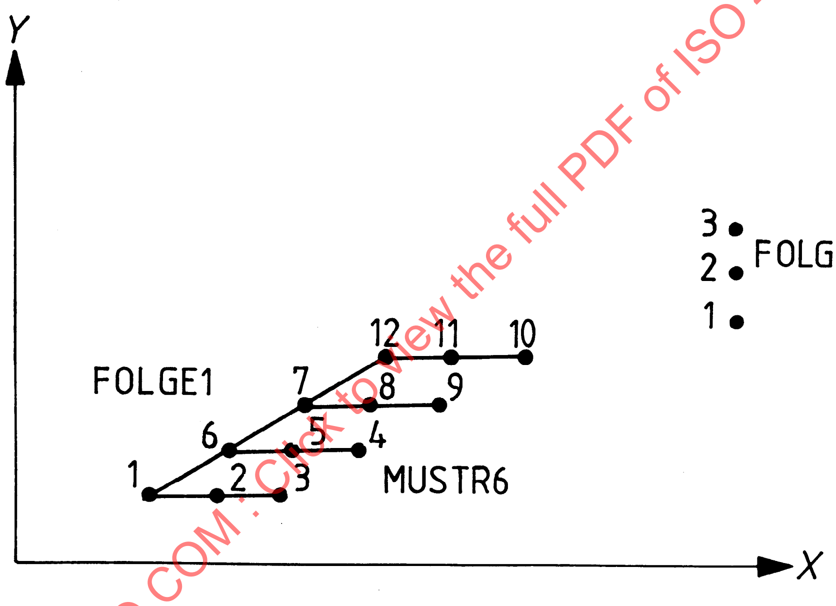
TRAFO specifies that the pattern of points is formed by transforming the second pattern in such a way that it is started at every point of the first pattern. The result is a parallelogram.

Pattern1 and pattern2 are the symbols for two linear patterns.

Angle specifies the angle by which pattern2 is to be rotated within the XY-plane before being connected with pattern1.

The locations in the resulting pattern have a sequence which is determined by the order of the locations in the second pattern used in the TRAFO pattern. The first row is in the order of the second pattern, the second row is in the reverse order, etc.

8.5.16.2 Example



MUSTR6 = PATERN/TRAFO, FOLGE1, FOLGE2, XYROT, – 90

Figure 36

8.5.16.3 Limitations

Pattern1 and pattern2 shall be linear patterns.

8.5.16.4 Syntax

< pattern parameter list > : : = TRAFO, < linear pattern spec > , < linear pattern spec > $^{1}_{0}$ [,XYROT, < scalar >]

8.5.17 Definition of a random pattern by points and patterns

PATERN/RANDOM, patpoint1, . . ., patpoint*n*

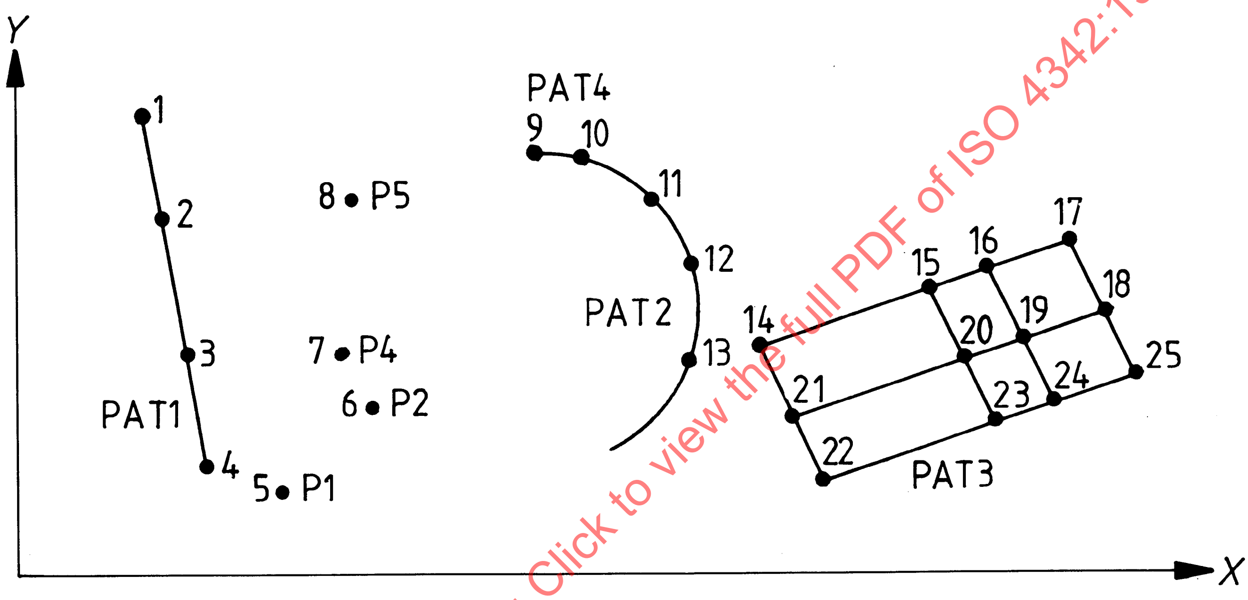
8.5.17.1 Semantics

RANDOM specifies that the pattern is formed by the list of patpoints in the order in which they are specified.

Patpoint1 specifies either a symbol for a point or a symbol for a pattern. Patpoint*n* specifies further symbols of points or patterns.

The sequence in the resulting pattern depends on the ordering in the statement.

8.5.17.2 Example



PAT4 = PATERN/RANDOM, PAT1, P1, P2, P4, P5, PAT2, PAT3

Figure 37

8.5.17.3 Limitations

None.

8.5.17.4 Syntax

< pattern parameter list > : : = RANDOM, [< point spec > | < pattern spec >] $_0^n$ [, < point spec > | < pattern spec >]

8.5.18 Definition of a pattern by a pattern and a mirroring line

PATERN/MIRROR, line, pattern

8.5.18.1 Semantics

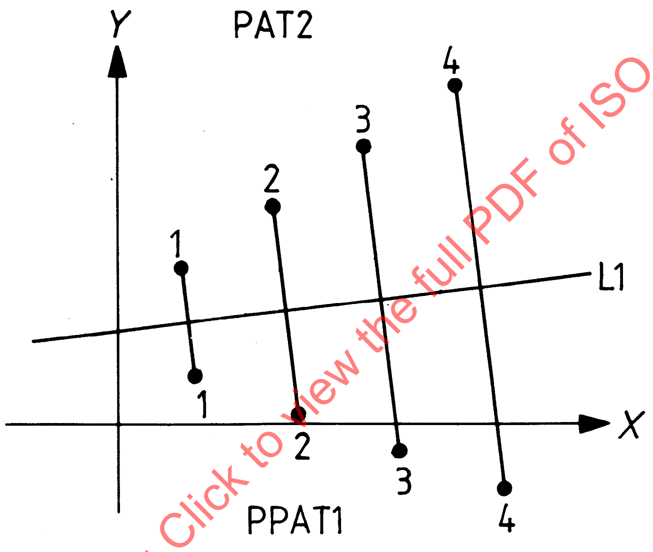
MIRROR specifies that the pattern is formed by mirroring the specified pattern about the specified line.

Line is the symbol for a line, about which the pattern is mirrored.

Pattern is the symbol for a pattern.

The sequence numbers remain in their original order.

8.5.18.2 Example



PPAT1 = PATERN/MIRROR, L1, PAT2

Figure 38

8.5.18.3 Limitations

None.

8.5.18.4 Syntax

< pattern parameter list > : : = MIRROR, < line spec > , < pattern spec >

8.5.19 Definition of a pattern by a linear pattern and a circular pattern

PATERN/CIRCUL, pattern1, pattern2

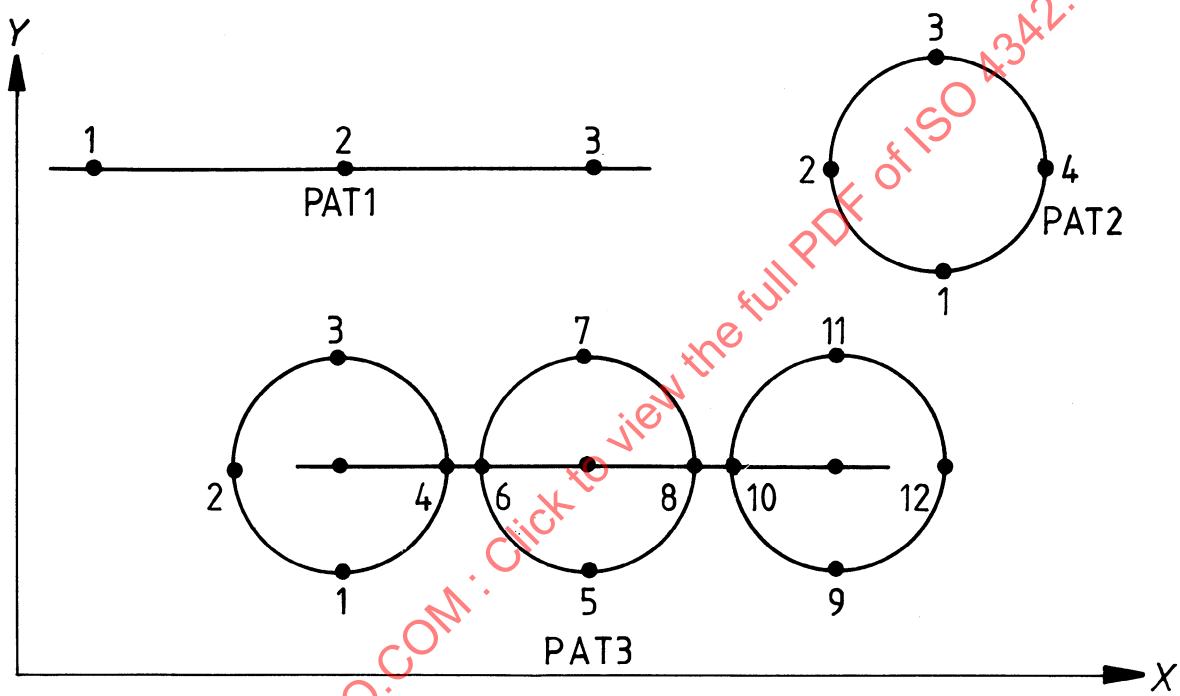
8.5.19.1 Semantics

CIRCUL specifies that the resulting pattern is formed in such a way that a circular pattern is placed with its centre point on every location of pattern1. The points of pattern1 do not appear in the resulting pattern.

Pattern1 is the symbol for a linear pattern.

Pattern2 is the symbol for a circular pattern.

8.5.19.2 Example



PAT3 = PATERN/CIRCUL, PAT1, PAT2

Figure 39

8.5.19.3 Limitations

Pattern1 shall be a linear pattern.

Pattern2 shall be a circular pattern.

8.5.19.4 Syntax

< pattern parameter list > : : = CIRCUL, < linear pattern spec > , < circular pattern spec >

8.6 Definitions of a line

8.6.1 Definition of a line

LINE/line parameter list

8.6.1.1 Semantics

The result of processing a line definition is a canonical form representing the projection of the line on the XY-plane.

8.6.1.2 Limitations

A line shall not be perpendicular to the XY-plane.

8.6.1.3 Sub-contents

For the definition of a line

a) by two points, see 8.6.2;

b) as a coordinate axis, see 8.6.3;

c) by a point and a parallel line, see 8.6.4;

d) by a point and a perpendicular line, see 8.6.5;

e) by a parallel line at a given distance, see 8.6.6;

f) by a point and the angle between this line and another one, see 8.6.7;

g) by a point and its slope relative to a line, see 8.6.8;

h) by its intersection with the named coordinate axis and its angle with the X-axis, see 8.6.9;

j) by its intersection with the named coordinate axis, and the slope of its angle with the X-axis, see 8.6.10;

k) by a point and as a tangent to a circle, see 8.6.11;

m) as a tangent to two circles, see 8.6.12;

n) by its angle with a line, and as a tangent to a circle, see 8.6.13;

p) as an intersection of two planes, see 8.6.14;

q) by a point, and as a tangent to a conic, see 8.6.15;

r) by a point and perpendicular to a conic at this point, see 8.6.16;

s) by a point and as a tangent to a tabulated cylinder, see 8.6.17;

t) by a point and perpendicular to a tabulated cylinder, see 8.6.18;

u) by a point and perpendicular to a conic, see 8.6.19.

8.6.1.4 Syntax

<line definition statement> : : = <identifier> = LINE/ <line parameter list>

8.6.2 Definition of a line by two points

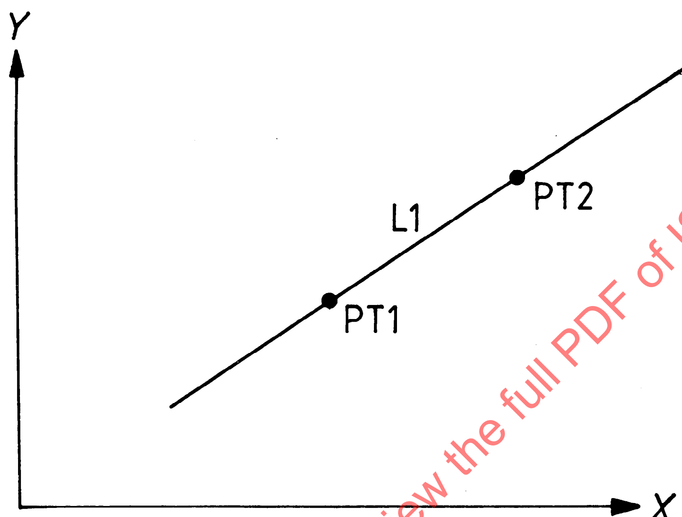
LINE/point1, point2

8.6.2.1 Semantics

Point1 and point2 are the symbols of two points.

The *x*, *y* or *x*, *y*, *z* coordinates of both points may be specified instead of a point.

8.6.2.2 Example



L1 = LINE/PT1, PT2

Figure 40

8.6.2.3 Limitations

The two points shall not have the same *x*- and *y*-coordinates (see 8.3.3). Where the lines are defined by the coordinates of the two points, the *z*-values (if given) are ignored.

8.6.2.4 Syntax

< line parameter list > : : = [< point spec > , < point spec >] | [< scalar > , < scalar > , < scalar > , < scalar >] $_0^1$ [, < scalar > , < scalar >]

8.6.3 Definition of a line as a coordinate axe

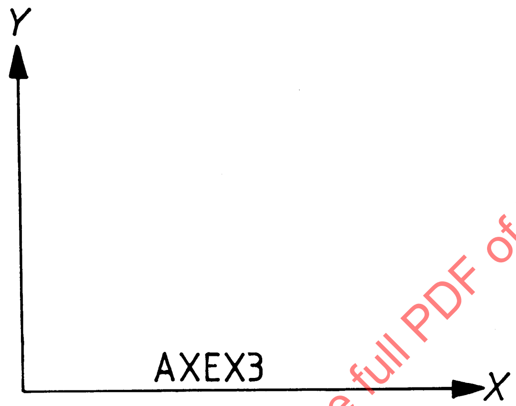
LINE/ XAXIS
YAXIS

8.6.3.1 Semantics

XAXIS specifies the X-axis.

YAXIS specifies the Y-axis.

8.6.3.2 Example



AXEX3 = LINE/XAXIS

Figure 41

8.6.3.3 Limitations

ZAXIS cannot be used.

8.6.3.4 Syntax

<line parameter list> : : = XAXIS | YAXIS

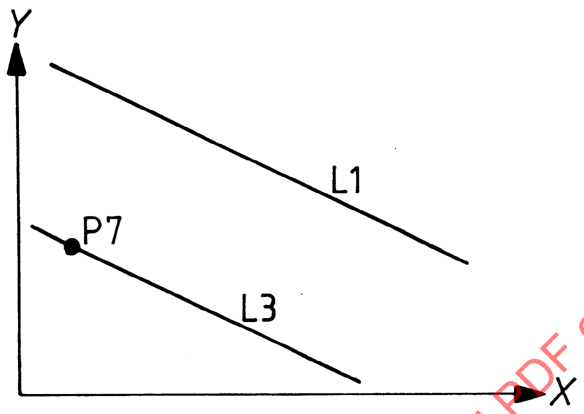
8.6.4 Definition of a line by a point and a parallel line

LINE/point, PARLEL, line

8.6.4.1 Semantics

Point and line are the symbols for a point and a line.

8.6.4.2 Example



L3 = LINE/P7, PARLEL, L1

Figure 42

8.6.4.3 Limitations

None.

8.6.4.4 Syntax

< line parameter list > : : = < point spec > , PARLEL, < line spec >

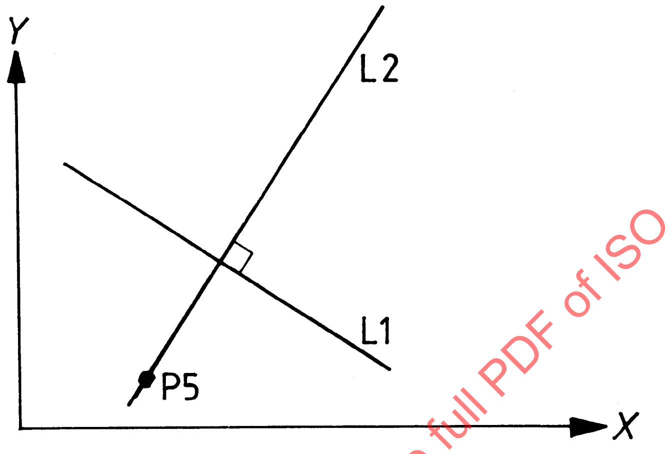
8.6.5 Definition of a line by a point and a perpendicular line

LINE/point, PERPTO, line

8.6.5.1 Semantics

Point and line are the symbols for a point and a line.

8.6.5.2 Example



L2 = LINE/P5, PERPTO, L1

Figure 43

8.6.5.3 Limitations

None.

8.6.5.4 Syntax

< line parameter list > : : = < point spec > , PERPTO, < line spec >

8.6.6 Definition of a line by a parallel line at a given distance

LINE/PARLEL, line, XLARGE / XSMALL / YLARGE / YSMALL, dist

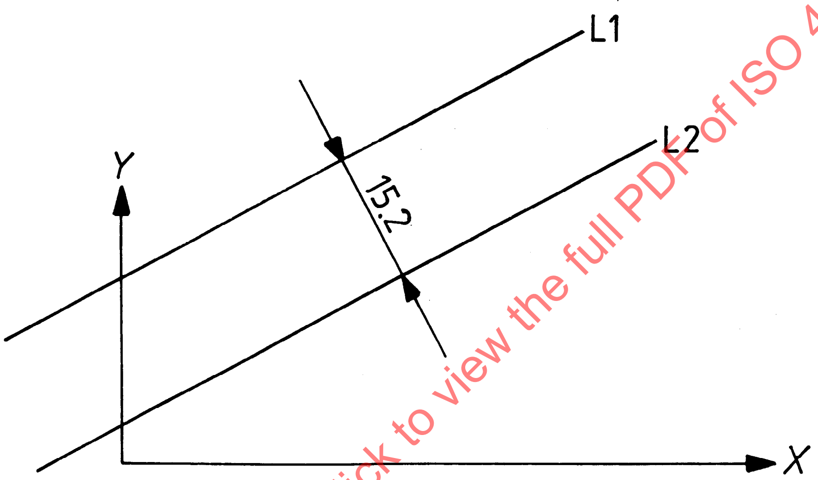
8.6.6.1 Semantics

Line is the symbol for a line.

The modifiers specify the sense of the algebraic displacement on a perpendicular to the given line.

Dist specifies the distance between the two lines.

8.6.6.2 Example



L2 = LINE/PARLEL, L1, XLARGE, 15.2

Figure 44

8.6.6.3 Limitations

When the given line is perpendicular to a coordinate axis, only the corresponding modifiers shall be used. The value of the distance shall not be negative.

8.6.6.4 Syntax

< line parameter list > : : = PARLEL, < line spec > ,[XLARGE | XSMALL | YLARGE | YSMALL], < scalar >

8.6.7 Definition of a line by a point and the angle between this line and another one

```
                              XAXIS
LINE/point, ATANGL, angle,    YAXIS
                              line
```

8.6.7.1 Semantics

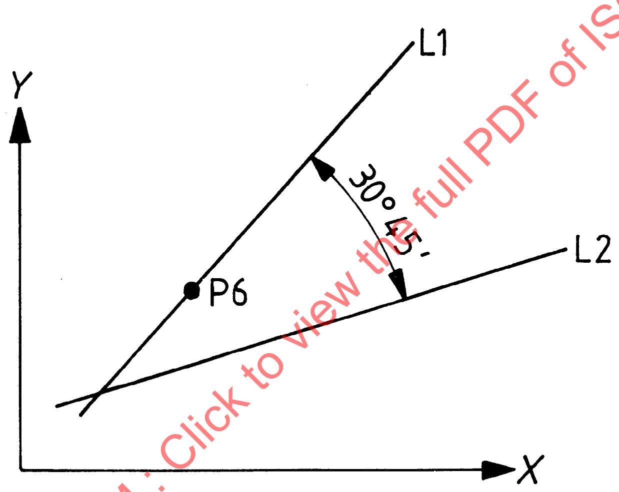
Point and line are the symbols for a point and a line.

ATANGL, angle specifies the angle from the referenced line to the line being defined; when the referenced line is not specified, the *X*-axis is assumed.

XAXIS specifies the *X*-axis as the referenced line.

YAXIS specifies the *Y*-axis as the referenced line.

8.6.7.2 Example



L1 = LINE/P6, ATANGL, 30.75, L2

Figure 45

8.6.7.3 Limitations

None.

8.6.7.4 Syntax

< line parameter list > : : = < point spec > , ATANGL, < scalar > $_0^1$ [,XAXIS | ,YAXIS |, < line spec >]

8.6.8 Definition of a line by a point and its slope relative to a line

LINE/point, SLOPE, slope, XAXIS / YAXIS / line

8.6.8.1 Semantics

Point and line are the symbols for a point and a line.

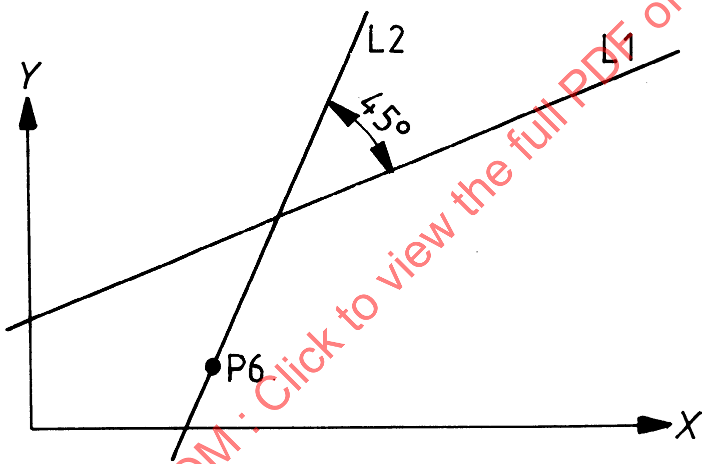
SLOPE, slope specifies the tangent of the angle from the referenced line to the line being defined.

XAXIS specifies the X-axis as the reference line from which the slope is measured.

YAXIS specifies the Y-axis as the reference line from which the slope is measured.

When the referenced line is not specified, the X-axis is assumed.

8.6.8.2 Example



L2 = LINE/P6, SLOPE, 1, L1

Figure 46

8.6.8.3 Limitations

A line perpendicular to the referenced line cannot be defined in this way.

8.6.8.4 Syntax

< line parameter list > : : = < point spec > , SLOPE, < scalar > $^1_0$ [,XAXIS | ,YAXIS |, < line spec >]

8.6.9 Definition of a line by its intersection with the named coordinate axis and its angle with the *X*-axis

LINE/ATANGL, angle, INTERC, XAXIS / YAXIS , intercept

8.6.9.1 Semantics

ATANGL, angle specifies the angle in degrees from the *X*-axis to the line.

INTERC specifies that the following value is the coordinate of the intersection point on the named coordinate axis.

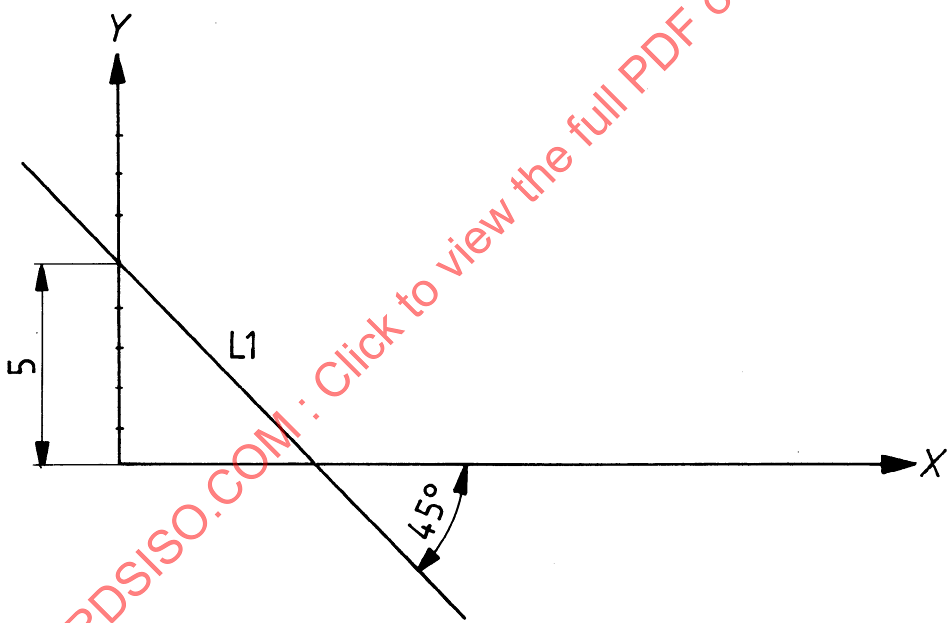
XAXIS specifies that the *X*-axis is the referenced axis.

YAXIS specifies that the *Y*-axis is the referenced axis.

Intercept specifies the coordinate of the intersection point.

When the named coordinate axis is not mentioned, the *Y*-axis is assumed.

8.6.9.2 Example



L1 = LINE/ATANGL, − 45, INTERC, YAXIS, 5

Figure 47

8.6.9.3 Limitations

None.

8.6.9.4 Syntax

< line parameter list > : : = ATANGL, < scalar > , INTERC $^{1}_{0}$ [,XAXIS | ,YAXIS], < scalar >

8.6.10 Definition of a line by its intersection with the named coordinate axis and the slope of its angle with the *X*-axis

LINE/SLOPE, slope, INTERC, XAXIS / YAXIS , intercept

8.6.10.1 Semantics

SLOPE, slope specifies the tangent of the angle from the *X*-axis to the line.

INTERC specifies that the following value is the coordinate of the intersection point on the named coordinate axis.

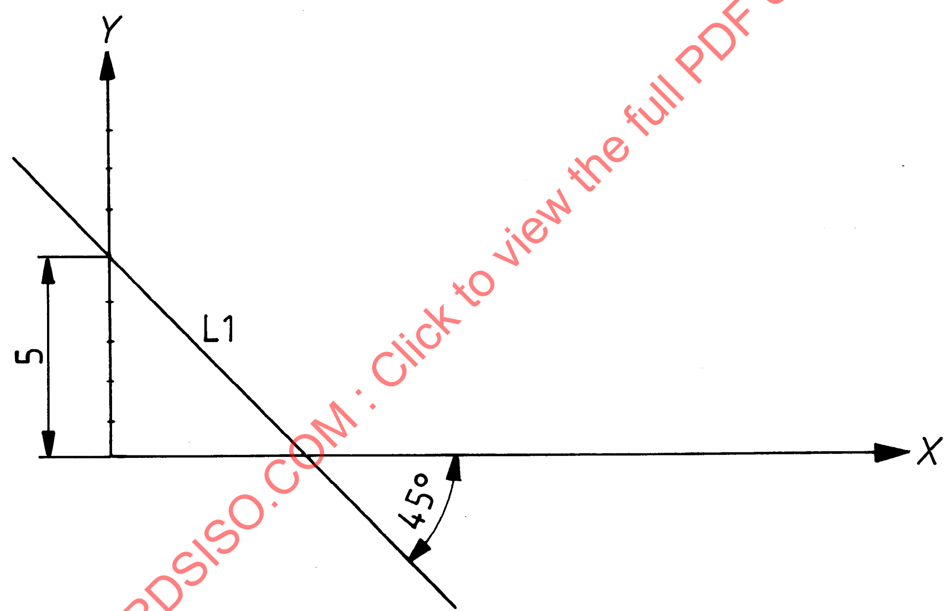
XAXIS specifies that the *X*-axis is the referenced axis.

YAXIS specifies that the *Y*-axis is the referenced axis.

Intercept specifies the coordinate value along the named coordinate axis.

When the named coordinate axis is not mentioned, the *Y*-axis is assumed.

8.6.10.2 Example



L1 = LINE/SLOPE, − 1, INTERC, YAXIS, 5

Figure 48

8.6.10.3 Limitations

A line perpendicular to the *X*-axis cannot be defined in this way.

8.6.10.4 Syntax

< line parameter list > : : = SLOPE, < scalar > , INTERC $^1_0$ [,XAXIS | ,YAXIS], < scalar >

8.6.11 Definition of a line by a point and as a tangent to a circle

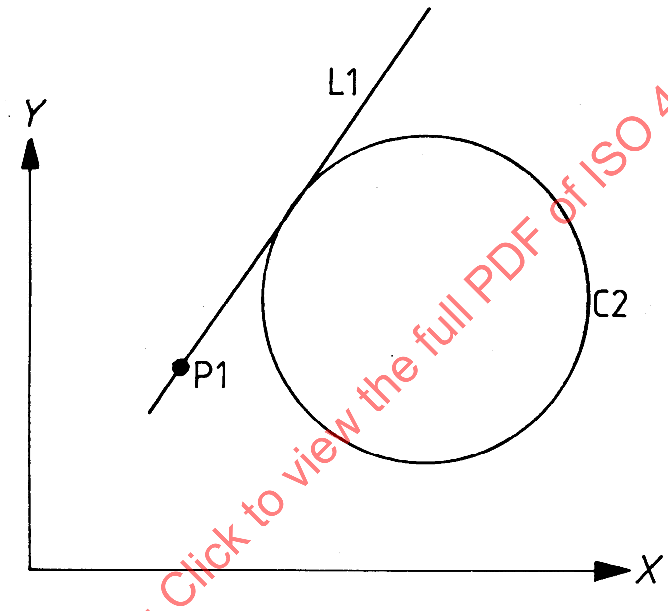
LINE, point, RIGHT / LEFT , TANTO, circle

8.6.11.1 Semantics

Point and circle are the symbols for a point and a circle.

The line can pass through two possible tangent points, either to the right or left of the centre of the circle as observed from the given point in the direction of the centre of the circle.

8.6.11.2 Example



L1 = LINE/P1, LEFT, TANTO, C2

Figure 49

8.6.11.3 Limitations

The point shall not be inside the circle.

8.6.11.4 Syntax

< line parameter list > : : = < point spec > ,[LEFT | RIGHT], TANTO, < circle spec >

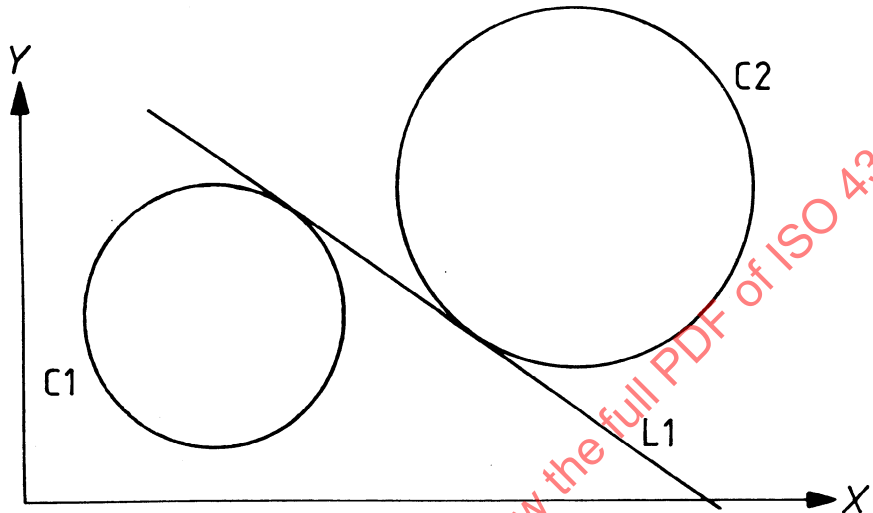
8.6.12 Definition of a line as a tangent of two circles

LINE/ $\begin{matrix}\text{RIGHT}\\\text{LEFT}\end{matrix}$, TANTO, circle1, $\begin{matrix}\text{RIGHT}\\\text{LEFT}\end{matrix}$, TANTO, circle2

8.6.12.1 Semantics

Circle1 and circle2 are the symbols for two circles. The line is tangential to both circles and can pass either to the right or to the left of each circle as observed from the centre of the first mentioned circle in the direction of the second circle.

8.6.12.1 Example



L1 = LINE/LEFT, TANTO, C1, RIGHT, TANTO, C2

Figure 50

8.6.12.3 Limitations

One circle shall not be completely inside the other and if the two circles intersect, the same RIGHT or LEFT modifier shall be used.

8.6.12.4 Syntax

< line parameter list > : : = [RIGHT | LEFT], TANTO, < circle spec > ,[RIGHT | LEFT], TANTO, < circle spec >

8.6.13 Definition of a line by its angle with a line, and as tangent to a circle

LINE/ATANGL, angle, line, {XLARGE | XSMALL | YLARGE | YSMALL}, TANTO, circle

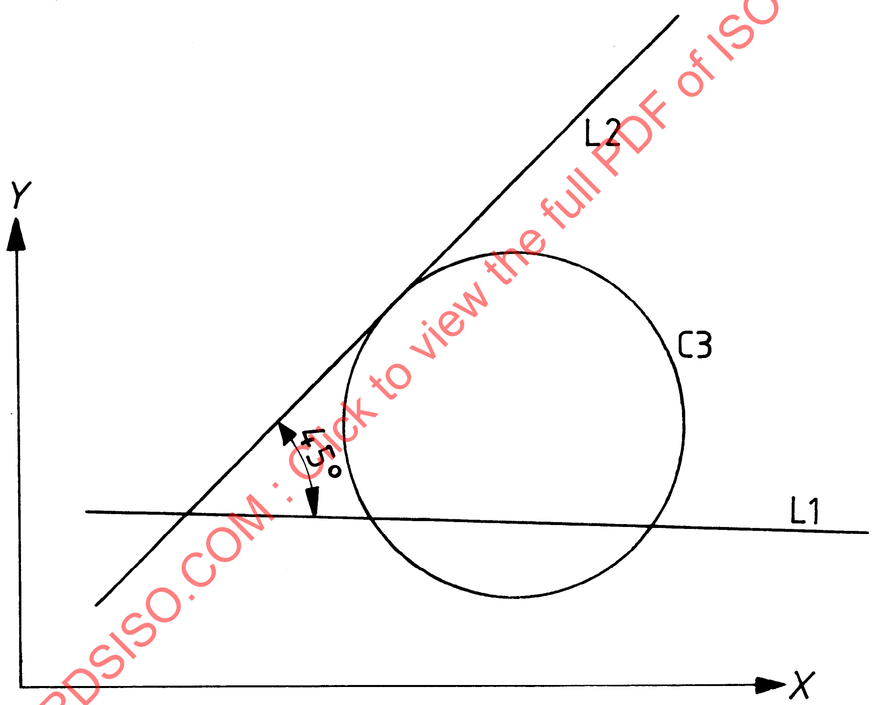
8.6.13.1 Semantics

Line and circle are the symbols for a line and a circle.

ATANGL, angle specifies the angle from the referenced line to the line being defined.

The modifiers specify the desired contact point with respect to the relative position of the projections of the possible points on the mentioned axis.

8.6.13.2 Example



L2 = LINE/ATANGL, 45, L1, YLARGE, TANTO, C3

Figure 51

8.6.13.3 Limitations

When the line being defined is perpendicular to a coordinate axis only the associated modifiers shall be used; for example, when perpendicular to the *X*-axis, only XLARGE or XSMALL shall be used.

