INTERNATIONAL STANDARD

ISO 22093

Second edition
2011-05-15

Industrial automation systems and integration — Physical device control — Dimensional Measuring Interface Standard (DMIS)

Systèmes d'automatisation industrielle et intégration — Contrôle du dispositif physique — Norme d'interface de mesurage dimensionnel (DMIS)

Reference number
ISO 22093:2011(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

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. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 22093 was prepared by the American National Standards Institute (ANSI) (as DMIS 5.2) and was adopted, under a special “fast track procedure”, by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 1, *Physical device control*.

This second edition cancels and replaces the first edition (ISO 22093:2003), which has been technically revised.

The significant changes are listed below:

Annex G: New annex to list deleted statements. Because of the confusion and ambiguity created if a new statement is created to replace or enhance an existing statement, this annex was added.

FEAT/LINE: Changed the definitions for the endpoints of the line to explicitly state that the first point is the starting point and the second point is the ending point.

Intrinsic functions: Changed CONCAT(str,str var_2) to CONCAT(str var_2).

Changed MN(x,x var_4) to MN(x var_4)

Changed MX(x,x var_4) (to MX(x var_4)

FEAT/OBJECT: removed ambiguity with input & output values.

DATDEF: the syntax did not allow for a single datum target, though this is explicitly allowed by Note 3 text.

Clause 5.3.1: added normative text for vector operators.

CONST (input format 6): added Ellipse to the var_1 options.

Clause 5.1.6 Removed the 80 character line length limitation and changed it to a max line length of 65,536 characters and the line is terminated with a carriage return(CR) and line feed(LF).

IF: Modified the var_x options.

TOL/ANGLWRT: Modified the var_1 options.

SCNSET: Added the minor word DEFLECTION with a deflection value.

VALUE: Added the minor word DEFLECTION to the var_1 option.

CONST (input format 15): Removed search_radius from a var_3 option.

CALIB: Added the ability to recalibrated sensors.

CRGDEF: Added the ability to define a carriage without volume parameters.

FROM: Added DME, RAM and SCALE options.

ROTAB: Added the ability to position a rotary table relative to a feature direction.

Annex C: The EBNF has been updated to reflect all changes to DMIS. Additionally several typographical errors have been corrected.

GECOMP: The GECOMP statement has been added to Annex G and removed from DMIS.

KEYCHAR: New major word which defines a key characteristic that associates nominal feature(s) and nominal tolerance(s) with an optional key characteristic criticality designation and assigns a unique label to it.

Industrial automation systems and integration — Physical device control — Dimensional Measuring Interface Standard (DMIS)

1 Scope

This International Standard defines a neutral language for communication between information systems and Dimensional Measurement Equipment (DME) called the Dimensional Measuring Interface Standard (DMIS). DMIS is an execution language for measurement part programs and provides an exchange format for metrology data such as features, tolerances, and measurement results.

DMIS conveys the product and equipment definitions along with the process and reporting information necessary to perform dimensional measurements that employ coordinate metrology. DMIS contains product definitions for nominal features, feature constructions, dimensional and geometric tolerances, functional datums, and part coordinate systems. It also communicates equipment definitions for various measurement sensors, measurement resources, and machine parameters. DMIS instructs the DME's motions and measurements for product acceptance or verification and for manufacturing process validation and control. Furthermore, DMIS guides the analysis of coordinate data to report and tag measurement results that ascertain product/process quality.

Finally, to aid in its implementation, application functional subsets of DMIS have been defined that ensure successful interoperability and to validate DMIS conformance. Also, DMIS addresses the associativity of DMIS product definitions with CAD information.

While primarily designed for communication between automated equipment, DMIS is designed to be both human-readable and human-writable, allowing inspection programs to be written and inspection results to be analyzed without the use of computer aids. With the enhancement of the High Level Language extensions, DMIS can function and be implemented as a complete DME language.

DMIS provides the vocabulary to pass inspection programs to dimensional measuring equipment and to pass measurement and process data back to an analysis, collection, and/or archiving system. A piece of equipment which interfaces to others, using the DMIS vocabulary, may do so directly or it may have a pre-processor to convert its own native data formats into the DMIS format and/or a postprocessor to convert the DMIS format into its own data structure.

An environment making use of the DMIS input and output formats as a data exchange standard is depicted in (Figure 1 — DMIS environment). As illustrated, an inspection program can be created by many different approaches. Inspection program creation can be assisted by CAD systems, non-graphical systems, automated systems, or constructed manually. A programming system may require a pre-processor which converts the program into DMIS format. A DMIS inspection program can then be executed on dissimilar dimensional measuring equipments. In (Figure 1 — DMIS environment), DME I has a DMIS pre-processor and post-processor which converts the DMIS data into its own unique data format. DME IV is utilizing DMIS as its native format and therefore no pre-processors or post-processors are required. Also, a host computer is being used to control DME II and DME III. The host has a post-processor which decodes the DMIS program and drives the two DMEs, either through DMIS formats, or through some user-defined data exchange format.

Resultant data may be passed back in DMIS format through various scenarios. For example, this data could be passed directly as DMIS or via a post-processor. Resultant data is typically passed to an analysis system and/or a storage system such as a Quality Information System (QIS).

The manual interface indicates that DMIS programs can be hand written, and results analyzed, without the use of computer aids. In addition, many other uses of the DMIS data exchange format could be applied.

The implementation of DMIS is dependent on individual users. DMIS simply defines a neutral data exchange format that can be transmitted via ASCII or UTF8 files from one DMIS supporting system to another. The method for the transmission, storage, and management of these files is user-dependent.

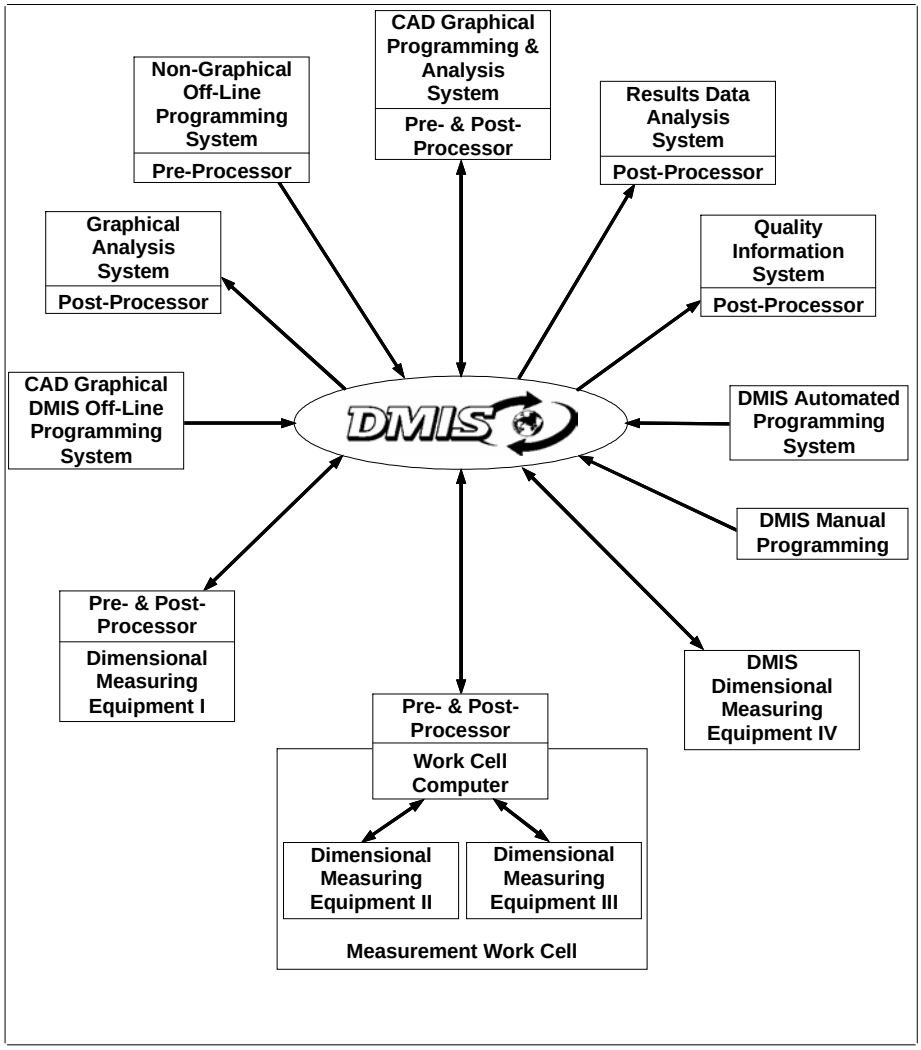


Figure 1 — DMIS environment

2 Conformance

The primary purpose of DMIS is to allow organizations to exchange and store measurement data among different dimensional measurement devices and computer applications both within their organizations, as well as with other organizations. DMIS is widely used and available for a broad range of measurement systems and applications. However, a DMIS file that is created by one DMIS product may not be fully or correctly interpreted by another DMIS product. Successful DMIS interchange can only be achieved if DMIS applications faithfully implement 1) the DMIS specification and 2) the appropriate formally recognized DMIS application profiles coupled with any addenda. DMIS is a large and complex standard. Vendors do not need to implement the entire standard. They implement functional characterized subsets. A characterized subset involves an application profile at a specific conformance level along with zero or many associated addenda at their specified conformance level.

The primary benefit of any DMIS profile is the ability to insure interoperability through the use of validation tools against DMIS instances and certification services for applications. Once an application has been certified through a testing service, behaviour of that application is predictable under the constraints of the profile.

2.1 DMIS conformance testing

DMIS conformance testing is a way of determining if a DMIS compliant product correctly implements the DMIS specification with its associated application profile.

Strictly speaking, this DMIS specification is solely an exchange file format. However, the term "DMIS" is often used to include a generator (a program which produces the DMIS), an interpreter (a program which reads the DMIS), as well as the metafiles (the actual DMIS input and output files). Together, the generator, metafiles, and interpreter form a total DMIS system.

Conformance of DMIS is defined in terms of conformance to a particular application profile of DMIS. Thus, the DMIS specification in conjunction with an application profile is necessary in order to test conformance of a total DMIS system.

Testing DMIS for conformance entails one or many of the following:

a) verifying that the metafiles are syntactically correct,

b) verifying that a generator produces conforming metafiles which accurately and correctly represent the intended results,

c) verifying that an interpreter can correctly and completely read the metafile and produce the intended results, and

d) verifying that the DMIS characterization file is syntactically correct and that it accurately represents the capabilities of the application.

An application profile conforms to DMIS if it adheres to all syntactic requirements defined in this standard.

Application software conforms syntactically to DMIS if it interprets all conforming DMIS application profiles.

Application software conforms semantically to DMIS if it interprets all metafiles that conform to DMIS application profiles according to all required semantics prescribed by this standard.

2.2 Conformance testing services

Conformance testing services will be recognized by the DSC to utilize test suites for validation of DMIS characterization files and to test implementations for conformance to one or many DMIS application profiles.

2.3 CHFile

A vendor's DMIS characterization file represents its compliance to the DMIS specification. Conformance testing will validate that the characterization file is syntactically correct and that it accurately represents the capabilities of the intended generator or interpreter.

2.4 Complete standard conformance

DMIS is a large and complex standard. Vendors do not need to implement the entire standard. However it is possible that a DMIS application may conform to the entire standard.

2.5 Application profiles

Currently, there are two application profiles that have been defined along with their two character identifier, Prismatic (PM) and Thin Walled (TW), to each of the three levels, as well as seven (7) addenda that expand the levels of conformance. The addenda along with their two character identifier are: Rotary Tables (RT), Multi Carriage(MC), Contact Scanning (CS), In-Process Verification (IP), Quality Information Systems (QI), Measurement Uncertainty (MU), and Soft Gaging (SG).

2.6 Conformance levels

Each application profile or addendum will define three levels of conformance.

— Level 1 – is essential (required) to meeting the profile's goals

— Level 2 – is important to meeting the profile's goals

— Level 3 – is beneficial to meeting the profile's goals.

2.7 Program Conformance Designation

Each DMIS program or DMIS module shall designate its conformance classes via a collection of one or more pairs of application profile / addenda with conformance level. An example of a DMIS program conformance designation within a DMISMN statement is:

```
DMISMN,'DMIS Program Interoperability',5.2,PM,2,RT,1,MC,1
```

2.8 Conformance claim

Upon positive notification from a recognized DMIS conformance certification service, a DMIS application may make a conformance claim.

An example of a conformance claim is:

MyProduct version x.y from DMISVendor conforms to DMIS Prismatic Profile, Level 2 with Rotary Table, Level 1.

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this international standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this international standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid international standards.

American Standard Code for Information Interchange (ASCII)

All DMIS vocabulary consists of characters from the ASCII 255-character set.

ANSI/ASME B89.4.1-1997

All terminology that refers to Coordinate Measuring Machines is used in accordance with this standard.

ANSI/CAM-I 101 1990

There are references to prior versions of DMIS in this document to provide for backward compatibility with the language.

ANSI Y14.26M-1987

Development of DMIS has been monitored by the community which has developed and maintained the Initial Graphics Exchange Specification (IGES).

APT: ANSI X3.37-1987

DMIS is similar in syntax to the syntax of the numerical control programming language APT (Automatically Programmed Tools). Some of the words in DMIS are taken from the APT vocabulary. It should be noted, however, that some words in DMIS, which are identical to words in APT, may have different meanings or a different syntax. The APT vocabulary should not be used as a reference when interpreting the meaning of DMIS vocabulary words.

ASME Y14.5M - 1994

All ASME Y14.5M - 1994 geometric dimensions and tolerances are incorporated in DMIS.

Dimensional Mark-up Language Specification for the transfer of dimensional inspection results (DML).

http://www.dmisstandard.com/DML/

ISO 10303 TC184/SC4

The emerging Standard Exchange for Product Model Data (STEP) international standard defines the computer-interpretable representation and unambiguous exchange of product information throughout the life of a product. As this community progresses toward a complete product data definition exchange standard, DMIS will evolve to maintain compatibility with their developments. DMIS has been designed with this compatibility and growth path in mind.

ISO 11562:1996 Geometrical Product Specifications (GPS) – Surface texture: Profile method – Metrological characteristics of phase correct filters.

ISO/IEC 14977:1996 (E) Information technology - Syntactic metalanguage - Extended BNF

4 Terms and definitions

The following terms and acronyms appear throughout this standard. When they are used, it is only within the context of the limited definitions set forth in this section.

4.1
actuals
Referring to features or tolerances, the actuals are the results computed from measured or constructed features. Actuals are determined by the measuring device and do not exist until a measurement or construction has occurred.

4.2
Automatically Programmed Tools (APT)
A computer language for numerical controlled (NC) machines.

4.3
calibration sequence
A sequence of statements contained in a CALIB…ENDMES block. A calibration sequence may be either a rotary table calibration sequence initiated with the CALIB/RTAB statement or a sensor calibration sequence initiated with the CALIB/SENS statement.

4.4
carriage
The moving component of a machine that carries the ram.

4.5
characterization file
A file created by a vendor that lists the DMIS statements and functions it supports.

4.6
Computer Aided Design (CAD) system
A computerized system for design and analysis. With respect to this standard, CAD systems are used to develop inspection part programs and to analyze inspection results.

4.7
Coordinate Measuring Machine (CMM)
A computerized inspection device for gathering discrete point data and analyzing dimensions and tolerances. Data is usually acquired using a touch probe or scanning probe as a sensor.

4.8
Dimensional Measuring Equipment (DME)
A class of equipment used to inspect parts and evaluate dimensions and tolerances. DMEs include, but are not limited to, coordinate measuring machines, video inspection equipment, optical comparators, robotic measuring devices, theodolites, photogrammetry, and laser-based measuring devices.

4.9
Dimensional Measuring Interface Standard (DMIS)
This is the name given to this standard. Reference is often made to DMIS vocabulary, DMIS inspection programs, DMIS files, and so forth. These are data files which conform to the formats and structure set forth in this standard.

4.10
DMIS National Standards Committee (DNSC)
The DMIS National Standards Committee (DNSC) was the committee formed in 1990 and accorded responsibility for the maintenance and support of DMIS until June of 2005 at which time the committee name changed to the DMIS Standards Committee (DSC). The DNSC was an accredited standards committee operating under ANSI-approved DMSC procedures as the accredited organization.

4.11
DMIS Standards Committee (DSC)
The DMIS Standards Committee (DSC) is the committee accorded responsibility for the maintenance and support of DMIS. The DSC is an accredited standards committee operating under ANSI-approved DMSC procedures as the accredited organization.

4.12
Dimensional Mark-up Language (DML)
The Dimensional Mark-up Language Specification is a neutral, non-proprietary, open format developed by the Automotive Industry Action Group (AIAG) that provides interoperability between the CMM, Coordinate Measuring Machines. The language is in a XML Schema Format.

4.13
DML Common Space
The coordinate system to which all features and transformations are referenced within DML.

4.14
Extended Backus-Naur Form (EBNF)
A syntactic metalanguage which is a notation for defining the syntax of a language by use of a number of rules. Each rule names part of the language (called a non-terminal symbol of the language) and then defines its possible forms.

4.15
feature
A feature is a point or group of points, which are identified and referred to as a feature. Usually, features are geometric elements of a part such as points, lines, planes, and circles.

4.16
filename
The term filename is used to indicate a computer readable file. A 'filename' can be a filename and extension or it can contain a full path specification.

4.17
geometric compensation
Software specific to the DME which provides for increased accuracy by mapping the DME volume.

4.18
High Level Language (HLL)
A collection of statements providing advanced capabilities that are identified in clause 5.2.4.

4.19
Initial Graphics Exchange Specification (IGES)
A specification for the exchange of CAD data.

4.20
inner
Signifies that probing occurs from inside, and material lies exterior to, the envelope of the defining feature.

4.21
In Process Verification (IPV)
A computerized system linking inspection and process verification to the manufacturing process.

4.22
input files
Unless otherwise noted, input files (and input programs) are inspection programs input to the dimensional measuring equipment.

4.23
label
A label is that part of a DMIS statement which fully identifies a feature, tolerance, coordinate system, sensor, output data format, datum, mating, macro routine, text string, or program line, etc. A label consists of a label type followed by a label name enclosed in parentheses except in the case of a label identifying a program line which contains only a label name enclosed in parentheses with no label type.

4.24
label type
The label type is that portion of a DMIS label which specifies the type of a DMIS entity which can be a nominal or actual feature, nominal or actual tolerance, nominal or actual coordinate system, a nominal or actual sensor, an output data format, a datum, a mating, etc., or one of several text string types including a clamp id, clamp serial number, a dimensional measuring device id, a DME software id, etc.

4.25
label name
The label name is that portion of a DMIS label enclosed in parentheses which identifies an individual DMIS entity out of all DMIS entities of the same label type.

4.26
line reducible feature
A line reducible feature is a feature that can be reduced to a geometric line. Line reducible features are given in (Table 8 — Reducible features).

4.27
measurement sequence
A sequence of statements contained in either a MEAS…ENDMES block or an RMEAS…ENDMES block.

4.28
multiple carriage
A DME having two or more carriages.

4.29
nominal
Referring to a feature or tolerance, this is the "as-designed" value defined in the product definition of the part. The nominal is known prior to measurement and is the value to which the actuals are compared for computing out-of-tolerance conditions.

4.30
orientation
The angular relationship to a coordinate system of an entity that can be a nominal or actual feature, nominal or actual sensor, or nominal or actual coordinate system.

4.31
orientational deviation
(RMEAS): The angular difference between the nominal and actual vectors of the measurement plane and the feature.

4.32
outer
Signifies that probing occurs from outside, and material lies interior to, the envelope of the defining feature.

4.33
output files
Unless otherwise noted, output files are inspection results output from the dimensional measuring equipment.

4.34
part coordinate system (PCS)
A datum reference frame associated with the part to be measured.

4.35
plane reducible feature
A plane reducible feature is a feature that can be reduced to a geometric plane. Plane reducible features are given in (Table 8 — Reducible features).

4.36
point reducible feature
A point reducible feature is a feature that can be reduced to a geometric point. Point reducible features are given in (Table 8 — Reducible features).

4.37
positional deviation
(RMEAS): The scalar difference between the nominal and actual locations of the measurement plane and the feature.

4.38
post-processor
A computer program which converts a file in DMIS format to a machine-specific format.

4.39
pre-processor
A computer program which converts a data file in a source machine specific format to a file in DMIS format.

4.40
Quality Information System (QIS)
A computerized system for the analysis and storage of inspection data.

4.41
ram
The moving component of a machine that carries the sensor.

4.42
receiving system
A CAD system, QIS, or other computerized system which receives a DMIS results file that is output from a DME.

4.43
rotary table system
The application of a rotary table(s) as an adjunct to a DME.

4.44
selective processing
An inspection process having the required conditional branching to alter the control of a program based on inspection results or pre-set variables. This capability is provided for by incorporation of HLL extensions.

4.45
sensor
A device used on a DME to acquire measurement data. For a CMM, the sensor is typically a touch probe. For a video inspection machine, the sensor is typically a camera. Other sensor types include infrared, lasers and non-contact electrical capacitance.

4.46
Standard for the Exchange of Product Model Data (STEP)
A specification for the exchange of product model data.

4.47
statement
Any DMIS definition or command.

4.48
tolerance
A measure of acceptable deviation from nominal conditions. Geometric tolerances in DMIS are as defined in the ASME Y14.5M-1994 Standard for Dimensioning and Tolerancing.

4.49
tool holder
A device that holds or stores sensors for automatic sensor changing.

4.50
vendor
Someone who provides hardware or software that interfaces to others through the DMIS vocabulary.

4.51
video inspection machine
A computerized inspection device which acquires data for analyzing dimensions and tolerances. The sensor is typically a camera.

4.52
vision inspection machine
A computerized inspection device which acquires data for analyzing dimensions and tolerances. The sensor is typically a camera or group of cameras.

4.53
work measuring zone
The measurement volume of a machine as specified by the supplier. More than one work-measuring zone can be specified for a given machine. Parallel systems, for example, would have a work-measuring zone defined for each carriage. The work measuring zone is also referred to as the working volume.

4.54
workpiece
An object or part to be inspected.

5 Language reference

5.1 Syntax and structure

The DMIS vocabulary is similar in syntax to the APT NC programming language, with major and minor words separated by the slash character "/". Output data formats are similar in syntax to the input data formats. Translators for the DMIS vocabulary can be simple single pass interpreters or complex multiple pass compilers, depending on the implementation method which each individual vendor chooses.

There are two basic types of DMIS statements: process-oriented command statements and geometry-oriented definition statements. Process command statements consist of motion statements, machine parameter statements, and other statements which are unique to the inspection process itself. Definition statements, on the other hand, are used to describe geometry, tolerances, coordinate systems, and other types of data, which may be included in a CAD database. Typically, part models do not include all of the data needed in the DMIS interface, so supplementary data must be added manually. The evolution of CAD systems, though, is in the direction of complete part models, and DMIS has been designed to be compatible with this growth path.

The DMIS vocabulary consists of ASCII characters which are combined to form words, labels, parameters and variables. These are then combined to form definition statements and command statements. These statements are combined to form program blocks. Program blocks, along with more statements, are combined to form entire DMIS programs.

5.1.1 Characters

In this specification, ASCII characters are represented by the corresponding decimal (not hexadecimal) numbers from the ASCII character table.

When DMIS statements are interpreted, upper and lower case alpha characters are considered to be the same. All of the following, for example, are interpreted as being the same by the DMIS system:

```
TEXT
Text
text
teXT
```

The only exception to this rule is that upper and lower case characters are significant in text strings passed with the TEXT, FILNAM, and other statements where text is enclosed with apostrophes. Note also that datum labels can only be upper case alphanumeric characters.

5.1.2 Numbers, words, label names, text strings, vectors, parameters, variables, and expressions

Characters are combined to form numbers, vocabulary words, labels, text strings, parameters, variables, and expressions.

5.1.2.1 Literal numbers

Literal numbers in DMIS input and output files consist of an optional sign, followed by one or more digits, optionally with a decimal point before the digits, between two digits, or after the last digit (+7.40, for example). Digits are the characters 0 (ASCII 48) through 9 (ASCII 57). A sign is a plus sign (+, ASCII 43) or a minus sign (-, ASCII 45). A decimal point is ASCII 46. Scientific notation (such as 3.7E02) is not allowed. All numbers are interpreted as decimal numbers. Only base ten numbers are used in DMIS.

Literal numbers in DMIS can be positive or negative. When negative, the number is preceded by a minus sign character '-'. Positive numbers are either preceded by a plus sign character '+', or by no sign.

A literal integer is a literal number that has no decimal point. A literal real number is a literal number that has a decimal point. A literal integer may be used, however, wherever a real number is required in DMIS.

The literal real numbers +0.0 and -0.0 represent the same number as 0.0; similarly for the literal integers +0, -0, and 0.

Every literal number is positive, negative, or zero. The literal numbers +0.0 and +0 are zero, not positive. The literal numbers -0.0 and -0 are zero, not negative.

Numerical data for angles can be represented as degrees, minutes and seconds separated by the colon character (:, ASCII 58), for example 4:03:47.00. Numerical data for times can be represented as hours, minutes and seconds separated by the colon character, for example 4:03:47. These representations cannot be used literally with mathematical operators or functions (COS(3 * 4:03:47.00), for example, is not allowed). Angles and times represented using a colon are not considered to be literal numbers. They are structured data with the usual rules for what characters are allowed; for example, the number of minutes must consist of exactly two characters, the first of which is a digit between 0 and 5, and the second of which is any digit.

5.1.2.2 Words

5.1.2.2.1 Major words

Major words consist of a minimum of two alphanumeric characters. A major word may either be a DMIS statement in itself, or it may indicate a class of statements. When it indicates a class of statements, the major word is modified by one or more minor words. Major words are listed in (Table 1 — DMIS major words). Characterization file major words are listed in (Table 2 — Characterization file major words).

Table 1 — DMIS major words

ACLRAT	ENDSEL	PREVOP
ALGDEF	ENDSIMREQT	PROCID
ASSIGN	ENDXTN	PROMPT
BADTST	EQUATE	PSTHRU
BOUND	ERROR	PTBUFF
CALIB	EVAL	PTMEAS
CALL	EXTENS	PUSH

CASE	EXTFIL	QISDEF
CLMPID	FEAT	RAPID
CLMPSN	FEDRAT	READ
CLOSE	FILDEF	RECALL
CMPNTGRP	FILNAM	REFMNT
CNFRMRUL	FINPOS	REPORT
CONST	FIXTID	RESUME
CRGDEF	FIXTSN	RMEAS
CRMODE	FLY	ROTAB
CROSCL	FROM	ROTATE
CRSLCT	GEOALG	ROTDEF
CUTCOM	GEOM	ROTSET
CZONE	GOHOME	SAVE
CZSLCT	GOTARG	SCNMOD
DATDEF	GOTO	SCNSET
DATSET	GROUP	SELECT
DATTRGDEF	IF	SENSOR
DECL	INCLUD	SIMREQT
DECPL	ITERAT	SNSDEF
DELETE	JUMPTO	SNSET
DEVICE	KEYCHAR	SNSGRP
DFTCAS	LITDEF	SNSLCT
DISPLY	LOCATE	SNSMNT
DMEHW	LOTID	TECOMP
DMEID	MACRO	TEXT
DMESW	MATDEF	THLDEF
DMESWI	MEAS	TOL
DMESWV	MFGDEV	TOOLDF
DMIS	MODE	TRANS
DMISMD	OBTAIN	UNCERTALG
DMISMN	OPEN	UNCERTSET
DO	OPERID	UNITS
ELSE	OUTPUT	VALUE
ENDAT	PAMEAS	VFORM
ENDCAS	PARTID	WINDEF
ENDDO	PARTRV	WKPLAN
ENDFIL	PARTSN	WRIST
ENDGO	PATH	WRITE
ENDIF	PLANID	XTERN
ENDMAC	POP	XTRACT

ENDMES	PRCOMP	

Table 2 — Characterization file major words

CHFIL1	ENDCH1
CHFIL2	ENDCH2
CHFIL3	ENDCH3
CHFILE	ENDCHF

5.1.2.2.2 Minor words

Minor words consist of a minimum of one alphanumeric character. Some minor words are preceded by a minus sign character. A minor word is used to modify a major word or to describe certain parameters in the statement. Minor words are listed in (Table 3 — DMIS minor words). Characterization file minor words are listed in (Table 4 — Characterization file minor words).

Table 3 — DMIS minor words

-PRBRAD	CYLCTY	INTOF	PART	SNS
-XDIR	CYLNDR	INTOL	PARTO	SNSET
-YDIR	CYLRADSEGMNT	IP	PATERN	SNSLCT
-ZDIR	CZSLCT	IPM	PAUSE	SNSMNT
2D	DATA	IPMM	PCENT	SOUND
2RC	DATTRG	IPS	PCS	SPART
3D	DEFALT	IPSS	PECK	SPH
4POINT	DEFLECTION	JOINTCONFIG	PERP	SPHERE
ABOVE	DELAY	KEEP	PERPTO	SPHRADSEGMNT
ABSL	DELETE	LAMBDAC	PICTURE	SPLINE
ACEL	DEPTH	LASER	PITCH	START
ACLRAT	DEV	LE	PIXBTN	STAT
ACT	DFTCAS	LEFT	PLANE	STDDEV_LIMIT
ADJUST	DIAM	LEFTY	PLOT	STOP
ALGOR	DIM	LIMIT	PM	STOR
ALL	DIRECT	LINE	POINT	STRGHT
ALLAXESTOUCH	DISK	LINEAR	POL	STROBE
ALLSA	DIST	LIST	POS	SUPPORTED
ALPHA	DISTANCE	LMC	POSACL	SURF
AMT	DISTB	LN2LN	POSVEL	SURFACE
AND	DME	LOCAL	PP	SYM
ANGDEC	DMIS	LONG	PRBRAD	SYMPLN
ANGDMS	DMISMD	LOW	PRCOMP	SYNC
ANGL	DMISMN	LOWPASS	PRINT	SYS

ANGLB	DML	LSTSQR	PRNTCHAR	TANGPL
ANGLE	DOUBLE	LT	PROBE	TANTO
ANGLR	DRAG	MACH	PROFL	TECOMP
ANGRAD	EDGELN	MAJOR	PROFP	TEMP
APPEND	EDGEPT	MAN	PROFS	TEMPC
APPRCH	EDIT	MATRIX	PROG	TEMPF
ARC	ELIMINATE	MAX	PROJCT	TEMPWC
ATTACH	ELLIPS	MAXINS	PROJLI	TEMPWF
AUTO	ELONGCYL	MC	PROJPT	TERM
AVG	END	MCS	PT2LN	TEXT
AVGDEV	ENDCAS	MESACL	PT2PL	THRU
AXDIR	ENDDO	MESVEL	PT2PT	TIME
AXIAL	ENDSEL	METER	PTBUFF	TITLE
BACK	ENTITY	MIDLI	PTDATA	TORRADSEGMNT
BADTST	EQ	MIDPL	PTMEAS	TORUS
BANDPASS	ERR	MIDPT	QI	TR
BELOW	ERRMODE	MIN	QUERY	TRIGER
BEZIER	ERROR	MINCIR	RAD	TRMATX
BF	EXCEPT	MINCON	RADIAL	TRNOUT
BND	EXTERN	MINMAX	RADIUS	TRUE
BOOL	EXTREM	MINOR	RAWDAT	TW
BOUND	FALSE	MM	RCTNGL	UNBND
BOX	FDATA	MMC	REAL	UNCERT
BSPLIN	FEAT	MMPS	RECFILT	UNITS
BUILD	FEATUR	MMPSS	REPORT	UNKNOWN
BUTTON	FEDRAT	MNTLEN	REQUNCERT	UNSUPPORTED
CART	FEET	MODE	RES	USERDF
CCW	FILNAM	MODEL	RETRCT	USETOL
CHAR	FILTER	MOVEPT	RETRIEVE	VEC
CHECK	FINPOS	MPM	REVSURF	VECBLD
CHORD	FIRST	MPMM	RFS	VECTOR
CIRCLE	FIXED	MU	RIGHT	VENDOR
CIRCULAR	FLAT	NAME	RIGHTY	VERSION
CIRLTY	FLIP	NE	ROTACL	VERTEX
CLRSRF	FOCUSN	NEXT	ROTARY	VIDEO
CM	FOCUSY	NODATA	ROTNUL	WAIT
CODE	FORCE	NOFLIP	ROTORG	WIDTH
COG	FORM	NOM	ROTTOT	WKPLAN
COMAND	FX	NOMINL	ROTVEL	XAXIS
COMM	FZ	NONCON	ROUND	XDIR

COMMON	GAUSS	NONE	RPM	XORIG
COMP	GCURVE	NOROT	RPMM	XVEC
COMPOS	GE	NOT	RPTSYC	XYAXIS
COMPOUND	GEOALG	NOTOUCH	RTAB	XYDIR
CONCEN	GLOBAL	NOTRAN	RULE	XYPLAN
CONE	GOTO	NUMERIC	RULEINTOL	XYZAXI
CONRADSEGMNT	GRID	NURBS	RULEOUTOL	XYZDIR
CONT	GROUP	OBJECT	RULEUNDET	YAXIS
CONTIN	GSURF	OBLQ	RULEUNSUP	YDIR
CORTOL	GT	OFF	RY	YORIG
COUNT	HEADCS	OFFSET	SCALEX	YZAXIS
CPARLN	HEADTOUCH	ON	SCALEY	YZDIR
CPROFL	HELICAL	OPEN	SCNACL	YZPLAN
CPROFS	HIGH	OPER	SCNMOD	ZAXIS
CRAD	HIGHPASS	OPTIMAL	SCNVEL	ZDIR
CRITICAL	HIST	OR	SEARCH	ZORIG
CRMODE	HUMID	ORIENT	SENS	ZVEC
CRNOUT	ILLEGALTOUCH	OUTER	SENSOR	ZXAXIS
CROSCL	INCH	OUTFIL	SEQNTL	ZXDIR
CRSLCT	INCR	OUTOL	SF	ZXPLAN
CT	INDEX	OUTPUT	SGAGE	
CURENT	INFRED	OVERWR	SHORT	
CURVE	INNER	PARAM	SIMUL	
CW	INPUT	PARLEL	SINGLE	
CYCLE	INTGR	PARPLN	SIZE	

Table 4 — Characterization file minor words

CMM	FORME
ENDSPT	FORMF
FORMA	FORMG
FORMB	FULL
FORMC	NONE
FORMD	NS

5.1.2.3 DMIS reserved words

DMIS reserved words consist of DMIS major words defined in (Table 1 — DMIS major words), DMIS minor words defined in (Table 3 — DMIS minor words) and intrinsic function names listed in (Table 6 — Intrinsic function words).

5.1.2.4 Labels

Labels consist of two components, a one to three character label type (for example F, TA, DAT...), followed immediately by a label name enclosed within parentheses. Labels are assigned in the inspection program to name features, tolerances, coordinate systems, sensors, output data formats, datums, macro routines, text strings, and program statements, etc; and each of these entities has a unique label type that is defined by DMIS.

All label names, except for datum and datum target labels, are from one to sixty-four (64) characters where the only permissible characters are the ASCII printable characters (code 32 to 126) except for the following:

Code 34 Double quote ----------"
Code 36 Dollar sign ----------$
Code 39 Apostrophe-quote-----'
Code 40 Left parenthesis -------(
Code 41 Right parenthesis -----)
Code 64 Commercial at----------@
Code 91 Left bracket----------[
Code 93 Right bracket ----------]

Datum label names are from one to two upper-case alpha characters for single datums, and two to four upper-case alpha characters with a ASCII code 45, dash-hypen-minus '-' in between for compound datums (for example AA-ZZ). Datum target label names are from one to two upper-case alpha characters followed by a numeric (for example A1, A2, BB3).

All label names are enclosed in parentheses. Leading spaces and trailing spaces within a label name are ignored. For example, FA(Hole 1) is equivalent to FA(Hole 1). Label names (except for feature nominals) can be issued only once and cannot be redefined in a particular program; and labels with the same label name (within parentheses) but with a different label type are unique.

In addition to directly specifying label names, the indirect reference '@' operator is available. At most one level of indirect reference may be used. With the indirect reference method, the '@' character precedes the name of a CHAR variable that contains a label name that is valid for the label type. If the character variable was declared as multi-dimensional, its name must be followed by an array index. The array index (base) begins at the number 1. Refer to example A.1.

Jumptargets are a special type of label that have an implied label type and are used for example in ERROR, JUMPTO or RESUME statements.

5.1.2.5 Text strings

Text strings are used extensively in DMIS. A text string is a grouping of UTF8 printable characters (code 32 to 126), that is enclosed with apostrophes. All UTF8 printable characters are allowed within a text string, but when an apostrophe is required it must be preceded by an additional apostrophe, for a total of two apostrophes. Space, UTF8 code 32, is considered as a printable character. An empty string is specified by using only two apostrophes with no characters in between, that is ''. For example if an operator prompt requesting a Supplier's part number were "Enter the Supplier's part number". An appropriate statement would be:

```
TEXT/QUERY,(Sup_Part),20,PRNTCHAR,LEFT,'Enter the Supplier''s part number'
```

In the word "Supplier"s" there are two apostrophes preceding the "s" this indicates that a single apostrophe is to be contained in the literal string, and that the apostrophe is not the end of the text string. This must be two individual apostrophe characters ('') not a double quote character (").

If a text string is very long, it may be divided into pieces and placed on two or more lines using the line continuation character (a single dollar sign "$"). If the line continuation character is used to divide a text string, it must be the last character other than spaces and tabs (before the carriage return and line feed) in the line to be continued. If a text string is divided using '$', any spaces and tabs following the '$' at the end of the line are not part of the text string. A single apostrophe must be the last character to designate the end of the text string. An appropriate statement would be:

```
TEXT/QUERY,(Sup_Part),20,PRNTCHAR,LEFT,'Enter the Supplier''s part number for the $
parts that we are going to measure in this group.  Do not enter more than $
a single part number.  All parts must have the same part number.'
```

The literal string resulting from this TEXT/QUERY statement would be held in the text label (Sup_Part) as:

Enter the Supplier's part number for the parts that we are going to measure in this group. Do not enter more than a single part number. All parts must have the same part number.

5.1.2.6 Parameters

Parameters that appear in DMIS statements are separated from each other and from minor words by commas. The type of each parameter is either an integer number, a real number, a character string or a label. Parameter types are defined in section 6, the Statement Reference section of this document.

A parameter's value can be set in a DMIS statement by an expression evaluating to the appropriate type. A numeric expression evaluating to an integer number may be used to replace a real number parameter. A numeric expression evaluating to a real number cannot be used to directly replace an integer number valued parameter. The INT() intrinsic function can be used to create an integer valued expression from a real number.

Otherwise, substituting a parameter in a DMIS statement of the wrong type causes an error to be flagged by the DME.

For example, the following statements will cause errors to occur because the wrong literal type is used:

```
GOTO/'1','2','3'
MEAS/POINT,F(pnt),2.3
TEXT/OPER,12*5
```

Use of the wrong variable type as parameters will cause similar errors. If we have declared the following variables:

```
DECL/REAL,RVAL
DECL/INTGR,IVAL
DECL/CHAR,80,CVAL
```

Then the following statements will cause errors to occur:

```
MEAS/CIRCLE,F(cir),RVAL
GOTO/0,0,CVAL
```

However, the following statement is correct:

```
GOTO/0,0,IVAL
```

5.1.2.7 Variables

All DMIS variables are declared before being used and include Boolean, character string, double, integer, long, real, and vector data types. Variable names consist of alphanumeric and underscore characters. The first character of a name must be a letter, and names may have up to sixteen(16) characters. Variable names must not use DMIS reserved words (refer to clause 5.1.2.3) and are not case sensitive. Variables may be explicitly declared as local, global, or common. All variables without an explicitly declared scope in the main program are global to the entire program. Variables without an explicitly declared scope in a module or macro are local to that module or macro. Multi-dimensional variable arrays may be declared. The variable array index (base) begins at the number 1. Further information about variables can be found throughout section 5.2.

A previously declared and assigned variable can be substituted for any parameter (that is, numeric value, Boolean value, vector value, character string, or label name) of the proper data type in any DMIS statement. When such a variable appears in a statement, any DMIS output text shall contain the value at the time of execution.

```
DECL/CHAR,10,myclamp
myclamp=ASSIGN/'CLAMP12'
```

```
CI(CLAMP)=CLMPID/myclamp
```

Variables declared with the DECL statement have no particular value until explicitly assigned a value.

5.1.2.7.1 Numeric types and storage limits

DMIS supports the following numeric variable types: DOUBLE, REAL, LONG and INTGR. DOUBLE and REAL are floating point representations of real numbers and LONG and INTGR are representations of integer numbers.

DMIS places the following storage limits for these numeric types:

DOUBLE – a minimum of 8 bytes of storage

REAL – a minimum of 4 bytes of storage

LONG – a minimum of 4 bytes of storage giving a minimum range –2147483648 to 2147483647

INTGR – a minimum of 2 bytes of storage giving a minimum range –32768 to 32767

The actual number ranges for REAL and DOUBLE that the above storage requirements allow may be system dependent. Because the numeric limits that these storage limits imply will represent numbers that are well beyond anything encountered in normal metrology, DMIS does not specify specific numeric limits for the DOUBLE and REAL storage types.

5.1.2.8 Expressions

A DMIS expression is formed from operators, operands and parentheses and can represent a simple data reference or a computation. Operands include literal numbers and text strings, variables and intrinsic function calls. The simplest expression consists of a single operand. More complex expressions consist of combinations of operands and operators or combinations of operands, operators and parentheses.

Expressions are referred to by the type of their result. A numeric expression evaluates to a number and may use the numeric operators described in section 5.3.1.1. A logical expression evaluates to a Boolean value (.TRUE. or .FALSE.) and may use the relational and/or logical operators described in sections 5.3.1.2 and 5.3.1.3. A vector expression evaluates to a vector and may use the vector operators described in clause 5.3.1.4.

A character expression evaluates to a character string and can only be a single operand: either a literal text string, a character variable or a call to an intrinsic function returning a character string; there are no character operators in DMIS.

An expression of the proper type can be substituted for any parameter in any DMIS statement. For example, a numeric expression can be substituted for a numeric parameter in a DMIS statement, and a character expression may replace any text string parameter in a DMIS statement.

The numeric type to which a numeric expression evaluates depends on the types of the operands in the expression. If all the operands are of the same type, the result will be of that type. If operands of different types are used then the type of the result will be determined by the highest ranking operand using the following ranking: type DOUBLE has the highest ranking, followed by REAL, followed by LONG and then by INTGR with the lowest ranking.

In a numeric expression containing sub-expressions enclosed by parentheses, the numeric type of each sub-expression is determined by the operands within the parentheses.

EXAMPLE 1:

```
$$
$$ The following example demonstrates a complex expression.
$$ The numeric type of the division in parentheses will be integer even though
$$ a real number appears outside the parentheses in the same expression.
```

```
$$
4.0+(2/3)
```

In the above example of a complex expression the numeric type of the division in parentheses will be integer even though a real number appears outside the parentheses in the same expression.

EXAMPLE 2:

```
$$
$$ The following example demonstrates substituting a character expression
$$ for a literal text string parameter
$$
TEXT/OPER,CONCAT('The current time is: ',STIME())
```

5.1.3 Variable assignments and use

DMIS provides for the assignment of previously declared variables. Variables may be obtained from a definition parameter (for example, OBTAIN), set to a measurement result (for example, VALUE), or assigned a value (for example, ASSIGN).

A previously declared and assigned variable can be substituted for any parameter (for example a numeric value, a literal text string, or label name) of the proper data type in any DMIS statement. A character string (type CHAR) variable can be substituted for any single quoted (apostrophe) string in the DMIS syntax.

EXAMPLE

```
$$
$$ The following example demonstrates substituting a character variable for a
$$ literal text string parameter
$$
DECL/GLOBAL,CHAR,80,MYTEXT
MYTEXT=ASSIGN/'This is my text message to the operator'
TEXT/OPER,MYTEXT
```

5.1.4 DMIS command and definition statements

Major words, minor words, parameters, labels, text, and variables are combined to form either command statements or definition statements.

5.1.4.1 Command statements

Command statements direct the DME or receiving system to perform some function. Often a command statement contains a reference to a previous definition statement. In this case, the definition statement is used in the execution of the command statement.

The major word indicates the command statement or command statement type. This is followed by a slash character "/" and one or more minor words, parameters, labels, text, and/or variables.

In some cases, a major word by itself is valid. In this case, no slash, minor words, parameters, labels, text, or variables occur in the statement.

Example:

```
DFTCAS
```

In other cases, a major word and a minor word, with no parameters, labels, text, or variables, is a valid DMIS statement.

Example:

```
BADTST/ON
```

Sometimes a major word with parameters and no minor words, labels, text, or variables may be required.

Example:

```
FROM/12.341,427.284,183.294
```

In still other cases, a major word along with minor words, parameters, labels, text, and/or variables is required.

Example:

```
CALL/EXTERN,DME,'bolt_circle',WAIT,5,20
```

The basic structure of a command statement is as follows:

```
MAJOR_WORD/MINOR_WORD(S),PARAMETER(S)
```

5.1.4.2 Definition statements

Definition statements describe the detailed information for specific cases of geometry, tolerance, machine specifics, and other conditions pertaining to inspection part programming. The basic structure of a definition statement is similar to that of a command statement, except that the major word is preceded by a label and an equal sign "=". A definition statement describes one specific set of conditions and assigns a label to it for easy reference. The label is then used in another statement to refer to the object defined.

5.1.4.3 Command and definition statement examples

As an example of the various statements described above, consider the following list of valid DMIS statements:

```
FINPOS/ON
F(CIRCLE_1)=FEAT/CIRCLE,INNER,$
CART,10,10,5,0,0,1,4
MEAS/CIRCLE,F(CIRCLE_1),3
GOTO/10,10,5
PTMEAS/CART,12,10,5,-1,0,0
PTMEAS/CART,8,10,5,1,0,0
PTMEAS/CART,10,12,5,0,-1,0
ENDMES
```

— The first line is a command statement consisting of only a major word and a minor word, with no parameters. This directs the fine positioning feature of the DME to be enabled.

— The second line is a definition statement containing a major word, a label, minor words, and parameters. It describes the size, location, and orientation of a feature that is a circle, and assigns to it the name "CIRCLE_1". With the line continuation character "$", this statement is continued on the third line.

— The fourth line is a command statement consisting of a major word with a minor word and a parameter which directs the DME to measure a circle. This command statement also has a pointer to the definition of the circle.

— The next line is a command statement consisting of only a major word and parameters that direct the DME to move to the given position.

— The next three lines are command statements consisting of a major word with a minor word and parameters that direct the DME to take point measurements at the given locations.

— The last line is a command statement that is a major word by itself which indicates that the measurement sequence is complete.

A list of major words is given in (Table 1 — DMIS major words), and a list of minor words is given in (Table 3 — DMIS minor words). These words, when combined properly with the appropriate parameters, make up the valid DMIS vocabulary. A detailed description of each command statement and definition statement in the DMIS vocabulary is given in Section 6.

5.1.5 Delimiters, blank lines, spaces, and tabs

5.1.5.1 Delimiters

Apostrophes, brackets, commas, parentheses, and slashes are used as delimiters in DMIS.

5.1.5.1.1 Apostrophe

The apostrophe -'- is used to delimit the start and end of a text string.

EXAMPLE

```
FILNAM/'041380385 test dated 2005/10/17',05.1
```

Use a total of two apostrophes, one before the one required, when an apostrophe is required within a text string.

EXAMPLE

```
TEXT/OPER,'Use Paul''s setup instructions from the last job.'
```

This results in the following message sent to the display device:

Use Paul's setup instructions from the last job.

5.1.5.1.2 Brackets

Brackets '[' and ']' are used as delimiters. An array index or a series of array indices (separated by commas) is preceded with a left bracket and followed by a right bracket.

5.1.5.1.3 Comma

Commas ',' are used as general delimiters to separate minor words and parameters:

```
MAJOR_WORD/MINOR_WORD1,MINOR_WORD2,param1,param2
```

5.1.5.1.4 Parentheses

Parentheses '(' and ')' are used as delimiters. A label name is preceded with a left parenthesis and followed by a right parenthesis.

For example, if a circle has a label name of CIRCLE_1, its nominal is denoted by the label:

```
F(CIRCLE_1)
```

All labels have balanced parentheses. That is, a left parenthesis is always followed by a right parenthesis.

Below is an example of an unconditional branch to a jumptarget which is labelled "BEGIN":

```
GOTARG/...
GOTO/...
ENDGO
JUMPTO /(BEGIN)
SNSLCT/...
(BEGIN)
MEAS/CIRCLE,F(CIRCLE_1),5
PTMEAS/...
PTMEAS/...
```

The JUMPTO statement causes an unconditional branch to the statement labelled BEGIN, which precedes the MEAS/CIRCLE statement.

Parentheses are also used in arithmetic, relational, logical, and character expressions, refer to clauses 5.3.1.1, 5.3.1.2, 5.3.1.3, and 5.3.1.4.

5.1.5.1.5 Slash

The forward slash character '/' is used to separate major and minor words. A major word can either be a statement by itself, or can have some modifying minor words or parameters. When a major word is used by itself, it has no slash character:

```
MAJOR_WORD
```

Example:

```
ENDMES
```

When a major word is followed by a minor word, the slash follows the major word and precedes the minor word.

```
MAJOR_WORD/MINOR_WORD
```

Example:

```
PRCOMP/OFF
```

When a major word is followed by one or more parameter(s), the slash follows the major word and precedes the first parameter.

```
MAJOR_WORD/parameter(s)
```

Example:

```
FROM/302,481,183
```

In the output of GCURVE, GSURF, and RAWDAT, each data point is preceded by a slash without major or minor words.

The slash character is also used as the division operator in numeric expressions.

5.1.5.2 Blank lines, spaces, and tabs

Blank lines have no significance in DMIS. Spaces and tabs have no significance except inside character strings and (spaces only) inside label names. Blank lines are insignificant and are ignored during translation. Spaces and tabs are not allowed within DMIS words, parameters and variables.

Spaces but not tabs may be used inside label names (but leading and trailing spaces are not part of a label name).

Spaces and tabs may be used:

a) between tokens of the language (DMIS words, variable names, numbers, and delimiters).

b) inside text strings passed in TEXT, FILNAM, and other statements using 'text'.

c) before or inside programming comments.

d) after a line continuation character.

Spaces and tabs may not be used anywhere else. Spaces and tabs have no significance where they are correctly used, except in text strings and label names.

Blank lines, spaces, and tabs may add clarity in manual programming or analysis, but they are ignored by computer programs in the translation of a DMIS file (except spaces in label names and spaces and tabs in text strings).

For example:

```
F(  CIR 1) = FEAT/CIRCLE, INNER,CART, 10, 10, 5, 0, 0, 1, 25
```

```
MEAS/CIRCLE, F(CIR 1   ), 5
```

is interpreted the same as:

```
F(CIR 1)=FEAT/CIRCLE,INNER,CART,10,10,5,0,0,1,25
MEAS/CIRCLE,F(CIR 1),5
```

5.1.6 Line length

A DMIS file contains statements of variable lengths. Each line is terminated with a carriage return(CR) and line feed(LF). There is no limit to the length of a statement. A statement can be on one line or split across multiple lines by using the line continuation character which is a single dollar sign '$'. A '$' used as a line continuation must be the last visible character on the line. Space and tabs (which are not visible) may occur after the '$' but before the carriage return and line feed that end the line. The line continuation character has no effect in a programming comment line. Each line is terminated with the system appropriate new line symbol. The maximum length of a line in a DMIS file will not exceed 65,536 characters including the new line symbol.

For example:

```
DISPLY/PRINT,V(OUT_1),TERM,$
V(OUT_2)
```

Is interpreted the same as:

```
DISPLY/PRINT,V(OUT_1),TERM,V(OUT_2)
```

And:

```
Q(MyLabel)=QISDEF/'This is a $100.00 program'$
,'100.00'
```

Is interpreted the same as:

```
Q(MyLabel)=QISDEF/'This is a $100.00 program','100.00'
```

5.1.7 Programming comments

Programming comments are lines of text inserted into a DMIS inspection program to aid in program debugging and to document portions of the program. They are not to be interpreted by any automated system.

Programming comments are signified by two dollars signs '$$' as the first two non-whitespace characters in a line. Spaces and tabs are allowed prior to the $$ to allow DMIS part programs to become structured per the programmer's desire. The computer program executing or converting DMIS commands will ignore all lines which begin with a double dollar sign.

The following are examples of a programming comment:

```
$$ This is a programming comment to be
   $$ ignored by the DMIS system.
$$ In this section of the program, the datums
   $$ will be established.
$$ This part of the program will be performed
   $$ in manual mode.
```

5.1.8 Operator input

5.1.8.1 The TEXT statement

Various forms of text are supported in the DMIS vocabulary. Header data are supported through the TEXT/QUERY and TEXT/RES statements. This allows each user to create header data to individual requirements.

The TEXT/QUERY statement is passed from the input program to the DME and requires a response from an operator or some other system. A part number, for example, may either be keyed in by an operator or read in

through a bar code scanner. The statement specifies a label name for the query, the number of characters allowed for the response, whether it is alphanumeric, numeric, or any printable UTF8 character, and whether it is right or left justified. When the response is entered, it is passed to the output file as a TEXT/RES statement with the same label name as the corresponding query. The other parameters (length, type, and justification) are also passed to the output file.

Typical queries in the input file might appear as follows:

```
TEXT/QUERY,(name),40,ALPHA,LEFT,'Enter Operator Name:'
TEXT/QUERY,(today),10,PRNTCHAR,LEFT,'Enter today''s date:'
```

Here, "name" and "today" are label names. The label names are followed by length, type, and justification parameters, and then by the message, which will be output to the operator.

The corresponding responses in the output file might appear as follows:

```
TEXT/RES,(name),40,ALPHA,LEFT,'John Smith'
TEXT/RES,(today),11,PRNTCHAR,LEFT,'2005/06/14'
```

The responses "John Smith" and "2005/06/14" have been typed in by the operator at run time.

Other forms of the TEXT statement allow text messages to be output to the screen (for the operator) and/or to the output file. Text may also be conditionally sent to the operator such that it will only be output if the DME executes a statement in manual mode. This allows operator instructions to be programmed and suppressed unless needed.

5.1.8.2 The PROMPT statement

The PROMPT statement provides a richer mechanism for operator prompting; and adds the ability to initialize program variables from operator input.

5.1.9 Data output

Output data from the DME can be sent to any of several devices. The data to be sent to each device can be specified to be in DMIS format, in vendor format, or both. Output of some DMIS statements is caused by the execution of an OUTPUT statement, whereas other DMIS statements are passed directly to all opened DMIS output files when executed. Each individual statement reference gives that statement's particular DMIS output characteristics. When a variable or expression is used as a parameter to a given input statement, it is the value of that variable or expression that appears in the output statement.

Example: The ROTDEF statement is passed to the output file when executed.

Output devices can be classified into two categories, those controlled by the DME and those controlled by the user. Input devices are controlled by the user.

5.1.9.1 Default output devices

The DME default device(s) to which output data will be sent is controlled by the DISPLY statement. This statement either turns all output off or specifies one or more of the following as output devices:

— TERM (display terminal at the DME)

— PRINT (printer)

— STOR (magnetic storage such as a floppy or hard disk)

— COMM (an auxiliary communications port)

The DISPLY statement must appear in the DMIS input program before any statements causing output. If the VFORM statement is referenced, it must be defined prior to its use in the DISPLY statement. The DMIS output file must have a FILNAM statement as its first line. When output is required on multiple devices, they must all be

identified in the first DISPLY statement to insure that the FILNAM statement will appear on all selected devices. As illustrated below the DISPLY statement can appear more than once in a DMIS program. It is important to note, however, that the last DISPLY statement is always active, and all previous settings are lost. To activate or reset a previous setting, a new DISPLY statement must be issued.

```
$$ Sample DMIS Input Program
DMISMN/'Sample DMIS 5.1 file        2007/02/04',05.1
V(Vendor_output)=VFORM/ALL
DISPLY/TERM,DMIS,PRINT,DMIS,STOR,V(Vendor_output)
FILNAM/'Sample DMIS Input Program',05.1
  executable statements
DISPLY/TERM,DMIS
  executable statements
DISPLY/PRINT,DMIS,STOR,V(Vendor_output)
  executable statements
ENDFIL
```

When STOR is used in the DISPLY statement, the data will be directed to the storage device. The filename chosen for the storage device is DME-dependent.

5.1.9.2 User defined input/output devices

The user device(s) to which output data will be sent, or from which input data is received, is controlled by the DEVICE and OPEN statements. The DEVICE statement identifies the device type (printer, terminal, communications port or storage device) and specifies a label for it. The OPEN statement opens the previously defined system device and establishes the connection's input/output attributes.

A device opened with the SNS, PCS, FEATUR, DML or RTAB parameter is accessed only with the SAVE or RECALL statements. Such devices are used for reading and writing DME specific or DML format system information.

Existing output storage devices can be appended to or overwritten. If the file specified in an OPEN statement being opened for either appending or overwriting does not exist, a new file is opened.

Output will go to a device opened with the FDATA parameter in either DMIS or vendor format as specified in the OPEN statement. The text and version from the FILNAM statement is written to new DMIS output files, that is, DMIS output files which do not already contain the FILNAM statement as their first line of output. The FILNAM statement is also passed to existing DMIS output files when they are opened for overwriting. If user defined DMIS output files are already open when the FILNAM statement is executed, the FILNAM statement is passed to those files if they do not yet contain the FILNAM statement as their first line of output. User defined DMIS output files to which the FILNAM statement is passed are storage devices initially opened with the OPEN/DID(lname),FDATA,DMIS,OUTPUT,OVERWR statement and which may be subsequently re-opened with the OPEN/DID(lname),FDATA,DMIS,OUTPUT,APPEND statement after being closed.

Output to a device opened with the DML parameter will include all F(), FA(), T(), TA(), D(), DA() information accumulated since the OPEN statement and ending at the SAVE statement. The DML Common Space is defined by the DA() specified parameter in the SAVE statement. Input from the RECALL statement will populate and define all F(), FA(), T(), TA(), D(), DA() information contained within the DML file. The DML Common Space will be equated to the coordinate system specified by the DA() parameter in the RECALL statement.

The ENDFIL statement at the end of the DMISMN main module will close all open devices. Open storage devices such as files are not deleted when closed. If the open device is a DMIS output file then the ENDFIL statement is passed to that output file before it is closed.

The CLOSE statement can be used to close devices opened with the OPEN statement before the end of a DMIS program, that is, before the ENDFIL statement is encountered. When a storage device such as a file is closed, it can be deleted by using the DELETE parameter. The storage device will not be deleted if either the KEEP or END

parameters are used, or when no parameter is specified. The KEEP parameter is of general use for all storage devices. The device or file is simply closed without being deleted. The END parameter is of particular use with DMIS output files. When the CLOSE/DID(lname),END statement is executed, an ENDFIL statement is passed to the storage device as if the end of the DMIS program had been reached, that is, as if the ENDFIL statement had been encountered.

```
DMISMN/'Sample program with DEVICE statements',05.1
V(V1)=VFORM/NOM,ACT,DEV,AMT
DISPLY/PRINT,V(V1),STOR,DMIS
FILNAM/'DEVICE statements program',05.1
$$ The DMIS default output file is opened and the FILNAM statement is
$$ passed to the file.  The vendor format printer is opened.

 executable statements

$$ Any DMIS output is passed to the default DMIS output file as
$$ statements are executed.

DID(mydata)=DEVICE/STOR,'/user/dmis/myfile.dat'
OPEN/DID(mydata),FDATA,DMIS,OUTPUT,OVERWR
$$ The user defined DMIS output file is opened and the FILNAM statement
$$ is passed to the file.

 executable statements

$$ Any DMIS output is passed to both the default DMIS output file and
$$ the user defined DMIS output file as statements are executed.

CLOSE/DID(mydata),KEEP
$$ The user defined DMIS output file is closed (no ENDFIL statement is
$$ passed to the file).

 executable statements

$$ Any DMIS output is now passed only to the default DMIS output file
$$ as statements are executed.

OPEN/DID(mydata),FDATA,DMIS,OUTPUT,APPEND
$$ The user defined DMIS output file is reopened.

 executable statements

$$ Again, any DMIS output is passed to both the default DMIS output
$$ file and the user defined DMIS output file as statements are
$$ executed.

CLOSE/DID(mydata),END
$$ The ENDFIL statement is written to the user defined DMIS output file
$$ before being closed.

ENDFIL
$$ The ENDFIL statement is written to the default DMIS output file
$$ before it is closed.  The vendor format print device is closed.
```

Below are additional examples of user defined output devices. They demonstrate how the DEVICE statement interacts with the OPEN, CLOSE and VFORM statements.

Example 1.

Two output devices, c:\users\reports\test_report_1.txt and c:\users\reports\test_report_2.txt, are opened simultaneously. Due to the OVERWR minor word option, both files will be overwritten should they already exist. The output of the measured point will be directed to DID(filestore1) in a user defined vendor format defined by the VFORM statement. In addition, the output of the point is also directed to output device DID(filestore2) in DMIS output format.

After the output has been performed, the sample DMIS program closes both devices. Device DID(filestore1) is closed with the KEEP minor word option which keeps the file, unlike the DELETE minor word option which would

remove the file after closing. Device DID(filestore2) is closed using the END minor word option which causes the ENDFIL statement to be written into the DMIS output file after which the file is closed.

```
FILNAM/'Some_File_Name',05.1
V(allout)=VFORM/ALL
DID(filestore1)=DEVICE/STOR,'c:\users\reports\test_report_1.txt'
DID(filestore2)=DEVICE/STOR,'c:\users\reports\test_report_2.txt'
OPEN/DID(filestore1),FDATA,V(allout),OUTPUT,OVERWR
OPEN/DID(filestore2),FDATA,DMIS,OUTPUT,OVERWR
F(point_1)=FEAT/POINT,CART,100,100,20,0,0,1
        MEAS/POINT,F(point_1),1
        ENDMES
OUTPUT/FA(point_1),TA(..),TA(..)
CLOSE/DID(filestore1),KEEP
CLOSE/DID(filestore2),END
```

Example 2.

In this example, every time the do-loop is re-executed, a new output file is created. The result of this sample code is that the files '\\data\mydrive\spc_001_file.spc through '\\data\mydrive\spc_130_file.spc will be created. Each of these output files will contain one measurement output of FA(Point_1) once this sample program is completed.

```
FILNAM/'This_File_Name',05.1
DID(incremental_file)=DEVICE/INCR,'\\data\mydrive\spc_???_file.spc'
V(act_dev)=VFORM/ACT,DEV
DECL/INTGR,i
DO/i,1,130,1
        OPEN/DID(incremental_file),FDATA,V(act_dev),OUTPUT,APPEND
        F(Point_1)=FEAT/POINT,CART,100,100,20,0,0,1
                MEAS/POINT,F(Point_1),1
                ENDMES
        OUTPUT/FA(Point_1),TA(..),TA(..)
        CLOSE/DID(incremental_file),KEEP
ENDDO
```

5.1.9.3 Output format

The data format in which the output data will be sent can be either DMIS format, a format unique to the DME vendor, or both. The DMIS format for output data is similar in syntax to the format for input data. Vendor formatted data varies widely with each vendor and is not standard.

The VFORM statement specifies the data content of vendor formatted output data for example nominals, actuals, deviations, etc. The actual physical format of these data, however, is left to each individual vendor. If the VFORM statement specifies nominals, actuals and deviations, for example, one DME may output nominals, then actuals, and then deviations. Another DME may output deviations, then nominals, and then actuals. One DME may output the data in a tabular format. Another DME may output data all in a single column. The DMIS vocabulary does not control the format of vendor data; it specifies the data content only.

5.1.9.4 DML Import/Export Capability

DML is an XML format definition tailored to the needs of dimensional results for discrete manufacturing. The purpose is to exchange results between applications that generate or use dimensional information. A typical scenario is where an inspection device collects dimensional data and sends the information to an SPC package for process analysis or a database for long term storage.

5.1.10 Program structure

DMIS program structure generally involves only one program file. However, the DMIS high level language (HLL) extensions allow modular programming. Modular programming provides the capability to reuse modules of existing code. As a result, a DMIS part program could consist of multiple files that as a whole define a complete inspection part program. Modularity can be obtained by using external file declarations (EXTFIL), include statements

(INCLUD), and modular (DMISMD) files. DMIS statements can be combined to form program units and blocks, including the program main unit and program module units.

5.1.10.1 Program units

Program units are designated by a file. There are two types of program units.

5.1.10.1.1 Program main

The program main unit is designated by the DMISMN statement and terminates with an ENDFIL statement.

5.1.10.1.2 Program module

A program module unit is designated by the DMISMD statement and terminates with an ENDFIL statement.

External DMIS files (that is, modules) containing declarations, procedures and functions for external reference by the program main unit involve the following:

```
DMISMD/'My_Module_ID',05.1
FILNAM/'My_Filename',05.1

  External Declarations

  Data Declarations

  Macro Definitions

  DMIS Statement Sequence

ENDFIL
```

The general structure of a DMIS high level language (HLL) module typically involves the following:

```
DMISMD/'SQRKEY_Feature_Definitions',05.1
FILNAM/'/user/dmis/defn/sqrkey',05.1
DECL/COMMON,REAL,X,Y,Z
DECL/GLOBAL,INTGR,sm_num
M(spechole)=MACRO/x,y,z
  executable statements
ENDMAC
M(crunchem)=MACRO/crunch_numbr,numl,num2
  executable statements
ENDMAC
ENDFIL
```

5.1.10.2 Program blocks

A program block is a logically grouped series of statements that perform some function. A program block is identified by the first and last statements in the block. There are multiple types of program blocks in DMIS.

5.1.10.2.1 Calibration sequences

Calibration sequences begin with the CALIB statement and are terminated with the ENDMES statement.

5.1.10.2.2 Measurement sequences

Measurement sequences begin with the MEAS or RMEAS statement and are terminated with the ENDMES statement.

5.1.10.2.3 Motion sequences

Motion sequences direct a series of non-measurement moves. Refer to clause 5.4.6 and sub-clauses for rules and examples.

5.1.10.2.4 IF Conditional branching

The IF...ENDIF block evaluates a logical expression and branches based on the test evaluation. An IF block begins with an IF statement, may contain an optional ELSE statement, and ends with an ENDIF statement.

5.1.10.2.5 Macro definitions

Macro routines are subroutines which can be executed with different values in the place of parameters. Macros begin with a MACRO definition statement and are terminated with an ENDMAC statement.

5.1.10.2.6 Select by case branching

The SELECT...ENDSEL block executes statements according to the results of one or more CASE tests. A SELECT block begins with a SELECT statement, contains multiple CASE blocks with an optional DFTCAS block statement, and terminates with an ENDSEL statement.

5.1.10.2.7 Case statement(s)

The CASE...ENDCAS block executes statement(s) bounded between a CASE and an ENDCAS statement dependent on tests on the SELECT argument.

5.1.10.2.8 Default case statement

The DFTCAS...ENDCAS block executes statement(s) bounded between a DFTCAS and an ENDCAS statement, only if none of the CASE clauses are activated in the SELECT block.

5.1.10.2.9 Looping sequence

The DO...ENDDO block provides the capability of repeating a sequence of instructions based on initial and limit values at an optionally specified increment. A DO loop begins with a DO statement and terminates with an ENDO statement.

5.1.10.2.10 External file declaration

The external file declaration block declares file(s) containing program units that are used within the local DMIS program. A external file declaration begins with an XTERN statement and terminates with an ENDXTN statement.

5.1.11 File structure

Programs (Input Files) and output files consist of a combination of command statements, definition statements, and program blocks.

5.1.11.1 Program file or input file

A DMIS inspection program consists of command statements, definition statements, and program blocks. DMIS programs always have a DMISMN statement as the first executable statement in the program and an ENDFIL statement as the last executable statement in the program.

5.1.11.2 Results file or output file

Dimensional measuring equipment DMIS output files are similar in syntax to the input files. Measurement results are passed in the form of actual feature and tolerance definitions that are similar in format to the input file (nominal) definition statements. In addition, certain command statements are passed to the output file to indicate machine parameter settings at the time of measurement, that are also similar in format to the input file command statements. The DMIS output file(s) always has a FILNAM statement as the first line and an ENDFIL statement as the last line.

5.1.12 Programming considerations

5.1.12.1 Program structure

Some general programming considerations should be kept in mind when generating an inspection program using the DMIS vocabulary. These considerations include: program declaration, program structure, entity definitions, branching, modality, and default settings.

5.1.12.2 Program declarations

The declarations in DMIS programs include external files, variables, and macro declarations. Details as given in clause 5.2.1 and sub clauses.

5.1.12.3 DMIS statements

The body of a DMIS program has specific structure, defined as follows.

5.1.12.3.1 Definition statements

All features, tolerances, and sensors are defined in the DMIS input program prior to their use in other statements.

5.1.12.3.2 Modal and non-modal statements

Statements in the DMIS vocabulary are either modal or non-modal refer to Table 13 — Modal statements.

Modal statements usually set some machine parameter or other condition that remains in effect until the statement is reissued. Non-modal statements are in effect only for a single execution of the statement. As a result, when branching in a program, caution should be exercised to ensure that modal settings are properly set both before and after the branch occurs. Branches are caused by the JUMPTO, IF, DO, SELECT, CALL, ITERAT, ERROR and RESUME statements.

5.1.12.4 Default settings

There are no default settings in the specification. As a result, users must be aware that initial DME settings or residual setting(s) from preceding operations may be in effect.

5.2 Execution and control

5.2.1 Declaration statements

The declarations in DMIS programs include external files, variables, and macro definitions.

5.2.1.1 External declaration statements

DMIS provides for the declaration of external files that are used within the local DMIS program unit or DMIS module (DMISMN, DMISMD). These files may contain external programs (that is, DMISMD), shell scripts, DME routines, and DMIS macros that may be invoked by the CALL/EXTERN statement. All external files must be declared within an XTERN…ENDXTN external declaration block. An XTERN statement starts the external declaration block. An ENDXTN statement ends the external declaration block. The external declaration block includes EXTFIL statements that declare a file pathname containing program definitions for the DMIS program unit.

Synopsis of an external declaration block is as follows:

```
XTERN
EXTFIL/var_1,'pathname'
ENDXTN
```

5.2.1.2 Variable declaration statements

All DMIS variables must be declared with the DECL statement. All DMIS variables must be declared before they are used. Typically, a variable declaration section is used to declare one or more variables of a specific data type for the DMIS program unit DMISMN, DMIS module DMISMD and macros.

Synopsis of a variable declaration section is as follows:

```
DECL/var_1 var_2 var_3
```

A previously declared and assigned variable can be substituted for any parameter (that is, numeric value, Boolean value, character string, vector value, or label name) of the proper data type in any DMIS statement. When such a variable appears in a statement, any DMIS output text shall contain the value at the time of execution.

```
DECL/CHAR,10,myclamp
myclamp=ASSIGN/'CLAMP12'
CI(CLAMP)=CLMPID/myclamp
```

5.2.2 Definition statements

5.2.2.1 Algorithmic definitions

The algorithmic definitions are used to define specific algorithms to particular inspections or features. The ALGDEF statement primarily supports video inspection devices with additional sensor capabilities as they relate to different vision algorithms and filters. The intent of the GEOALG statement is to designate a specific substitute feature data fitting algorithm (for example, least squares, minmax, minimum circumscribed, maximum inscribed) for calculating resultant features from the measured data.

The feature actual or "FA" that results from a measurement sequence always contains all individual point information. If point buffering was enabled with the PTBUFF statement when the feature was measured, all individual point information will be accessible. The effect of a substitute data fitting algorithm will never remove measured data but may cause the calculation of a feature using less than all of the measured data.

5.2.2.2 Boundary definitions

The BOUND statement assigns boundaries to features or tolerances. The feature(s) and/or tolerance definition(s) referenced in the BOUND statement must occur in the file prior to the BOUND statement. Also, the BOUND statement must occur prior to the measurement of a feature that is bounded. When a tolerance is bounded, the BOUND statement must occur prior to the EVAL or OUTPUT statements for the feature to which the tolerance is applied.

5.2.2.3 Geometry definitions

The geometry definitions are used to define specific geometry with CAD data information with the GEOM statement.

5.2.2.4 Macro definitions

DMIS provides for the definition of macro routines to perform repetitive functions without having to reprogram all of the statements. A macro is declared and defined in a MACRO...ENDMAC block. Once defined a macro is available for invoking by using a CALL statement. Typically, a macro definition section is used to define macros for use within the DMIS program unit or DMIS module (DMISMN, DMISMD). Synopsis of a macro definition section is as follows:

```
M(lname)=MACRO/argument_list

  external declarations

  data declarations

  DMIS statement sequences

ENDMAC
```

5.2.2.5 Manufacturing device definitions

Statements defining manufacturing devices and tools, MFGDEV and TOOLDF, are associated with a compensation statement, CUTCOM, to allow process adjustment based on inspection results. These statements provide the necessary data to close the loop for manufacturing cells requiring in-process verification.

5.2.2.6 Mating definitions

Mating definitions are used to define mating information between either features and geometry or between features and other features with the MATDEF statement.

5.2.2.7 Quality information system definitions

Quality Information Systems (QIS) definitions are provided to support quality information management. A QIS definition defines a QIS variable. QIS variables may be used with the OBTAIN, PROMPT and REPORT statements. QIS variables may be user-defined with the QISDEF statement or predefined using the statements listed in Table 5 — Predefined QIS variable types.

Example of user-defined QIS variable

```
Q(boss1)=QISDEF/'supervisor','Jill'
```

Example of predefined QIS variable

```
OP(oper_id)=OPERID/'Jack'
```

Table 5 — Predefined QIS variable types

Statement	Label Prefix	Meaning
CLMPID	CI	part holding clamp id
CLMPSN	CS	part holding clamp serial number
CUTCOM	CC	compensation or process adjustment
DMEID	DI	dimensional measuring device id
DMESWI	DS	DME software id
DMESWV	DV	DME software version
FIXTID	FI	part holding fixture id
FIXTSN	FS	part holding fixture serial number
LOTID	LI	lot id
MFGDEV	MD	manufacturing device
OPERID	OP	operator id

PARTID	PN	part id
PARTRV	PR	part revision level
PARTSN	PS	part serial number
PLANID	PL	inspection plan id
PREVOP	PV	previous operation
PROCID	PC	inspection procedure
TOOLDF	TL	tool used on a manufacturing device

QIS variables can not be referenced using just the variable name, the type identifyer and the variable name must be used.

Example of variable reference:

```
Q(boss1)=QISDEF/'supervisor','Jill'
```

Would be referenced as:

```
DECL/CHAR,32,msg,supe
supe=OBTAIN/Q(boss1),2
msg=ASSIGN/CONCAT('Supervisor: ',supe,' supervised.')
TEXT/OUTFIL,msg
```

5.2.2.8 Tolerance definitions

The tolerance definitions in DMIS are generic, in that there are no pointers to the feature to which the tolerance is applied. The association between the tolerance and the feature is managed with the EVAL and OUTPUT statements. The tolerance definition specified with a TOL statement need not occur in a DMIS file immediately following the feature definition, but the tolerance definition must occur before being used by either an EVAL or OUTPUT statement.

5.2.3 Program statement sequences

5.2.3.1 Selective processing

Selective processing is the capability to allow program flow control to be altered based on either the results of measurements, or the values of externally supplied parameters. This capability can be obtained by using the HLL extensions. This includes using the OBTAIN and ASSIGN statements along with the HLL conditional branching statements (for example, IF, SELECT).

5.2.3.2 Programs and modules

The body of an inspection program (that is, feature definitions, tolerance definitions, bound statements, and output statements) must be sequenced properly. In general, entities must be defined prior to their use. All DMIS input programs must begin with a DMISMN statement and all DMIS modules must begin with a DMISMD statement. Both must end with an ENDFIL statement. A DMISMN statement indicates to the receiving system that the program is the DMIS input main program. In a DMIS input main program, the ENDFIL statement indicates that the end of the main program has been reached. In a DMIS input module, the ENDFIL statement transfers program execution back to the calling program at the line following the CALL statement. DMIS input modules that are not part of the main DMIS program can be called. All modules will have a DMISMD statement as their first line of executable code. This allows for the separation of modules and main programs and accommodates a more flexible approach.

5.2.3.3 Evaluation

The EVAL and OUTPUT statements are commonly used immediately following a measurement sequence to evaluate features with tolerances. DME's with memory constraints may require feature actuals to be saved after measurement and later recalled for evaluation or output. When the SAVE statement is used for a feature and the PTBUFF/ON statement was used prior to the time of measurement, the feature actual and its data points are

saved. If the PTBUFF/OFF statement was used prior to the time of measurement, then only the feature actuals are saved.

For example, two features are required to verify parallelism and one of them might have been measured prior to the EVAL or OUTPUT statements. Since it is required that the feature actual(s) exists, because of possible memory constraints, the feature actual should be saved by the SAVE statement.

The use of the EVAL statement is optional in determining the calculation of a tolerance to a feature because the evaluation of a tolerance to a feature can also be performed with the OUTPUT statement. The EVAL and OUTPUT statements are commonly used immediately following a measurement sequence but can occur anywhere following the measurement sequence providing the feature actual was saved with the SAVE statement.

The EVAL statement can be used to generate a tolerance actual, that is, a TA(lname), from which data can then be obtained with the OBTAIN statement and used in a selective processing test instead of being sent to an output file with the OUTPUT statement. The EVAL and OUTPUT statements can support several tolerance labels for one to two feature labels. Two feature labels are required for relationship tolerances for example with the TOL/ANGLB, TOL/DISTB and TOL/SYM statements.

When executed, the OUTPUT or EVAL statements cause the evaluation of the feature actual(s), FA(lname), and the tolerance actual, TA(lname). Additionally, the OUTPUT statement causes data to be output. The OUTPUT statement may specify a report format, which is additional information to be sent to the output file. If this is the case, the REPORT statement format must have been defined prior to the OUTPUT statement. Evaluation means the results are available for further use.

Output statements specify output data content, format, and the device(s) to which the data will be sent. Output statements can also be used to send instructions to the receiving system.

5.2.3.4 Measurement

When executed, the CALIB, MEAS or RMEAS statements cause a feature to be measured. They must occur after the feature is defined, after the BOUND statement (if the feature is bounded) and prior to the EVAL and OUTPUT statements for that feature.

5.2.3.5 Miscellaneous

Miscellaneous statements provide for the communication of user-specific data for special applications including the DMIS, DMEHW, and DMESW statements.

5.2.3.6 Example program statement sequences

As an example of the program statement sequences described above, consider the following:

```
$$ group 1) ------------------------------------------------
R(REP_1)=REPORT/DATE
F(CYL_1)=FEAT/CYLNDR,INNER,CART,0,0,5,0,0,1,8
F(PLN_1)=FEAT/PLANE,CART,0,0,0,0,0,-1
F(PLN_2)=FEAT/PLANE,CART,0,0,10,0,0,1
F(PLN_3)=FEAT/PLANE,CART,0,0,10.25,0,0,1
T(PERPX)=TOL/PERP,.05,RFS,DAT(A),RFS
T(dia_1)=TOL/DIAM,-.002,+.002
$$ group 2) ------------------------------------------------
BOUND/F(CYL_1),F(PLN_1),F(PLN_2)
BOUND/T(PERPX),F(PLN_1),F(PLN_3)
$$ group 3) ------------------------------------------------
MEAS/CYLNDR,F(CYL_1),12
  executable statements
ENDMES
SAVE/FA(CYL_1)
```

```
$$ group 4) ------------------------------------------------
EVAL/FA(CYL_1),T(PERPX)
$$(OPTIONAL STATEMENT)
$$ group 5) ------------------------------------------------
OUTPUT/FA(CYL_1),TA(PERPX),TA(dia_1),R(REP_1)
```

The statements are split into five groups for the purpose of illustration.

In group 1, the feature definitions and tolerance definitions may be listed in any order, providing they precede:

a) any required BOUND statements,

b) EVAL statements, or

c) OUTPUT statements.

The BOUND statements for features and tolerance, group 2, when required, must occur after the referenced definitions.

Once the feature definitions and required BOUND statements are issued, the feature can be measured in group 3.

The optional EVAL statement can occur immediately after the measurement sequence or at the end of the file.

The OUTPUT statement, group 5, may be listed in any order with other DMIS statements, but must follow the MEAS sequence for the corresponding feature measurement, and must follow the required tolerance definition. The OUTPUT statement can occur at the end of the file.

Although the REPORT and TOL statements are shown in group 1, they can be defined anywhere in groups 1, 2, or 4, as long as they appear prior to the OUTPUT statement.

5.2.4 High Level Language (HLL)

5.2.4.1 Functionality

The primary benefits to DMIS for the HLL and related I/O functions are as follows:

— Provide structure to DMIS programs.

— Incorporate more standard data types.

— Permit variable arrays.

— Permit variable assignments for example, PI=ASSIGN/ 3.1416

— Permit arithmetic assignments for example, A=ASSIGN/PI * (R**2)

— Permit logical assignments for example, XX=ASSIGN/(DIAM .GT. 1.25)

— Incorporate numeric, character, and system intrinsic functions for example, COS,CONCAT,TIME

— Provide selective processing

— Allow nested IFs.

— Allow unconditional JUMPTOs

— Incorporate value branching for example, SELECT…CASE…ENDCAS…ENDSEL

— Incorporate DO looping

— Incorporate external calling for example, CALL/EXTERN

— Permit modular programming

— Permit inclusion of existing statements for example, INCLUD

— Expand Input / Output for example, DEVICE, OPEN, CLOSE, READ, and WRITE

5.2.4.2 Branching and conditionals

The DMIS vocabulary allows for conditional branching in an inspection program based on the results of a measured value, obtained value, variable, arithmetic expression or logical expression. This is performed with the logical IF…ELSE…ENDIF block, DO…ENDDO block, and SELECT…CASE…ENDCAS…ENDSEL block branching statements. Unconditional branching is performed with the JUMPTO, or CALL statements. Further, specialized conditional branching is achieved with the ERROR, RESUME and ITERAT statements.

5.2.4.2.1 IF conditional branching

The IF…ELSE…ENDIF block statements provide the capability to transfer control of the program based on a logical expression. A typical IF…ELSE…ENDIF block will consist of the following:

```
IF/(logical expression)

  executable statement(s)
ELSE

  executable statement(s)
ENDIF
```

If the logical expression evaluates to .TRUE., control of the program is passed to the statements following the IF statement and executed until an ELSE or ENDIF statement is encountered. If the logical expression evaluates to .FALSE. control of the program is passed to the statements following the next occurrence of an ELSE statement. If an ENDIF statement is encountered before an ELSE statement, the conditional is ended. ELSE statements are optional. Nested IF statements are allowed and must have a corresponding ENDIF statement.

5.2.4.2.2 SELECT conditional branching

The SELECT...ENDSEL block, which begins with a SELECT statement, provides the capability to execute statements according to the results of one or more equality tests.

A typical SELECT...ENDSEL block will consist of the following:

```
SELECT/arg
 CASE/arg_1

   executable statement(s)
 ENDCAS
 CASE/arg_2
   executable statement(s)
 ENDCAS
 CASE/arg_3

   executable statement(s)
 ENDCAS
 DFTCAS
```

```
   executable statement(s)

  ENDCAS
ENDSEL
```

The block of statements following the first CASE statement whose literal argument equals the value of the SELECT statement's argument will be executed until the ENDCAS statement is encountered and then program execution will continue at the first statement following the ENDSEL statement. If no CASE statement's literal argument equals the SELECT argument and if the optional DFTCAS is present, then the statements between the DFTCAS statement and the ENDCAS statement are executed. An ENDSEL statement designates the end of the SELECT...ENDSEL block.

Details of the operation of the SELECT...ENDSEL block are given in section 6.168. An example is given in A.29.

5.2.4.2.3 DO looping

The DO…ENDDO block statements provide the capability of repeating a sequence of executable statement(s). The looping is dependent on the initial and limit values and the optionally specified increment value. A typical DO…ENDDO block will consist of the following:

```
DO/index,initial,limit,increment

  executable statement(s)

ENDDO
```

Based on the DO loop index that has been previously been declared as an integer variable, a looping sequence is initiated. The index is initially set and continues looping at the optionally specified increment until the limit value has be reached. The increment variable is optional, and if not present, the increment variable is set to 1.

5.2.4.2.4 Unconditional JUMPTO

In DMIS, the JUMPTO statement is used to unconditionally transfer the execution of the program to the statements following a program line that contains only a label name enclosed in parentheses with no label type. A typical JUMPTO statement will consist of the following:

```
JUMPTO/(jumptarget1)
(jumptarget2)

  executable statement(s)

(jumptarget1)

  executable statement(s)
```

The JUMPTO statement is used to direct control to any valid location backward or forward in the same program unit. A program cannot branch into or out of a measurement or calibration block, this is only allowed in conditional or looping blocks. These program blocks are described in clause 5.1.10.2 and sub clauses.

5.2.4.3 Macros

DMIS provides for the definition of macro routines to perform repetitive functions without having to reprogram all of the statements. DMIS macro routines can be defined internally in the program unit or externally in a modular program unit.

5.2.4.3.1 Macro declaration and structure

A macro routine is declared and defined by a sequence of DMIS statements starting with MACRO and ending with ENDMAC. All DMIS statements except DMISMN, DMISMD, and ENDFIL may be used in a macro routine. A MACRO statement includes an optional comma-separated parameter list. Each parameter is a name that would be valid as a variable name or such a name enclosed in apostrophes. If a name is enclosed in apostrophes, it

represents a label name. These parameters may be used in the statements of the macro routine. A label name MACRO parameter can be used in a statement of the macro routine only to represent a label name. Nested macros, that is, a macro within a macro, are allowed..

5.2.4.3.2 Macro operation

A macro routine is invoked by using a CALL statement. External macros must be called with the CALL/EXTERN statement. A CALL statement that invokes a macro routine must have the same number of parameters as the MACRO statement being invoked.

When a macro routine is called, it is executed as follows. First, substitute the statements between MACRO and ENDMAC for the CALL statement. Second, substitute the text of each parameter of the CALL statement for the corresponding parameter of the MACRO statement wherever a parameter of the MACRO statement is used in any statement in the macro routine. Third, execute the statements normally. A parameter of a CALL statement corresponding to a label name in the MACRO it calls must be enclosed in parentheses. When the text of such a parameter is substituted in the statements of the macro routine, only the part inside the parentheses is used (the parentheses are not included).

The following example illustrates the structure and operation of a macro. In (Figure 2 — Hole pattern to be inspected with a macro routine), a hole pattern in a plate is depicted. It is desired to measure the four circles through the use of a macro routine. The macro will be called HOLPAT, and each circle will be measured by taking four point measurements. A macro to perform these measurements is as follows:

```
DMISMN/'my_program',05.1
FILNAM/'Some_Program',05.1
DECL/LOCAL,REAL,X_val,Y_val
DECL/LOCAL,CHAR,64,Ralf_1
M(HOLPAT)=MACRO/X1,Y1,Diam1,Phred_1
DECL/LOCAL,INTGR,I
DECL/LOCAL,REAL,Iv,Jv,Rad1,Theta
F(@Phred_1)=FEAT/CIRCLE,INNER,CART,X1,Y1,0,0,0,1,Diam1
Rad1=ASSIGN/0.5*Diam1
GOTO/X1,Y1,5
GOTO/X1,Y1,0
MEAS/CIRCLE,F(@Phred_1),4
   DO/I,0,3
      Theta=ASSIGN/I*90
       Iv=ASSIGN/COS(DTOR(Theta))
       Jv=ASSIGN/SIN(DTOR(Theta))
      PTMEAS/CART,(Rad1*Iv)+X1,(Rad1*Jv)+Y1,0,-Iv,-Jv,0
   ENDDO
ENDMES
GOTO/X1,Y1,0
GOTO/X1,Y1,5
OUTPUT/F(@Phred_1),4
ENDMAC
```

In this example, the macro called "HOLPAT" is defined with four parameters: X1, Y1, Diam1 and Phred_1.

The GOTO statements position the probe in the center of the circle. The circle is defined, and the MEAS statement is then issued to measure the circle. A DO loop is used to calculate the nominal PTMEAS points in CART format. Finally, the sensor is repositioned and feature data are output. The ENDMAC statement signifies the end of the macro routine.

To perform the measurement of all four holes, it is only necessary to call the macro routine four times.

```
Ralf_1=ASSIGN/'CIRL_2'
CALL/M(HOLPAT),10,14,5,Ralf_1
X_val=ASSIGN/10
Y_val=ASSIGN/48
```

```
Ralf_1=ASSIGN/'CIRL_1'
CALL/M(HOLPAT),X_val,Y_val,10,Ralf_1
Ralf_1=ASSIGN/'CIRL_3'
CALL/M(HOLPAT),40,48,20, Ralf_1
Ralf_1=ASSIGN/'CIRL_4'
CALL/M(HOLPAT),40,14,3, Ralf_1

ENDFIL
```

In the first CALL statement in the example above, X1 is replaced by 10, Y1 is replaced by 14, Diam1 is replaced by 5, and Phred_1 is replaced by Ralf_1.

In the second CALL statement in the example above, X1 is replaced by X_val, Y1 is replaced by Y_val, Diam1 is replaced by 10, and Phred_1 is replaced by Ralf_1.

The values of X_val and Y_val will be changed after execution of the macro to 10 and 48, respectively. Also after execution, the variables I, Iv, Jv, Rad1, and Theta, which are declared inside the macro will no longer be valid variables.

The last parameter on the first line of the HOLPAT macro could also be handled by replacing Phred_1 with a name in apostrophes as follows.

```
M(HOLPAT)=MACRO/X1,Y1,Diam1,'label_1'
```

In this case, the two lines in the first half of the example that use Phred_1 would be changed to the following

```
F(label_1)=FEAT/CIRCLE,INNER,CART,X1,Y1,0,0,0,1,Diam1
OUTPUT/F(label_1),4
```

and the last half of the example would be written by putting the text of the label in parentheses, as follows.

```
CALL/M(HOLPAT),10,14,5,(CIRL_2)
X_val=ASSIGN/10
Y_val=ASSIGN/48
CALL/M(HOLPAT),X_val,Y_val,10,(CIRL_1)
CALL/M(HOLPAT),40,48,20, (CIRL_3)
CALL/M(HOLPAT),40,14,3, (CIRL_4)
```

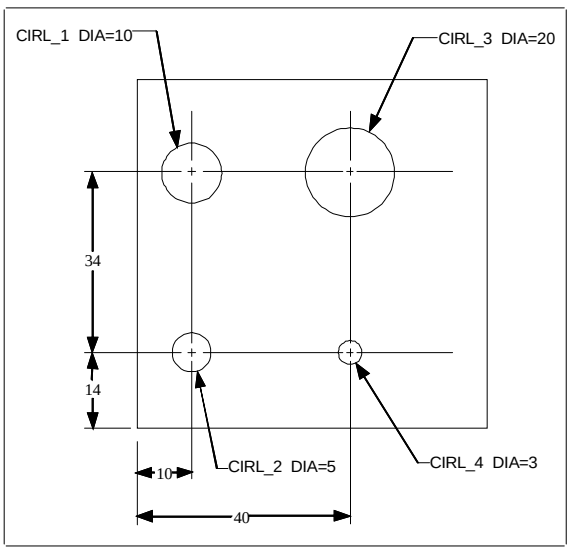


Figure 2 — Hole pattern to be inspected with a macro routine

5.2.4.4 Data types

5.2.4.4.1 Logical values

Logical values are either .TRUE. or .FALSE.. Logical data type is BOOL.

5.2.4.4.2 Numeric values

Numeric values are integral or floating point signed numbers whose range is determined by its data type. Numeric data types are INTGR, LONG, REAL, and DOUBLE.

5.2.4.4.3 Character values

Character values are a sequence of characters as defined in clause 5.1.1, Characters. Character data type is CHAR.

5.2.4.4.4 Vector values

A vector value in DMIS is an ordered triplet of double precision real numbers representing the Cartesian X, Y and Z components of a vector respectively. Vector data type is VECTOR. There are no literal vectors in DMIS. A vector value is created by first declaring a vector variable with the DECL statement using the VECTOR minor word; this variable is then filled with the desired values using either the VCART or VPOL intrinsic functions.

Vector values are not defined in any particular coordinate system and are without units. Therefore, they do update with coordinate system changes or changes in units.

A vector value may not be used as a direct substitute for DMIS parameters in a DMIS statement. For example, the following syntax is not allowed in DMIS:

```
DECL/VECTOR,pnt
pnt=ASSIGN/VCART(0,0,10)
GOTO/pnt
```

Instead, the X, Y and Z components of the vector value must be accessed with the VECX, VECY and VECZ intrinsic functions. The following is the corrected syntax:

```
DECL/VECTOR,pnt
pnt=ASSIGN/VCART(0,0,10)
GOTO/VECX(pnt),VECY(pnt),VECZ(pnt)
```

Refer to A.36.

5.2.4.5 Intrinsic functions

DMIS intrinsic functions consist of the function name, an open parenthesis, optional parameter list, and a closed parenthesis.

Table 6 — Intrinsic function words

ABS	INDX	RTOD	TAN
ACOS	INT	SCFEAT	TRIM
ASIN	LEN	SCSNS	UPC
ATAN	LN	SDATE	VAL
ATAN2	LOG	SDATETIME	VCART
BADGT	LONG	SELAPSETIME	VCROSS
BADPT	LWC	SENSNOTOUCH	VDOT

CHR	MN	SERROR	VECX
CONCAT	MOD	SHORT	VECY
COS	MX	SIGN	VECZ
DBLE	NINT	SILTCH	VMAG
DTOR	ORD	SIN	VMCS
ELEMNT	PTDATA	SMODE	VPCS
EOF	QTEMP	SQRT	VPOL
EOLN	RAND	STIME	VUNIT
EXIST	RL	STR	
EXP	RPT	SUBSTR	

5.3 Mathematics

5.3.1 Operators

Operators are used with operands (literals, variables and intrinsic functions) to create complex expressions which may be used to replace any parameters of the proper data type in any DMIS statement.

All operators except unary +, unary -, and (unary) .NOT. are binary operators. That is, they take one operand on the left and one on the right. The three unary operators take one operand on the right.

For all binary operators, except multiplication and division of vectors by numbers, both operands must be of the same type (numeric, logical, character, or vector).

Where the form of an expression does not make it clear how operations are to be grouped, (a - b + c), for example, precedence and associativity rules are applied. DMIS uses the precedence rules listed in (Table 7 — Precedence rules). Associativity rules vary by type of operator and are discussed in the subclauses below.

5.3.1.1 Numeric operators

Numeric operators yield numeric values.

Numeric expressions may use the following numeric operators:

a) + Addition and unary plus (i.e. keep the same sign as the operand)

b) - Subtraction and unary minus (i.e. reverse the sign of the operand)

c) * Multiplication

d) / Division

e) ** Exponentiation

EXAMPLES

```
B+C
A*B/2.0
(((B-C)*E)**A)+D
```

The operands of numeric operators must all be of numeric type. The subtypes of numeric (INTGR, LONG, REAL, and DOUBLE) may be used together in any combination. The numeric subtype yielded by a numeric operator is determined as described in clause 5.1.2.8.

In the case of division of integers in an expression containing only integers, the value of the result is truncated, not rounded, and no conversion to a real type is performed. For example, the value of the expression (5/3) is 1, and the value of the expression (-7/2) is -3.

Where an expression has numeric operators of the same precedence and grouping is not done with parentheses, the operators are left associative. For example, (a - b - c - d) is evaluated the same way as (((a - b) - c) - d).

5.3.1.2 Relational operators

Relational operators yield logical values.

Where numeric operands are used, numeric subtypes may be mixed.

Logical expressions use the following relational operators:

a) .EQ. Equal to
Operands may be numeric, character, logical, or vector.

b) .NE. Not equal to
Operands may be numeric, character, logical, or vector.

c) .LT. Less than
Operands may be numeric or character.

d) .LE. Less than or equal to
Operands may be numeric or character.

e) .GT. Greater than
Operands may be numeric or character.

f) .GE. Greater than or equal to
Operands may be numeric or character.

In the case of character operands, the following rules apply.

— The value to be compared for single characters is the numeric ASCII value.

— Two character strings are equal if and only if they have the same number of characters and at each position in the strings the characters are the same.

— Character string s1 is greater than string s2 if either:

  — at the first position where both strings have a character and the characters differ, the character in s1 is greater than the character in s2. For example, s1 is greater if s1='bit' and s2='bid'; also if s1='bit' and s2='bide'.

  — both strings have the same characters in the same positions up to the end of s2, but s1 is longer. For example, s1 is greater if s1='bite' and s2='bit'.

— Character string s2 is less than string s1 if and only if s1 is greater than s2.

— In the case of a CHAR variable, the number of characters given in the DECL statement is never used. Only the number of characters in the current contents of the variable is used.

Relational operators are not associative. It is an error to write an expression using relational operators in which associativity is required in order to interpret the expression. For example, the expression (a .LT. b .LT. c) is not allowed.

5.3.1.3 Logical operators

Logical operators yield logical values.

Logical expressions may use the following logical operators:

a) .AND. The .AND. operator shall yield .TRUE. if and only if both of its operands evaluate to .TRUE..

b) .OR. The .OR. operator shall yield .TRUE. if and only if one or both of its operands evaluate to .TRUE..

c) .NOT. The .NOT. operator shall yield .TRUE. if and only if its operand evaluates to .FALSE. . Likewise, it shall yield .FALSE. if and only if its operand evaluates to .TRUE..

The operands of logical operators must all be of logical type.

The .AND. and .OR. operators, which have the same precedence, are left associative. For example, (a .AND. b .OR. c) means the same as ((a .AND. b) .OR. c)

5.3.1.4 Vector operators

Vector expressions may use the following vector operators:

a) + Addition and unary plus (i.e. keep the same sign as the operand)

b) - Subtraction and unary minus (i.e. reverse the sign of the operand)

c) * Scalar multiplication (i.e. changing the magnitude of a vector)

d) / Scalar division (i.e. changing the magnitude of a vector)

EXAMPLES

```
DECL/VECTOR,A,B,C
A=ASSIGN/VCART(1,2,3)
B=ASSIGN/VCART(4,5,6)
C=ASSIGN/A+B-VCART(3,4,5)
$$ C is now 2,3,4
C=ASSIGN/A*2.0
$$ C is now 2,4,6
C=ASSIGN/(-A+(B*0.5))*2
$$ C is now 2,1,0
```

Except for scaling, the operands of vector operators must all be of vector type. For scaling, one operand must be of vector type and the other must be of numeric type.

Where an expression has vector operators of the same precedence and grouping is not done with parentheses, the operators are left associative. For example, (a - b - c - d) is evaluated the same way as (((a - b) - c) - d).

5.3.1.5 Precedence rules

DMIS expressions use the precedence rules for operators, which include nested parentheses, given in (Table 7 — Precedence rules).

Table 7 — Precedence rules

Operator	Description	Precedence
`Grouping`		

Operator	Description	Precedence
()	Group operations	9
Arithmetic Operators		
+	Positive value	7
-	Negative value	7
**	Exponentiation	6
*	Multiplication	5
/	Division	5
+	Addition	4
-	Subtraction	4
Intrinsic Functions		3
Relational Operators		
.EQ.	Equal to	2
.NE.	Not equal to	2
.LT.	Less than	2
.LE.	Less than or equal to	2
.GT.	Greater than	2
.GE.	Greater than or equal to	2
Logical Operators		
.AND.	Logical and	1
.OR.	Logical or	1
.NOT.	Logical negate	8
Precedence: 1 is lowest, 9 is highest.		
Like precedence is evaluated from left to right.		

5.3.2 Features

Features are geometric elements, which may or may not be on the workpiece or part. An example of a feature that is not on the part is a point in space. If the feature is on the part, it may be referred to in two ways: as a feature nominal or as a feature actual. Feature nominals must be defined in the program. Feature actuals are either measured or constructed by the DME during program execution. When a feature is constructed, at least one of the features used in its construction must be a measured or constructed feature. The result of a construction is a feature actual, 'FA(lname)'.

5.3.2.1 Feature definition

DMIS supports a set of feature definitions which are simple geometric elements. Additionally, other types of feature definitions can be used to define complex two dimensional curves and complex three dimensional surfaces.

5.3.2.1.1 Supported features

The following features are supported:

— Arc, with the FEAT/ARC (input format 1) and FEAT/ARC (input format 2) statements,

— Circle, with the FEAT/CIRCLE statement,

— Cone, with the FEAT/CONE statement,

— Conical radial segments with the FEAT/CONRADSEGMNT statement,

— Centered parallel line (cparln), with the FEAT/CPARLN statement,

— Cylinder, with the FEAT/CYLNDR statement,

— Cylindrical radial segments with the FEAT/CYLRADSEGMNT statement,

— Edge point, with the FEAT/EDGEPT statement,

— Ellipse, with the FEAT/ELLIPS statement,

— Elongated cylinder, with the FEAT/ELONGCYL statement,

— Generic curve (gcurve), with the FEAT/GCURVE statement,

— Generic feature, with the FEAT/GEOM statement,

— Generic surface (gsurf), with the FEAT/GSURF statement,

— Line, with the FEAT/LINE statement,

— Object, with the FEAT/OBJECT statement,

— Opposite symmetric planes (sympln), with the FEAT/SYMPLN statement,

— Parallel planes (parpln), with the FEAT/PARPLN statement,

— Pattern, with the FEAT/PATERN statement,

— Plane, with the FEAT/PLANE statement,

— Point, with the FEAT/POINT statement,

— Rectangle (right rectangular prism), with the FEAT/RCTNGL statement,

— Sphere, with the FEAT/SPHERE statement,

— Spherical radial segments with the FEAT/SPHRADSEGMNT statement,

— Surface-of-revolution, with the FEAT/REVSURF statement,

— Toroidal radial segments with the FEAT/TORRADSEGMNT statement, and

— Torus, with the FEAT/TORUS statement.

5.3.2.1.2 Complex features

Two feature types (FEAT/GCURVE and FEAT/GSURF) are provided to define complex curves and surfaces. These are required to support features which are not otherwise supported in the vocabulary. They can be used to define known complex geometry or they can also be used to digitize parts of unknown geometry. The nominal definitions for FEAT/GCURVE and FEAT/GSURF statements may simply assign a label to the group of points to be measured on the feature. Since, in this case, the feature has no nominal definition, the point data which was taken in measuring the feature is output as the actual.

A nominal definition for an unknown complex surface called SURF_1 might be as follows:

```
F(SURF_1)=FEAT/GSURF
```

The corresponding feature actual would be as follows when probe compensation is off: (set with the PRCOMP/OFF statement)

```
FA(SURF_1)=FEAT/GSURF,RAWDAT
/x,y,z
/x,y,z
/...
/...
ENDAT
```

The x,y,z data would represent raw data, that is, center of probe.

Or, the corresponding feature actual would be as follows when probe compensation is on: (set with the PRCOMP/ON statement)

```
FA(SURF_1)=FEAT/GSURF
/x,y,z
/x,y,z
/...
ENDAT
```

The x,y,z data would represent points on the surface. Some DMEs may also support output of the incoming i,j,k vectors from the PTMEAS statements. In this case, the vectors would be normal to the respective features defined with the FEAT/GCURVE, or FEAT/GSURF statements.

5.3.2.2 Reducible features

There are several feature types that are reducible to a simple geometric element of point, line, or plane type.

Table 8 — Reducible features

Feature	Reducible Feature		
	Point (x,y,z)	Line (x,y,z,i,j,k)	Plane (x,y,z,ni,nj,nk)
ARC	Yes – Center Point	No	No
CIRCLE	Yes – Center Point	No	No
CONE	Yes – Vertex	Yes – Axis	No
CONRADSEGMNT	Yes – Vertex	Yes – Axis	No
CPARLN	Yes – Center Point	Yes – Along Length vector	No
CYLNDER	No	Yes – Axis	No
CYLNDER-BOUND	Yes – Center Point	Yes – Axis	No
CYLRADSEGMNT	Yes – Center Point	Yes – Axis	No
EDGEPT	Yes	No	No

ELLIPS	Yes – Center Between Foci	Yes – from Focus1 to Focus2	No
ELONGCYL	No	No	Yes – Center Plane
GCURVE	No	No	No
GEOM	No	No	No
GSURF	No	No	No
LINE	No	Yes	No
OBJECT	No	No	No
PARPLN	No	No	Yes – Center Plane
PATERN	No	No	No
PLANE	No	No	Yes
POINT	Yes	No	No
RCTNGL	Yes – Center Point	No	No
REVSURF	No	Yes – Axis	No
SPHERE	Yes – Center Point	No	No
SPHRADSEGMNT	Yes – Center Point	No	No
SYMPLN	No	No	Yes – Center Plane
TORRADSEGMNT	Yes – Center Point	Yes	Yes
TORUS	Yes – Center Point	Yes	Yes
Reducible features are those features that can be unambiguously reduced.			
Derived Features are those features that can be derived from other feature's arguments.			

5.3.2.3 Feature reference

Features are referred to in two ways: as a feature nominal or as a feature actual.

5.3.2.3.1 Feature nominal

The feature nominal is the feature definition that comes from the CAD model or part drawing. It is the as-designed feature. The feature nominal definition is passed in the inspection program to the DME, and gives the nominal size, location, and orientation of the feature. A label name is assigned to the feature in the feature nominal definition. The label name is enclosed in parentheses and preceded by a label type "F", where the "F" indicates that this is the feature nominal definition.

A feature nominal definition for a circle called CIRCLE_1 might be as follows:

```
F(CIRCLE_1)=FEAT/CIRCLE,INNER,CART,10,10,5,0,0,1,8
```

Feature nominal coordinates shall retain their original values during any subsequent coordinate system transformations, with nominal values unchanged throughout the execution of the system. Therefore, a nominal definition is not directly associated with or expressed in any coordinate system; however, the nominal definition shall be utilized as a location relative to the origin of the current coordinate system at the time it is applied. Conversely, feature actual coordinates always reference the same absolute location within 3D space, but are transformed (when used) into the current coordinate system.

5.3.2.3.2 Feature actual

A feature actual FA(lname) is created by measuring or constructing a feature, or defining an FA(lname). The feature actual definition gives the measured size, location, and orientation of the feature. The actual definition is output by the DME into the output file, preceded by a label type FA(lname) to indicate a feature actual (refer to clause 5.1.9, Data output). The label name of the actual is the same as the label name of the corresponding nominal.

For example, CIRCLE_1 might be output as:

```
FA(CIRCLE_1)=FEAT/CIRCLE,INNER,CART,9.89,9.93,5,0,0,1,7.97
```

Here, we see that the actual center location deviated from the nominal definition in clause 5.3.2.3.1 by 0.11 in the X direction and by 0.07 in the Y direction, and the actual diameter deviated by 0.03.

5.3.2.4 Measured feature

DMEs collect point data and use these data to construct features. A circle, for example, may be measured by calculating the best-fit circle through five measured points. The points themselves are not features, but are only used to compute the circle. When data are used in this manner, the circle is referred to as a measured feature.

As shown in clause 5.3.2.3.1 and 5.3.2.3.2, feature orientation consists of an x,y,z point and an i,j,k unit vector. The use of the vector is described in the individual statement definition in section 6. Unless otherwise indicated, all normal vectors point away from the part surface.

5.3.2.5 Constructed feature

The DMIS vocabulary provides for the construction of features using other features. When a DME is instructed to construct a feature from other features, at least one of the features used in the construction must be a feature type FA(lname). The circle on which the holes in a bolt hole pattern lie, for example, can be constructed by measuring the bolt holes and then calculating the best-fit circle through their centers. In this case, the circle is called a constructed feature. The difference between a constructed feature and a measured feature is that the constructed feature is computed from other features, whereas the measured feature is computed from measured point data. Point data are usually lost after the feature actual definition is computed.

In the example of the bolt hole pattern, the DME can be instructed to compute the circle through the center-point of all measured holes, or it can be instructed to compute the center-point through one measured hole and the nominals of the other holes. It is not valid, however, to instruct the DME to compute the circle through only the nominals of the holes. In this case, the circle definition could have been computed without using any measurement results and therefore can be done without the DME. Since a construction must use at least one feature actual, a previously constructed feature, or a defined FA(lname), the result of a construction is a feature actual.

Construction statements cause the DME to construct a feature from previously defined features and at least one previously measured feature. The result of the construction process is a feature actual. The feature nominal must be defined prior to the construction of the feature actual. There are several formats for this statement, and there are no defaults. The specification for each format is given in the Statement Reference section (section 6.14 to section 6.28).

5.3.2.6 Feature data access

If a feature has been measured with point buffering enabled with the PTBUFF/ON statement, then its corresponding individual point nominal and measured data can be referenced by the following syntax:

```
F(lname)[n]
FA(lname)[n]
F(lname)[n,m]
FA(lname)[n,m]
```

Where:

n is the beginning point index of the feature's individual point data.

m is the ending point index of the feature's individual point data.

Note that if the measurement program is in automatic mode at the time of measurement, the individual point data would correspond to the nominal points as generated by the DME's internal algorithm and their subsequent actual measurements. If scanning is active, when operating in programmed mode and the PAMEAS statement is used to direct the scan path, any nominal data generated by the DME's internal algorithm will be on the defined path. Any nominal data generated by the DME's internal algorithm is based on scanning parameters in effect at the time of feature measurement. If the PTMEAS statement is used when operating in programmed mode, the individual point nominal data will correspond to the programmed PTMEAS statements and their subsequent actual measurements.

For example, the measured point data for the second PTMEAS statement of a circle CIR1 would be addressed as FA(CIR1)[2] or FA(CIR1)[2,2] and the nominal point data for the second, third, and fourth PTMEAS statement of a plane PL1 would be addressed as F(PL1)[2,4]. If the latter example of referencing PTMEAS points was used in an OUTPUT statement at a time when probe compensation is on (set with the PRCOMP/ON statement), then three separate lines of output would result.

```
F(PL1)[2]=FEAT/PLANE,PTDATA,CART,x2,y2,z2,i2,j2,k2
F(PL1)[3]=FEAT/PLANE,PTDATA,CART,x3,y3,z3,i3,j3,k3
F(PL1)[4]=FEAT/PLANE,PTDATA,CART,x4,y4,z4,i4,j4,k4
```

The actual point data for the second, third and fourth measured points on the same plane referenced with FA(PL1)[2,4] would produce the following three lines of output at a time when probe compensation is on (set with the PRCOMP/ON statement):

```
FA(PL1)[2]=FEAT/PLANE,PTDATA,CART,x2,y2,z2,prbdiam2,i2,j2,k2
FA(PL1)[3]=FEAT/PLANE,PTDATA,CART,x3,y3,z3,prbdiam3,i3,j3,k3
FA(PL1)[4]=FEAT/PLANE,PTDATA,CART,x4,y4,z4,prbdiam4,i4,j4,k4
```

In the above example, the individual x,y,z coordinates of each point would correspond to the resulting probe compensated coordinates of the measured feature's surface or edge. The prbdiam parameter for each point corresponds to the effective probe diameter used to measure each point of the plane's surface. The individual i,j,k vector direction for each point corresponds to the direction of the probe compensation from probe center to the x,y,z compensated coordinates. (Figure 3 — Feature data access) illustrates this for 3 points on a measured circle.

Any feature (including a point feature) measured while point buffering is enabled can have its individual point data referenced via the subscript format. Regardless of the setting of probe compensation at measurement time, both compensated and uncompensated point data can be accessed depending on the setting of probe compensation at the time of output. The form of the actual individual data output depends on the setting of probe compensation at the time of output.

The actual point data for the second, third and fourth measure points on the same plane referenced with FA(PL1)[2,4] would produce the following three lines of output at a time when probe compensation is off (set with the PRCOMP/OFF statement):

```
FA(PL1)[2]=FEAT/PLANE,PTDATA,RAWDAT,x2,y2,z2,prbdiam2,i2,j2,k2
FA(PL1)[3]=FEAT/PLANE,PTDATA,RAWDAT,x3,y3,z3,prbdiam3,i3,j3,k3
FA(PL1)[4]=FEAT/PLANE,PTDATA,RAWDAT,x4,y4,z4,prbdiam4,i4,j4,k4
```

In the above example, the individual x, y, z coordinates of each point would correspond to the resulting probe center coordinates when triggered by the measured feature's surface or edge. The prbdiam parameter for each point corresponds to the effective probe diameter used to measure each point of the plane's surface. The individual i, j, k vector direction for each point corresponds to the direction of probe compensation from probe center to the x, y, z compensated coordinates. If a feature is not understood by the system, then the RAWDAT format for each feature type should be utilized instead of the above formats.