8.6.13.4 Syntax

< line parameter list > : : = ATANGL, < scalar > , < line spec > ,[XLARGE | XSMALL | YLARGE | YSMALL],
TANTO, < circle spec >

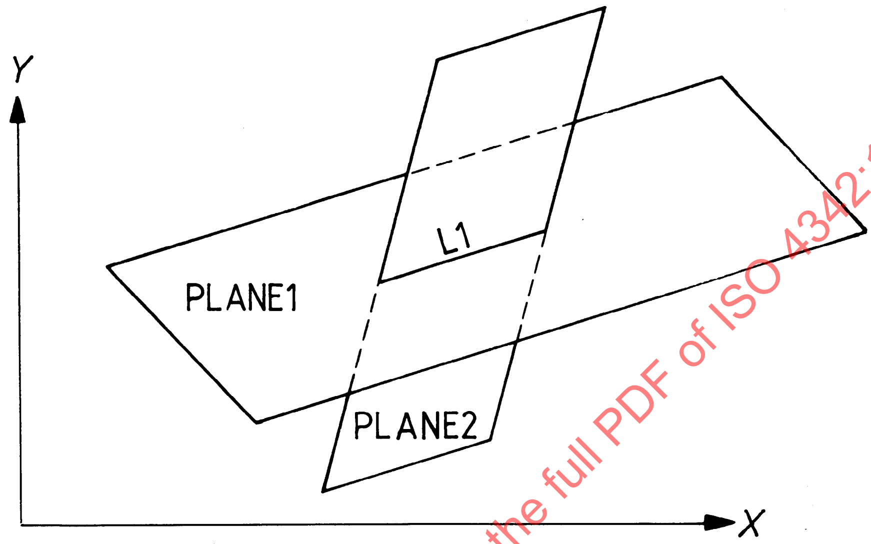
8.6.14 Definition of a line as an intersection of two planes

LINE/INTOF, plane1, plane2

8.6.14.1 Semantics

Plane1 and plane2 are the symbols for two planes.

8.6.14.2 Example



L1 = LINE/INTOF, PLANE1, PLANE2

Figure 52

8.6.14.3 Limitations

The two planes shall not be parallel; only one of the planes may be perpendicular to the XY-plane.

8.6.14.4 Syntax

< line parameter list > : : = INTOF, < plane spec > , < plane spec >

8.6.15 Definition of a line by a point and as a tangent to a conic

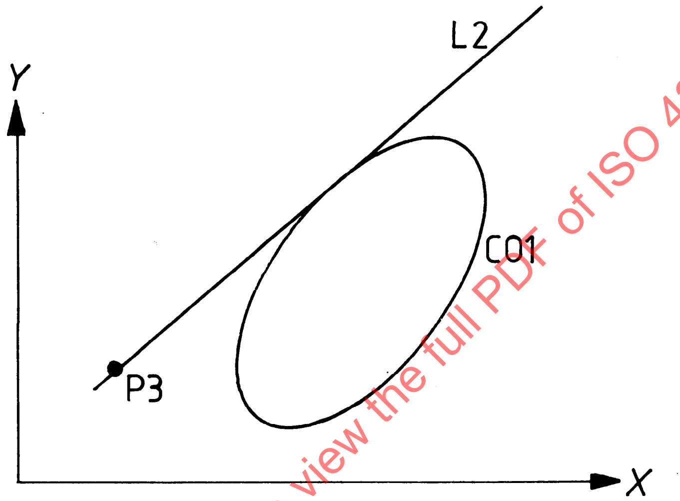
LINE/point, LEFT
RIGHT , TANTO, conic

8.6.15.1 Semantics

Point and conic are the symbols for a point and a conic.

The line can pass either to the right or to the left of the conic, when observed from the point in the direction of the focus of the conic.

8.6.15.2 Example



L2 = LINE/P3, LEFT, TANTO, CO1

Figure 53

8.6.15.3 Limitations

The point and the focus shall be on opposite sides of the conic.

8.6.15.4 Syntax

< line parameter list > : : = < point spec > ,[LEFT | RIGHT], TANTO, < conic spec >

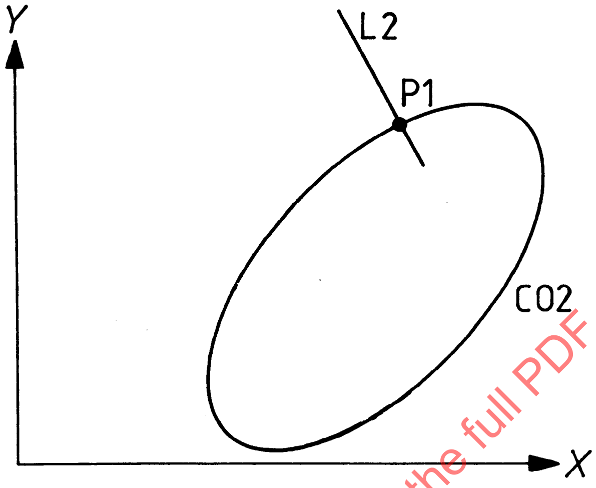
8.6.16 Definition of a line by a point, and perpendicular to a conic at this point

LINE/point, PERPTO, conic

8.6.16.1 Semantics

Point and conic are the symbols for a point and a conic; the point shall lie on the conic.

8.6.16.2 Example



L2 = LINE/P1, PERPTO, CO2

Figure 54

8.6.16.3 Limitations

None.

8.6.16.4 Syntax

< line parameter list > : : = < point spec > , PERPTO, < conic spec >

8.6.17 Definition of a line by a point and as a tangent to a tabulated cylinder

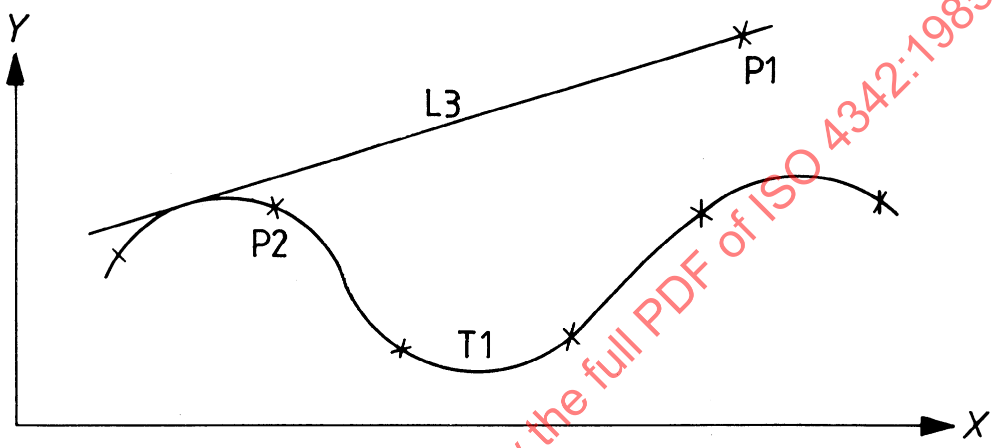
LINE/point1, TANTO, tabcyl, point2

8.6.17.1 Semantics

Point1 and tabcyl are the symbols for a point and a tabulated cylinder perpendicular to the XY-plane.

Point2 is the symbol for a point near the desired point of tangency; it allows selection of the segment of the tabulated cylinder to which the line must be tangent.

8.6.17.2 Example



L3 = LINE/P1, TANTO, T1, P2

Figure 55

8.6.17.3 Limitations

Multiple possibilities are not allowed for a single tabulated cylinder segment selected with respect to the second point.

8.6.17.4 Syntax

< line parameter list > : : = < point spec > , TANTO, < tabulated cylinder spec > , < point spec >

8.6.17.5 Cross-reference (see annex D)

TABCYL

8.6.18 Definition of a line by a point, and perpendicular to a tabulated cylinder

LINE/point1, PERPTO, tabcyl, point2

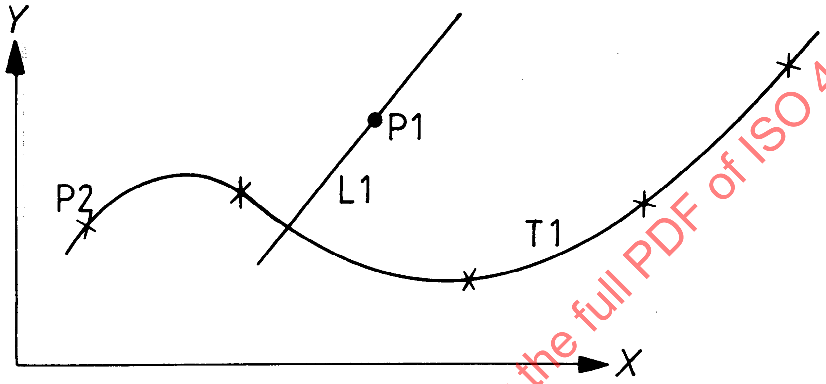
8.6.18.1 Semantics

Point1 and tabcyl are the symbols for a point and a tabulated cylinder perpendicular to the XY-plane.

Point2 is the symbol for a point near the intersection point of the desired line with the tabulated cylinder.

This nearby point allows selection of the segment of the tabulated cylinder to which the line must be normal.

8.6.18.2 Example



L1 = LINE/P1, PERPTO, T1, P2

Figure 56

8.6.18.3 Limitations

Multiple possibilities are not allowed for a single tabulated cylinder segment selected with respect to the second point.

8.6.18.4 Syntax

< line parameter list > : : = < point spec > , PERPTO, < tabulated cylinder spec > , < point spec >

8.6.19 Definition of a line by a point and perpendicular to a conic

LINE/point1, PERPTO, conic, point2, point3

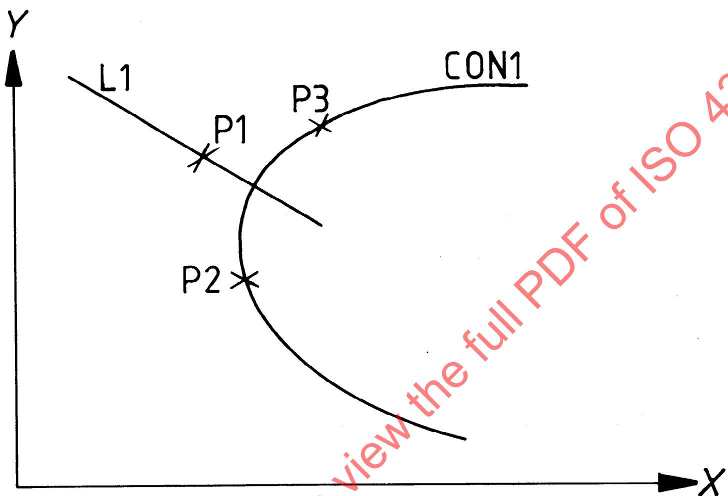
8.6.19.1 Semantics

Point1 is the symbol for a point through which the line passes.

Point2 and point3 are points on the conic between which the required line will pass.

Conic is the symbol for a conic.

8.6.19.2 Example



L1 = LINE/P1, PERPTO, CON1, P2, P3

Figure 57

8.6.19.3 Limitations

None.

8.6.19.4 Syntax

< line parameter list > : : = < point spec > , PERPTO, < conic spec > , < point spec > , < point spec >

8.7 Definitions of a plane

8.7.1 Definition of a plane

PLANE/plane parameter list

8.7.1.1 Semantics

A plane is a surface which contains all points of a straight line joining any two points on the surface.

8.7.1.2 Limitations

None.

8.7.1.3 Sub-contents

For the definition of a plane

1) by the coefficients of the plane equation $ax + by + cz - d = 0$, see 8.7.2;
2) passing through three non-collinear points, see 8.7.3;
3) passing through a point and parallel to a given plane, see 8.7.4;
4) parallel to a given plane at a given distance, see 8.7.5;
5) passing through a point and perpendicular to a vector, see 8.7.6;
6) passing through two points and perpendicular to a given plane, see 8.7.7;
7) perpendicular to two intersecting planes and passing through a point, see 8.7.8;
8) through a point and tangential to a cylinder, see 8.7.9.

8.7.1.4 Syntax

< plane definition statement > : : = < identifier > = PLANE/ < plane parameter list >

8.7.2 Definition of a plane by the coefficients of the plane equation $ax + by + cz - d = 0$

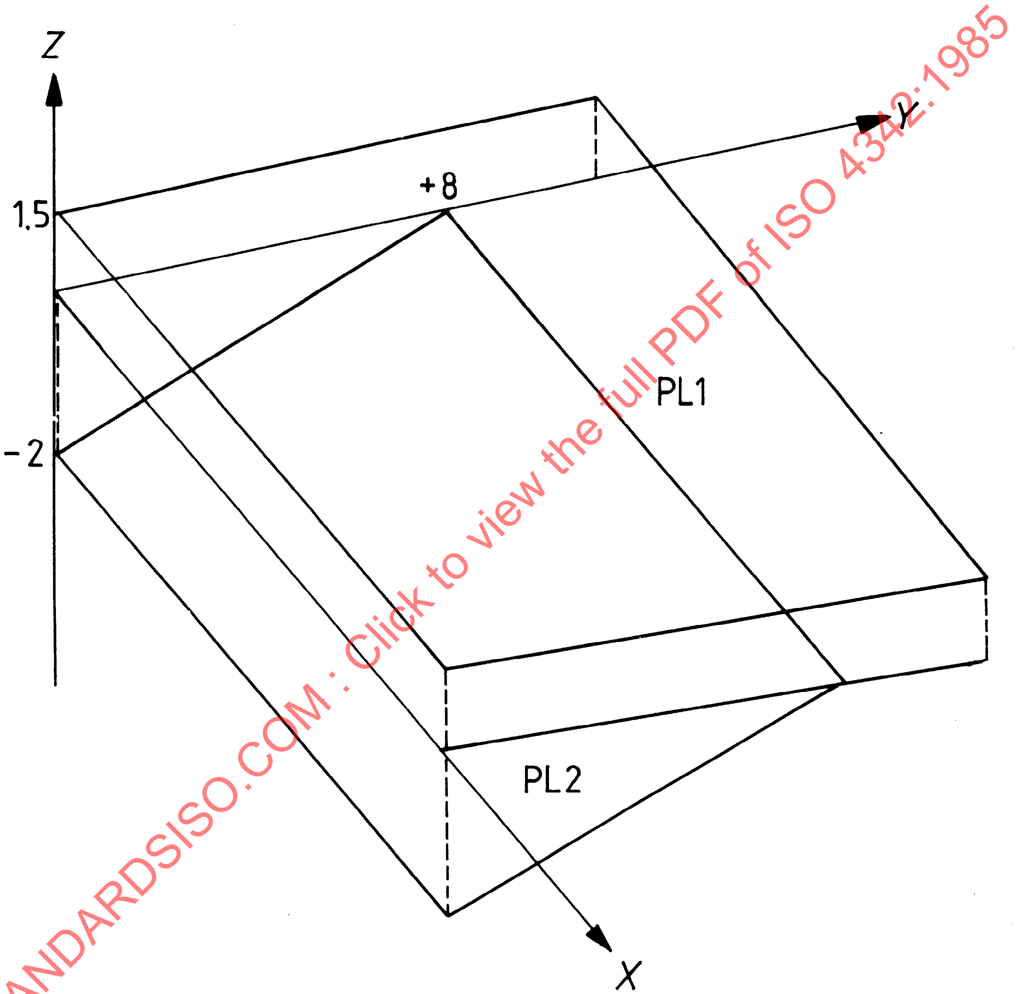
PLANE/a, b, c, d

8.7.2.1 Semantics

a, b, c specify the x, y, z components of the plane normal unit vector.

d specifies the distance of the plane from the part coordinate origin in the direction of the plane normal unit vector.

8.7.2.2 Example



PL1 = PLANE/0, 0, 1, 1.5

PL2 = PLANE/0, – .5, 2, 4

Figure 58

8.7.2.3 Limitations

None.

8.7.2.4 Syntax

< plane parameter list > : : = < scalar > , < scalar > , < scalar > , < scalar >

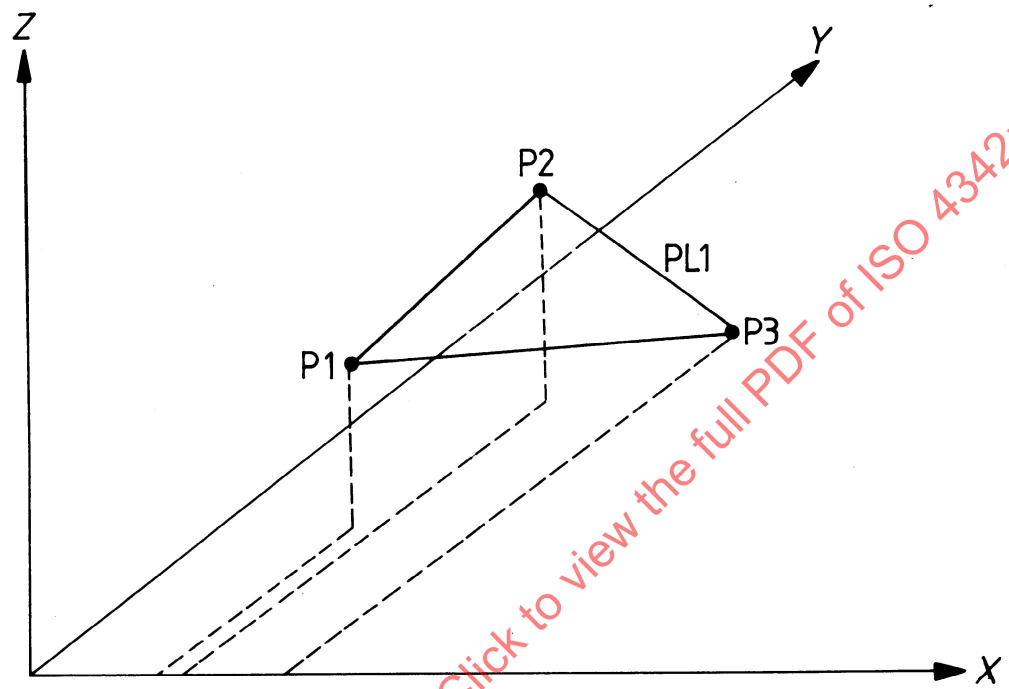
8.7.3 Definition of a plane passing through three non-collinear points

PLANE/point1, point2, point3

8.7.3.1 Semantics

Point1, point2, and point3 are the symbols for three points.

8.7.3.2 Example



PL1 = PLANE/P1, P2, P3

Figure 59

8.7.3.3 Limitations

The three points shall not be collinear or coincident.

8.7.3.4 Syntax

< plane parameter list > : : = < point spec > , < point spec > , < point spec >

8.7.4 Definition of a plane passing through a point and parallel to a given plane

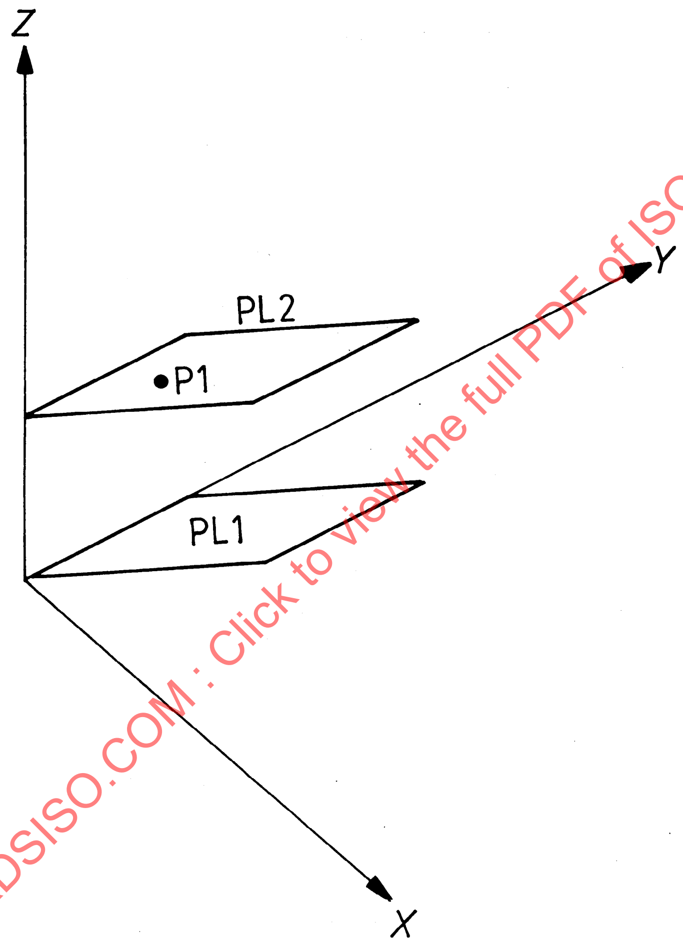
PLANE/point, PARLEL, plane

8.7.4.1 Semantics

Point is the symbol for a point.

Plane is the symbol for a plane.

8.7.4.2 Example



PL2 = PLANE/P1, PARLEL, PL1

Figure 60

8.7.4.3 Limitations

None.

8.7.4.4 Syntax

< plane parameter list > : : = < point spec > , PARLEL, < plane spec >

8.7.5 Definition of a plane parallel to a given plane at a given distance

```
                                XLARGE
                                XSMALL
PLANE/PARLEL, plane,            YLARGE  , distance
                                YSMALL
                                ZLARGE
                                ZSMALL
```

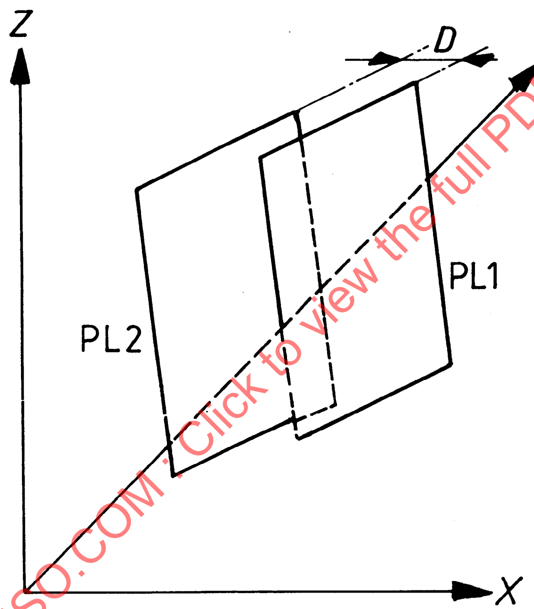
8.7.5.1 Semantics

Plane is the symbol for a plane.

The modifier specifies the sense of the algebraic displacement on the perpendicular to the given plane.

Distance specifies the distance between the two planes.

8.7.5.2 Example



PL1 = PLANE/PARLEL, PL2, XLARGE, D1

Figure 61

8.7.5.3 Limitations

When the plane is parallel to a coordinate axis, only the modifiers associated with the other axes shall be used. The value of the distance shall not be negative.

8.7.5.4 Syntax

< plane parameter list > : : = PARLEL, < plane spec > ,[XLARGE | XSMALL | YLARGE | YSMALL |
ZLARGE | ZSMALL], < scalar >

8.7.6 Definition of a plane passing through a point and perpendicular to a vector

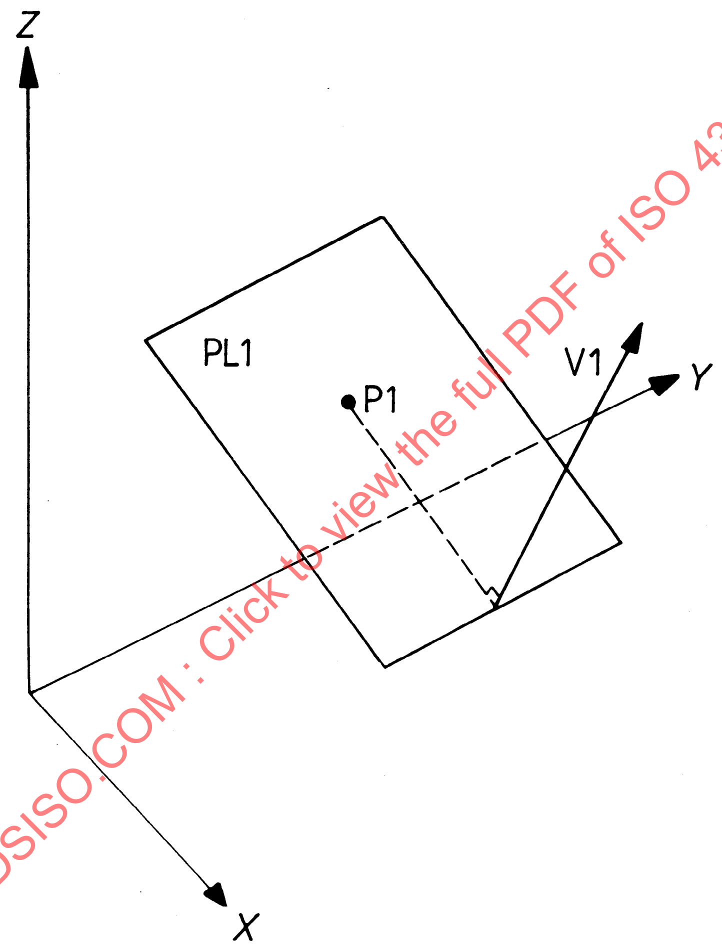
PLANE/point, PERPTO, vector

8.7.6.1 Semantics

Point is the symbol for a point.

Vector is the symbol for a vector.

8.7.6.2 Example



PL1 = PLANE/P1, PERPTO, V1

Figure 62

8.7.6.3 Limitations

None.

8.7.6.4 Syntax

< plane parameter list > : : = < point spec > , PERPTO, < vector spec >

8.7.7 Definition of a plane passing through two points and perpendicular to a given plane

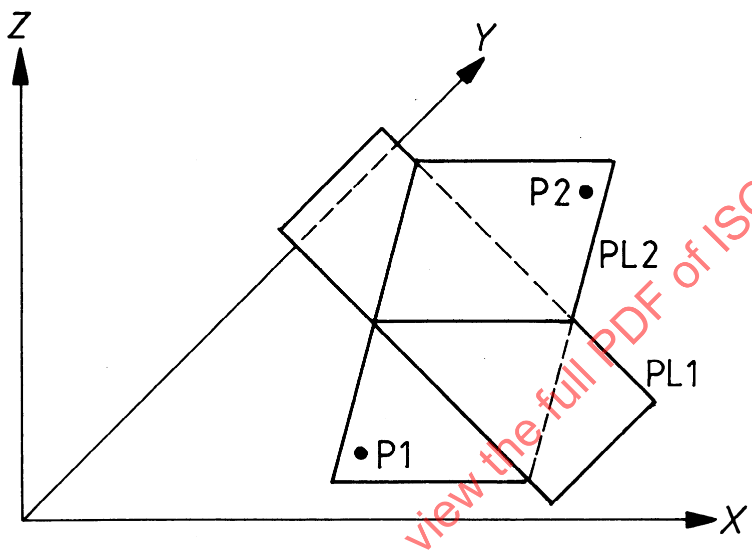
PLANE/point1, point2, PERPTO, plane

8.7.7.1 Semantics

Plane is the symbol for a plane.

Point1 and point2 are the symbols for points.

8.7.7.2 Example



PL2 = PLANE/P1, P2, PERPTO, PL1

Figure 63

8.7.7.3 Limitations

A line joining the two points shall not be perpendicular to the given plane.

8.7.7.4 Syntax

< plane parameter list > : : = < point spec > , < point spec > , PERPTO, < plane spec >

8.7.8 Definition of a plane perpendicular to two intersecting planes and passing through a point

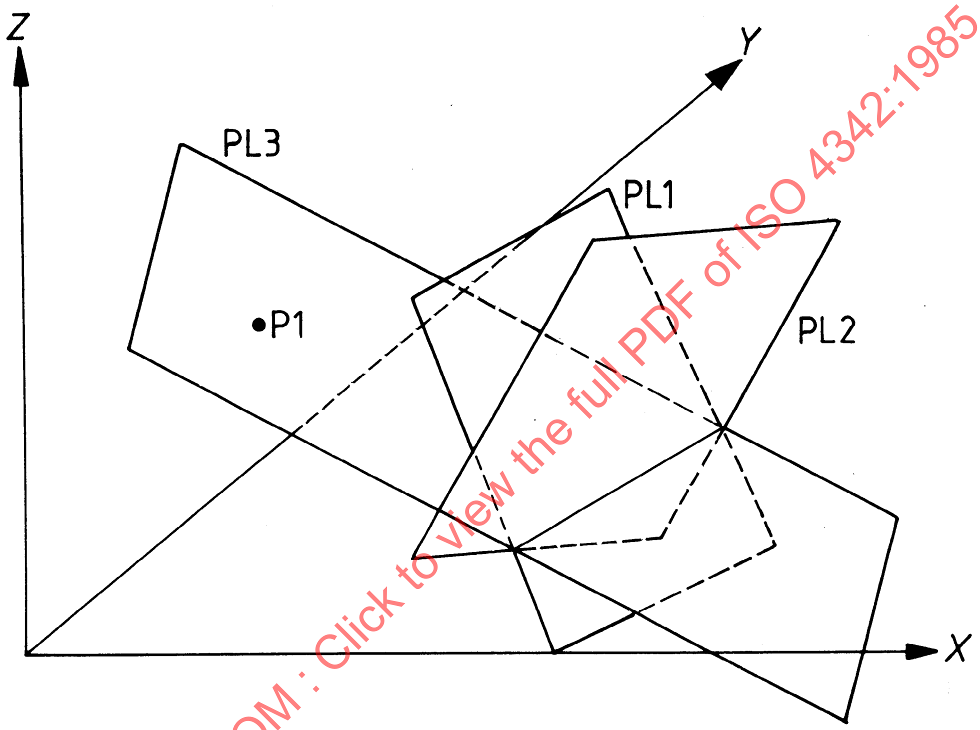
PLANE/point, PERPTO, plane1, plane2

8.7.8.1 Semantics

Point is the symbol for a point.

Plane1 and plane2 are the symbols for two planes.

8.7.8.2 Example



PL3 = PLANE/P1, PERPTO, PL1, PL2

Figure 64

8.7.8.3 Limitations

The two planes shall intersect.

8.7.8.4 Syntax

< plane parameter list > : : = < point spec > , PERPTO, < plane spec > , < plane spec >

8.7.9 Definition of a plane through a point and tangential to a cylinder

PLANE/point, RIGHT / LEFT , TANTO, cylinder

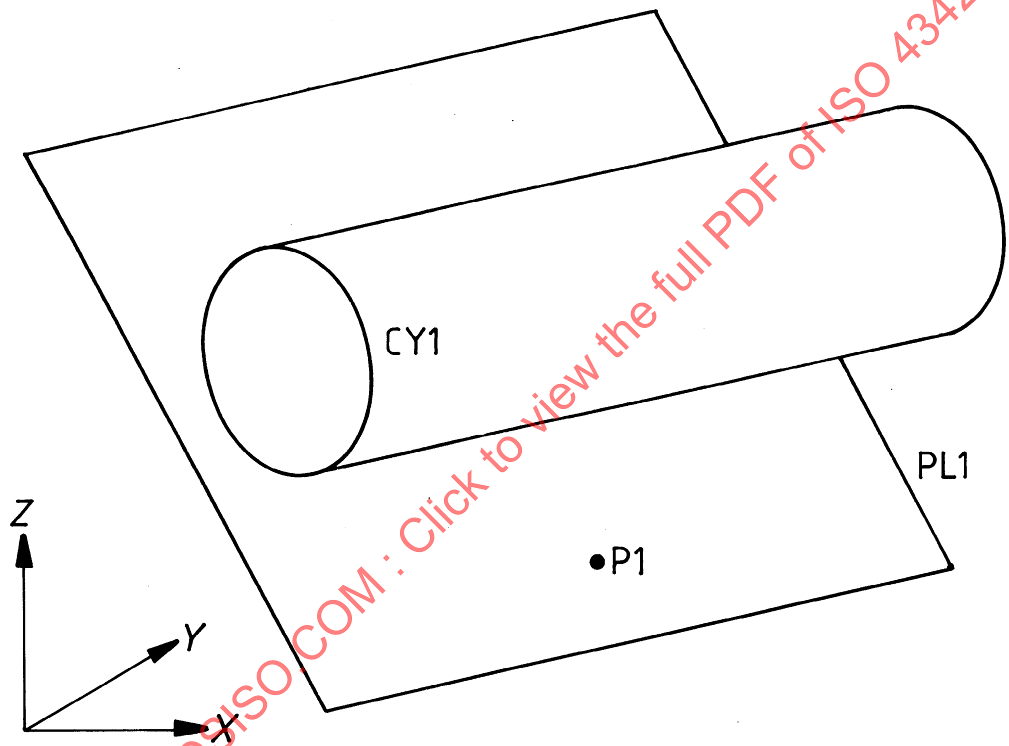
8.7.9.1 Semantics

Point is the symbol for a point.

Cylinder is the symbol for a cylinder.

The plane can be tangential either to the right or left of the longitudinal axis of the cylinder when viewed from the point to the axis of the cylinder in the direction of the cylinder axis vector.

8.7.9.2 Example



PL1 = PLANE/P1, LEFT, TANTO, CY1

Figure 65

8.7.9.3 Limitations

The point shall not lie inside the cylinder.

8.7.9.4 Syntax

< plane parameter list > : : = < point spec > ,[LEFT | RIGHT], TANTO, < cylinder spec >

8.8 Definitions of a vector

8.8.1 Definition of a vector

VECTOR/vector parameter list

8.8.1.1 Semantics

A vector is that quantity which has both magnitude and direction. In cases where a specific magnitude is not defined for a vector, the unit vector (magnitude equal to one) is assumed.

8.8.1.2 Limitations

The vector length shall not be zero.

8.8.1.3 Sub-contents

For the definition of a vector

1) by its components, see 8.8.2;
2) by two points, see 8.8.3;
3) perpendicular to a given plane, see 8.8.4;
4) by a scalar times a vector, see 8.8.5;
5) by the cross product of two given vectors, see 8.8.6;
6) by normalizing a given vector, a given point, or given components, see 8.8.7;
7) by a length and a given angle in a given plane, see 8.8.8;
8) parallel to the intersection of two given planes, see 8.8.9;
9) by the addition or subtraction of two given vectors, see 8.8.10;
10) in the XY-plane making a specified angle with a given line, see 8.8.11.

8.8.1.4 Syntax

< vector definition statement > : : = < identifier > = VECTOR/ < vector parameter list >

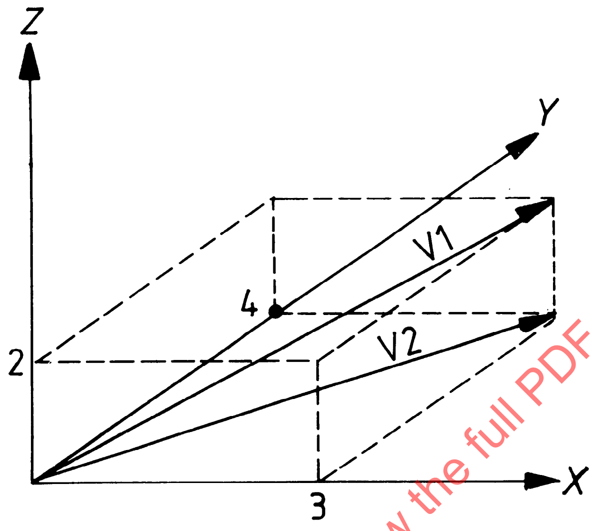
8.8.2 Definition of a vector by its components

VECTOR/x, y, z

8.8.2.1 Semantics

x, y, z represent the components along the X, Y and Z axes respectively.

8.8.2.2 Example



V1 = VECTOR/3, 4, 2

V2 = VECTOR/3, 4, 0

Figure 66

8.8.2.3 Limitations

None.

8.8.2.4 Syntax

< vector parameter list > : : = < scalar > , < scalar > , < scalar >

8.8.3 Definition of a vector by two points

VECTOR/point1, point2

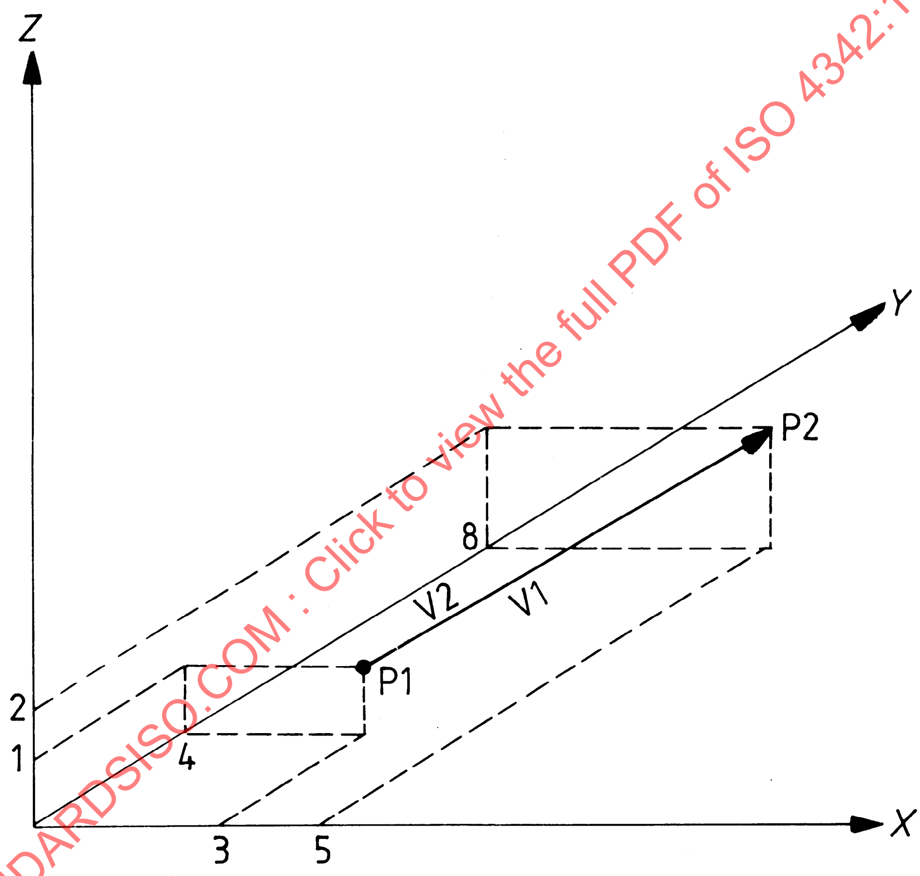
8.8.3.1 Semantics

Point1 is the symbol for the origin of the vector.