The type of the individual point values for a feature nominal are derived at the time of measurement. If a point was specified in polar coordinates within a PTMEAS statement, it will be output in polar coordinates during point data output, otherwise if in Cartesian coordinates it will be output in Cartesian coordinates.

The use of the subscripted label format is limited to certain vocabulary statements which include: CONST (input format 1), CONST (input format 4), CONST (input format 13), DATDEF, OBTAIN, OUTPUT and XTRACT statements. Refer to the detailed format description (and notes) on each of those statements for additional information on the use of subscripted label formats.

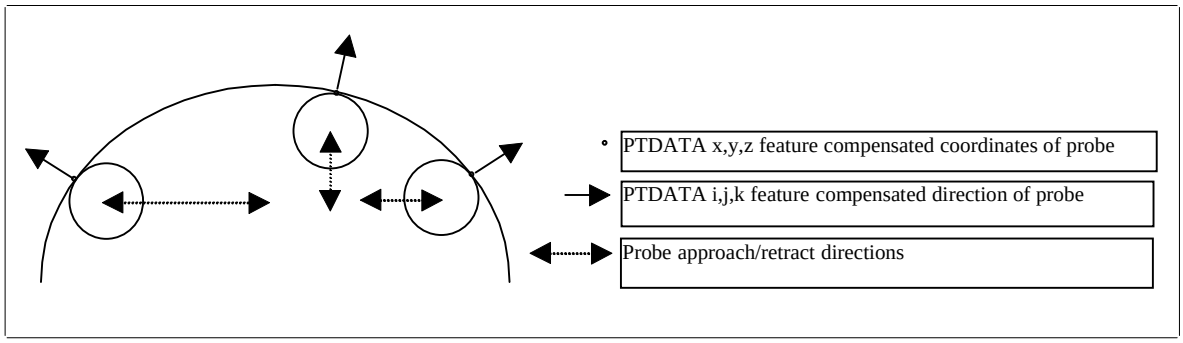


Figure 3 — Feature data access

5.3.2.7 Boundaries

Boundaries can be applied to both features and tolerances to create bounded features and tolerances.

The boundary definition is used to define boundaries on both features and tolerances. Unbounded features and tolerances are bounded with planes. Some tolerances, such as 3D position and total runout, require that features be bounded. A projected tolerance zone is an example of a bounded tolerance.

5.3.2.7.1 Features

Planes and cones are unbounded by definition. Lines and cylinders may be bounded or unbounded depending on their definitions. Unbounded features can be bounded with planes using the BOUND statement. The planes may either exist on the part or they may exist only in space.

Constructions of offset planes or planes through points constructed with MOVEPT can end up "in space" and yet they are valid feature actuals.

5.3.2.7.2 Tolerances

Tolerances may also be bounded using the BOUND statement. For example, this may be used when required to define a projected tolerance zone for perpendicularity. Tolerance boundaries may be used for other reasons, such as to bound profile tolerances. Bounded tolerances remain bounded for the duration of the program.

5.3.2.7.3 Usage

Boundary information may be used by either the DME, the system receiving results, or both. If a CAD system is to receive the DME output file, for example, feature boundary information is required to construct a model of the measured part. If the DME does not use the boundary information, it is passed to the output file without change or interpretation. If the DME does use boundary information, the actual boundaries will be passed to the output file when the feature data is output.

5.3.2.7.4 Boundary example

An example of a bounded feature and tolerance is shown in (Figure 4 — Boundaries applied to a feature and tolerance). This is a plate with a hole through it. The hole has a perpendicularity tolerance applied with a projected tolerance zone of 0.25. The hole is represented as a cylinder in the DMIS vocabulary. The cylinder is then bounded with PLANE_1 and PLANE_2. The perpendicularity tolerance is bounded with PLANE_2 and PLANE_3,

where PLANE_3 is 0.25 above the surface of the plate. Note that the direction vectors of the bounding planes point away from the surface of the part. The statements that describe this condition might read as follows:

```
MODE/AUTO
T(PERP_TOL)=TOL/PERP,.25,RFS,DAT(A)
F(PLANE_1)=FEAT/PLANE,CART,5,5,0,0,0,-1
F(PLANE_2)=FEAT/PLANE,CART,5,5,5,0,0,1
F(PLANE_3)=FEAT/PLANE,CART,5,5,5.25,0,0,1
MEAS/PLANE,F(PLANE_1),3
ENDMES
MEAS/PLANE,F(PLANE_2),3
ENDMES
CONST/PLANE,F(PLANE_3),OFFSET,FA(PLANE_2)
BOUND/F(CYL_1),F(PLANE_1),F(PLANE_2)
BOUND/T(PERP_TOL),F(PLANE_2),F(PLANE_3)
```

In the example, PLANE_1 and PLANE_2 exist on the part, while PLANE_3 does not. If the DME used the boundary information in computing the actual feature and tolerance, it would output the actuals for the planes and indicate them as actuals in the BOUND statement sent to the output file. Actual boundary plane definitions are also sent to the output file by execution of the BOUND statement, if the boundary information is utilized:

```
BOUND/FA(CYL_1),FA(PLANE_1),FA(PLANE_2)
FA(PLANE_1)=FEAT/PLANE,CART,5,5,0,0,0,-1
FA(PLANE_2)=FEAT/PLANE,CART,5,5,5,0,0,1
BOUND/TA(PERP_TOL),FA(PLANE_2),FA(PLANE_3)
FA(PLANE_2)=FEAT/PLANE,CART,5,5,5,0,0,1
FA(PLANE_3)=FEAT/PLANE,CART,5,5,5.25,0,0,1
```

If the DME did not use the boundary information, it would pass the BOUND statement to the output file as it was read. Nominal boundary plane definitions are also sent to the output file by execution of the BOUND statement, if the boundary information is not utilized:

```
BOUND/F(CYL_1),F(PLANE_1),F(PLANE_2)
F(PLANE_1)=FEAT/PLANE,CART,5,5,0,0,0,-1
F(PLANE_2)=FEAT/PLANE,CART,5,5,5,0,0,1
BOUND/T(PERP_TOL),F(PLANE_2),F(PLANE_3)
F(PLANE_2)=FEAT/PLANE,CART,5,5,5,0,0,1
F(PLANE_3)=FEAT/PLANE,CART,5,5,5.25,0,0,1
```

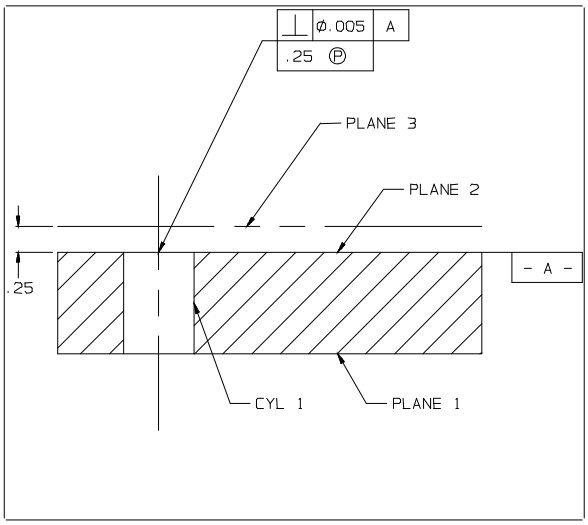


Figure 4 — Boundaries applied to a feature and tolerance

5.3.2.7.5 Bounding features

The BOUND statement must occur prior to the measurement of a feature that is bounded. When bounding a feature, the bounding plane's direction vector (that is, normal) designates the direction that is trimmed away. Furthermore, the bounding planes are applied per precedence order via the current BOUND and any subsequent BOUND statements. Refer to Figure 5 — Feature boundaries.

```
F(ConeSeg)=FEAT/CONE,INNER,CART,0,0,2,0,0,-1,30
F(Top)=FEAT/PLANE,CART,0,0,1,0,0,1
F(Bottom)=FEAT/PLANE,CART,0,0,0,0,0,-1
F(AnglePlane)=FEAT/PLANE,CART,1,0,0,0.949,0.0,0.316
BOUND/F(ConeSeg),F(Top),F(Bottom),F(AnglePlane)
```

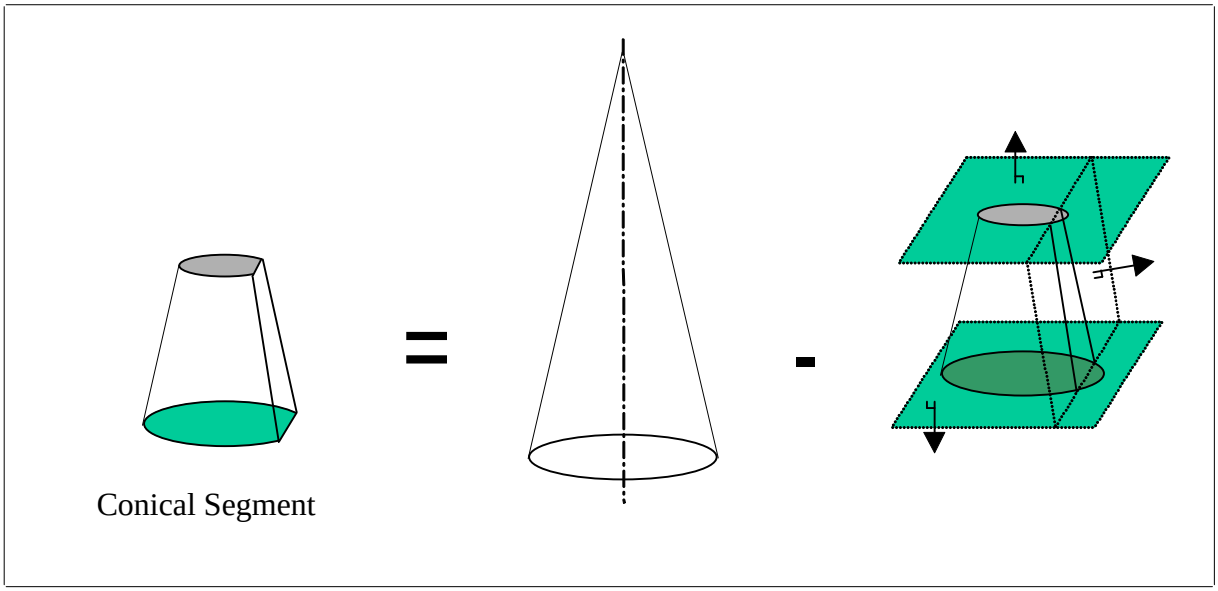


Figure 5 — Feature boundaries

5.3.2.7.6 Bounding tolerances

When a tolerance is bounded, the BOUND statement must occur prior to the EVAL or OUTPUT statements for the feature to which the tolerance is applied. When bounding a tolerance, the bounding plane's direction vector (that is, normal) indicates the included tolerance zone. Refer to Figure 6 — Tolerance boundaries.

```
T(PERP_TOL)=TOL/PERP,.005,DAT(A)
F(OffsetPlane)=FEAT/PLANE,CART,5,5,5.25,0,0,-1
F(Top)=FEAT/PLANE,CART,5,5,5,0,0,1
BOUND/T(PERP_TOL),F(Top),F(OffsetPlane)
```

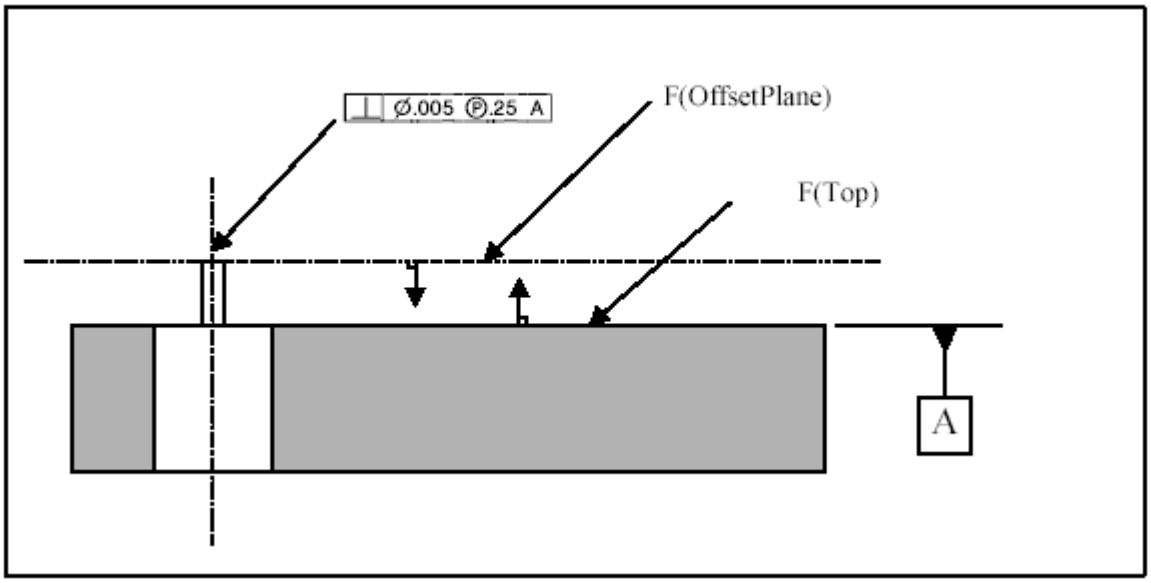


Figure 6 — Tolerance boundaries

5.3.3 Tolerances

5.3.3.1 Tolerance definitions

Tolerance definitions are used to describe generic tolerances. They also provide for labels to be assigned to each tolerance. These labels are used with the EVAL and OUTPUT statements to associate the tolerances to the features.

5.3.3.1.1 Supported tolerances

DMIS supports tolerances according to the [ASME Y14.5M - 1994 Standard for Dimensioning and Tolerancing]. The tolerances supported are:

— angle between features (direct)

— angle (direct)

— angle with respect to another feature (direct)

— angularity (orientation)

— circular runout (runout)

— circularity (form)

— composite position (location)

— composite line profile (profile)

— composite surface profile (profile)

— concentricity (location)

— cylindricity (form)

— diameter (direct)

— distance between features (direct)

— distance with respect to another feature (direct)

— flatness (form)

— parallelism (orientation)

— perpendicularity (orientation)

— position (location)

— profile of a line (profile)

— profile of a point (profile)

— profile of a surface (profile)

— radius (direct)

— straightness (form)

— symmetry (location)

— total runout (runout)

— width (direct)

In addition, tolerances have been included to support traditional practices, including bilateral positional (non-ASME locations) and user defined tolerances with the (TOL/USETOL) statement.

5.3.3.1.2 Referencing tolerances

Tolerances are referred to in two ways: as nominal tolerances and as actual tolerances. Nominal tolerances are those found in the CAD model or part drawing and are the "as-designed" tolerances. They are passed in the inspection program to the DME. Actual tolerances are the evaluated tolerances computed from measured or constructed features. A tolerance actual is created by an OUTPUT or an EVAL statement. An actual flatness, for example, is the width of the zone within which the measured feature has been computed to lie. Tolerances are assigned labels in the nominal definitions, and the actuals have the same label names as the corresponding nominals. Nominal labels are preceded by a "T" label type and actuals are preceded by a "TA" label type. The syntax of the actuals is very similar, but not identical, to the nominals. Refer to specific tolerance statements detailed in section 6 for additional information.

5.3.3.1.3 Tolerancing features

Tolerance definitions are generic, in that they do not have references to features. The relationship between features and tolerances is managed through the application of the OUTPUT and EVAL statements. The following example illustrates a sequence of statements to associate a feature CIRCLE_1 with a diameter tolerance DIAM_1 and a position tolerance POS_1:

```
F(CIRCLE_1)=FEAT/CIRCLE,INNER,CART,10,10,5,0,0,1,.250
T(DIAM_1)=TOL/DIAM,-.001,.0005
T(POS_1)=TOL/POS,2D,.005,MMC
MODE/AUTO,PROG,MAN
```

```
MEAS/CIRCLE,F(CIRCLE_1),4
ENDMES
OUTPUT/FA(CIRCLE_1),TA(DIAM_1),TA(POS_1)
```

In this way, the same tolerance label can be used for several different features, thereby minimizing the number of tolerance statements required in a program. Note, however, that bounded tolerances remain bounded for the duration of the program. Refer to section 5.3.2.7 for further explanation.

5.3.3.1.4 Material condition modifiers on features of size.

To consider the material condition modifier for positional tolerances applied to a feature of size (for example cylinders, or parallel planes) then a size tolerance (for example diameter or width) must be previously associated with the feature of size.

Given the code segment:

```
F(cyl)=FEAT/CYLNDR, ...
MEAS/CYLNDR,F(cyl),17
ENDMES
T(diam)=TOL/DIAM,-.010,.010
T(pos_m)=TOL/POS,3D,.014,MMC,DAT(K),DAT(C),DAT(P)
```

To consider the MMC material condition modifier for FA(cyl), then the size tolerance must be associated with FA(cyl) before the positional tolerance associated via one of three approaches:

```
EVAL/FA(cyl),T(diam)
OUTPUT/FA(cyl),TA(pos_m)
```

or:

```
OUTPUT/FA(cyl),TA(diam)
OUTPUT/FA(cyl),TA(pos_m)
```

or:

```
OUTPUT/FA(cyl),TA(diam),TA(pos_m)
```

5.3.3.2 Simultaneous requirements

The application of the OUTPUT and EVAL statements cause the evaluation of a tolerance condition relating two features (such as the distance between the features and the angle between the features) or the evaluation of one or more tolerances related to single feature (such as the size of the feature, the orientation of the feature, and the location of the feature). In the latter case features are evaluated individually.

It is often the case that it is required to evaluate location and profile tolerances of several features simultaneously. Where two or more features or patterns of features are located by basic dimensions related to common datum features referenced in the same order of precedence and at the same material condition they are often considered as a composite pattern with the geometric tolerances applied simultaneously.

The SIMREQT/ ENDSIMREQT block is used to control the simultaneous evaluation of tolerances. When a SIMREQT statement is issued all subsequent individual evaluations and output of tolerances initiated by EVAL and OUTPUT statements are suspended. When the ENDSIMREQT statement is encountered all tolerance evaluations specified in the SIMREQT/ ENDSIMREQT block (with EVAL and OUTPUT statements) are performed simultaneously and then the output specified in the SIMREQT/ ENDSIMREQT block (with OUTPUT statements) is performed.

If material condition modifiers are used on datum references or an incomplete datum reference frame is specified, i.e., one in which all degrees of freedom of rotation and translation are not removed, then datum reference frame mobility will be present. The movement of the datum reference frame within this mobility is recorded in the output of the SRA(lname) = ENDSIMREQT/TRMATX… statement. The ENDSIMREQT appears in the output file after all feature and tolerance output specified in the SIMREQT/ ENDSIMREQT block.

All profile and location tolerances evaluated or output in a SIMREQT/ ENDSIMREQT block must reference the same datum reference frame, i.e., they must reference the same datums in the same order with the same material condition modifiers.

An example of the use of the SIMREQT/ ENDSIMREQT block for the simultaneous evaluation of the true position of a hole pattern and surface profiles:

```
$$ Measure holes and surfaces to be toleranced
F(HOLE1)=FEAT/CYLNDR,INNER,CART,2.593,1.687,4.5,0,0,-1
MEAS/CYLNDR,F(HOLE1),6
...
ENDMES
F(HOLE2)=FEAT/CYLNDR,INNER,CART,2.593,3.312,4.5,0,0,-1
MEAS/CYLNDR,F(HOLE2),6
...
ENDMES
F(PLANE1)=FEAT/PLANE,CART,7.354,2.660,3.440,0.5,0,0.8660
MEAS/PLANE,F(PLANE1),14
...
ENDMES
F(PLANE2)=FEAT/PLANE,CART,9,2.5,2,1,0,0
MEAS/PLANE,F(PLANE2),6
...
ENDMES
$$ Measure datum features and assign datum labels
F(PLANEA)=FEAT/PLANE,CART,3,3,4.5,0,0,1
MEAS/PLANE,F(PLANEA),10
...
ENDMES
DATDEF/FA(PLANEA),DAT(A)
F(HOLEB)=FEAT/CYLNDR,INNER,CART,4,2.5,4.5,0,0,-1,2
MEAS/CYLNDR,F(HOLEB),16
...
ENDMES
DATDEF/FA(HOLEB),DAT(B)
$$ Define tolerances
T(DIAM1)=TOL/DIAM,-0.010,0.010
T(DIAM2)=TOL/DIAM,-0.015,0.010
T(DIAMB)=TOL/DIAM,-0.005,0.005
T(PROF1)=TOL/PROFS,-0.015,0.015,DAT(A),DAT(B),MMC
T(POS1)=TOL/POS,3D,0.010,MMC,DAT(A),DAT(B),MMC
$$ Evaluate the size of datum B
EVAL/FA(HOLEB),T(DIAMB)
$$ All the tolerances require a equivalent DAT sequence (e.g., DAT(A),DAT(B),MMC)
SR(1)=SIMREQT/FIRST
OUTPUT/FA(HOLE1),TA(DIAM1),TA(POS1)
OUTPUT/FA(HOLE2),TA(DIAM2),TA(POS1)
OUTPUT/FA(PLANE1),TA(PROF1)
OUTPUT/FA(PLANE2),TA(PROF1)
$$ This evaluates the simultaneous requirement to the tolerance's datum sequence
$$ including the appropriate datum material condition modifiers and creates a
$$ SRA(1) that shows the common PCS used to evaluate the simultaneous
$$ requirement that will be output.
ENDSIMREQT
```

The DMIS output file would contain output like:

```
SR(1)=SIMREQT/FIRST
FA(HOLE1)=FEAT/CYLNDR,INNER,CART,2.585,1.642,4.502,0,0,-1
TA(DIAM1)=TOL/DIAM,.003,INTOL
TA(POS1)=TOL/POS,3D,0.009,INTOL,MMC,DAT(A),DAT(B),MMC
```

```
FA(HOLE2)=FEAT/CYLNDR,INNER,CART,2.593,3.316,4.504,0,0,-1
TA(DIAM2)=TOL/DIAM,-0.007,INTOL
TA(POS1)=TOL/POS,3D,0.004,INTOL,MMC,DAT(A),DAT(B),MMC
FA(PLANE1)=FEAT/PLANE,CART,7.358,2.663,3.449,0.498,-.005,0.867
TA(PROF1)=TOL/PROFS,-0.008,0.004,INTOL,DAT(A),DAT(B),MMC
FA(PLANE2)=FEAT/PLANE,CART,9.003,2.498,2.009,.999,.0122,-.008
TA(PROF1)=TOL/PROFS,.001,0.009,INTOL,DAT(A),DAT(B),MMC
SRA(1)=ENDSIMREQT/TRMATX,.999,.002,-.003,.001,.987,.045,$
-.067,.004,.923,.002,-.004,-.003
```

5.3.4 Key Characteristics

The KEYCHAR statement provides a means to uniquely identify an association between nominal feature(s) and nominal tolerance(s) with an optional criticality designator. That is, one or two previously defined F(lname) with one or more defined T(lname) can be assigned with a MINOR, MAJOR, or CRITICAL criticality designator. The execution of an EVAL statement with a KC(lname) causes the evalution of the feature(s) specified in the KEYCHAR statement to the tolerances specified in the KEYCHAR statement. The execution of an output statement with a KC(lname) results in the key characteristic, feature nominal(s) and tolerance nominal(s) being passed to the output file. The execution of an output statement with a KCA(lname) results in the evaluation of the feature(s) specified in the KEYCHAR statement to the tolerances specified in the KEYCHAR statement, and the passing of the key characteristic, feature actual(s) and tolerance actual(s) to the output file.

5.3.5 Datums

The DATDEF statement provides a means to assign a datum label, that is, DAT(x), to one or more previously defined, measured, or constructed features. This datum label can then be used with the DATSET statement to define part coordinate systems. Datum labels are also used with the LOCATE, TRANS, ROTATE, and TOL statements. Compound datum labels, that is, DAT(x-x) are the result of feature constructions, and can be substituted where appropriate, for a DAT(x) in any statement format.

5.3.5.1 Datum targets

The DATTRGDEF statement provides a means to define a datum target and assign a datum target label, that is, DATTRG(x), to one or more previously defined, measured, or constructed features. This datum target label can then be used with the DATDEF statement.

```
F(PlaneA)=FEAT/PLANE,CART,5,5,5,0,0,1
F(PtA1)=FEAT/POINT,CART,1,1,5,0,0,1
F(PtA2)=FEAT/POINT,CART,1,9,5,0,0,1
F(PtA3)=FEAT/POINT,CART,9,1,5,0,0,1

DATTRGDEF/F(PtA1),DATTRG(A1)
DATTRGDEF/F(PtA2),DATTRG(A2)
DATTRGDEF/F(PtA3),DATTRG(A3)


DATDEF/DATTRG(A1),DATTRG(A2),DATTRG(A3),F(PlaneA),DAT(A)
```

5.3.6 Coordinate systems

DMIS coordinates are expressed in either a Cartesian coordinate system, refer to (Figure 7 — Right-handed Cartesian coordinate frame), or a 3-D polar coordinate system, refer to (Figure 8 — Right-handed polar coordinate frame)

Cartesian coordinates are rectilinear 3-D coordinates which are also called rectangular coordinates. The three axes of 3-D Cartesian coordinates conventionally denoted the x-axis, y-axis, and z-axis are chosen to be linear and mutually perpendicular. DMIS references 3-D Cartesian coordinates as x, y, and z. All vectors are expressed relative to the Cartesian coordinate system as i,j,k where 'i' is a distance along the x-axis from the origin, 'j' is a distance along the y-axis from the origin, 'k' is a distance along the z-axis from the origin. A vector i,j,k provides a normal direction for an associated point, the location of which is expressed in either Cartesian or polar coordinates.

3-D polar coordinates are also called cylindrical coordinates and are referred to as polar coordinates throughout this standard. DMIS references 3-D polar coordinates as r, a, and h. Where 'r' is a radial distance from the origin, 'a' is the azimuth or angle, and 'h' is a distance from the origin on the working plane, along the centerline axis of the cylindrical coordinate system.

The coordinate 'r' must be greater than or equal to zero. The coordinate 'a' must be a value between –360.00º and 360.00º if the angle is expressed in decimal degrees, between -2π and 2π if the angle is expressed in radians, or between -360:00:00.00 and 360:00:00.00 if the angle is expressed in degrees:minutes:seconds.

The coordinate 'a' is dependent on the working plane used in the definition of the 3-D polar coordinates. Transformation of 3-D polar coordinates to Cartesian coordinates or transformation of Cartesian coordinates to 3-D polar coordinates is dependent on the working plane used in the definition of the 3-D polar coordinates. The origin of the polar coordinate system is equal to the origin of the Cartesian coordinate system.

If the XY plane of the Cartesian coordinate frame is the working plane, coordinate 'a' is the angle from the x -axis toward the y-axis.

If the YZ plane of the Cartesian coordinate frame is the working plane, coordinate 'a' is the angle from the y -axis toward the z-axis.

If the ZX plane of the Cartesian coordinate frame is the working plane, coordinate 'a' is the angle from the z -axis toward the x-axis.

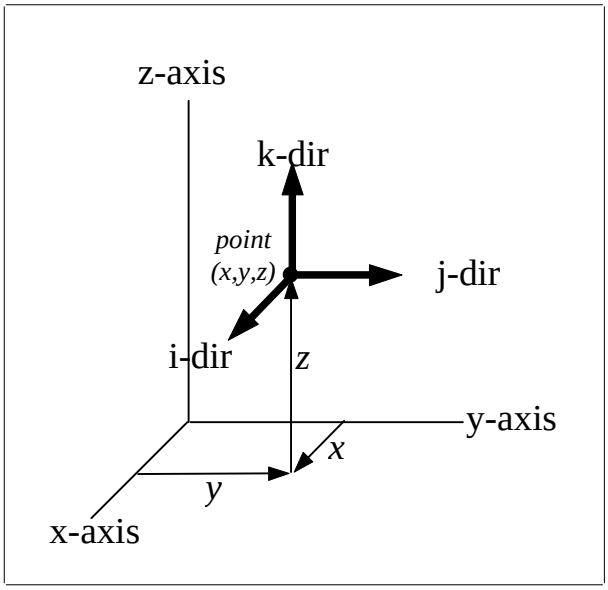


Figure 7 — Right-handed Cartesian coordinate frame

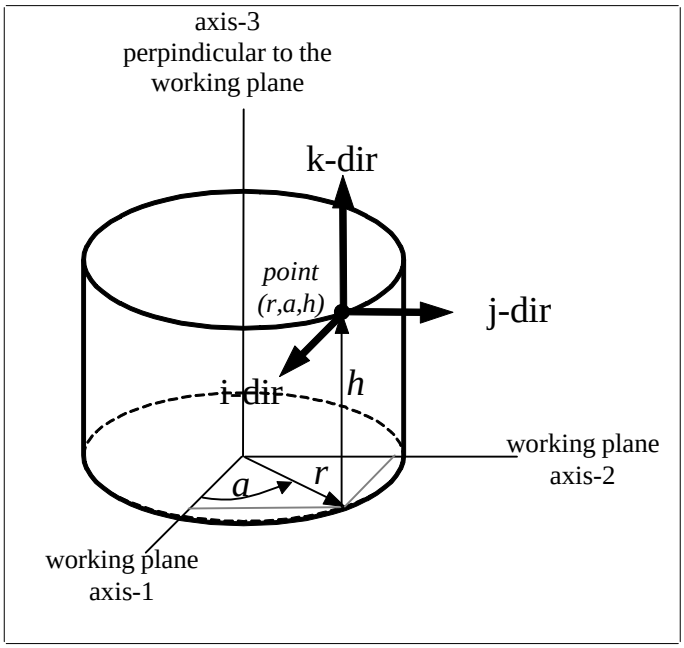


Figure 8 — Right-handed polar coordinate frame

5.3.6.1 Definition of the coordinate frame handedness

DMIS uses only the right-hand rule for coordinate frames. A right-handed coordinate frame is such that the z-positive direction vector multiplied in a cross product with the x-positive direction vector will yield the y-positive direction vector. Handedness is also defined with the right hand, where the index finger points in the positive x, the thumb points in the positive z, and the middle finger points to positive y.

(Figure 7 — Right-handed Cartesian coordinate frame) , shows a right-handed Cartesian coordinate frame.

(Figure 8 — Right-handed polar coordinate frame), shows a right-handed polar coordinate frame.

5.3.6.2 Definition of the handedness of rotation

DMIS uses only the right hand rule convention for angular relationships. A positive right-handed angular rotation in a right-handed coordinate frame is:

- In the x-y plane from positive x to positive y, where 0 degrees is along the positive x-axis.

- In the y-z plane from positive y to positive z, where 0 degrees is along the positive y-axis.

- In the z-x plane from positive z to positive x, where 0 degrees is along the positive z-axis.

A right hand rotation around an axis is defined with the thumb of the right hand pointing along the positive direction of the axis, and the fingers pointing in the direction of positive rotation.

All rotations are expressed as right-hand rotations in right-hand coordinate frames.

(Figure 9 — Right-handed (positive) rotation), illustrates a right-handed (positive) rotation.

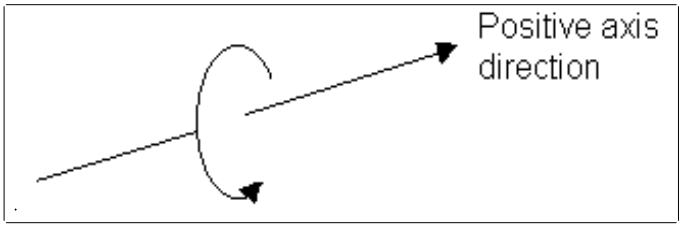


Figure 9 — Right-handed (positive) rotation

5.3.6.3 Coordinate frame transformation

All transforms will map a coordinate frame from a right-handed coordinate frame to a right-handed coordinate frame. All transforms must must preserve the lengths of vectors and the angles between them (i.e. every transformation must be a rotation, a translation, or a combination of the two with no scaling, skewing or mirroring).

5.3.6.4 Construction

Because the features of parts are defined and toleranced in reference to datums, a part coordinate system must be established reflecting those datums before the features can be measured. Also, since a DME has its own coordinate system, that is, a system of three mutually orthogonal axes of motion known as the machine coordinate system, the part coordinate system must be created within the machine coordinate system.

In DMIS a complete part coordinate system consists of three mutually orthogonal planes whose paired intersections represent the axes, and whose mutual intersection depicts the origin. DMEs establish part coordinate systems by measuring datum features or references and specifying them as the required elements of the part coordinate system.

In DMIS a part coordinate system consists of two separate and equivalent (but not identical) transformations: a D(lname), or defined coordinate system, and a DA(lname), or actual coordinate system. All feature nominals are referenced relative to a nominal coordinate system D(lname). All feature actuals are referenced relative to an actual coordinate system DA(lname). Both a defined coordinate system and an actual coordinate system are created simultaneously during execution of any of the datuming statements (for example DATSET, ROTATE, TRANS, LOCATE). The nominal coordinate system is created using only feature nominals, including the nominals for any feature actuals, specified in a datuming statement. The associated actual coordinate system is created from the list of features as specified, whether nominal or actual, in the same datuming statement.

Once a datum reference frame or part coordinate system has been defined, all feature locations and orientations are established with respect to it. New coordinate systems can be established through translation or rotation of the coordinate system, or by reissuing the DATSET statement. All coordinate systems are assigned labels. Translations and rotations of the part coordinate system can be defined by a specified amount or by an alignment with a feature. Refer to clause 5.3.2.3 and sub clauses for more information about the relationship of coordinate systems to features.

The part coordinate systems established by the DATSET, LOCATE, ROTATE, and TRANS statements can be saved with the SAVE statement and recalled for later use with the RECALL statement. They can also be rotated or translated to establish new coordinate systems. When creating a coordinate system the following rules apply:

— The primary datum is the first datum assigned.

— The secondary is always perpendicular to the primary datum. The DME will force perpendicularity.

— The tertiary is always perpendicular to both the primary and secondary datums. Here again, the DME will force perpendicularity.

5.3.6.5 Three step approach

The DATSET statement provides for the definition of the orientation, alignment, and origin, of the part coordinate system (refer to the DATSET statement). The TRANS statement can also define the origin of the part coordinate system, or translate it to establish a new one. The ROTATE statement can also define the alignment of the part coordinate system, or rotate it to establish a new one.

Together or in any combination, the DATSET, ROTATE, and TRANS statements may be combined to establish part coordinate systems in three steps:

a) ORIENTING the PRIMARY AXIS,

b) ALIGNING the SECONDARY AXIS,

c) ESTABLISHING the ORIGIN.

5.3.6.5.1 Orientation

The first step in creating a part coordinate system is to establish the direction of the primary axis. This direction might be, for example, the normal to a datum plane. Once determined, this axis becomes the primary axis establishing the orientation of the part coordinate system. For example, this axis can be the z-axis:

```
D(mcsx)=DATSET/MCS
F(plane_1)=FEAT/PLANE,CART,0,0,0,0,0,1
MEAS/PLANE,F(plane_1),3
PTMEAS/CART,0.5,1,0,0,0,1
PTMEAS/CART,0.5,10,0,0,0,1
PTMEAS/CART,10,.5,0,0,0,1
ENDMES
DATDEF/FA(plane_1),DAT(A)
D(orient_1)=DATSET/DAT(A),ZDIR,ZORIG
```

With the primary axis and plane established, the directions of the two remaining part coordinate system axes may be optionally determined, (in some cases, a primary axis and plane are sufficient). These remaining axes lie in the primary plane and are perpendicular to each other and the primary axis. Note that because of imperfections in manufacturing processes, the actual measured features used and assigned as datums with the DATDEF statement may not be perfectly orthogonal; however, the DME makes the necessary adjustments to ensure the construction of a part coordinate system that is perfectly orthogonal. Therefore, when these feature actuals are used to define the secondary and tertiary axes of the part coordinate system, they may not be coincident with the datum planes of the part coordinate system.

5.3.6.5.2 Alignment

The second step in creating a part coordinate system is to establish the direction of the secondary datum referenced on the part for the alignment of the secondary axis. This must be established with a minimum of two points and could be the X, Y, or Z axis as long as it does not conflict with the primary axis (must be orthogonal to the primary axis). This is commonly either the X or Y axis. For example:

```
D(orient_1)=DATSET/DAT(A),ZDIR,ZORIG
$$ Measure two circles, construct a line through
$$ their centers and align the Y axis to the
$$ vector of the line.  The alignment in this
$$ case will be established with the ROTATE
$$ and DATSET statements to illustrate their application.

F(circle_1)=FEAT/CIRCLE,INNER,CART,0,0,0,0,0,1,10
MEAS/CIRCLE,F(circle_1),4
PTMEAS/CART,-5.0,0.0,0,1,0,0
PTMEAS/CART,5.0,0.0,0,-1,0,0
PTMEAS/CART,0.0,-5.0,0,0,1,0
```

```
PTMEAS/CART,0.0,5.0,0,0,-1,0
ENDMES
F(circle_2)=FEAT/CIRCLE,INNER,CART,0,20,0,0,0,1,10
MEAS/CIRCLE,F(circle_2),4
PTMEAS/CART,-5.0,20.0,0,1,0,0
PTMEAS/CART,5.0,20.0,0,-1,0,0
PTMEAS/CART,0.0,25.0,0,0,-1,0
PTMEAS/CART,0.0,15.0,0,0,1,0
ENDMES
F(line_1)=FEAT/LINE,CART,UNBND,0,0,0,0,1,0,0,0,1
CONST/LINE,F(line_1),BF,FA(circle_1),FA(circle_2)
DATDEF/FA(line_1),DAT(B)
D(align_1)=ROTATE/ZAXIS,DAT(B),YDIR
```

Now that the secondary axis is established, the perpendicularity of the third or tertiary part coordinate system axis is locked in. The direction of the third or tertiary axis is defined by the right hand rule.

5.3.6.5.3 Origin

The third step in establishing a part coordinate system is to determine the zero starting point or origin . This final step is accomplished with the TRANS statement. For example:

```
$$ Set the X and Y axis origin to the center
$$ of FA(circle_1) and the Z axis origin on
$$  FA(plane_1)
DATDEF/FA(circle_1),DAT(C)
D(REF_SYS_1)=TRANS/XORIG,DAT(C),YORIG,DAT(C),ZORIG,DAT(A)
$$ Save the part coordinate system for later
$$ re-use.
SAVE/DA(REF_SYS_1)
```

5.3.6.6 Single step approach

An alternative to the previously described approach is to establish the part coordinate system using one DATSET statement. It is important to note, however, that when the following procedure is utilized, the DATSET statement must remain intact. The use of the tertiary portion of the DATSET statement is optional and is totally dependent upon the specific application.

For example:

```
D(mcs_dat)=DATSET/MCS
F(plane_1)=FEAT/PLANE,CART,0,0,0,0,0,1
MEAS/PLANE,F(plane_1),4
PTMEAS/CART,-5.0,20.0,0,0,0,1
PTMEAS/CART,5.0,20.0,0,0,0,1
PTMEAS/CART,0.0,25.0,0,0,0,1
PTMEAS/CART,0.0,15.0,0,0,0,1
ENDMES
DATDEF/FA(plane_1),DAT(A)
F(circle_1)=FEAT/CIRCLE,INNER,CART,0,0,0,0,0,1,10
MEAS/CIRCLE,F(circle_1),4
PTMEAS/CART,-5.0,0.0,0,1,0,0
PTMEAS/CART,5.0,0.0,0,-1,0,0
PTMEAS/CART,0.0,-5.0,0,0,1,0
PTMEAS/CART,0.0,5.0,0,0,-1,0
ENDMES
DATDEF/FA(circle_1),DAT(C)
F(circle_2)=FEAT/CIRCLE,INNER,CART,0,20,0,0,0,1,10
MEAS/CIRCLE,F(circle_2),4
PTMEAS/CART,-5.0,20.0,0,1,0,0
PTMEAS/CART,5.0,20.0,0,-1,0,0
```

```
PTMEAS/CART,0.0,25.0,0,0,-1,0
PTMEAS/CART,0.0,15.0,0,0,1,0
ENDMES
F(line_1)=FEAT/LINE,CART,UNBND,0,0,0,0,1,0,0,0,1
CONST/LINE,F(line_1),BF,FA(circle_1),FA(circle_2)
DATDEF/FA(line_1),DAT(B)
D(ref_sys_1)=DATSET/DAT(A),ZDIR,ZORIG,DAT(B),XDIR,YORIG,DAT(C),XORIG
```

In some instances, one might find it advantageous to control the probe radius compensation during the establishment of the part coordinate system. One common approach is to use the PRCOMP/ON statement prior to the part alignment sequence. If you desire more control, it is possible to use the PRCOMP/OFF statement and utilize the PRBRAD minor word in the TRANS statement to control the desired coordinate sy stem position. Let's review one possible scenario with two solutions.

Solution 1:

```
MODE/MAN
PRCOMP/OFF
WKPLAN/XYPLAN
$$ Establish Datum A
F(plane_A)=FEAT/PLANE,CART,0,0,0,0,0,1
MEAS/PLANE,F(plane_A),3
PTMEAS/CART,-5.0,20.0,0,0,0,1
PTMEAS/CART,5.0,20.0,0,0,0,1
PTMEAS/CART,0.0,25.0,0,0,0,1
ENDMES
DATDEF/FA(plane_A),DAT(A)
D(tran_Z)=TRANS/ZORIG,DAT(A)
$$ Establish Datum B
F(line_B)=FEAT/LINE,UNBND,CART,0,0,0,1,0,0,0,-1,0
MEAS/LINE,F(line_B),2
PTMEAS/CART,0.20,0.0,0,0,-1,0
PTMEAS/CART,15.0,0.0,0,0,-1,0
ENDMES
DATDEF/FA(line_B),DAT(B)
D(rotate_X)=ROTATE/ZAXIS,DAT(B),XDIR
$$ Establish Datum C
F(point_C)=FEAT/POINT,CART,0,0,0,-1,0,0
MEAS/POINT,F(point_C),1
PTMEAS/CART,0.0,0.0,0,1,0,0
ENDMES
DATDEF/FA(point_C),DAT(C)
D(tran_xy)=TRANS/XORIG,DAT(C),YORIG,DAT(B)
$$ Move Part Coordinate System to Part Surface
D(Set)=TRANS/XORIG,PRBRAD,YORIG,PRBRAD,ZORIG,PRBRAD
```

Solution 2:

```
MODE/MAN
PRCOMP/ON
WKPLAN/XYPLAN
$$ Establish Datum A
F(plane_A)=FEAT/PLANE,CART,0,0,0,0,0,1
MEAS/PLANE,F(plane_A),3
PTMEAS/CART,-5.0,20.0,0,0,0,1
PTMEAS/CART,5.0,20.0,0,0,0,1
PTMEAS/CART,0.0,25.0,0,0,0,1
ENDMES
DATDEF/FA(plane_A),DAT(A)
$$ Establish Datum B
F(line_B)=FEAT/LINE,UNBND,CART,0,0,0,1,0,0,0,-1,0
MEAS/LINE,F(line_B),2
```

```
PTMEAS/CART,0.20,0.0,0,0,-1,0
PTMEAS/CART,15.0,0.0,0,0,-1,0
ENDMES
DATDEF/FA(line_B,DAT(B)
$$ Establish Datum C
F(point_C)=FEAT/POINT,CART,0,0,0,-1,0,0
MEAS/POINT,F(point_C),1
PTMEAS/CART,0.0,0.0,0,-1,0,0
ENDMES
DATDEF/FA(point_C),DAT(C)
$$ Establish Part Coordinate System
D(origin)=DATSET/DAT(A),ZDIR,ZORIG,DAT(B),XDIR,YORIG,DAT(C),XORIG
```

5.3.6.7 Complex coordinate system construction

The process of constructing a part coordinate system becomes more complex when there are no longer perpendicular relationships between the features measured to establish datums and the datums themselves. For a complex alignment, the target measurement locations for at least one datum will be on a curved or canted surface. This means the orthogonal planes of the coordinate system cannot be determined independently. For example, the location of the primary datum targets on a curved surface, and therefore the primary datum direction, cannot be exactly determined until a complete coordinate system has been established.

Complex part coordinate systems for the purpose of dimensional inspection are often established using physical holding fixtures. These holding fixtures locate the part with respect to a fixture coordinate system defined with plate surfaces, bores or bosses, tooling balls or a combination of these. These simple features allow for simple coordinate system construction using the DATSET, ROTATE and TRANS statements.

The part is aligned to the holding fixture variously using a series of net pads, tooling balls, tangent bars, and fixed or spring-loaded circular or diamond pins. The effect of these fixture components is to remove all degrees of freedom of motion. The DME software can construct a complete coordinate system by removing these degrees of freedom in a manner similar to a holding fixture, allowing the physical holding fixture to be emulated.

5.3.6.7.1 Iterative coordinate system construction

One method of constructing a complex coordinate system is to measure the datum targets and establish a simple coordinate system from them using the DATSET, ROTATE and TRANS statements. Using this newly constructed coordinate system, the datum targets are re-measured, and a more refined coordinate system is constructed. This process is repeated until the desired convergence is achieved, that is, the coordinate system changes are less than a specified amount in a subsequent iteration.

In an iterative coordinate system construction, the ITERAT statement is used to control convergence testing and program branching. Two program statement labels are specified in the ITERAT statement parameter list. The first is the program location to which program execution is transferred in order to repeat the datum target measurement and simple coordinate system construction for the next iteration. The second is a program location to which program execution is transferred if the specified convergence is not achieved within the specified number of measurement iterations. Once convergence is achieved, the program execution is transferred to the statement following the ITERAT statement.

The following example shows the iterative construction of a part coordinate system on a part in which the three target points for the primary datum are on a curved surface.

```
MODE/MAN
DECL/GLOBAL,DOUBLE,dConverg
F(LOCATOR_1)=FEAT/POINT,CART,4235.0,856.0854,1145.0,0.0024,0.9999,-0.0114
F(LOCATOR_2)=FEAT/POINT,CART,4147.0,848.6901,1187.0,0.0,1.0,0.0
F(LOCATOR_3)=FEAT/POINT,CART,4292.0,858.1987,1340.0,-0.0022,0.9999,0.0115
F(DAT_A)=FEAT/PLANE,CART,4235.0,856.0854,1145.0,0.0,1.0,0.0
F(SLOT_B)=FEAT/CPARLN,INNER,ROUND,CART,4160.0,860.6941,1156.0$
,0.0,1.0,0.0,0.0,0.0,1.0,16.0,13.0
F(CIR_C)=FEAT/CIRCLE,INNER,CART,4265.0049,863.6899,130.0,0.0,1.0,0.0,13.0
```

```
F(DAT_B)=FEAT/LINE,UNBND,CART,4265.0049,863.6899,130.0,0.0,0.0,1.0,0.0,1.0,0.0
$$
(STARTING_POINT)
$$
MEAS/POINT,F(LOCATOR_1),1
ENDMES
MEAS/POINT,F(LOCATOR_2),1
ENDMES
MEAS/POINT,F(LOCATOR_3),1
ENDMES
MEAS/CIRCLE,F(CIR_C),4
ENDMES
MEAS/CPARLN,F(SLOT_B),5
ENDMES
CONST/PLANE,F(DAT_A),OFFSET,FA(LOCATOR_1),FA(LOCATOR_2),FA(LOCATOR_3)
DATDEF/FA(DAT_A),DAT(A)
CONST/LINE,F(DAT_B),OFFSET,FA(CIR_C),FA(SLOT_B)
DATDEF/FA(DAT_B),DAT(B)
DATDEF/FA(CIR_C),DAT(C)
D(orien_1)=DATSET/DAT(A),YDIR,YORIG
D(orien_2)=ROTATE/YAXIS,DAT(B),ZDIR
D(orien_3)=TRANS/XORIG,DAT(C),ZORIG,DAT(C)
D(orien_4)=ROTATE/XAXIS,1.97
D(REF_SYS_1)=TRANS/XORIG,-4263.234,YORIG,863.23,ZORIG,130.0
SAVE/DA(REF_SYS_1)
MODE/AUTO,MAN
dConverg=ITERAT/(STARTING_POINT),(FAILED),0.05,ABSL,5,YAXIS,FA(LOCATOR_1),$
FA(LOCATOR_2),FA(LOCATOR_3),XAXIS,FA(CIR_C),FA(SLOT_B),ZAXIS,FA(CIR_C)
  executable statements
JUMPTO/(EOP)
$$
(FAILED)
TEXT/OPER,'Convergence failed!'
(EOP)
ENDFIL
```

The datum target feature nominals are defined in the desired final part coordinate system. The datum target features are first measured manually. Datum features are constructed. A complete part coordinate system is constructed from the datum features using the three-step DATSET, ROTATE and TRANS statement method. The coordinate system orientation and origin is then moved to match the coordinate system in which the datum target feature nominals were defined. The ITERAT statement is then used to test for convergence. The datum target features are then re-measured in automatic mode and the datum feature construction and coordinate system construction process is repeated until the convergence of 0.05 is achieved at which time the executable statements comprising the main body of the program are executed. If the 5 iteration limit is exceeded program execution branches causing the message "Convergence failed!" to be sent to the operator.

5.3.6.7.2 Matching feature actuals and feature nominals

The DATSET, ROTATE and TRANS statements are used to construct a coordinate system with the measured datum features at the origin and with the measured datum feature axes pointing along the X, Y, or Z directions. Often the datum target feature nominals are not near the origin or parallel to a major axis. The establishment of the desired coordinate system then requires subsequent rotations and translations by nominal amounts.

The LOCATE statement allows for the construction of a complete coordinate system by matching measured feature actuals to defined feature nominals. The coordinate system thus created matches the coordinate system in which the datum target feature nominals were defined. The use of the LOCATE statement removes the necessity of using the DATSET, ROTATE and TRANS statements.

Because the LOCATE statement has feature location, direction and type information available, it can often better estimate the final converged coordinate system, resulting in significantly fewer measurement iterations than the DATSET, ROTATE and TRANS statement methods.

The LOCATE statement uses a list of feature actuals, matings, datums or a combination of these to construct a complete coordinate system. There must be a sufficient number of features, matings, or datums specified to remove all degrees of freedom specified in the LOCATE statement. The use of matings with the LOCATE statement is described in 5.3.6.7.3 and 5.3.6.7.5, the use of datums is described in 5.3.6.7.4, and over-constrained coordinate systems are described in 5.3.6.7.5.

What follows is an example of the same iterative alignment used in 5.3.6.7.1 that employs the LOCATE statement in place of the DATSET, ROTATE and TRANS statements:

```
MODE/MAN
DECL/GLOBAL,DOUBLE,dConverg
F(LOCATOR_1)=FEAT/POINT,CART,4235.0,856.0854,1145.0,0.0024,0.9999,-0.0114
F(LOCATOR_2)=FEAT/POINT,CART,4147.0,848.6901,1187.0,0.0,1.0,0.0
F(LOCATOR_3)=FEAT/POINT,CART,4292.0,858.1987,1340.0,-0.0022,0.9999,0.0115
F(SLOT_B)=FEAT/CPARLN,INNER,ROUND,CART,4160.0,860.6941,1156.0$
,0.0,1.0,0.0,0.0,0.0,1.0,16.0,13.0
F(CIR_C)=FEAT/CIRCLE,INNER,CART,4265.0049,863.6899,130.0,0.0,1.0,0.0,13.0
$$
(STARTING_POINT)
$$
MEAS/POINT,F(LOCATOR_1),1
ENDMES
MEAS/POINT,F(LOCATOR_2),1
ENDMES
MEAS/POINT,F(LOCATOR_3),1
ENDMES
MEAS/CIRCLE,F(CIR_C),4
ENDMES
MEAS/CPARLN,F(SLOT_B),5
ENDMES
D(REF_SYS_1)=LOCATE/XYZDIR,XYZAXI,FA(LOCATOR_1),FA(LOCATOR_2)$
,FA(LOCATOR_3),FA(CIR_C),FA(SLOT_B)
SAVE/DA(REF_SYS_1)
$$
MODE/AUTO,MAN
dConverg=ITERAT/(STARTING_POINT),(FAILED),0.05,ABSL,5,YAXIS,FA(LOCATOR_1),$
FA(LOCATOR_2),FA(LOCATOR_3),XAXIS,FA(CIR_C),FA(SLOT_B),ZAXIS,FA(CIR_C)
  executable statements
$$
JUMPTO/(EOP)
$$
(FAILED)
TEXT/OPER,'Convergence failed!'
(EOP)
ENDFIL
```

Again, the datum target features are measured manually. The LOCATE statement is used to find the coordinate system which matches the measured feature actuals to the feature nominal definitions. The ITERAT statement is used to re-measure the datum target features until the convergence of 0.05 is achieved. The resulting coordinate system will be the same (within the convergence factor of 0.05) as the one in the example in section 5.3.6.7.1, but it is achieved with less program code and possibly in fewer measurement iterations.

5.3.6.7.3 Holding fixture emulation

When a part is placed in a physical holding fixture, extended features on the part mate with the various components of the fixture. A surface may contact a tooling ball at one point, the lowest point on another surface will mate with a net pad, and a part edge may mate with a cylindrical pin acting as a trim stop. In all these cases, features on the part are mating with features on the fixture that are of a different type: a plane with a sphere, a plane with a point, and a line with a cylinder.

When feature actuals are specified in the LOCATE statement, they are matched to their corresponding feature nominals. In order to fully emulate the action of a holding fixture it is necessary to be able to specify matings between the measured feature actuals and feature nominal definitions representing holding fixture components. By fully emulating a holding fixture, measurement iterations can be eliminated. Extended mating features on a part can be measured once as planes, lines, generic surfaces or generic curves. These extended features can then be mated to nominally defined fixture components without re-measurement.

In the following example datum targets on a car door are measured as planes and lines. These measured features are then mated to nominally defined fixture components representing spheres and cylinders. A complete coordinate system is then constructed without measurement iteration emulating a fixture coordinate system using three spherical tooling balls and three cylindrical trim stops. The program assumes that an approximate coordinate system exists before measurement.

```
$$ measure datum targets as extended features.
$$
$$ Refer to (Figure 10 — Datum targets on a car door), for an example of datum
targets on a car door.
$$
MODE/PROG,MAN
F(A1)=FEAT/PLANE,CART,3415.730,920.416,532.578,0.0029,0.7728,-0.6347
MEAS/PLANE,F(A1),3
PTMEAS/CART,3420.065,918.138,529.767,0.0029,0.7728,-0.6347
PTMEAS/CART,3415.069,923.866,536.884,0.0030,0.7867,-0.6173
PTMEAS/CART,3412.057,919.244,531.083,0.0029,0.7756,-0.6312
ENDMES
F(A2)=FEAT/PLANE,CART,2774.553,917.095,528.327,-0.0025,0.7594,-0.6506
MEAS/PLANE,F(A2),3
PTMEAS/CART,2781.584,913.730,524.207,-0.0025,0.7594,-0.6506
PTMEAS/CART,2773.890,923.323,535.927,-0.0026,0.7852,-0.6193
PTMEAS/CART,2768.185,914.231,524.846,-0.0026,0.7610,-0.6487
ENDMES
F(A3)=FEAT/PLANE,CART,3145.283,697.323,1791.108,0.0043,0.8990,0.4379
MEAS/PLANE,F(A3),3
PTMEAS/CART,3164.413,700.855,1783.690,0.0043,0.8990,0.4379
PTMEAS/CART,3148.853,690.620,1804.849,0.0039,0.8990,0.4379
PTMEAS/CART,3122.584,700.494,1784.784,0.0032,0.8991,0.4378
ENDMES
F(B1)=FEAT/LINE,UNBND,CART,3216.241,886.203,1333.322,$
  -1.0000,-0.0001,0.0000,0.0002,-0.2106,0.9776
MEAS/LINE,F(B1),3
PTMEAS/CART,3216.241,886.203,1333.322,0.0002,-0.2106,0.9776
PTMEAS/CART,3205.588,886.203,1333.320,-0.0005,-0.2105,0.9776
PTMEAS/CART,3194.935,886.201,1333.322,-0.0004,-0.2105,0.9776
ENDMES
F(B2)=FEAT/LINE,UNBND,CART,2987.715,886.120,1332.454,$
  -1.0000,-0.0026,0.0000,0.0009,-0.2109,0.9775
MEAS/LINE,F(B2),3
PTMEAS/CART,2987.715,886.120,1332.454,0.0009,-0.2109,0.9775
PTMEAS/CART,2972.953,886.083,1332.454,0.0009,-0.2111,0.9775
PTMEAS/CART,2957.052,886.040,1332.454,0.0002,-0.2112,0.9774
ENDMES
F(C1)=FEAT/LINE,UNBND,CART,3446.980,982.042,1019.170,$
```

```
  0.0213,-0.0547,0.9983,0.9997,-0.0030,-0.0260
MEAS/LINE,F(C1),3
PTMEAS/CART,3446.980,982.042,1019.170,0.9997,-0.0030,-0.0260
PTMEAS/CART,3447.299,981.398,1032.481,0.9998,-0.0029,-0.0215
PTMEAS/CART,3447.560,980.556,1046.289,0.9999,-0.0031,-0.0141
ENDMES
$$ nominal definitions of fixture components
F(TB_A1)=FEAT/SPHERE,OUTER,CART,3415.581,925.569,530.118,6.350
F(TB_A2)=FEAT/SPHERE,OUTER,CART,2774.372,923.027,526.796,6.350
F(TB_A3)=FEAT/SPHERE,OUTER,CART,3146.197,700.646,1796.978,6.350
F(TS_B1)=FEAT/CYLNDR,OUTER,CART,3205.585,885.032,1338.755,0.0024$
,-0.9775,-0.2111,6.350
F(TS_B2)=FEAT/CYLNDR,OUTER,CART,2972.958,884.909,1337.889,0.0024$
,-0.9775,-0.2111,6.350
F(TS_C1)=FEAT/CYLNDR,OUTER,CART,3452.857,981.382,1032.361,0.0042$
,0.9981,0.0608,6.350
$$ mate measured datum targets with nominal fixture components
MA(MAT_A1)=MATDEF/F(TB_A1),FA(A1),PT2PL,BF,0.000,0.000
MA(MAT_A2)=MATDEF/F(TB_A2),FA(A2),PT2PL,BF,0.000,0.000
MA(MAT_A3)=MATDEF/F(TB_A3),FA(A3),PT2PL,BF,0.000,0.000
MA(MAT_B1)=MATDEF/F(TS_B1),FA(B1),LN2LN,BF,0.000,0.000
MA(MAT_B2)=MATDEF/F(TS_B2),FA(B2),LN2LN,BF,0.000,0.000
MA(MAT_C1)=MATDEF/F(TS_C1),FA(C1),LN2LN,BF,0.000,0.000
$$ construct the part coordinate system
D(PART)=LOCATE/MA(MAT_A1),MA(MAT_A2),MA(MAT_A3),MA(MAT_B1),MA(MAT_B2),MA(MAT_C1)
```

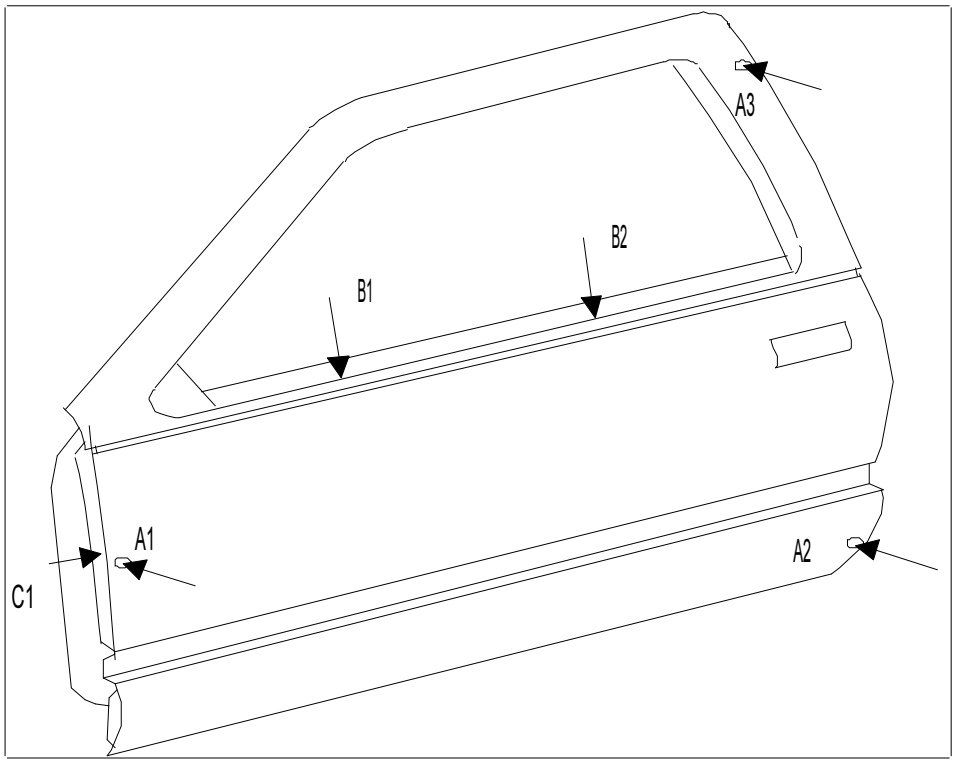


Figure 10 — Datum targets on a car door

5.3.6.7.4 Datums

Datums as defined with the DATDEF statement can be used in the LOCATE statement. A datum can remove positional degrees of freedom depending on the datum feature type. For example, a plane can remove two

rotational degrees of freedom and one translational degree of freedom, a cylinder can remove two rotational degrees of freedom and two translational degrees of freedom, and a measured line can remove one rotational degree of freedom and one translational degree of freedom. A datum removes all possible remaining degrees of freedom. The datums take precedence in the order in which they appear in the LOCATE statement in the same manner as datums take precedence in a feature control frame.

The example in (Figure 11 — Plate with a hole and slot), shows a coordinate system construction for a simple plate with a hole and slot.

```
F(DATUM_A)=FEAT/PLANE,CART,2.5000,2.5000,0.0000,0.0000,0.0000,1.0000
MEAS/PLANE,F(DATUM_A),3
PTMEAS/CART,5.0000,0.0000,0.0000,0.0000,0.0000,1.0000
PTMEAS/CART,0.0000,5.0000,0.0000,0.0000,0.0000,1.0000
PTMEAS/CART,5.0000,5.0000,0.0000,0.0000,0.0000,1.0000
ENDMES
DAT(A)=DATDEF/FA(DATUM_A)
F(DATUM_B)=FEAT/CIRCLE,INNER,CART,2.0000,2.0000,0.0000,0.0000,0.0000,1.0000,0.6468
MEAS/CIRCLE,F(DATUM_B),4
PTMEAS/CART,1.6766,2.0000,0.0000,0.0000,0.0000,1.0000
PTMEAS/CART,2.0000,2.3234,0.0000,0.0000,0.0000,1.0000
PTMEAS/CART,2.3234,2.0000,0.0000,0.0000,0.0000,1.0000
PTMEAS/CART,2.0000,1.6766,0.0000,0.0000,0.0000,1.0000
ENDMES
DAT(B)=DATDEF/FA(DATUM_B)
F(DATUM_C)=FEAT/CPARLN,INNER,ROUND,CART,2.0000,7.5000,0.0000$
,0.0000,1.0000,0.0000,0.0000,0.0000,1.0000,1.5000,0.5000
MEAS/CPARLN,F(DATUM_C),5
PTMEAS/CART,6.5000,2.2500,0.0000,-1.0000,0.0000,0.0000
PTMEAS/CART,8.5000,2.2500,0.0000,-1.0000,0.0000,0.0000
PTMEAS/CART,7.5000,1.7500,0.0000,1.0000,0.0000,0.0000
PTMEAS/CART,6.7500,2.0000,0.0000,0.0000,1.0000,0.0000
PTMEAS/CART,8.2500,2.0000,0.0000,0.0000,-1.0000,0.0000
ENDMES
DAT(C)=DATDEF/FA(DATUM_C)
D(1)=LOCATE/XYZDIR,XYZAXI,DAT(A),DAT(B),DAT(C)
```

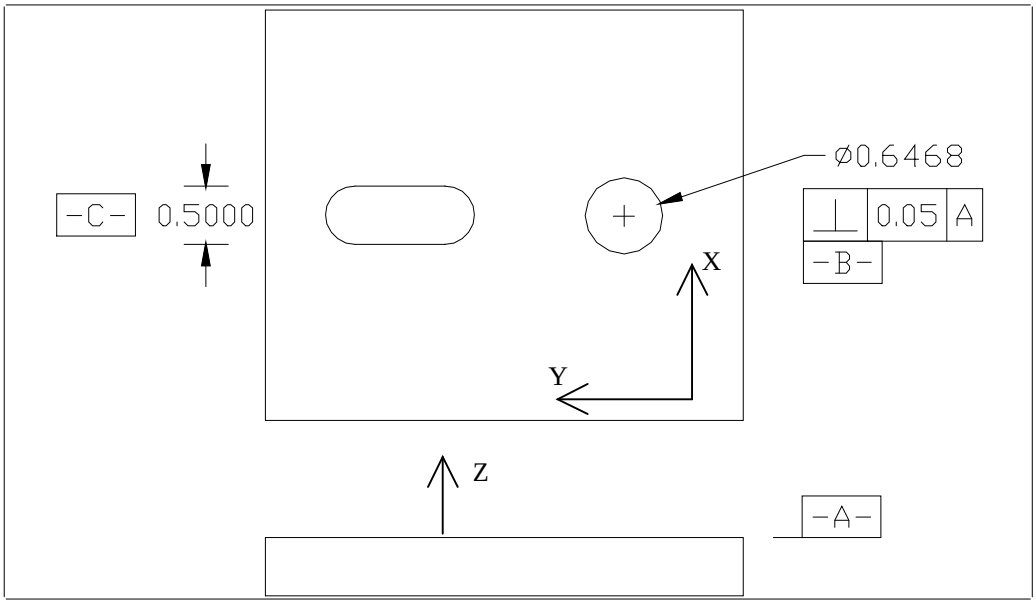


Figure 11 — Plate with a hole and slot

5.3.6.7.5 Over specified coordinate systems

It is possible to specify more features and matings than are necessary to remove all degrees of freedom of motion uniquely. In cases where a coordinate system is over specified a best-fit alignment results. If only features are used in a LOCATE statement, all features have an equal weight in the coordinate system construction. If matings are used, differential fitting tolerances can be specified. This allows specified features to be held more tightly or loosely than others. Using this technique, part deviations along certain degrees of freedom of motion can be balanced.

The example shows the balancing of cross-car deviations on automotive glass. The primary and secondary datum targets are sufficient to just remove all degrees of freedom of motion forward and backward and up and down as well as all rotational degrees of freedom. A single point would be sufficient to remove the translational degree of freedom in the cross-car direction. Instead, two points are specified. By defining matings between the measured points and their nominal locations and applying differential tolerances to those matings, cross-car deviations can be balanced. Zero tolerances are applied to the primary and secondary datum targets to hold those features exactly at nominal. A non-zero tolerance is applied to the cross-car direction points allowing them to deviate from nominal effectively balancing any errors.

```
$$ measure datum targets
$$
$$ Refer to (Figure 12 — LOCATE and MATDEF on automotive glass), for a
$$ graphical example of the MATDEF and LOCATE definitions
$$ on an automotive glass panel.
$$
F(P1)=FEAT/POINT,CART,5251.3456,0.0016,1738.6048,0.7707,0.0004,0.6371
MEAS/POINT,F(P1),1
PTMEAS/CART,5251.3456,0.0016,1738.6048,0.7707,0.0004,0.6371
ENDMES
F(P2)=FEAT/POINT,CART,5371.9536,-620.2612,1370.2181,0.9245,-0.2324,0.3021
MEAS/POINT,F(P2),1
PTMEAS/CART,5371.9536,-620.2612,1370.2181,0.9245,-0.2324,0.3021
ENDMES
F(P3)=FEAT/POINT,CART,5371.9487,620.2511,1370.2140,0.9245,0.2324,0.3021
MEAS/POINT,F(P3),1
PTMEAS/CART,5371.9487,620.2511,1370.2140,0.9245,0.2324,0.3021
ENDMES
F(E1)=FEAT/POINT,CART,5427.6352,452.3434,1277.7473,0.2116,0.0480,-0.9762
MEAS/POINT,F(E1),1
PTMEAS/CART,5427.6352,452.3434,1277.7473,0.2116,0.0480,-0.9762
ENDMES
F(E2)=FEAT/POINT,CART,5427.6386,-452.3534,1277.7503,0.2116,-0.0480,-0.9762
MEAS/POINT,F(E2),1
PTMEAS/CART,5427.6386,-452.3534,1277.7503,0.2116,-0.0480,-0.9762
ENDMES
F(S1)=FEAT/POINT,CART,5268.9839,-689.0167,1544.6344,-0.3977,-0.9139,0.0817
MEAS/POINT,F(S1),1
PTMEAS/CART,5268.9839,-689.0167,1544.6344,-0.3977,-0.9139,0.0817
ENDMES
F(S2)=FEAT/POINT,CART,5268.9736,689.0148,1544.6345,-0.3978,0.9138,0.0816
MEAS/POINT,F(S2),1
PTMEAS/CART,5268.9736,689.0148,1544.6345,-0.3978,0.9138,0.0816
ENDMES
$$ define matings
MA(MAT_P1)=MATDEF/F(P1),FA(P1),PT2PL,BF,0.0,0.0
MA(MAT_P2)=MATDEF/F(P2),FA(P2),PT2PL,BF,0.0,0.0
MA(MAT_P3)=MATDEF/F(P3),FA(P3),PT2PL,BF,0.0,0.0
MA(MAT_E1)=MATDEF/F(E1),FA(E1),PT2PL,BF,0.0,0.0
MA(MAT_E2)=MATDEF/F(E2),FA(E2),PT2PL,BF,0.0,0.0
MA(MAT_S1)=MATDEF/F(S1),FA(S1),PT2PL,BF,1.0,-1.0
```

```
MA(MAT_S2)=MATDEF/F(S2),FA(S2),PT2PL,BF,1.0,-1.0
$$ construct the coordinate system
D(1)=LOCATE/MA(MAT_P1),MA(MAT_P2),MA(MAT_P3),MA(MAT_E1),MA(MAT_E2),$
MA(MAT_S1),MA(MAT_S2)
```

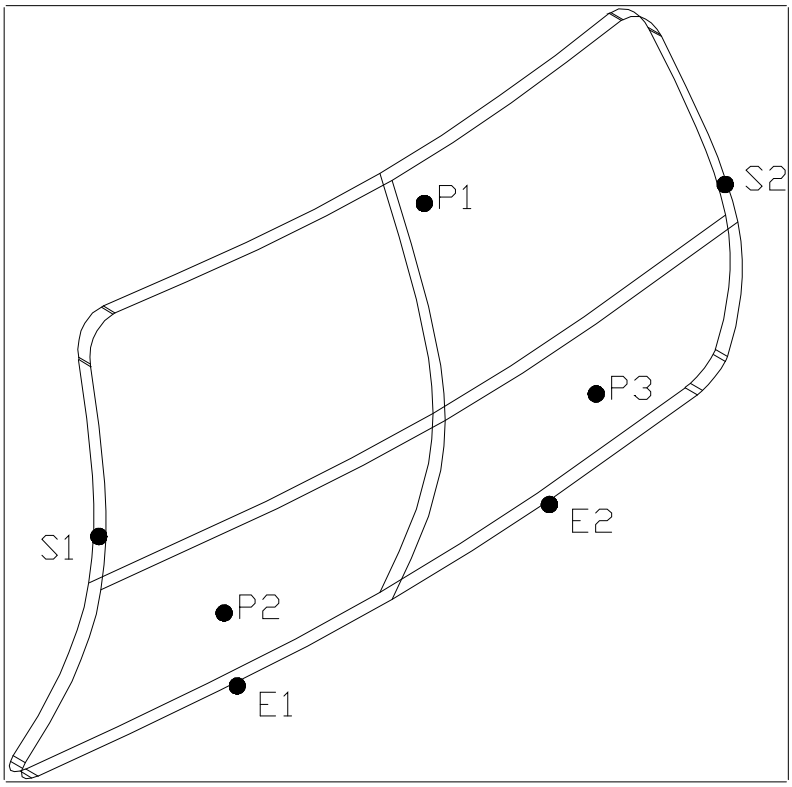


Figure 12 — LOCATE and MATDEF on automotive glass

5.3.7 Measurement uncertainty

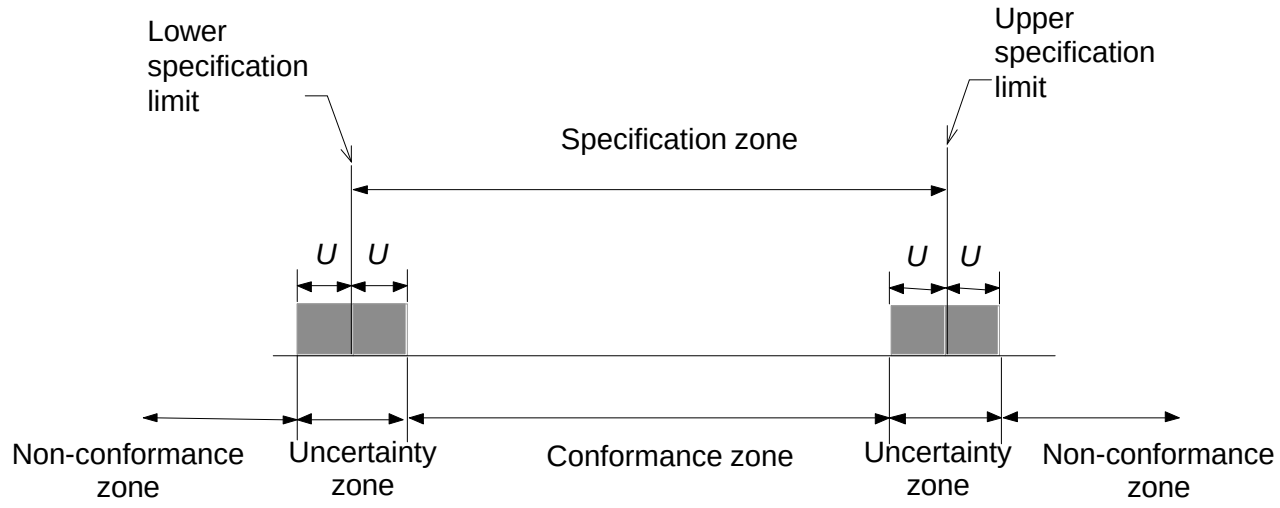
All measurements have uncertainties, and these uncertainties can play important roles in the assessment of conformance to design specifications and for the establishment of traceability of measurements to national and international standards. The uncertainty of a measurement is defined as the parameter, associated with the result of the measurement, that characterizes the dispersion of values that could reasonably be attributed to the measurand. For dimensional metrology, it is necessary to associate with each toleranced parameter its uncertainty at a specified level of confidence. For manufacturers to adhere to current standards, inspection reports must be supplemented with statements like, "The uncertainty of the diameter of this nominally 6 mm diameter hole is ±0.002 mm, at 95% confidence." Obtaining uncertainty statements for measurements derived from commonly encountered dimensional measuring equipment is not trivial. The requirement is not satisfied by merely reporting the DME manufacturer's estimate of the single point uncertainty, or by results of any of the performance evaluation tests specified by various standards bodies, or even by a full geometric characterization of the device. In its 15530 series, the International Organization for Standardization has recognized five possible techniques of determining the uncertainty of measurement in coordinate metrology. These include sensitivity analysis, expert judgment, substitution, computer simulation, and statistical estimation from measurement history. It is not within the scope of the Dimensional Measuring Interface Standard to specify the use of any one of these methods. Whatever the methodology employed, however, all potential sources of measurement error must be considered and must be propagated in a statistically valid way to yield an uncertainty on each measurement result.

5.3.7.1 Measurement traceability

Traceability of a measurement result is defined as the property of the result of the measurement whereby it can be related to stated references, usually national or international standards, through an unbroken chain of comparisons all having stated uncertainties.

5.3.7.2 Design specification conformance/non-conformance

The proof of conformance or non-conformance to design specifications generally requires the stipulation of uncertainties in the pertinent measurements and the use of those uncertainties in agreed-upon decision rules. For example, ISO 14253-1, defines a conformance zone which is narrower than the design specification zone by an amount termed the "expanded uncertainty" at each specification limit. Proof of conformance requires that the measurement result must lie within the conformance zone. Also according to this rule, proof of non-conformance is indicated if the measurement result lies beyond one or the other of the specification limits by an amount equal to the expanded uncertainty. Refer toFigure 13 — ISO 14253-1 Conformance Decision Rule . Proof of conformance or non-conformance is also linked to a level of confidence through the definition of the expanded uncertainty, which will be discussed further in clause 5.3.7.7. It should also be clear from the foregoing that measurement results lying within the expanded uncertainty zones cannot be cleanly classified as conforming or non-conforming. Various other standards organizations (e.g. ASME) have developed alternative decision rules, recognizing that the final acceptance/rejection may, in certain applications, be best served by using different confidence levels on either side of the specification limits. Thus, in general, decision rules deserve a level of flexibility within DME software, and within the DMIS standard as well. It should be noted that from its earliest versions DMIS has provided, in the TOL statements, a mechanism for reporting an in or out of tolerance condition. This has been based on the simplest rule of conformance which identifies the conformance zone as identical to the specification zone (i.e., setting U = 0 in Figure 13 — ISO 14253-1 Conformance Decision Rule). With access to measurement uncertainty data, more sophisticated analyses are possible.



U represents the “expanded uncertainty” of the measurement

Figure 13 — ISO 14253-1 Conformance Decision Rule

5.3.7.3 Specifying an algorithm for uncertainty evaluation

Given the variety of methods for uncertainty evaluation, it is necessary to provide a means of identifying specific methods for future reference within an inspection program. The UNCERTALG statement provides this functionality.

5.3.7.4 Specifying a conformance decision rule

Given the variety of decision rules, it is necessary to provide a means of identifying specific conformance rules for future reference within an inspection program. The CNFRMRUL statement provides this functionality.

5.3.7.5 Activating an uncertainty evaluation algorithm and decision rule

The UNCERTSET statement activates or deactivates an uncertainty algorithm and, optionally, a conformance decision rule. A decision rule cannot be activated in the absence of an uncertainty algorithm activation; the

converse, however, is permitted by indicating NONE in place of the decision rule identifier. Once activated, a given uncertainty algorithm (and, perhaps an accompanying decision rule) will remain in force until deactivated by an UNCERTSET/OFF statement or supplanted by another UNCERTSET activating another uncertainty algorithm and decision rule.

5.3.7.6 DMIS uncertainty data

DMIS recognizes two parameters as key information on measurement uncertainty. The first of these is the assessment of any mean error (bias) in the measurement. The second parameter is the combined uncertainty which is the standard deviation about the mean error. In the ideal, the mean error will be close to zero, although this is by no means assured, as uncompensated systematic errors in measurement often exist. (For example, part measurements at non-standard temperature and without temperature correction can easily show significant bias.) If the mean error is negligible, then the expanded uncertainty may be obtained by multiplying the combined uncertainty by a "coverage factor" that relates the desired confidence level to the expanded uncertainty. Assuming a normal distribution of measurement errors with zero bias, a coverage factor of 2 corresponds to a confidence level of 95% while a value of 3 corresponds to a 99.7% confidence level. Where measurement bias is not negligible, it must either be corrected for directly or incorporated into the measurement uncertainty evaluation. One approach to such incorporation is to include the absolute value of the bias in the calculation of the expanded uncertainty.

5.3.7.7 Accessing uncertainty data and conformance rule results

Only measurement results that have tolerances applied to them can yield uncertainty data. Tolerances are evaluated with respect to a feature actual or a pair of feature actuals, where appropriate, by the execution of either an EVAL or OUTPUT statement. Such evaluations may include the reporting of decision rule results and/or measurement uncertainty data in the TA(lname) tolerance actuals statement.

A fairly comprehensive example of use of measurement uncertainty statements and the corresponding output file data is as follows:

EXAMPLE

Input file code fragment:

```
DECL/GLOBAL,CHAR,11,uastatus,drastatus
DECL/GLOBAL,DOUBLE lotol,uptol
DECL/GLOBAL,DOUBLE diam_dev, expanded_uncert, mean_error, combined_uncert

$$ Specify an uncertainty evaluation algorithm and a conformance decision rule

U(Simulation)=UNCERTALG/ALGOR,1
DR(ISO14253)=CNFRMRUL/RULE,2

$$ Activate uncertainty evaluation algorithm and conformance decision rule
$$ Assess DME support level for uncertainty
$$ NOTE: This next statement activates the following output:
$$ UNCERTSET/ON,UA(Simulation),DR(ISO14253)
$$ U(Simulation)= UNCERTALG/ALGOR,1
$$ DR(ISO14253)= CNFRMRUL/RULE,2
UNCERTSET/ON,U(Simulation),DR(ISO14253)
$$ NOTE: UNCERTSET statement causes the following 3 lines to be output
$$ UNCERTSET/ON,U(Simulation),DR(ISO14253)
$$ UA(Simulation)=UNCERTALG/SUPPORTED
$$ DRA(ISO14253)= CNFRMRUL/UNSUPPORTED
uastatus=OBTAIN/UA(Simulation),1
drastatus=OBTAIN/DRA(ISO14253),1

$$ Define feature nominal, measure feature, specify tolerance,
$$ apply tolerance to feature
```

```
F(circle)=FEAT/CIRCLE,INNER,CART,  0.000,  0.000,  10.000,0.000,0.000,1.000,10.000
MEAS/CIRCLE,F(circle),4
PTMEAS/CART,5.000,0.000,10.000,-1.000,0.000,0.000
PTMEAS/CART,-5.000,0.000,10.000,1.000,0.000,0.000
PTMEAS/CART,0.000,5.000,10.000,0.000,-1.000,0.000
PTMEAS/CART,0.000,-5.000,10.000,0.000,1.000,0.000
ENDMES
T(diam)=TOL/DIAM,-0.010,0.010
OUTPUT/FA(circle),T(diam)

$$ Decision rules can also be programmed, in case the DME can't
$$ support them directly.

IF/(uastatus .EQ. 'SUPPORTED')
  IF/(drastatus .EQ. 'UNSUPPORTED')
    lotol=OBTAIN/T(diam),2
    uptol=OBTAIN/T(diam),3
    diam_dev=OBTAIN/TA(diam),2
    mean_error=OBTAIN/TA(diam),4
    combined_uncert=OBTAIN/TA(diam),5
     expanded_uncert=ASSIGN/ABS(mean_error)+2*combined_uncert
IF/(diam_dev .GT. lotol+expanded_uncert .AND. diam_dev .LT. uptol- $ expanded_uncert)
      TEXT/OUTFIL,'Diameter is in tolerance at 95% confidence level.'
    ELSE
IF/(diam_dev .LT. lotol-expanded_uncert .OR. diam_dev .GT. $ uptol+expanded_uncert)
        TEXT/OUTFIL,'Diameter is out of tolerance at 95% confidence level.'
      ELSE
        TEXT/OUTFIL,'At 95% confidence level, diameter INTOL/OUTOL is unknown.'
      ENDIF
    ENDIF
  ENDIF
ENDIF
```

Corresponding output file code fragment:

```
UNCERTSET/ON,UA(Simulation),DR(ISO14253)
U(Simulation)= UNCERTALG/ALGOR,1
DR(ISO14253)= CNFRMRUL/RULE,2
UA(Simulation)=UNCERTALG/SUPPORTED
DRA(ISO14253)= CNFRMRUL/UNSUPPORTED
FA(circle)=FEAT/CIRCLE,INNER,CART,0.032,-0.008,10.000,0.000,0.000,1.000,9.995
TA(diam)=TOL/DIAM,-0.005,  RULEUNSUP,0.0005,0.001
TEXT/OUTFIL,' Diameter is in tolerance at 95% confidence level.'
```

5.4 Equipment control

5.4.1 ZYZ Euler angles

The convention of specifying a coordinate system relative to a reference coordinate system by the means of three angles of rotation: around the reference system Z axis, the resultant coordinate system Y axis and the further resultant Z axis. Refer to Figure 14 — Euler transformation 1, Figure 15 — Euler transformation 2, Figure 16 — Euler transformation 3, and Figure 17 — Euler transformation 4.

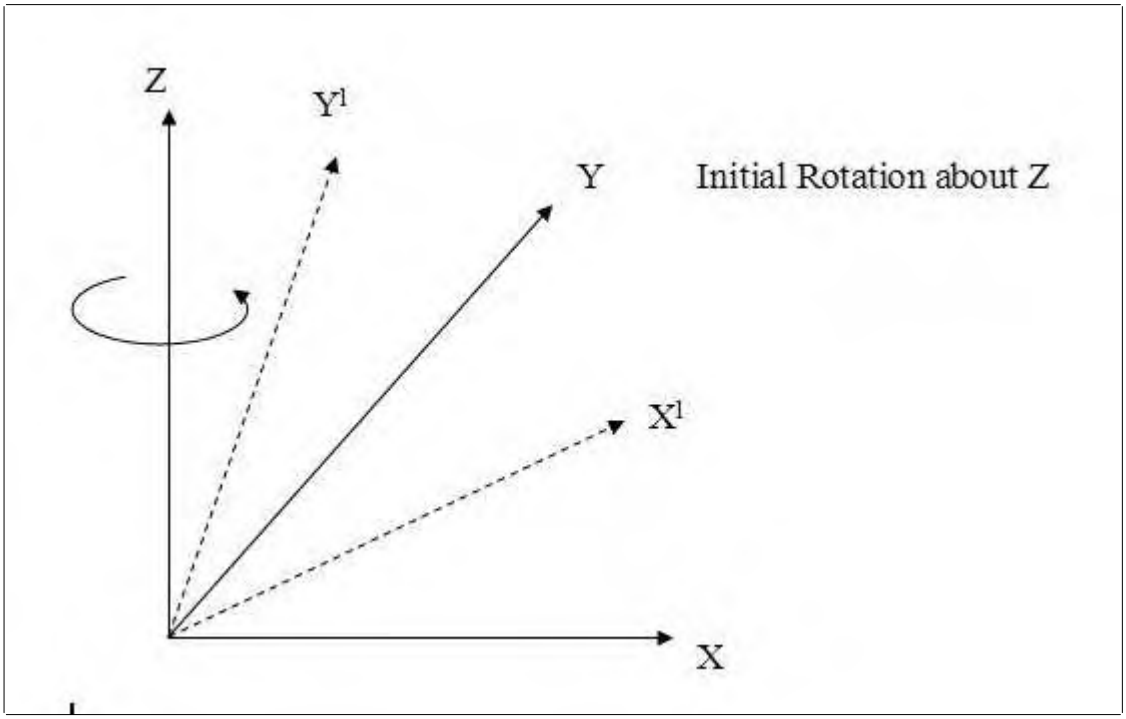


Figure 14 — Euler transformation 1

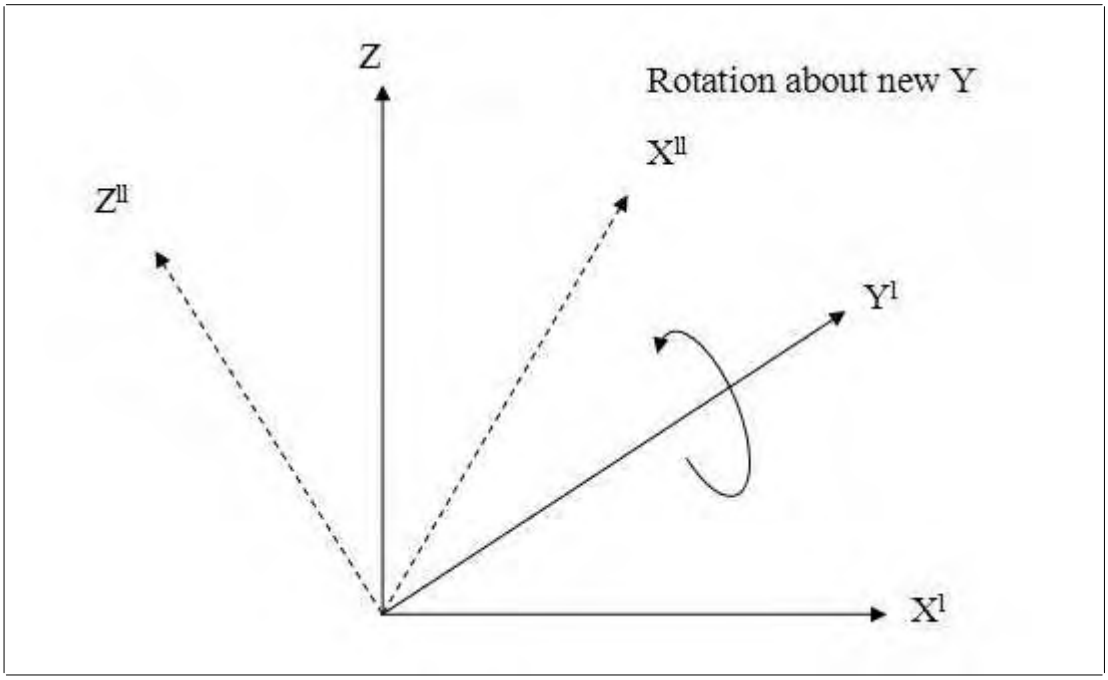


Figure 15 — Euler transformation 2

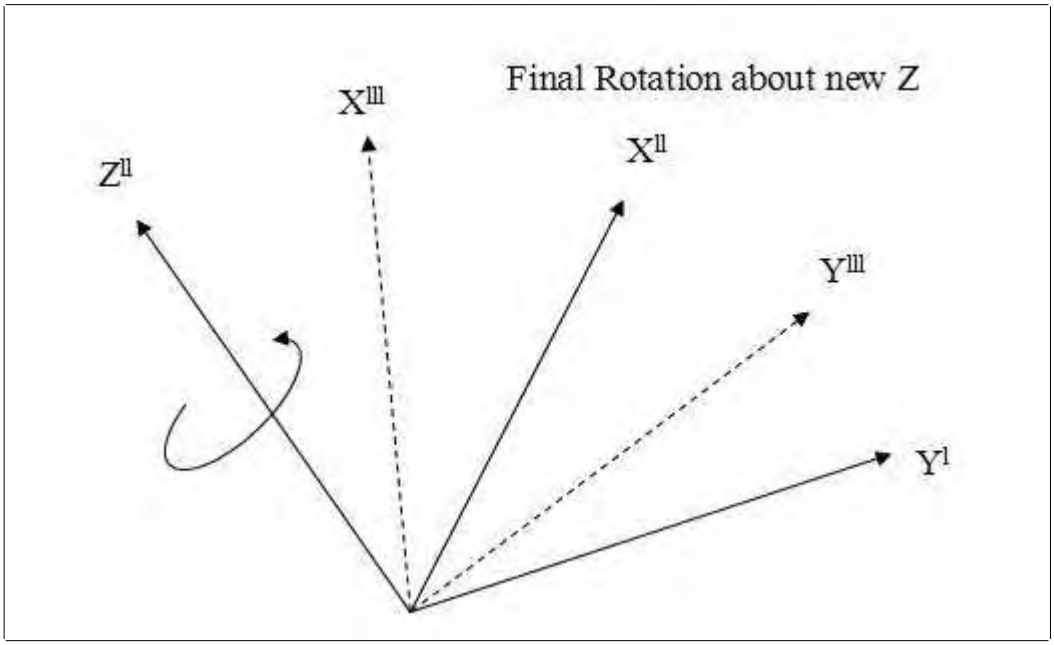


Figure 16 — Euler transformation 3

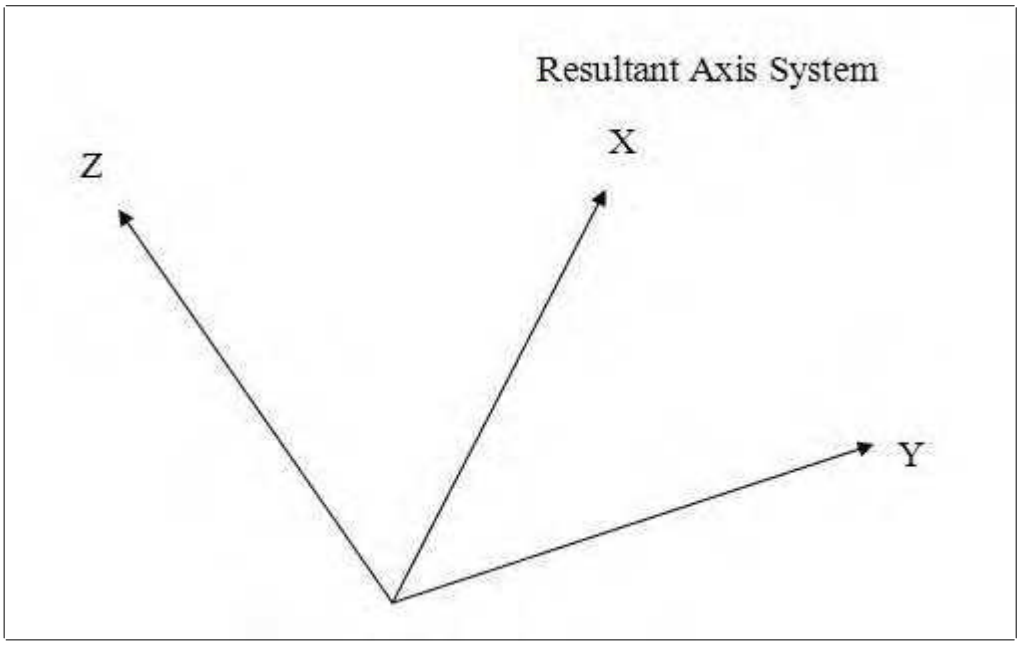


Figure 17 — Euler transformation 4

5.4.2 Machine parameters

Machine parameters control machine acceleration rates, machine feed rates, error condition control, modes of motion and execution, temperature and probe compensations, and the units used in the program.

5.4.2.1 Error condition

The BADTST, ERROR, and RESUME statements and the BADGT(), BADPT(), SERROR(), SILTCH(), and SENSNOTOUCH() intrinsic functions are used in DMIS programs to effect program execution in the event of an error.

DMIS allows for error handling by two methods: interrupt and polling.

In the interrupt method, program branching is set up before statements which may cause an error condition are executed. When an error is encountered, program execution is automatically transferred to the pre-defined location.

In the polling method, high-level language statements testing for an error condition must be used after every motion statement for which error handling is desired. If an error is detected, program execution can be transferred to a remote error handling program module or handled locally. Program branching is explicit as part of the error testing logic.

The use of interrupt and polling methods of error handling are somewhat mutually exclusive. Turning on the polling method suppresses the interrupt method for some error conditions. However, once inside an interrupt method error handling routine, using the polling method of error detection and handling can be very useful.

The use of the ERROR statement allows for branching in the event of an error by the interrupt method. Branching can be to statement labels within the inspection program, or to DME specific error recovery routines. The ERROR statement allows for the use of minor words to handle an unexpected probe trigger with ILLEGALTOUCH, or the absence of an expected probe trigger with NOTOUCH. The ERROR statement also allows for handling of DME specific errors by using error codes that the DME lists in the ERROR section of the characterization file. The current DME error code, that is, the code for the error which caused program execution to branch, can be determined by using the SERROR() intrinsic function. The DME specific error code for illegal or unexpected probe trigger can be found using the SILTCH() intrinsic function and the DME specific error code for no probe trigger when one was expected can be found using the SENSNOTOUCH() intrinsic function.

Note: A no probe trigger condition is often caused by the probe moving beyond the search distance specified in the SNSET/SEARCH statement.

Once in an error handling portion of the DMIS program to which program execution has been transferred because of the action of the ERROR statement, further program branching by the action of the ERROR statement is not allowed, that is, program interruption due to errors is disabled.

When the error has been handled, program execution can be transferred back to the calling portion of the program using the RESUME statement. The RESUME statement allows for the use of minor words to control the exact point at which program execution resumes. The statement which caused the error can be re-executed by using CURENT, that statement can be skipped and the next DMIS statement in the program executed by using NEXT. If the error occurred within a measurement block then END can be used to skip the entire measurement block or START can be used to re-measure from the beginning of the measurement block. If the error is unrecoverable, STOP can be used to abort program execution entirely. The RESUME statement can also be used to transfer program execution to a statement label.

When the RESUME statement is executed, interrupt error handling is re-enabled. For this reason, there should be no confusion between the action of a JUMPTO/(jumptarget) statement and a RESUME/(jumptarget) statement. A JUMPTO/(jumptarget) statement can be used to branch inside an error handling routine without re-enabling program interruption. A RESUME/(lname) marks the end of an error handling routine with program interruption being re-enabled.

The BADTST statement is used to enable and disable error condition polling. The polling method handles only two error conditions: an illegal or unexpected probe trigger, and no probe trigger when one was expected. When polling is enabled, program branching using the ERROR statement is disabled for those error conditions only. Furthermore, DME specific handling of unexpected and no probe trigger error conditions is also suspended. When such an error condition is encountered, program execution is simply passed to the next statement in the DMIS program.

When error polling has been enabled by execution of the BADTST/ON statement, error conditions can be detected using the BADGT() and BADPT() intrinsic functions. BADGT() will return .TRUE. if the last probe motion produced an illegal or unexpected probe trigger, BADPT() will return .TRUE. if the last point measurement failed to cause the probe to trigger. It is important to test the value of BADGT() after every GOTO statement and the value of BADPT() after every MEAS/POINT…ENDMES block when polling is enabled. Otherwise, the DME may be instructed to move while an error condition exists and physical damage may result. Execution of the BADTST/OFF statement

re-enables interrupt error handling if previously enabled with the ERROR statement and/or re-enables DME specific error handling routines.

The following is an example of the use of error handling by both interrupt and polling methods:

```
DMISMN,'Error handling example',05.1
DISPLY/PRINT,DMIS
FILNAM/'Circle search',05.1
UNITS/MM,ANGDEC
$$ align the part
  executable statements
MODE/PROG,MAN
$$ turn polling on for next probe move
BADTST/ON
$$ move over next circle to be measured
GOTO/0,0,10
$$ check for error
IF/(BADGT())
  MODE/MAN
   TEXT/MAN,'Move clear and open the clamp at circle circ1'
   MODE/PROG,MAN
   $$ retry probe move
   GOTO/0,0,10
   IF/(BADGT())
   $$ no recovery for 2nd failure, quit program
   JUMPTO/(end)
   ENDIF
ENDIF
$$ turn polling off
BADTST/OFF
$$ turn interrupt on for all errors, branch to (errblk)
ERROR/(errblk),ALL
$$ declare variable for number of retries
DECL/INTGR,retries
$$ push current coordinate system for later recall
PUSH/DATSET
$$ measure a circle, reset number of retries to zero
retries=ASSIGN/0
F(circ1)=FEAT/CIRCLE,INNER,CART,0,0,0,0,0,1,40
MEAS/CIRCLE,F(circ1),4
PTMEAS/CART,20,0,0,-1,0,0
PTMEAS /CART,-20,0,0,1,0,0
PTMEAS /CART,0,20,0,0,-1,0
PTMEAS /CART,0,-20,0,0,1,0
ENDMES
$$ restore coordinate system for reporting
POP/DATSET
OUTPUT/FA(circ1)
$$ rest of program
  executable statements
$$ jump around error routine to end of program
JUMPTO/(end)
$$ the error handling routine
(errblk)
$$ test the error code
SELECT/SERROR()
  CASE/'ILLEGALTOUCH'
    $$ unexpected touch, abort program
    RESUME/STOP
```

```
  ENDCAS
  CASE/'NOTOUCH'
    IF/(retries .LT. 5)
      $$ search for circle along X axis
      D(temp)=TRANS/XORIG,5
      $$ restart measurement block
      RESUME/START
    ELSE
      $$ too many retries, skip feature
      RESUME/END
    ENDIF
  ENDCAS
  DFTCAS
    $$ unknown error, abort program
    RESUME/STOP
  ENDCAS
ENDSEL
(end)
ENDFIL
```

5.4.2.2 Mode

The MODE statement defines the mode in which the DME will execute the program. Mode can specify one or more of the following: automatic mode (AUTO), programmed mode (PROG), and manual mode (MAN). The mode specified effects execution of measurement, motion, and sensor calibration statements.

The MODE statement specifies the mode in which the DME will execute CALIB, GOTARG, MEAS and RMEAS statements. It specifies the primary mode in which the DME will try to execute the statements. It may also specify secondary and tertiary modes. If the DME cannot execute a measurement or motion sequence in the primary mode, it will then try to execute it in the secondary mode. If it cannot do this, it will execute the statement in the tertiary mode. In the following example, automatic mode is the primary mode, programmed mode is the secondary mode, and manual mode is the tertiary mode:

```
MODE/AUTO,PROG,MAN
```

In the above example, the DME will try to execute moves and measurements in automatic mode. When it encounters feature measurements or motions that cannot be performed automatically, it will execute them in programmed mode. If it cannot execute the programmed mode, it will go into manual mode. These decisions are made on a feature-by-feature and motion-by-motion basis. That is, the decision is made each time a new CALIB, GOTARG, MEAS or RMEAS statement is encountered.

AUTO always precedes PROG or MAN, and PROG always precedes MAN. Valid forms of the MODE statement are:

```
MODE/AUTO,PROG,MAN
MODE/AUTO,MAN
MODE/PROG,MAN
MODE/MAN
```

PROG can never be set as a primary mode with AUTO as a secondary mode. Examples of invalid forms of the MODE statement follow:

```
MODE/PROG,AUTO,MAN
MODE/MAN,PROG,AUTO
MODE/MAN,AUTO
```

5.4.2.3 Motion

The ACLRAT statement sets the rates of acceleration for measurement, machine motion, and rotary tables.

The FEDRAT statement sets the velocities for measurement, machine motion, and rotary tables.

5.4.2.4 Probe compensation

Probe compensation is a process that uses the characteristics of a sensor, usually derived from calibration, to bring measured data to the surface of the measured feature.

Automatic probe compensation can be turned on or off with the PRCOMP statement.

5.4.2.5 Temperature compensation

The machine's temperature compensation can be turned on or off with the TECOMP statement. A part's temperature compensation can be turned on or off along with specifying a part thermal expansion coefficient with the TECOMP statement.

5.4.2.6 Units

The UNITS statement sets the units of measure for distance, angles, and temperature. This statement may be used multiple times in a program.

After the UNITS statement is issued, all definitions that follow will be in the units specified by the statement. At the time when a new UNITS statement is issued, all previously defined features, tolerances, sensors, and coordinate systems will be referenced in these new units. In the following example, the point and tolerance is defined in inches. At the time of execution of the OUTPUT statement, the point and tolerance values will be reported as millimeters to include the measured values of the point.

```
MODE/AUTO,PROG,MAN
UNITS/INCH,ANGDEC
DECL/DOUBLE,MM_X,INCH_X
T(TX1)=TOL/CORTOL,XAXIS,-0.010,0.010
F(PT1)=FEAT/POINT,CART,1.0,1.0,1.0,0,0,1
MEAS/POINT,F(PT1),1
ENDMES
INCH_X=OBTAIN/FA(PT1),3
UNITS/MM,ANGDEC
OUTPUT/FA(PT1),TA(TX1)
MM_X=OBTAIN/FA(PT1),3
TEXT/OUTFIL,CONCAT('Point X in mm is: ',STR(MM_X))
TEXT/OUTFIL,CONCAT('Point X in inch is: ',STR(INCH_X))
```

Output from the TEXT statements to the operator is:

Point X in mm is: 25.4
Point X in inch is: 1.0

Output from the TEXT statements to DMIS output file(s) is:

```
TEXT/OUTFIL,'Point X in mm is: 25.4'
TEXT/OUTFIL,'Point X in inch is: 1.0'
```

5.4.3 Rotary tables

Rotary table statements provide a set of capabilities required to support several applications.

Multiple rotary tables can be defined with the ROTDEF statement and calibrated with the CALIB statement.

Rotary motion is controlled with the ROTAB statement which also maintains control of part coordinate system updating. Updating options include total, partial, or none.

The ROTSET statement is used to reset the angular counter value for the rotary table.

Velocity and acceleration are controlled with the FEDRAT and ACLRAT statements respectively.

5.4.3.1 Rotary table calibration

The rotary table calibration is performed with the CALIB/RTAB statement. The CALIB/RTAB statement provides a means by which the location and orientation of the rotary table can be accurately established relative to the machine coordinates. Rotary table calibration can be performed by providing the DME with either a specified algorithm, a single feature definition that establishes the axis of rotation or a pair of feature definitions that establishes both the axis of rotation and the rotary table top. The CALIB sequence is terminated with an ENDMES statement.

Various DMIS statements can be used in a rotary table calibration block. A list of statements that may be issued between the CALIB/RTAB and ENDMES statements is shown in (Table 9 — DMIS statements allowed in a rotary table calibration block).

Table 9 — DMIS statements allowed in a rotary table calibration block

(jumptarget)	DMESW	FEDRAT	OBTAIN	SAVE/SA(lname)
ACLRAT	DO	FINPOS	PAMEAS	SELECT
ASSIGN	ELSE	FROM	PTMEAS	SNSET
BADTST	ENDCAS	GOHOME	RAPID	SNSLCT
CASE	ENDDO	GOTARG	RECALL/S(lname)	TEXT
CRSLCT	ENDGO	GOTO	RECALL/SA(lname)	VALUE
CZSLCT	ENDIF	IF	ROTAB/ROTNUL	WKPLAN
DFTCAS	ENDSEL	INCLUD	ROTSET	
DMEHW	ERROR	JUMPTO	SAVE/S(lname)	

If the INCLUD statement is used to call external code within a rotary table calibration block, the external code must contain only statements allowed within a CALIB/RTAB ...ENDMES block. IF...ENDIF, DO...ENDDO, and SELECT...ENDSEL blocks must be fully contained within the CALIB/RTAB...ENDMES block. If the JUMPTO statement is used in a rotary table calibration block, the (jumptarget) to which program control will be transferred must be within the same CALIB/RTAB ...ENDMES block. Program control cannot be transferred to a (jumptarget) inside a rotary table calibration block from outside that rotary table calibration block.

5.4.4 Sensors and sensor-related

When dealing with different types of DMEs, a major difference is the sensor used in measurement. DMIS currently supports a variety of sensors including: probes, cameras, lasers, infrared sensors and non-contact electrical capacitance sensors. Additionally, scanning routines of various types are also supported. The SNSDEF statement has several formats including touch probes, video cameras, lasers, infrared and non-contact electrical capacitance. Further, these sensors can be defined in either a fixed or an indexable condition.

5.4.4.1 Sensor definition

Sensors must first be defined either within the part program or external to one. There are several formats to the SNSDEF statement. Each definition provides an allowance to define either a fixed or indexable sensor.

5.4.4.2 Sensor assignment

Each sensor has a label assigned in the sensor definition. Once a sensor has been calibrated, its calibration data can be stored using the SAVE statement. It can later be recalled with the RECALL statement or deleted with the DELETE statement. The SAVE, RECALL, and DELETE statements can apply to sensors defined/calibrated either within a part program or external to one. Sensor calibration data is saved and recalled using the sensor label as a

reference, allowing several sensors to be saved simultaneously. When a sensor is saved and then recalled, it simply allows the sensor to be used again without having to be recalibrated. Some DMEs automatically store calibration data when a sensor is calibrated. This may allow several sensors to be calibrated and used without use of the SAVE and RECALL statements. These DMEs may only require the use of the SNSLCT statement to select a sensor that was previously defined and calibrated.

5.4.4.3 Sensor calibration

The CALIB statement is optional in an inspection program. If the sensor calibration is to be done in manual mode, or if the calibrating feature is of a type that is not supported by the DMIS vocabulary, the CALIB statement can be omitted. In this case, the sensor is calibrated by an operator prior to beginning execution of the DMIS part program.

Sensors should be calibrated prior to use either within the part program or external to one. A probe, for example, is calibrated for tip diameter compensation prior to measurement. Likewise, the image taken from a camera on a video DME must also be calibrated for size compensation. Once this has been done, calibration data can be stored and then recalled at a later time.

Sensors are calibrated from a feature of known size. The feature definition of the calibrating feature is passed to the DME in the inspection program. The CALIB statement is used to calibrate the sensor, and is similar in structure to the MEAS and RMEAS statements for feature measurement. Calibration may be performed in manual, programmed, or automatic mode. In manual mode the sensor is calibrated by the operator, in programmed mode the calibration sequence in the inspection program is followed, and in automatic mode the DME performs the calibration sequence using its own algorithm.

The calibrating feature definition is the same as any other feature nominal definition. It therefore has size, location, and orientation information. The size, location and orientation of the feature are important, and are with respect to the machine coordinate system. Once a coordinate system has been established, the DME can be programmed to go to a calibration fixture and calibrate sensors under servo control.

Various DMIS statements can be used in a sensor calibration block. A list of statements that may be issued between the CALIB/SENS and ENDMES statements is shown in (Table 10 — DMIS statements allowed in a sensor calibration block).

Table 10 — DMIS statements allowed in a sensor calibration block

(jumptarget)	DMESW	ERROR	INCLUD	SELECT
ACLRAT	DO	FEDRAT	JUMPTO	SNSET
ASSIGN	ELSE	FINPOS	OBTAIN	TEXT
BADTST	ENDCAS	FROM	PAMEAS	VALUE
CASE	ENDDO	GOHOME	PTMEAS	WKPLAN
CZSLCT	ENDGO	GOTARG	RAPID	
DFTCAS	ENDIF	GOTO	ROTAB	
DMEHW	ENDSEL	IF	ROTSET	

If the INCLUD statement is used to call external code within a sensor calibration block, the external code must contain only statements allowed within a CALIB/SENS...ENDMES block. IF...ENDIF, DO...ENDDO, and SELECT...ENDSEL blocks must be fully contained within the CALIB/SENS...ENDMES block. If the JUMPTO statement is used in a sensor calibration block, the (jumptarget) to which program control will be transferred must be within the same CALIB/SENS...ENDMES block. Program control cannot be transferred to a (jumptarget) inside a sensor calibration block from outside that sensor calibration block.

5.4.4.4 Sensor motion

Some hardware configurations require that the motion of the sensor be controlled explicitly during both motion and measurement, for example a continuously indexable probe head or a posable inspection robot.

5.4.4.4.1 Sensor axis system

In order to be able to specify the orientation of a sensor head relative to the part coordinate system (PCS), it is necessary to define the axis system used as a reference. Using the Euler angle convention of ZYZ (refer to clause 5.4.1), the resultant head orientation will be such that the active probe tip shaft (or the main direction of an optical sensor) is aligned with the Z axis pointing towards the sensor. The alignment of the tool with respect to the X and Y axes will be tool dependent, e.g. for a single stripe laser sensor, it might be useful to specify the X axis as normal to the laser plane.

Specifying probe orientation relative to PCS makes the program equipment independent, so a laser probe inspection sequence written for a bridge-type CMM would be transferable to an inspection robot without modification of motion or measurement statements.

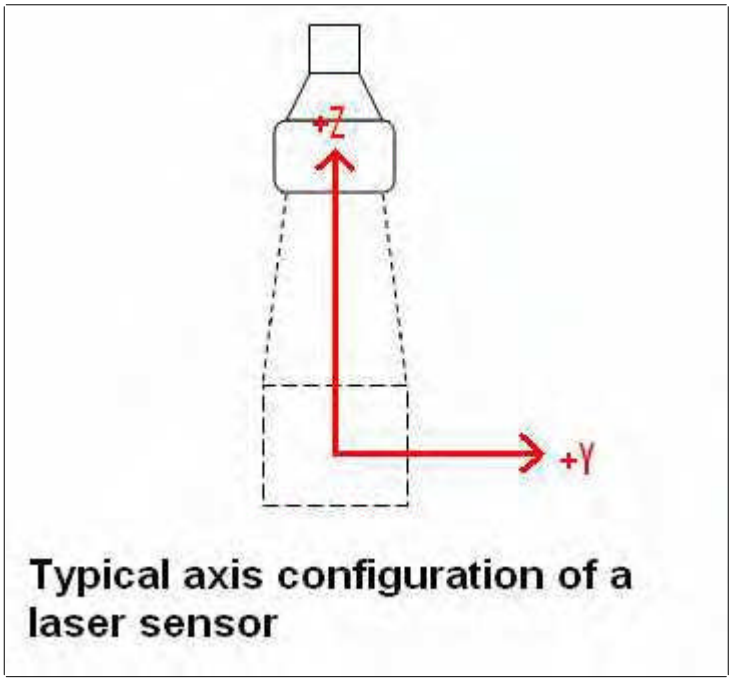


Figure 18 — Sensor axis system

5.4.4.5 Sensor setting

The SNSET statement is used to specify sensor setting for a variety of sensors. Probes, for example, have settings including: distances for approach, search, and retract, etc. Also included are settings for video devices, that is, light settings, window subsets, filters, etc. The attributes selected by the SNSET statements executed for a given carriage apply to the currently selected sensor for that carriage.