Point2 is the symbol for the end of the vector.

The x, y, z coordinates of both points may be specified.

8.8.3.2 Example



V1 = VECTOR/3, 4, 1, 5, 8, 2

V2 = VECTOR/P1, P2

Figure 67

8.8.3.3 Limitations

None.

8.8.3.4 Syntax

< vector parameter list > : : = < scalar > , < scalar > , < scalar > , < scalar > , < scalar > , < scalar >

< vector parameter list > : : = < point spec > , < point spec >

8.8.4 Definition of a vector perpendicular to a given plane

VECTOR/PERPTO, plane,	POSX POSY POSZ NEGX NEGY NEGZ

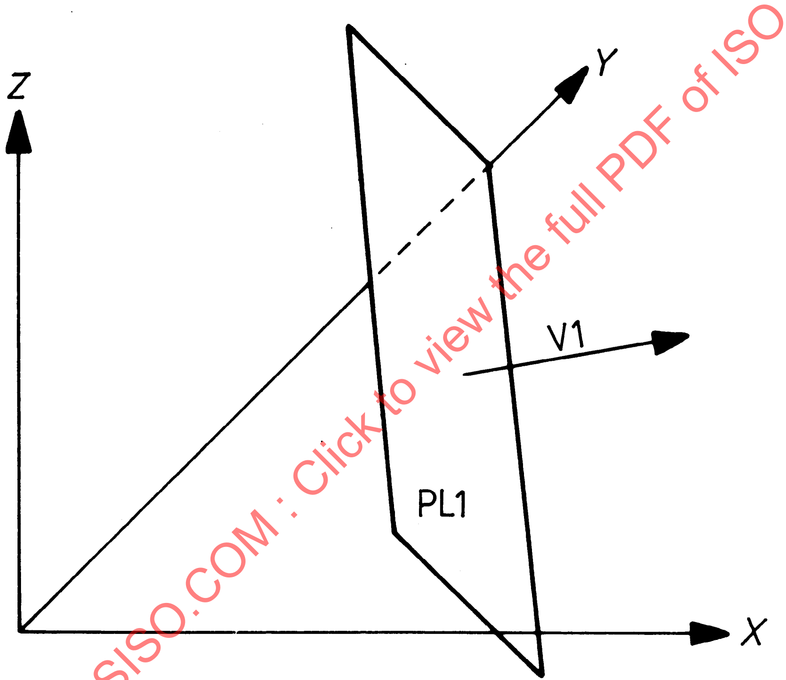
8.8.4.1 Semantics

Plane is the symbol for a plane.

The modifier specifies the sense of the vector.

The resulting vector is normalized.

8.8.4.2 Example



V1 = VECTOR/PERPTO, PL1, POSX

Figure 68

8.8.4.3 Limitations

When the given plane is perpendicular to a coordinate axis, only the corresponding modifier shall be used.

8.8.4.4 Syntax

< vector parameter list > : : = PERPTO, < plane spec > ,[POSX | POSY | POSZ | NEGX | NEGY | NEGZ]

8.8.5 Definition of a vector by a scalar times a vector

VECTOR/scalar, TIMES, vector

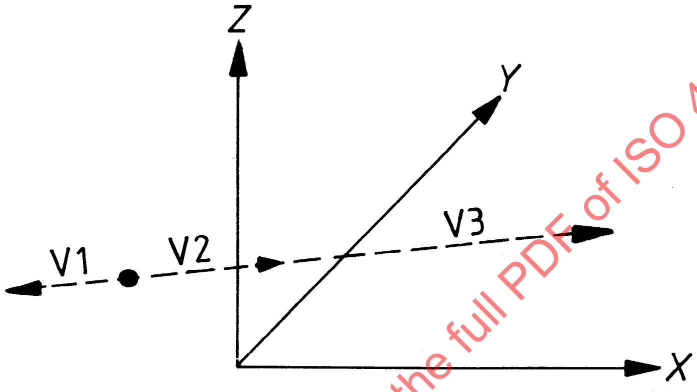
8.8.5.1 Semantics

Vector is the symbol for a vector.

Scalar specifies the ratio between the desired vector and the given one.

TIMES specifies that the required vector is produced by multiplying the referenced vector by the referenced scalar.

8.8.5.2 Example



V3 = VECTOR/4, TIMES, V1

V2 = VECTOR/ – 1, TIMES, V1

Figure 69

8.8.5.3 Limitations

None.

8.8.5.4 Syntax

< vector parameter list > : : = < scalar > , TIMES, < vector spec >

8.8.6 Definition of a vector by the cross product of two given vectors

VECTOR/vector1, CROSS, vector2

8.8.6.1 Semantics

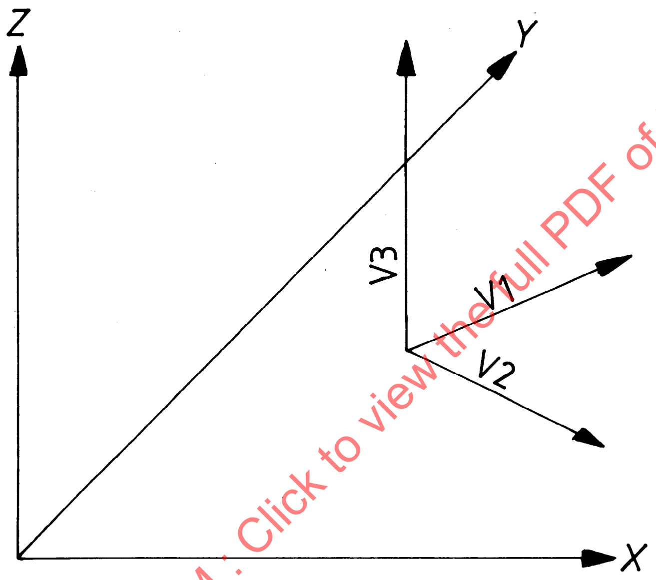
Vector1 and vector2 are the symbols of two vectors.

CROSS specifies that the required vector is the cross product of the two given vectors.

The coordinate system defined by vector1, vector2 and the desired vector is positive.

The resulting vector is normalized.

8.8.6.2 Example



V3 = VECTOR/V2, CROSS, V1

Figure 70

8.8.6.3 Limitations

The two vectors shall not be parallel.

8.8.6.4 Syntax

< vector parameter list > : : = < vector spec > , CROSS, < vector spec >

8.8.7 Definition of a vector by normalizing a given vector, a given point, or given components

```
                  x, y, z
VECTOR/UNIT,      vector
                  point
```

8.8.7.1 Semantics

UNIT specifies that the required vector is produced by normalizing.

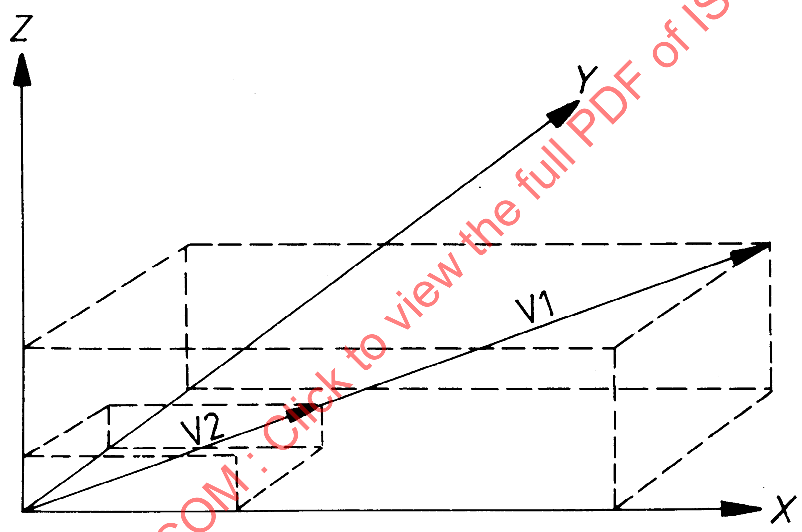
Vector is the symbol for a vector.

Point is the symbol for the end of a vector which starts at the origin.

x, y, z are the components of the end of a vector which starts at the origin.

The normalized vector has the same direction as the given vector with a magnitude equal to one.

8.8.7.2 Example



V2 = VECTOR/UNIT, V1

Figure 71

8.8.7.3 Limitations

None.

8.8.7.4 Syntax

< vector parameter list > : : = UNIT ,[< scalar > , < scalar > , < scalar > | < vector spec > | < point spec >]

8.8.8 Definition of a vector by a length and a given angle in a given plane

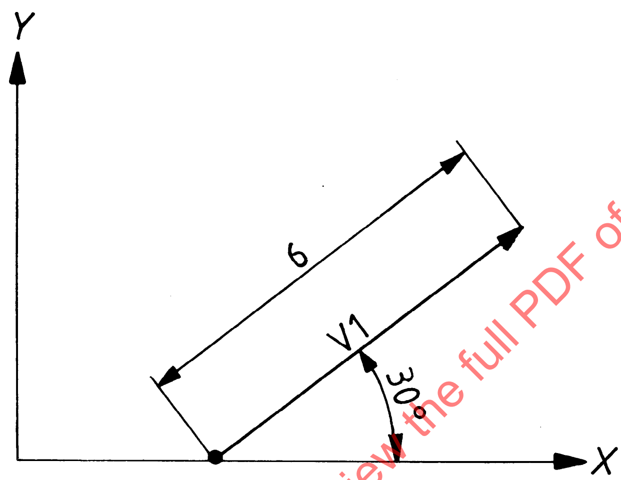
```
                                                XYPLAN
VECTOR/LENGTH, length, ATANGL, angle,  YZPLAN
                                                ZXPLAN
```

8.8.8.1 Semantics

LENGTH, length specifies the vector length.

ATANGL, angle specifies the angle in the indicated plane.

8.8.8.2 Example



V1 = VECTOR/LENGTH, 6, ATANGL, 30, XYPLAN

Figure 72

8.8.8.3 Limitations

Length may not have negative value.

8.8.8.4 Syntax

< vector parameter list > : : = LENGTH, < scalar > , ATANGL, < scalar > ,[XYPLAN | YZPLAN | ZXPLAN]

8.8.9 Definition of a vector parallel to the intersection of two given planes

```
                                           POSX
                                           POSY
VECTOR/PARLEL, INTOF, plane1, plane2,      POSZ
                                           NEGX
                                           NEGY
                                           NEGZ
```

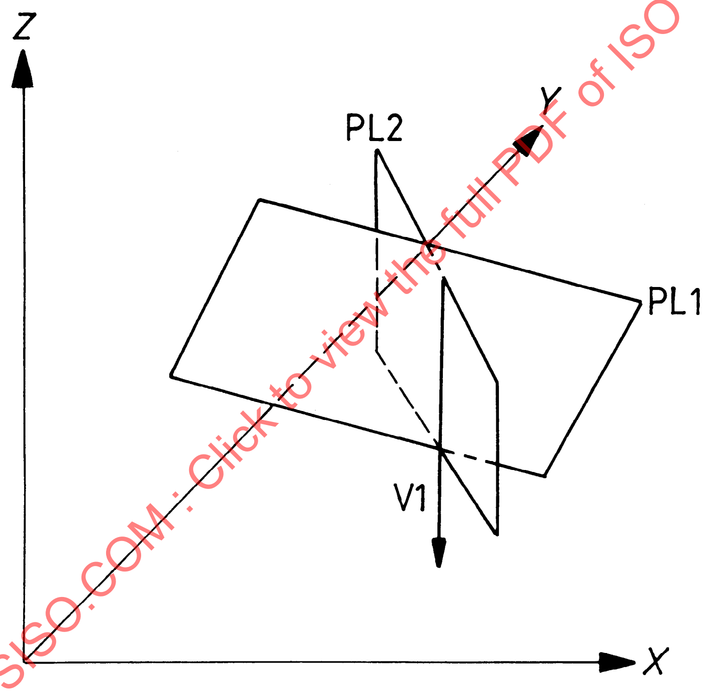
8.8.9.1 Semantics

Plane1 and plane2 are the symbols for two planes.

The modifier specifies the sense of the vector.

The resulting vector is normalized.

8.8.9.2 Example



V1 = VECTOR/PARLEL, INTOF, PL1, PL2, NEGZ

Figure 73

8.8.9.3 Limitations

When the intersected line is parallel to a coordinate axis, only the corresponding modifier shall be used.

8.8.9.4 Syntax

< vector parameter list > : : = PARLEL, INTOF, < plane spec > , < plane spec > ,[POSX | POSY | POSZ | NEGX | NEGY | NEGZ]

8.8.10 Definition of a vector by the addition or subtraction of two given vectors

VECTOR/vector1,	PLUS	, vector2
	MINUS	

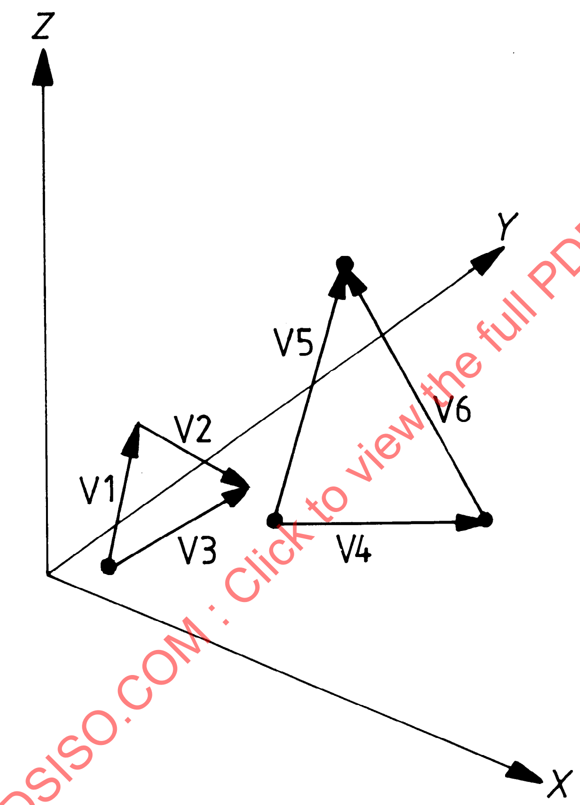
8.8.10.1 Semantics

Vector1 and vector2 are the symbols for vectors.

PLUS specifies that the required vector is an addition of the two referenced vectors.

MINUS specifies that the required vector is a result of subtracting the second referenced vector from the first.

8.8.10.2 Example



V3 = VECTOR/V1, PLUS, V2

V6 = VECTOR/V5, MINUS, V4

Figure 74

8.8.10.3 Limitations

None.

8.8.10.4 Syntax

< vector parameter list > : : = < vector spec > ,[PLUS | MINUS], < vector spec >

8.8.11 Definition of a vector in the XY-plane making a specified angle with a given line

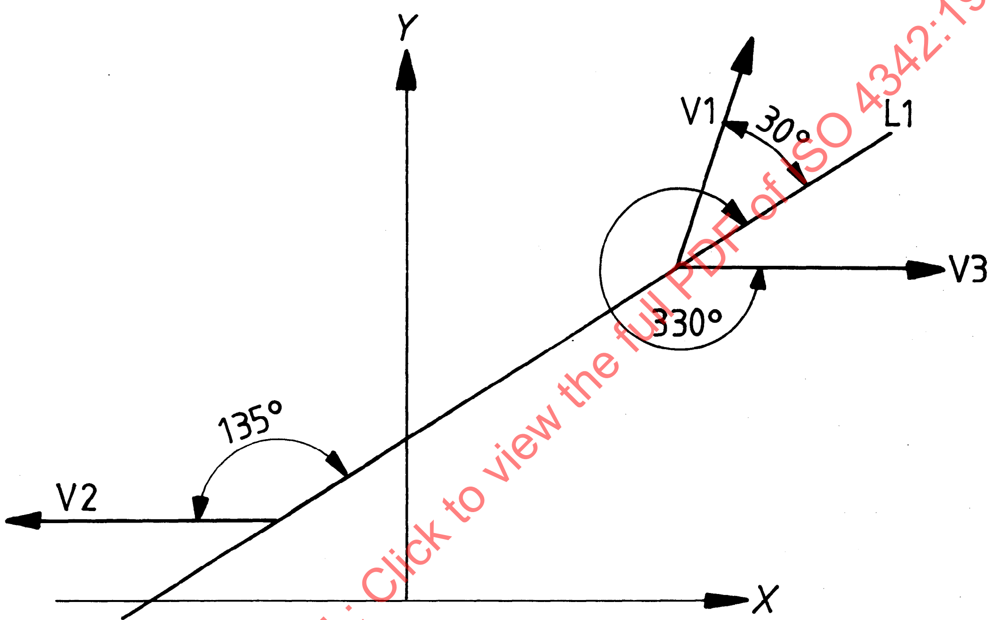
VECTOR/ATANGL, angle, line POSX / NEGX / POSY / NEGY

8.8.11.1 Semantics

Line is the symbol for a line.

ATANGL, angle is the symbol for the angle measured from the line to the vector.

8.8.11.2 Example



V1 = VECTOR/ATANGL, 30, L1, POSY

V2 = VECTOR/ATANGL, 135, L1, NEGX

V3 = VECTOR/ATANGL, 330, L1, POSX

Figure 75

8.8.11.3 Limitations

None.

8.8.11.4 Syntax

< vector parameter list > : : = ATANGL, < scalar > , < line spec > ,[POSX | NEGX | POSY | NEGY]

8.9 Definitions of a circle

8.9.1 Definition of a circle

CIRCLE/circle parameter list

8.9.1.1 Semantics

Circle is always to be considered as a circular cylinder perpendicular to the XY-plane.

Unless the circle centre is part of the definition, the z value for the centre point is determined by the current ZSURF.

8.9.1.2 Limitations

The radius value shall be positive.

XLARGE, XSMALL, YLARGE, YSMALL, when they are used, must be significant to distinguish among the possible solutions.

8.9.1.3 Sub-contents

For the definition of a circle

a) by the coordinates of the centre and its radius, see 8.9.2;
b) by the centre point and its radius, see 8.9.3;
c) by the centre point and a tangential line, see 8.9.4;
d) by the centre point and a point on the circumference, see 8.9.5;
e) by three points on the circumference, see 8.9.6;
f) by the centre point and a tangential circle, see 8.9.7;
g) by two tangential lines and its radius, see 8.9.8;
h) by a tangential line, a point on the circumference and its radius, see 8.9.9;
j) by two tangential circles and its radius, see 8.9.10;
k) by a tangential line, a tangential circle and its radius, see 8.9.11;
m) by a tangential line, a tangential tabulated cylinder and its radius, see 8.9.12;
n) by three tangential lines, see 8.9.13;
p) by two points on its circumference and its radius, see 8.9.14;
q) by a point on the circumference, a tangential circle and its radius, see 8.9.15;
r) by a tangential line, a tangential conic and its radius, 8.9.16;
s) by two tangential tabulated cylinders and its radius, see 8.9.17;
t) by a tangential circle, a tangential tabulated cylinder and its radius, see 8.9.18;
u) by a point on its circumference, a tangential tabulated cylinder and its radius, see 8.9.19;
v) by a line through its centre, a tangential tabulated cylinder and its radius, see 8.9.20;
w) by the coordinates of its centre, the direction cosines of the axis of its right cylinder, and its radius, see 8.9.21.

8.9.1.4 Syntax

<circle definition statement> : : = <identifier> = CIRCLE/ <circle parameter list>

8.9.2 Definition of a circle by the coordinates of the centre and its radius

CIRCLE/*x*, *y*, *z*, *r*

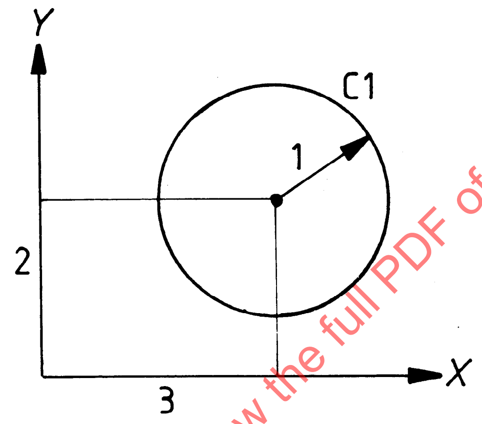
8.9.2.1 Semantics

x, *y*, *z* are the coordinates of the centre.

r is the radius.

If *z* is omitted the value is given by the current ZSURF.

8.9.2.2 Example



C1 = CIRCLE/3, 2, 1

Figure 76

8.9.2.3 Limitations

None.

8.9.2.4 Syntax

< circle parameter list > : : = < scalar > , < scalar > , < scalar > , $^1_0$ [< scalar >]

8.9.3 Definition of a circle by its centre point and its radius

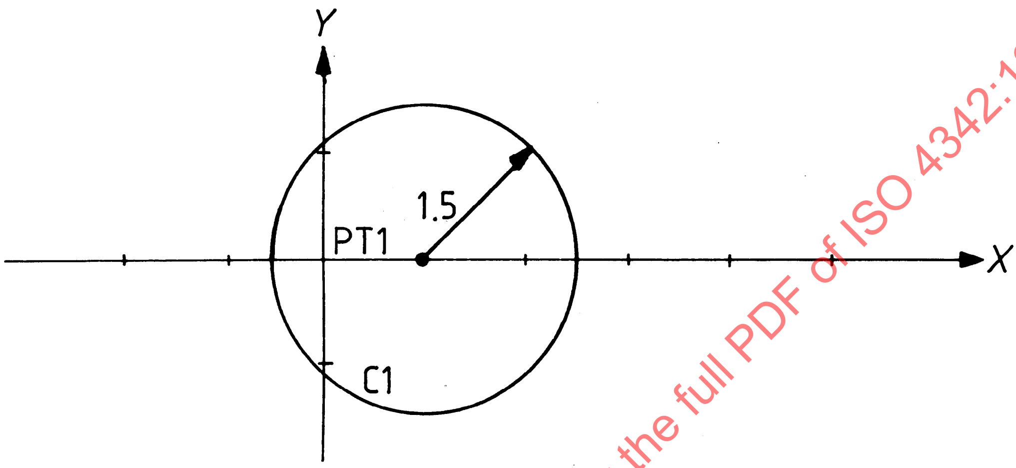
CIRCLE/CENTER, point, RADIUS, *r*

8.9.3.1 Semantics

CENTER, point specifies that the centre of the required circle is identified by the specified point.

RADIUS, *r* specifies the value of the radius of the circle.

8.9.3.2 Example



C1 = CIRCLE/CENTER, PT1, RADIUS, 1.5

Figure 77

8.9.3.3 Limitations

None.

8.9.3.4 Syntax

<circle parameter list> : : = CENTER, <point spec> , RADIUS, <scalar>

8.9.4 Definition of a circle by its centre point and a tangential line

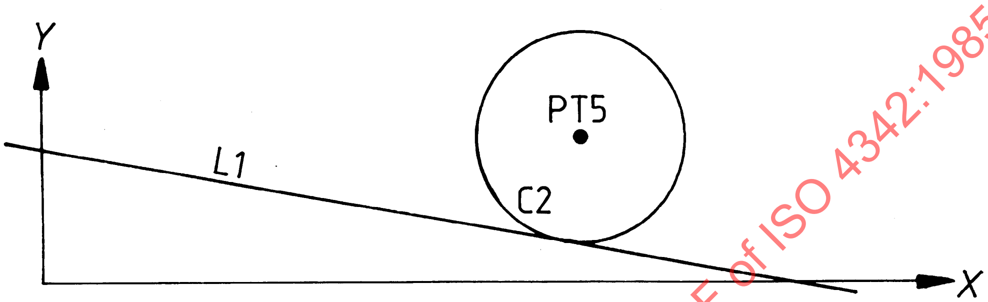
CIRCLE/CENTER, point, TANTO, line

8.9.4.1 Semantics

CENTER, point specifies that the centre of the required circle is identified by the specified point.

Line is the symbol for a line and identifies the tangential line.

8.9.4.2 Example



C2 = CIRCLE/CENTER, PT5, TANTO, L1

Figure 78

8.9.4.3 Limitations

None.

8.9.4.4 Syntax

< circle parameter list > : : = CENTER, < point spec > , TANTO, < line spec >

8.9.5 Definition of a circle by its centre point and a point on the circumference

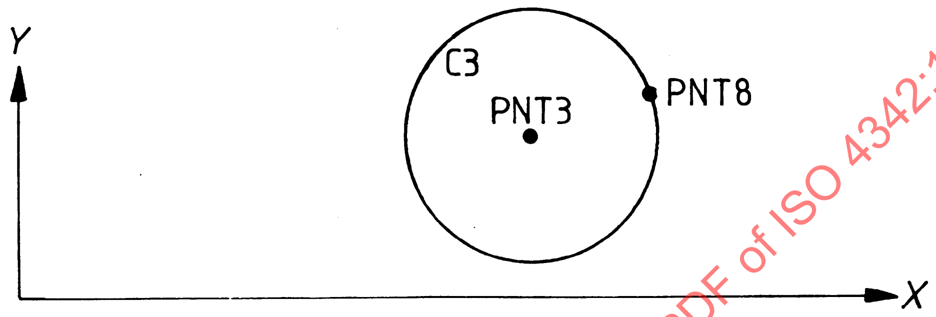
CIRCLE/CENTER, point1, point2

8.9.5.1 Semantics

CENTER, point1 specifies that the centre of the required circle is identified by the specified point1.

Point2 is the symbol for a point and identifies a point on the circumference.

8.9.5.2 Example



C3 = CIRCLE/CENTER, PNT3, PNT8

Figure 79

8.9.5.3 Limitations

None.

8.9.5.4 Syntax

< circle parameter list > : : = CENTER, < point spec > , < point spec >

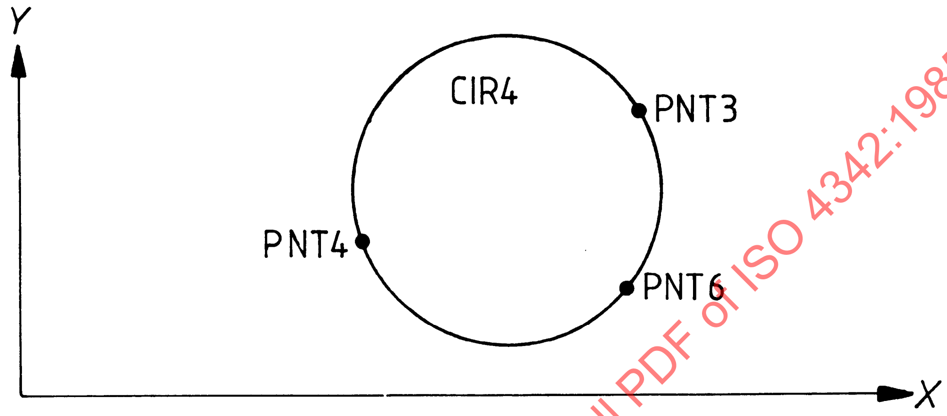
8.9.6 Definition of a circle by three points on the circumference

CIRCLE/point1, point2, point3

8.9.6.1 Semantics

Point1, point2 and point3 are symbols for points and identify three points on the circumference.

8.9.6.2 Example



CIR4 = CIRCLE/PNT4, PNT3, PNT6

Figure 80

8.9.6.3 Limitations

No two points shall coincide in their x- and y-coordinates nor shall the three points lie on the same straight line.

8.9.6.4 Syntax

< circle parameter list > : : = < point spec > , < point spec > , < point spec >

8.9.7 Definition of a circle by its centre point and a tangential circle

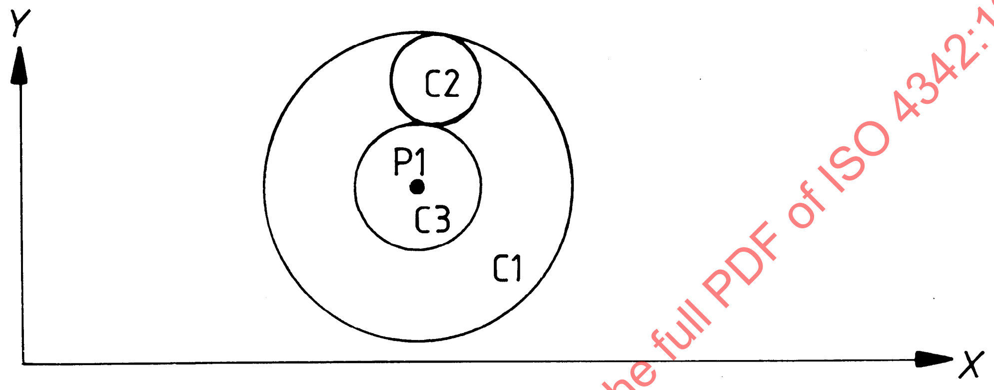
CIRCLE/CENTER, point, LARGE / SMALL , TANTO, circle

8.9.7.1 Semantics

CENTER, point specifies that the centre of the required circle is identified by the specified point.

Circle is the symbol for a circle and identifies the tangential circle.

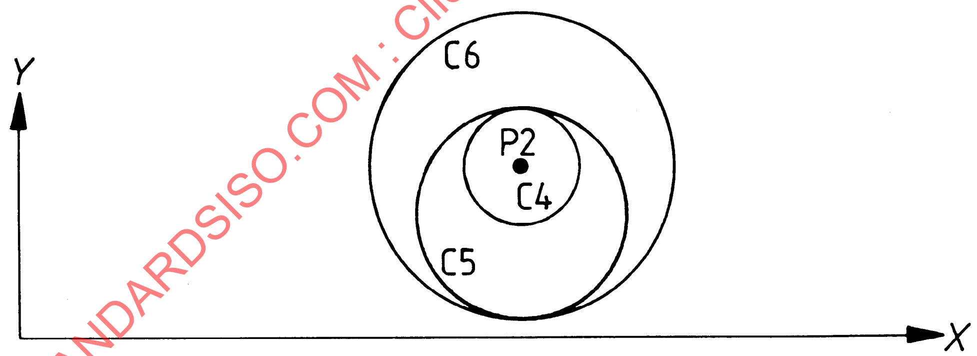
8.9.7.2 Example



C1 = CIRCLE/CENTER, P1, LARGE, TANTO, C2

C3 = CIRCLE/CENTER, P1, SMALL, TANTO, C2

Figure 81 a)



C4 = CIRCLE/CENTER, P2, SMALL, TANTO, C5

C6 = CIRCLE/CENTER, P2, LARGE, TANTO, C5

Figure 81 b)

8.9.7.3 Limitations

The point shall not coincide with the centre point of the given circle.

8.9.7.4 Syntax

< circle parameter list > : : = CENTER, < point spec > ,[LARGE | SMALL], TANTO, < circle spec >

8.9.8 Definition of a circle by two tangential lines and its radius

	XLARGE		XLARGE	
CIRCLE/	XSMALL YLARGE	, line1,	XSMALL YLARGE	, line2, RADIUS, *r*
	YSMALL		YSMALL	

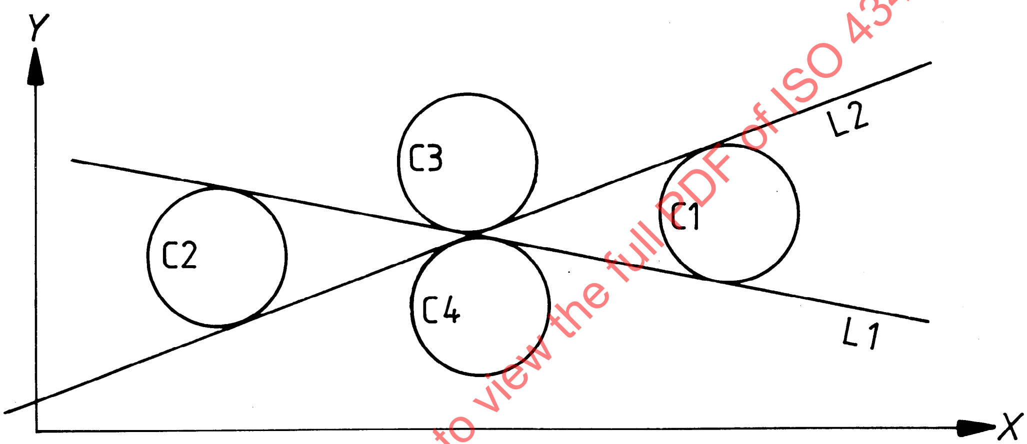
8.9.8.1 Semantics

Line1 and line2 are symbols for lines and name the two tangential lines.

XLARGE, XSMALL, YLARGE and YSMALL specify the position of the desired circle with respect to the tangential line.

RADIUS, *r* specifies the value of the radius of the circle.

8.9.8.2 Example



C1 = CIRCLE/YLARGE, L1, XLARGE, L2, RADIUS, 0.5

C2 = CIRCLE/YSMALL, L1, YLARGE, L2, RADIUS, 0.5

C3 = CIRCLE/YLARGE, L2, YLARGE, L1, RADIUS, 0.5

C4 = CIRCLE/YSMALL, L1, YSMALL, L2, RADIUS, 0.5

Figure 82

8.9.8.3 Limitations

XLARGE or XSMALL, respectively YLARGE or YSMALL, shall not be used when the associated line is parallel to the corresponding coordinate axis.

The two lines shall not be parallel.

8.9.8.4 Syntax

< circle parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL], < line spec > ,[XLARGE | XSMALL | YLARGE | YSMALL], < line spec > , RADIUS, < scalar >

8.9.9 Definition of a circle by a tangential line, a point on the circumference and its radius

CIRCLE/TANTO, line, XLARGE / XSMALL / YLARGE / YSMALL, point, RADIUS, *r*

8.9.9.1 Semantics

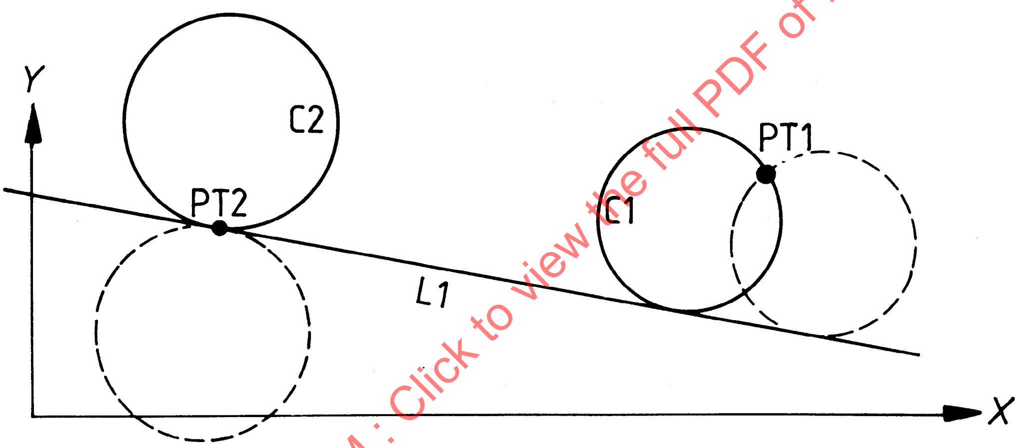
Line is the symbol for a line and identifies the tangential line.

Point is the symbol for a point and identifies a point on the circumference.

XLARGE, XSMALL, YLARGE and YSMALL allow the selection of the considered circle from the two possible circles with respect to the relative position of their centres. In the case of the point lying on the tangential line (i.e. it is at the point of tangency), XLARGE or its alternatives allow the selection of the position of the desired circle with respect to the tangent line.

RADIUS, *r* specifies the value of the radius of the circle.

8.9.9.2 Example



C1 = CIRCLE/TANTO, L1, XSMALL, PT1, RADIUS, 0.5

C2 = CIRCLE/TANTO, L1, YLARGE, PT2, RADIUS, 0.75

Figure 83

8.9.9.3 Limitations

The point shall not have a larger distance from the line than twice the radius.