5.4.4.6 Sensor mount

A relationship between the probe coordinate system and the machine coordinate system is established through the use of the SNSMNT statement. Its function is to aid in accurately modelling the probe at the CAD system and supporting output of the actual probe offset in the output file.

5.4.4.7 Sensor selection

Once sensors have been defined and calibrated, and settings activated, the appropriate sensor must be selected with the SNSLCT statement, when not already in place.

The SNSLCT statement provides for automatic sensor changing when supported by the DME, and the sensor holder is appropriately defined with the THLDEF statement. Safe positioning prior to an automatic sensor change is encouraged to avoid sensor collision.

When a SNSLCT statement specifies that the sensor be orientated either in PCS or HEADCS (the local coordinate system of the sensor), the motion produced will involve the rotation of the sensor axes alone, i.e. it will not cause motion of the CMM frame.

Examples:

```
SNSLCT/SA(R1),HEADCS,-45.0,90.0
```

In the example, the result is to activate the sensor R1, and orient it such that the absolute angles of the sensor head are -45 and 90. The motion produced will involve only the head axes.

```
SNSLCT/SA(R1),PCS,45,90,180
```

Instances of the simplest possible use of SNSLCT may be found in examples A.7, A.9, A.19.2, A.28, and A.31. Instances of adjusting wrists using SNSLCT may be found in examples A.30.1, A.30.5, and A.30.7. Instances of using SELECT to select a probe on a multiprobe sensor may be found in example A.30.4.

5.4.4.8 Video systems

The LITDEF statement has two formats to define any type of lighting arrangement from any orientation. Strobe lighting and image acquisition are mutually managed through either a cycle or a trigger mode.

Two formats of the WINDEF statement are provided to define a subset of a video system's field of view - a viewing window.

5.4.5 Carriages

Multiple carriage DMEs are defined as being coordinate measuring machines, composed of 2 or more carriages related to a single machine coordinate system, operating on a single part from a single DMIS part program.

To accommodate these systems, the CRGDEF statement has the necessary parameters to define the work zone and ram axis motion for each independent carriage. The CRGDEF statement also assigns a label to a carriage.

The CRSLCT statement marks the beginning of either a section of the part program relating exclusively to the selected carriage or a common section which is related to no carriage.

DMIS statements can be used in any section, with the following limits and interpretations:

5.4.5.1 Statement execution

The DME is responsible for the parallel simultaneous execution of each carriage within the limits imposed by the CRMODE statement.

Since each carriage is executing independently, variable data can not be shared among different carriage sections. The DME internally associates a carriage attribute to the data. The CRSLCT/ALL statement marks the beginning of a common section wherein data belonging to any carriage section may be used. Data defined in common sections have common attributes and may be used, but not modified, in any carriage section.

All DMIS statements not explicitly listed in (Table 11 — Carriage common statements) and (Table 12 — Carriage specific statements) can be used in every section, with the following exceptions:

— The RECALL and SAVE statements become carriage specific when they manage sensors and become carriage common statements when they manage rotary tables.

— The TRANS statement becomes carriage specific when the translation of the coordinate system is based on the PRBRAD modifier.

Even if the SNSET statement is related to the DME motion and hence to carriage specific information, the use of the SNSET statement is allowed in a carriages common section, because very often these parameters are common defaults to be shared among all carriages.

5.4.5.2 Carriage common statements

The information contained in the ROTDEF and TECOMP statements is considered global and must be programmed in common sections of a multiple carriage program.

The use of the PRBRAD or –PRBRAD minor word in a TRANS statement is not allowed in a common section of a multiple carriage program.

On multiple carriage DME's, certain statements only have meaning when programmed in a common section, because they supply global information to all the carriages, so it is better to avoid them in a specific carriage section (refer to clause Table 11 — Carriage common statements).

The statements related to a rotary table must, in particular, be programmed in a common section. This is for safety reasons, to prevent crashes of the carriages which are not involved in the rotary table motion.

A carriage common section can accept all DMIS statements, excluding those listed in (Table 12 — Carriage specific statements).

Table 11 — Carriage common statements

CRGDEF	(*) MODE	ROTAB	ROTSET
CRMODE	POP	ROTDEF	TECOMP
CZONE	PUSH		
NOTE **(*) Exceptions**			

For safety reasons, the MODE statement is a carriage common statement only when the DME switches from a CNC mode (AUTO or PROG) to MAN mode and vice versa. No MODE statement, including AUTO or PROG and MAN, (for example: MODE/AUTO,MAN) is allowed in a carriage specific section, as this will generate an error from the DME.

5.4.5.3 Carriage specific statements

When programming multi-carriage DME's some statements are only meaningful within one specific carriage section, because they manage information, (for example: motion), measured points and dynamic parameters which are not valid for any other carriages (refer to clause Table 12 — Carriage specific statements).

The statements related to the sensors management must, in particular, be programmed in the specific carriage section, because even if sensor components could have the same dimensions they refer to different physical objects available on different carriages. In this way, the programmer can easily make changes to each carriage configuration without effecting the configurations of all the others.

A specific carriage section can accept all DMIS statements, with the exception of those listed in (Table 11 — Carriage common statements).

Table 12 — Carriage specific statements

ACLRAT	FINPOS	RAPID	SNSLCT
CALIB/SENS	FROM	REFMNT	SNSMNT
CMPNTGRP	GOHOME	SCNMOD	THLDEF
CROSCL	GOTARG	SCNSET	WRIST
CZSLCT	GOTO	SENSOR	
ENDGO	GROUP	SNSDEF	
EXTENS	PAMEAS	SNSET	
FEDRAT	PTMEAS	SNSGRP	

5.4.5.4 Modal statements

When executed within a common section, modal statements will establish global settings for all carriages.

When executed within a carriage specific section, modal statements effect only that specific carriage.

Modal statements effect the DME, and change its general behavior (refer to clause Table 13 — Modal statements).

This new behavior remains in effect until the DME executes a new issue of the same statement.

With the exception of the carriage-specific statements, all the modal statements effect all carriages when executed in a common section.

If the DME's modal status is changed in a carriage specific section, that specific carriage will retain its own setting at the end of the section execution. It does not return to any status previously defined in a carriage common section.

Table 13 — Modal statements

Statement	Status
ACLRAT	Settings stay in force until the next ACLRAT statement.
BADTST	ON/OFF statement.
CRMODE	Settings stay in force until the next CRMODE statement.
CROSCL	ON/OFF statement.
DECPL	Settings stay in force until the next DECPL statement.
DISPLY	Settings stay in force until the next DISPLY statement.
DMIS	ON/OFF statement.
ERROR	ON/OFF statement.
FEDRAT	Settings stay in force until the next FEDRAT statement.
FINPOS	ON/OFF statement.
FLY	Settings stay in force until the next FLY statement.
GEOALG	Settings stay in force until the next GEOALG statement.
MODE	Settings stay in force until the next MODE statement.
PRCOMP	ON/OFF statement.

Statement	Status
PTBUFF	ON/OFF statement.
SCNMOD	ON/OFF statement.
SCNSET	Settings stay in force until the next SCNSET statement.
SNSET	Settings stay in force until the next SNSET statement.
TECOMP	Settings stay in force until the next TECOMP statement.
UNITS	Settings stay in force until the next UNITS statement.
WKPLAN	Settings stay in force until the next WKPLAN statement.

5.4.5.5 Measurement

When measurement statements refer to a multi-point feature that will be measured from more than one carriage, the MEAS and RMEAS statements must be programmed in a common section. The CRSLCT statements used within the measurement block specify the carriage by which GOTO, PAMEAS and PTMEAS statements are executed.

The execution mode of the subsections programmed in a carriage common section will be performed according to the current CRMODE status. A measurement block included in a carriage common section, can be considered as if it were a program itself. This means that in a measurement block all the carriage specific sections and carriage common sections are allowed.

All the carriage specific data modifications that take place in a measurement block included in a carriages common section, will permanently change those data values; they will be used as the default data for any further execution of carriage sections of the same carriage.

5.4.5.6 Branching

Conditional blocks must begin and terminate in the same carriage or common section.

The JUMPTO statement must address a jumptarget included in the same section. As an exception, a JUMPTO statement in a common section may address a jumptarget in another common section.

5.4.5.7 Output

A multiple carriage DME will provide DMIS or vendor format output in the same sequence as the part program, independent from the parallel execution.

5.4.5.8 Coordinate systems for a multiple carriage system

When a coordinate system is recalled or created within a carriage common section, subsequent carriage specific sections will begin execution with the last coordinate system established in the preceding common section.

When executed within a carriage specific section, any DMIS statement that changes the coordinate system effects only that specific carriage.

Upon entering a common section, no implicit coordinate system recall occurs. If carriage common DMIS statements are to be executed requiring a particular coordinate system, the DMIS program must either recall a previously defined coordinate system or create a new fully constrained coordinate system. If no DMIS statements are executed that create or recall a coordinate system within the common section, subsequent carriage specific sections will resume execution with the same carriage specific coordinate system they had previously.

5.4.6 Motion control

5.4.6.1 Motion statement

GOHOME and GOTO are motion statements that are issued alone or as part of a measurement sequence. The GOTO statement can also be grouped two or more, into a motion sequence within a GOTARG…ENDGO block. The GOTARG statement is similar in structure to the MEAS statement.

5.4.6.2 Motion sequence

The motion sequence is a program block that begins with the GOTARG statement, contains two or more GOTO statements and ends with the ENDGO statement. The GOTARG statement signifies that a non-measurement motion sequence is to be performed and defines the endpoint to which the DME will travel. The GOTO statements following the GOTARG statements provide the sequence of moves to arrive at the endpoint. The last motion statement before the ENDGO must have the same endpoint as the GOTARG statement. GOTO statements that move sensor axes, i.e. specify HEADCS or PCS as parameters are not permitted in a motion sequence.

5.4.6.2.1 MODE and the motion sequence

As directed by a MODE statement, some motion sequences may be performed in automatic mode, some in programmed mode, and some in manual mode. The decision is made by the DME each time a GOTO or GOTARG statement is encountered.

In automatic mode, the DME is responsible for supporting continuous motion within the multi-point motion specified. In this case, the DME calculates its own path to the endpoint. In programmed mode, the GOTO statements are executed. The motion sequence is complete when the ENDGO statement is encountered. In manual mode, the operator moves the DME manually to the endpoint. Refer to the GOTARG statement for further explanation.

5.4.6.2.2 Motion sequence example

The motion sequence program block begins with the GOTARG statement and ends with the ENDGO statement. The only valid statement within a GOTARG motion sequence is a GOTO. The following is an example of a GOTARG…ENDGO block to drive the sensor to a point at -20.5,35.5,107.5:

```
GOTARG/-20.5,35.5,107.5
GOTO/20.5,35.5,120
GOTO/-20.5,35.5,120
GOTO/-20.5,35.5,107.5
ENDGO
```

5.4.6.2.3 FLY motion

Fly mode provides for the ability to move in a continuous manner, without necessarily passing through GOTO or measure approaching points.

The relative distance away from the GOTO points is controlled by a radius value indicating the maximum limit allowed to start the trajectory approximation. Refer to (Figure 19 — FLY, radius example).

Figure 20 — FLY mode example, shows the different behaviour of the CMM when FLY mode is set OFF or ON executing the GOTO sequence below:

```
$$ G1
GOTO/10,0,0
$$ G2
GOTO/10,10,0
$$ G3
GOTO/0,10,0
```

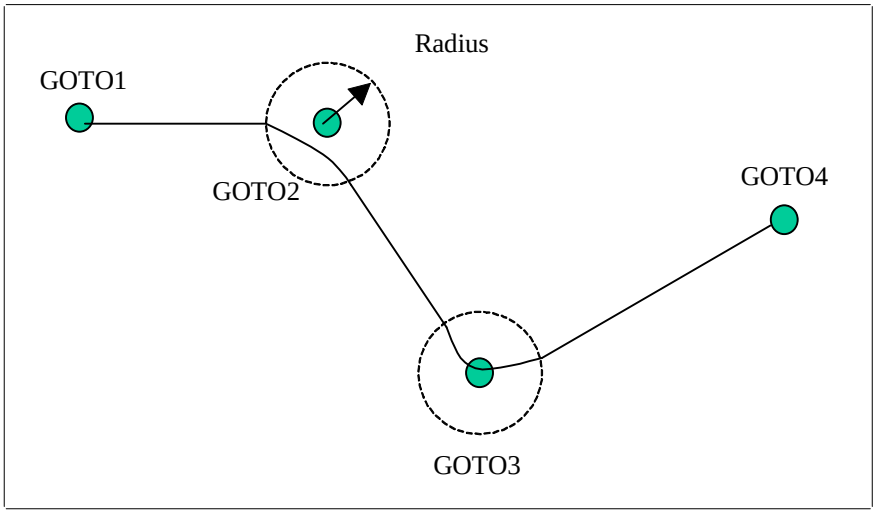


Figure 19 — FLY, radius example

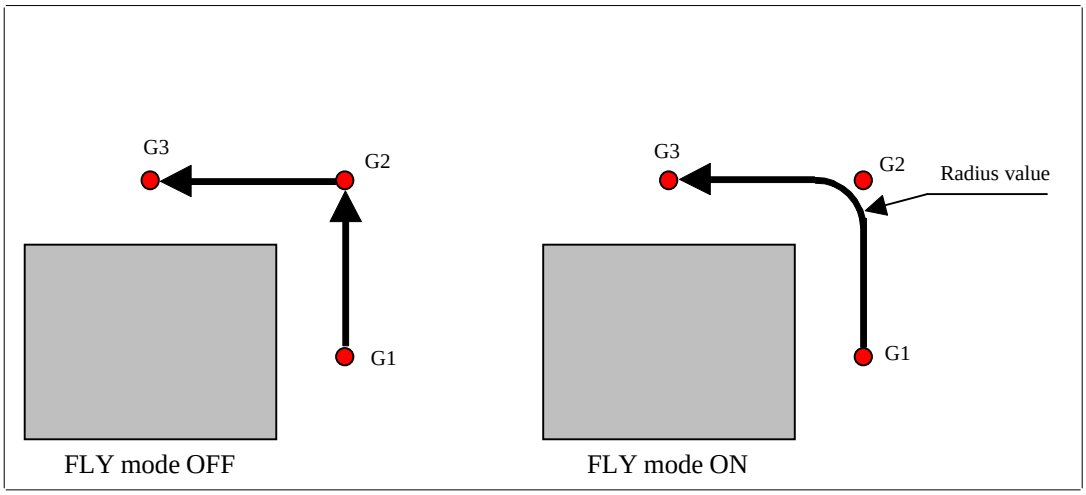


Figure 20 — FLY mode example

5.4.6.2.4 Sensor motion

In most cases, sensor motion is as a consequence of other motion of the hardware, e.g. the movement of a CMM frame. However, with hardware such as a continuously indexable probe head, or an inspection robot, it is possible and desirable to specify motion that also includes motion of the sensor hardware, perhaps to explicitly align the sensor with a feature prior to inspection.

The sensor motion can be specified as either explicit motion of the sensor axes (i.e. as rotations in the local head coordinate system), or relative to the part coordinate system.

Example 1

```
GOTO/10,10,10,HEADCS,-45.0,37.6
```

In example 1, the GOTO statement requires that the location of the probe tip (or laser sensor equivalent) move to the location (10,10,10). In addition, the HEADCS parameter indicates that the sensor axes should also move, in this case to (-45.0,37.6). The assumption is that the sensor has two moveable axes: if not this results in an error condition. Such explicit reference to the sensor axes is very useful (and is a commonly accepted way of expressing sensor motion) in performing absolute movement of the sensor prior to a parking procedure.

Example 2

```
GOTO/CART,10,10,10,PCS,45,37.6,180
```

In example 2, the GOTO statement requires that the location of the probe tip (or laser sensor equivalent) again move to the location (10,10,10). In this case the PCS parameter gives an absolute alignment of the sensor relative to the part coordinate system. The three values specified for the orientation are the ZYZ convention Euler angles that define the transformation between part coordinates and local sensor coordinates (refer to clause 5.4.1). While not as readily understandable as expressing the values in absolute head angles, this form of expression is consistent regardless of the orientation of the part relative to machine coordinates, so long as the part coordinate systems is created in the same way (i.e. by measuring the same datum features). This makes programs portable between DME configuration, with respect to both machine frame (CMM, robot, etc) and probing system (continuous head, laser, etc). This convention of expressing multi-axis motion is useful when aligning the sensor to a feature axis prior to measurement.

Note: the choice of the ZYZ Euler convention is not arbitrary. The final "Z" represents a rotation around the sensor Z axis (as well as its meaning in the Euler convention). As this is the value the user is most likely to want to change, e.g. to rotate a laser sensor plane, it is very useful that it is explicit. The "ZY" choice (as opposed to "XY", "ZX" or "YX") is analogous to the HEADCS convention, where the first rotation is (usually) around the sensor Z axis and the second rotation is around the sensor Y axis. However, this analogy should be used advisedly because the HEADCS convention will tend to be sensor specific.

Example 3

```
GOTO/HEADCS,-45.0,37.6
```

Example 3 shows a GOTO statement where the target location is not specified. This implicitly means that the motion is to involve all axes of the machine, i.e. frame and sensor axes, so as to maintain the location of the tip at the current location, but to orient the sensor according to the explicit sensor axis angles.

Note: the multi-axis motion described in example 3, is NOT the same as movement of the sensor axes alone which is considered to be a sensor selection and should be actioned using the SNSLCT statement.

5.4.7 Measurement control

The measurement sequence is a program block which instructs the DME to perform the measurement of a feature. The sequence begins with the MEAS or RMEAS statement and ends with the ENDMES statement. It may be executed in several ways, depending on the mode in which the machine executes the sequence. The type of measurement (for example, MEAS/CIRCLE or RMEAS/CIRCLE) must match the type of the defined nominal (for example, F(lname)=FEAT/CIRCLE).

Feature measurements can be performed by either single point measurement, or scanning measurement. With single point measurement, the individual measurement points for a feature are either defined by PTMEAS statements when in programmed mode, or by the DME's internal algorithm when in automatic or manual mode. With scanning measurement, the path that the scan will follow is defined by the PATH statement and then referenced by the PAMEAS statement when in programmed mode, or by the DME's internal algorithm when in automatic mode.

5.4.7.1 Substitute data fitting and scan filtering algorithms

To provide further control of your inspection results a substitute feature data fitting algorithm may be applied. When the GEOALG statement is executed, the algorithm used will not remove any measured data from a feature actual. The GEOALG statement is modal and will effect the results for the feature type in the GEOALG statement until another GEOALG statement for that feature type is executed.

EXAMPLE

```
DECL/INTGR,Nbr_points
MODE/AUTO,PROG,MAN
GEOALG/CIRCLE,DEFALT
```

```
F(Cir_all_data)=FEAT/CIRCLE,INNER,CART,100.00,50.00,100.00,0.0,0.0,1.0,300.0
MEAS/CIRCLE,F(Cir_0),634
PTMEAS/-50.0,50.0,98.0,1.0,0.0,0.0
...
PTMEAS/250.0,50.0,98.0,-1.0,0.0,0.0
...
PTMEAS/100.0,-100.0,98.0,1.0,0.0,0.0
...
PTMEAS/100.0,200.0,98.0,-1.0,0.0,0.0
...
ENDMES
$$ The circle was initially measured with the default GEOALG, for circles, on
$$ this system in effect.  This way we have all measured information available
$$ in the point buffer, if we need to use it for other evaluations.
$$
$$ We will now construct a new circle using an algorithm that will remove any
$$ erroneous data that may have been collected, and then apply a 500 UPR filter
$$ to the measured data.
$$
GEOALG/CIRCLE,LSTSQR,ELIMINATE,STDDEV,0.0013,FILTER,CIRCULAR,500
$$
$$ start by defining another nominal circle the same as the original circle.
$$
F(Cir_filtered)=FEAT/CIRCLE,INNER,CART,100.00,50.00,100.00,0.0,0.0,1.0,300.0
$$
$$ now construct a new circle from the previously measured circle while
$$ the external algorithm is in effect.
$$
CONST/CIRCLE,F(Cir_filtered),TR,FA(Cir_all_data)
$$
$$ Now set the GEOALG for circles back to the system default.
GEOALG/CIRCLE,DEFALT
```

If the DME does not support the requested substitute feature algorithm, then an error message is sent to the appropriate output devices currently opened. The error message will notify the users that the feature was not data fitted with the designated substitute feature algorithm and that the feature will be fitted with the default algorithm.

Scan filtering algorithms are defined with the GEOALG statement and apply to all feature actuals of the defined feature type until another GEOALG statement for that feature type is executed. Data collected during measurement may be filtered, but filtering is not required.

An algorithm referenced by the GEOALG statement always contains an algorithm to use to construct a substitute feature from the individual point information. It may optionally contain a filter that will be applied to determine the individual point information that corresponds to FA (). If a new feature is constructed from a feature actual after execution of a GEOALG statement for that feature type and with filtering parameters, the new feature's individual point information are filtered and may not be exactly the same individual point information as the feature from which they were constructed.

5.4.7.2 MODE and the measurement sequence

Depending on the MODE statement, the DME may decide whether the feature is to be measured automatically or by using the programmed measurement sequence. With a single MODE statement, some features may be measured in automatic mode, some in programmed mode, and some in manual mode. The DME makes the decision each time a MEAS or RMEAS statement is encountered.

The DME can perform in any one of three modes. When in manual mode, the DME ignores the statements after the MEAS or RMEAS statement (up to and including the ENDMES statement). In this mode, an operator manually moves the DME and the sequence is considered complete when the operator has taken the required number of point measurements. In automatic mode, the DME again ignores the statements after the MEAS or RMEAS statement. In this mode, the DME uses its own internal algorithm to perform the measurement of the feature. In

programmed mode, the DME follows the statements after the MEAS or RMEAS statement to perform the measurement. The measurement sequence is complete when the ENDMES statement is encountered.

The statements after the MEAS or RMEAS statement are optional. If it is known that all DMEs on which the program will be executed have algorithms for the feature to be measured, or if all DMEs will execute the program in manual mode, then only the MEAS or RMEAS and ENDMES statements are needed. If, however, the sequence is to be performed under servo control by a DME that does not have an algorithm for the feature then the measurement sequence must be programmed

5.4.7.3 Allowable statements

Various DMIS statements can be used in a measurement sequence. When machine parameters are changed in a measurement sequence, care must be taken to ensure that different DMEs executing the measurement in different modes will all be in a proper state after the measurement is complete. A list of statements that may be issued between the MEAS, or RMEAS and ENDMES statements is shown in (Table 14 — DMIS statements allowed in a measurement block):

Table 14 — DMIS statements allowed in a measurement block

(jumptarget)	DMESW	FEDRAT	OBTAIN	SAVE/SA(lname)
ACLRAT	DO	FINPOS	PAMEAS	SCNSET
ASSIGN	ELSE	FROM	PTMEAS	SELECT
BADTST	ENDCAS	GOHOME	RAPID	SNSET
CASE	ENDDO	GOTARG	RECALL/S(lname)	SNSLCT
CRSLCT	ENDGO	GOTO	RECALL/SA(lname)	TEXT
CZSLCT	ENDIF	IF	ROTAB	VALUE
DFTCAS	ENDSEL	INCLUD	ROTSET	WKPLAN
DMEHW	ERROR	JUMPTO	SAVE/S(lname)	

If the INCLUD statement is used to call external code within a measurement block, the external code must contain only statements allowed within a MEAS, or RMEAS...ENDMES block. IF...ENDIF, DO...ENDDO, and SELECT...ENDSEL blocks must be fully contained within the MEAS, or RMEAS...ENDMES block. If the JUMPTO statement is used in a measurement block, the (jumptarget) to which program control will be transferred must be within the same MEAS, or RMEAS...ENDMES block. Program control cannot be transferred to a (jumptarget) inside a measurement block from outside that measurement block.

5.4.7.4 Changing sensors

Sensors can be changed with the SNSLCT statement during a programmed measurement sequence. If this is done, and if automatic mode is active, the DME, which tries to execute the sequence in automatic mode, will not have information pertaining to the sensor to be selected, and must determine this for itself. Care should be exercised when issuing a SNSLCT statement in automatic mode. A SNSLCT statement issued in a measurement sequence will not be recognized in automatic mode. A safer programming practice would be to issue a MODE/PROG statement for a measurement sequence in which the sensor will be changed. This is not, however, required by DMIS.

5.4.7.5 Unsupported feature results

If a DME encounters a CALIB, MEAS or RMEAS statement for a feature which it does not support in automatic mode and if a programmed measurement sequence is given, the DME will measure the feature using the programmed sequence and output point data for the feature measurement.

For RAWDAT output, each measured data point is preceded with a slash and followed with a carriage return and line feed. The feature definition RAWDAT output is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

If, for example, a particular DME did not support the feature definition for a sphere, and a MEAS/SPHERE statement was encountered with a programmed measurement sequence, the DME would follow the sequence and output data in a format similar to the following:

```
FA(SPH1)=FEAT/SPHERE,RAWDAT
/x,y,z
/x,y,z
/x,y,z
/x,y,z
ENDAT
```

In this case, the label SPH1 is passed to the output file. RAWDAT indicates that the feature type is not supported, but point data is being output. The uncompensated x,y,z data are the point data sampled in the programmed measurement sequence. The ENDAT statement signifies the end of the data stream.

5.4.7.6 Data access

Feature nominal target point data defined by PTMEAS statements and the resulting actual point data can be referenced if the corresponding feature was measured with point buffer on. Individual point information resulting from PAMEAS statements can also be accessed if the corresponding feature was measured with point buffer on. Point data can be accessed through its corresponding feature via the option subscript syntax F(lname)[n] and FA(lname)[n]. This capability and syntax are described in clause 5.3.2.6.

5.4.7.7 Scanning measurement control

There are two methods of defining scan measurement using the PAMEAS statement or by using a combination of the SCNMOD and PTMEAS statements.

Scanning a feature refers to the inspection of points on the part other than those which are explicitly called for in the feature MEAS or RMEAS…ENDMES block. Scanning measurement is done using point sampling rates and/or point spacing that can be controlled by the programmer or determined automatically by the DME.

Scanning is controlled by SCNMOD/ON and OFF when the scan is directed by PTMEAS statements and PAMEAS when using MODE/PROG.

Control of scanning a feature is also dependent on the mode in which the DME is operating. When in automatic or manual mode the statements within a measurement block are not executed, so control of a scan is dependent on the DME's internal algorithm. When in programmed mode control of a scan is dependent on the statements within a measurement block.

5.4.7.7.1 Definition

Scanning sampling rate parameters are defined using the SCNSET statement, which allows for a variety of point sampling options. The filtering options used while scanning are defined using the GEOALG statement, which also allows for point elimination. The path used for a scan is defined with the PATH statement, and is applied with the PAMEAS statement.

5.4.7.7.2 Activation and control

Scanning is activated and deactivated explicitly with the SCNMOD/ON and SCNMOD/OFF when the scan is directed by PTMEAS statements. The setting specified by the SCNMOD statement is modal and applies to the measurement of all subsequent features in the program until the setting is modified by another SCNMOD statement. When the DME is operating in automatic or manual mode and scanning is active, control of the scan is performed by the DME's internal algorithm. Scanning measurement in programmed mode is explicitly controlled by

programming multiple PTMEAS statements while SNCMOD is ON and by referencing one or more defined paths with the PAMEAS statement when SCNMOD is OFF.

When the scan is directed by PTMEAS statements, other non-measurement statements indicate an interruption in the scan. There must be a minimum of 2 sequential PTMEAS statements in any scan segment within a measurement block, the first of which is the beginning point of the scan and the last is the ending point of the scan. Any non-measurement statements, such as GOTO or ROTAB statements, within a measurement block are not a part of the scan. The APPRCH, RETRCT and CLRSRF parameters of the SNSET statement apply to the first and last PTMEAS statements in any scan segment.

Example: an example of incorrect scanning syntax:

```
SCNMOD/ON
SCNSET/DRAG,.1
MODE/PROG,MAN
F(CIR_11)=FEAT/CIRCLE,INNER,CART,-20.5,35.5,107.5,0.0,0.0,1.0,10
MEAS/CIRCLE,F(CIR_11),4
PTMEAS/CART,-15.5,35.5,107.5,-1,0,0
GOTO/-20.5,35.5,107.5
PTMEAS/CART,-20.5,40.5,107.5,0,-1,0
PTMEAS/CART,-20.5,30.5,107.5,0,1,0
ENDMES
SCNMOD/OFF
```

In the previous example the measure block contains a scan segment that does not contain 2 sequential PTMEAS statements. The GOTO statement must interrupt the scan, and a single PTMEAS statement does not define a scan segment. This will produce an error.

When the scan is directed by at least one PAMEAS statement, each PAMEAS statement defines a scan segment. Each PAMEAS statement must contain references to previously defined scan paths. The defined scan paths explicitly define the scans to be used for measurement of the feature. Each PAMEAS statement may also contain parameters to direct rotary table motion or sensor/wrist motion that explicitly defines the scan to be used for measurement of the feature. The APPRCH and RETRCT parameters of the SNSET statement apply to the first and last measurement points in any scan segment.

5.4.7.7.3 Continuous path scanning

The defined behaviour when scanning a feature using multiple PAMEAS statements is to treat each PAMEAS statement as representing a discrete scan segment, i.e. at the end of each segment scanned, the probe should back off to the specified RETRCT distance and move to the position determined using the current APPRCH value before scanning the next segment.

Example 1

```
MODE/PROG,MAN
F(PLN_1)=FEAT/PLANE,CART,0.0,0.0,0.0,0.0,0.0,1.0
P(Scan_PLN1_1)=PATH/ARC,CART,0.0,20.0,0.0,0.0,0.0,1.0,20.0,-180.0,90.0,0,1,0
P(Scan_PLN1_2)=PATH/LINE,CART,-20.0,20.0,0.0,-20.0,40.0
MEAS/PLANE,F(PLN_1),3
PAMEAS/P(Scan_PLN1_1)
PAMEAS/P(Scan_PLN1_2)
ENDMES
```

Example 1 measures a plane using two scan segments: an arc, followed by a line. Because the scans are specified using two PAMEAS statements, they are assumed to be non-contiguous, i.e. the probe will retract at the end of the first segment.

Example 2

```
MODE/PROG,MAN
F(PLN_1)=FEAT/PLANE,CART,0.0,0.0,0.0,0.0,0.0,1.0
```

```
P(Scan_PLN1_1)=PATH/ARC,CART,0.0,20.0,0.0,0.0,0.0,1.0,20.0,-180.0,90.0,0,1,0
P(Scan_PLN1_2)=PATH/LINE,CART,-20.0,20.0,0.0,-20.0,40.0
MEAS/PLANE,F(PLN_1),3
PAMEAS/P(Scan_PLN1_1),P(Scan_PLN1_2)
ENDMES
```

Example 2 measures a plane using the same two scan segments as example 1. However, because the scans are specified using a single PAMEAS statement they are assumed to be contiguous, i.e. the probe will scan the two segments as if they were a single scan, without break.

5.4.7.8 The MEAS statement

The MEAS statement signifies that a measurement of a feature is to be performed. The measurement sequence begins with a MEAS statement and ends with the ENDMES statement. The MEAS…ENDMES block may contain statements found in (Table 14 — DMIS statements allowed in a measurement block).

In the following example, four points are taken in order to measure a circle called "CIR_11":

Example 1

```
MODE/PROG,MAN
F(CIR_11)=FEAT/CIRCLE,INNER,CART,-20.5,35.5,107.5,0.0,0.0,1.0,10
MEAS/CIRCLE,F(CIR_11),4
PTMEAS/CART,-15.5,35.5,107.5,-1,0,0
PTMEAS/CART,-25.5,35.5,107.5,1,0,0
GOTO/-20.5,35.5,107.5
PTMEAS/CART,-20.5,40.5,107.5,0,-1,0
PTMEAS/CART,-20.5,30.5,107.5,0,1,0
ENDMES
```

In example 1 the MEAS statement signifies that a circle called CIR_11 is to be measured by taking four point measurements. The following statements specify that point measurements will be used, the location of the points to be taken, and an intermediate move. The ENDMES statement signifies the end of the sequence.

Example 2

```
SCNMOD/ON
SCNSET/DRAG,.1
MODE/PROG,MAN
F(CIR_11)=FEAT/CIRCLE,INNER,CART,-20.5,35.5,107.5,0.0,0.0,1.0,10
MEAS/CIRCLE,F(CIR_11),4
PTMEAS/CART,-15.5,35.5,107.5,-1,0,0
PTMEAS/CART,-25.5,35.5,107.5,1,0,0
GOTO/-20.5,35.5,107.5
PTMEAS/CART,-20.5,40.5,107.5,0,-1,0
PTMEAS/CART,-20.5,30.5,107.5,0,1,0
ENDMES
SCNMOD/OFF
```

Example 2 is the same as example 1 above, except that the SCNMOD/ON statement explicitly activates scanning. Because the system is operating in programmed mode and scanning has been activated, the system will scan two arcs in the circle. The first circular arc will begin at the point defined in the first PTMEAS statement, and end at the point defined by the second PTMEAS statement. The system will retract then move to the point defined by the GOTO statement. The system will then scan the second circular arc, beginning at the point defined by the third PTMEAS statement, and ending at the point defined by the fourth PTMEAS statement. The system will collect measurement data based on the parameters set with the SCNSET statement and the number of measurement points in the MEAS statement is ignored.

Example 3

```
MODE/PROG,MAN
```

```
SCNSET/DRAG,.1
F(CIR_11)=FEAT/CIRCLE,INNER,CART,-20.5,35.5,107.5,0.0,0.0,1.0,10
P(Scan_CIR_11)=PATH/ARC,CART,-20.5,35.5,107.5,0.00,0.00,1.00,5.00,0.0,363.0
GOTO/-20.5,35.5,115
GOTO/-20.5,35.5,107.5
MEAS/CIRCLE,F(CIR_11),4
PAMEAS/P(Scan_CIR_11)
ENDMES
```

Example 3 measures the same feature as example 1 and 2 above, except that it contains a PAMEAS statement and does not contain PTMEAS statements. The scan path is referenced with the PAMEAS statement within the measurement block, and the scan is explicitly defined with the PATH statement. Because the system is operating in programmed mode, the explicit scan path definition will be followed to measure the feature. If a scan path is to contain two or more path segments that are to be scanned contiguously, i.e. without retracting from the surface between the scan segments, then they should both be referenced by a single PAMEAS statement:

Example 4

```
PAMEAS/P(Scan_ARC_1),P(Scan_LINE_1),P(Scan_ARC_2)
```

Example 4 shows PAMEAS statement that references several PATH statements. In this example, the scan should begin at the start of the first segment, continue into the second segment without leaving the surface and continue with the third segment (again remaining in contact with the surface), before finally retracting from the surface at the end of the third segment. If the scan paths are not defined to be contiguous, i.e. the end of one path is not coincident with the start of the following path, then an error condition occurs.

5.4.7.9 The RMEAS statement

The RMEAS statement signifies that a relative measurement of a feature is to be performed. The measurement sequence begins with an RMEAS statement and ends with the ENDMES statement. The RMEAS…ENDMES block may contain statements found in (Table 14 — DMIS statements allowed in a measurement block).

In the following examples, four points are taken in order to measure a circle called "RCIR_12":

Example 1

```
MODE/PROG,MAN
F(RPOINT_1)=FEAT/POINT,CART,-10.5,35.5,107.5,0.0,0.0,1.0
MEAS/POINT,  F(RPOINT_1),1
PTMEAS/CART,-10.5,35.5,107.5,0.0,0.0,1.0
ENDMES
F(RCIR_12)=FEAT/CIRCLE,INNER,CART,-20.5,35.5,107.5,0.0,0.0,1.0,10
RMEAS/CIRCLE,F(RCIR_12),4,FA(RPOINT_1)
PTMEAS/CART,-15.5,35.5,107.5,-1,0,0
PTMEAS/CART,-25.5,35.5,107.5,1,0,0
GOTO/-20.5,35.5,107.5
PTMEAS/CART,-20.5,40.5,107.5,0,-1,0
PTMEAS/CART,-20.5,30.5,107.5,0,1,0
ENDMES
```

In example 1 the RMEAS statement signifies that a circle called "RCIR_12" is to be measured by taking four point measurements relative to a point called "RPOINT_1". The following statements specify that point measurements will be used, the location of the points to be taken, and an intermediate move. The nominal values for the circle measurement points is programmed, but the location that the machine measures the points will be adjusted relative to the deviation of the reference point as defined in the RMEAS (input format 1) statement definition. The ENDMES statement signifies the end of the sequence.

Example 2

```
MODE/PROG,MAN
F(RPOINT_1)=FEAT/POINT,CART,-10.5,35.5,107.5,0.0,0.0,1.0
```

```
MEAS/POINT,  F(RPOINT_1),1
PTMEAS/CART,-10.5,35.5,107.5,0.0,0.0,1.0
ENDMES
SCNMOD/ON
SCNSET/DRAG,.1
F(RCIR_12)=FEAT/CIRCLE,INNER,CART,-20.5,35.5,107.5,0.0,0.0,1.0,10
RMEAS/CIRCLE,F(RCIR_12),4,FA(RPOINT_1)
PTMEAS/CART,-15.5,35.5,107.5,-1,0,0
PTMEAS/CART,-25.5,35.5,107.5,1,0,0
GOTO/-20.5,35.5,107.5
PTMEAS/CART,-20.5,40.5,107.5,0,-1,0
PTMEAS/CART,-20.5,30.5,107.5,0,1,0
ENDMES
SCNMOD/OFF
```

Example 2 is the same as example 1 above, except that the SCNMOD/ON statement explicitly activates scanning. Because the system is operating in programmed mode and scanning has been activated, the system will scan two arcs in the circle. The first circular arc will begin at the point defined in the first PTMEAS statement, and end at the point defined by the second PTMEAS statement. The system will then move to the point defined by the G OTO statement. The system will then scan the second circular arc, beginning at the point defined by the third PTMEAS statement, and ending at the point defined by the fourth PTMEAS statement. The system will collect measurement data based on the parameters set with the SCNSET statement and the number of measurement points in the RMEAS statement is ignored. All measurements are relative to feature FA(RPOINT_1) as required by the RMEAS statement.

Example 3

```
MODE/PROG,MAN
F(RPOINT_1)=FEAT/POINT,CART,-10.5,35.5,107.5,0.0,0.0,1.0
MEAS/POINT,  F(RPOINT_1),1
PTMEAS/CART,-10.5,35.5,107.5,0.0,0.0,1.0
ENDMES
F(RCIR_12)=FEAT/CIRCLE,INNER,CART,-20.5,35.5,107.5,0.0,0.0,1.0,10
P(Scan_RCIR_12)=PATH/ARC,CART,-20.5,35.5,107.5,0.00,0.00,1.00,5.00,0.0,363.0
GOTO/-20.5,35.5,115
GOTO/-20.5,35.5,107.5
SCNSET/DRAG,.1
RMEAS/CIRCLE,F(RCIR_12),4,FA(RPOINT_1)
PAMEAS/P(Scan_RCIR_12)
ENDMES
```

Example 3 measures the same feature as example 1 and 2 above, except that it contains a PAMEAS statement and does not contain PTMEAS statements. The scan path is referenced with the PAMEAS statement within the measurement block, and the scan is explicitly defined with the PATH statement. Because the system is operating in programmed mode, the explicit scan path definition will be followed to measure the feature. The circle called "RCIR_12" is to be measured relative to a point called "RPOINT_1". The PAMEAS statement references a previously defined path "P(Scan_RCIR_12)". The nominal values for the circle measurement points is programmed, but the location that the machine measures the points will be adjusted relative to the deviation of the reference point as defined in the RMEAS (input format 1) statement definition. The ENDMES statement signifies the end of the sequence.

5.4.7.10 Sensor motion during measurement

Depending upon the hardware configuration, it is possible and desirable to change the orientation, as well as the position, of a sensor during measurement. This may be to take individual tactile touches, or during laser scanning, for example.

5.4.7.10.1 Individual tactile measurement

When using a continuously indexable head to take tactile touches, it is possible to take a touch in at least three combinations of motion: using the CMM frame (say) alone; using the probe axes alone; or using a combination of machine frame and head axes.

Example 1

```
MEAS/CIRCLE,F(CIR1),4
PTMEAS/CART,20,0,0,-1,0,0
PTMEAS/CART,0,20,0,0,-1,0
PTMEAS/CART,-20,0,0,1,0,0
PTMEAS/CART,0,-20,0,0,1,0
ENDMES
```

Example 1 shows the default option when specifying a PTMEAS statement: that the motion produced during measurement will involve the movement of the CMM frame alone.

Example 2

```
MEAS/CIRCLE,F(CIR1),4
PTMEAS/CART,20,0,0,-1,0,0,HEADCS,0,0,HEADTOUCH
PTMEAS/CART,0,20,0,0,-1,0,HEADCS,90,0,HEADTOUCH
PTMEAS/CART,-20,0,0,1,0,0,HEADCS,180,0,HEADTOUCH
PTMEAS/CART,0,-20,0,0,1,0,HEADCS,-90,0,HEADTOUCH
ENDMES
```

Example 2 shows the option of using the option of using the sensor axes alone to perform measurement.

Example 3

```
MEAS/CIRCLE,F(CIR1),4
PTMEAS/CART,20,0,0,-1,0,0,HEADCS,0,0,ALLAXESTOUCH
PTMEAS/CART,0,20,0,0,-1,0,HEADCS,90,0,ALLAXESTOUCH
PTMEAS/CART,-20,0,0,1,0,0,HEADCS,180,0,ALLAXESTOUCH
PTMEAS/CART,0,-20,0,0,1,0,HEADCS,-90,0,ALLAXESTOUCH
ENDMES
```

Example 3 shows the option of using the option of using the sensor axes and CMM axes in combination to perform measurement.

Example 4

```
MEAS/LINE,F(LINE1),4
PTMEAS/CART,20,0,-5,1,0,0,EADTOUCH
PTMEAS/CART,30,0,-5,1,0,0,HEADTOUCH
PTMEAS/CART,40,0,-5,1,0,0,HEADTOUCH
PTMEAS/CART,50,0,-5,1,0,0,HEADTOUCH
ENDMES
```

Example 4 shows the use of PTMEAS and HEADTOUCH or ALLAXESTOUCH without HEADCS and/or PCS.

The choice of motion may be dependent upon the upon the speed of inspection as opposed to predictability of contact point. When using the HEADTOUCH option, the trajectory of the probe tip during the inspection will not be linear and will most likely follow an arc. Therefore the “nominal” contact point will be different to the value specified in the PTMEAS statement: the difference will depend upon the specific sensor motion as well as the current value for APPRCH (as specified in the SNSET statement). However, it is likely that motion of the sensor axes alone will be the quickest way to take the point.

5.4.7.10.2 Tactile scanning

When performing a tactile scan using a continuously indexable sensor, for example, it is often critical to be able to orient the probe at the points along the scan. This can be achieved either by explicitly providing the angles required at each position in the scan, or by specifying the relative orientation of the probe to the part. This second approach is achieved by specifying three angles of rotation relative to the part coordinate system, using the Euler ZYZ convention (refer to clause 5.4.1).

5.4.7.10.3 Non-contact scanning

When scanning with a laser probing system mounted on an inspection robot, for example, it is often critical to be able to orient the probe at the points along the scan. This is most easily achieved by specifying the relative orientation of the probe and, therefore, the laser plane(s), to the part. This is achieved by specifying three angles of rotation relative to the part coordinate system, using the Euler ZYZ convention (refer to clause 5.4.1).

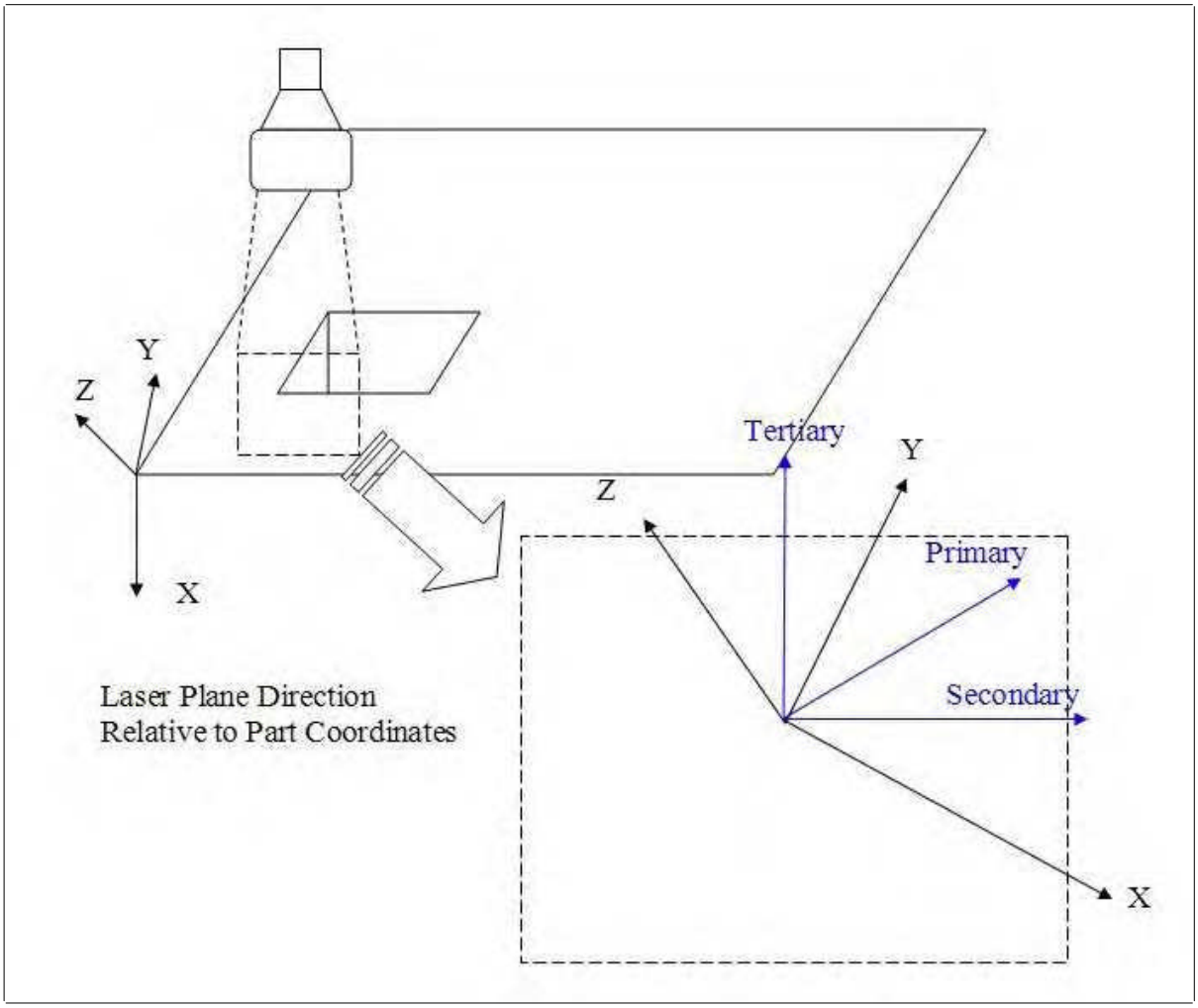


Figure 21 — Sensor / part coordinate system

5.4.8 Axis configuration

Some DME hardware with multiple rotary axes, e.g. inspection robots, some continuously indexable head, can achieve motion or measurement results using multiple configurations of the axes. It is usually desirable to resolve the implicit ambiguity by stating that a DME is to take up a given configuration or pose.

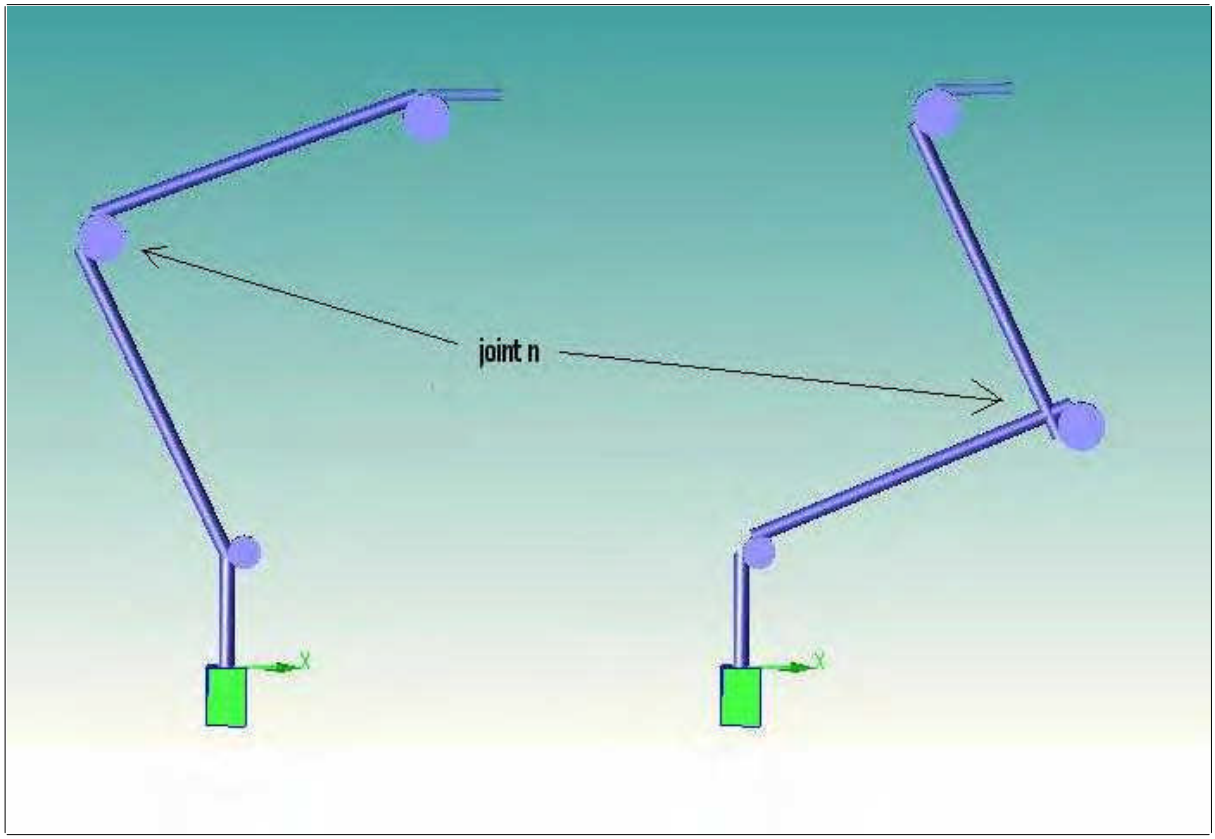


Figure 22 — Two possible robot joint configurations

Figure 22 shows an inspection robot in two possible poses for a given sensor position and orientation. The choice of pose can be made by specifying a property of joint n. In this case an “in-out” option might be appropriate. For example:

```
DMEHW/JOINTCONFIG,'JOINT_N_OUT'
GOTO/100,200,300
```

or:

```
DMEHW/JOINTCONFIG,'JOINT_N_IN'
GOTO/100,200,300
```

In the above code fragments , the interpretation of “JOINT_N” and “IN-OUT” are DME dependent, specifically the number and type of joints.

A similar approach can be taken for a continuously indexable two axis probe head mounted on a CMM frame.

5.5 Characterization file

All vendors are required to have characterization files that describe the extent of DMIS support. The file is an ASCII file, specifying which statements in the DMIS vocabulary are supported, partially supported, or not supported. This file will contain the description of statements for input and output formats of all DMIS statements, as well as other information concerning the system's capabilities or limitations.

5.5.1 Usage

5.5.1.1 Describe the level of support to this standard

Not all vendors will be able to fully support the Dimensional Measuring Interface Standard, nor is that intended. Since DMIS is to be a data exchange standard for a wide variety of vendors, it must be able to simultaneously address the needs of low intelligence controllers as well as those of the more sophisticated inspection equipment.

Rather than limiting the DMIS vocabulary to the lowest common denominator of vendor product capability, the standard defines characterization files, which allow a wide class of vendors to support various subsets of the vocabulary.

5.5.1.2 Intent

To clarify the intent of the characterization file, two possible scenarios for its use in generating DMIS inspection programs are briefly described below. These scenarios are not to be considered part of this standard and are given as illustrative examples only.

Example 1: One scenario for use of the characterization files would be to store them on a CAD system in a "DME library". In this case, the CAD system generates an inspection program in the CAD native format without regard to the target DME. After the program is complete, it is run through an algorithm which decomposes it for the target DME. The same inspection program can be later decomposed for other DMEs. Another method might be to have the operator specify the target DME prior to developing the program. The CAD system could then reference the library while the program was being written and ensure that only DMIS statements supported by all the target DMEs are generated.

Example 2: In scenario 2, the CAD system generates a DMIS program without regard to the target DME. Since both this program and the characterization files are in a neutral format, a decomposition algorithm which is independent of the CAD system can be developed in a standard programming language. The algorithm might be developed by a non-CAD vendor or by the user.

Another use of the characterization file is as a double check of machine capability required to run an inspection program. In this case, a characterization file depicting the requirements of the inspection program can be sent to the DME. The DME checks this file against its own file to determine if it is able to perform the inspection prior to running the program. The standard does not require that a DME have this capability, but it may be supplied as an option desired by users. This again is an implementation issue. What is required by the Dimensional Measuring Interface Standard is that the vendor supplies a characterization file describing its own capabilities.

Refer to (Figure 23 — Use of the DME characterization files), this figure depicts a situation where three DMEs are interfaced to a CAD system using the DMIS format. The DMEs are classified as ranging from a low level intelligence system (DME A) to a high level intelligence system (DME C). The low intelligence system might support very little of the DMIS vocabulary. The high intelligence system, on the other hand, might support much of the DMIS vocabulary. Clearly, the sizes and efficiencies of the DMIS files, which will perform the same inspection on the three types of machines, are quite different. It is desired to create a single DMIS inspection program that will run on all three machines.

Many other scenarios exist. This standard only defines the format for the characterization file. Various CAD vendors, DME vendors and other users are free to develop various tools for their implementation and use. Refer to section 6 for a detailed description of each DMIS statement and to Annex C for the detailed format of the characterization file.

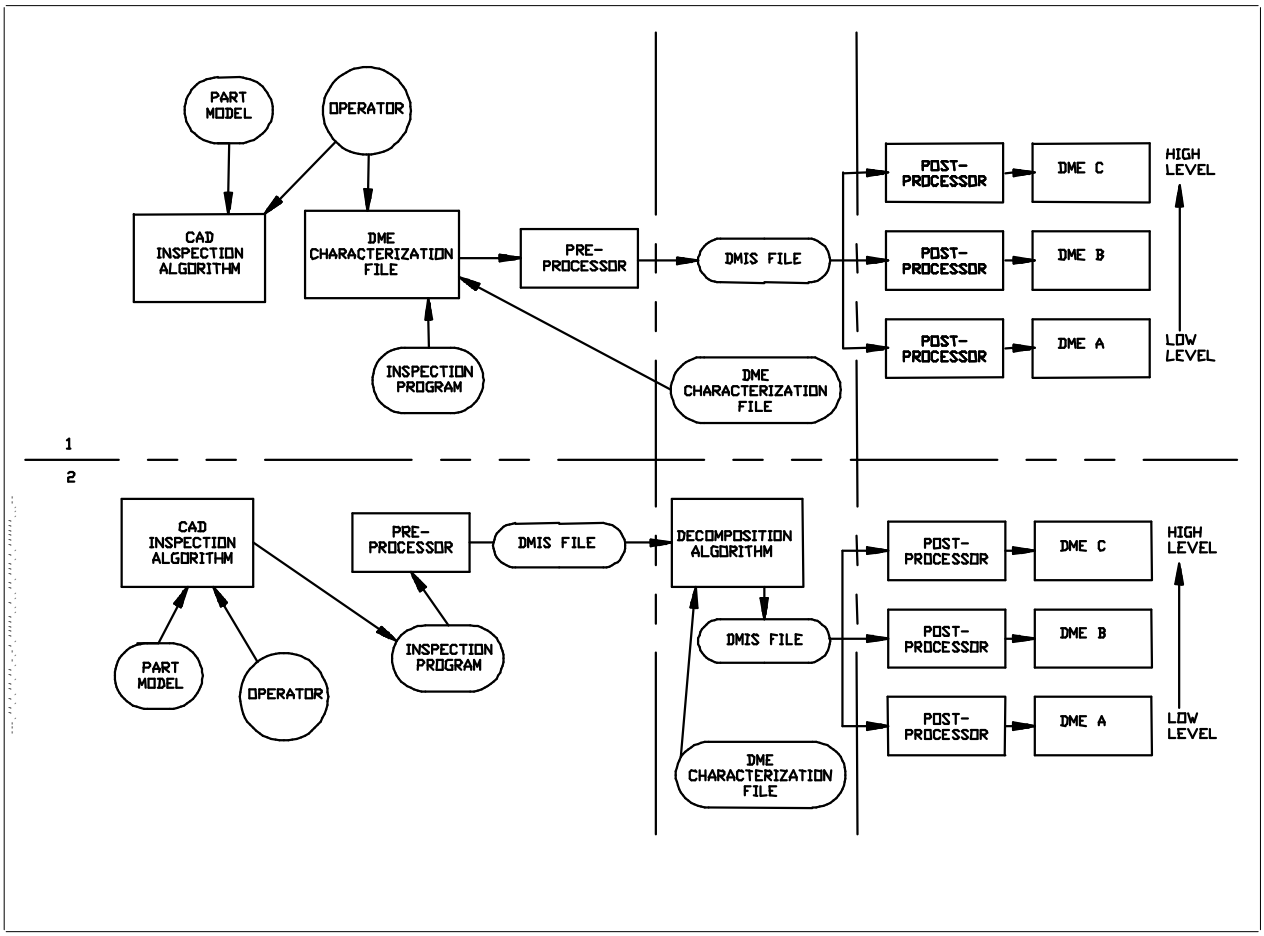


Figure 23 — Use of the DME characterization files

5.5.2 Characterization file format

The following provides definitions and examples for the characterization file development. In this file format:

— CHFILE/INPUT signifies the beginning of the input formats supported by the DME.

— CHFILE/OUTPUT signifies the beginning of the output formats supported by the DME.

— ENDCHF signifies the end of either major section (INPUT or OUTPUT) of the characterization file.

— CHFILx signifies the beginning of characterization file section x.

— ENDCHx signifies the end of characterization file section x.

— $$ signifies the beginning of a comment and must be the first 2 characters at the beginning of a line. Refer to clause 5.1.7.

The characterization file consists of two major sections: input and output. The input and output sections are each blocked by the statements CHFILE/minor word and ENDCHF, where minor words are INPUT or OUTPUT respectively. These major sections are then divided into three minor sections:

a) Supported DMIS statements: supported statements must be contained in the CHFILE/INPUT, CHFIL1...ENDCH1 and CHFILE/OUTPUT, CHFIL1...ENDCH1 sections of the characterization file. Support for

DMIS statements must be specified within those sections in Extended Backus Naur Form (EBNF) as defined in clause 5.5.3 and shown in Annex C.

b) DME specific parameters: any DME specific parameters must be contained in the CHFILE/INPUT, CHFIL2...ENDCH2 section of the characterization file. The following DME specific parameters are required when the indicated DMIS command is designated to be fully functional in the CHFILE/INPUT, CHFIL1 section of the characterization file. If none of these statements are supported, an empty CHFIL2...ENDCH2 section must be included in the characterization file.

```
CHFIL2
ALGDEF

$$ Identifies the algorithms supported by the DME by their code number and description.
$$ Format:

ALGDEF/CODE,n
      n,'text'
      n,'text'
      .
      .
      .

ENDAT
ERROR

$$ Identifies the error codes supported by the DME by their code number and description.
$$ Format:

ERROR/CODE,n
      n,'text'
      n,'text
      .
      .
      .

ENDAT
FILDEF

$$ Identifies the video filters supported by the DME by their code number and description.
$$ Format:

FILDEF/CODE,n
      n,'text'
      n,'text
      .
      .
      .

ENDAT
ENDCH2
```

The above parameters are optional when the indicated command is designated to be non-supported. Additional machine dependent parameters shown in Annex D, section D.1 are also listed here. These additional parameters are optional.

c) User defined options, functions and parameters for special applications. (Refer to Annex D, section D.2). Any user defined options, functions and parameters for special applications must be contained in the CHFILE/INPUT, CHFILn...ENDCHn sections of the characterization file. These sections are optional. Each

section is blocked by the major words CHFILn and ENDCHn where n is the section number and n >=3. Only the sections used need to be present.

EXAMPLE

CHFILE/INPUT
CHFIL1

DMIS Input Grammar Definition (in EBNF format)

See inputFile in Annex C
ENDCH1
CHFIL2

Machine parameters, see Annex D (may be empty)

ENDCH2
CHFILn (one or more optional sections starting with n=3)

User defined options, see Annex D, section D.2

ENDCHn
ENDCHF

CHFILE/OUTPUT
CHFIL1

DMIS Output Grammar Definition (in EBNF format)

See outputFile in Annex C

ENDCH1
ENDCHF

5.5.3 Syntax for CHFIL1...ENDCH1 section

The description of the vendor's level of support of the DMIS specification will be described using the framework of [ISO/IEC 14977:1996 (E) Information technology - Syntactic metalanguage - Extended BNF], using allowable extensions to the standard. Extended BNF (Extended Backus Naur Form, or EBNF) is a widely used notation for the specification of context-free languages as DMIS-input and DMIS-output files. EBNF is restricted for use within the CHFIL1...ENDCH1 sections.

EBNF has four components:

a) A set of tokens, known as terminal symbols: in a DMIS file, the major and minor words represent terminal symbols.

b) A set of non-terminal symbols: Each of these is a variable, which are described by a production.

c) A set of productions: Each production consists of a left hand side which consists of a non-terminal symbol, a right hand side that consists of a sequence of non-terminal and/or terminal symbols, and the "=" symbol that separates the left hand side from the right hand side. Example: **datsetMatrix = TRMATX , c , matrix;**

d) A designation of one of the non-terminal symbols as the start symbol. In the characterization file the start symbol for DMIS-input files is "inputFile" and for DMIS-output files it is "outputFile".

In the characterization file, many productions give several definitions for the non-terminal symbol on the left hand side. This is done by providing alternative right hand sides separated by the symbol "|", which can be read as "or".

EBNF allows the same non-terminal symbol to appear on the left side of several different productions, but that form of providing alternatives is not used in the characterization file.

Standard EBNF uses a number of meta-symbols, not all of which are used in this DMIS standard. The standard EBNF meta-symbols used in this DMIS standard are:

a) = meaning "is defined as",

b) | meaning "or",

c) ' ' apostrophes used to enclose a string of one or more literal characters,

d) ( ) parentheses used to indicate a group,

e) [ ] square brackets used to indicate optional items (see example below),

f) n*[ ] an integer followed by an asterisk, followed by terms in square brackets meaning repetition zero to n times (see example below),

g) , comma meaning the term on the right of the comma follows the term on the left of the comma

h) ; semicolon meaning "end of statement", and

i) (* *) meaning "a comment is contained within".

In addition to the standard use of symbols as just described, the EBNF used for this DMIS standard uses the following conventions.

a) # means "end of DMIS line".

b) A term written entirely in upper case letters (**TRUE**, for example) is a DMIS reserved word. In a DMIS file, these words should appear exactly as they appear in the characterization file, with two exceptions. First, if a word is defined in the characterization file in terms of characters, that is how it should appear in a DMIS file. For example, the characterization file has **TRUE = '.' , ('T'|'t') , ('R'|'r') , ('U'|'u') , ('E'|'e') , '.';**, so that **TRUE** may be spelled ".TRUE." or ".true." or ".truE.", etc. Second, in a DMIS file, the letters in a reserved word not defined in the characterization file may be either lower case or upper case. In EBNF terminology, a DMIS reserved word is a terminal symbol.

c) A term starting with a lower case letter is a non-terminal symbol of the language (for example **stringVal**). Every such term will be found on the left side of a production somewhere in the characterization file.

d) A term starting with an upper case letter followed by a lower case letter (for example, **CharString**) is a non-terminal symbol that has a dummy definition in the characterization file. The allowed "spellings" of these terms are defined in the text of this DMIS standard. The complete list of such terms is: **BoolVarName, CharString, DeclVarName, IntString, IntVarName, LblName, MacroVarName, RealString, RealVarName , StringVarName, VectorVarName.**

Example of the uses of square brackets:

Given a non-terminal symbol of featureLabel containing a value of "MyLabel"

[featureLabel] MyLabel is optional in the syntax.

3*[featureLabel] MyLabel is optional in the syntax and may repeat. MyLabel may occur zero to 3 times.

featureLabel , 3*[featureLabel] MyLabel is not optional in the syntax. MyLabel must occur once, and may occur up to four times.

5.5.3.1 Rules

The vendor is responsible for the creation of the characterization file. The characterization file represents the DMIS statements supported by the vendor, and information pertinent to the system's capabilities.

Each record in the characterization file has an unlimited length.

5.5.3.1.1 Organization of statements

Every statement in DMIS that has an input format must appear in the input section of the characterization file. Likewise, every DMIS statement that has an output format must appear in the output section of the characterization file.

In the characterization file, the start symbol for DMIS-input files is "inputFile" and for DMIS-output files, it is "outputFile".

5.5.3.1.2 Notation of the DMIS grammar in Extended Backus Naur Form (EBNF)

— A comment within that area of the characterization file being described in Extended Backus Naur Form is enclosed in "(* *)", such as (* this is a comment *).

— The character # is used to indicate the end of a DMIS statement in the DMIS program, or DMIS output.

— All DMIS non-terminal symbols that are first productions of DMIS statements are given in Table 15 — non-terminal symbols for DMIS statements.

— All non-terminal symbols for DMIS statements are defined in this specification so that the characterization file will be consistent and therefore more easily man readable. The input section of a characterization file containing mandatory sections CHFILE/INPUT CHFIL1...ENDCH1 ENDCHF and CHFILE/OUTPUT CHFIL1...ENDCH1 ENDCHF and non-terminal symbols for all DMIS statements defined in this specification is given in Annex C.

— To indicate that a DMIS statement is fully supported by a vendor, the comment "(* FULL *)" will appear at the end of the line for the first productions of that DMIS statement. Refer to clause 5.5.4.

— To indicate that a DMIS statement is partially supported by a vendor, the comment "(* PART *)" will appear at the end of the line for those forst productions. If a statement is partially supported, the full production of the statement will be given and those areas of the production that are not supported are to be enclosed in comment symbols ("(* *)"). Refer to the example of the CONST/POINT statement in clause 5.5.4. If a non-terminal symbol is partially supported by one DMIS statement, but fully supported by others, then it is permissible to create a copy of that non-terminal symbol (with an appropriate name), with the unsupported option(s) commented out. When this is done, the comment "(*PART*)" will appear at the end of the line containing the new non-terminal symbol. Refer to the example of GEOLAG/CONE in clause 5.5.4.

— To indicate that a DMIS statement is not supported by a vendor, the line is enclosed in comment symbols (" (* *)"). The text " not supported " will be included in the line for the first production of that DMIS statement. The full production of the DMIS statement will be included in the characterization file with the lines enclosed in comment symbols. Refer to example of the DO and ENDDO statements in clause 5.5.4.

— To indicate that a statement is a non-pure DMIS extension supported by a vendor, the comment "(* EXTENSION *)" will appear at the end of the line for those productions that define the statement.

— The vendor will state in the header of the characterization file whether line continuations (refer to 5.1.2.5) are supported, and if so, whether there is a maximum total number of characters allowed in a continued line.

5.5.3.1.3 Standard DMIS characterization file grammar

An example of a DMIS 5.1 input format characterization file is given in Annex C.

Table 15 — non-terminal symbols for DMIS statements

DMIS STATEMENT	NON-TERMINAL SYMBOL	DMIS STATEMENT	NON-TERMINAL SYMBOL
$$ (comment character)	commentStm	INCLUD	includStm
(jumptarget)	jumpStm	ITERAT	iteratStm
ACLRAT	aclratStm	JUMPTO	jumptoStm
ALGDEF	algdefStm	KEYCHAR	keycharStm
ASSIGN	assignStm	LITDEF	litdefStm
BADTST	badtstStm	LOCATE	locateStm
BOUND	boundStm	LOTID	lotidStm
CALIB	calibStm	MACRO	macroStm
CALL	callStm	MATDEF	matdefStm
CASE	caseStm	MEAS	measStm
CLMPID	clmpidStm	MFGDEV	mfgdevStm
CLMPSN	clmpsnStm	MODE	modeStm
CLOSE	closeStm	OBTAIN	obtainStm
CMPNTGRP	cmpntgrpStm	OPEN	openStm
CNFRMRUL	cnfrmrulStm	OPERID	operidStm
CONST	constStm	OUTPUT	outputStm
CRGDEF	crgdefStm	PAMEAS	pameasStm
CRMODE	crmodeStm	PARTID	partidStm
CROSCL	crosclStm	PARTRV	partrvStm
CRSLCT	crslctStm	PARTSN	partsnStm
CUTCOM	cutcomStm	PATH	pathStm
CZONE	czoneStm	PLANID	planidStm
CZSLCT	czslctStm	POP	popStm
DATDEF	datdefStm	PRCOMP	prcompStm
DATSET	datsetStm	PREVOP	prevopStm
DATTRGDEF	dattrgdefStm	PROCID	procidStm
DECL	declStm	PROMPT	promptStm
DECPL	decplStm	PSTHRU	psthruStm
DELETE	deleteStm	PTBUFF	ptbuffStm
DEVICE	deviceStm	PTMEAS	ptmeasStm
DFTCAS	dftcasStm	PUSH	pushStm
DISPLY	displyStm	QISDEF	qisdefStm
DMEHW	dmehwStm	RAPID	rapidStm
DMEID	dmeidStm	READ	readStm
DMESW	dmeswStm	RECALL	recallStm
DMESWI	dmeswiStm	REFMNT	refmntStm
DMESWV	dmeswvStm	REPORT	reportStm
DMIS	dmisStm	RESUME	resumeStm

DMIS STATEMENT	NON-TERMINAL SYMBOL	DMIS STATEMENT	NON-TERMINAL SYMBOL
DMISMD	dmismdStm	RMEAS	rmeasStm
DMISMN	dmismnStm	ROTAB	rotabStm
DO	doStm	ROTATE	rotateStm
ELSE	elseStm	ROTDEF	rotdefStm
ENDAT	endatStm	ROTSET	rotsetStm
ENDCAS	endcasStm	SAVE	saveStm
ENDDO	enddoStm	SCAN	scanStm
ENDFIL	endfilStm	SCNMOD	scnmodStm
ENDGO	endgoStm	SCNPLN	scnplnStm
ENDIF	endifStm	SCNSET	scnsetStm
ENDMAC	endmacStm	SELECT	selectStm
ENDMES	endmesStm	SENSOR	sensorStm
ENDSEL	endselStm	SNSDEF	snsdefStm
ENDSYMREQT	endsymreqtStm	SNSET	snsetStm
ENDXTN	endxtnStm	SNSGRP	snsgrpStm
EQUATE	equateStm	SNSLCT	snslctStm
ERROR	errorStm	SNSMNT	snsmntStm
EVAL	evalStm	SYMREQT	symreqtStm
EXTENS	extensStm	TECOMP	tecompStm
EXTFIL	extfilStm	TEXT	textStm
FEAT	featStm	THLDEF	thldefStm
FEDRAT	fedratStm	TOL	tolStm
FILDEF	fildefStm	TOOLDF	tooldfStm
FILNAM	filnamStm	TRANS	transStm
FINPOS	finposStm	UNCERTALG	uncertalgStm
FIXTID	fixtidStm	UNCERTSET	uncertsetStm
FIXTSN	fixtsnStm	UNITS	unitsStm
FLY	flyStm	VALUE	valueStm
FROM	fromStm	VFORM	vformStm
GEOALG	geoalgStm	WINDEF	windefStm
GEOM	geomStm	WKPLAN	wkplanStm
GOHOME	gohomeStm	WRIST	wristStm
GOTARG	gotargStm	WRITE	writeStm
GOTO	gotoStm	XTERN	xternStm
GROUP	groupStm	XTRACT	xtractStm
IF	ifStm		

5.5.4 Example DMIS characterization file grammar

```
CHFILE/INPUT
$$
$$ Comments outside of the EBNF grammar are preceded by $$ as the
$$ first 2 characters of the line.
$$
$$
CHFIL1
(*-------------------------------------------------------------*)
(*-------------------------------------------------------------*)
(*  DMIS INPUT Grammar Version: 5.1 Release 1                 *)
(*-------------------------------------------------------------*)
(*-------------------------------------------------------------*)
(*-- This example is not meant to show the entire   ----------*)
(*-- characterization file.  It is meant to show how ----------*)
(*-- to describe productions that are not supported ----------*)
(*-------------------------------------------------------------*)
(*-- The example is given for the input section only ----------*)
(*-------------------------------------------------------------*)
(*-------------------------------------------------------------*)
(* inputFile is the start symbol                              *)
(* Line continuation is fully supported, with no upper limit  *)
(* on the total number of characters that may appear on a     *)
(* continued line.                                            *)
(*-------------------------------------------------------------*)
(*-------------------------------------------------------------*)
(* A "c" is a DMIS comma.                                     *)
(*-------------------------------------------------------------*)
(*-------------------------------------------------------------*)
(*-- The productions of items not specific to the statements --*)
(*-- in this example are not shown for brevity ---------------*)
(*-------------------------------------------------------------*)


(*-------------------------------------------------------------*)
(* starting symbol and dmisStatementList                      *)
(*-------------------------------------------------------------*)


inputFile=
   dmisFirstStatement , [dmisStatementList] , endfilStm ;

dmisStatementList =
   [dmisStatementList] , dmisStatement

(*-------------------------------------------------------------*)
(* dmisFirstStatement                                         *)
(*-------------------------------------------------------------*)


dmisFirstStatement=
    dmismdStm
   |dmismnStm ;

(*=============================================================*)
(* all other valid DMIS statements except endfilStm           *)
(*=============================================================*)

dmisStatement =
```

```
      aclratStm           (* FULL *)
   |  algdefStm           (* FULL *)
      |...
   |  constStm            (* PART *)
      |...
(* |  doStm              NOT SUPPORTED *)
      |...
(* |  enddoStm           NOT SUPPORTED *)
      |...
   |  geoalgStm           (* PART *)
      |... ;

(*-------------------------------------------------------------*)
(* CONST *)
(*-------------------------------------------------------------*)

constStm =
      CONST , '/' , constMinor , # ;

constMinor =
      constArc
   |  constCircle
   |  constCone
   |  constCylndr
   |  constCparln
   |  constEllips
   |  constEdgept
   |  constGeom
   |  constGcurve
   |  constGsurf
   |  constLine
   |  constPatern
   |  constPlane
(* |  constPoint          NOT SUPPORTED *)
   |  constRctngl
   |  constSgage
   |  constSpart
   |  constSphere
   |  constSympln
   |  constTorus ;

constArc =
      ARC , c , fLabel , c , bfConst
   |  ARC , c , fLabel , c , projctConst
   |  ARC , c , fLabel , c , trConst ;

constCircle =
      CIRCLE , c , fLabel , c , bfConst
   |  CIRCLE , c , fLabel , c , coneConst
   |  CIRCLE , c , fLabel , c , intofConst
   |  CIRCLE , c , fLabel , c , projctConst
   |  CIRCLE , c , fLabel , c , tantoConst
   |  CIRCLE , c , fLabel , c , trConst
   |  CIRCLE , c , fLabel , c , retrieve2 ;

constCompound =
      COMPOUND , c , fLabel , c , buildConst ;

constCone =
      CONE , c , fLabel , c , bfConst
```

```
    |  CONE , c , fLabel , c , trConst ;

constCparln =
       CPARLN , c , fLabel , c , bfConst
    |  CPARLN , c , fLabel , c , projctConst
    |  CPARLN , c , fLabel , c , trConst
    |  CPARLN , c , fLabel , c , retrieve2 ;

constCylndr =
       CYLNDR , c , fLabel , c , bfConst
    |  CYLNDR , c , fLabel , c , trConst
    |  CYLNDR , c , fLabel , c , retrieve1 ;

constEllips =
       ELLIPS , c , fLabel , c , bfConst
    |  ELLIPS , c , fLabel , c , intofConst
    |  ELLIPS , c , fLabel , c , projctConst
    |  ELLIPS , c , fLabel , c , trConst ;

constEdgept =
       EDGEPT , c , fLabel , c , retrieve4 ;

constGeom =
       GEOM , c , fLabel , c , nearptConst ;

constGcurve =
       GCURVE , c , fLabel , c , bfConst
    |  GCURVE , c , fLabel , c , trConst ;


constGsurf =
       GSURF , c , fLabel , c , bfConst
    |  GSURF , c , fLabel , c , trConst ;

constLine =
       LINE , c , fLabel , c , bfConst
    |  LINE , c , fLabel , c , intofConst
    |  LINE , c , fLabel , c , midliConst
    |  LINE , c , fLabel , c , offsetConst
    |  LINE , c , fLabel , c , partoConst
    |  LINE , c , fLabel , c , perptoConst
    |  LINE , c , fLabel , c , projliConst
    |  LINE , c , fLabel , c , tantoConst
    |  LINE , c , fLabel , c , trConst ;

constPatern =
       PATERN , c , fLabel , c , bfConst
    |  PATERN , c , fLabel , c , trConst
    |  PATERN , c , fLabel , c , buildConst ;

buildConst =
       BUILD , c , faLabel , c , featureList ;

constPlane =
       PLANE , c , fLabel , c , bfConst
    |  PLANE , c , fLabel , c , midplConst
    |  PLANE , c , fLabel , c , offsetConst
    |  PLANE , c , fLabel , c , partoConst
    |  PLANE , c , fLabel , c , perptoConst
    |  PLANE , c , fLabel , c , tantoConst
```

```
   |  PLANE , c , fLabel , c , trConst ;

(* constPoint =
       POINT , c , fLabel , c , cogConst
   |   POINT , c , fLabel , c , curveConst
   |   POINT , c , fLabel , c , extremConst
   |   POINT , c , fLabel , c , intofConst
   |   POINT , c , fLabel , c , midptConst
   |   POINT , c , fLabel , c , moveptConst
   |   POINT , c , fLabel , c , pierceConst
   |   POINT , c , fLabel , c , projptConst
   |   POINT , c , fLabel , c , trConst
   |   POINT , c , fLabel , c , vertexConst
   |   POINT , c , fLabel , c , retrieve1 ;
*)

retrieve1 =
            RETRIEVE , c , rentVal , c , featureActualList ;

retrieve2 =
            RETRIEVE , c , rentVal , c , rentVal , c , featureActualList ;

retrieve4 =
            RETRIEVE , c , rentVal , c , rentVal , c , rentVal , c ,
            rentVal , c , featureActualList ;

retrieve2b =
            RETRIEVE , c , rentVal , [c , rentVal , c , rentVal , c ,
            rentVal , c , rentVal] , c , featureActualList ;

constRctngl =
       RCTNGL , c , fLabel , c , bfConst
   |   RCTNGL , c , fLabel , c , trConst ;

constSgage =
       SGAGE , c , seLabel , c , sgageConst ;

constSpart =
       SPART , c , stLabel , c , spartConst ;

constSphere =
       SPHERE , c , fLabel , c , bfConst
   |   SPHERE , c , fLabel , c , trConst
       SPHERE , c , fLabel , c , retrieve2b ;

constSympln =
       SYMPLN , c , fLabel , c , bfConst ;

constTorus =
       TORUS , c , fLabel , c , bfConst
   |   TORUS , c , fLabel , c , trConst ;

bfConst =
       BF , c , featureList
   |   BF , c , indexedFeatureList ;

(*
cogConst =
       COG , c , featureList   ; NOT SUPPORTED
*)
```

```
coneConst =
       CONE , c , DIAM , c , rentVal , c , faLabel
    |  CONE , c , DIST , c , rentVal , c , faLabel ;

(*
curveConst =
       CURVE , c , faLabel , c , featureLabel  ;  NOT SUPPORTED
*)

(*
extremConst =
       EXTREM , c , MIN , c , faLabel , c , extremConstDir  NOT SUPPORTED
    |  EXTREM , c , MAX , c , faLabel , c , extremConstDir ;
*)

extremConstDir =
       extremConstAxial
    |  extremConstVectorial
    |  extremConstFeature
    |  extremConstRadial ;

extremConstAxial =
       posDir ;

extremConstVectorial =
       VEC , c , vector ;

extremConstFeature =
       featureLabel ;

extremConstRadial =
       RADIAL ;

intofConst =
       INTOF , c , faLabel , c , featureLabel ;

midliConst =
       MIDLI , c , faLabel , c , featureLabel ;

midplConst =
       MIDPL , c , faLabel , c , featureLabel ;

(*
midptConst =
       MIDPT , c , faLabel , c , featureLabel  ;  NOT SUPPORTED
*)

(*
moveptConst =
       MOVEPT , c , faLabel , c , vector
    |  MOVEPT , c , faLabel , c , featureLabel , c , rentVal ; NOT SUPPORTED
*)

nearptConst =
       NEARPT , c , faLabel ;

offsetConst =
       OFFSET , c , featureList ;
```

```
partoConst =
       PARTO , c , faLabel , c , THRU , c , featureLabel
   |   PARTO , c , fLabel , c , THRU , c , faLabel ;

perptoConst =
       PERPTO , c , faLabel , c , THRU , c , featureLabel
   |   PERPTO , c , fLabel , c , THRU , c , faLabel ;

(*
pierceConst =
       PIERCE , c , faLabel , c , featureLabel ;  NOT SUPPORTED
*)

projctConst =
       PROJCT , c , faLabel , [c , featureLabel] ;

projliConst =
       PROJLI , c , faLabel , [c , featureLabel] ;

(*
projptConst =
       PROJPT , c , faLabel , [c , featureLabel]   ;  NOT SUPPORTED
*)

sgageConst =
       featureNominalList ;

spartConst =
       featureActualList ;

tantoConst =
       TANTO , c , faLabel , [c , THRU] , c , featureLabel
   |   TANTO , c , fLabel , c , THRU , c , faLabel ;

trConst =
       TR , c , faLabel , [c , datumLabel] ;

(*
vertexConst =
       VERTEX , c , faLabel    ;  NOT SUPPORTED
*)

(*-------------------------------------------------------------*)
(* DO *)
(*-------------------------------------------------------------*)

(*
doStm =                        NOT SUPPORTED
       DO , '/' , doMinor , # ;

doMinor =                   NOT SUPPORTED
       intVar , c , intVal , c , intVal , [c , intVal] ;
*)

(*-------------------------------------------------------------*)
(* ENDDO *)
(*-------------------------------------------------------------*)

(*
enddoStm =                     NOT SUPPORTED
```

```
      ENDDO , # ;
*)

(*-------------------------------------------------------------*)
(* GEOALG *)
(*-------------------------------------------------------------*)

geoalgStm =
      GEOALG , '/' , geoalgMinor , # ;

geoalgMinor =
      geoalgArc
    | geoalgCircle
    | geoalgCone                  (* PART *)
    | geoalgConradsegmnt
    | geoalgCparln
    | geoalgCylndr
    | geoalgCylradsegmnt
    | geoalgEllips
    | geoalgElongcyl
    | geoalgGcurve
    | geoalgGsurf
    | geoalgLine
    | geoalgObject
    | geoalgParpln
    | geoalgPlane
    | geoalgRctngl
    | geoalgRevsurf
    | geoalgSphere
    | geoalgSphradsegmnt
    | geoalgSympln
    | geoalgTorus
    | geoalgTorradsegmnt ;


geoalgArc =
      ARC , c , geoalgSpec1 ;

geoalgCircle =
      CIRCLE , c , geoalgSpec2 ;

geoalgCone =
      CONE , c , geoalgSpec2 ;

geoalgConradsegmnt =
      CONRADSEGMNT , c , geoalgSpec2 ;

geoalgCparln =
      CPARLN , c , geoalgSpec3 ;

geoalgCylndr =
      CYLNDR , c , geoalgSpec2 ;

geoalgCylradsegmnt =
      CYLRADSEGMNT , c , geoalgSpec2 ;

geoalgEllips =
      ELLIPS , c , geoalgSpec1 ;

geoalgElongcyl =
```

```
       ELONGCYL , c , geoalgSpec2 ;

geoalgGcurve =
       GCURVE , c , geoalgSpec4 ;

geoalgGsurf =
       GSURF , c , geoalgSpec5 ;

geoalgLine =
       LINE , c , geoalgSpec1 ;

geoalgObject =
       OBJECT , c , geoalgSpec3 ;

geoalgParpln =
       PARPLN , c , geoalgSpec2 ;

geoalgPlane =
       PLANE , c , geoalgSpec1 ;

geoalgRctngl =
       RCTNGL , c , geoalgSpec1 ;

geoalgRefsurf =
       REVSURF , c , geoalgSpec6 ;

geoalgSphere =
       SPHERE , c , geoalgSpec2 ;

geoalgSphradsegmnt =
       SPHRADSEGMNT , c , geoalgSpec2 ;

geoalgSympln =
       SYMPLN , c , geoalgSpec2 ;

geoalgTorus =
       TORUS , c , geoalgSpec2 ;

geoalgTorradsegmnt =
       TORRADSEGMNT , c , geoalgSpec2 ;

geoalgSpec1 =
       LSTSQR , [c , geoalgFilterSettings]
   |   MINMAX , [c , geoalgFilterSettings]
   |   DEFALT , [c , geoalgFilterSettings]
   |   EXTERN , c , geoalgExternFunc , [c , paramList] ,
            [c , geoalgFilterSettings] ;

geoalgSpec2 =
       LSTSQR , [c , geoalgFilterSettings]
   |   MINMAX , [c , geoalgFilterSettings]
   |   MAXINS , [c , geoalgFilterSettings]
   |   MINCIR , [c , geoalgFilterSettings]
   |   DEFALT , [c , geoalgFilterSettings]
   |   EXTERN , c , geoalgExternFunc , [c , paramList] ,
            [c , geoalgFilterSettings] ;

geoalgSpec3 =
       DEFALT , [c , geoalgFilterSettings]
   |   EXTERN , c , geoalgExternFunc , [c , paramList] ,
```

```
            [c , geoalgFilterSettings] ;

geoalgSpec4 =
       LSTSQR , [c , geoalgFilterSettings]
    |  MINMAX , [c , geoalgFilterSettings]
    |  BSPLIN , [c , geoalgFilterSettings]
    |  DEFALT , [c , geoalgFilterSettings]
    |  EXTERN , c , geoalgExternFunc , [c , paramList] ,
            [c , geoalgFilterSettings] ;

geoalgSpec5 =
       LSTSQR , [c , geoalgFilterSettings]
    |  MINMAX , [c , geoalgFilterSettings]
    |  BEZIER , [c , geoalgFilterSettings]
    |  NURBS , [c , geoalgFilterSettings]
    |  DEFALT , [c , geoalgFilterSettings]
    |  EXTERN , c , geoalgExternFunc , [c , paramList] ,
            [c , geoalgFilterSettings] ;

geoalgSpec6 =
       LSTSQR , [c , geoalgFilterSettings]
    |  BSPLIN , [c , geoalgFilterSettings]
    |  DEFALT , [c , geoalgFilterSettings]
    |  EXTERN , c , geoalgExternFunc , [c , paramList] ,
            [c , geoalgFilterSettings] ;

geoalgExternFunc =
       DMIS , c , mLabel
    |  DME , c , stringVal
    |  SYS , c , stringVal ;

geoalgFilterSettings =
       [geoalgEliminate , c] , geoalgFilter
    |  geoalgEliminate ;

geoalgEliminate =
       ELIMINATE , c , STDDEVLIMIT , c , rentVal
    |  ELIMINATE , c , OFF ;

geoalgFilter =
       FILTER , c , LAMBDAC , c , geoalgFilterType
    |  FILTER , c , CIRCULAR , c , geoalgFilterType
    |  FILTER , c , OFF ;

geoalgFilterType =
       LOWPASS , c , rentVal , c , geoalgFilterCurve
    |  HIGHPASS , c , rentVal , c , geoalgFilterCurve
    |  BANDPASS , c , rentVal , c , rentVal , c , geoalgFilterCurve ;

geoalgFilterCurve =
       GAUSS
    |  TWORC
    |  SPLINE
    |  RCTNGL ;

(*-------------------------------------------------------------*)
(*-------------------------------------------------------------*)
(*--- The rest of productions would follow ---*)
(*-------------------------------------------------------------*)
(*-------------------------------------------------------------*)
```

```
ENDCH1
CHFIL2
ENDCH2
ENDCHF
```

EXAMPLE In CHFIL2, there are three major words shown. For ALGDEF, the specific algorithms supported by the DME are listed by their code number and description. For ERROR, the error codes supported by the DME are listed by their code number and description. For FILDEF, the video filters supported by the DME are listed by their code number and description. The rest of the CHFIL2 section could include other (DME) machine dependent parameters.

```
$$ Characterization File example
$$ INPUT section
CHFILE/INPUT
CHFIL1
inputFile = dmisProg
... = ...
...

ENDCH1
CHFIL2
$$ machine dependent parameters
ALGDEF/CODE,INTGR
    INTGR,CHAR
    INTGR,CHAR
ENDAT
ERROR/CODE,INTGR
    INTGR,CHAR
    INTGR,CHAR
ENDAT
FILDEF/CODE,INTGR
    INTGR,CHAR
    INTGR,CHAR
ENDAT
ENDCH2
CHFIL3
$$ user defined options
ENDCH3
ENDCHF
$$ OUTPUT section
CHFILE/OUTPUT
CHFIL1

outputFile = ...
... = ...
...

ENDCH1
```

```
CHFIL2
ENDCH2
CHFIL3
ENDCH3
ENDCHF
```

6 Statement reference

Throughout this section, the word var_n is used frequently to represent minor words or parameters that can have more than one value. The n in the var_n can be any number.

Each major vocabulary word is described on a separate page.

Each of these pages is formatted in an identical manner, as follows:

— The major word is used as a heading on the top of the page.

— Function: describes the function of the DMIS statement.

— Input Formats: describe the DME's input statement(s).

— Output Formats: describe the DME's output statement(s).

— Where: Minor words and parameters are described.

— Notes: are general notes pertaining to the use of the statement.

Statements are listed in alphabetical order.

6.1 ACLRAT

Function: Used to set the acceleration for measurements, path-directed moves, and rotary tables.

Input Formats:

can be: **ACLRAT/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **HEDROTACL,var_2**
or: **HEDMESACL,var_2**
or: **HEDSCNACL,var_2**
or: **MESACL,var_2**
or: **POSACL,var_2**
or: **ROTACL,var_3**
or: **SCNACL,var_2**

var_2 can be: **MPMM,n**
or: **MMPSS,n**
or: **IPMM,n**
or: **IPSS,n**
or: **var_4**

var_3 can be: **RPMM,n**
or: **var_4**

var_4 can be: **PCENT,n**
or: **HIGH**
or: **LOW**
or: **DEFALT**

DEFALT signifies the DME's default acceleration is to be used.

HEDMESACL signifies that the sensor head measurement accelerations is to be set.

HEDROTACL signifies that the sensor head rotary acceleration is to be set.

HEDSCNACL signifies that the sensor head scanning acceleration is to be set.

HIGH signifies that the DME's internally stored high value is to be used.

IPMM signifies that the value of 'n' is inches per minute per minute.

IPSS signifies that the value of 'n' is inches per second per second.

LOW signifies the DME's internally stored low value is to be used.

MESACL signifies that the measurement acceleration or the acceleration of the sensor for measurement/contact moves is to be set.

MMPSS signifies that the value of 'n' is millimeters per second per second.

MPMM signifies that the value of 'n' is meters per minute per minute.

n is a positive real number representing the acceleration value.

PCENT signifies that the value of 'n' is the percent of maximum as a fraction less than or equal to 1.0 and greater than or equal to 0.01, for example, 0.75 means 75%.

POSACL signifies that the positional acceleration or the acceleration of the sensor for path-directed moves is to be set.

ROTACL signifies that the rotary table's rotational acceleration is to be set.

RPMM signifies that the value of 'n' is revolutions per minute per minute.

SCNACL signifies that the scanning acceleration is to be set.

Note: The following statements are interrelated in the use of rotary tables: ROTDEF, ROTSET, ROTAB, FEDRAT, ACLRAT, and CALIB.

6.2 ALGDEF

Function: Defines an algorithm and assigns a label to it.

Input Formats:

can be: **VA(lname)=ALGDEF/var_1**

Output Formats:

can be: **VA(lname)=ALGDEF/var_1**

Where:

var_1 can be: **CODE,n**
or: **'name' var_2**

var_2 can be: **,parameter var_2**
or: **does not exist**

CODE signifies that the algorithm is being defined with a numeric code.

lname is an alphanumeric label name assigned to the algorithm.

n is a positive integer that represents a DME specific algorithm.

'name' is a text string, enclosed with apostrophes, identifying an algorithm name.

parameter can be a value or a variable.

The ALGDEF statement is passed to the output file when activated by the SNSET statement.

Note 1: The characterization file contains the various algorithms that are supported by the particular DME. Each of these algorithms is assigned an integer code in the characterization file. The intent of this statement is to support those DMEs (particularly video devices), that maintain several alternate algorithms for use in the inspection or evaluation process.

Note 2: Codes for algorithms can, for example, specify counting, intersections, line fits, circle fits, and maximum or minimum points, etc.

6.3 ASSIGN

Function: Assigns the value of an expression to a variable or array element. Any previous variable value is lost.

Input Formats:

can be: **varname=ASSIGN/expr**

Output Formats:

can be: **None**

Where:

expr is a numeric expression, logical expression, vector expression or character expression, refer to section 5.1.2.8.

varname is the name of the previously declared variable or array element to which the value of the expression is assigned.

The data type of the previously declared variable must be consistent with the data type of the expression. The value of a Boolean expression can only be assigned to Boolean variable, the value of a character expression can only be assigned to a character variable and the value of a numeric expression can only be assigned to either a real type or integer type numeric variable. If an expression of one type (Boolean, character or numeric) is assigned to a variable of another type an error occurs.

Assignments between different numeric types are allowed. When a numeric expression that evaluates to a real number is assigned to a variable declared as an INTRG or a LONG, the value is truncated to an integer. And if the expression evaluates to a number outside the storage range of the declared variable type an error occurs.

If the length of the assigned string is larger that the CHAR variable, the result will be truncated to the size of the variable.

6.4 BADTST

Function: Defines a program section in which software checking and recovery for illegal or unexpected touch, or no touch errors is activated for GOTO and point feature measurement only.

Input Formats:

can be: **BADTST/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **ON**
or: **OFF**

OFF toggles off the check in the program section.

ON toggles on the check in the program section.

When a BADTST/ON statement has been executed, program branching with the ERROR statement or DME error recovery for illegal or unexpected touches and no touch conditions is suspended for all GOTO statements and point feature measurements (MEAS/POINT...ENDMES block) only, until a BADTST/OFF statement is encountered. When BADTST is active, the DME is expected to stop the current motion sequence of either a GOTO statement or point feature measurement but it is up to the program to perform any necessary motion to recover from the error condition. To determine if an error condition occurred for a GOTO statement, the intrinsic function BADGT() and for a point-feature measurement, the intrinsic function BADPT() are utilized.

6.5 BOUND

Function: Applies boundaries to features and tolerances.

Input Formats:

can be: **BOUND/var_1 var_2 var_3**

Output Formats:

can be: **BOUND/var_4 var_2 var_3**

Where:

var_1 can be: **F(lname1)**
or: **T(lname2)**

var_2 can be: **,F(lname3)**
or: **,FA(lname4)**

var_3 can be: **var_2 var_3**
or: **does not exist**

var_4 can be: **FA(lname5)**
or: **TA(lname6)**
or: **F(lname1)**
or: **T(lname2)**

F(lname1) is the label of the previously defined feature nominal to be bounded.

F(lname3) is the label of a previously defined nominal bounding plane that is to be one of the bounds for F(lname1) or T(lname2).

FA(lname4) is the label of a previously measured bounding plane that is to be one of the bounds for F(lname1) or T(lname2).

FA(lname5) is the label of the bounded feature actual. This appears in the output format signifying that the bounding definition has been applied to the feature, and the feature actual now represents the bounded feature.

T(lname2) is the label of the previously defined tolerance nominal to be bounded.

TA(lname6) is the label of the bounded tolerance actual. This appears in the output format signifying that the bounding definition has been applied to the tolerance, and the tolerance actual now represents the bounded tolerance.

When the DME ignores boundary information, a feature nominal label for the feature being bounded will be output by the bound statement. Conversely, when the DME utilizes the boundary information, a feature actual label for the feature being bounded will be output by the bound statement.

The BOUND statement along with bounding plane definitions is passed to the output file when executed.

Note 1: Features that are unbounded by definition include: planes, lines, cones, and cylinders. Though cylinders and lines can also be bounded by definition

Note 2: There are no limits in DMIS for the number of planes that can be used to bound a feature or tolerance. In the following example, there are three (3) bounding planes for the cone.

Example:

BOUND/F(cone_1),F(pln1),FA(Angela),FA(Paola)

6.6 CALIB

Function: Used to calibrate and recalibrate a sensor element or rotary table prior to taking measurements and to establish the location of the calibration artifact.

Input Formats:

can be: **CALIB/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **SENS,var_2**
or: **RTAB,var_3**
or: **MASTER,'artifactname'**

var_2 can be: **S(lname1),var_4**
or: **RECALIB**

var_3 can be: **RT(lname2),var_5**
or: **RECALIB**

var_4 can be: **FA(lname3),var_6**
or: **F(lname4),var_6**

var_5 can be: **F(lname5),n**
or: **FA(lname6),FA(lname7)**

var_6 can be: **n**
or: **'text'**
or: **'text',n**

'artifactname' is the name of the calibration artifact.

F(lname4) is the label of a previously defined feature nominal of known dimensions or characteristics to be used for the calibration.

F(lname5) is the label of the point reducible feature nominal that will be measured to define the rotary table's axis of rotation.

FA(lname3) is the label of a previously measured feature of known dimensions or characteristics to be used for the calibration.

FA(lname6)
FA(lname7) are the labels of two previously measured or constructed features that define the rotary table's axis of rotation and origin.

MASTER signifies that the DME will establish the location of the calibration artifact as specified by the 'artifactname' within the DME space.

n is a positive integer that is the number of measurements to be taken for the calibration.

RECALIB signifies that the currently active sensor or rotary table will be recalibrated by canned cycle routine.

RT(lname2) is the label of the rotary table to be calibrated.

RTAB signifies that a rotary table is to be calibrated.

S(lname1) is the label of the sensor to be calibrated.

SENS signifies that a sensor is to be calibrated.

'text' is the name of an algorithm or subroutine, enclosed with apostrophes, resident to the DME that will be used for the calibration.

When a rotary table is calibrated, precedence rules apply. The first feature actual removes as many degrees of freedom as possible and the second feature actual removes any remaining degrees of freedom. In total five degrees of freedom must be constrained: the axis of the rotary table [2 rotational degrees of freedom] and the origin of the rotary table [3 translational degrees of freedom].

Note 1: The objective for the CALIB statement is to provide a means by which the location and orientation of sensors and rotary tables can be accurately established relative to the machine coordinates.

Note 2: The location and orientation specified in the nominal definition for the feature to be used in calibration is irrelevant for manual calibration (MODE/MAN). They are significant, however, if calibration is to occur in automatic or programmed mode. In this case, they are positioned with respect to the current coordinate system. The size specified is the actual size, which is known prior to calibration.

Note 3: When automatic mode is in effect, the calibration is done automatically using the DME's algorithms.

When programmed mode is in effect, the move and measure statements following the CALIB statement are used to measure the calibrating feature. The calibration sequence is terminated with an ENDMES statement.

When manual mode is in effect and the number of points, n, to be used in the calibration has been specified, the calibration is complete when the operator has measured n points on the feature. When the number of points to be used in the calibration has not been specified, the DME specific algorithm determines when the calibration is complete.

6.7 CALL

Function: Used to call and invoke execution of a macro or an external DMIS macro, DME routine, or system program/script.

Input Formats:

can be: **CALL/var_1**

Output Formats:

can be: **CALL/var_1**

Where:

var_1 can be: **M(lname1) var_2**
or: **EXTERN,var_3**

var_2 can be: **,parameter var_2**
or: **does not exist**

var_3 can be: **DME,'routinename' var_4 var_2**
or: **SYS,'pathname' var_4 var_2**
or: **DMIS,var_5**

var_4 can be: **,WAIT**
or: **,CONT**
or: **,ATTACH**
or: **does not exist**

var_5 can be: **M(lname1) var_2**
or: **'module_id'**

ATTACH signifies that the called external routine will be executed after completion of the main program.

CONT signifies that the main program will issue the external call and continue with it's own execution.

DME signifies a DME related routine defined by a routine name.

DMIS signifies a routine written in the DMIS language.

EXTERN signifies a call to invoke the execution of an external routine. This external routine could consist of DMIS statements, DME specific statements, shell script, or high-level language programs. For CALL/EXTERN,DMIS the pathname of the DMIS input module containing the DMIS macro must be previously declared with the EXTFIL/DMIS statement in the DMIS XTERN...ENDXTN block declaration section of the program. For CALL/EXTERN,DME the pathname of the DME routine must be previously declared with the EXTFIL/DME statement in the XTERN...ENDXTN block declaration section of the program. If the ATTACH mode is not supported, the system will default to the CONT mode.

M(lname1) is the label of a macro that is in the current program or in a file referenced by an EXTFIL statement. DMIS macro calls that have been included with the DMIS INCLUD statement are not considered as external calls.

'module_id' is the name of a module defined by the DMISMD statement in an external DMIS input module, enclosed with apostrophes.

parameter is the text that will be substituted in place of the corresponding parameter of the MACRO being called. A CALL parameter can be a value, a label name enclosed in parentheses, such as (label1), or a declared variable name. A label name name enclosed in parentheses can be used only if the corresponding parameter of the called MACRO is a name enclosed in apostrophes.

'pathname' is a fully qualified pathname of the external file including parent directory names, enclosed with apostrophes. (for example, '/usr/dme/mysubfile')

'routinename' is the external DME routine name, enclosed with apostrophes.

SYS signifies a program written in a system language (for example, 'C') or a shell program script that is located by a system pathname.

WAIT signifies that the main program will wait until the external routine execution has been completed.

The CALL statement is passed to the output file when executed.

6.8 CASE

Function: To define a possible branching selection in a SELECT...ENDSEL block. CASE statements occur only in SELECT...ENDSEL blocks.

Input Formats:

can be: **CASE/arg_1**

Output Formats:

can be: **None**

Where:

arg_1 is a literal integer number or a literal text string.

Refer to clause 5.2.4.2.2 and clause 6.168 for details.

Note: An example is given in A.29.

6.9 CLMPID

Function: Defines the identification of a part holding clamp, and assigns a label to it.

Input Formats:

can be: **CI(lname)=CLMPID/'text'**

Output Formats:

can be: **CI(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the part holding clamp.

'text' is a text string, enclosed with apostrophes, that identifies the part holding clamp.

The CLMPID statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined part holding clamp.

6.10 CLMPSN

Function: Defines the identification of a part holding clamp's serial number, and assigns a label to it.

Input Formats:

can be: **CS(lname)=CLMPSN/'text'**

Output Formats:

can be: **CS(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the part holding clamp's serial number.

'text' is a text string, enclosed with apostrophes, that identifies the part holding clamp's serial number.

The CLMPSN statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined part holding clamp serial number.

6.11 CLOSE

Function: Disconnects the specified device identification label from a system device or file.

Input Formats:

can be: **CLOSE/DID(lname1) var_1**

Output Formats:

can be: **CLOSE/DID(lname1) var_1**

Where:

var_1 can be: **,KEEP**
or: **,DELETE**
or: **,END**
or: **does not exist**

DELETE signifies that the specified file is deleted after being closed.

DID(lname1) is the device identification label assigned by the DEVICE statement to the system device to be closed.

END signifies that the ENDFIL statement is to be output to the specified DMIS output file before closing the file. The file is not deleted after being closed.

KEEP signifies that the specified file is not deleted after being closed. The ENDFIL statement is not added.

The CLOSE statement disconnects the system device identified by the DID(lname1) at any point in the input DMIS program. Furthermore, the ENDFIL statement will have the same effect as the execution of a CLOSE/END statement on any opened DMIS output devices, will have the same effect as the execution of a CLOSE/KEEP statement on all other opened storage devices, and will have the same effect as a CLOSE statement on all other devices open at program termination.

The 'does not exist' on var_1 only applies to non-storage devices defined as PRINT, TERM or COMM in the DEVICE statement. The KEEP, DELETE, and END parameters only apply to storage devices defined as STOR or INCR in the DEVICE statement.

The CLOSE statement is passed to the output file when executed.

6.12 CMPNTGRP

Function: Defines a component group without a sensor, and assigns a label to it.

Input Formats:

can be: **SG(lname)=CMPNTGRP/BUILD var_1**

Output Formats:

can be: **SG(lname)=CMPNTGRP/BUILD var_1**

Where:

var_1 can be: **,SG(lname1) var_2**
or: **,SW(lname2) var_2**
or: **,SX(lname3) var_2**
or: **,RM(lname4) var_2**

var_2 can be: **var_1**
or: **does not exist**

lname is an alphanumeric label name assigned to the component group.

RM(lname4) is the label of a previously defined sensor reference mount.

SG(lname1) is the label of a previously defined sensor component group.

SW(lname2) is the label of a previously defined sensor wrist.

SX(lname3) is the label of a previously defined sensor extension.

The CMPNTGRP statement is passed to the output file when executed.

6.13 CNFRMRUL

Function: Defines a conformance rule and assigns to it a label.

Input Formats:

can be: **DR(lname)= CNFRMRUL/var_1**

Output Formats:

can be: **DR(lname)= CNFRMRUL/var_1**
or: **DRA(lname)= CNFRMRUL/var_2**

Where:

var_1 can be: **RULE, n**
or: **'name' var_3**

var_2 can be: **SUPPORTED**
or: **UNSUPPORTED**

var_3 can be: **,parameter var_3**
or: **does not exist**

lname is an alphanumeric label name assigned to the algorithm.

n is a positive integer that represents a DME specific conformance decision rule.

'name' is a text string, enclosed with apostrophes, identifying a conformance decision rule.

parameter can be a value or a variable.

RULE signifies that the conformance decision rule is being defined with a numeric code.

SUPPORTED signifies that the DME supports the previously defined decision rule.

UNSUPPORTED signifies that the DME does not support the previously defined decision rule.

The CNFRMRUL statement is passed to output when activated by the UNCERTSET statement.

Note: The characterization file contains the various conformance rules that are supported by the particular DME. Each of these rules is assigned an integer code in the characterization file.

6.14 CONST (input format 1)

Function: Causes the DME to construct a geometric feature, given the labels of other features used in the construction.

Input Formats:

can be: **CONST/var_1,F(lname1),BF,FA(lname2),var_2 var_3**

Output Formats:

can be: **CONST/var_1,F(lname1),BF,FA(lname2),var_2 var_3**

Where:

var_1 can be: **ARC**
or: **CIRCLE**
or: **CONE**
or: **CPARLN**
or: **CYLNDR**
or: **ELLIPS**
or: **GCURVE**
or: **GSURF**
or: **LINE**
or: **PARPLN**
or: **PLANE**
or: **RCTNGL**
or: **SPHERE**
or: **SYMPLN**
or: **TORUS**

var_2 can be: **FA(lname3)**
or: **F(lname4)**

var_3 can be: **,var_2 var_3**
or: **does not exist**

ARC signifies that a circular arc is to be constructed.

BF signifies that the constructed feature is a best fit using the default or the appropriate GEOALG data fitting algorithm through the features that follow.

CIRCLE signifies that a circle is to be constructed.

CONE signifies that a cone is to be constructed.

CPARLN signifies that a centered parallel line feature is to be constructed.

CYLNDR signifies that a cylinder is to be constructed.

ELLIPS signifies that an ellipse is to be constructed.

F(lname1) is the label of the previously defined feature nominal to be constructed.

F(lname4) is the label of a previously defined feature to be used for construction.

FA(lname2)
FA(lname3) are the labels of previously defined, measured or constructed features to be used for construction.

GCURVE signifies that a generic curve is to be constructed.

GSURF signifies that a generic surface is to be constructed.

LINE signifies that a line is to be constructed.

PARPLN signifies that a parallel plane is to be constructed.

PLANE signifies that a plane is to be constructed.

RCTNGL signifies that a right rectangular prism is to be constructed.

SPHERE signifies that a sphere is to be constructed. Refer to (Figure B.7 — Constructed sphere).

SYMPLN signifies that a symmetric plane is to be constructed.

TORUS signifies that a torus is to be constructed.

The minimum number of previously defined features required for these constructions is given as follows:

ARC.....3	CPARLN,OPEN.....3	LINE 2	SPHERE.....4
CIRCLE 3	CYLNDR 6	PARPLN.....4	SYMPLN.....6
CONE.....6	ELLIPS.....5	PLANE 3	TORUS.....9
CPARLN,FLAT 5	GCURVE.....2	RCTNGL 9	
CPARLN,ROUND...5	GSURF 3		

In the case of a circular arc construction, the end points will be defined as the points where the best-fit circular arc is intersected by radial lines drawn through FA(lname2) and the last feature specified in the format list. All points are coplanar and lie on the circular arc. Refer to (Figure B.2 — Constructed circular arc)

In the case of a circle construction, the points are coplanar and lie on the circle. Refer to (Figure B.4 — Constructed circle) and (Figure B.5 — Constructed circle).

In the case of a cone construction, the direction vector associated with the cone will point along the cone's axis from the vertex to the open end of the cone. Refer to (Figure B.3 — Constructed cone).

In the case of an ellipse construction, the points are coplanar and lie on the ellipse. Refer to (Figure B.6 — Constructed ellipse).

In the case of a right rectangular prism construction, a total of nine(9) points are required. The first three points define its primary plane, the next two points define the secondary plane, and the remaining four points each lie on the remaining planes. Refer to (Figure B.8 — Constructed rectangle).

Since every constructed feature has a feature nominal definition, F(lname1) specified in the program, there should be no ambiguities in the construction. When more than one result is possible from a given construction, the desired result is that which most closely agrees with the feature nominal definition.

When a feature is constructed as the best fit of other point-reducible features, no probe compensation will be applied, regardless of the current probe compensation state.

According to clause 5.3.2.6, if a feature has been measured with point buffering enabled, by a PTBUFF/ON statement, then its individual point data can be referenced by appending the [n] option to F(lname) or FA(lname), where n is the point index from the feature's programmed PTMEAS statements. When the index [n] option for specifying point data is used, the current probe compensation state shall determine whether probe compensation will be applied to the specified data points during the feature construction, as if the constructed feature were measured with that point data with the MEAS or RMEAS statements. If the best fit feature is constructed with point buffering enabled, by a PTBUFF/ON statement, then the individual point data is copied to the point buffer of the constructed feature and is then accessible with the [n] point data index as if it were a feature measured with the MEAS or RMEAS statements. Non-indexed features (specified with FA(lname2) without the [n] index option) and indexed feature point data (specified with FA(lname2)[n] or FA(lname2)[n,m]) cannot be mixed in a single CONST/F(lname),BF statement.

The CONST (input format 1) statement is passed to the output file when executed.

6.15 CONST (input format 2)

Function: Causes the DME to construct a line, given the labels of other features used in the construction.

Input Formats:

can be: **CONST/LINE,F(lname1),var_1**

Output Formats:

can be: **CONST/LINE,F(lname1),var_1**

Where:

var_1 can be: **MIDLI,FA(lname2),var_2**
or: **PROJLI,FA(lname2) var_3**

var_2 can be: **FA(lname3)**
or: **F(lname4)**

var_3 can be: **,FA(lname3)**
or: **,F(lname4)**
or: **does not exist**

F(lname1) is the label of the previously defined feature nominal to be constructed.

F(lname4) is the label of a previously defined feature to be used for construction.

FA(lname2) **FA(lname3)** are the labels of previously defined, measured or constructed features to be used for construction.

LINE signifies that a line is to be constructed.

MIDLI signifies that the feature to be constructed is to be the mid-line of the two previously defined features.

PROJLI signifies that the feature to be constructed is to be the projection of the previously defined line reducible feature onto the specified plane or working plane (if var_3 does not exist).

Since all constructed features have a feature nominal definition, for example, F(lname1) specified in the program, there should be no ambiguities in the construction. When more than one result is possible from a given construction, the desired result is that which most closely agrees with the feature nominal definition.