When the line is parallel to a coordinate axis then only one pair of the modifiers, XLARGE, etc., is appropriate. If, for example, the line is parallel to the X-axis then only XLARGE or XSMALL should be used : the exception is when the point lies on the line, in which case YLARGE or YSMALL would be appropriate.

8.9.9.4 Syntax

< circle parameter list > : : = TANTO, < line spec > ,[XLARGE | XSMALL | YLARGE | YSMALL],
< point spec > , RADIUS, < scalar >

8.9.10 Definition of a circle tangential to two circles and having a given radius

CIRCLE/	XLARGE XSMALL YLARGE YSMALL	,	IN OUT	, circle1,	IN OUT	, circle2, RADIUS, *r*

8.9.10.1 Semantics

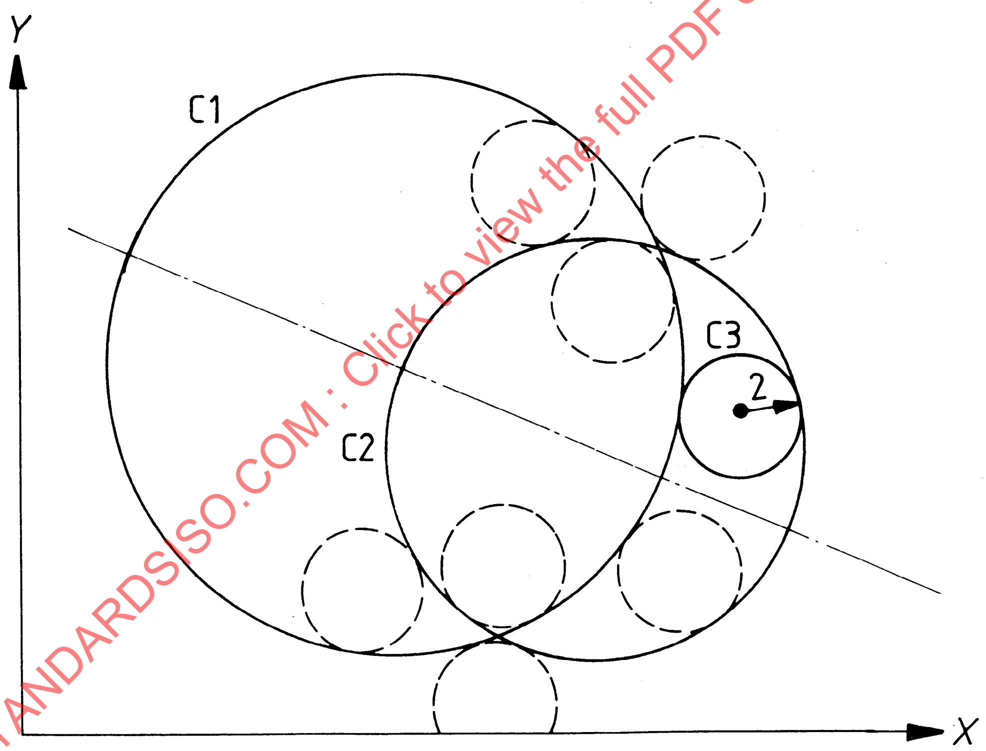
Circle1 and circle2 are the symbols for two circles.

XLARGE, XSMALL, YLARGE, YSMALL allow the selection of the considered circle from among the possible ones with respect to the relative position of their centres. XLARGE or XSMALL, YLARGE or YSMALL may not be used when the line defined by the centres of the two circles is perpendicular to the corresponding coordinates axes.

RADIUS, *r* specifies the value of the radius of the circle.

The maximum number of possible circles is eight. According to the value of the radius, some of these circles may not be real; nevertheless, the three modifiers must be present.

8.9.10.2 Example



C3 = CIRCLE/YLARGE, IN, C2, OUT, C1, RADIUS, 2

Figure 84

8.9.10.3 Limitations

The modifiers used and the given radius value shall correspond to a real and single solution.

8.9.10.4 Syntax

< circle parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL],[IN | OUT],
< circle spec > ,[IN | OUT], < circle spec > , RADIUS, < scalar >

8.9.11 Definition of a circle by a tangential line, a tangential circle and its radius

	XLARGE		XLARGE			
CIRCLE/	XSMALL	line,	XSMALL	IN		circle, RADIUS, *r*
	YLARGE'		YLARGE '	OUT '		
	YSMALL		YSMALL			

8.9.11.1 Semantics

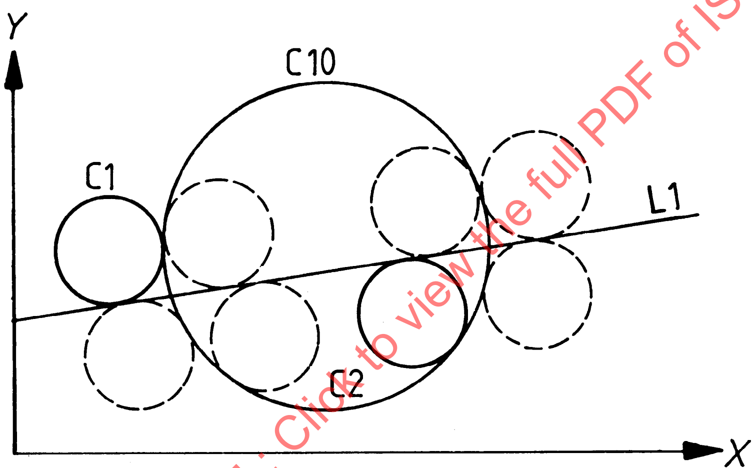
Line is the symbol for a line and identifies the tangential line.

Circle is the symbol for a circle and identifies the tangential circle.

RADIUS, *r* specifies the value of the radius of the circle.

The maximum number of possible circles is eight. According to the value of the radius, some of these circles may not be real; nevertheless the three modifiers must be present.

8.9.11.2 Example



C1 = CIRCLE/YLARGE, L1, XSMALL, OUT, C10, RADIUS, 0.5

C2 = CIRCLE/YSMALL, L1, XLARGE, IN, C10, RADIUS, 0.5

Figure 85

8.9.11.3 Limitations

The line shall not have a larger distance from the nearest point on the circumference than twice the radius.

8.9.11.4 Syntax

< circle parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL], < line spec > ,[XLARGE | XSMALL | YLARGE | YSMALL],[IN | OUT], < circle spec > , RADIUS, < scalar >

8.9.12 Definition of a circle by a tangential line, a tangential tabulated cylinder and its radius

	XLARGE		XLARGE	
CIRCLE/	XSMALL	, line,	XSMALL	, tabcyl, point, RADIUS, *r*
	YLARGE		YLARGE	
	YSMALL		YSMALL	

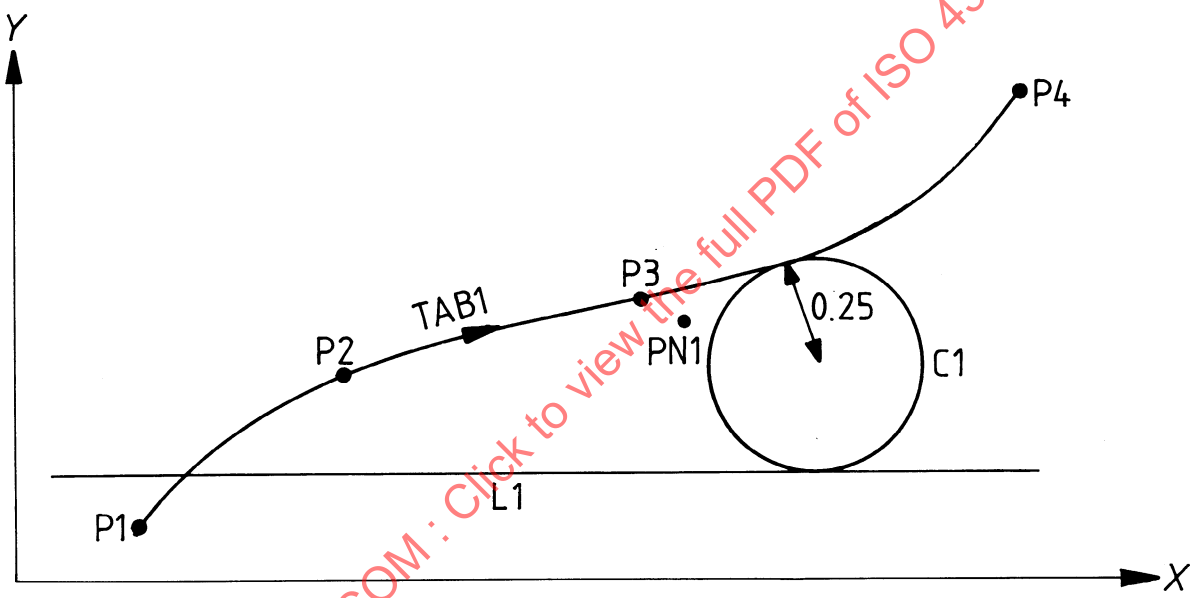
8.9.12.1 Semantics

The first XLARGE, XSMALL, YLARGE or YSMALL modifier gives the position of the required circle relative to the line, and the second one gives the position relative to tabcyl, a tabulated cylinder perpendicular to the *XY*-plane.

Point is the symbol of a point in the vicinity of the tangency point of the circle and the tabulated cylinder.

RADIUS, *r* specifies the value of the radius of the circle.

8.9.12.2 Example



C1 = CIRCLE/YLARGE, L1, YSMALL, TAB1, PN1, RADIUS, 0.25

Figure 86

8.9.12.3 Limitations

None.

8.9.12.4 Syntax

< circle parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL], < line spec >
,[XLARGE | XSMALL | YLARGE | YSMALL], < tabulated cylinder spec > ,
< point spec > , RADIUS, < scalar >

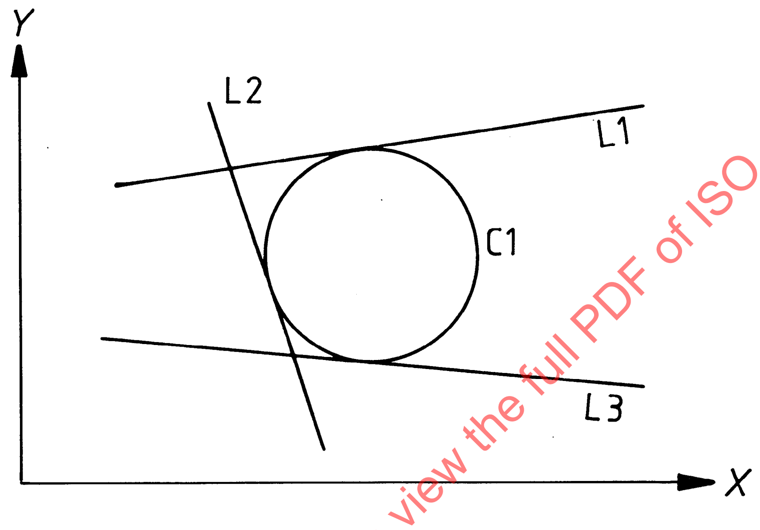
8.9.13 Definition of a circle by three tangential lines

CIRCLE/	XLARGE XSMALL YLARGE YSMALL	, line1,	XLARGE XSMALL YLARGE YSMALL	, line2,	XLARGE XSMALL YLARGE YSMALL	, line3

8.9.13.1 Semantics

Line1, line2, line3 are the symbols for three lines.

8.9.13.2 Example



C1 = CIRCLE/YSMALL, L1, XLARGE, L2, YLARGE, L3

Figure 87

8.9.13.3 Limitations

The three lines shall not be parallel nor shall any two be coincident.

8.9.13.4 Syntax

< circle parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL], < line spec > ,[XLARGE | XSMALL | YLARGE | YSMALL], < line spec > ,[XLARGE | XSMALL | YLARGE | YSMALL], < line spec >

8.9.14 Definition of a circle by two points on its circumference and its radius

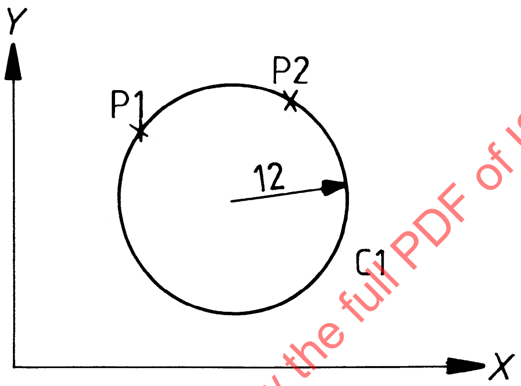
```
          XLARGE
CIRCLE/   XSMALL   , point1, point2, RADIUS, r
          YLARGE
          YSMALL
```

8.9.14.1 Semantics

Point1 and point2 are the symbols for two points.

RADIUS, *r* specifies the value of the radius of the circle.

8.9.14.2 Example



C1 = CIRCLE/YSMALL, P1, P2, RADIUS, 12

Figure 88

8.9.14.3 Limitations

The two points shall not coincide in their *x*- and *y*-coordinates and the distance between the two points shall not be larger than twice the radius.

8.9.14.4 Syntax

< circle parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL], < point spec > , < point spec >,
RADIUS, < scalar >

8.9.15 Definition of a circle by a point on its circumference, a tangential circle and its radius

CIRCLE/point, $\left\{\begin{matrix}\text{XLARGE}\\\text{XSMALL}\\\text{YLARGE}\\\text{YSMALL}\end{matrix}\right\}$, $\left\{\begin{matrix}\text{IN}\\\text{OUT}\end{matrix}\right\}$, circle, RADIUS, *r*

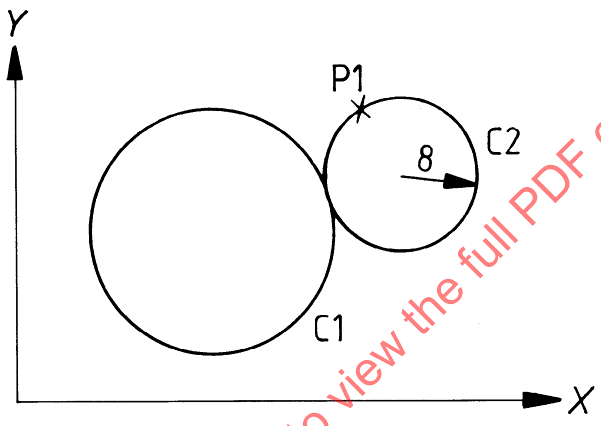
8.9.15.1 Semantics

Point is the symbol for a point.

Circle is the symbol for a circle.

RADIUS, *r* specifies the value of the radius of the circle.

8.9.15.2 Example



C2 = CIRCLE/P1, YSMALL, OUT, C1, RADIUS, 8

Figure 89

8.9.15.3 Limitations

The point shall not have a larger distance from the nearest point on the circumference than twice the radius.

8.9.15.4 Syntax

< circle parameter list > : : = < point spec > ,[XLARGE | XSMALL | YLARGE | YSMALL],[IN | OUT],
< circle spec > , RADIUS, < scalar >

8.9.16 Definition of a circle by a tangential line, a tangential conic and its radius

```
        XLARGE         XLARGE
CIRCLE/ XSMALL , line, XSMALL , IN  , conic, point, RADIUS, r
        YLARGE         YLARGE   OUT
        YSMALL         YSMALL
```

8.9.16.1 Semantics

Line is the symbol for a line.

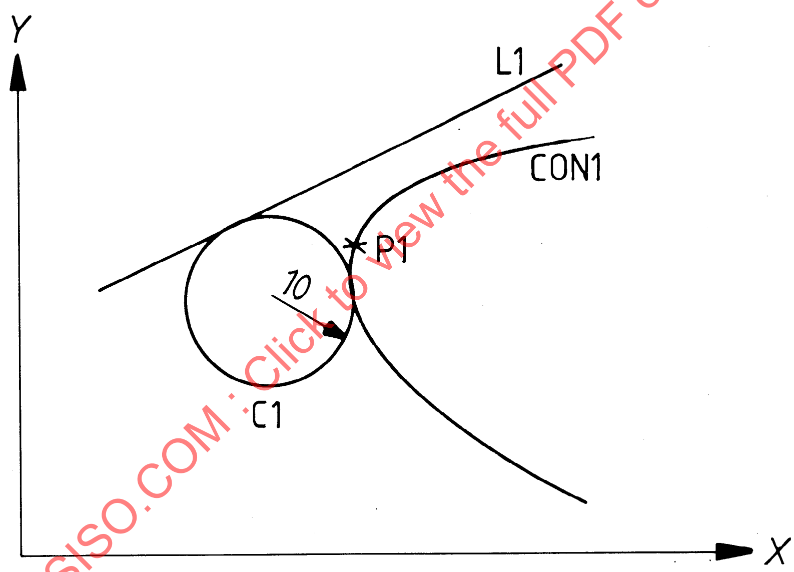
Conic is the symbol for a conic.

Point is the symbol for a point.

RADIUS, *r* specifies the value of the radius of the circle.

The maximum number of possible circles is eight. According to the value of the radius and the figure, some of these circles may not be real; nevertheless the three modifiers must be present.

8.9.16.2 Example



C1 = CIRCLE/YSMALL, L1, XSMALL, OUT, CON1, P1, RADIUS, 10

Figure 90

8.9.16.3 Limitations

The line shall not have a larger distance from the nearest point on the circumference than twice the radius.

8.9.16.4 Syntax

< circle parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL], < line spec > ,[XLARGE | XSMALL |
YLARGE | YSMALL],[IN | OUT], < conic spec > , < point spec > ,
RADIUS, < scalar >

8.9.17 Definition of a circle by two tangential tabulated cylinders and its radius

	XSMALL		XSMALL	
CIRCLE/	XLARGE YSMALL	, tabcyl1, point1,	XLARGE YSMALL	, tabcyl2, point2, RADIUS, *r*
	YLARGE		YLARGE	

8.9.17.1 Semantics

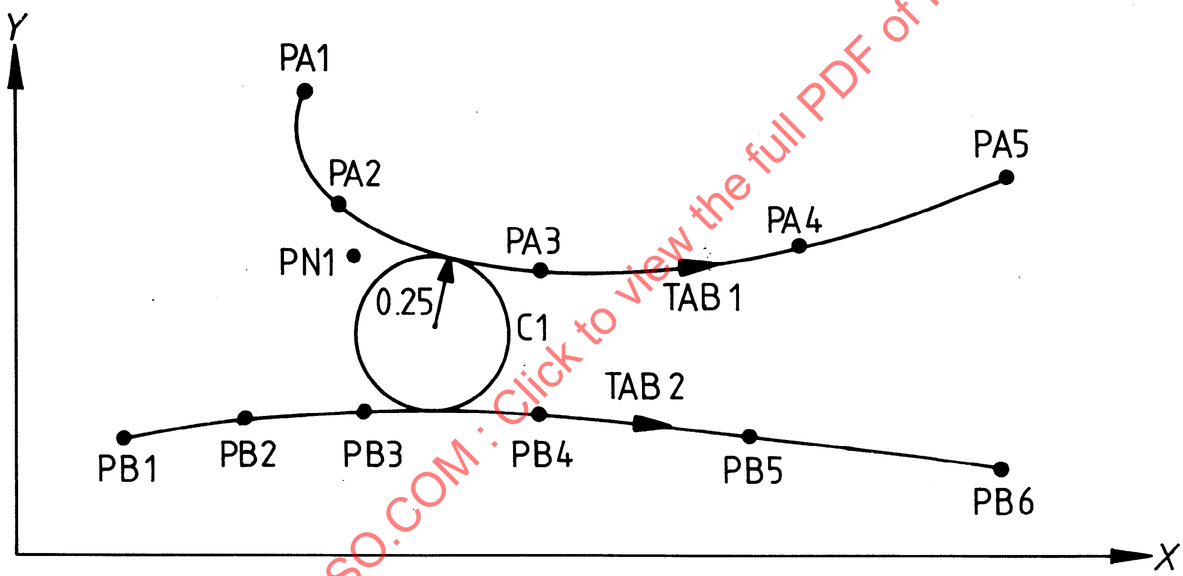
Tabcyl1 and tabcyl2 are the symbols for two tabulated cylinders perpendicular to the XY-plane.

Point1 and point2 are the symbols for points in the vicinity of the required tangency point of the circle and the tabulated cylinder immediately preceding the point.

RADIUS, *r* specifies the value of the radius of the circle.

The modifiers XSMALL, XLARGE, YSMALL, YLARGE give the position of the circle relative to the following tabulated cylinder looking along the direction of the tabulated cylinder.

8.9.17.2 Example



C1 = CIRCLE/YLARGE, TAB2, PB3, YSMALL, TAB1, PN1, RADIUS, 0.25

Figure 91

8.9.17.3 Limitations

None.

8.9.17.4 Syntax

< circle parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL], < tabulated cylinder spec > , < point spec > ,
[XLARGE | XSMALL | YLARGE | YSMALL], < tabulated cylinder spec > , < point spec > ,
RADIUS, < scalar >

8.9.18 Definition of a circle by a tangential circle, a tangential tabulated cylinder and its radius

CIRCLE/ IN | OUT , circle, XLARGE | XSMALL | YLARGE | YSMALL , tabcyl, point, RADIUS, *r*

8.9.18.1 Semantics

Circle is the symbol for a circle.

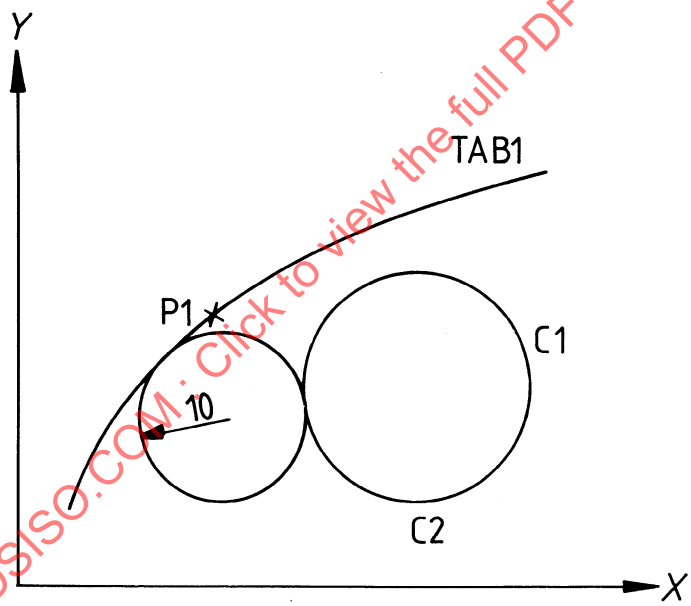
Tabcyl is the symbol for a tabulated cylinder perpendicular to the *XY*-plane.

Point is the symbol for a point.

RADIUS, *r* specifies the value of the radius of the circle.

The maximum number of possible circles is four. According to the value of the radius and the figure some of these circles may not be real; nevertheless the two modifiers must be present.

8.9.18.2 Example



C2 = CIRCLE/OUT, C1, XLARGE, TAB1, P1, RADIUS, 10

Figure 92

8.9.18.3 Limitations

None.

8.9.18.4 Syntax

< circle parameter list > : : = [IN | OUT], < circle spec > ,[XLARGE | XSMALL | YLARGE | YSMALL], < tabulated cylinder spec > , < point spec > , RADIUS, < scalar >

8.9.19 Definition of a circle by a point on its circumference, a tangential tabulated cylinder and its radius

CIRCLE/point1, {XLARGE | XSMALL | YLARGE | YSMALL}, tabcyl, point2, RADIUS, *r*

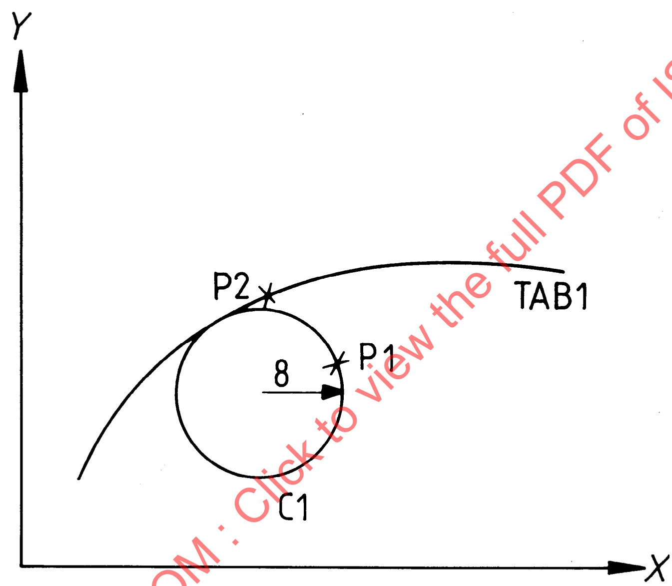
8.9.19.1 Semantics

Point1 and point2 are the symbols for two points.

Tabcyl is the symbol for a tabulated cylinder perpendicular to the *XY*-plane.

RADIUS, *r* specifies the value of the radius of the circle.

8.9.19.2 Example



C1 = CIRCLE/P1, YSMALL, TAB1, P2, RADIUS, 8

Figure 93

8.9.19.3 Limitations

None.

8.9.19.4 Syntax

< circle parameter list > : : = < point spec > ,[XLARGE | XSMALL | YLARGE | YSMALL],
< tabulated cylinder spec > , < point spec > , RADIUS, < scalar >

8.9.20 Definition of a circle by a line through its centre, a tangential tabulated cylinder and its radius

```
                    XLARGE
CIRCLE/ON, line,    XSMALL  , tabcyl, point, RADIUS, r
                    YLARGE
                    YSMALL
```

8.9.20.1 Semantics

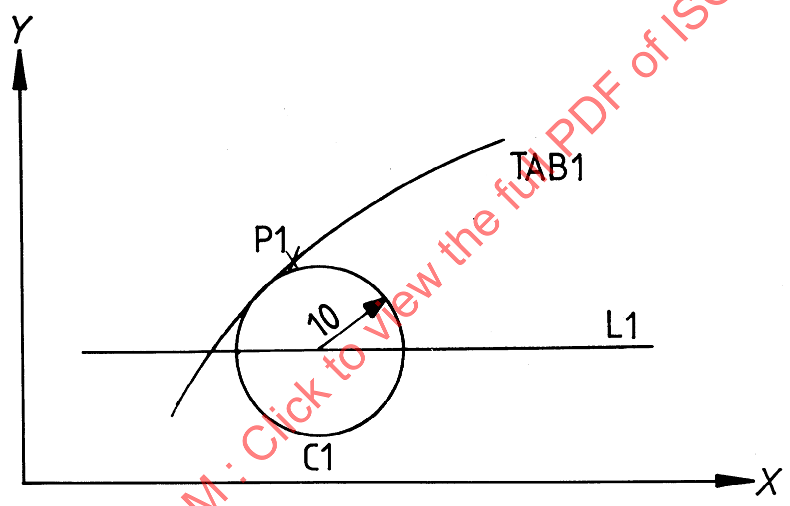
ON, line specifies that the centre of the required circle lies on the specified line.

Tabcyl is the symbol for a tabulated cylinder perpendicular to the XY-plane.

Point is the symbol for a point.

RADIUS, *r* specifies the value of the radius of the circle.

8.9.20.2 Example



C1 = CIRCLE/ON, L1, XLARGE, TAB1, P1, RADIUS, 10

Figure 94

8.9.20.3 Limitations

None.

8.9.20.4 Syntax

< circle parameter list > : : = ON, < line spec > ,[XLARGE | XSMALL | YLARGE | YSMALL],
< tabulated cylinder spec > , < point spec > , RADIUS , < scalar >

8.9.21 Definition of a circle by the coordinates of its centre, the direction cosines of the axis of its right vertical cylinder and its radius

CIRCLE/x, y, z, i, j, k, r

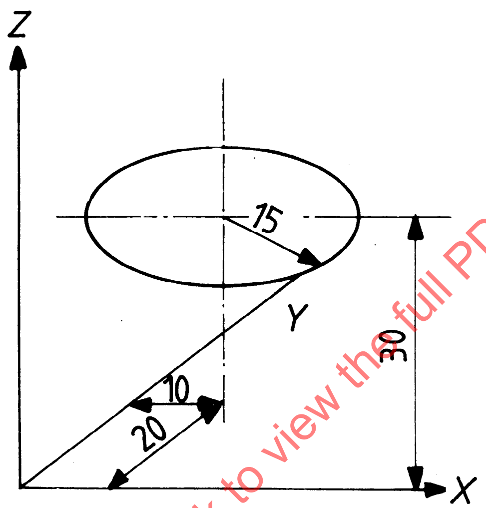
8.9.21.1 Semantics

x, y and z are the Cartesian coordinates of the centre.

i, j and k are the direction cosines of the axis of a right vertical cylinder.

r is the radius.

8.9.21.2 Example



C1 = CIRCLE/10, 20, 30, 0, 0, 1, 15

Figure 95

8.9.21.3 Limitations

None.

8.9.21.4 Syntax

< circle parameter list > : : = < scalar > $^{6}_{6}$ [, < scalar >]

8.10 Definitions of a cylinder

8.10.1 Definition of a cylinder

CYLNDR/cylinder parameter list

8.10.1.1 Semantics

A cylinder is the focus of all points at a constant distance from a given line.

8.10.1.2 Limitations

XLARGE, XSMALL, YLARGE, YSMALL, ZLARGE, ZSMALL, when they are used, shall be significant to distinguish among the possible solutions.

8.10.1.3 Sub-contents

For the definition of a cylinder

a) by substituting for symbols in the canonical form, see 8.10.2;

b) by two tangential planes and its radius, see 8.10.3;

c) by a tangential plane, tangential cylinder and its radius, see 8.10.4;

d) by two tangential cylinders and its radius, see 8.10.5.

8.10.1.4 Syntax

< cylinder definition statement > : : = < identifier > = CYLNDR/ < cylinder parameter list >

8.10.2 Definition of a cylinder by substituting for symbols in the canonical form

CYLNDR/point, vector, *r*

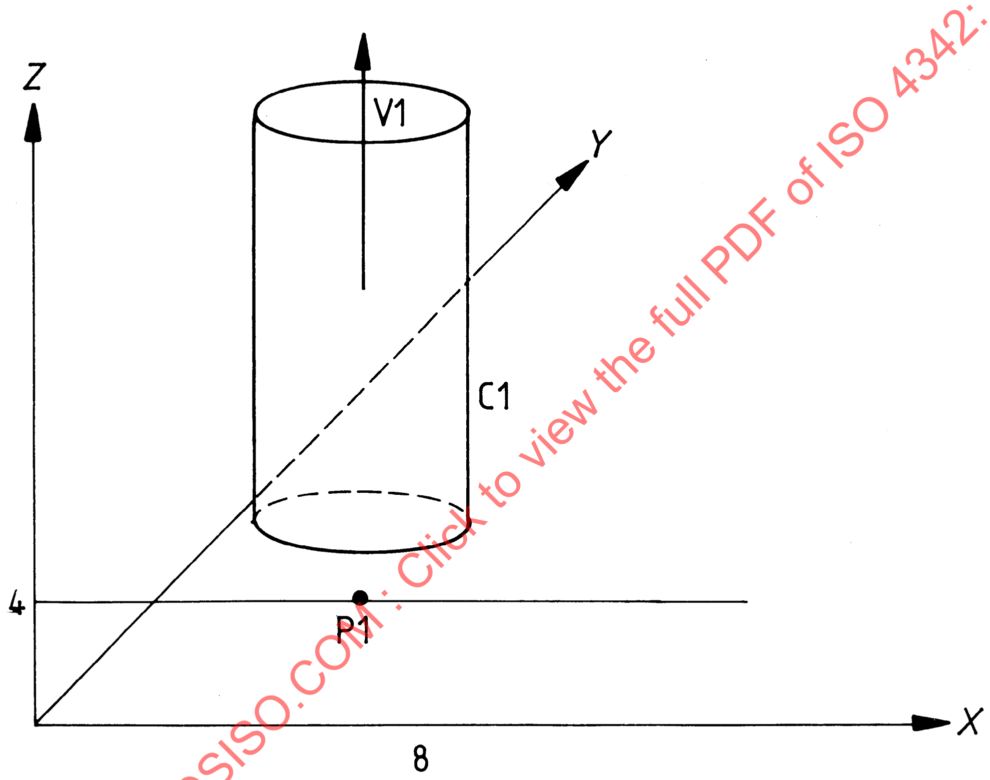
8.10.2.1 Semantics

Point is the symbol for a point on the axis of the cylinder; this may also be specified by the x, y, z coordinates.

Vector is the symbol for a vector in the direction of the axis; this may also be specified by the components of the unit vector in the direction of the axis.

r specifies the radius of the cylinder.

8.10.2.2 Example



C1 = CYLNDR/8, 0, 4, 0, 0, 1, 1

C1 = CYLNDR/P1, 0, 0, 1, 1

C1 = CYLNDR/P1, V1, 1

Figure 96

8.10.2.3 Limitations

None.

8.10.2.4 Syntax

< cylinder parameter list > : : = $\frac{6}{6}$ [< scalar > ,] < scalar > | < point spec > $\frac{4}{4}$ [, < scalar >] |
< point spec > , < vector spec > , < scalar >

8.10.3 Definition of a cylinder by two tangential planes and its radius

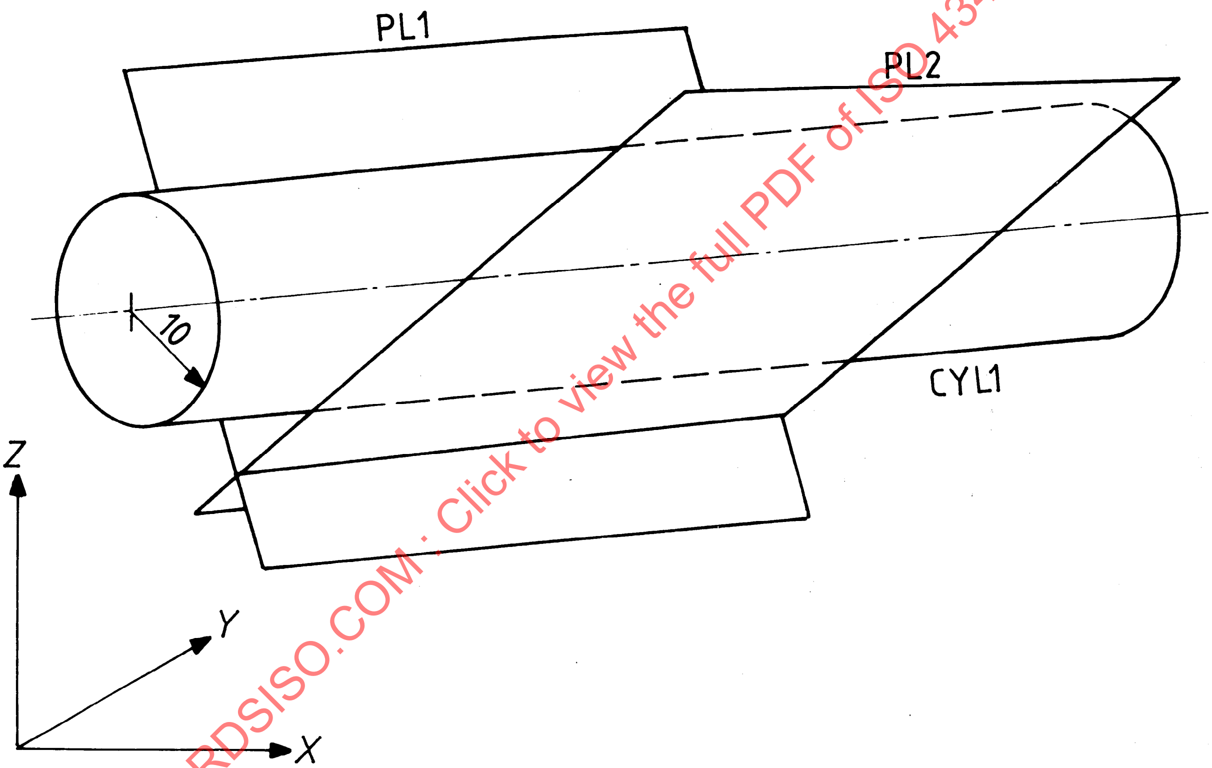
	XLARGE		XLARGE	
	XSMALL		XSMALL	
CYLNDR/	YLARGE YSMALL	, plane1,	YLARGE YSMALL	, plane2, RADIUS, *r*
	ZLARGE		ZLARGE	
	ZSMALL		ZSMALL	

8.10.3.1 Semantics

Plane1 and plane2 are the symbols for two planes.

RADIUS, *r* specifies the value of the radius of the cylinder.

8.10.3.2 Example



CYL1 = CYLNDR/ZLARGE, PL1, ZLARGE, PL2, RADIUS, 10

Figure 97

8.10.3.3 Limitations

XLARGE or XSMALL, YLARGE or YSMALL, respectively ZLARGE or ZSMALL, shall not be used when the associated line is parallel to the corresponding coordinate axis.