In the case of two skew lines or axes (A) and (B), both of which are bounded, the bounded midline can be defined in the following way:

Two line segments C and D can be defined which connect the corresponding endpoints of line A and B. The MIDLI is the line segment connecting the midpoints of the lines C and D. Refer to (Figure B.11 — Constructed line).

The mid-line bisects intersecting lines or is parallel to, and half way between, parallel lines.

In the case of two skew lines or axes (A) and (B), one or both of which are unbounded, the unbounded midline can be described in the following way:

Lines A and B have a mid-plane which is defined as perpendicular to the line of closest approach between A and B and passing through the midpoint of that line. The midline between A and B, then, lies in the mid-plane and is the bisector of the angle between the respective projections of A and B into the mid-plane. Refer to (Figure B.9 — Constructed line).

The CONST (input format 2) statement is passed to the output file when executed.

6.16 CONST (input format 3)

Function: Causes the DME to construct a plane, given the labels of other features to use in the construction.

Input Formats:

can be: **CONST/PLANE,F(lname1),MIDPL,FA(lname2),var_1**

Output Formats:

can be: **CONST/PLANE,F(lname1),MIDPL,FA(lname2),var_1**

Where:

var_1 can be: **FA(lname3)**
or: **F(lname4)**

F(lname1) is the label of the previously defined feature nominal to be constructed.

F(lname4) is the label of a previously defined feature to be used for construction.

FA(lname2)
FA(lname3) are the labels of previously defined, measured or constructed features to be used for construction.

MIDPL signifies that the plane to be constructed is to be a mid-plane between the two features that follow.

PLANE signifies that a plane is to be constructed.

Since all constructed features have a feature nominal definition, for example, F(lname1) specified in the program, there should be no ambiguities in the construction. When more than one result is possible from a given construction, the desired result is that which most closely agrees with the feature nominal definition.

The mid-plane bisects intersecting planes, lines or axes or is half way between and parallel to parallel planes, lines, or axes.

Two skew lines (A) and (B) have a mid-plane which is defined as perpendicular to the line of closest approach between A and B passing through the midpoint of that line. Refer to (Figure B.9 — Constructed line).

The CONST (input format 3) statement is passed to the output file when executed.

6.17 CONST (input format 4)

Function: Causes the DME to construct a point, given the labels of other features to use in the construction.

Input Formats:

can be: **CONST/POINT,F(lname1),var_1**

Output Formats:

can be: **CONST/POINT,F(lname1),var_1**

Where:

var_1 can be: **MIDPT,FA(lname2),var_2**
or: **PIERCE,FA(lname2),var_2**
or: **VERTEX,FA(lname2)**
or: **PROJPT,FA(lname2) var_3**
or: **MOVEPT,FA(lname2),var_4**
or: **CURVE,FA(lname2),var_2**
or: **EXTREM,var_5,FA(lname2),var_6**
or: **COG,FA(lname3) var_7**

var_2 can be: **FA(lname4)**
or: **F(lname5)**

var_3 can be: **,FA(lname4)**
or: **,F(lname5)**
or: **does not exist**

var_4 can be: **dx,dy,dz**
or: **F(lname5),dist**
or: **FA(lname4),dist**

var_5 can be: **MIN**
or: **MAX**

var_6 can be: **XDIR**
or: **YDIR**
or: **ZDIR**
or: **VEC,i,j,k**
or: **F(lname5)**
or: **FA(lname4)**
or: **RADIAL**

var_7 can be: **,FA(lname3) var_8**
or: **,F(lname6) var_8**

var_8 can be: **var_7**
or: **does not exist**

COG signifies that the feature to be constructed is to be the center of gravity (geometric center) from the previously defined point reducible features.

CURVE signifies that the feature to be constructed is a point at the intersection of two previously defined features. The features involved in the construction must contain both location and orientation data.

dist is the incremental distance along the feature axis.

dx,dy,dz are the incremental distances in Cartesian coordinates.

EXTREM signifies that the feature to be constructed is the measurement point for the previously defined feature that is the most extreme in the direction specified in var_5.

F(lname1) is the label of the previously defined feature nominal to be constructed.

F(lname5) is the label of a previously defined feature to be used for the construction.

F(lname6)	is the label of a previously defined point reducible feature used for the construction.
FA(lname2) **FA(lname4)**	are the labels of previously defined, measured or constructed features to be used for the construction.
FA(lname3)	is the label of a previously defined, measured or constructed point reducible feature to be used for the construction.
i,j,k	is the unit vector along which the most extreme point will be found.
MAX	signifies that the extreme point will be calculated along the direction specified in var_6.
MIDPT	signifies that the feature to be constructed is to be the midpoint of the two previously defined features that follow.
MIN	signifies that the extreme point will be calculated against the direction specified in var_6.
MOVEPT	signifies that the feature to be constructed is to be offset from a previously measured point by the specified incremental distances or along the specified feature axis by the specified distance.
PIERCE	signifies that the feature to be constructed is a point at the intersection of the line reducible feature represented by FA(lname2) with the surface of F(lname5) or FA(lname4). In the case of multiple solutions for the constructed pierced point feature, the one that is nearest the nominal definition will be utilized.
POINT	signifies that a point is to be constructed.
PROJPT	signifies that the feature to be constructed is to be the projection of the previously defined feature onto a specified plane or line reducible feature or, if var_3 does not exist, onto the working plane.
RADIAL	signifies that the extreme point will be calculated along a direction pointing away from the center and perpendicular to the surface of the measured feature FA(lname2) at the measured point.
VEC	signifies that the direction for the extreme evaluation will be specified as a unit vector.
VERTEX	signifies that the feature to be constructed is the vertex of the previously defined cone that follows.
XDIR **YDIR** **ZDIR**	signifies that the most extreme point will be calculated along the specified direction.

When VERTEX is specified, FA(lname2) is a cone. Refer to (Figure B.14 — Constructed point).

When PROJPT is specified, FA(lname2) is a point reducible feature. Refer to (Figure B.12 — Constructed mid-point).

When MOVEPT is specified, FA(lname2) is a point reducible feature. Refer to (Table 8 — Reducible features).

When CURVE is specified, FA(lname2) and FA(lname4) or F(lname5) are two features that contain position and orientation data. Both must be point reducible features, the first of which is measured. The features are treated as planes which when intersected, form a line representing the curve line. The point on the line is controlled by projecting the nominal point from the feature declaration (FEAT/POINT) onto the curve line. Refer to (Figure B.32 — Construction of a point on a curve) and (Figure B.33 — Construction of a point on a curve).

Since all constructed features have a feature nominal definition, for example, F(lname1) specified in the program, there should be no ambiguities in the construction. When more than one result is possible from a given construction, the desired result is that which most closely agrees with the feature nominal definition.

When MIDPT is specified, the FA(lname2) and FA(lname4) or F(lname5) are two point reducible features, the first of which is measured. Refer to (Figure B.12 — Constructed mid-point).

The RADIAL parameter can be used with circular arc, circle, cylinder, ellipse, cone and sphere features.

According to clause 5.3.2.6, if a feature has been measured with point buffering enabled, with a PTBUFF/ON statement, then its individual point data can be referenced by appending the [n] option to F(lname) or FA(lname), where n is the point index from the feature's programmed PTMEAS statements. Using this subscripted label option is equivalent to using an F(lname) or FA(lname) of a feature type POINT and can only be used when a feature type POINT is appropriate. In the case of CONST/POINT, use of the subscripted label option is only appropriate when used with the MIDPT, PROJPT, or MOVEPT minor words.

For constructed features of type POINT, the following should be used to define the ijk component of the resulting point feature:

- When MIDPT is specified , the resulting ijk for the mid point is the same as the feature nominal ijk definition.
- When VERTEX is specified, the resulting ijk for the vertex is the same as the cone's ijk direction.
- When PROJPT is specified, the resulting ijk for the projection is in the direction from the resulting projected feature to the feature that was being projected. In the case where the resulting feature is already on the line or plane feature being projected to, the resulting ijk should be the same as the nominal ijk for the constructed feature.
- When MOVEPT is specified , the resulting ijk for the point is the same as the feature nominal ijk definition.
- When CURVE is specified, the resulting ijk for the point is the same as the feature nominal ijk definition.
- When EXTREM is specified, the resulting ijk for the extreme point is the same as the direction specified in var_6 .
- When COG is specified, the resulting ijk for the center of gravity point is the same as its nominal definition.
- When PIERCE is specified, the resulting ijk for the pierce point is the direction of the surface normal at that point.

The CONST (input format 4) statement is passed to the output file when executed.

6.18 CONST (input format 5)

Function: Causes the DME to construct a geometric feature, given the labels of other features to use in the construction.

Input Formats:

Can be: **CONST/var_1,F(lname1),PROJCT,FA(lname2) var_2**

Output Formats:

Can be: **CONST/var_1,F(lname1),PROJCT,FA(lname2) var_2**

Where:

var_1 can be: **ARC**
or: **CIRCLE**
or: **CPARLN**
or: **ELLIPS**
or: **GCURVE**

var_2 can be: **,FA(lname3)**
or: **,F(lname4)**
or: **does not exist**

ARC signifies that a circular arc is to be constructed.

CIRCLE signifies that a circle is to be constructed.

CPARLN signifies that a close ended parallel line feature is to be constructed.

ELLIPS signifies that an ellipse is to be constructed.

F(lname1) is the label of the previously defined feature nominal to be constructed.

F(lname4) is the label of a previously defined feature to be used for construction.

FA(lname2)
FA(lname3) are the labels of previously defined, measured or constructed features to be used for construction.

GCURVE signifies that a generic curve is to be constructed.

PROJCT signifies that the feature to be constructed is to be the projection of the previously defined and measured feature onto the specified plane or working plane (if var_2 does not exist).

Since every constructed feature has a feature nominal definition; for example, F(lname1), specified in the program, there should be no ambiguities in the construction. When more than one result is possible from a given construction, the desired result is that which most closely agrees with the feature nominal definition. Refer to (Figure B.14 — Constructed point).

The CONST (input format 5) statement is passed to the output file when executed.

6.19 CONST (input format 6)

Function: Causes the DME to construct a geometric feature, given the labels of other features used in the construction.

Input Formats:

can be: **CONST/var_1,FA(lname1),var_3**

Output Formats:

can be: **CONST/var_1,FA(lname1),var_3**

Where:

var_1 can be: **CIRCLE,F(lname2),var_2**
or: **LINE,F(lname2),var_2**
or: **POINT,F(lname2),INTOF**
or: **ELLIPS,F(lname2),INTOF**

var_2 can be: **TANTO**
or: **INTOF**

var_3 can be: **FA(lname3)**
or: **F(lname4)**

CIRCLE signifies that a circle is to be constructed.

F(lname2) is the label of the previously defined feature nominal to be constructed.

F(lname4) is the label of a previously defined feature to be used for construction.

FA(lname1)
FA(lname3) are the labels of previously defined, measured or constructed features to be used for construction.

INTOF signifies that the constructed feature is given by the intersection of the features that follow; at least the first of which must be a previously measured feature.

LINE signifies that a line is to be constructed.

POINT signifies that a point is to be constructed.

TANTO signifies that the constructed feature is to be tangent to the features that follow; at least the first of which must be a previously measured feature.

In the case of constructing a circle tangent to two coplanar lines, the lines must not be parallel. The constructed circle's diameter will be that specified in the feature nominal definition. Refer to (Figure B.18 — Constructed circle). When required to construct a circle tangent to two parallel lines, use format 7, where the through point is both the circle's tangent point, and the point on the second line.

In the case of constructing a circle tangent to two coplanar circles, the circles must not be circumscribed. The constructed circle's diameter will be that specified in the feature nominal definition. When the nominal diameter is less than the minimum tangent circle, the minimum tangent circle will be constructed. Refer to (Figure B.16 — Constructed circle).

When CIRCLE and INTOF are specified, FA(lname1) is an actual plane, and var_3 is the label of a previously defined nominal or actual cone, cylinder, or sphere. The cone's or cylinder's axis must be perpendicular to the plane of FA(lname1). Refer to (Figure B.18 — Constructed circle).

When LINE and TANTO are specified the FA(lname1) is an actual circle and var_3 is the label of a previously defined nominal or actual circle, coplanar with FA(lname1). Refer to (Figure B.20 — Constructed line).

When LINE and INTOF are specified, the FA(lname1) is an actual plane and var_3 is the label of a previously defined nominal or actual plane. Refer to (Figure B.22 — Constructed line).

When POINT and INTOF are specified then:

When FA(lname1) is a line reducible feature then;

var_3 can be: a previously defined line, coplanar with FA(lname1). Refer to (Figure B.20 — Constructed line). The resulting ijk vector of the constructed feature is defined as the cross product of the feature

specified by var_3 with FA(lname1).

or: a previously defined circle, coplanar with FA(lname1). Refer to (Figure B.20 — Constructed line). The resulting ijk vector of the constructed feature is defined as the point of tangency where the line intersects the circle and its direction is outward for OUTER circles and inward for INNER circles.

or: a previously defined plane. Refer to (Figure B.20 — Constructed line). The resulting ijk vector of the constructed feature is defined with the same vector direction of the plane feature in var_3.

or: a previously defined line, not coplanar with FA(lname1). The resulting ijk vector of the constructed feature is defined as the cross product of the feature specified by var_3 with FA(lname1). And the xyz coordinate of the point feature will be the midpoint between the closest points of the line features.

or: a previously defined generic surface. The resulting xyz coordinate is the intersection point of the line with the surface. The resulting ijk vector is the vector of the surface at that point.

or: a previously defined generic curve, coplanar with FA(lname1). The resulting xyz coordinate is the intersection point of the line with the curve. The resulting ijk vector is the cross product of the direction vector of curve (specified by var_3) at that point with FA(lname1).

or: a previously defined generic curve, not coplanar with FA(lname1). The resulting xyz coordinate is the midpoint of the nearest point on the curve to the line. The resulting ijk vector is the cross product of the direction vector of curve (specified by var_3) at its nearest point to the line with FA(lname1).

When FA(lname1) is a circle feature then,

var_3 can be: a previously defined, measured, or constructed line, coplanar with FA(lname1). Refer to (Figure B.22 — Constructed line). The resulting ijk vector of the constructed feature is defined as the cross product of the feature specified by var_3 with FA(lname1).

or: a previously defined measured, or constructed circle, coplanar with FA(lname1). Refer to (Figure B.22 — Constructed line). The resulting ijk vector of the constructed feature is defined as the point of tangency where the line intersects the circle and its direction is outward for OUTER circles and inward for INNER circles.

When FA(lname1) is a plane reducible feature then,

var_3 can be: a previously defined line intersecting FA(lname1). The resulting ijk vector of the constructed feature is defined with the same vector direction as the plane feature.

or: a previously defined cone whose axis intersects FA(lname1). The resulting ijk vector of the constructed feature is defined with the same vector direction of the plane feature.

or: a previously defined cylinder whose axis intersects FA(lname1). The resulting ijk vector of the constructed feature is defined with the same vector direction of the plane feature.

or: a previously defined generic curve intersecting FA(lname1). The resulting ijk vector of the constructed feature is the normal direction of the plane FA(lname1).

When FA(lname1) is a generic surface then,

var_3 can be: a previously defined line intersecting FA(lname1). The resulting ijk vector of the constructed feature is the normal direction of the generic surface at the point of intersection.

When FA(lname1) is a generic curve then,

var_3 can be: a previously defined line, coplanar with FA(lname1). The resulting xyz coordinate is the intersection point of the line with the curve. The resulting ijk vector is the cross product of the direction vector of curve (specified by var_3) at that point with FA(lname1).

or: a previously defined line, not coplanar with FA(lname1). The resulting xyz coordinate is the midpoint of the nearest point on the curve to the line. The resulting ijk vector is the cross product of the direction vector of curve (specified by var_3) at its nearest point to the line with FA(lname1).

or: a previously defined plane, which intersects FA(label1). The resulting ijk vector of the constructed feature is the normal direction of the plane.

When ELLIPS and INTOF are specified then:

When FA(lname1) is a cylinder or cone feature then,

var_3 can be: a previously defined plane. The resulting ijk vector of the constructed feature is defined with the same vector direction of the plane feature. When FA(lname1) is a plane reducible feature then,

var_3 can be: a previously defined cylinder or cone whose axis intersects FA(lname1). The resulting ijk vector of the constructed feature is defined with the same vector direction of the plane feature.

Since every constructed feature has a feature nominal definition; for example, F(lname2) specified in the program, there should be no ambiguities in the construction. When more than one result is possible from a given construction, the desired result is that which most closely agrees with the feature definition nominal.

When CIRCLE and TANTO are specified FA(lname1) is an actual line or circle and var_3 is the label of a previously defined nominal, or actual line or circle, coplanar with FA(lname1). Refer to (Figure B.16 — Constructed circle).

The CONST (input format 6) statement is passed to the output file when executed.

6.20 CONST (input format 7)

Function: Causes the DME to construct a geometric feature, given the labels of other features to use in the construction.

Input Formats:

can be: **CONST/var_1,var_2**

Output Formats:

can be: **CONST/var_1,var_2**

Where:

var_1 can be: **CIRCLE,F(lname1),TANTO**
or: **LINE,F(lname1),var_3**
or: **PLANE,F(lname1),var_3**

var_2 can be: **FA(lname2),THRU,var_4**
or: **F(lname3),THRU,FA(lname4)**

var_3 can be: **PERPTO**
or: **TANTO**
or: **PARTO**

var_4 can be: **FA(lname4)**
or: **F(lname5)**

CIRCLE signifies that a minimum diameter circle is to be constructed.

F(lname1) is the label of the previously defined feature nominal to be constructed.

F(lname3) is the label of a previously defined feature nominal to be used in the construction. Note that it need not be previously measured.

F(lname5) is the label of a previously defined point reducible feature to be used as the through point.

FA(lname2) is the label of a previously defined, measured or constructed feature to be used in the construction.

FA(lname4) is the label of a previously defined, measured or constructed point reducible feature to be used as the through point.

LINE signifies that a line is to be constructed.

PARTO signifies that the constructed feature is to be parallel to the following features.

PERPTO signifies that the constructed feature is to be perpendicular to the following features.

PLANE signifies that a plane is to be constructed.

TANTO signifies that the constructed feature is to be tangent to the following features.

THRU signifies that the feature being constructed passes through the following point.

When CIRCLE and TANTO are specified:

FA(lname4) F(lname5) is: a previously defined, or actual line or circle, in which case the constructed circle will lie in the plane defined by FA(lname2) or F(lname3) and the through point. Note that FA(lname2) or F(lname3) and the through point must be coplanar. Refer to (Figure B.23 — Constructed point).

or: a previously defined plane, in which case the through point must not lie in the plane of FA(lname2) or F(lname3). The constructed circle will lie in a plane perpendicular to the plane of FA(lname2) or F(lname3) the through point. Refer to (Figure B.27 — Constructed circle).

When LINE and PERPTO are specified, FA(lname2) or F(lname3) is the label of a previously defined nominal or actual line or plane. Refer to (Figure B.30 — Constructed line).

When LINE and TANTO are specified, FA(lname2) or FA(lname1) is the label of a previously defined nominal or actual circle, and the through point is in the plane of the circle but not within the circle. Refer to (Figure B.24 — Constructed circle).

When LINE and PARTO are specified, FA(lname2) or FA(lname1) is the label of a previously defined nominal line, actual line or line reducible feature. Refer to (Figure B.27 — Constructed circle) and (Table 8 — Reducible features).

When PLANE and PERPTO are specified, FA(lname2) or FA(lname1) is the label of a previously defined nominal line, actual line or line reducible feature. Refer to (Figure B.30 — Constructed line) and (Table 8 — Reducible features).

When PLANE and TANTO are specified, FA(lname2) or FA(lname1) is the label of a previously defined nominal or actual circle, in which case the constructed plane will be perpendicular to the plane of the circle. The through point must not lie within the cylindrical drive volume produced by a projection of the circle along the circle's vector. Refer to (Figure B.25 — Constructed line).

When PLANE and PARTO are specified, FA(lname2) or FA(lname1) is the label of a previously defined nominal or actual plane. Refer to (Figure B.28 — Constructed line).

Since every constructed feature has a feature nominal definition, F(lname1) specified in the program, there should be no ambiguities in the construction. When more than one result is possible from a given construction, the desired result is that which most closely agrees with the feature nominal definition.

The CONST (input format 7) statement is passed to the output file when executed.

6.21 CONST (input format 8)

Function: Causes the DME to construct a geometric feature, given the labels of other features to use in the construction.

Input Formats:

can be: **CONST/var_1,F(lname1),OFFSET,FA(lname2) var_2**

Output Formats:

can be: **CONST/var_1,F(lname1),OFFSET,FA(lname2) var_2**

Where:

var_1 can be: **LINE**
or: **PLANE**

var_2 can be: **,FA(lname2) var_2**
or: **,F(lname3) var_2**
or: **does not exist**

F(lname1) is the label of the previously defined feature nominal to be constructed.

F(lname3) is the label of a previously defined feature to be used for the construction.

FA(lname2) is the label of a previously defined, measured or constructed feature to be used for the construction.

LINE signifies that a line is to be constructed.

OFFSET signifies that the constructed feature is formed by the offsets obtained from the feature nominal definitions.

PLANE signifies that a plane is to be constructed.

Note: The minimum number of point reducible features for line and plane constructs are 2 and 3 respectively.

The CONST (input format 8) statement is passed to the output file when executed.

6.22 CONST (input format 9)

Function: Causes the DME to construct a soft gauge, and assign to it a label, given the labels of other features to use in the construction.

Input Formats:

can be: **CONST/SGAGE,SE(lname1) var_1**

Output Formats:

can be: **CONST/SGAGE,SE(lname1) var_1**

Where:

var_1 can be: **,F(lname2) var_2**

var_2 can be: **var_1**
or: **does not exist**

F(lname2) is the label of a previously defined feature nominal to be used for the construction.

SE(lname1) is an alphanumeric label assigned to the soft gauge. This label then identifies the soft gauge to be constructed containing the feature nominals for comparison to the feature actuals.

SGAGE signifies a group of feature nominals to be established as the soft gauge.

The order of the features as they appear in the above statement must correspond to the correct feature in the CONST/SPART statement. This will cause the appropriate features to be compared with each other.

The CONST (input format 9) statement is passed to the output file when executed.

6.23 CONST (input format 10)

Function: Causes the DME to construct a soft part, and assign to it a label, given the labels of other features to use in the construction.

Input Formats:

can be: **CONST/SPART,ST(lname1) var_1**

Output Formats:

can be: **CONST/SPART,ST(lname1) var_1**

Where:

var_1 can be: **,FA(lname2) var_2**

var_2 can be: **var_1**
or: **does not exist**

FA(lname2) is the label of a previously defined, measured or constructed feature to be used for the construction.

SPART signifies a group of features to be compared to a soft gauge.

ST(lname1) is an alphanumeric label assigned to the soft part. This label then identifies the part to be constructed containing the measured features for the comparison to the soft gauge.

The order of the features as they appear in the above statement must correspond to the correct feature in the CONST/SGAGE statements. This will cause the appropriate features to be compared with each other.

The CONST (input format 10) statement is passed to the output file when executed.

6.24 CONST (input format 11)

Function: Causes the DME to construct a geometric feature, given the labels of other features to use in the construction.

Input Formats:

can be: **CONST/var_1,F(lname1),TR,FA(lname2) var_2**

Output Formats:

can be: **CONST/var_1,F(lname1),TR,FA(lname2) var_2**

Where:

var_1 can be: **ARC**
or: **CIRCLE**
or: **CONE**
or: **CPARLN**
or: **CYLNDR**
or: **ELLIPS**
or: **GCURVE**
or: **GSURF**
or: **LINE**
or: **PATERN**
or: **PLANE**
or: **POINT**
or: **RCTNGL**
or: **SPHERE**
or: **TORUS**

var_2 can be: **,D(lname3)**
or: **,DA(lname4)**
or: **does not exist**

ARC signifies that a circular arc is to be constructed.

CIRCLE signifies that a circle is to be constructed.

CONE signifies that a cone is to be constructed.

CPARLN signifies that a centered parallel line feature is to be constructed.

CYLNDR signifies that a cylinder is to be constructed.

D(lname3)
DA(lname4) is the label for which nominal part coordinate system, D(lname3), or actual part coordinate system, DA(lname4), is used to perform the transformation.

ELLIPS signifies that an ellipse is to be constructed.

GCURVE signifies that a generic curve is to be constructed.

GSURF signifies that a generic surface is to be constructed.

F(lname1) is the label of the previously defined feature nominal to be constructed.

FA(lname2) is the label of a previously defined, measured or constructed feature to be used for the construction.

LINE signifies that a line is to be constructed.

PATERN signifies that a pattern is to be constructed.

PLANE signifies that a plane is to be constructed.

POINT signifies that a point is to be constructed.

RCTNGL signifies that a right rectangular prism is to be constructed.

SPHERE signifies that a sphere is to be constructed. Refer to (Figure B.7 — Constructed sphere).

TORUS signifies that a torus is to be constructed.

TR signifies that the constructed feature is a copy of like characteristic data of the previously measured or constructed feature excepting that the previously measured or constructed feature is transformed to the specified coordinate system or to the current coordinate system when no coordinate system has been specified. The source feature type does not necessarily need to be the same as the target feature type.

The CONST (input format 11) statement is passed to the output file when executed.

6.25 CONST (input format 12)

Function: Causes the DME to construct a geometric feature, given the label of a cone feature to use in the construction.

Input Formats:

can be: **CONST/CIRCLE,F(lname1),CONE,var_1,FA(lname2)**

Output Formats:

can be: **CONST/CIRCLE,F(lname1),CONE,var_1,FA(lname2)**

Where:

var_1 can be: **DIAM,diameter**
or: **DIST,distance**

CIRCLE signifies that a circle is to be constructed.

DIAM signifies that the circle will be constructed on the actual cone where the cone has a diameter of the value specified by 'diameter'.

diameter is a positive real number representing the diameter of the circle that will be constructed.

DIST signifies that the circle will be constructed at a specified distance from the actual cone vertex along the actual cone axis.

distance is a non-zero real number representing the distance from the actual cone vertex along the actual cone axis at which location the circle will be constructed. A negative distance indicates a distance in the opposite direction of the actual cone axis.

F(lname1) is the label of the previously defined nominal circle feature to be constructed.

FA(lname2) is the label of the previously defined, measured or constructed cone feature to be used for the construction.

The CONST (input format 12) statement is passed to the output file when executed.

6.26 CONST (input format 13)

Function: Causes the DME to construct a geometric feature, given the label of a feature defined in a previous FEAT/GEOM statement.

Input Formats:

can be: **CONST/GEOM,F(lname1),NEARPT,FA(lname2)**

Output Formats:

can be: **CONST/GEOM,F(lname1),NEARPT,FA(lname2)**

Where:

F(lname1) is the label of the previously defined feature nominal to be constructed, defined in a previous FEAT/GEOM statement.

FA(lname2) is the label of a previously defined, measured or constructed point-reducible feature to be used for the construction.

GEOM signifies that a geometry feature is to be constructed.

NEARPT signifies that the feature to be constructed is the point contained within the geometry of F(lname) which is nearest to the previously measured or constructed FA(lname2) point.

The CONST (input format 13) statement is passed to the output file when executed.

6.27 CONST (input format 14)

Function: Causes the DME to construct a parent feature (compound, pattern) from actuals of those features specified in the parent feature's nominal definition.

Input Formats:

can be: **CONST/var_1**

Output Formats:

can be: **CONST/var_1**

Where:

var_1 can be: **COMPOUND,F(lname1),BUILD**
or: **PATERN,F(lname1),BUILD**

F(lname1) is the label of the previously defined compound or pattern feature nominal to be constructed.

BUILD signifies that the target parent feature (compound, pattern) will be constructed from actuals of those features specified in the parent feature's nominal definition.

COMPOUND signifies that a compound feature is to be constructed.

PATERN signifies that a pattern feature is to be constructed.

All of the features in the parent feature's nominal definition must have been previously measured or constructed.

For pattern features, only feature-of-size features that have appropriate direction vectors (for example a bolt hole circle) can be used for the construction.

The CONST (input format 14) statement is passed to the output file when executed.

6.28 CONST (input format 15)

Function: Constructs a feature out of data collected by several measurements.

Input Formats:

can be: **CONST/var_1,FA(lname1) var_2**

Output Formats:

can be: **CONST/var_1,FA(lname1) var_2**

Where:

var_1 can be: **POINT,F(lname),RETRIEVE,search_radius**
or: **CIRCLE,F(lname),RETRIEVE,search_radius,depth**
or: **CPARLN,F(lname),RETRIEVE,search_radius,depth**
or: **CYLNDR,F(lname),RETRIEVE,search_radius**
or: **EDGEPT,F(lname),RETRIEVE,distance,patch_radius,depth,search_radius**
or: **SPHERE,F(lname),RETRIEVE,search_radius var_3**

var_2 can be: **,FA(lname2) var_2**
or: **does not exist**

var_3 can be: **,i,j,k**
or: **does not exist**

CIRCLE signifies that a circle is to be retrieved from scanned data. Refer to (Figure B.34 — Construction, CONST/CIRCLE,… RETRIEVE).

CPARLN signifies that a centered parallel line feature is to be retrieved from scanned data. Refer to (Figure B.35 — Construction, CONST/CPARLN,… RETRIEVE).

CYLNDR signifies that a cylinder is to be retrieved from scanned data.

depth is a non-negative real number that defines a measuring depth.

distance is a non-negative real number that is the distance from the surfaces edge where measurements to adjust orientation and location are taken.

EDGEPT signifies that an edge point is to be retrieved from scanned data. Refer to (Figure B.36 — Construction, CONST/EDGEPT,… RETRIEVE).

F(lname) is the label of a previously defined feature to be retrieved from the scanned data.

FA(lname1) is the label of a previously measured feature containing the scanned data.

FA(lname2) is the label of a previously measured feature containing the scanned data.

i,j,k is a unit vector describing the sphere's axis.

patch_radius is a non-negative real number that is a radius around the edge points reference used to adjust location and orientation.

POINT signifies that a point is to be retrieved from scanned data.

RETRIEVE signifies that data to be used for the construction will be retrieved from previously scanned data.

search_radius is a non-negative real number that is an area around the nominal feature, where the actual feature can be expected.

SPHERE signifies that a sphere is to be retrieved from scanned data.

The values for search_radius and depth can be zero. If a search_radius of zero is used, the complete scanned data will be used.

The search_radius is the radius of a cylinder. All scanned points within this cylinder are used for the retrieval of the feature. The cylinder's axis is defined by the feature's direction, the cylinder's axis passes through the feature's centerpoint.

The CONST (input format 15) statement is passed to the output file when executed.

6.29 CRGDEF

Function: Defines an independent sensor carrying carriage, and assigns a label to it.

Input Formats:

can be: **CR(lname)=CRGDEF var_1**

Output Formats:

can be: **CR(lname)=CRGDEF var_1**

Where:

var_1 can be: **/sx,sy,sz,dx,dy,dz,xi,xj,xk,yi,yj,yk,zi,zj,zk,ai,aj,ak**
or: **does not exist**

ai,aj,ak is the unit direction vector describing the axis of motion of the carriage's ram. The ram is the moving component of the carriage that carries the sensor.

dx,dy,dz is the sensor's linear measurement respectively in x,y,z, relative to its respective sx,sy,sz positions.

lname is an alphanumeric label name assigned to the carriage.

sx,sy,sz are the smallest x,y,z coordinates from the DME's zero, to the carriage's zero reference. Refer to further normative text below.

xi,xj,xk is the unit vector for the carriage's x axis in the direction of travel from the sx position.

yi,yj,yk is the unit vector for the carriage's y axis in the direction of travel from the sy position.

zi,zj,zk is the unit vector for the carriage's z axis in the direction of travel from the sz position.

The DME zero position is the origin of the machine coordinate system. The sx,sy,sz are the machine coordinates from the zero position to the extreme minimum position of the carriage, or to its zero reference as identified in the characterization file. All unit vectors are relative to the machine coordinate system. The work zone or measurement volume of the carriage is determined by the diagonal corners, that is, sx,sy,sz and dx,dy,dz.

The CRGDEF statement is required for systems where more than one carriage exists, and optional for systems with a single carriage. It can also be used to specify the DME's ram vector when required to accommodate special applications or post processing of the DMIS program. Refer to (Figure B.45 — A dual system illustration) for an illustration of a dual system.

The CRGDEF statement is passed to the output file when executed.

6.30 CRMODE

Function: Selection of the mode in which DMIS statements are executed.

Input Formats:

can be: **CRMODE/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **SEQNTL**
or: **SIMUL**
or: **SYNC**

SEQNTL signifies that the following part program statements (and consequent DME operations) are executed strictly in accordance with the part program sequence, without simultaneous execution by different carriages, until a CRMODE/SIMUL or CRMODE/SYNC statement is found.

SIMUL signifies that the parallel but independent execution of the different carriages and the related part program interpreters may be resumed, starting from the end of the current "common" section.

SYNC signifies that the movement of the carriages during measurement statement execution is to be synchronized, so that the sensor on each carriage passes through its respective x,y,z coordinate at the same time.

The CRMODE in effect is valid for all DMIS statements following it until it is redefined. CRMODE must be defined within a CRSLCT/ALL section.

Note 1: The CRMODE statement defines whether the part program (or part of it) will be executed with the carriages operating in the programmed sequence or simultaneously.

Note 2: The CRMODE/SEQNTL statement is useful for first time testing of part program execution, or as an alternative to collision zone programming for intricate paths between two carriages.

Note 3: The CRMODE/SYNC statement provides a mechanism which allows DMEs to perform synchronized movement of carriages during measurement cycles.

Note 4: When operating in MODE/MAN, a CRMODE/SYNC statement will be executed as a CRMODE/SIMUL statement.

6.31 CROSCL

Function: Used to enable or disable sensor motion to the negative (far) side of the rotary axis centerline of a DME with two linear and one rotary axis.

Input Formats:

can be: **CROSCL/var_1**

Output Formats:

can be: **CROSCL/var_1**

Where:

var_1 can be: **ON**
or: **OFF**

OFF signifies that the measurement sensor is positioned on the positive (near) side of the rotary axis centerline.

ON signifies that the measurement sensor is positioned on the negative (far) side of the rotary axis centerline.

The specified CROSCL is modal and remains in effect until it is changed by another CROSCL statement.

The CROSCL statement is passed to the output file when executed.

Note: When GOTO/x,y,z or when PTMEAS/x,y,z,0,0,1 are specified in a part program which is then run on a 2 linear and 1 rotary axis machine, there are two possible conversion results. CROSCL is used to select the desired possibility. Example: GOTO/0,10,20 can be converted to 10,20 with a rotation of 0 degrees or –10,20 with a rotation of 180 degrees. The latter case is selected by having CROSCL/ON in effect.

6.32 CRSLCT

Function: Defines the carriage associated to the subsequent part program code.

Input Formats:

can be: **CRSLCT/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **CR(lname1)**
or: **ALL**

ALL signifies that the following statements must be executed synchronously with respect to all carriage processes.

CR(lname1) is the label of a carriage, and the following statements must be executed by the process associated to the carriage defined by CR(lname).

The CRSLCT/ALL statement identifies a section of the DMIS program that is common to the carriages.

The CRSLCT/ALL statement suspends execution until all prior carriage processes are complete.

This synchronization allows the sharing of any variable data belonging to different carriage processes and to produce common data usable from any carriage process.

At the end of a common section, all carriage processes are resumed to continue the relevant part program section execution.

Variable data (measured points, measured elements, coordinate reference systems, any variable or array,..), cannot be shared between different carriage processes executing simultaneously without the risk of temporal data inconsistency. Any carriage process is free to access data belonging to itself and to use common or constant data.

Any data declared or created on a carriage section may be considered by the DME as belonging to the carriage itself. Data declared or created on a common section may be considered common, and so readable from any carriage.

Note 1: Each new carriage selection by means of the CRSLCT statement signifies that the previous carriage section is terminated. The CRSLCT statement is valid for any declaration or execution statement and does not only apply to positioning and measurement movements.

Note 2: Through the CRSLCT statement, any carriage process can easily recognize and execute relevant sections until a common synchronized section is found.

6.33 CUTCOM

Function: Defines a compensation or process adjustment for a manufacturing device, and assigns a label to it.

Input Formats:

can be: **CC(lname)=CUTCOM/MD(lname1),var_1**

Output Formats:

can be: **CC(lname)= MD(lname1),var_1**

Where:

var_1 can be: **ADJUST,TL(lname2),var_2,var_3,amt**
or: **PARAM,x,y,z,i,j,k**
or: **MATRIX,dx,dy,dz,ix,iy,iz,jx,jy,jz,kx,ky,kz**
or: **USERDF,'text'**

var_2 can be: **LEFT**
or: **RIGHT**

var_3 can be: **XYPLAN**
or: **YZPLAN**
or: **ZXPLAN**

i,j,k is the vector for the parameter adjustment.

ADJUST signifies that a cutter compensation adjustment for a specific tool is to follow.

amt is a real number representing the value for the amount of adjustment.

dx,dy,dz are the delta coordinates for the matrix adjustment.

ix,iy,iz are the i vectors for the x,y,z adjustment.

jx,jy,jz are the j vectors for the x,y,z adjustment.

kx,ky,kz are the k vectors for the x,y,z adjustment.

lname is an alphanumeric label name assigned to the compensation or process adjustment.

LEFT signifies that cutter compensation is directed in the left direction.

MATRIX signifies that the compensation to follow is in matrix form.

MD(lname1) is the label of the manufacturing device to be adjusted.

PARAM signifies that the compensation to follow is in parameter form.

RIGHT signifies that cutter compensation is directed in the right direction.

'text' is a text string, enclosed with apostrophes, that identifies the compensation or process adjustment. This parameter is particularly useful for applications in the electronics industry.

TL(lname2) is the label of the tool to be compensated.

USERDF signifies that the compensation to follow is user defined.

x,y,z are the coordinates for the parameter adjustment.

XYPLAN signifies the XYPLAN orientation.

YZPLAN signifies the YZPLAN orientation.

ZXPLAN signifies the ZXPLAN orientation.

The CUTCOM statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the CUTCOM definition.

Note: This statement is associated with the MFGDEV and TOOLDF statements for applications to adjust the manufacturing process based on inspection results.

6.34 CZONE

Function: Identifies a potential collision path between different carriages, and assigns a label to it.

Input Formats:

can be: **CZ(lname)=CZONE**

Output Formats:

can be: **None**

Where:

lname is an alphanumeric label name assigned to the collision path.

It is not necessary to suspend movement of a carriage in the overlap zone unless the path of the second carriage crosses the path of the first.

If one carriage has executed a CZSLCT/CZ(lname),ON statement and the other carriage encounters the same, the second carriage's movement is allowed to be suspended until the first carriage encounters the CZSLCT/CZ(lname),OFF statement. Movement of the second carriage can then be allowed to resume.

Note 1: In some multiple carriage DME configurations, there could be some overlap of measuring volume. This possibility is also apparent when sensor configurations are considered.

Note 2: The CZONE statement allows the user to define the critical paths of the carriages when the paths intersect.

Note 3: Refer to example reference A.9.

6.35 CZSLCT

Function: Delimits a possible collision path between different carriages.

Input Formats:

can be: **CZSLCT/CZ(lname1),var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **ON**
or: **OFF**

CZ(lname1) is the label of a previously defined collision path.

OFF signifies that the collision zone is released.

ON signifies that the following movements associated with the carriage have a collision hazard potential, if movements of a second carriage (with the same collision zone label) are allowed to occur.

A collision zone must begin and end inside a carriage part program section.

The following example sequence must be considered as an error, because the CRSLCT/CR(lname) statement occurs between the CZSLCT/CZ(lname1),ON and CZSLCT/CZ(lname1),OFF statements:

```
  executable statements
CZSLCT/CZ(lname1),ON
FEAT/PLANE,CART,2.5000,2.5000,0.0000,0.0000,0.0000,1.0000
MEAS/PLANE,F(DATUM_A),3
PTMEAS/CART,5.0000,0.0000,0.0000,0.0000,0.0000,1.0000
PTMEAS/CART,0.0000,5.0000,0.0000,0.0000,0.0000,1.0000
PTMEAS/CART,5.0000,5.0000,0.0000,0.0000,0.0000,1.0000
ENDMES
DAT(A)=DATDEF/FA(DATUM_A)
CRSLCT/CR(lname)
F(DATUM_B)=FEAT/CIRCLE,INNER,CART,2.0000,2.0000,0.0000$
,0.0000,0.0000,1.0000,0.6468
MEAS/CIRCLE,F(DATUM_B),4
PTMEAS/CART,1.0000,2.0000,0.0000,0.0000,0.0000,1.0000
PTMEAS/CART,2.0000,3.0000,0.0000,0.0000,0.0000,1.0000
PTMEAS/CART,3.0000,2.0000,0.0000,0.0000,0.0000,1.0000
PTMEAS/CART,2.0000,1.0000,0.0000,0.0000,0.0000,1.0000
ENDMES
CZSLCT/CZ(lname1),OFF

  executable statements
```

Note 1: The carriages of a multiple carriage DME operate on different measuring volumes, generally sharing a common volume in which collisions may occur. During program execution, a carriage performs movement in a collision zone only if the same collision zone is not active in another carriage process.

Note 2: Refer to example reference A.9.

6.36 DATDEF

Function: Assigns a datum label to one or more previously defined, measured or constructed feature(s).

Input Formats:

can be: **DATDEF/var_1**

Output Formats:

can be: **DATDEF/var_1**

Where:

var_1 can be: **FA(lname1),DAT(x)**
or: **FA(lname2),DAT(x-x)**
or: **F(lname3),DAT(x)**
or: **var_2,F(lname4),DAT(x)**

var_2 can be: **DATTRG(xn) var_3**

var_3 can be: **,var_2**
or: **does not exist**

DAT(x) is the datum label assigned to the feature. X is one or two upper case alpha characters.

DAT(x-x) is the compound datum label assigned to a FEAT/COMPOUND. Each x is one or two upper case alpha characters separated by a dash character.

DATTRG(xn) is the datum target used to establish a datum. xn is the label of the datum target (for example A1, A2, A3)

F(lname3) is the label of the feature nominal to be associated with the datum.

F(lname4) is the label of the previously defined feature nominal represented by one or more preceding datum targets.

FA(lname1) is the label of the previously measured or constructed feature to be associated with the datum.

FA(lname2) is the label of the previously constructed feature to be associated with the compound datum.

According to clause 5.3.2.6, if a feature has been measured with point buffering enabled, with a PTBUFF/ON statement, then its individual point data can be referenced by appending the [n] option to F(lname) or FA(lname), where n is the point index from the feature's programmed PTMEAS statements. Using this subscripted label option is equivalent to using an F(lname) or FA(lname) of a feature type POINT and therefore can only be used when a feature type POINT is appropriate.

The DATDEF statement is passed to the output file at the time it is executed.

Note 1: Datums are referenced in the DATSET, ROTATE, TRANS, and TOL statements. This statement simply assigns a datum label to a previously measured feature or features.

(For proper datum label usage refer to ASME Y14.5M-1994)

Note 2: The compound datum, DAT(x-x), is a FEAT/COMPOUND

Note 3: The datum, DAT(x), can be established by one or many datum targets.

6.37 DATSET

Function: Defines and activates a datum set, or part coordinate system, and assigns a label to it. Datum sequence per ASME Y14.5M-1994.

Input Formats:

can be: **D(lname)=DATSET/var_1**

Output Formats:

can be all: **D(lname)=DATSET/var_1**
DA(lname)=DATSET/TRMATX,a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3

Where:

var_1 can be: **MCS**
or: **var_2**
or: **TRMATX,a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3**
or: **DRF,TA(lname) var_3,var_4,var_4**

var_2 can be: **DAT(x),var_4 var_5 var_6 var_7**
or: **DAT(x),var_4 var_5 var_7 var_6**
or: **DAT(x) var_8 var_6 ,DAT(x),var_4 var_5**
or: **DAT(x) var_8,DAT(x),var_4 var_5 var_6**

var_3 can be: **,UPTIER**
or: **,LOTIER**
or: **does not exist**

var_4 can be: **XDIR**
or: **-XDIR**
or: **YDIR**
or: **-YDIR**
or: **ZDIR**
or: **-ZDIR**

var_5 can be: **,XORIG var_5**
or: **,YORIG var_5**
or: **,ZORIG var_5**
or: **does not exist**

var_6 can be: **,DAT(x),var_4 var_5**
or: **,DAT(x) var_8**
or: **does not exist**

var_7 can be: **,DAT(x) var_8**
or: **does not exist**

var_8 can be: **,XORIG var_5**
or: **,YORIG var_5**
or: **,ZORIG var_5**

a1,a2,a3, b1,b2,b3, c1,c2,c3, d1,d2,d3 satisfy the following transformation equations:

(a1)x + (b1)y + (c1)z + d1 = x'
(a2)x + (b2)y + (c2)z + d2 = y'
(a3)x + (b3)y + (c3)z + d3 = z'

Where: x', y', and z' are the coordinates in the current part coordinate system (after the transformation has been applied) of any point, and x, y, and z are the coordinates in the previous coordinate system of the same point.

DAT(x) is the datum label used to define the part coordinate system axis that follows.

DRF signifies that the datum reference frame used to calculate the last OUTPUT or EVAL of the TA(lname) tolerance is to be created and activated.

lname is the label name assigned to the part coordinate system. A new label is required for each

DATSET statement.

LOTIER signifies the datum reference frame used by the lower tier of a composite tolerance is to be created and activated.

MCS signifies that the DME's machine coordinate system is to be reset and activated, causing the previous part coordinate system (if any), to be cancelled.

TA(lname) is the label of the previously evaluated tolerance with DRF (for example TOL/POS, TOL/PROFS).

TRMATX signifies that the alignment and transformation information from the previous coordinate system is being output or input.

UPTIER signifies the datum reference frame used by the upper tier of a composite tolerance is to be created and activated.

XDIR signifies that the positive x direction is given by the direction vector of the preceding feature. Refer to the paragraph below on DATSET/DRF.

-XDIR signifies that the negative x direction is given by the direction vector of the preceding feature. Refer to the paragraph below on DATSET/DRF.

XORIG signifies that the x-component of the datum is used to establish the x-axis origin.

YDIR signifies that the positive y direction is given by the direction vector of the preceding feature. Refer to the paragraph below on DATSET/DRF.

-YDIR signifies that the negative y direction is given by the direction vector of the preceding feature. Refer to the paragraph below on DATSET/DRF.

YORIG signifies that the y-component of the datum is used to establish the y-axis origin.

ZDIR signifies that the positive z direction is given by the direction vector of the preceding feature. Refer to the paragraph below on DATSET/DRF.

-ZDIR signifies that the negative z direction is given by the direction vector of the preceding feature. Refer to the paragraph below on DATSET/DRF.

ZORIG signifies that the z-component of the datum is used to establish the z-axis origin.

XORIG, YORIG and ZORIG can only apply once in any given DATSET statement.

When using the TRMATX minor word in the input format of DATSET statement, the vectors A, B, and C defined by the first nine parameters following TRMATX (A=(a1,a2,a3), B=(b1,b2,b3), C=(c1,c2,c3)) must each have length 1, each of them must be perpendicular to the other two, and A,B,C (in that order) must form a right handed coordinate system.

If the DATSET/DRF statement is specified, the coordinate system that will be created is the tolerance coordinate system used to calculate the last OUTPUT or EVAL of the TA(lname). The two var_4 parameters define respectively how the primary and the secondary directions will be represented in the created coordinate system.

The DATSET statement is passed to the output file at the time it is executed. XORIG, YORIG and ZORIG can only apply once in any given DATSET.

Note 1: If a DATSET statement is not fully defined with two directions and three origins then the orientation and/or offset of the unconstrained axes are implementation specific.

Note 2: The DATSET statement provides for the establishment of the orientation, alignment, and origin, of the part coordinate system. The TRANS statement can also establish the origin of the part coordinate system, or translate it to establish a new one. The ROTATE statement can also establish the alignment of the part coordinate system, or rotate it to establish a new one. Together or in any combination, the DATSET, ROTATE, and TRANS statements may be combined to establish part coordinate systems. Refer to clause 5.3.6 and sub clauses.

6.38 DATTRGDEF

Function: Assigns a datum target label to one or more previously defined, measured or constructed feature(s).

Input Formats:

can be: **DATTRGDEF/var_1**

Output Formats:

can be: **DATTRGDEF/var_1**

Where:

var_1 can be: **FA(lname1) var_2,DATTRG(xn)**
or: **F(lname2) var_2,DATTRG(xn)**

var_2 can be: **,FA(lname1) var_2**
or: **,F(lname2) var_2**
or: **does not exist**

DATTRG(xn) is the datum target label. xn is the label of the datum target (for example A1, A2, A3)

F(lname2) is the label of the feature nominal to be associated with the datum target.

FA(lname1) is the label of the previously defined, measured or constructed feature to be associated with the datum target.

According to clause 5.3.2.6, if a feature has been measured with point buffering enabled, by a PTBUFF/ON statement, then its individual point data can be referenced by appending the [n] option to F(lname) or FA(lname), where n is the point index from the feature's programmed PTMEAS statements. Using this subscripted label option is equivalent to using an F(lname) or FA(lname) of a feature type POINT and therefore can only be used when a feature type POINT is appropriate.

The DATTRGDEF statement is passed to the output file when executed.

Note: Datum targets designate specific point, line, or area features that are used to establish a single datum.

(For proper datum target label usage refer to ASME Y14.5M-1994)

6.39 DECL

Function: Declares variable names and their corresponding data type to be used in a program.

Input Formats:

can be: **DECL/var_1 var_2 var_3**

Output Formats:

can be: **None**

Where:

var_1 can be: **COMMON,**
or: **GLOBAL,**
or: **LOCAL,**
or: **does not exist**

var_2 can be: **BOOL**
or: **INTGR**
or: **LONG**
or: **REAL**
or: **DOUBLE**
or: **CHAR,n**
or: **VECTOR**

var_3 can be: **,varname var_4**
or: **,varname[index1 var_5] var_4**

var_4 can be: **var_3**
or: **does not exist**

var_5 can be: **,indexn var_5**
or: **does not exist**

BOOL signifies a Boolean data type that may only have the value of .TRUE. or .FALSE..

CHAR signifies a character string data type.

COMMON signifies variables that are defined system wide (for example, environmental variables) and can be addressed by all programs within a system. The meaning and implementation of COMMON variables is DME specific.

DOUBLE signifies a real number data type requiring a minimum of 8 bytes of storage in floating-point format.

GLOBAL signifies variables that remain active during a program execution and can be accessed from anywhere within the program's scope.

index1 is a positive integer representing the number of elements for the first dimension of an array.

indexn is a positive integer representing the number of elements for the nth dimension of a multidimensional array.

INTGR signifies an integer number data type requiring a minimum of 2 bytes of storage.

LOCAL signifies that variables are active only in the definition level (for example, macro, module) and serve as the buffer memory of values. After leaving the definition level, these variables no longer exist.

LONG signifies an integer number data type requiring a minimum of 4 bytes of storage.

n is a positive integer that is the number of characters assigned to the variable.

REAL signifies a real number data type requiring a minimum of 4 bytes of storage in floating-point format.

varname is the variable name of the declared data type, and is up to 16 characters in length.

VECTOR signifies an ordered collection of three DOUBLE values. Vectors are always stored internally

as Cartesian values, and no implicit conversion between coordinate systems occurs.

When a character string data type is declared, the data assigned to the variable name is enclosed with apostrophes. The 'n' value does not include the apostrophes.

Any number of variables of the same type can be declared with a single DECL statement.

All DMIS variables must be declared before they are used. Variable names consist of 1 to 16 alphanumeric characters and underscores. The first character of a name must be an alpha character. Variable names must not use DMIS reserved words, refer to 5.1.2.3. If var_1 is not specified, GLOBAL is implied for a DMISMN and LOCAL is implied for a MACRO. A variable array index (base) begins at the number 1.

6.40 DECPL

Function: Specifies the number of decimal places to the right of the decimal point, to output for specified parameters.

Input Formats:

can be: **DECPL/var_1**

Output Formats:

can be: **DECPL/var_1**

Where:

var_1 can be: **ALL,var_2**
or: **var_3,var_2 var_4**

var_2 can be: **DEFALT**
or: **n**

var_3 can be: **ANGLE**
or: **DIST**
or: **HUMID**
or: **DEV**
or: **TEMP**
or: **VEC**

var_4 can be: **,var_3,var_2 var_4**
or: **does not exist**

ALL signifies selection of all output parameters.

ANGLE signifies angular output parameters.

DEFALT signifies the default DME number of digits to the right of the decimal point to be used for the selected output parameters.

DEV signifies selection of tolerance and deviation output parameters.

DIST signifies selection of linear output parameters.

HUMID signifies selection of humidity output parameters.

n is a positive integer number representing the desired number of digits to the right of the decimal point to output for the selected parameter.

TEMP signifies selection of temperature output parameters.

VEC signifies selection of vector output parameters.

DEV sets the decimal places for ANGDEC and ANGRAD and sets the precision of the arcseconds for ANGDMS.

The DECPL statement is passed to the output file when executed.

6.41 DELETE

Function: Deletes part coordinate system datum sets, sensor calibration data, and measured feature actual data that was previously saved.

Input Formats:

can be: **DELETE/var_1 var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **D(lname1)**
or: **DA(lname2)**
or: **S(lname3)**
or: **SA(lname4)**
or: **FA(lname5)**
or: **RT(lname6)**
or: **ALLSA var_3**

var_2 can be: **,DID(lname7)**
or: **does not exist**

var_3 can be: **,EXCEPT,SA(lname4) var_4**
or: **does not exist**

var_4 can be: **,SA(lname4) var_4**
or: **does not exist**

ALLSA signifies that all calibrated sensors will be deleted except those following the optional EXCEPT minor word.

D(lname1) is the label of the nominal part coordinate system that is to be deleted.

DA(lname2) is the label of the actual part coordinate system that is to be deleted.

DID(lname7) is the device identification label, assigned by the DEVICE statement to the system device.

EXCEPT signifies that the specified calibrated sensors will not be deleted.

FA(lname5) is the label of the measured feature that is to be deleted.

RT(lname6) is the label of the rotary table that is to be deleted.

S(lname3) is the label of the sensor that is to be deleted. This only applies to programs written for ANSI/CAM-I 101 1990.

SA(lname4) is the label of the calibrated sensor that is to be deleted, or if the ALLSA and EXCEPT minor words are used, which calibrated sensor is not to be deleted.

6.42 DEVICE

Function: Assigns a label to a system device.

Input Formats:

can be: **DID(lname)=DEVICE/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **PRINT,'devicename'**
or: **TERM,'devicename'**
or: **COMM,'devicename'**
or: **STOR,'filename'**
or: **INCR,'filemask'**

COMM signifies that a communications port is being identified.

'devicename' is the name of the system device, enclosed with apostrophes, (for example, printer port, terminal display, or communication port id). This name contains printable characters.

'filemask' is the filename, enclosed with apostrophes, masked with one or more consecutive '?'. The DME will automatically create the incremented filename by replacing the '?'s within the filemask with base 10 integer values, left padded with zeros. The incremented filename is the next in the sequence derived from existing output filenames matching the filemask. The sequence starts with '1'.

'filename' is the name of the file, enclosed with apostrophes, to be used to store information.

INCR signifies that a storage device is being identified with an incremented filename.

lname is an alphanumeric label name assigned to the defining device.

PRINT signifies that a printer device is being identified.

STOR signifies that a storage device is being identified.

TERM signifies that a terminal device is being identified.

The DEVICE statement defines the device identification label. The device is not accessible for input/output until the OPEN statement is used.

Note: For example reference, refer to A.10

6.43 DFTCAS

Function: Provides an optional default case in a SELECT...ENDSEL block, if and only if none of the CASE statements returns true, the statement(s) from DFTCAS to ENDCAS are executed. If present, DFTCAS must come after all CASE...ENDCAS blocks in the SELECT...ENDSEL block.

Input Formats:

can be: **DFTCAS**

Output Formats:

can be: **None**

Note: For example reference, refer to A.29.

6.44 DISPLY

Function: Specifies the control and format of output data to DME default device(s).

Input Formats:

can be: **DISPLY/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **var_2**
or: **OFF**

var_2 can be: **PRINT,var_3 var_4**
or: **TERM,var_3 var_4**
or: **STOR,var_3 var_4**
or: **COMM,var_3 var_4**

var_3 can be: **DMIS**
or: **V(lname1)**
or: **DMIS,V(lname1)**

var_4 can be: **,var_2**
or: **does not exist**

COMM signifies that the output format(s) specified by the following minor word(s) will be output to the auxiliary communications port.