The two planes shall not be parallel.

8.10.3.4 Syntax

< cylinder parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL | ZLARGE | ZSMALL], < plane spec > ,
[XLARGE | XSMALL | YLARGE | YSMALL | ZLARGE | ZSMALL], < plane spec > ,
RADIUS, < scalar >

8.10.4 Definition of a cylinder by a tangential plane, tangential cylinder and its radius

```
          XLARGE          XLARGE
          XSMALL          XSMALL
CYLNDR/   YLARGE , plane, YLARGE , IN  , cylinder, RADIUS, r
          YSMALL          YSMALL   OUT
          ZLARGE          ZLARGE
          ZSMALL          ZSMALL
```

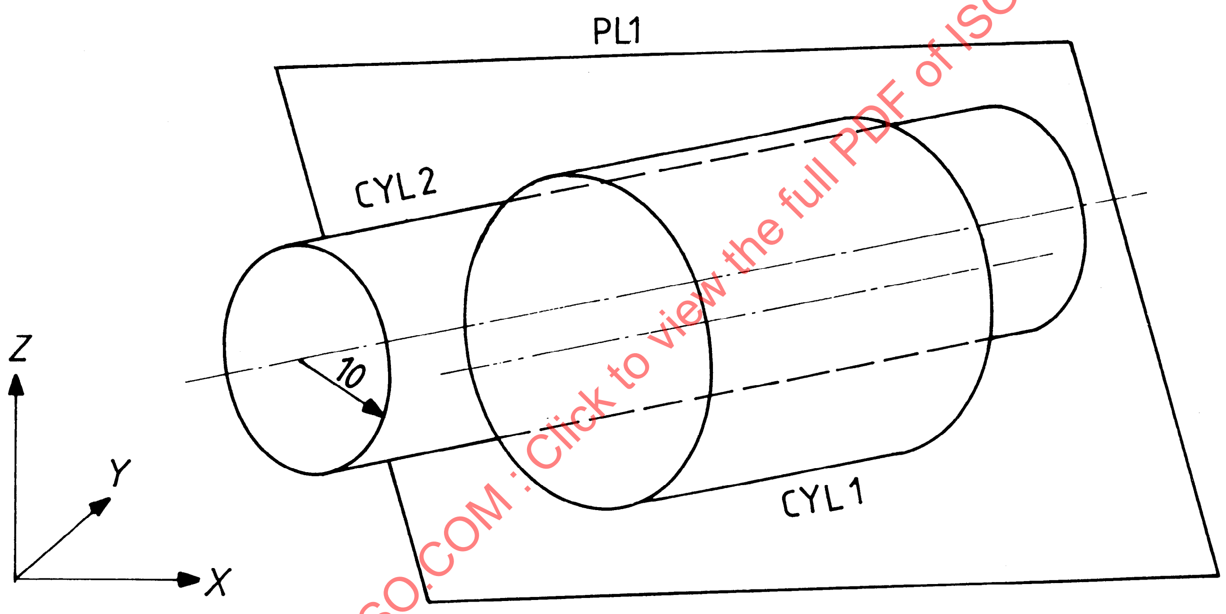
8.10.4.1 Semantics

Plane is the symbol for a plane.

Cylinder is the symbol for a cylinder.

RADIUS, *r* specifies the value of the radius of the cylinder.

8.10.4.2 Example



CYL2 = CYLNDR/XLARGE, PL1, ZLARGE, OUT, CYL1, RADIUS, 10

Figure 98

8.10.4.3 Limitations

The plane shall be parallel to the axis of the cylinder. The distance of the plane from the nearest point of the cylinder shall be less than twice the value of the radius.

8.10.4.4 Syntax

< cylinder parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL | ZLARGE | ZSMALL], < plane spec > ,
[XLARGE | XSMALL | YLARGE | YSMALL | ZLARGE | ZSMALL],[IN | OUT],
< cylinder spec > , RADIUS, < scalar >

8.10.5 Definition of a cylinder by two tangential cylinders and its radius

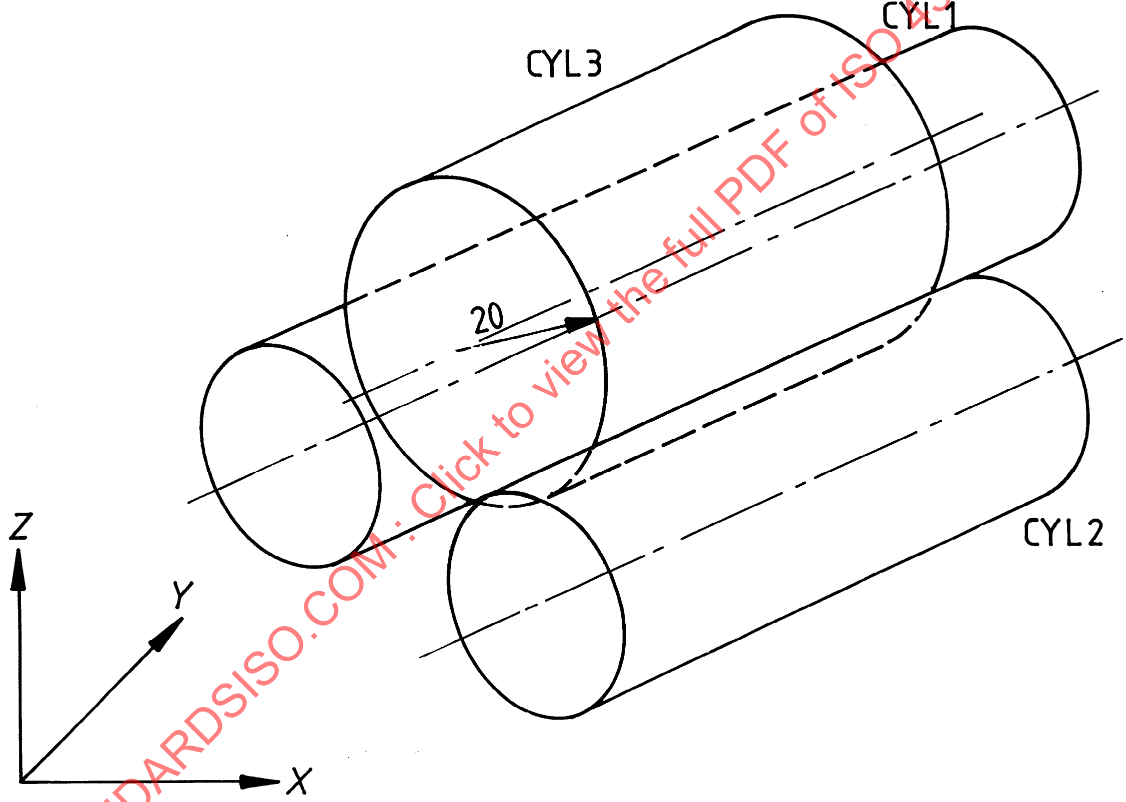
CYLNDR/ {XLARGE | XSMALL | YLARGE | YSMALL | ZLARGE | ZSMALL}, {IN | OUT}, cylinder1, {IN | OUT}, cylinder2, RADIUS, *r*

8.10.5.1 Semantics

Cylinder1 and cylinder2 are the symbols for two cylinders.

RADIUS, *r* specifies the value of the radius of the cylinder.

8.10.5.2 Example



CYL3 = CYLNDR/ZLARGE, OUT, CYL1, OUT, CYL2, RADIUS, 20

Figure 99

8.10.5.3 Limitations

The two cylinders shall be parallel. The modifiers used and the value of the radius shall correspond to a real and single solution.

8.10.5.4 Syntax

< cylinder parameter list > : : = [XLARGE | XSMALL | YLARGE | YSMALL | ZLARGE | ZSMALL],[IN | OUT], < cylinder spec > ,[IN | OUT], < cylinder spec > , RADIUS, < scalar >

8.11 Definitions of a sphere

8.11.1 Definition of a sphere

SPHERE/sphere parameter list

8.11.1.1 Semantics

A sphere is a surface such that all points on the surface are equidistant from a fixed point.

9.11.1.2 Limitations

The radius value shall be positive.

8.11.1.3 Sub-contents

For the definition of a sphere

a) by the coordinates of its centre and its radius, see 8.11.2;

b) by its centre point and its radius, see 8.11.3;

c) by its centre point and a point on the sphere, see 8.11.4;

d) by its centre point and a tangential plane, see 8.11.5;

e) by four points on the sphere, see 8.11.6.

8.11.1.4 Syntax

< sphere definition statement > : : = < identifier > = SPHERE/ < sphere parameter list >

8.11.2 Definition of a sphere by the coordinates of its centre and its radius

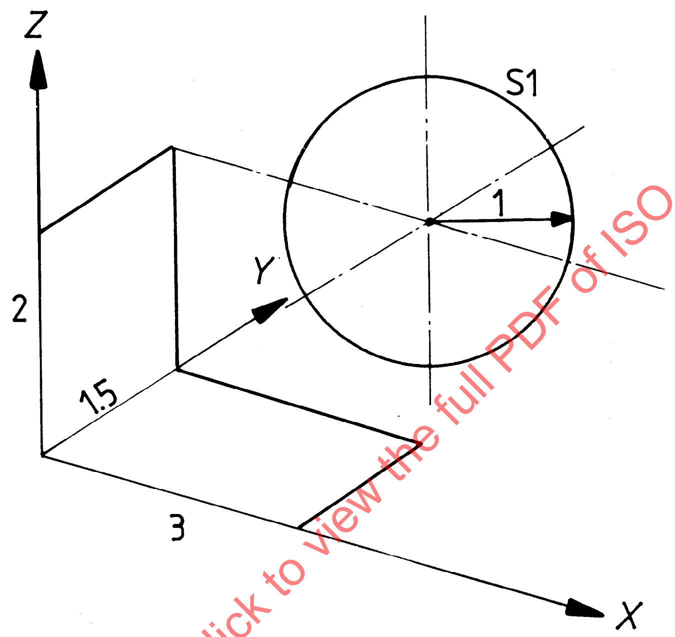
SPHERE/*x*, *y*, *z*, *r*

8.11.2.1 Semantics

x, *y*, *z* are the coordinate values of the centre.

r is the radius.

8.11.2.2 Example



S1 = SPHERE/3, 1.5, 2, 1

Figure 100

8.11.2.3 Limitations

None.

8.11.2.4 Syntax

< sphere parameter list > : : = < scalar > , < scalar > , < scalar > , < scalar >

8.11.3 Definition of a sphere by its centre point and its radius

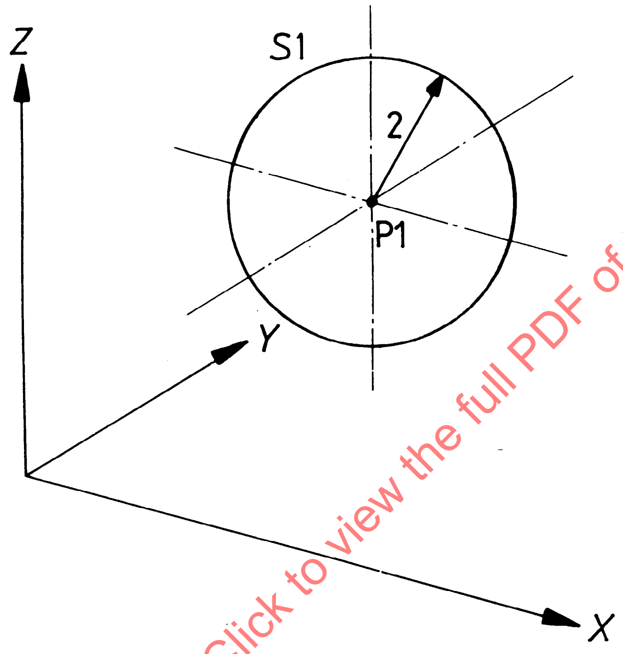
SPHERE/CENTER, point, RADIUS, *r*

8.11.3.1 Semantics

CENTER, point specifies that the centre of the required sphere is identified by the specified point.

RADIUS, *r* specifies the value of the radius of the sphere.

8.11.3.2 Example



S1 = SPHERE/CENTER, P1, RADIUS, 2

Figure 101

8.11.3.3 Limitations

None.

8.11.3.4 Syntax

< sphere parameter list > : : = CENTER, < point spec > , RADIUS, < scalar >

8.11.4 Definition of a sphere by its centre point and a point on the sphere

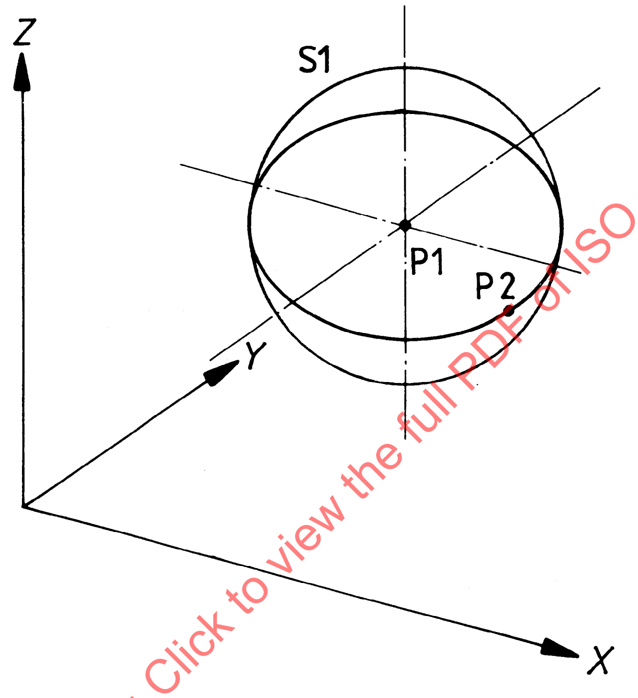
SPHERE/CENTER, point1, point2

8.11.4.1 Semantics

CENTER, point1 specifies that the centre of the required sphere is identified by the specified point1.

Point2 is the symbol for a point on the surface of the sphere.

8.11.4.2 Example



S1 = SPHERE/CENTER, P1, P2

Figure 102

8.11.4.3 Limitations

None.

8.11.4.4 Syntax

< sphere parameter list > : : = CENTER, < point spec > , < point spec >

8.11.5 Definition of a sphere by its centre point and a tangential plane

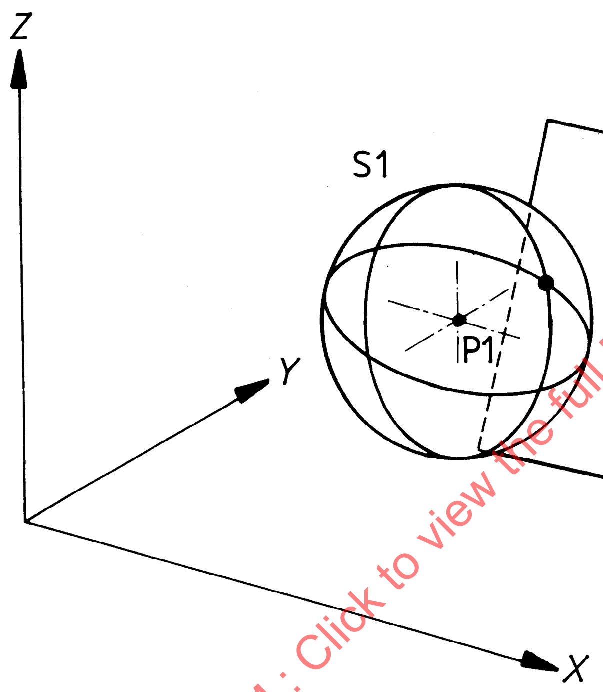
SPHERE/CENTER, point, TANTO, plane

8.11.5.1 Semantics

CENTER, point specifies that the centre of the required sphere is identified by the specified point.

Plane is the symbol for a plane tangential to the surface of the sphere.

8.11.5.2 Example



S1 = SPHERE/CENTER, P1, TANTO, PL1

Figure 103

8.11.5.3 Limitations

None.

8.11.5.4 Syntax

<sphere parameter list> : : = CENTER, <point spec> , TANTO, <plane spec>

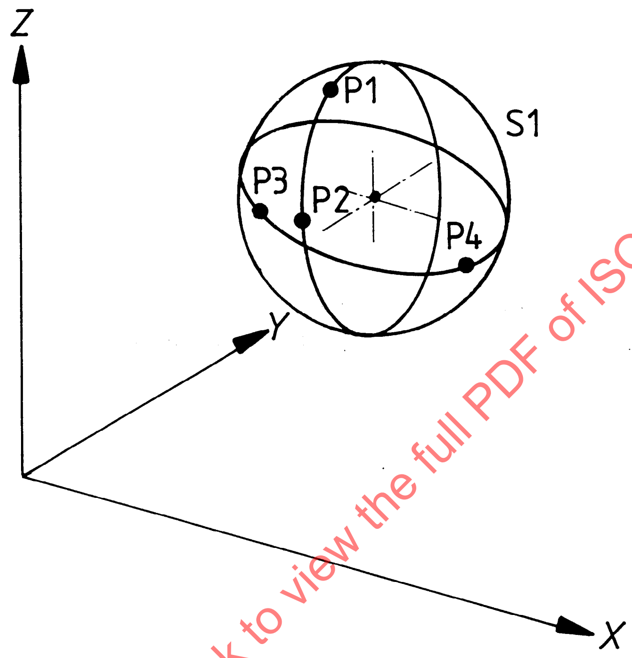
8.11.6 Definition of a sphere by four points on the sphere

SPHERE/point1, point2, point3, point4

8.11.6.1 Semantics

Point1, point2, point3, point4 are the symbols for four points lying on the surface of the sphere.

8.11.6.2 Example



S1 = SPHERE/P1, P2, P3, P4

Figure 104

8.11.6.3 Limitations

No two points shall be coincident nor shall three or more points be on a straight line.

8.11.6.4 Syntax

< sphere parameter list > : : = < point spec > $^{3}_{3}$ [, < point spec >]

8.12 Definitions of a cone

8.12.1 Definition of a cone

CONE/cone parameter list

8.12.1.1 Semantics

The vertex is the fixed point through which the generatrix passes. The half-angle is the constant angle made by the generatrix with the axis. Only one sheet is defined, the one in the direction of the axis vector.

8.12.1.2 Limitations

The half-angle shall be positive.

8.12.1.3 Sub-contents

For the definition of a cone

1) by its vertex point, its axis vector and its half-angle, see 8.12.2;
2) by the coordinates of its vertex, the vector components of its axis and the cosine of its half-angle, see 8.12.3.

8.12.1.4 Syntax

< cone definition statement > : : = < identifier > = CONE/ < cone parameter list >

8.12.2 Definition of a cone by its vertex point, its axis vector and its half-angle

CONE/point, vector, angle

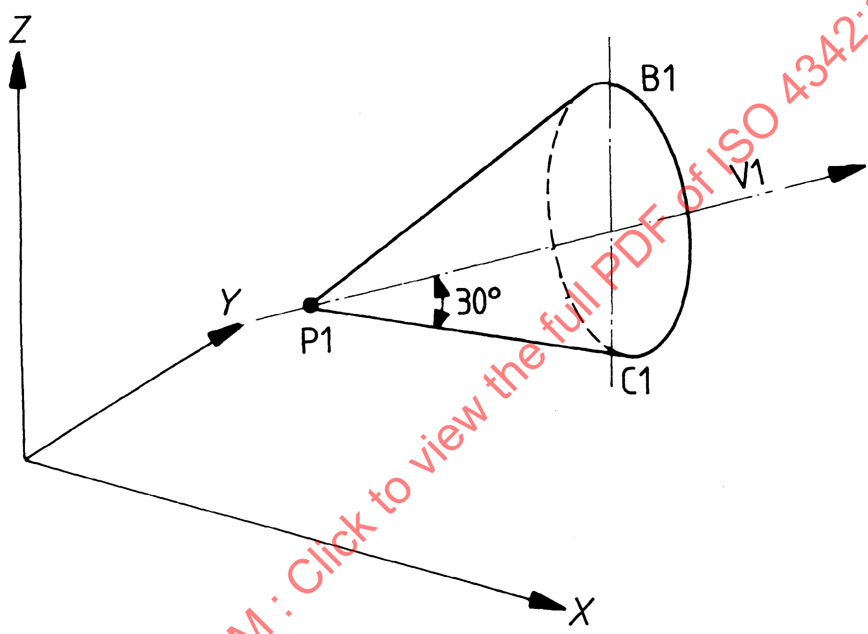
8.12.2.1 Semantics

Point is the symbol for a point and names the vertex.

Vector is the symbol for a vector and gives the direction of the axis.

Angle is the value in degrees of the half-angle.

8.12.2.2 Example



C1 = CONE/P1, V1, 30

Figure 105

8.12.2.3 Limitations

None.

8.12.2.4 Syntax

< cone parameter list > : : = < point spec > , < vector spec > , < scalar >

8.12.3 Definition of a cone by the coordinates of its vertex, the vector components of its axis and the cosine of its half-angle

CONE/x, y, z, a_x, a_y, a_z, cosine

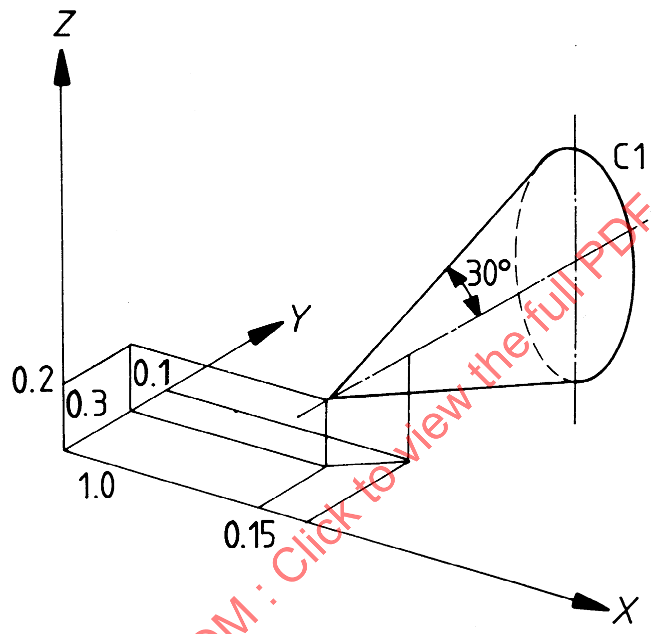
8.12.3.1 Semantics

x, y, z are the coordinate values of the vertex.

a_x, a_y, a_z are the component values in x, y, z respectively of a vector in the direction of the axis.

Cosine is the value of the cosine of the half-angle.

8.12.3.2 Example



C1 = CONE/1.0, 0.3, 0.2, 0.15, 0.1, 0, 0.866

Figure 106

8.12.3.3 Limitations

None.

8.12.3.4 Syntax

< cone parameter list > : : = < scalar > $^{6}_{6}$ [, < scalar >]

8.13 Definitions of an ellipse

8.13.1 Definition of an ellipse

ELLIPS/ellipse parameter list

8.13.1.1 Semantics

An ellipse is always to be considered as an elliptical cylinder perpendicular to the XY-plane.

An ellipse is the locus of a point which moves so that the sum of its distances from two fixed points is constant.

8.13.1.2 Limitations

None.

8.13.1.3 Sub-contents

For the definition of an ellipse by its centre point, length of its semi-major axis, length of its semi-minor axis and the angle made by the major axis with the X-axis, see 8.13.2.

8.13.1.4 Syntax

< ellipse definition statement > : : = < identifier > = ELLIPS/ < ellipse parameter list >

8.13.2 Definition of an ellipse by its centre point, length of its semi-major axis, length of its semi-minor axis and the angle made by the major axis with the X-axis

ELLIPS/CENTER, point, semi-major, semi-minor, angle

8.13.2.1 Semantics

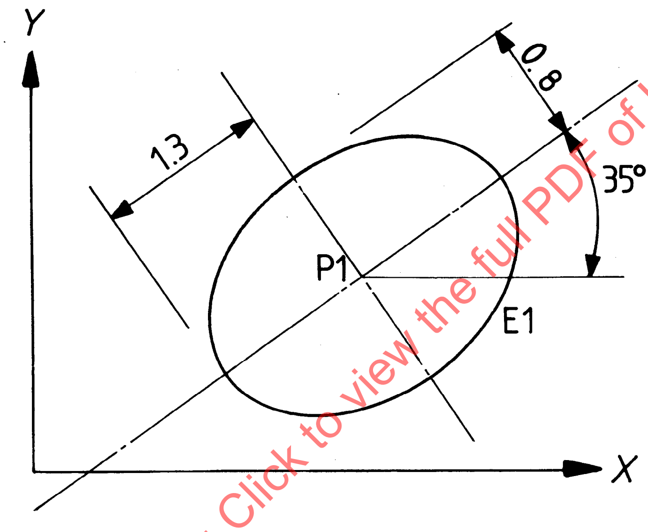
CENTER, point specifies that the centre of the ellipse, being equidistant from the two foci, is identified by the specified point.

Semi-major is the value of half the length of the major axis.

Semi-minor is the value of half the length of the minor axis.

Angle is the value of the angle made by the major axis with the X-axis.

8.13.2.2 Example



E1 = ELLIPS/CENTER, P1, 1.3, 0.8, 35

Figure 107

8.13.2.3 Limitations

None.

8.13.2.4 Syntax

< ellipse parameter list > : : = CENTER, < point spec > $^3_3$ [, < scalar >]

8.14 Definitions of a hyperbola

8.14.1 Definition of a hyperbola

HYPERB/hyperbola parameter list

8.14.1.1 Semantics

A hyperbola is the locus of a point which moves so that the difference between its distances from two fixed points is constant. Both sheets of the hyperbola are defined.

A hyperbola is always to be considered as a hyperbolical cylinder perpendicular to the XY-plane.

8.14.1.2 Limitations

None.

8.14.1.3 Sub-contents

For the definition of a hyperbola by the centre, the length of half the transverse axis, the length of half the conjugate axis and the angle between the transverse axis and the X-axis of the coordinate system, see 8.14.2.

8.14.1.4 Syntax

< hyperbola definition statement > : : = < identifier > = HYPERB/ < hyperbola parameter list >

8.14.2 Definition of a hyperbola by the centre, the length of half the transverse axis, the length of half the conjugate axis and the angle between the transverse axis and the *X*-axis of the coordinate system

HYPERB/CENTER, point, number1, number2, angle

8.14.2.1 Semantics

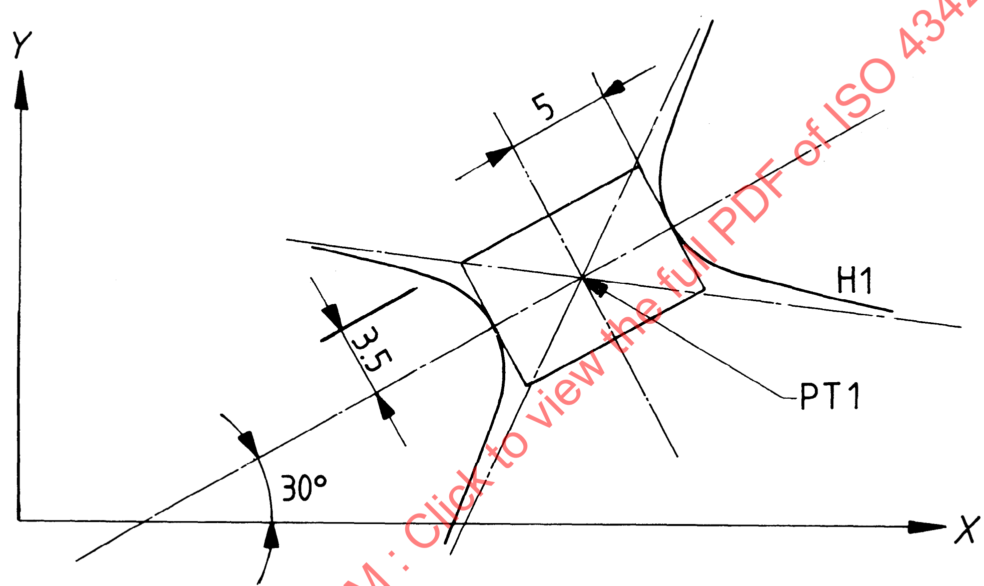
CENTER, point specifies that the centre of the hyperbola is identified by the specified point.

Number1 is the length of half the transverse axis.

Number2 is the length of half the conjugate axis.

Angle is the angle between the transverse axis and the *X*-axis of the coordinate system.

8.14.2.2 Example



H1 = HYPERB/CENTER, PT1, 5, 3.5, 30

Figure 108

8.14.2.3 Limitations

None.

8.14.2.4 Syntax

< hyperbola parameter list > : : = CENTER, < point spec > $^3_3$ [, < scalar >]

8.15 Definitions of a lofted conic

8.15.1 Definition of a lofted conic

LCONIC/lofted conic parameter list

8.15.1.1 Semantics

A lofted conic is a conic cylinder perpendicular to the XY-plane defined by points or a combination of points and slopes. A definition point may be a point symbol or a pair of x and y coordinates.

8.15.1.2 Limitations

None.

8.15.1.3 Sub-contents

For the definition of a lofted conic

a) by five points, see 8.15.2;

b) by four points and one slope, see 8.15.3;

c) by three points and two slopes, see 8.15.4.

8.15.1.4 Syntax

<lofted conic definition statement> : : = <identifier> = LCONIC/ <lofted conic parameter list>

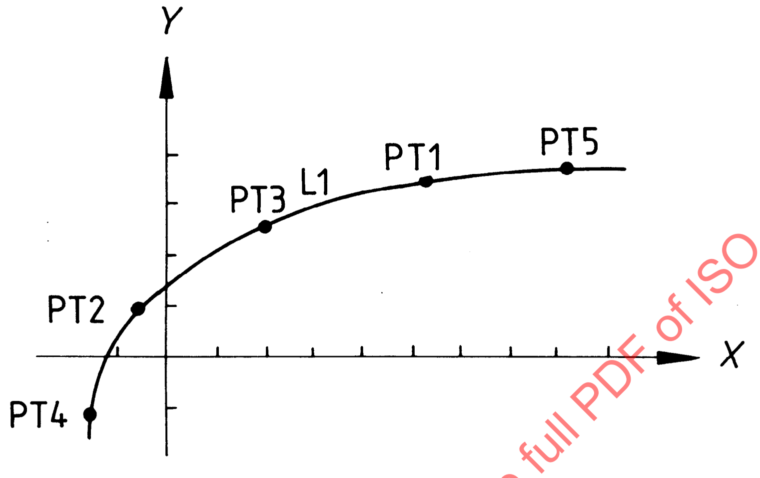
8.15.2 Definition of a lofted conic by five points

LCONIC/point1, point2, point3, point4, point5

8.15.2.1 Semantics

Point1 to point5 are the symbols for points from which the conic equation is derived.

8.15.2.2 Example



L1 = LCONIC/ − 1.3, − 1, − 0.7, 1, 2, 2.6, 5, 3.5, 8, 3.85

L1 = LCONIC/PT1, PT2, PT3, PT4, PT5

Figure 109

8.15.2.3 Limitations

None.

8.15.2.4 Syntax

< lofted conic parameter list > : : = < point option > $\frac{4}{4}$ [, < point option >]

8.15.3 Definition of a lofted conic by four points and one slope

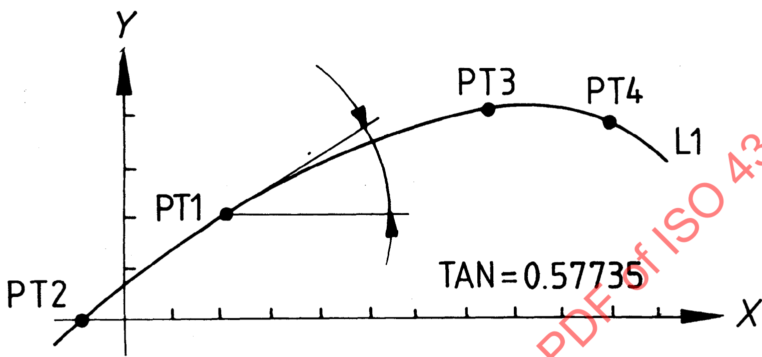
LCONIC/point1, SLOPE, slope, point2, point3, point4

8.15.3.1 Semantics

Point1 to point4 are the symbols for points from which the conic equation is derived.

SLOPE, slope specifies the slope of the tangent line at point1 to the conic, expressed as the tangent of the slope angle.

8.15.3.2 Example



L1 = LCONIC/2, 2, SLOPE, 0.57735, − 1, 0, 7.5, 4.2, 10, 4

L1 = LCONIC/PT1, SLOPE, 0.57735, PT2, PT3, PT4

Figure 110

8.15.3.3 Limitations

None.

8.15.3.4 Syntax

< lofted conic parameter list > : : = < point option > , SLOPE, < scalar > , $^3_3$ [, < point option >]

8.15.4 Definition of a lofted conic by three points and two slopes

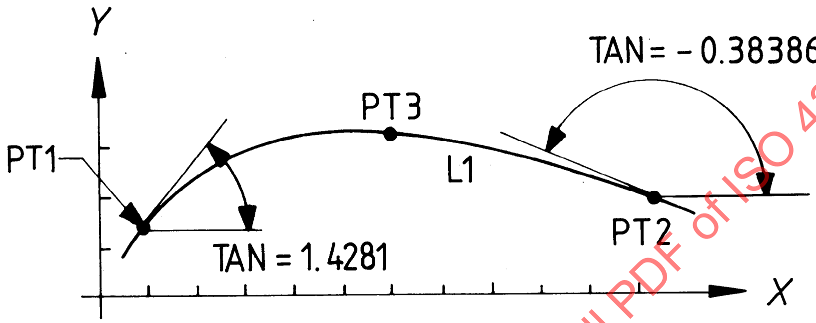
LCONIC/point1, SLOPE, slope1, point2, SLOPE, slope2, point3

8.15.4.1 Semantics

Point1 to point3 are the symbols for points from which the conic equation is derived.

SLOPE, slope1 specifies the slope of the tangent line at point1. SLOPE, slope2 specifies the slope of the tangent line at point2. The slope values are expressed as the tangents of the slope angles.

8.15.4.2 Example



L1 = LCONIC/1, 1.35, SLOPE, 1.4281, 11.25, 2, SLOPE, – 0.38386, 6, 3.25

L1 = LCONIC/PT1, SLOPE, 1.4281, PT2, SLOPE, – 0.38386, PT3

Figure 111

8.15.4.3 Limitations

Point1 and point2 shall be end points and point3 shall lie between them.

8.15.4.4 Syntax

< lofted conic parameter list > : : = < point option > , SLOPE, < scalar > ,
< point option > , SLOPE, < scalar > , < point option >

8.16 Definitions of a general conic

8.16.1 Definition of a general conic

GCONIC/general conic parameter list

8.16.1.1 Semantics

A general conic is a conic cylinder perpendicular to the XY-plane defined by coefficients of a conic equation.

8.16.1.2 Limitations

None.

8.16.1.3 Sub-contents

For the definition of a general conic

a) by the general conic equation, see 8.16.2;

b) by an equation where y is expressed as a function of x, see 8.16.3;

c) by an equation where x is expressed as a function of y, see 8.16.4;

8.16.1.4 Syntax

<general conic definition statement> : = <identifier> = GCONIC/ <general conic parameter list>

8.16.2 Definition of a conic by its general mathematical equation

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

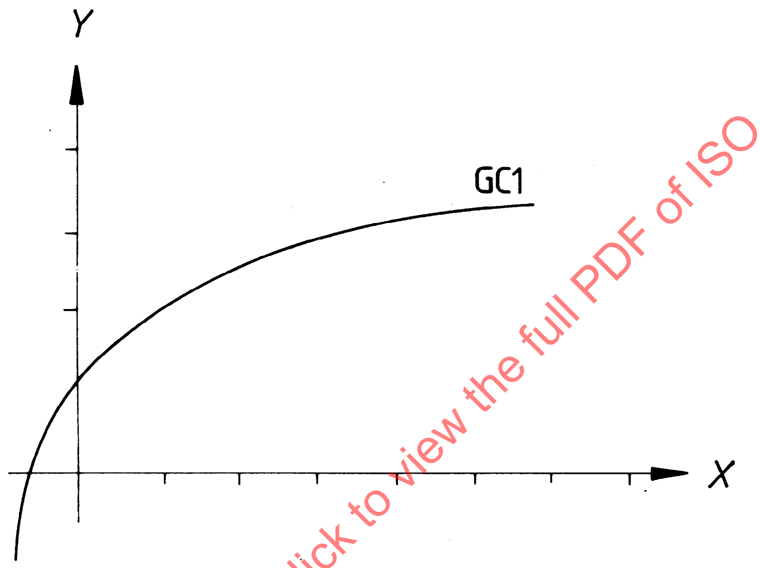
8.16.2.1 Semantics

A, B, C, D, E and F are the coefficients of the conic equation.

The equation has the following form :

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

8.16.2.2 Example



GC1 = GCONIC/3, 0, 8, – 39, 16, – 22.6

Figure 112

8.16.2.3 Limitations

None.

8.16.2.4 Syntax

< general conic parameter list > : : = < scalar > $^{5}_{5}$ [, < scalar >]

8.16.3 Definition of a conic by a mathematical equation where y is expressed as a function of x

GCONIC/P, Q, R, S, T

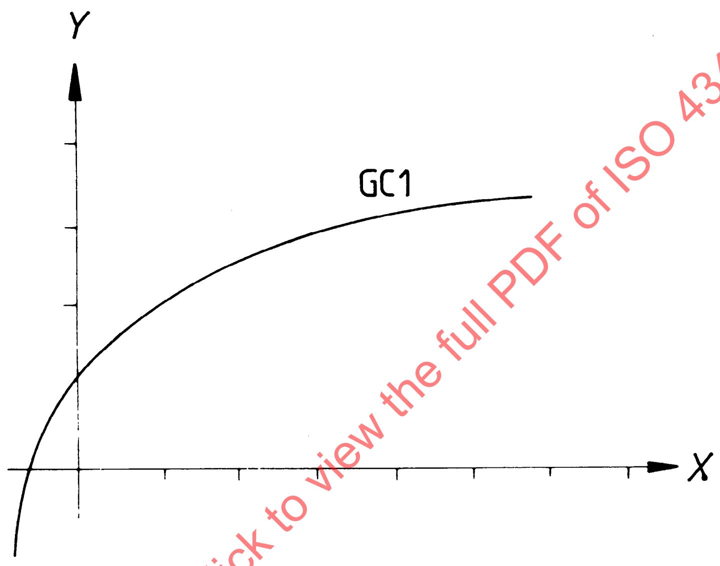
8.16.3.1 Semantics

P, Q, R, S and T are the coefficients of the conic equation.

The equation is of the following form :

$$y = Px + Q \pm \sqrt{Rx^2 + Sx + T}$$

8.16.3.2 Example



GC1 = GCONIC/0, – 1, – .4, 4.9, 3.8

Figure 113

8.16.3.3 Limitations

None.

8.16.3.4 Syntax

< general conic parameter list > : : = < scalar > $^{4}_{4}$ [, < scalar >]

8.16.4 Definition of a conic by a mathematical equation where x is expressed as a function of y

GCONIC/P, Q, R, S, T, FUNOFY

8.16.4.1 Semantics

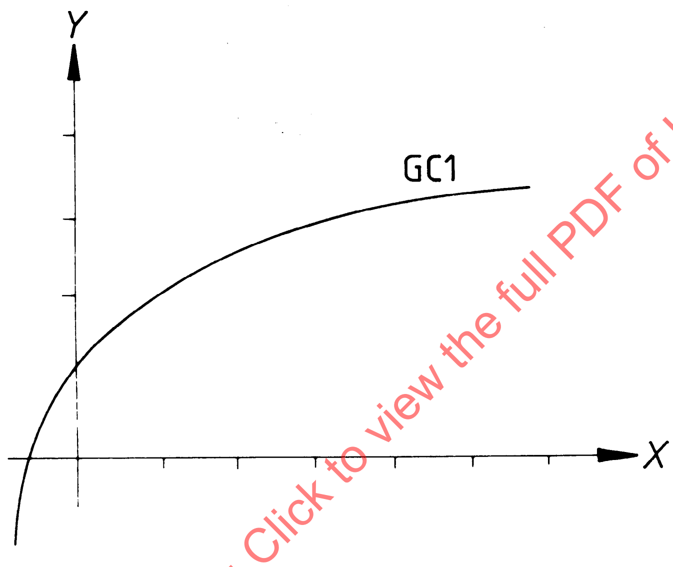
P, Q, R, S and T are the coefficients of the conic equation.

FUNOFY indicates that the equation defines x as a function of y.

The equation has the following form :

$$x = Py + Q \pm \sqrt{Ry^2 + Sy + T}$$

8.16.4.2 Example



GC1 = GCONIC/0, 6.5, – 2.7, – 5.3, 50, FUNOFY

Figure 114

8.16.4.3 Limitations

None.

8.16.4.4 Syntax

< general conic parameter list > : : = < scalar > $^{4}_{4}$ [, < scalar >], FUNOFY

8.17 Definitions of a general quadric

8.17.1 Definition of a general quadric

QUADRIC/quadric parameter list

8.17.1.1 Semantics

A general quadric is a surface defined by a quadratic equation with three variables.

8.17.1.2 Limitations

None.

8.17.1.3 Sub-contents

For the definition of a quadric by the coefficients of a quadric equation, see 8.17.2.

8.17.1.4 Syntax

< quadric definition statement > : : = < identifier > = QUADRIC/ < quadric parameter list >

8.17.2 Definition of a quadric surface by the ten coefficients of the quadric equation

QUADRIC/number1, number2, ..., number10

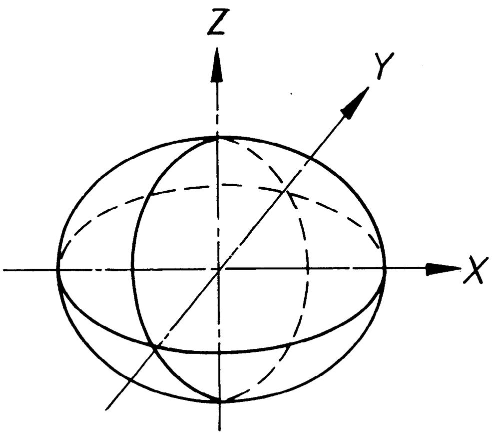
8.17.2.1 Semantics

The general equation of a quadric surface is

$$Ax^2 + By^2 + Cz^2 + Fyz + Gzx + Hxy + Px + Qy + Rz + D = 0$$

Number1 to number10 are the coefficients $A, B, C, F, G, H, P, Q, R$ and D in that order.

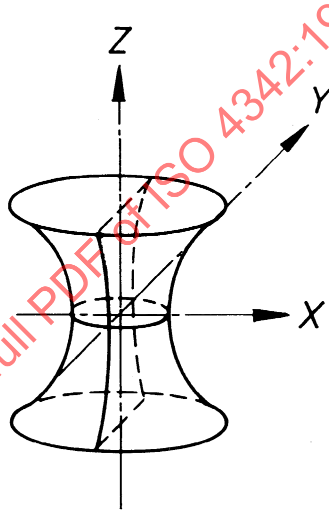
8.17.2.2 Examples



Real ellipsoid

$$\frac{X^2}{A^2} + \frac{Y^2}{B^2} + \frac{Z^2}{C^2} = 1$$

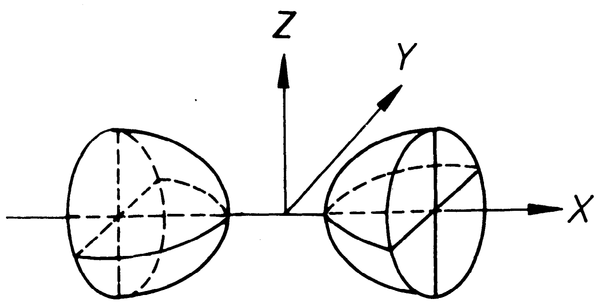
Figure 115a)



Hyperboloid of one sheet

$$\frac{X^2}{A^2} + \frac{Y^2}{B^2} - \frac{Z^2}{C^2} = 1$$

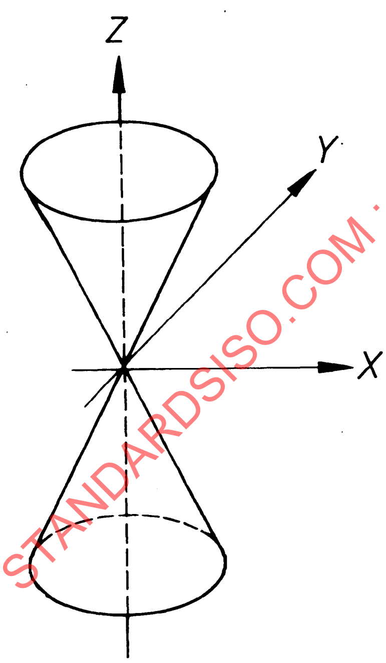
Figure 115b)



Hyperboloid of two sheets

$$\frac{X^2}{A^2} - \frac{Y^2}{B^2} - \frac{Z^2}{C^2} = 1$$

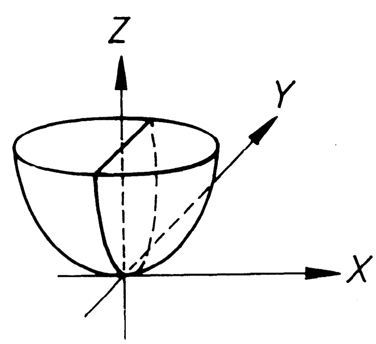
Figure 115c)



Elliptic cone

$$\frac{X^2}{A^2} + \frac{Y^2}{B^2} - \frac{Z^2}{C^2} = 0$$

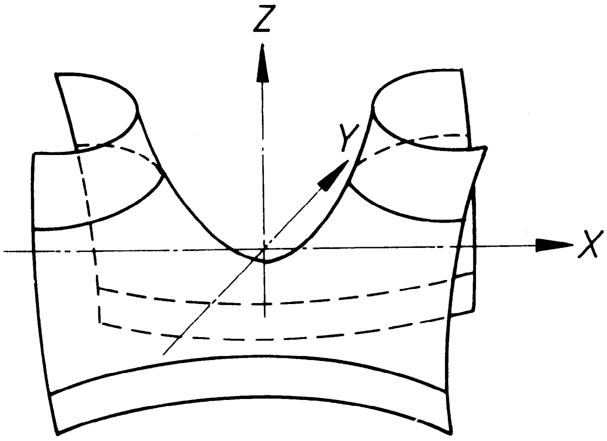
Figure 115d)



Elliptic paraboloid

$$\frac{X^2}{A^2} + \frac{Y^2}{B^2} = 2CZ$$

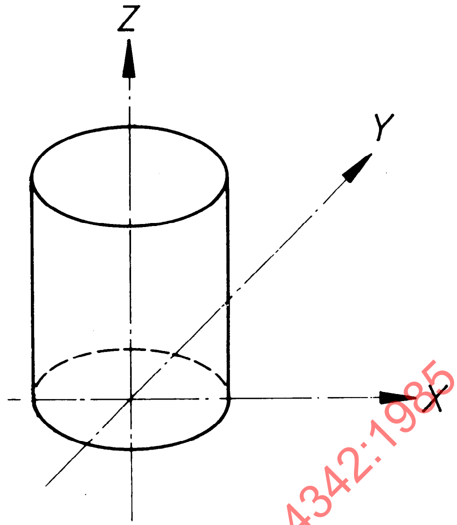
Figure 115e)



Hyperbolic paraboloid

$$\frac{X^2}{A^2} - \frac{Y^2}{B^2} = 2CZ$$

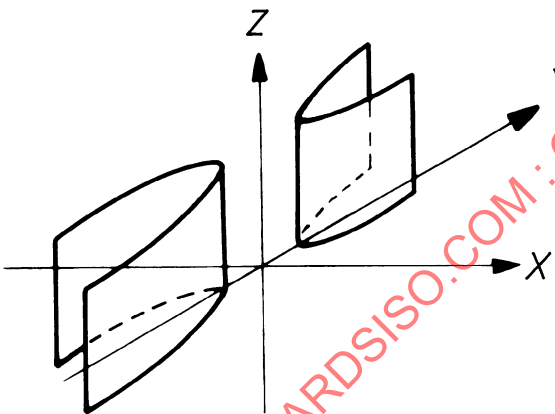
Figure 115f)



Elliptic cylinder

$$\frac{X^2}{A^2} + \frac{Y^2}{B^2} = 1$$

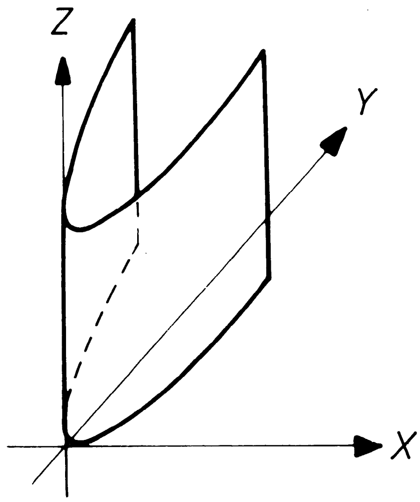
Figure 115g)



Hyperbolic cylinder

$$\frac{X^2}{A^2} - \frac{Y^2}{B^2} = 1$$

Figure 115h)



Parabolic cylinder

$$Y^2 = PX$$

Figure 115j)

8.17.2.3 Limitations

None.

8.17.2.4 Syntax

< quadric parameter list > : : = $^{9}_{9}$ [< scalar > ,] < scalar >

8.18 Definitions of a tabulated cylinder

8.18.1 Definition of a tabulated cylinder

TABCYL/tabcyl1 parameter list

8.18.1.1 Semantics

A tabulated cylinder TABCYL is the cylindrical surface generated by a line (the generatrix) moving parallel to itself along a space curve (the directrix). This curve is defined by a table of points known to lie on the surface and a number of interpolation methods for "fairing" between the points. After the curve has been constructed, the tabulated cylinder is extended at each end by lines tangential to the curve through the first and last points. These extension lines have finite length, the value of which is implementation dependent.

A tabulated cylinder has direction which is from the first mentioned point in its definition to the last point.

The commonly used method of construction SPLINE is a series of cubics which result in a curve passing through each of the points with tangency and curvature continuity at each point. Its analogy in physical terms is a thin wooden spline bent to pass through a series of given points.

TRFORM, matrix indicates transformation of tabulated cylinder input data by the specified matrix. When the X, Y, Z data format is used, the generatrix is transformed. When other formats are used, only points, slopes, and normals are transformed.

8.18.1.2 Limitations

Minimum specification of a TABCYL should contain three points or two points and a normal condition or a slope.

With a tabulated cylinder defined whose directrix is in a plane (NOX, NOY, NOZ), any transformation matrix is limited to transformations within that plane.

8.18.1.3 Sub-contents

For the definition of a tabulated cylinder

a) by a series of points or x, y coordinates with optional transformation matrix and optional end slope specification, see 8.18.2;

b) by a series of polar coordinates with optional transformation matrix and optional end slope specification, see 8.18.3;

c) by a series of points, a vector to which the generatrix is parallel and an optional transformation matrix, see 8.18.4;

d) at a distance from a parallel tabulated cylinder, see 8.18.5.

8.18.1.4 Syntax

< tabulated cylinder definition statement > : : = < identifier > = TABCYL/ < tabulated cylinder parameter list >

8.18.2 Definition of a tabulated cylinder by a series of points or coordinates of points with optional transformation matrix and optional end slope specification

$$\text{TABCYL/}\begin{array}{l}\text{NOX}\\ \text{NOY}\\ \text{NOZ}\end{array}\text{, SPLINE, TRFORM, matrix,}\begin{array}{l}\text{point1}\\ a_1,\ b_1\end{array}\text{,}\begin{array}{l}\text{NORMAL, angle1}\\ \text{SLOPE, slope1}\\ \text{TANTO, line1}\end{array}\text{,}\begin{array}{l}\text{point2}\\ a_2,\ b_2\end{array}\text{, ... ,}\begin{array}{l}\text{point}n\\ a_n,\ b_n\end{array}\text{,}\begin{array}{l}\text{NORMAL, angle}n\\ \text{SLOPE, slope}n\\ \text{TANTO, line}n\end{array}$$

8.12.2.1 Semantics

NOX specifies that no x coordinate is utilised from the point descriptions and that the Y-axis is the reference axis; similarly NOY indicates no y coordinate and that the Z-axis is the reference axis and NOZ indicates no z coordinate and that the X-axis is the reference axis.

Point1 to point$n$ are the symbols for points.

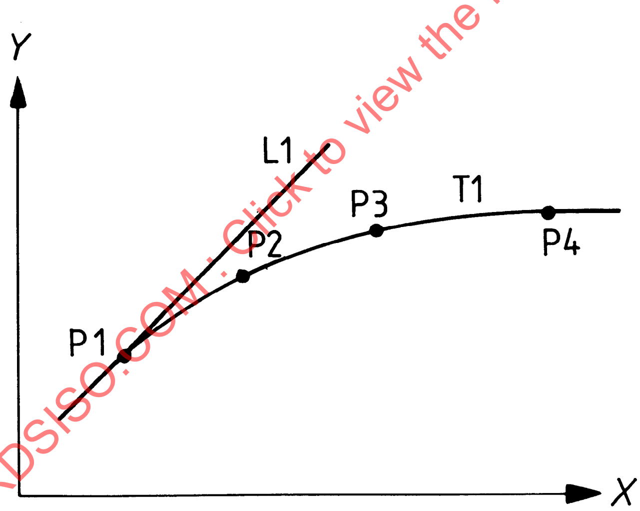
a_1, b_1 to a_n, b_n are the Cartesian coordinates for points, dependent on the NOX, NOY, NOZ modifier.

NORMAL, angle1 and NORMAL, angle$n$ specify the angles between the reference axis and the direction of the normals to the first and last points respectively.

SLOPE, slope1 and SLOPE, slope$n$ specify the tangent of the angles between the reference axis and the tabulated cylinder at the first and last points respectively.

TANTO, line1 and TANTO, line$n$ specify a line to which the tabulated cylinder is tangential at the beginning and at the end respectively.

8.18.2.2 Example



T1 = TABCYL/NOZ, SPLINE, P1, TANTO, L1, P2, P3, P4

Figure 116

8.18.2.3 Limitations

A tangent line shall only be defined in the NOZ mode, not in the NOX nor NOY mode.

8.18.2.4 Syntax

< tabulated cylinder parameter list > : : = [NOX | NOY | NOZ], SPLINE $_0^1$ [, TRFORM, < matrix spec >],
< point option > $_0^1$ [,TANTO, < line spec > | ,SLOPE, < scalar > |
,NORMAL, < scalar >] $_1^{137}$ [, < point option >] $_0^1$ [,TANTO,
< line spec > | ,SLOPE, < scalar > | ,NORMAL, < scalar >]

8.18.3 Definition of a tabulated cylinder by polar coordinates of a series of points with optional transformation matrix and optional end slope specification

TABCYL/ {RTHETA | THETAR}, SPLINE, TRFORM, matrix, a_1, b_1, {NORMAL, angle1 | SLOPE, slope1 | TANTO, line1}, a_2, b_2, ..., a_n, b_n, {NORMAL, angle*n* | SLOPE, slope*n* | TANTO, line*n*}

8.18.3.1 Semantics

RTHETA specifies that the radial distance is given before the angle and THETAR that the angle is given before the radial distance.

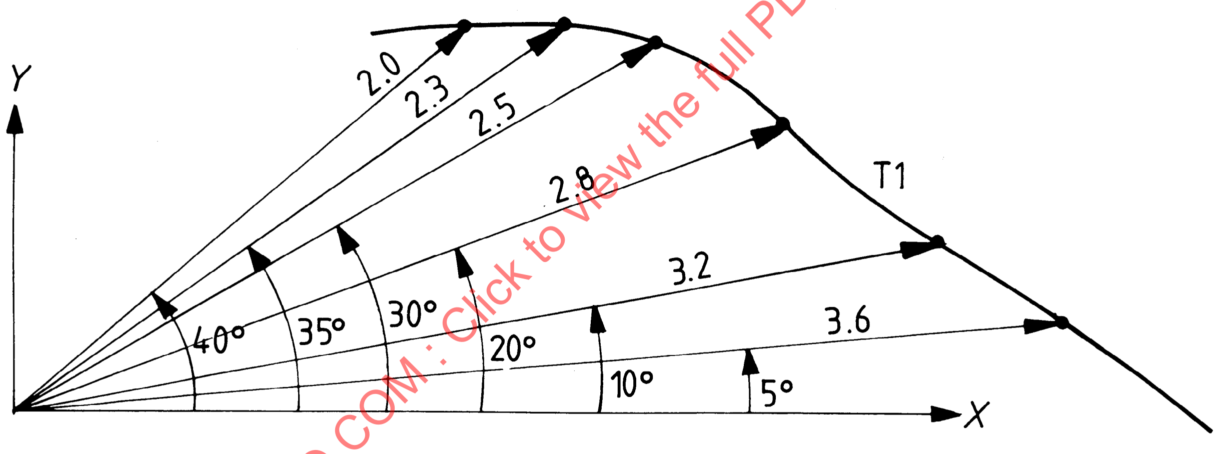
a_1, b_1 to a_n, b_n are the polar coordinates of points dependent on RTHETA or THETAR.

NORMAL, angle1 and NORMAL, angle*n* specify the angles between the X-axis and the directions of the normals of the first and last points respectively.

SLOPE, slope1 and SLOPE, slope*n* specify the tangent of angles between the X-axis and the tabulated cylinder at the first and last points respectively.

TANTO, line1 and TANTO, line*n* specify a line to which the tabulated cylinder is tangential at the beginning and at the end respectively.

8.18.3.2 Example



T1 = TABCYL/RTHETA, SPLINE, 2, 40, 2.3, 35, 2.5, 30, 2.8, 20, 3.2, 10, 3.6, 5

Figure 117

8.18.3.3 Limitations

A tabulated cylinder shall only be defined perpendicular to the XY-plane.

8.18.3.4 Syntax

< tabulated cylinder parameter list > : : = [RTHETA | THETAR], SPLINE $_0^1$ [,TRFORM, < matrix spec >],
< point option > $_0^1$ [,TANTO, < line spec > | ,SLOPE, < scalar > |
,NORMAL, < scalar >] $_1^{137}$ [, < point option >] $_0^1$ [,TANTO,
< line spec > | ,SLOPE, < scalar > | ,NORMAL, < scalar >]

8.18.4 Definition of a tabulated cylinder by a series of points, a vector to which the generatrix is parallel and an optional transformation matrix

TABCYL/XYZ, SPLINE, TRFORM, matrix, vector, point1 x_1, y_1, z_1, ..., point$n$ x_n, y_n, z_n

8.18.4.1 Semantics

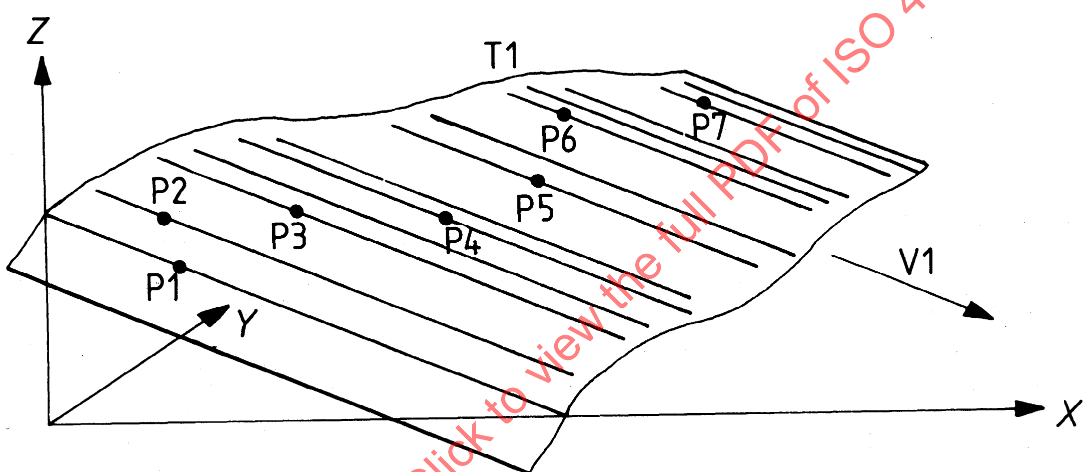
XYZ specifies that x, y and z coordinate values are required.

Vector is the symbol for a vector, which specifies the direction of the tabulated cylinder.

Point1 to point$n$ are the symbols for points.

x_1, y_1, z_1 to x_n, y_n, z_n are the coordinates of points.

8.18.4.2 Example



T1 = TABCYL/XYZ, SPLINE, V1, P1, P2, P3, P4, P5, P6, P7

Figure 118

8.18.4.3 Limitations

A minimum of four points are required with x, y and z coordinates of each point being indicated.

8.18.4.4 Syntax

< tabulated cylinder parameter list > : : = XYZ, SPLINE $_0^1$ [,TRFORM, < matrix spec >], < vector spec > $_3^{138}$ [, < point spec > | , < scalar > , < scalar > , < scalar >]

8.18.5 Definition of a tabulated cylinder at a distance from a parallel tabulated cylinder

$$\text{TABCYL/PARLEL, tabcyl, point, } \begin{array}{l} \text{XLARGE} \\ \text{XSMALL} \\ \text{YLARGE} \\ \text{YSMALL} \\ \text{ZLARGE} \\ \text{ZSMALL} \end{array} , \ d$$

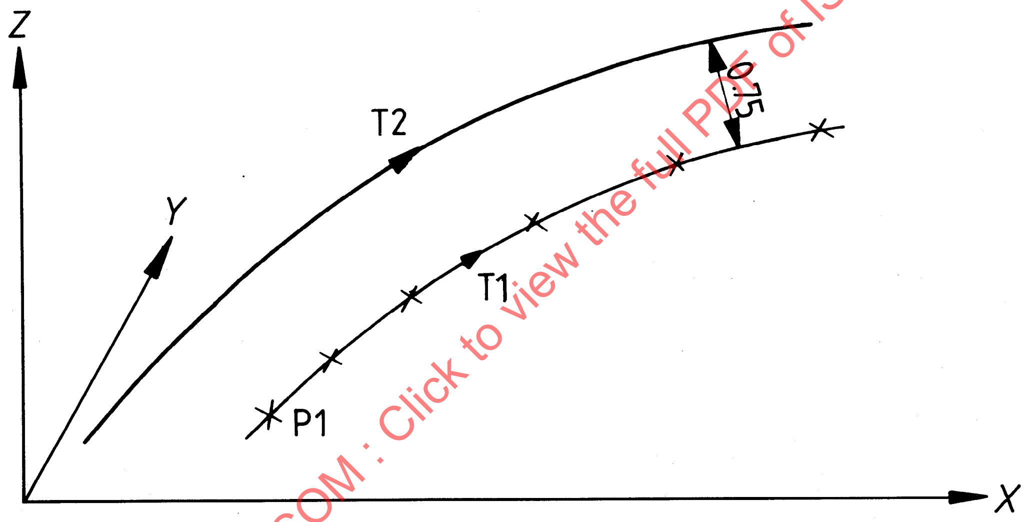
8.18.5.1 Semantics

Tabcyl is the symbol for a tabulated cylinder.

d is the distance from another tabulated cylinder.

Point is the symbol for a point where the modifier will eventually be considered.

8.18.5.2 Example



T2 = TABCYL/PARLEL, T1, P1, XSMALL, 0.75

Figure 119

8.18.5.3 Limitations

All points on the previously defined tabulated cylinder must be coplanar.

Parallelism for tabulated cylinders higher than second order is approximate.

8.18.5.4 Syntax

< tabulated cylinder parameter list > : : = PARLEL, < tabulated cylinder spec > $^{1}_{0}$ [, < point spec >],
[XLARGE | XSMALL | YLARGE | YSMALL | ZLARGE | ZSMALL], < scalar >

8.19 Definitions of a matrix

8.19.1 Definition of a matrix

MATRIX/matrix parameter list

8.19.1.1 Semantics

A matrix is a set of twelve parameters. These parameters are the coefficients of three linear equations defining the mathematical relationship between two coordinate systems.

By convention in use with REFSYS for defining geometry, the new system is the one in which the coordinates are given or known and the old system is the one in which the new coordinates have to be established.

8.19.1.2 Limitations

None.

8.19.1.3 Sub-contents

For the definition of a matrix

a) by twelve parameters, see 8.19.2;

b) by defining the translation of the old coordinate system into the new system, see 8.19.3;

c) by defining the rotation in the referenced plane of the old coordinate system into the new system, see 8.19.4;

d) by defining the scaling of the old coordinate system into the new system, see 8.19.5;

e) by defining a symmetry with respect to a line or a plane, see 8.19.6;

f) by defining a symmetry with respect to the referenced plane or the intersection of the referenced planes, see 8.19.7;

g) by three planes, see 8.19.8;

h) by a point and two vectors, see 8.19.9;

j) as the product of two matrices, see 8.9.10;

k) as the inverse of a given matrix, see 8.19.11;

8.19.1.4 Syntax

< matrix definition statement > : : = < identifier > = MATRIX/ < matrix parameter list >

8.19.2 Definition of a matrix by twelve parameters

MATRIX/a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2, a_3, b_3, c_3, d_3

8.19.2.1 Semantics

MATRIX is a rectangular array of numbers consisting of three rows of four numbers each, used to define the relationship of one coordinate system to another, through the system :

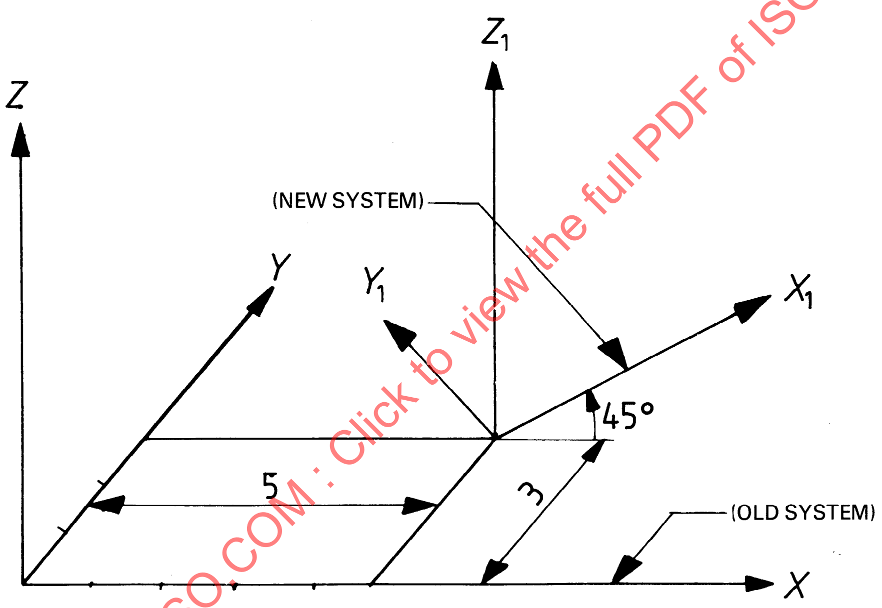
$$x = a_1x_1 + b_1y_1 + c_1z_1 + d_1$$

$$y = a_2x_1 + b_2y_1 + c_2z_1 + d_2$$

$$z = a_3x_1 + b_3y_1 + c_3z_1 + d_3$$

Made equal to zero, these three equations represent respectively the canonical forms of the three reference planes of the new system with respect to the old system of coordinates.

8.19.2.2 Example



M1 = MATRIX/0.707, 0.707, 0, 5, − 0.707, 0.707, 0, 3, 0, 0, 1, 0

Figure 120

In this example, the X-axis of the new coordinate system has direction cosines of 0.707, 0.707, 0, relative to the old coordinate system. The Y- and Z-axes have direction cosines of − 0.707, 0.707, 0 and 0, 0, 1, respectively.

The origin of the new system has coordinates 5,3,0, in the old system.

8.19.2.3 Limitations

With no scaling, $a_n^2 + b_n^2 + c_n^2 = 1$. Scaling is accomplished by scaling a_n, b_n, c_n coefficients (see 8.19.5). Any scaling should normally be equal on all axes in order to achieve predictable transformations, especially of canonical forms.

8.19.2.4 Syntax

<matrix parameter list> : : = <scalar> $^{11}_{11}$ [, <scalar>]

8.19.3 Definition of a matrix by defining the translation of the old coordinate system into the new system

MATRIX/TRANSL, d_1, d_2, d_3

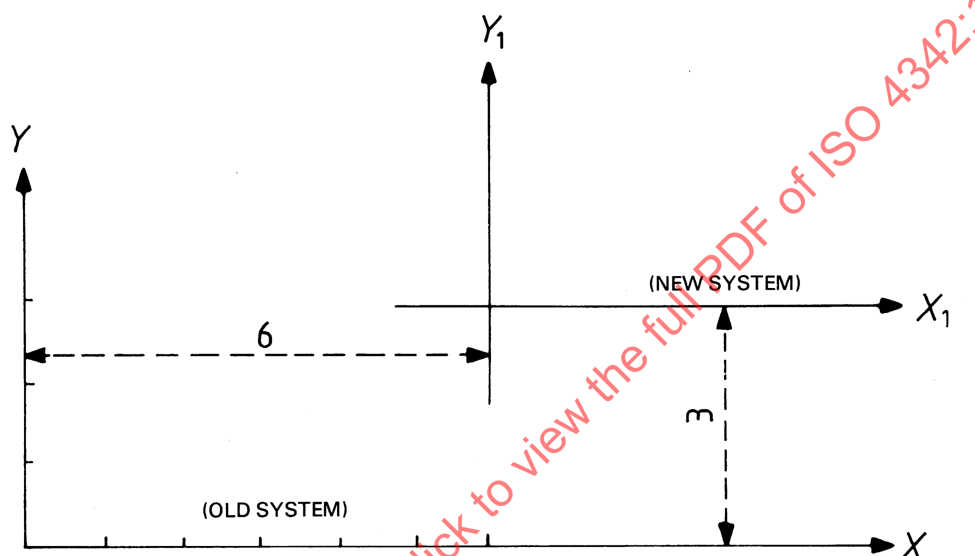
8.19.3.1 Semantics

TRANSL specifies that the new coordinate system is created by translating the old system.

The parameters d_1, d_2 and d_3 are the coordinates of the origin of the new coordinate system in the old system.

When d_3 is omitted, $d_3 = 0$ is assumed.

8.19.3.2 Example



M3 = MATRIX/TRANSL, 6, 3, 0

M3 = MATRIX/1, 0, 0, 6, 0, 1, 0, 3, 0, 0, 1, 0

Figure 121

8.19.3.3 Limitations

None.

8.19.3.4 Syntax

< matrix parameter list > : : = TRANSL $^3_2$ [, < scalar >]

8.19.4 Definition of a matrix by defining the rotation in the referenced plane of the old coordinate system into the new system

```
         XYROT
MATRIX/  YZROT  , angle
         ZXROT
```

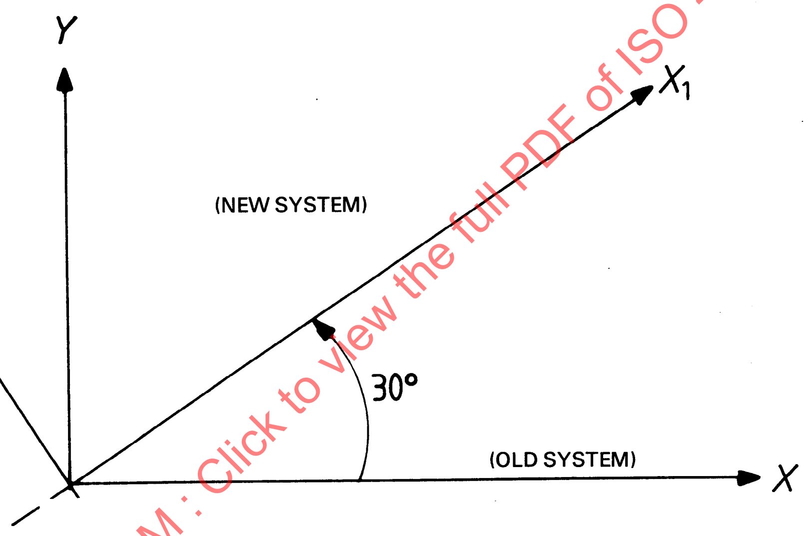
8.19.4.1 Semantics

XYROT specifies that the new coordinate system is created by a rotation of the old system in the XY-plane, YZROT a rotation in the YZ-plane and ZXROT a rotation in the ZX-plane.

Angle represents the algebraic value of the rotation in the referenced plane.

The positive sense of rotation is defined from the first referenced axis to the second axis.

8.19.4.2 Example



M5 = MATRIX/XYROT, 30.0

M5 = MATRIX/0.886, − 0.5, 0, 0, 0.5, 0.866, 0, 0, 0, 0, 1, 0

Figure 122

8.19.4.3 Limitations

None.

8.19.4.4 Syntax

< matrix parameter list > : : = [XYROT | YZROT | ZXROT], < scalar >

8.19.5 Definition of a matrix by defining the scaling of the old coordinate system into the new system

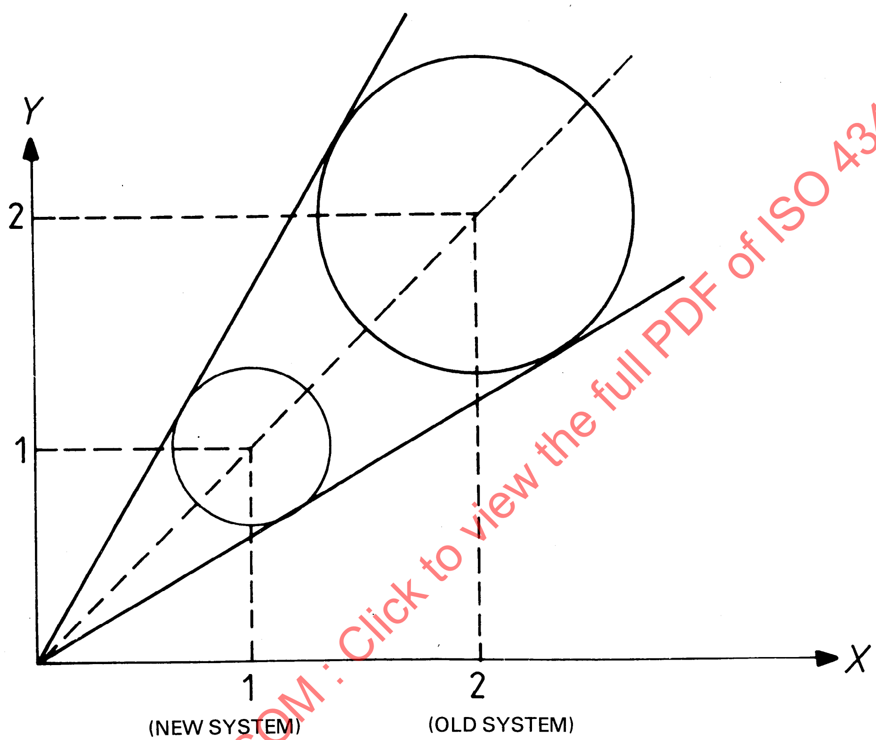
MATRIX/SCALE, scaling factor

8.19.5.1 Semantics

SCALE specifies that the new coordinate system is created by scaling the old system.

Factor represents the ratio between a unit vector in the old system and the corresponding unit vector in the new system.

8.19.5.2 Example



M1 = MATRIX/SCALE, 2

M1 = MATRIX/2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0

Figure 123

8.19.5.3 Limitations

None.

8.19.5.4 Syntax

< matrix parameter list > : : = SCALE, < scalar >

8.19.6 Definition of a matrix by defining a symmetry with respect to a line or a plane

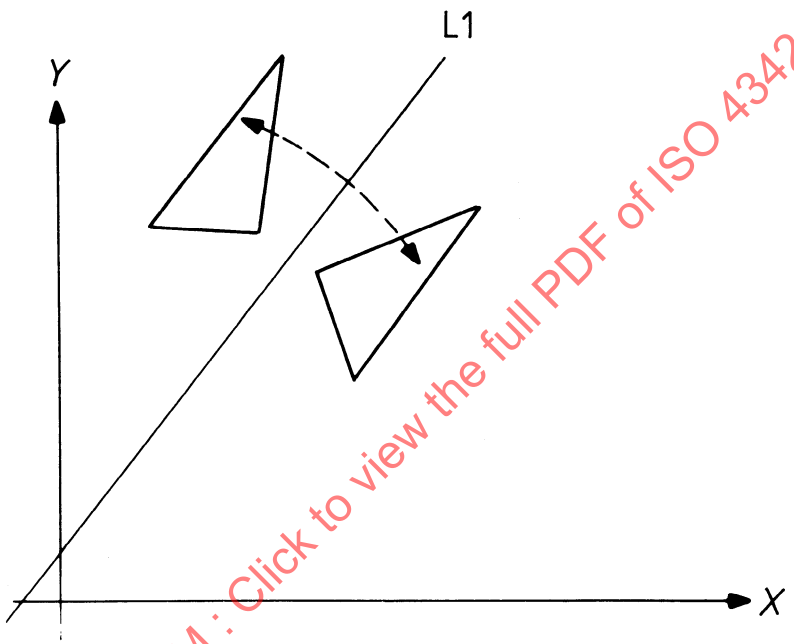
MATRIX/MIRROR, line
plane

8.19.6.1 Semantics

MIRROR specifies that the new coordinate system is created by mirroring the old system about the specified line or plane.

Line or plane is the symbol for a line or a plane.

8.19.6.2 Example



M2 = MATRIX/MIRROR, L1

Figure 124

8.19.6.3 Limitations

None.

8.19.6.4 Syntax

< matrix parameter list > : : = MIRROR, [< line spec > | < plane spec >]

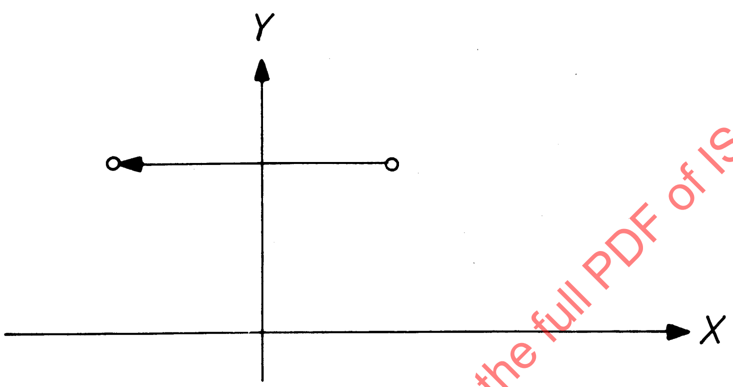
8.19.7 Definition of a matrix by defining a symmetry with respect to the referenced plane or the intersection of the referenced planes

```
                   XYPLAN   XYPLAN   XYPLAN
MATRIX/MIRROR,     YZPLAN , YZPLAN , YZPLAN
                   ZXPLAN   ZXPLAN   ZXPLAN
```

8.19.7.1 Semantics

MIRROR specifies that the new coordinate system is created by mirroring the old system about the specified plane if only one plane is given or the intersection of the planes if two or three are given.

8.19.7.2 Example



M1 = MATRIX/MIRROR, YZPLAN

M1 = MATRIX/ – 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, – 1, 0

Figure 125

8.19.7.3 Limitations

If two or three planes are specified they shall intersect.

8.19.7.4 Syntax

< matrix parameter list > : : = MIRROR $^3_1$ [,XYPLAN | ,YZPLAN | ,ZXPLAN]

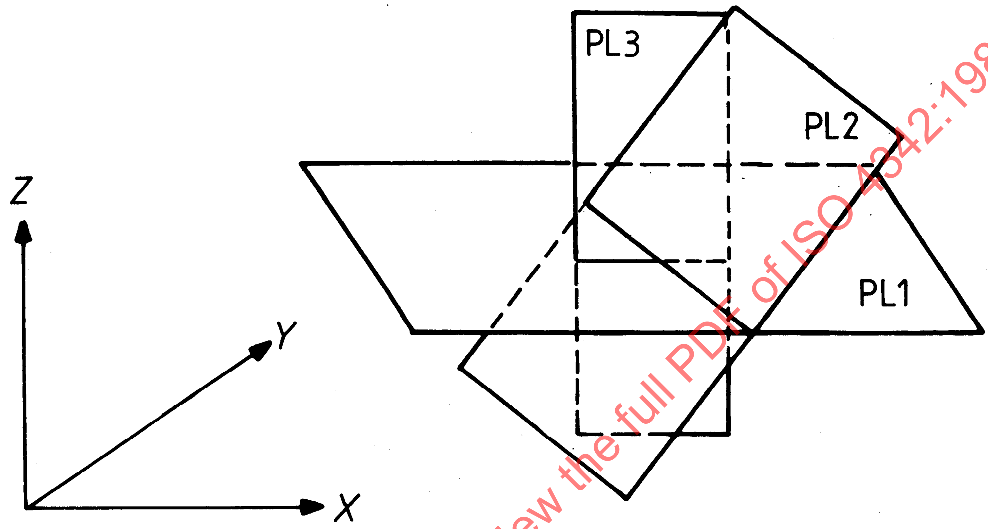
8.19.8 Definition of a matrix by three planes

MATRIX/plane1, plane2, plane3

8.19.8.1 Semantics

Plane1, plane2, plane3 are the symbols for three planes; they are the *XY*-, *YZ*-, *ZX*-planes of the new system of coordinates.

8.19.8.2 Example



M1 = MATRIX/PL1, PL2, PL3

Figure 126

8.19.8.3 Limitations

None.

8.19.8.4 Syntax

< matrix parameter list > : : = < plane spec > , < plane spec > , < plane spec >

8.19.9 Definition of a matrix by a point and two vectors

MATRIX/point, vector1, vector2

8.19.9.1 Semantics

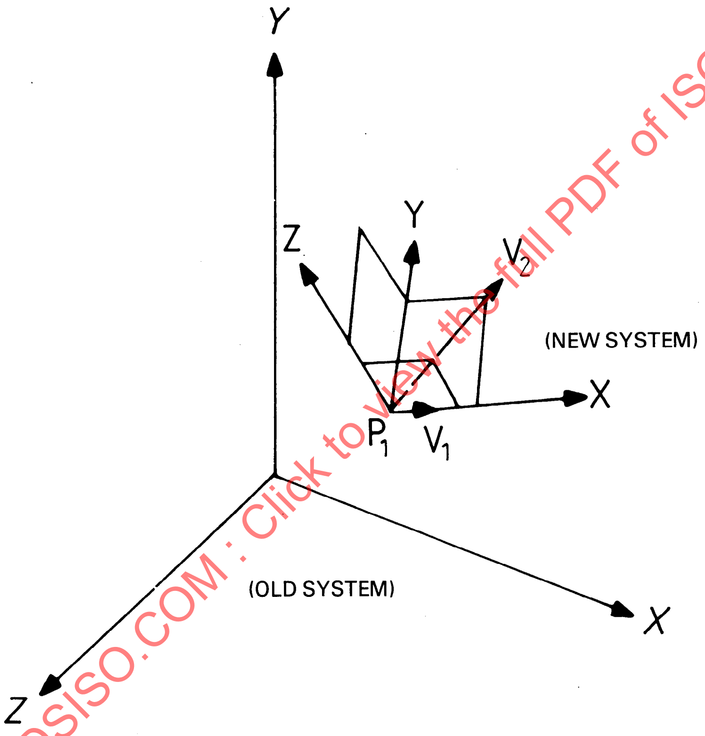
Point is the symbol for a point of origin for the new coordinate system.

Vector1 is the symbol for a vector of the X-axis of the new coordinate system.

Vector2 is the symbol for the vector of the XY-plane of the new coordinate system.

The Z-axis is determined by the cross product of vector1 and vector2.

8.19.9.2 Example



M1 = MATRIX/P1, V1, V2

Figure 127

8.19.9.3 Limitations

The two vectors shall not be parallel.

8.19.9.4 Syntax

< matrix parameter list > : : = < point spec > , < vector spec > , < vector spec >

8.19.10 Definition of a matrix as the product of two matrices

MATRIX/	Matrix1 XYROT, angle1 YZROT, angle1 ZXROT, angle1 TRANSL, d_1, d_2 TRANSL, d_1, d_2, d_3 SCALE, scaling factor1	,	Matrix2 XYROT, angle2 YZROT, angle2 ZXROT, angle2 TRANSL, d_4, d_5 TRANSL, d_4, d_5, d_6 SCALE, scaling factor2

8.19.10.1 Semantics

Matrix1 and matrix2 are the symbols for two matrices.

XYROT specifies a rotation of the specified angle in the XY-plane, similarly YZROT in the YZ-plane and ZXROT in the ZX-plane.

Angle1 and angle2 represents the algebraic value of rotation defined in the given plane.

TRANSL specifies a translation by the specified parameters.

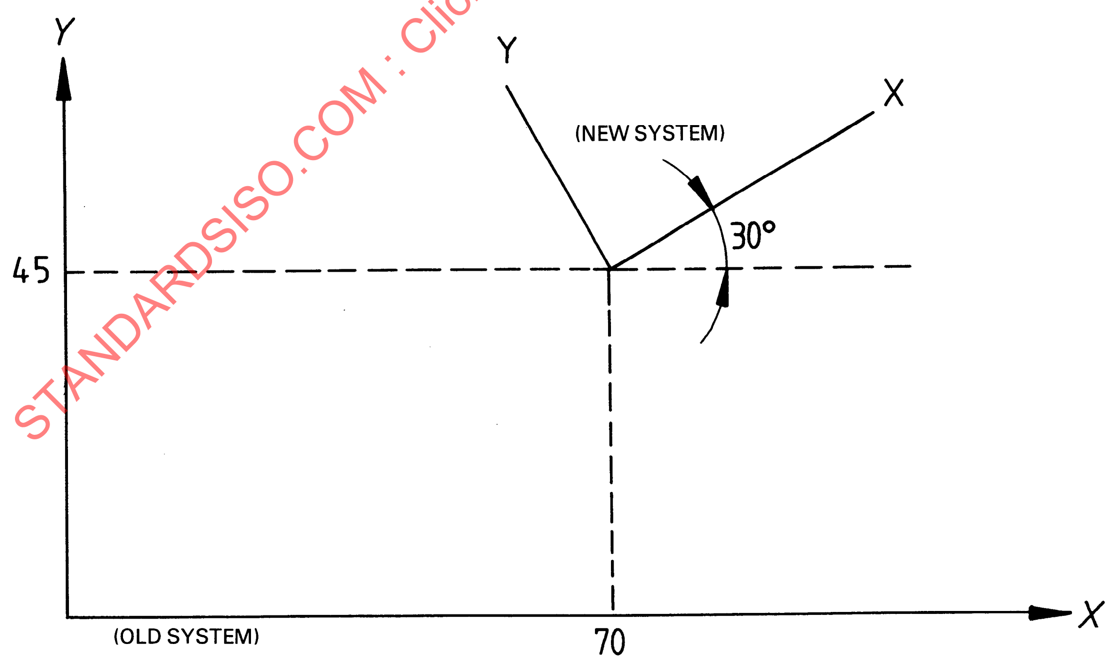
The parameters d_1 to d_6 are the coordinates of the origin of the new coordinate system in the old system, in X, Y, Z respectively. If only two parameters are given then these are assumed to be in X and Y respectively.

SCALE specifies a scaling according to the specified scaling factors.

Scaling factor1 and scaling factor2 represent the ratio between a unit vector in the old system and its corresponding unit vector in the new system.

The order in which the product is computed depends on the implementation, but it is recommended that the second matrix (which appears second in the definition) be applied first.

8.19.10.2 Example



M1 = MATRIX/XYROT, 30, TRANSL, 70, 45

Figure 128

8.19.10.3 Limitations

None.

8.19.10.4 Syntax

< matrix parameter list subset > : : = TRANSL $^{3}_{2}$ [, < scalar >] | [XYROT | YZROT | ZXROT],
< scalar > | SCALE, < scalar >

< matrix parameter list > : : = [< matrix spec > | < matrix parameter list subset >],
[< matrix spec > | < matrix parameter list subset >]

8.19.11 Definition of a matrix as the inverse of a given matrix

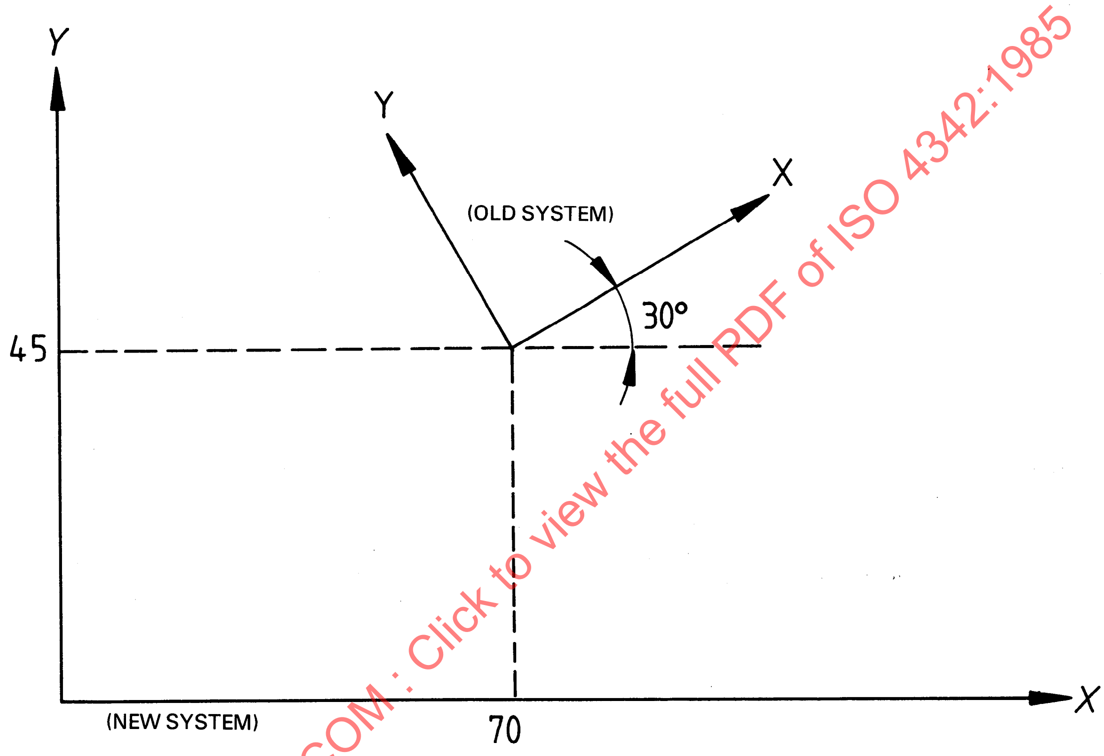
MATRIX/INVERS, matrix

8.19.11.1 Semantics

INVERS specifies that the new coordinate system is the inverse of the specified system.

Matrix is the symbol for a matrix.

8.19.11.2 Example



M1 = MATRIX/XYROT, 30, TRANSL, 70, 45
M2 = MATRIX/INVERS, M1

Figure 129

8.19.11.3 Limitations

The determinant associated with the given matrix shall not be equal to zero.

8.19.11.4 Syntax

< matrix parameter list > : : = INVERS, < matrix spec >

8.20 Definitions of a ruled surface

8.20.1 Definition of a ruled surface

RLDSRF/ruled surface parameter list

9.20.1.1 Semantics

A ruled surface is a surface generated by using straight lines to connect two plane curves.

8.20.1.2 Limitations

None.

8.20.1.3 Sub-contents

For the definition of a ruled surface

a) by a surface and an intersecting plane given by either three points or two points and a vector, and a second surface and an intersecting plane given by either three points or two points and a vector, see 8.20.2;

b) by a surface and an intersecting plane given by either three points or two points and a vector, and a second surface (degenerate) given by a point, see 8.20.3.

8.20.1.4 Syntax

< ruled surface definition statement > : : = < identifier > = RLDSRF/ < ruled surface parameter list >

NOTE — These definitions are included for completeness of input language, and are recognized as not currently being complete or rigorous. Such definitions will be included in future work.

8.20.2 Definition of a ruled surface by a surface and an intersecting plane given by either three points or two points and a vector, and a second surface and intersecting plane given by either three points or two points and a vector

RLDSRF/surface1, point1, point2, $\begin{matrix}\text{point3}\\\text{vector1}\end{matrix}$, surface2, point4, point5, $\begin{matrix}\text{point6}\\\text{vector2}\end{matrix}$

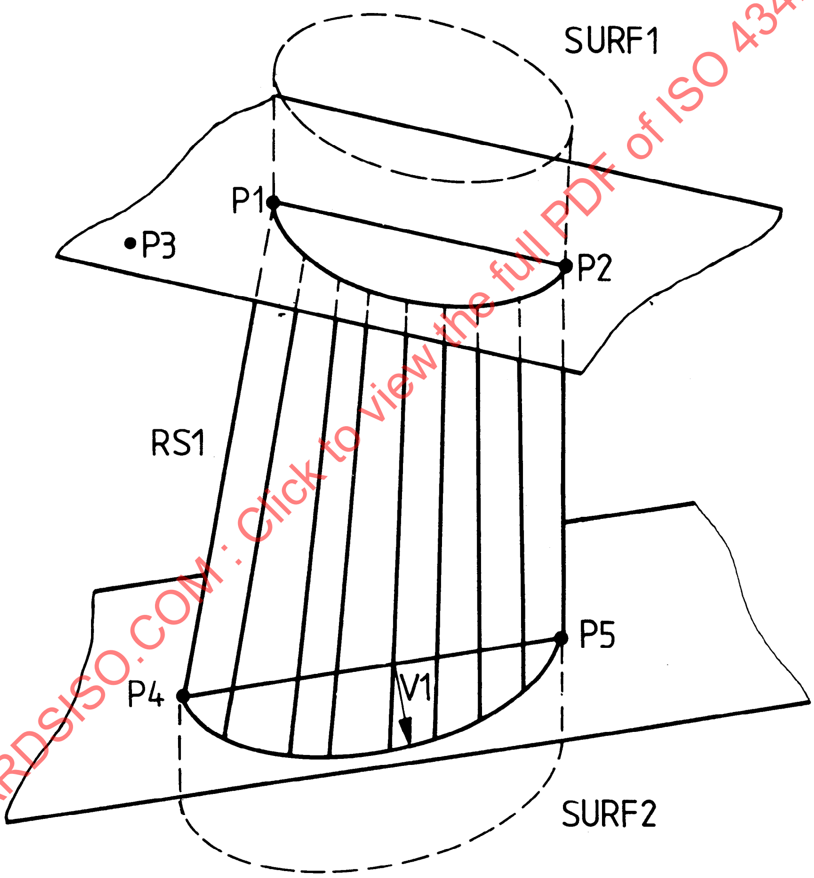
8.20.2.1 Semantics

Surface1 and surface2 are the symbols for two surfaces.

Point1 to point6 are the symbols for points.

Vector1 and vector2 are the symbols for two vectors.

8.20.2.2 Example



RS1 = RLDSRF/SURF1, P1, P2, P3, SURF2, P4, P5, V1

Figure 130

8.20.2.3 Limitations

None.

8.20.2.4 Syntax

< ruled surface parameter list > : : = < surface spec > $_{2}^{2}$ [, < point spec >],[< point spec > | < vector spec >],
< surface spec > $_{2}^{2}$ [, < point spec >],[< point spec > | < vector spec >]

8.20.3 Definition of a ruled surface by a surface and an intersecting plane given by either three points or two points and a vector, and a second surface (degenerate) given by a point

RLDSRF/surface, point1, point2, point3 / vector , point4

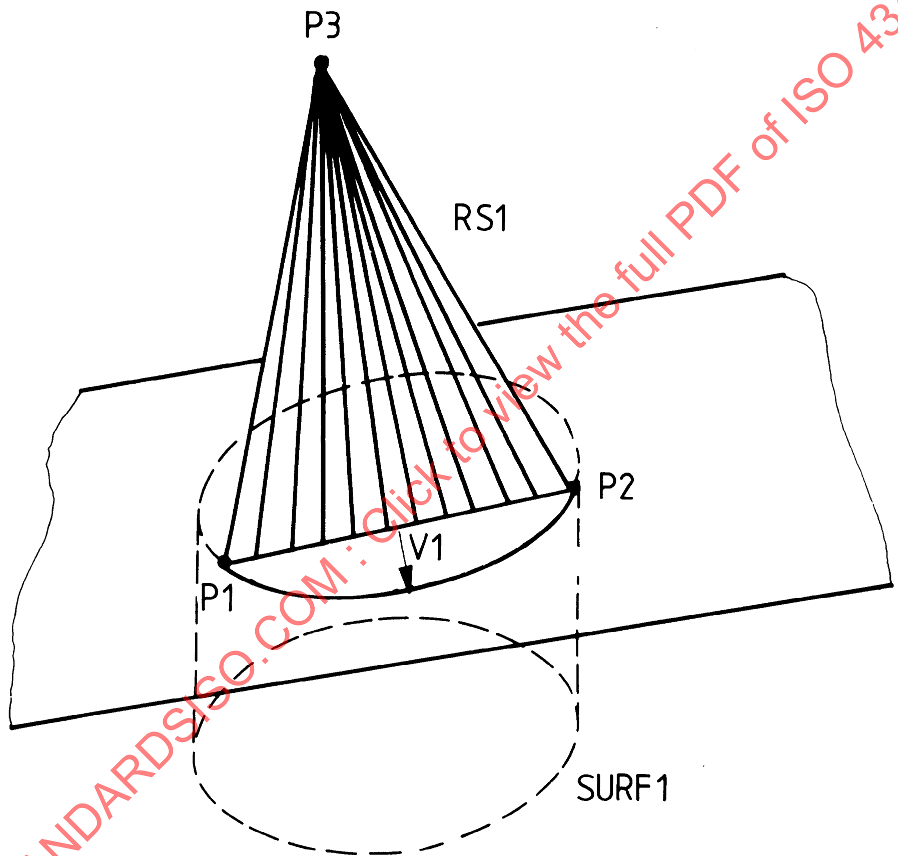
8.20.3.1 Semantics

Surface is the symbol for a surface.

Point1 to point4 are the symbols for points.

Vector is the symbol for a vector.

8.20.3.2 Example



RS1 = RLDSRF/SURF1, P1, P2, V1, P3

Figure 131

8.20.3.3 Limitations

None.

8.20.3.4 Syntax

< ruled surface parameter list > : : = < surface spec > $^2_2$ [, < point spec >],[< point spec > | < vector spec >], < point spec >

9 Geometric execution statements

9.1 General comments

9.1.1 Abbreviations

ds : drive surface

ps : part surface

cs : check surface

GO*** should be taken to mean any member of the set of continuous motion statements GOFWD, GOBACK, GOLFT, GORGT, GOUP or GODOWN.

9.1.2 Definition

modal : A modal word is one whose effect continues through the remainder of the program or until countermanded by some new entry recognized by the system as a replacement for the prior condition.

9.1.3 Limitations

None.

9.1.4 Syntax

< geometrical execution statement > : : = < motion statement > | < tool path control statement > |
< startup direction control statement > |
< transform cutter location statement > |
< processor output file control statement >

9.2 General semantics

This sub-clause covers the description of those words which control the cutter movements required to produce the workpiece, the geometry of which has already been described.

Among these words, the most important are the motion statements which classify the principal types of motion as given below.

9.2.1 Types of motion statements

Simple motion statements	point-to-point statements	relative motion — GODLTA absolute motion — GOTO
	continuous motion statements	initial — GO continuous cutting GOLFT, GORGT, GOUP, GODOWN sequence statement GOFWD, GOBACK
multiple point-to-point statement		GOTO/pattern

The semantics of continuous motion statements and their associated statements are context-dependent. Because of this and before the individual statements are described together with their detailed syntax, an overall semantic description of their uses and mutual relationships is given.

9.2.2 Surface modifiers

In continuous motion statements, symbolically named surfaces are prefixed by one of the modifiers TO, ON, PAST or TANTO.

TO or PAST indicate :

a) the side of a surface on which a cutter should be positioned in a GO statement;

or

b) the side of a check surface on which a cutter should terminate its current motion with respect to its original position and its direction of motion in a GO*** statement.

ON indicates that motion should be terminated on the ds, cs or ps in a GO statement or on the cs in a GO*** statement.

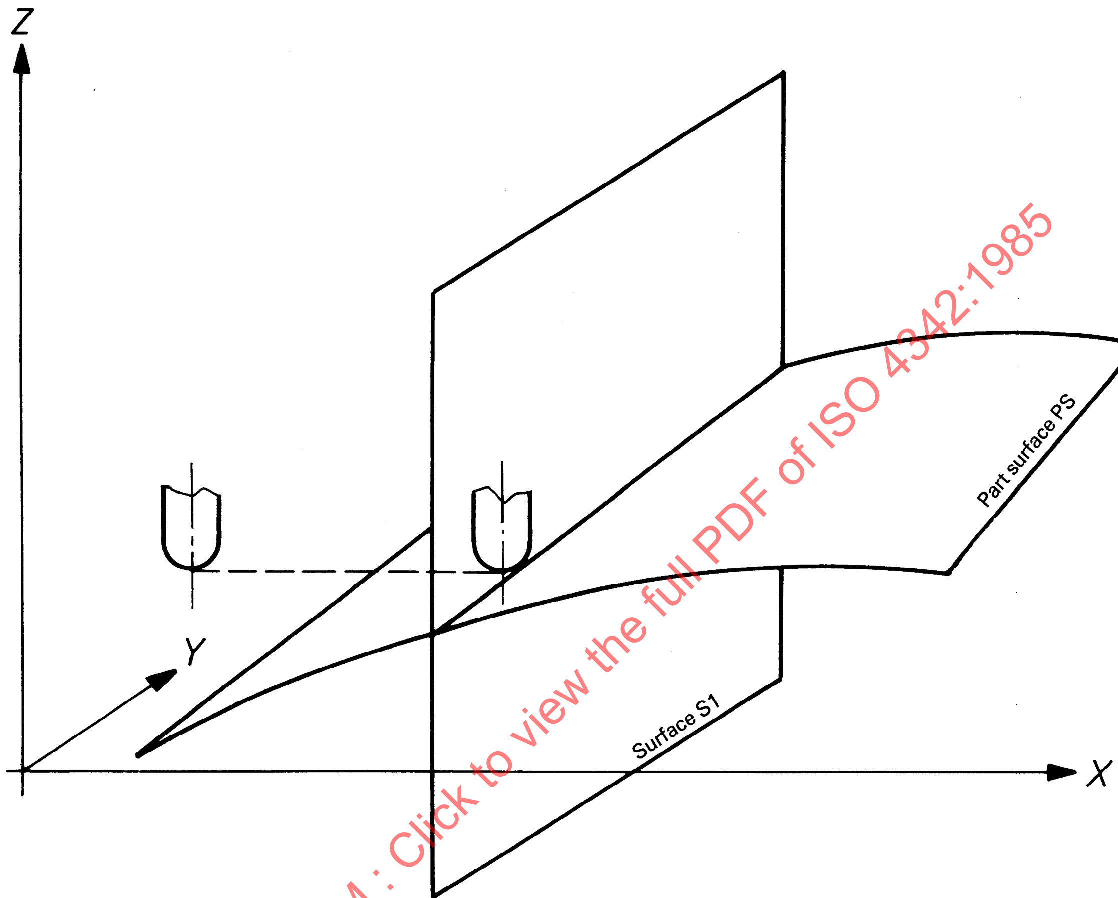
TANTO indicates that motion should be terminated when the cutter reaches that point where the ds and cs are mutually tangent.

The absence of such a modifier before the symbolic name of a ds, ps or cs in a GO statement or of a cs in a GO*** statement indicates a TO modifier by default.

Startup-GO- : (initial continuous motion statement). This motion statement positions the tool within tolerance of one, two or three controlling surfaces taking account of their modifiers. The order of occurrence of the three possible surfaces in a "startup" statement is ds, ps, cs.

9.2.3 One surface startup, non-directed, with effective part surface

The previously defined part surface is effective in the startup motion. The cutter is moved the shortest possible distance to the required position relative to the specified surfaces.

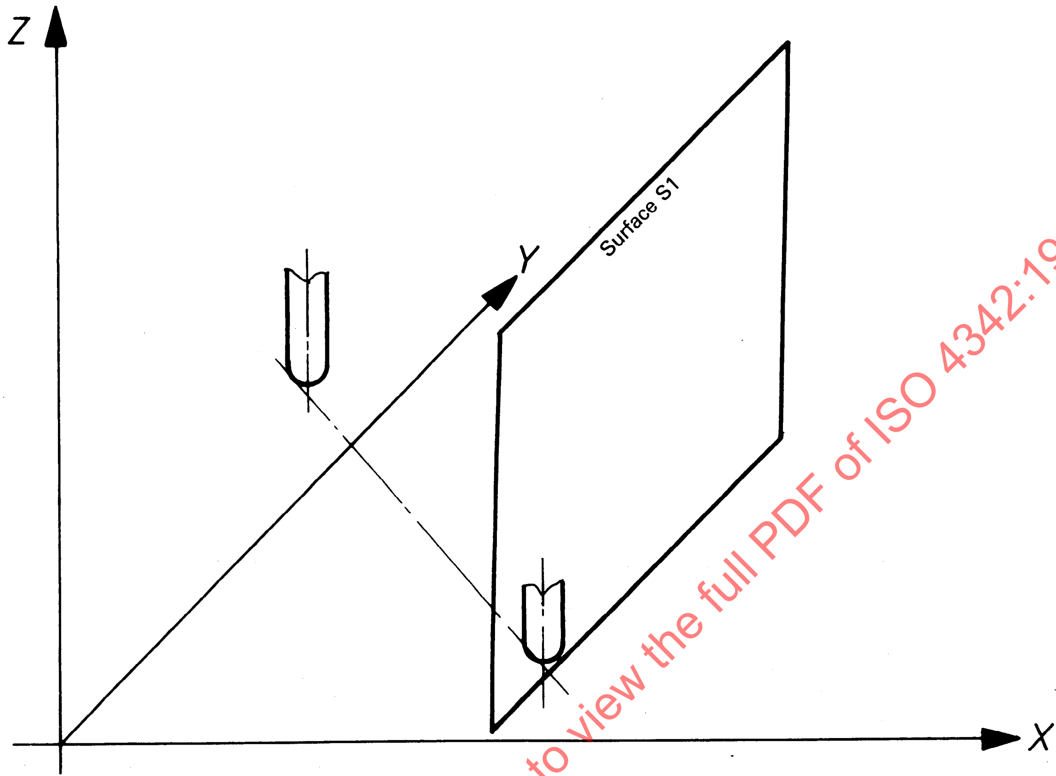


PSIS/PS
GO/TO, S1

Figure 132

9.2.4 One surface startup, non-directed, without effective part surface

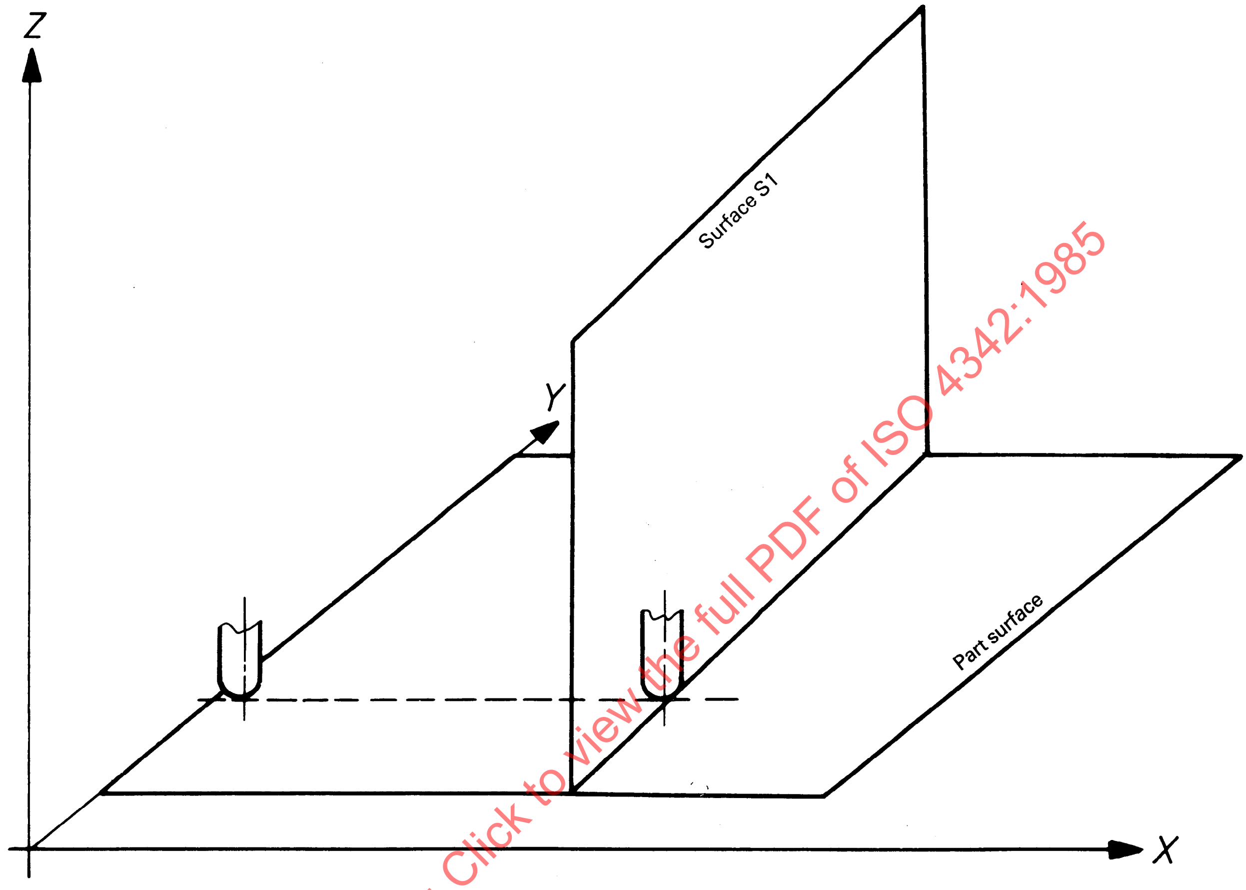
If no part surface has been previously established, the part surface in the startup motion is assumed to be the *XY*-plane and the resulting motion is shown in figure 133a) :



GO/TO, S1

Figure 133 a)

If the tool lies in the XY-plane, the motion takes place along the minimum distance to S1. If AUTOPS has been previously defined, a plane parallel to the XY-plane at the cutter's current z height is assumed as part surface and the resulting motion is shown in figure 133b) :

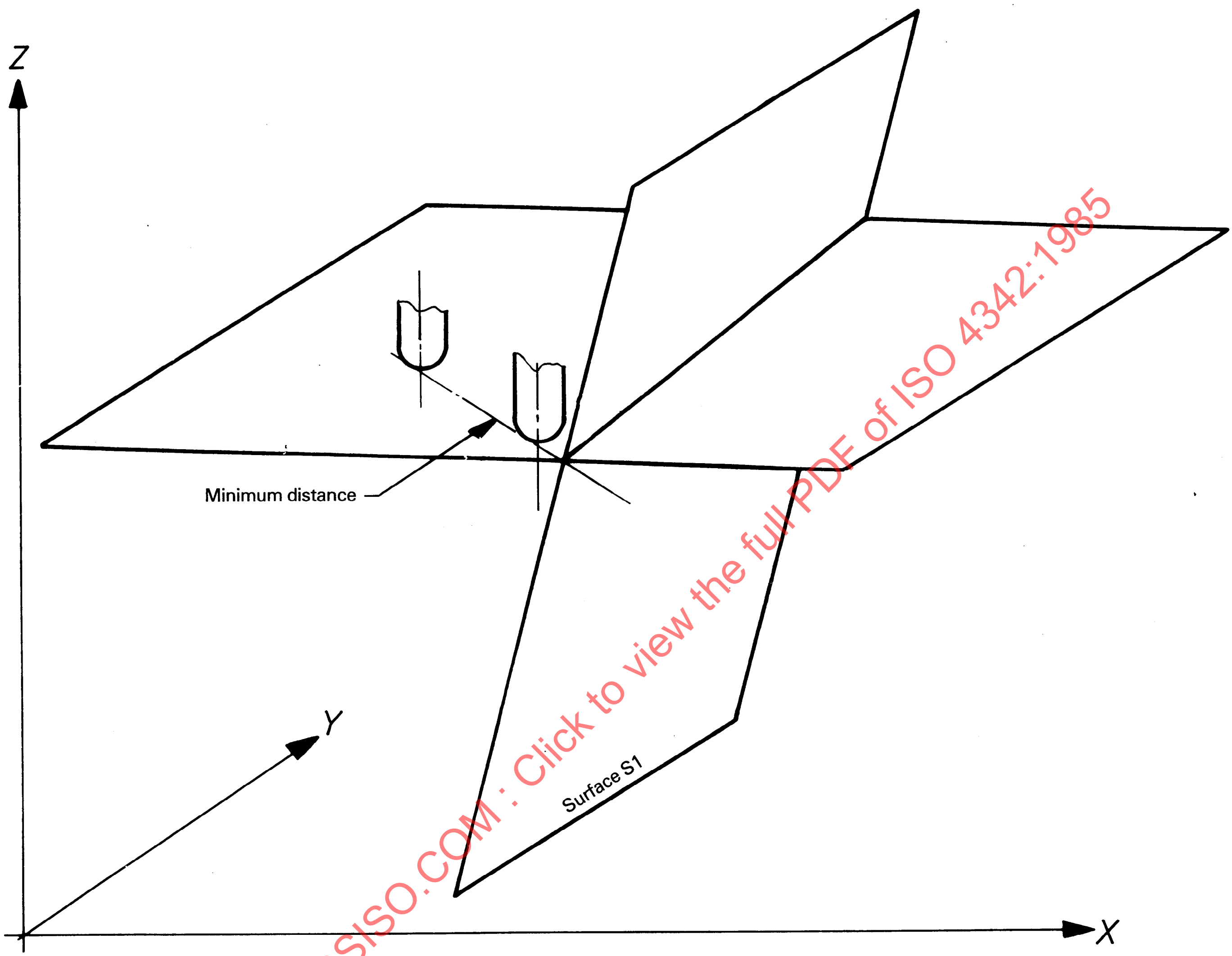


AUTOPS
GO/TO, S1

Figure 133b)

9.2.5 One surface startup, non-directed, independent of part surface

The NOPS statement used prior to a single surface startup indicates that, for this startup only, the cutter is to be moved from its present position to the surface referenced in the startup by the minimum distance without regard to part surface.

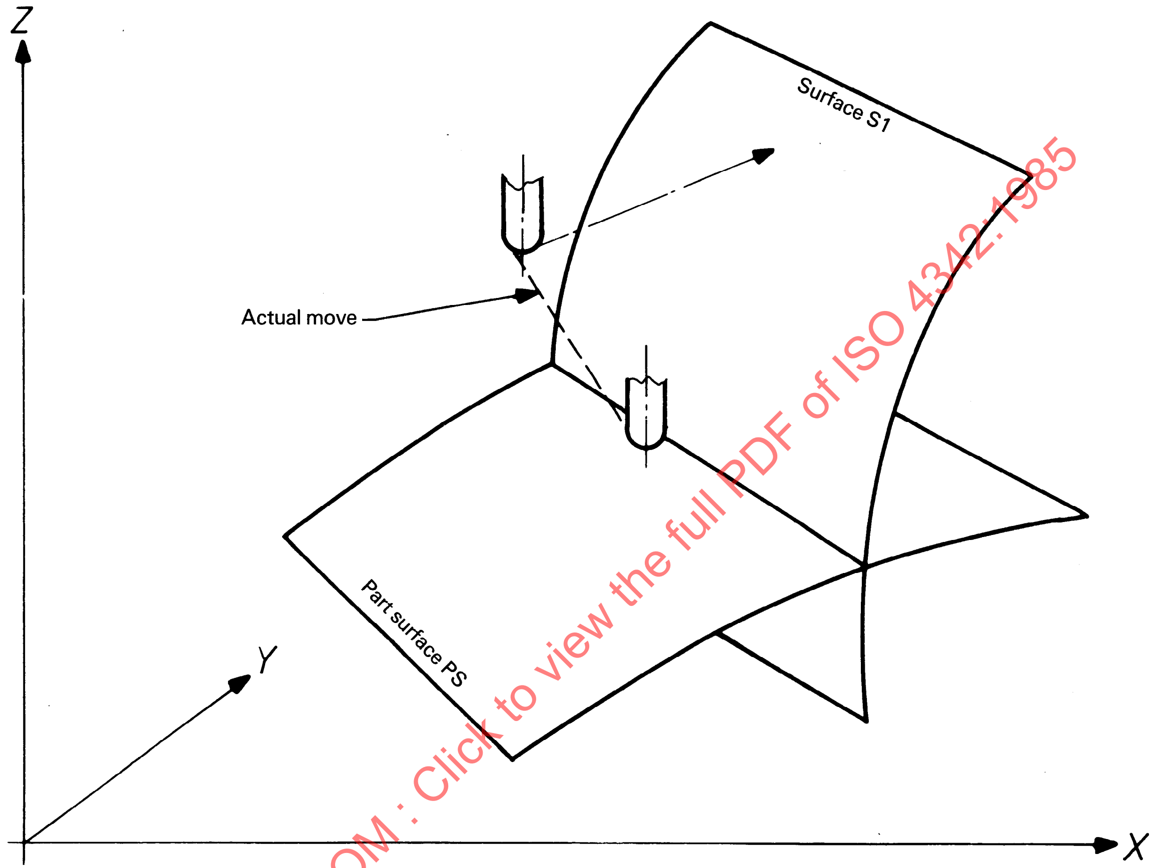


NOPS
GO/TO, S1

Figure 134

9.2.6 One surface startup, directed, with effective part surface

The statement INDIRV/ or INDIRP/ is used as an aid in determining the move in the direction of the referenced surfaces but the tool does not necessarily move exactly in the vector direction because of the added constraint of having to move within tolerance of the part surface. If the part surface is not specified, the previously established part surface (default XY-plane, plane established by AUTOPS, surface specified by PSIS, or surface specified by a two or three surface startup) is used.



PSIS/PS

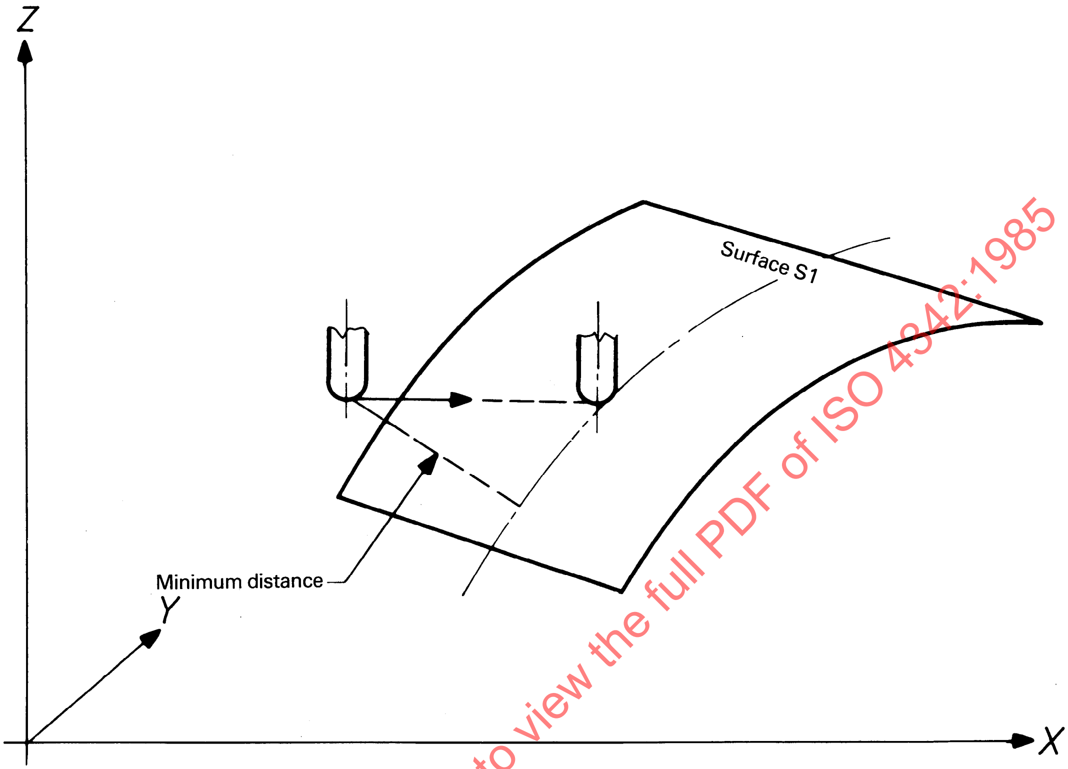
INDIRV/V

GO/TO, S1

Figure 135

9.2.7 One surface startup, directed, independent of part surface

In this case, the cutter moves exactly in the direction of the vector to the referenced surface without regard to the minimum distance to the surface.



INDIRV/V

NOPS

GO/TO, S1

Figure 136

9.2.8 Two surface startup, non-directed

The first surface is intended to be the drive surface, the second becomes the part surface for later motion statements.

The cutter is moved the shortest possible distance to the required position relative to the specified surfaces.

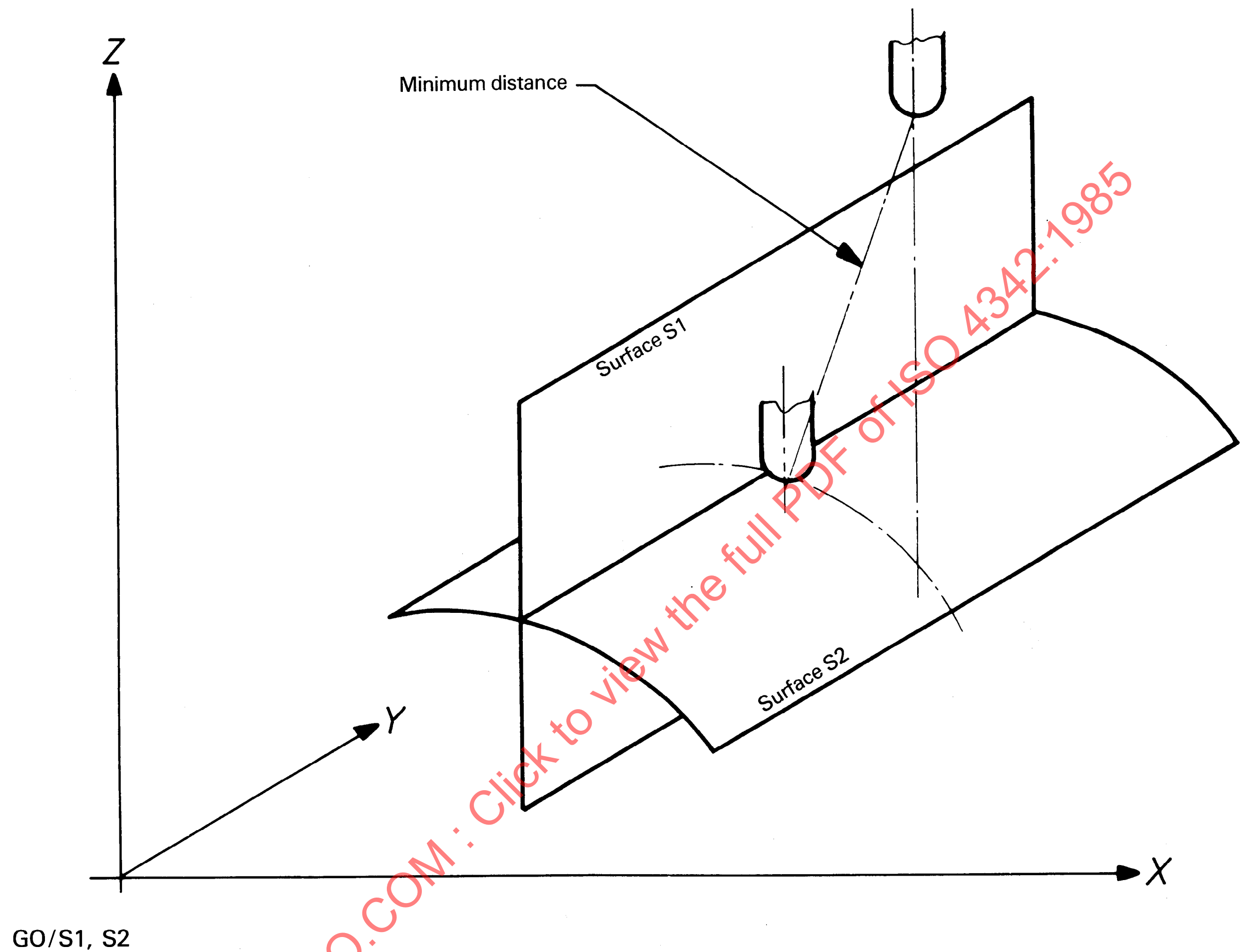
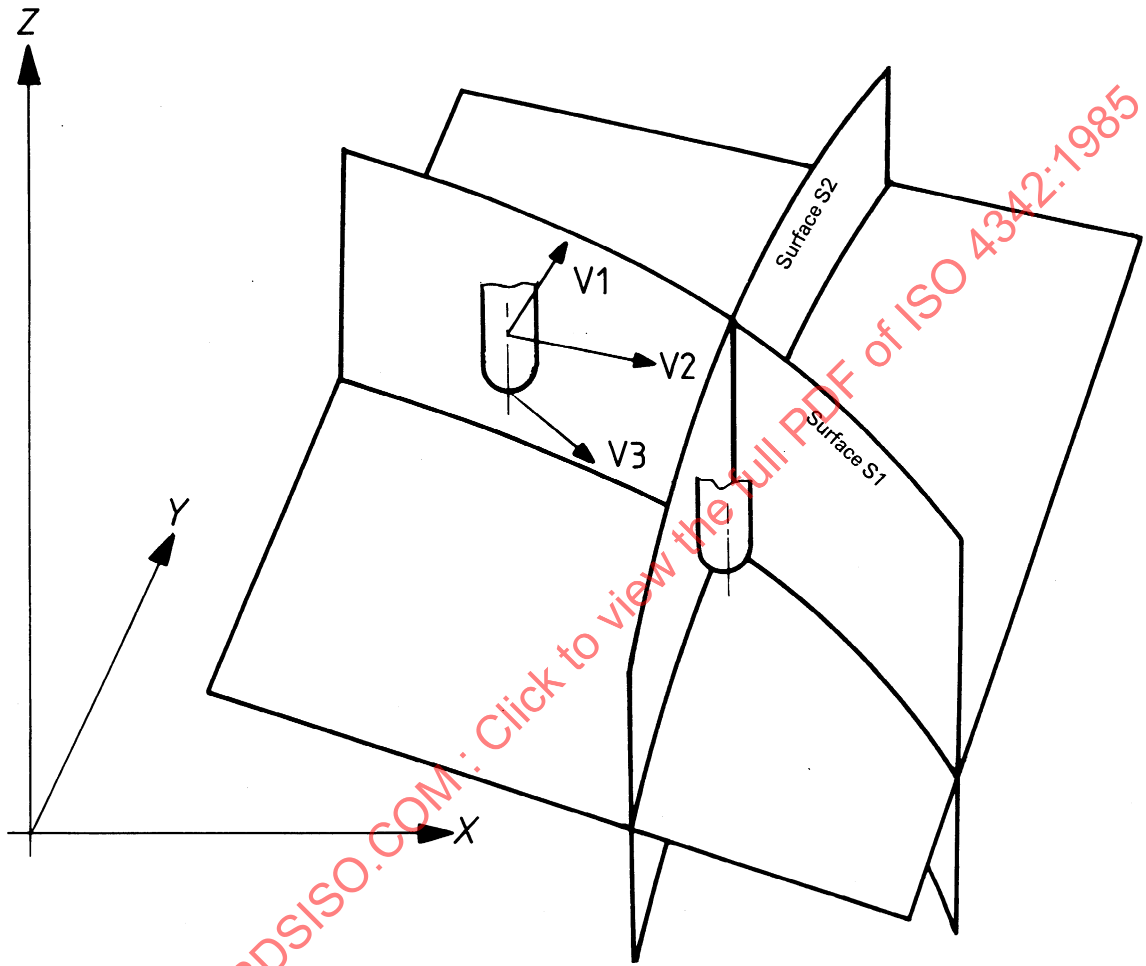


Figure 137

9.2.9 Two surface startup, directed

The startup statement can be preceded by an INDIRV/ or INDIRP/ statement. One or two vectors may be referenced to the right of the slant in the INDIRV/ statement. They refer to the drive surface and part surface respectively.

This approach is used mainly to place the cutter at the desired stopping position when an ambiguity exists. The tool does not necessarily move in the vector direction because of the added constraint of having to move to a position within tolerance of both surfaces.



SRFVCT/V1, V2

INDIRV/V3

GO/S1, PAST, S2

Figure 138

9.2.10 Three surface startup, non-directed

The first surface is intended to be a drive surface, the third a check surface; the second becomes the part surface for later motion statements.

The tool is moved to the closest position at which it can satisfy the given relationships. The TANTO modifier can be used only with the third surface and indicates that the first and third surfaces are tangential at the desired tool location.

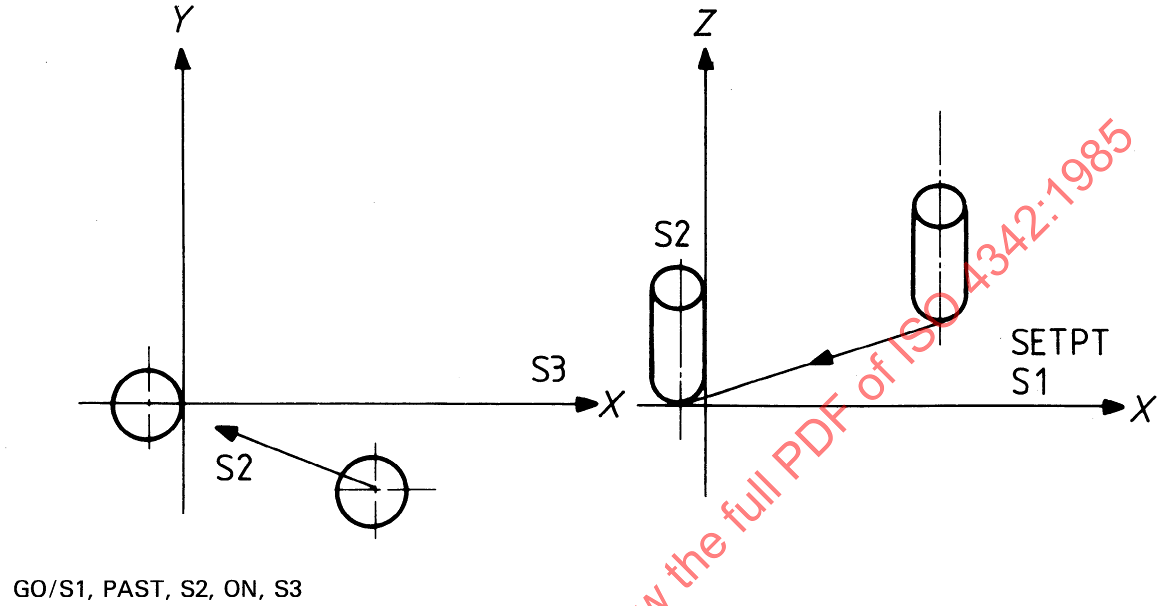


Figure 139

9.2.11 Three surface startup, directed

As mentioned in the case of two surface startup, directed, the statements INDIRV/ or INDIRP/ and SRFVCT/ are used only to establish a general search direction and the TO or PAST side of the surfaces referenced; so they do not necessarily define the actual tool movement. The tool is moved to the closest position at which it can satisfy the given relationships.

9.3 Continuous motion statement

A continuous motion statement (GOLFT, GORGT, GOFWD, GOBACK, GOUP, GODOWN) executes tool movement along two surfaces, a part surface and a drive surface until a third surface — a check surface — is reached. Motion of the tool with respect to the surfaces is achieved by displacement of the control point of the tool and by adjustment of the inclination of the tool axis.

Normally in milling and drilling machines, the tool control point is the point of intersection of the tool axis and the tool tip. In turning, the control point is normally the centre of the tool tip radius.

The tool usually remains in contact with one particular surface during several successive continuous motion statements. This surface is usually designated the part surface. It is set up in a preceding GO or PSIS statement or indicated by an AUTOPS statement.

The drive surface for a continuous motion statement is the first surface to appear in the statement. Whether the tool control point moves along the part and drive surfaces in an ON or offset condition and whether the motion terminates ON or offset with respect to a check surface, is indicated by the appropriate surface modifiers.

The part programmer may assign a tolerance to either or both sides of each surface using an OUTTOL, INTOL or TOLER statement (see 9.4.5). In the absence of a part programmed tolerance, default value is assumed.*

The tolerance can be regarded as a shell of the specified thickness wrapped uniformly round the appropriate surfaces.

Using this tolerance concept, the part programmer limits the discrepancy permitted to exist between the calculated but approximate tool path and the geometrically defined surface being cut.

During each cut for a surface whose modifier is ON, the control point of the tool shall never move outside the specified tolerance band. For a surface with the modifiers TO and PAST, at any instant during a cut sequence, at least one point on the tool's envelope shall lie within the tolerance band of that surface.

* The preferred value is 0,012 7 mm (0.000 5 in).

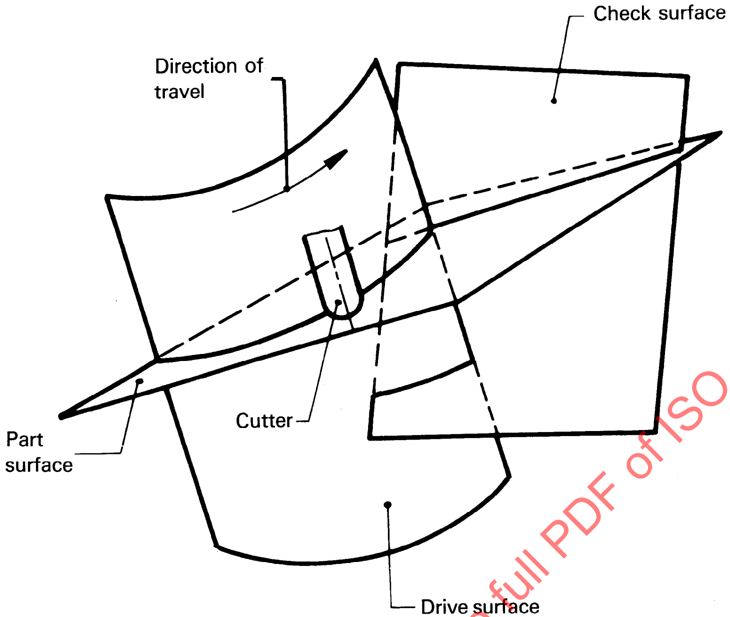


Figure 140 a)

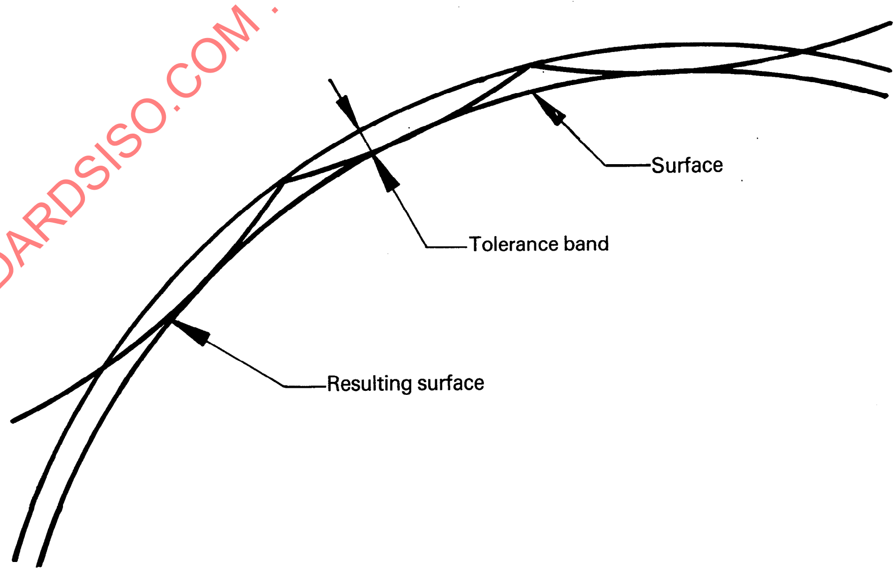


Figure 140 b)

Before the processing of a continuous motion statement is begun, the current forward direction of motion FWD is regarded as the tangential direction of motion at the end point of the last cut sequence or the direction indicated by an INDIRV or INDIRP statement if such a statement is part programmed between the last motion statement and the one about to be processed.

At the beginning of a continuous motion sequence, the tool will have been placed within tolerance of a part surface and a drive surface. There are only two possible and mutually opposite directions in which motion can take place — namely along the line of the mutual tangent to the ps and the ds. Which of these two possible directions is desired is indicated by the major word. Direction is always given with respect to the last known direction of motion FWD, GOFWD, GOBACK, GOLFT and GORGT may be chosen to indicate direction as indicated in figure 140c).

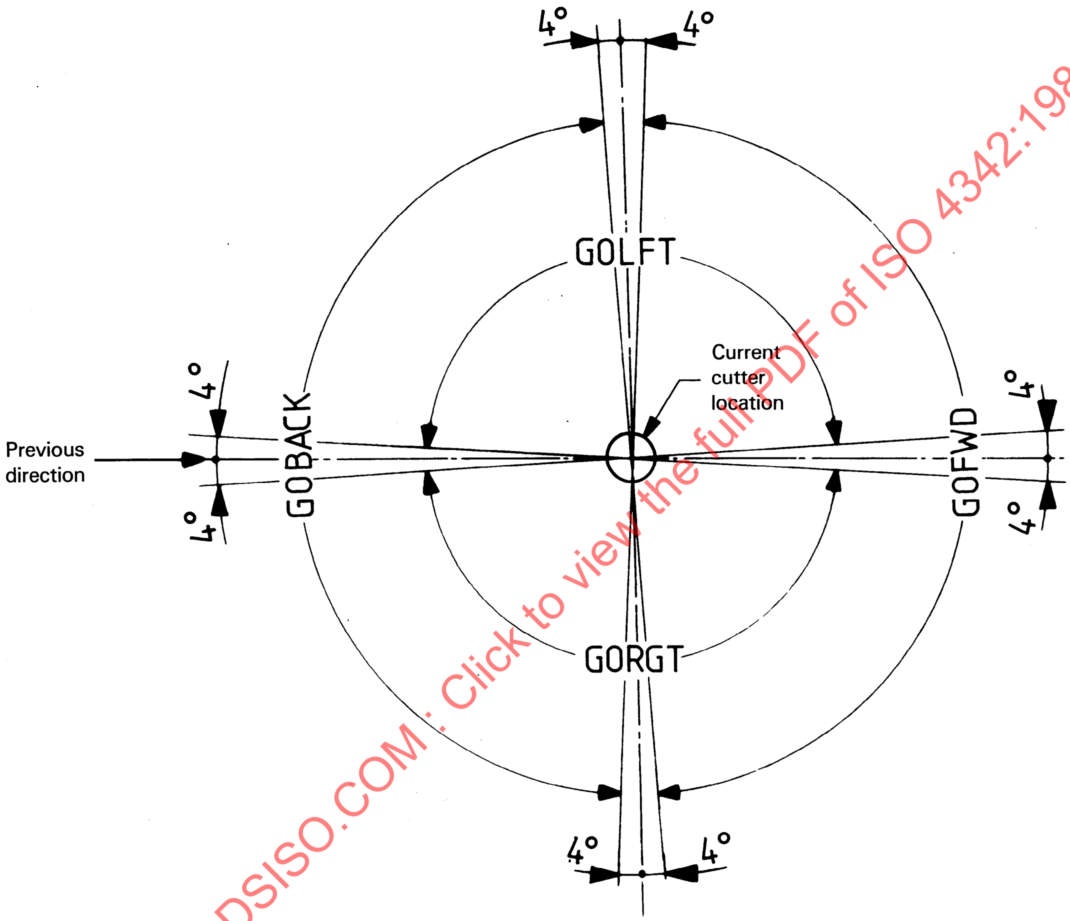


Figure 140c)

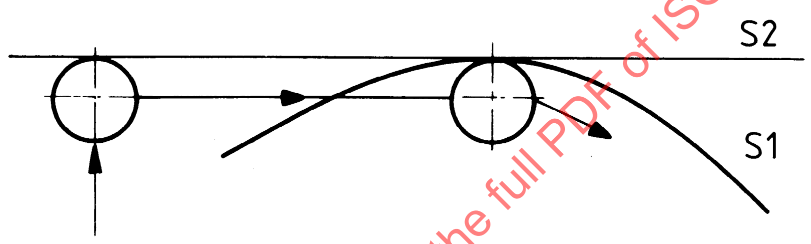
GOUP and GODOWN are used when some component of the motion is to be along the tool axis; GOUP when motion tends to be back into the tool from the tool end point and GODOWN when motion is to be in the line of the tool axis from the tool end point out of the tool. When a previous motion statement has been a GOUP or a GODOWN and movement has been exactly along the tool axis, the FWD direction is taken as the last non-vertical move executed.

Continuous path motion statements shall first reference a drive surface (the first mentioned surface in the statement) and there may follow either no check surface, one check surface or multiple check surfaces.

The normal statement contains a drive surface and one check surface. However, in many cases a series of motion statements may be programmed in which the check surface of one statement becomes the drive surface of the next. In such a series, the check surface may be omitted from the statements in all statements except the last one of the series. In such cases the processor will assume that the absent check surface is the drive surface of the next statement. However the modifier of the implied check surface shall also be deduced and such series of statements should be preceded by a tool-to-surface orientation command — namely TLLFT, TLRGT or TLON. The table below indicates the modifier which should be assumed by the processor for the various combinations of tool-to-surface orientation or motion statements. It should be noted that a GOBACK and GOFWD statement, when used in an implied check surface series (and only when used in this situation) implies a TANTO modifier for the implied check surface in the previous statement.

Table

Tool-to-surface command (programmed at the start of a series of implied cs statements)	Motion command of statement (n + 1)	Implied cs modifier of statement n
TLLFT	GOLFT	TO
TLRGT	GORGT	TO
TLLFT	GORGT	PAST
TLRGT	GOLFT	PAST
TLON	GOLFT or GORGT	ON
any	GOFWD or GOBACK	TANTO
any	GOUP or GODOWN	error

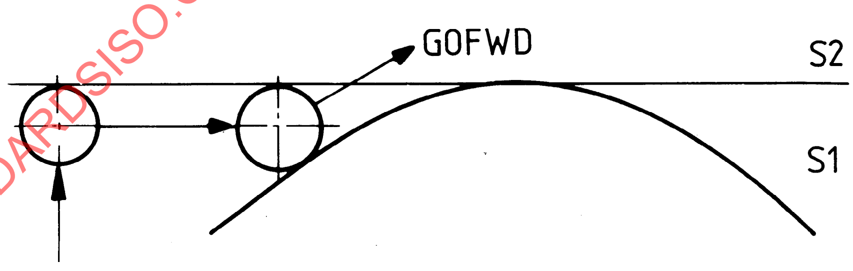


TLRGT

GORGT/S2

GOFWD/S1

Figure 140 d)



TLRGT

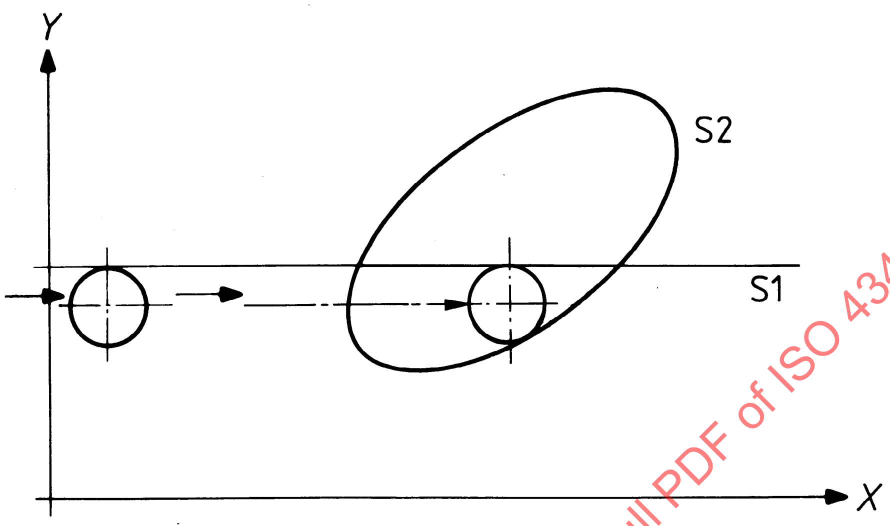
GORGT/S2, TO, S1

GOFWD/S1 . . .

Figure 140 e)

The implied check surface statement format shall not be used in a GOUP or GODOWN statement or in a statement preceding either of these commands.

When there is more than one stopping position relative to the check surface that meets the specified requirements, the couplet "*n*, INTOF" may be specified to indicate that the *n*th position is desired. If the first position is desired, "1, INTOF" may be specified but is not necessary.



GOFWD/S1, TO, 2, INTOF, S2

Figure 140 f)

Difficulties may arise in deciding which of two surfaces the tool will meet while in motion, therefore, a second cs's may be given in the terminal continuous motion command. The resulting cutter path depends on which of the two cs's is reached first.

The format for multiple check surfaces is

$$\text{GO***/ds,} \left\{ \begin{array}{l} \text{TO} \\ \text{ON} \\ \text{PAST} \\ \text{TANTO} \end{array} \right\} \text{, cs1, id1,} \left\{ \begin{array}{l} \text{TO} \\ \text{ON} \\ \text{PAST} \\ \text{TANTO} \end{array} \right\} \text{, cs2, id2}$$

where id1 represents the label name of the statement to which the program should transfer if the first cs controls and id2 represents the label name of the statement to which the program should transfer when the second cs controls.

As a possible case of multiple check surface usage, take for example rough and finish cuts with different sized cutters : the large diameter cutter may not reach into a fillet radius.

If the statement containing the multiple check surfaces is part of a MACRO or LOOP body, the transfer shall be only forward; that is the statements labelled with id1 and id2 shall appear later in the program than the statement with multiple cs itself.