DMIS signifies that the output will be in DMIS format.

OFF signifies that output will not be sent to any DISPLY device.

PRINT signifies that the output format(s) specified by the following minor word(s) will be output as a printed report.

STOR signifies that the output format(s) specified by the following minor word(s) will be output to magnetic storage.

TERM signifies that the output format(s) specified by the following minor word(s) will be output to the video terminal.

V(lname1) is the label of the output label in vendor format, as specified by the VFORM statement.

The DISPLY statement may be used more than once in a DMIS input program. VFORM must be previously defined when used in a DISPLY statement. Refer to clause 5.1.9 (Data output) and sub clauses for additional information.

There is no interaction between DISPLY devices and DID devices.

6.45 DMEHW

Function: Communicates specific DME hardware information to a servo-driven DME to control execution of move and measurement statements.

Input Formats:

can be: **DMEHW/var_1**

Output Formats:

can be: **DMEHW/var_1**

Where:

var_1 can be: **CONTIN**
or: **PAUSE**
or: **SINGLE**
or: **AUTO**
or: **JOINTCONFIG var_2**

var_2 can be: **,LEFTY var_3**
or: **,RIGHTY var_3**
or: **var_3**

var_3 can be: **,ABOVE var_4**
or: **,BELOW var_4**
or: **var_4**

var_4 can be: **,FLIP var_5**
or: **,NOFLIP var_5**
or: **var_5**

var_5 can be: **,'jointparam'**
or: **,'jointparam' var_5**
or: **does not exist**

AUTO signifies that DMEHW/SINGLE is to be cleared.

CONTIN signifies that the automatic execution of motion statements in an inspection program is to be resumed. The DMEHW/CONTIN statement can be in the inspection program, or it can be issued by the operator at the DME.

'jointparam' is a DME specific parameter that specifies a joint kinematic configuration. Each parameter can take one of two values, and as such represents a binary selection.

PAUSE signifies that the DME program is to halt execution of motion statements until a DMEHW/CONTIN statement is issued. The DME will continue to execute non-motion statements in the input file until a motion statement is encountered. All execution of statements will then stop.

JOINTCONFIG signifies for a DME with linked chain kinematic structures, i.e. robots, the extra parameters that define which of the inverse kinematic solutions should be chosen. Typically there are a number of solutions for each tool position and orientation that need to be disambiguated.

SINGLE signifies that the DME is to execute in single step mode; it executes each motion statement individually and waits for a DMEHW/CONTIN statement after each is complete. SINGLE has the same effect as issuing a DMEHW/PAUSE statement after each motion statement. SINGLE is cleared and continuous execution is resumed when a DMEHW/AUTO statement is issued. A DMEHW/AUTO statement can be issued in the program, or by the operator at the DME.

The DMEHW statement is passed to the output file when executed.

Note: A DMEHW statement controls the execution of the following motion and measure statements: CALIB, DMESW, GOTARG, GOHOME, GOTO, MEAS, PAMEAS, PTMEAS, RMEAS, ROTAB, and SNSLCT

6.46 DMEID

Function: Defines the identification of a Dimensional Measuring Device, and assigns a label to it.

Input Formats:

can be: **DI(lname)=DMEID/'text'**

Output Formats:

can be: **DI(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the Dimensional Measuring Device.

'text' is a text string, enclosed with apostrophes, that identifies the DME.

The DMEID statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the dimensional measuring device label name.

6.47 DMESW

Function: Used to control data or the processing of data sent in the input file to the DME.

Input Formats:

can be: **DMESW/var_1**

Output Formats:

can be: **DMESW/var_1**

Where:

var_1 can be: **COMAND,'command'**
or: **CONTIN**
or: **DELAY,n**
or: **PAUSE**

'command' is the command sent to the receiving system, enclosed with apostrophes.

COMAND signifies that the string of alphanumeric characters, enclosed with apostrophes, that follow will be interpreted as a DME-specific command. This is used to code DME-specific instructions that are not within the capability of the DMIS interface.

CONTIN signifies that the DME is to continue processing data. All data received after the DMESW/PAUSE statement and before the next DMESW/CONTIN statement are ignored by the DME.

DELAY signifies that the DME is to delay processing data for 'n' seconds.

n is a positive integer, that is the length of time in seconds (for example, DMESW/DELAY,600)

PAUSE signifies that the DME is to stop processing data. All data received after the DMESW/PAUSE statement is encountered and before the next DMESW/CONTIN statement is encountered, are ignored by the DME.

The DMESW statement is passed to the output file when executed.

6.48 DMESWI

Function: Defines the identification of the DME's software, and assigns a label to it.

Input Formats:

can be: **DS(lname)=DMESWI/'text'**

Output Formats:

can be: **DS(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the DMEs software.

'text' is a text string, enclosed with apostrophes, that identifies the DME's software.

The DMESWI statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined DME software label name.

6.49 DMESWV

Function: Defines the identification of the DME's software version, and assigns a label to it.

Input Formats:

can be: **DV(lname)=DMESWV/'text'**

Output Formats:

can be: **DV(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the DME's software version.

'text' is a text string, enclosed with apostrophes, that identifies the DME's software version.

The DMESWV statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined DME software version label name.

6.50 DMIS

Function: Defines when statements are processed, or ignored by the DME.

Input Formats:

can be: **DMIS/var_1**

Output Formats:

can be: **DMIS/var_1**

Where:

var_1 can be: **ON**
or: **OFF**

OFF signifies that the processing of statements following this statement is OFF, or ignored by the DME.

ON signifies that the processing of statements following this statement is ON, or continued.

Unlike the DMEHW statement where only the execution of motion statements can be halted while non-motion statements continue to be executed, this statement causes all statements following a DMIS/OFF statement to be ignored by the DME.

The statements between the DMIS/OFF and DMIS/ON statements can, however, be processed by the CAD system to meet specific application needs. For example, Computer Aided Engineering data can be formatted to identify the probe extension definitions required to define a specific probe configuration.

The information between the DMIS/OFF and DMIS/ON statements is passed to the output file when encountered.

The DMIS statement is passed to the output file when executed.

6.51 DMISMD

Function: External program identification for a DMIS input module.

Input Formats:

can be: **DMISMD/'module_id',version var_1**

Output Formats:

can be: **DMISMD/'module_id',version var_1**

Where:

var_1 can be: **,var_2, conform_level var_1**
or: **does not exist**

var_2 can be: **CT**
or: **FX**
or: **IP**
or: **MC**
or: **MU**
or: **PM**
or: **QI**
or: **RY**
or: **SF**
or: **TW**

conform_level is a base 10 positive integer as defined in clause 2.6.

CT signifies that the module conforms to Contact Scanning Application Profile Addendum

FX signifies that the program conforms to Five-Axis Scanning Application Profile Addendum

IP signifies that the module conforms to IPV Application Profile Addendum

MC signifies that the module conforms to Multiple Carriage Application Profile Addendum

'module_id' is a series of printable UTF8 characters, enclosed with apostrophes, identifying the module.

MU signifies that the module conforms to Measurement Uncertainty Application Profile Addendum

PM signifies that the module conforms to Prismatic Application Profile

QI signifies that the module conforms to QIS Application Profile Addendum

RY signifies that the module conforms to Rotary Table Application Profile Addendum

SF signifies that the module conforms to Soft Gauge Application Profile Addendum

TW signifies that the module conforms to Thin Wall Application Profile

version defines the 'major.minor' revision of DMIS.

The format for 'major.minor' revision of DMIS will be XX.x, where XX is the major version number and x is the minor version number. The major version number shall be base 10 positive integer values, left padded with zeros. The minor version number shall be a single base 10 positive integer value or zero.

The DMISMD statement is passed to the output file with execution of the CALL statement that invokes this module.

Note: The module_id string of UTF8 characters must begin and end with an apostrophe. The DMISMD statement designates the beginning of a DMIS input module. It must be the first executable statement in the DMIS input module. Refer to the DMISMN statement, for additional information or A.11 for an example. The DMIS input module will end with an ENDFIL statement. Program execution will be transferred back to the calling program at the line following the CALL statement.

6.52 DMISMN

Function: Specifies program identification, DMIS version, and optional DMIS conformance class for a DMIS input program.

Input Formats:

can be: **DMISMN/'text',version var_1**

Output Formats:

can be: **DMISMN/'text',version var_1**

Where:

var_1 can be: **,var_2, conform_level var_1**
or: **does not exist**

var_2 can be: **CT**
or: **FX**
or: **IP**
or: **MC**
or: **MU**
or: **PM**
or: **QI**
or: **RY**
or: **SF**
or: **TW**

conform_level is a base 10 positive integer as defined in clause 2.6.

CT signifies that the program conforms to Contact Scanning Application Profile Addendum

FX signifies that the program conforms to Five-Axis Scanning Application Profile Addendum

IP signifies that the program conforms to IPV Application Profile Addendum

MC signifies that the program conforms to Multiple Carriage Application Profile Addendum

MU signifies that the program conforms to Measurement Uncertainty Application Profile Addendum

PM signifies that the program conforms to Prismatic Application Profile

QI signifies that the program conforms to QIS Application Profile Addendum

RY signifies that the program conforms to Rotary Table Application Profile Addendum

SF signifies that the program conforms to Soft Gauge Application Profile Addendum

'text' is a series of printable UTF8 characters, enclosed with apostrophes.

TW signifies that the program conforms to Thin Wall Application Profile

version defines the 'major.minor' revision of DMIS.

The format for 'major.minor' revision of DMIS will be XX.x, where XX is the major version number and x is the minor version number. The major version number shall be base 10 positive integer values, left padded with zeros. The minor version number shall be a single base 10 positive integer value or zero.

Note: This string of UTF8 characters must begin and end with an apostrophe. When the string of characters must extend to another line, use a single dollar sign, '$', at the end of the line. The DMISMN statement designates the beginning of the main input program and it must be the first line of executable code in the DMIS input program. Refer to the example shown in A.11.

6.53 DO

Function: provides the capability of repeating a sequence of instructions based on an initial and limit value at a specified increment.

Input Formats:

can be: **DO/index,initial,limit var_1**

Output Formats:

can be: **None**

Synopsis: **DO/**index,initial,limit,increment

executable statement(s)

ENDDO

Where:

var_1 can be: **,increment**
or: **does not exist**

increment is a non-zero integer representing the increment value. If omitted, the increment is 1.

index is a previously declared variable representing the DO loop index.

initial is an integer representing the initial value of the DO loop index variable.

limit is an integer representing the limit value of the DO loop index variable.

The variables above must be an integer or integer variable. When DO loops are nested, care should be exercised to include the corresponding ENDDO statements for every DO statement.

6.54 ELSE

Function: Provides an optional branch within an IF...ENDIF block.

Input Formats:

can be: **ELSE**

Output Formats:

can be: **None**

The ELSE statement follows an associated IF statement. If the result of the IF statement is false, program control is transferred to the statement following the ELSE statement.

6.55 ENDAT

Function: Signifies the end of a data stream.

Input Formats:

can be: **None**

Output Formats:

can be: **ENDAT**

Used to terminate data in a raw data listing.

6.56 ENDCAS

Function: Indicates the end of the current CASE...ENDCAS or DFTCAS...ENDCAS block, initiated with the CASE or DFTCAS statement.

Input Formats:

can be: **ENDCAS**

Output Formats:

can be: **None**

6.57 ENDDO

Function: Indicates the end of a DO...ENDDO block, initiated with a DO statement.

Input Formats:

can be: **ENDDO**

Output Formats:

can be: **None**

6.58 ENDFIL

Function: Specifies the end of the program or input module.

Input Formats:

can be: **ENDFIL**

Output Formats:

can be: **ENDFIL**

This is always the last statement in a program or input module.

At the end of a program, the ENDFIL statement is passed to all opened DMIS output files, and all output files and devices are closed.

At the end of an input module, the ENDFIL statement transfers program execution back to the calling program at the line following the CALL statement.

The ENDFIL statement will have the same effect as the execution of a CLOSE/END statement on any opened DMIS output devices, will have the same effect as the execution of a CLOSE/KEEP statement on all other opened storage devices, and will have the same effect as a CLOSE statement on all other devices open at program termination.

The ENDFIL statement is passed to the output file when executed.

6.59 ENDGO

Function: Indicates the end of a GOTARG...ENDGO block, initiated with the GOTARG statement.

Input Formats:

can be: **ENDGO**

Output Formats:

can be: **None**

6.60 ENDIF

Function: Indicates the end of an IF...ENDIF or IF...ELSE...ENDIF block, initiated with the IF statement.

Input Formats:

can be: **ENDIF**

Output Formats:

can be: **None**

6.61 ENDMAC

Function: Terminates a MACRO sequence definition, initiated with the M(lname)=MACRO/... statement.

Input Formats:

can be: **ENDMAC**

Output Formats:

can be: **None**

6.62 ENDMES

Function: Indicates the end of a CALIB...ENDMES, MEAS...ENDMES, or RMEAS...ENDMES block, initiated with the CALIB, MEAS, or RMEAS statements.

Input Formats:

can be: **ENDMES**

Output Formats:

can be: **None**

6.63 ENDSEL

Function: Indicates the end of a SELECT...CASE...ENDCAS...DFTCAS...ENDCAS...ENDSEL block, initiated with the SELECT statement.

Input Formats:

can be: **ENDSEL**

Output Formats:

can be: **None**

6.64 ENDSIMREQT

Function: Indicates the end of a simultaneous requirement block initiated with the SIMREQT statement and causes the simultaneous execution of all EVAL and OUTPUT statements in the block.

Input Formats:

can be: **ENDSIMREQT**

Output Formats:

can be: **SRA(lname)=ENDSIMREQT/TRMATX,a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3**

Where:

a1,a2,a3, b1,b2,b3, c1,c2,c3, d1,d2,d3 satisfy the following transformation equations:

(a1)x + (b1)y + (c1)z + d1 = x'
(a2)x + (b2)y + (c2)z + d2 = y'
(a3)x + (b3)y + (c3)z + d3 = z'

Where: x', y', and z' are the coordinates in the current part coordinate system (after the transformation has been applied) of any point, and x, y, and z are the coordinates in the previous coordinate system of the same point.

TRMATX signifies that the alignment and transformation information from the current coordinate system is being output or input.

lname is alphanumeric label name assigned to the simultaneous requirement.

The ENDSIMREQT statement results in the simultaneous execution of all EVAL and OUTPUT statements per the common datum reference frame, defined by the tolerances, with datum sequences and datum material condition modifiers. The resulting transformation matrix representing the actual datum reference frame mobility used in the simultaneous evaluation is output.

The label name in the output format of ENDSIMREQT is the same as the label name in the SIMREQT statement for SIMREQT...ENDSIMREQT block.

The ENDSIMREQT statement is passed to the output file when executed.

6.65 ENDXTN

Function: Indicates the end of an external declaration, XTERN...ENDXTN block, initiated with the XTERN statement.

Input Formats:

can be: **ENDXTN**

Output Formats:

can be: **None**

6.66 EQUATE

Function: Recovers the current part coordinate system by equating two datum sets or part coordinate systems, or associates a part coordinate system with a CAD coordinate system.

Input Formats:

can be: **EQUATE/DA(lname1),DA(lname2)**

or: **EQUATE/DA(lname1),CADCS,DID(lname3),var_1**

Output Formats:

can be all: **EQUATE/DA(lname1),DA(lname2)**
EQUATE/TRMATX,a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3

or: **EQUATE/DA(lname1),CADCS,a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3**

Where:

var_1 can be: **a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3**
or: **'text'**

a1,a2,a3, b1,b2,b3, c1,c2,c3, d1,d2,d3 satisfy the following transformation equations:

(a1)x + (b1)y + (c1)z + d1 = x'
(a2)x + (b2)y + (c2)z + d2 = y'
(a3)x + (b3)y + (c3)z + d3 = z'

Where: x', y', and z' are the coordinates in the current part coordinate system (after the transformation has been applied) of any point, and x, y, and z are the coordinates in the previous coordinate system of the same point.

CADCS signifies that a CAD coordinate system will be equated to a part coordinate system.

DA(lname1) is the label of the datum set or coordinate system to be equated from.

DA(lname2) is the label of the datum set or coordinate system to be equated to.

DID(lname3) is the device identification label of an opened CAD file containing coordinate system definitions.

'text' is a series of printable UTF8 characters, enclosed in apostrophes, that signifies the identifier of the CAD coordinate system name.

TRMATX signifies that the alignment and transformation information is being output.

The EQUATE statement creates a transformation from DA(lname1) to DA(lname2) and updates previously measured features and coordinate systems by the transformation when referenced such that the effect is that previously measured features appear to have been measured in a single part coordinate system.

The EQUATE statement is used to change the position and/or orientation of a part while retaining previous dimensional information. For example, to measure features on a part that are not accessible from a single part orientation.

The EQUATE statement is passed to the output file when executed.

Note 1: Refer to A.13 for an example of the use of the EQUATE statement.

Note 2: The output format containing the CADCS minor word is only used with the input format containing the CADCS minor word.

6.67 ERROR

Function: Used to define the handling of DME error codes

Input Formats:

can be: **ERROR/var_1**

Output Formats:

can be: **ERROR/var_1**

Where:

var_1 can be: **(jumptarget),var_2**
or: **OFF**
or: **AUTO,var_2**

var_2 can be: **ALL**
or: **ILLEGALTOUCH**
or: **NOTOUCH**
or: **ercode**

ALL signifies that any error will cause program execution to branch to the specified jumptarget.

AUTO signifies that a DME specific search and recovery algorithm to handle the error is to be enabled.

ercode is a number representing a DME specific error code. Error codes are identified in the characterization file.

ILLEGALTOUCH signifies that if an illegal or unexpected probe trigger occurs, program execution will branch to the specified jumptarget.

(jumptarget) is the location to which the program execution control will be transferred, when an error condition is encountered.

NOTOUCH signifies that if no touch is detected within the search distance specified by the last SNSET/SEARCH statement, program execution will branch to the specified jumptarget.

OFF signifies that error handling is to be turned off.

If an error occurs then the Intrinsic Function SERROR() returns the error code ('ILLEGALTOUCH', 'NOTOUCH' or ercode as specified in the characterization file) as a type CHAR, SCFEAT() contains the feature label and SMODE() returns the state of the MODE statement.

The appropriate ERROR statement is passed to the output file when an error condition is encountered.

Note 1: The DME error code for ILLEGALTOUCH and NOTOUCH can be obtained using the intrinsic functions SILTCH() and SENSNOTOUCH().

Note 2: This statement provides the capability of handling errors that would otherwise suspend processing when error conditions are encountered while in automatic or programmed mode. Typically, this statement is placed in the beginning of a program. When an error condition is encountered that is associated with the error code in this statement, control of the program is passed to the line with the statement (jumptarget).

6.68 EVAL

Function: Evaluates named feature(s) to the tolerance(s).

Input Formats:

can be: **EVAL/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **FA(lname1),T(lname2) var_2**
or: **FA(lname3),var_3,T(lname4)**
or: **var_4,FA(lname5),T(lname4)**
or: **KC(lname6)**

var_2 can be: **,T(lname2) var_2**
or: **does not exist**

var_3 can be: **F(lname7)**
or: **FA(lname5)**
or: **DAT(lname8)**

var_4 can be: **F(lname9)**
or: **DAT(lname8)**

DAT(lname8) is the datum label of the datum associated with a measured feature in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

F(lname7) is the label of the feature nominal associated with a measured feature in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

F(lname9) is the label of the feature nominal associated with a measured feature in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

FA(lname1) is the label of the feature actual to be evaluated.

FA(lname3) is the label of the first feature actual to be associated with a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB. When used in the EVAL/FA(lname1), F(lname7), T(lname2) form, FA(lname1) represents the measured feature associated with a feature nominal F(lname7) in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB. When used in the EVAL/FA(lname1), DAT(lname2), T(lname2) form, FA(lname1) represents the feature actual associated with a datum DAT(lname2) in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

FA(lname5) is the label of the second feature to be associated with a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB. When used in the EVAL/F(lname9), FA(lname3), T(lname2) form, FA(lname3) represents the measured feature associated with a feature nominal F(lname9) in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB. When used in the EVAL/DAT(lname8), FA(lname3), T(lname2) form, FA(lname3) represents the feature actual associated with a datum DAT(lname8) in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

KC(lname6) is the label of the key characteristic label defined in the KEYCHAR statement that has associated feature(s) and tolerance(s), and may also have a key characteristic designation of CRITICAL, MAJOR, or MINOR.

T(lname2) is the label of the tolerance associated with the FA(lname).

T(lname4) is the label of the relationship tolerance, for example, TOL/ANGLB or TOL/DISTB, associated with the measured features.

When a tolerance is specified, the execution of the EVAL statement results in the computation of the tolerance actual(s), for example, TA(lname). The labels of these actuals can then be used in conjunction with the following statements: OBTAIN, VALUE, IF, CUTCOM, or OUTPUT. The pattern defined by the FEAT/PATERN statement definition can be used to evaluate the composite positioning defined by the TOL/COMPOS statement tolerance for several features associated with one tolerance.

Note: The EVAL statement is useful to generate actuals required for selective processing, or IPV, but not necessarily intended for output.

6.69 EXTENS

Function: Defines a sensor extension component, and assigns a label to it.

Input Formats:

can be: **SX(lname)=EXTENS/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **dx,dy,dz**
or: **VEC,i,j,k,length**

dx,dy,dz is the nominal offset from the mounting point for the extension to the mounting point for the subsequent sensor component.

i,j,k is the direction perpendicular to the surface of the extension mounting point for the subsequent sensor component.

lname is an alphanumeric label name assigned to the sensor extension.

length is a real number representing the length of the extension.

VEC signifies that the extension is defined using a direction vector and a length along the vector.

Note: Refer to example A.30.2 and (Figure A.7 — Multi-probes from multiple ports on the head) for illustration and sample code used to define a sensor component using the EXTENS statement.

6.70 EXTFIL

Function: Declares an external file definition.

Input Formats:

can be: **EXTFIL/var_1,'filename'**

Output Formats:

can be: **None**

Where:

var_1 can be: **DMIS**
or: **DME**

DME signifies an external file containing DME control program language routines.

DMIS signifies a DMISMD external file typically containing DMIS MACRO routines.

'filename' is a fully qualified filename of the external file including parent directory names, enclosed with apostrophes.

For example, '/usr/dme/mysubfile'

All EXTFIL statement definitions must occur within a XTERN...ENDXTN block. An EXTFIL statement definition will be checked to verify that the file exists. To satisfy any CALL/EXTERN, DMIS macros, the defined EXTFIL/DMIS files will be searched according to its definition precedence until the macro is found.

6.71 FEAT/ARC (input format 1)

Function: Defines a nominal circular arc or constructs an actual circular arc, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/ARC,var_1,var_2,i,j,k,rad,ang1,ang2 var_3**

or: **FA(lname)=FEAT/ARC,var_1,var_2,i,j,k,rad,ang1,ang2 var_3**

Output Formats:

can be: **F(lname)=FEAT/ARC,var_1,var_2,i,j,k,rad,ang1,ang2 var_3**

or: **FA(lname)=FEAT/ARC,var_1,var_2,i,j,k,rad,ang1,ang2 var_3**

or:
```
FA(lname)=FEAT/ARC,RAWDAT
 /rx1,ry1,rz1
 /rx2,ry2,rz2
 /...
ENDAT
```

or: **F(lname)[n]=FEAT/ARC,PTDATA,var_4 var_5**

or: **FA(lname)[n]=FEAT/ARC,PTDATA,var_6,prbdiam var_7**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_3 can be: **,is,js,ks**
or: **does not exist**

var_4 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_5 can be: **,in,jn,kn**
or: **does not exist**

var_6 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_7 can be: **,ip,jp,kp**
or: **does not exist**

ang1 is the start angle and is determined by either the major axis of the current working plane or by the vector defined by the presence of var_3.

ang2 is the included angle of the circular arc with respect to ang1.

ARC signifies that the feature is a circular arc.

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

i,j,k is the direction vector of the plane in which the circular arc lies.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the circular arc is to be measured (that is, a fillet).

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

is,js,ks is the starting direction vector within the plane of the circular arc.

lname is an alphanumeric label name assigned to the feature.

n	is a positive integer that is the index value of a specific individual point.
OUTER	signifies that outside of the circular arc is to be measured (that is, a round).
POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
r,a,h	are the polar coordinates of the center-point of the circular arc.
rad	is a positive real number representing the radius of the circular arc.
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data points.
x,y,z	are the Cartesian coordinates of the center-point of the circular arc.
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

If var_3 is not supplied, then the start angle is determined by the major axis of the current working plane.

The FEAT/ARC statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.72 FEAT/ARC (input format 2)

Function: Defines a nominal circular arc or constructs an actual circular arc, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/ARC,4POINT,var_1,e1x,e1y,e1z,mx,my,mz,e2x,e2y,e2z,$**
cx,cy,cz

or: **FA(lname)=FEAT/ARC,4POINT,var_1,e1x,e1y,e1z,mx,my,mz,e2x,e2y,e2z,$**
cx,cy,cz

Output Formats:

can be: **F(lname)=FEAT/ARC,4POINT,var_1,e1x,e1y,e1z,mx,my,mz,e2x,e2y,e2z,$**
cx,cy,cz

or: **FA(lname)=FEAT/ARC,4POINT,var_1,e1x,e1y,e1z,mx,my,mz,e2x,e2y,e2z,$**
cx,cy,cz

or:
```
FA(lname)=FEAT/ARC,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/ARC,PTDATA,var_2 var_3**

or: **FA(lname)[n]=FEAT/ARC,PTDATA,var_4,prbdiam var_5**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_3 can be: **,in,jn,kn**
or: **does not exist**

var_4 can be: **var_2**
or: **RAWDAT,rx1,ry1,rz1**

var_5 can be: **,ip,jp,kp**
or: **does not exist**

4POINT signifies that the circular arc is being defined with four points.

ARC signifies that the feature is a circular arc.

CART signifies that Cartesian coordinates are to follow.

cx,cy,cz are the Cartesian coordinates for the center of the circular arc.

e1x,e1y,e1z are the Cartesian coordinates for the first end point of the circular arc.

e2x,e2y,e2z are the Cartesian coordinates for the second end point of the circular arc.

ENDAT signifies the end of raw data.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the circular arc is to be measured (that is, a fillet).

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

mx,my,mz are the Cartesian coordinates for the midpoint on the circular arc.

n is a positive integer that is the index value of a specific individual point.

OUTER	signifies that outside of the circular arc is to be measured (that is, a round).
POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data points.
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/ARC statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.73 FEAT/CIRCLE

Function: Defines a nominal circle or constructs an actual circle, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/CIRCLE,var_1,var_2,i,j,k,diam**

or: **FA(lname)=FEAT/CIRCLE,var_1,var_2,i,j,k,diam**

Output Formats:

can be: **F(lname)=FEAT/CIRCLE,var_1,var_2,i,j,k,diam**

or: **FA(lname)=FEAT/CIRCLE,var_1,var_2,i,j,k,diam**

or:
```
FA(lname)=FEAT/CIRCLE,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/CIRCLE,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/CIRCLE,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **var_3**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

CIRCLE signifies that the feature is a circle.

diam is a positive real number representing the diameter of the circle.

ENDAT signifies the end of raw data.

i,j,k is the direction vector of the plane in which the circle lies.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the circle is to be measured (that is, a hole).

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of a specific individual point.

OUTER signifies that outside of the circle is to be measured (that is, a boss).

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified points.

PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
r,a,h	are the polar coordinates of the center-point of the circle.
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data points.
x,y,z	are the Cartesian coordinates of the center-point of the circle.
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/CIRCLE statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.74 FEAT/COMPOUND

Function: Defines a nominal compound feature using previously defined features, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/COMPOUND,var_1,F(lname1),F(lname1) var_3**

Output Formats:

can be: **F(lname)=FEAT/COMPOUND,var_1 var_3**

or: **FA(lname)=FEAT/COMPOUND,FA(lname2) var_1 var_3**

or:
```
FA(lname)=FEAT/COMPOUND,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/COMPOUND,PTDATA,var_4 var_5**

or: **FA(lname)[n]=FEAT/COMPOUND,PTDATA,var_6,prbdiam var_7**

Where:

var_1 can be: **AXIAL,var_2,i,j,k**
or: **PLANE,var_2,i,j,k**
or: **SPHERE,var_2**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_3 can be: **,F(lname1) var_3**
or: **does not exist**

var_4 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_5 can be: **,in,jn,kn**
or: **does not exist**

var_6 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_7 can be: **,ip,jp,kp**
or: **does not exist**

AXIAL signifies that the feature is an axial feature such as a cylinder or cone.

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

PLANE signifies that the feature is a planar feature.

F(lname1) is a feature nominal label used in defining the compound.

FA(lname2) is a measured feature actual label used in the compound.

i,j,k is the direction vector of the feature. For axial, the direction vector is along the axis of the axial feature. For planar, the direction vector is the plane normal pointing away from the part's mass.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

COMPOUND signifies that the feature is a compound feature.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

r,a,h are the polar coordinates of a point locating the feature. For AXIAL, the point lies on the axis of the feature. For PLANE, the point lies on the plane. For SPHERE, the point is the center-point of the sphere.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1 are the Cartesian coordinates of raw data point(s).

rx2,ry2,rz2

...

SPHERE signifies that the feature is a spherical feature.

x,y,z are the Cartesian coordinates of a point locating the feature. For AXIAL, the point lies on the axis of the feature. For PLANE, the point lies on the plane. For SPHERE, the point is the center-point of the sphere.

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/COMPOUND statement is passed to the output file by execution of the OUTPUT statement.

Note 1: This feature definition is used to be assigned with the DAT(x-x) compound datum feature DATDEF.

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

Note 3: For AXIAL compound features, only coaxial axial features such as cones or cylinders can be assigned.

For PLANE compound features, only coplanar features such as planes can be assigned.

For SPHERE compound features, only spheres that share the same center-point can be assigned.

Note 4: Construction is the only method of creating a COMPOUND feature actual.

6.75 FEAT/CONE

Function: Defines a nominal cone or constructs an actual cone, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/CONE,var_1,var_2,i,j,k,ang**

or: **FA(lname)=FEAT/CONE,var_1,var_2,i,j,k,ang**

Output Formats:

can be: **F(lname)=FEAT/CONE,var_1,var_2,i,j,k,ang**

or: **FA(lname)=FEAT/CONE,var_1,var_2,i,j,k,ang**

or:
```
FA(lname)=FEAT/CONE,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/CONE,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/CONE,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **var_3**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

ang is a non-negative real number representing the included angle of the cone in current system units.

CART signifies that Cartesian coordinates are to follow.

CONE signifies that the feature is a cone.

ENDAT signifies the end of raw data.

i,j,k is the direction vector associated with the cone, which points along the cone's axis from the vertex to the open end of the cone. Refer to (Figure B.37 — A cone feature, vector illustration).

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the cone is to be measured (that is, a conical hole).

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of a specific individual point.

OUTER signifies that the outside of the cone is to be measured (that is, a conical boss).

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

r,a,h are the polar coordinates of the vertex.

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data point(s).

x,y,z are the Cartesian coordinates of the vertex.

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/CONE statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.76 FEAT/CONRADSEGMNT

Function: Defines a feature nominal or constructs a feature actual conical radial segment and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/CONRADSEGMNT,var_1,var_2,startrad,endrad, $**
si,sj,sk,ei,ej,ek

or: **FA(lname)=FEAT/CONRADSEGMNT,var_1,var_2,startrad,endrad, $**
si,sj,sk,ei,ej,ek

Output Formats:

can be: **F(lname)= FEAT/CONRADSEGMNT,var_1,var_2,startrad,endrad, $**
si,sj,sk,ei,ej,ek

or: **FA(lname)= FEAT/CONRADSEGMNT,var_1,var_2,startrad,endrad, $**
si,sj,sk,ei,ej,ek

or: **FA(lname)=FEAT/CONRADSEGMNT,RAWDAT**
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT

or: **F(lname)[n]=FEAT/CONRADSEGMNT,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/CONRADSEGMNT,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,sx,sy,sz,ex,ey,ez**
or: **POL,sr,sa,sh,er,ea,eh**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **var_3**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

CONRADSEGMNT signifies that the feature is a conical radial segment.

ei,ej,ek is the longitude end vector used to bound the longitude of the radial segment feature.

ENDAT signifies the end of raw data.

endrad is a non-negative real number representing the radius at the radial segment feature's end point.

er,ea,eh are the polar coordinates of a end point on the axis used to bound the length of the feature.

ex,ey,ez are the Cartesian coordinates of a end point on the axis used to bound the length of the feature.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies internal radial segment feature.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

OUTER signifies external radial segment feature.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data point(s).

si,sj,sk is the longitude start vector used to bound the longitude of the radial segment feature.

sr,sa,sh are the polar coordinates of a start point on the axis used to bound the length of the feature.

startrad is a non-negative real number representing the radius at the radial segment feature's start point.

sx,sy,sz are the Cartesian coordinates of a start point on the axis used to bound the length of the feature.

The location(s) and vectors(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/CONRADSEGMNT statement is passed to the output file by execution of the OUTPUT statement.

Note 1: Feature direction, included angle, conical apex, and length can be calculated from these statement parameters.

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

Note 3: 3D radial segment features are partial cylinder, cone, sphere, or torus features that cannot be considered as a feature-of-size. Therefore it will NOT use a size tolerance, most likely a TOL/RAD or a TOL/PROFS. This 3D radial segment feature is similar to a 2D FEAT/ARC as compared to a 2D FEAT/CIRCLE.

Note 4: The ambiguity in the determination of the included angle is resolved using the right-hand rule of rotation (refer to 5.3.6.2) where the positive axis points in a direction from the start point to the end point and the rotation about this axis vector is from the start vector toward the end vector.

6.77 FEAT/CPARLN

Function: Defines a nominal, or constructs an actual, centered parallel line by center point and length axis, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/CPARLN,var_1,var_2,var_3,i,j,k,i1,j1,k1,len,width**

or: **FA(lname)=FEAT/CPARLN,var_1,var_2,var_3,i,j,k,i1,j1,k1,len,width**

Output Formats:

can be: **F(lname)=FEAT/CPARLN,var_1,var_2,var_3,i,j,k,i1,j1,k1,len,width**

or: **FA(lname)=FEAT/CPARLN,var_1,var_2,var_3,i,j,k,i1,j1,k1,len,width**

or:
```
FA(lname)=FEAT/CPARLN,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/CPARLN,PTDATA,var_4 var_5**

or: **FA(lname)[n]=FEAT/CPARLN,PTDATA,var_6,prbdiam var_7**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **ROUND**
or: **FLAT**
or: **OPEN**

var_3 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_4 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_5 can be: **,in,jn,kn**
or: **does not exist**

var_6 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_7 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

CPARLN signifies that the feature is a centered parallel line. This implies that at least two opposite parallel lines that are of equal length.

ENDAT signifies the end of raw data.

FLAT signifies that the centered parallel line feature is closed and has flat (or rectangular) ends.

i,j,k is the vector normal to and pointing away from the surface in which the centered parallel line feature lies.

This will be the actual normal vector on output if the RMEAS/ statement has the VECBLD option.

i1,j1,k1 is the centered parallel line length orientation vector, which is perpendicular to the normal vector. This will be the actual length vector on output if the RMEAS/ statement has the ORIENT option.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the centered parallel line feature is to be measured.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

len is a positive real number representing the length of the centered parallel line along the specified length axis vector. The length is measured in the plane defined by the center point (x,y,z) and normal vector (i,j,k).

n is a positive integer that is the index value of the individual point.

OPEN signifies that the centered parallel line feature has no ends within the specified length.

OUTER signifies that the outside of the centered parallel line feature is to be measured.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

r,a,h are the polar coordinates of the center of the centered parallel line feature.

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

ROUND signifies that the centered parallel line feature is closed and has round ends.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data point(s).

width is a positive real number representing the width of the centered parallel line along the mutually orthogonal axis to the surface normal and the length axis vector. The width is measured in the plane defined by the center point (x,y,z) and normal vector (i,j,k).

x,y,z are the Cartesian coordinates of the center point of the centered parallel line feature.

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/CPARLN statement is passed to the output file by execution of the OUTPUT statement.

Note 1: Refer to (Figure B.38 — A centered parallel line feature) for a pictorial representation of a centered parallel line feature.

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.78 FEAT/CYLNDR

Function: Defines a nominal cylinder or constructs an actual cylinder, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/CYLNDR,var_1,var_2,i,j,k,diam var_3**

or: **FA(lname)=FEAT/CYLNDR,var_1,var_2,i,j,k,diam var_3**

Output Formats:

can be: **F(lname)=FEAT/CYLNDR,var_1,var_2,i,j,k,diam var_3**

or: **FA(lname)=FEAT/CYLNDR,var_1,var_2,i,j,k,diam var_3**

or:
```
FA(lname)=FEAT/CYLNDR,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/CYLNDR,PTDATA,var_4 var_5**

or: **FA(lname)[n]=FEAT/CYLNDR,PTDATA,var_6,prbdiam var_7**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_3 can be: **,len**
or: **does not exist**

var_4 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_5 can be: **,in,jn,kn**
or: **does not exist**

var_6 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_7 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

CYLNDR signifies that the feature is a cylinder.

diam is a positive real number representing the diameter of the cylinder.

ENDAT signifies the end of raw data.

i,j,k is the direction vector associated with the cylinder and points along the cylinder's axis.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the cylinder is to be measured (that is, a cylindrical hole).

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

len is a non-negative real number representing the length of the cylinder from x,y,z (which is designated as the base-point when len is specified) along the cylinder's direction vector. Specifying a length automatically bounds the cylinder.

n is a positive integer that is the index value of the individual point.

OUTER signifies that the outside of the cylinder is to be measured (that is, a cylindrical boss/stud).

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

r,a,h are the polar coordinates of any point on the unbounded cylinder's axis; the point is the center-point for bounded cylinders when a length has not been specified; and the point is the base-point for bounded cylinders when a length has been specified.

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data point(s).

x,y,z are the Cartesian coordinates of any point on the unbounded cylinder's axis; the point is the center-point for bounded cylinders when a length has not been specified; and the point is the base-point for bounded cylinders when a length has been specified.

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/CYLNDR statement is passed to the output file by execution of the OUTPUT statement.

Note 1: Refer to (Figure B.39 — A cylinder feature with the 'len' parameter specified) for a pictorial representation of a bounded cylinder feature.

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.79 FEAT/CYLRADSEGMNT

Function: Defines a feature nominal or constructs a feature actual cylindrical radial segment and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/CYLRADSEGMNT,var_1,var_2,rad, $**
si,sj,sk,ei,ej,ek

or: **FA(lname)=FEAT/CYLRADSEGMNT,var_1, var_2,rad, $**
si,sj,sk,ei,ej,ek

Output Formats:

can be: **F(lname)= FEAT/CYLRADSEGMNT,var_1, var_2,rad, $**
si,sj,sk,ei,ej,ek

or: **FA(lname)= FEAT/CYLRADSEGMNT,var_1, var_2,rad, $**
si,sj,sk,ei,ej,ek

or: **FA(lname)=FEAT/CYLRADSEGMNT,RAWDAT**
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT

or: **F(lname)[n]=FEAT/CYLRADSEGMNT,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/CYLRADSEGMNT,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,sx,sy,sz,ex,ey,ez**
or: **POL,sr,sa,sh,er,ea,eh**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **var_3**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

CYLRADSEGMNT signifies that the feature is a cylindrical radial segment.

ei,ej,ek is the longitude end vector used to bound the longitude of the feature.

ENDAT signifies the end of raw data.

er,ea,eh are the polar coordinates of a end point on the axis used to bound the length of the feature.

ex,ey,ez are the Cartesian coordinates of a end point on the axis used to bound the length of the feature.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies internal radial segment feature.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n	is a positive integer that is the index value of the individual point.
OUTER	signifies external radial segment feature.
POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
rad	is a positive real number representing the radius of the radial segment.
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data point(s).
si,sj,sk	is the longitude start vector used to bound the longitude of the feature.
sr,sa,sh	are the polar coordinates of a start point on the axis used to bound the length of the feature.
sx,sy,sz	are the Cartesian coordinates of a start point on the axis used to bound the length of the feature.

The location(s) and vectors(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/CYLRADSEGMNT statement is passed to the output file by execution of the OUTPUT statement.

Note 1: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

Note 2: 3D radial segment features are partial cylinder, cone, sphere, or torus features that cannot be considered as a feature-of-size. Therefore it will NOT use a size tolerance, most likely a TOL/RAD or a TOL/PROFS. This 3D radial segment feature is similar to a 2D FEAT/ARC as compared to a 2D FEAT/CIRCLE.

Note 3: The ambiguity in the determination of the included angle is resolved using the right-hand rule of rotation (refer to 5.3.6.2) where the positive axis points in a direction from the start point to the end point and the rotation about this axis vector is from the start vector toward the end vector.

6.80 FEAT/EDGEPT

Function: Defines a nominal point or constructs an actual point on the edge of a surface, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/EDGEPT,var_1,i,j,k,i1,j1,k1**

or: **FA(lname)=FEAT/EDGEPT,var_1,i,j,k,i1,j1,k1**

Output Formats:

can be: **F(lname)=FEAT/EDGEPT,var_1,i,j,k,i1,j1,k1**

or: **FA(lname)=FEAT/EDGEPT,var_1,i,j,k,i1,j1,k1**

or:
```
FA(lname)=FEAT/EDGEPT,RAWDAT
/rx1,ry1,rz1
ENDAT
```

or: **F(lname)[n]=FEAT/EDGEPT,PTDATA,var_2 var_3**

or: **FA(lname)[n]=FEAT/EDGEPT,PTDATA,var_4,prbdiam var_5**

Where:

var_1 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_2 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_3 can be: **,in,jn,kn**
or: **does not exist**

var_4 can be: **var_2**
or: **RAWDAT,rx1,ry1,rz1**

var_5 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

i,j,k is the vector normal to and pointing away from the edge in which the edge point lies, that can be used for probe compensation.

i1,j1,k1 is the vector normal to and pointing away from the surface adjacent to the edge in which the edge point lies.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

r,a,h are the polar coordinates of the edge point itself.

RAWDAT signifies that the uncompensated Cartesian coordinates of the measured point (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1 are the Cartesian coordinates of the raw data point.

x,y,z are the Cartesian coordinates of the edge point itself.

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output each data point is preceded with a slash and followed by a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/EDGEPT statement is passed to the output file by execution of the OUTPUT statement.

Note 1: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

Note 2: Refer to (Figure B.41 — An edgepoint feature) for an illustration of an edge point feature.

6.81 FEAT/ELLIPS

Function: Defines a nominal ellipse or constructs an actual ellipse, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/ELLIPS,var_1,var_2,var_3,i,j,k,diam**

or: **FA(lname)=FEAT/ELLIPS,var_1,var_2,var_3,i,j,k,diam**

Output Formats:

can be: **F(lname)=FEAT/ELLIPS,var_1,var_2,var_3,i,j,k,diam**

or: **FA(lname)=FEAT/ELLIPS,var_1,var_2,var_3,i,j,k,diam**

or:
```
FA(lname)=FEAT/ELLIPS,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/ELLIPS,PTDATA,var_4 var_5**

or: **FA(lname)[n]=FEAT/ELLIPS,PTDATA,var_6,prbdiam var_7**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,f1x,f1y,f1z,f2x,f2y,f2z**
or: **POL,f1r,f1a,f1h,f2r,f2a,f2h**

var_3 can be: **MAJOR**
or: **MINOR**

var_4 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_5 can be: **,in,jn,kn**
or: **does not exist**

var_6 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_7 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

diam is a positive real number representing the MAJOR or MINOR diameter, as specified by var_3 of the FEAT/ELLIPS statement.

ELLIPS signifies that the feature is an ellipse.

ENDAT signifies the end of raw data.

f1r,f1a,f1h
f2r,f2a,f2h are the polar coordinates of the two foci.

f1x,f1y,f1z
f2x,f2y,f2z are the Cartesian coordinates of the two foci.

i,j,k is the direction vector of the plane in which the ellipse lies.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the ellipse is to be measured (that is, an elliptical hole).

ip,jp,kp	is the probe compensation direction of the specified point. Refer to section 5.3.2.6.
lname	is an alphanumeric label name assigned to the feature.
MAJOR	signifies that the major diameter is to be defined.
MINOR	signifies that the minor diameter is to be defined.
n	is a positive integer that is the index value of the individual point.
OUTER	signifies that outside of the ellipse is to be measured (that is, an elliptical boss).
POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data point(s).
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

The TOL/DIAM is associated with the MAJOR or MINOR diameter defined here. All location requirements are calculated using the ellipse center (that is, the midpoint of the two foci).

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/ELLIPS statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.82 FEAT/ELONGCYL

Function: Defines a feature nominal or constructs a feature actual elongated cylinder (elongated hole) and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/ELONGCYL,var_1,var_2,ia,ja,ka,size,radius var_3**

or: **FA(lname)=FEAT/ELONGCYL,var_1,var_2,ia,ja,ka,size,radius var_3**

Output Formats:

can be: **F(lname)=FEAT/ELONGCYL,var_1,var_2,ia,ja,ka,size,radius var_3**

or: **FA(lname)=FEAT/ELONGCYL,var_1,var_2,ia,ja,ka,size,radius var_3**

or: **FA(lname)=FEAT/ELONGCYL,RAWDAT**

/rx1,ry1,rz1

/rx2,ry2,rz2

/...

ENDAT

or: **F(lname)[n]=FEAT/ELONGCYL,PTDATA,var_4 var_5**

or: **FA(lname)[n]=FEAT/ELONGCYL,PTDATA,var_6,prbdiam var_7**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z,i,j,k**
or: **POL,r,a,h,i,j,k**

var_3 can be: **,len**
or: **does not exist**

var_4 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_5 can be: **,in,jn,kn**
or: **does not exist**

var_6 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_7 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

ELONGCYL signifies that the feature is an elongated cylinder.

ENDAT signifies the end of raw data.

i,j,k is the direction vector of the center-plane.

ia,ja,ka is the direction vector associated with the feature and points along the feature's axial length.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the cylinder is to be measured (that is, a cylindrical hole).

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

len	is a non-negative real number representing the axial length of the feature from x,y,z (which is designated as the base-point when length is specified) along the feature's direction vector. Specifying a length automatically bounds the feature.
n	is a positive integer that is the index value of the individual point.
OUTER	signifies external elongated cylinder (that is elongated cylindrical boss).
POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
r,a,h	are the polar coordinates of a point on the center-plane.
radius	is a positive real number representing radius of the radial ends of the elongated cylinder.
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data point(s).
size	is a positive real number representing the distance between the extents of the elongated cylinder.
x,y,z	are the Cartesian coordinates of a point on the center-plane.
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/ELONGCYL statement is passed to the output file by execution of the OUTPUT statement.

Note 1: Used if a feature of size is located with its center-plane; Refer to (Figure B.40 — An elongated cylinder feature).

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.83 FEAT/GCURVE

Function: Defines a generic curve, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/GCURVE,var_1**

Output Formats:

can be: **F(lname)=FEAT/GCURVE,var_1**

or: **FA(lname)=FEAT/GCURVE,var_2**

or:
```
FA(lname)=FEAT/GCURVE,CART,i,j,k
/x,y,z
/x,y,z
/...
ENDAT
```

or:
```
FA(lname)=FEAT/GCURVE,POL,i,j,k
/r,a,h
/r,a,h
/...
ENDAT
```

or:
```
FA(lname)=FEAT/GCURVE,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/GCURVE,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/GCURVE,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **CART,x,y,z,i,j,k**
or: **POL,r,a,h,i,j,k**
CART,x,y,z,i,j,k,PTDATA,xd,yd,zd,id,jd,kd var_7
POL,r,a,h,i,j,k,PTDATA,rd,ad,hd,id,jd,kd var_8

var_2 can be: **CART,x,y,z,i,j,k,PTDATA,xd,yd,zd,id,jd,kd var_7**
or: **POL,r,a,h,i,j,k,PTDATA,rd,ad,hd,id,jd,kd var_8**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **var_3**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

var_7 can be: **,xd,yd,zd,id,jd,kd var_7**
or: **,xd,yd,zd,id,jd,kd**

var_8 can be: **,rd,ad,hd,id,jd,kd var_8**
or: **,rd,ad,hd,id,jd,kd**

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

GCURVE signifies that the feature is a generic curve.

i,j,k is the direction vector of the plane in which the generic curve lies.

id,jd,kd is a vector normal to and pointing away from the surface in which the point lies.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

r,a,h are the polar coordinates of a point on the plane in which the curve lies.

rd,ad,hd are the polar coordinates of a point on the generic curve.

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured points (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data points.

x,y,z are the Cartesian coordinates of a point on the plane in which the curve lies.

xd,yd,zd are the Cartesian coordinates of a point on the generic curve.

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

GCURVE is used to assign a curve to a label and can be used when digitizing a part. The feature nominal definition specifies the plane in which the feature should lie. Output data from a measured GCURVE can represent points on the curve if probe compensation is enabled or can represent raw data, x,y,z, or r,a,h, data for center of probe, when probe compensation is disabled. Refer to clause 5.3.2.1.2 for further information.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/GCURVE statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.84 FEAT/GEOM

Function: Defines a nominal geometry feature from previously defined geometry data, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/GEOM,G(lname1),var_1**

Output Formats:

can be: **F(lname)=FEAT/GEOM,G(lname1),var_2,i,j,k**

or: **FA(lname)=FEAT/GEOM,G(lname1),var_2,i,j,k**

Where:

var_1 can be: **CART**
or: **POL**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

CART signifies that Cartesian coordinates are to be used.

G(lname1) is the label of previously defined geometry data.

GEOM signifies that the feature is geometry data.

i,j,k is the direction vector of the geometry.

lname is an alphanumeric label name assigned to the feature.

POL signifies that polar coordinates are to be used.

r,a,h are the polar coordinates of a point on the geometry.

x,y,z are the Cartesian coordinates of a point on the geometry.

The FEAT/GEOM statement is passed to the output file by execution of the OUTPUT statement.

The nominal definition contains all coordinates and vectors on the specified geometry. The coordinates and vector for the feature nominal output format are those corresponding to the most recent feature actual. If the feature has not been measured or constructed then the coordinates and vector will be arbitrary on the geometry.

6.85 FEAT/GSURF

Function: Defines a nominal generic surface, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/GSURF var_1**

Output Formats:

can be: **F(lname)=FEAT/GSURF var_1**

or: **FA(lname)=FEAT/GSURF,var_2**

or:
```
FA(lname)=FEAT/GSURF,CART
/x,y,z var_3
/x,y,z var_3
/...
ENDAT
```

or:
```
FA(lname)=FEAT/GSURF,POL
/r,a,h var_3
/r,a,h var_3
/...
ENDAT
```

or:
```
FA(lname)=FEAT/GSURF
/x,y,z var_3
/x,y,z var_3
/...
ENDAT
```

or:
```
FA(lname)=FEAT/GSURF,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/GSURF,PTDATA,var_4 var_5**

or: **FA(lname)[n]=FEAT/GSURF,PTDATA,var_6,prbdiam var_7**

Where:

var_1 can be: **,CART,PTDATA,x,y,z,i,j,k,x,y,z,i,j,k var_8**
or: **,POL,PTDATA,r,a,h,i,j,k,r,a,h,i,j,k var_9**
or: **,CART**
or: **,POL**
or: **does not exist**

var_2 can be: **CART,PTDATA,x,y,z,i,j,k,x,y,z,i,j,k var_8**
or: **POL,PTDATA,r,a,h,i,j,k,r,a,h,i,j,k var_9**

var_3 can be: **,i,j,k**
or: **does not exist**

var_4 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_5 can be: **,in,jn,kn**
or: **does not exist**

var_6 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_7 can be: **,ip,jp,kp**
or: **does not exist**

var_8 can be: **,x,y,z,i,j,k var_8**
or: **,x,y,z,i,j,k**

var_9 can be: **,r,a,h,i,j,k var_9**
or: **,r,a,h,i,j,k**

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

GSURF signifies that the feature is a generic surface.

i,j,k is the direction vector representing the normal direction of the generic surface at the specified point and pointing away from the part's mass

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

r,a,h are the polar coordinates of a point on the generic surface.

rp,ap,hp are the polar coordinates of a point on the generic surface.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data point(s).

x,y,z are the Cartesian coordinates of a point on the generic surface.

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

When probe compensation is on, the surface i,j,k vectors of each PTMEAS are required. Refer to clause 5.3.2.1.2 for further information.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/GSURF statement is passed to the output file by execution of the OUTPUT statement.

Note 1: GSURF is used to assign a label name to a surface and can be used when digitizing a part.

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.86 FEAT/LINE

Function: Defines a nominal line or constructs an actual line, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/LINE,var_1,ni,nj,nk**

or: **FA(lname)=FEAT/LINE,var_1,ni,nj,nk**

Output Formats:

can be: **F(lname)=FEAT/LINE,var_1,ni,nj,nk**

or: **FA(lname)=FEAT/LINE,var_1,ni,nj,nk**

or:
```
FA(lname)=FEAT/LINE,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/LINE,PTDATA,var_4 var_5**

or: **FA(lname)[n]=FEAT/LINE,PTDATA,var_6,prbdiam var_7**

Where:

var_1 can be: **UNBND,var_2**
or: **BND,var_3**

var_2 can be: **CART,x,y,z,i,j,k**
or: **POL,r,a,h,i,j,k**

var_3 can be: **CART,e1x,e1y,e1z,e2x,e2y,e2z**
or: **POL,e1r,e1a,e1h,e2r,e2a,e2h**

var_4 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_5 can be: **,in,jn,kn**
or: **does not exist**

var_6 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_7 can be: **,ip,jp,kp**
or: **does not exist**

BND signifies that a bounded line is to be defined.

CART signifies that Cartesian coordinates are to follow.

e1r,e1a,e1h are the polar coordinates of the starting point of the line.

e2r,e2a,e2h are the polar coordinates of the end point of the line.

e1x,e1y,e1z are the Cartesian coordinates of the starting point of the line.

e2x,e2y,e2z are the Cartesian coordinates of the end point of the line.

ENDAT signifies the end of raw data.

i,j,k is the direction vector of the line.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

LINE signifies that the feature is a line.

n	is a positive integer that is the index value of the individual point.
ni,nj,nk	is the normal vector of the plane in which the line lies, that can be used for probe compensation.
POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
r,a,h	are the polar coordinates of a point on the line.
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data point(s).
UNBND	signifies that an unbounded line is to be defined.
x,y,z	are the Cartesian coordinates of a point on the line.
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/LINE statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.87 FEAT/OBJECT

Function: Defines a "user created" object that has definable characteristics in another file that the DME has the capability to measure, and assigns a label to it. Identifies either a nominal object or constructs an actual object.

Input Formats:

can be: **F(lname)=FEAT/OBJECT,parm var_1**

or: **FA(lname)=FEAT/OBJECT,parm var_1**

Output Formats:

can be: **F(lname)=FEAT/OBJECT,parm var_1**

or: **FA(lname)=FEAT/OBJECT,parm var_1**

or:
```
FA(lname)=FEAT/OBJECT,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/OBJECT,PTDATA,var_2 var_3**

or: **FA(lname)[n]=FEAT/OBJECT,PTDATA,var_4,prbdiam var_5**

Where:

var_1 can be: **,parm var_1**
or: **does not exist**

var_2 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_3 can be: **,in,jn,kn**
or: **does not exist**

var_4 can be: **var_2**
or: **RAWDAT,rx1,ry1,rz1**

var_5 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

OBJECT signifies that an object is to be defined.

parm in the input format this is a value or variable of any DECL type, the meaning of which is determined by the implementation. For the output format, it is the evaluated value of the parameter.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data point(s).

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/OBJECT statement is passed to the output file by execution of the OUTPUT statement.

Note 1: The intent of this statement is to provide a means by which previously defined and coded feature requirements can be accessed for evaluation by a DME. The DME can in turn fully utilize its capabilities and functionality to perform the required evaluations. This statement is used in conjunction with TOL/USETOL, defined in section 6.211.

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.88 FEAT/PARPLN

Function: Defines a feature nominal or constructs a feature actual with opposite parallel planes (slot, block), and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/PARPLN,var_1,var_2,width**

or: **FA(lname)=FEAT/PARPLN,var_1,var_2,width**

Output Formats:

can be: **F(lname)=FEAT/PARPLN,var_1,var_2,width**

or: **FA(lname)=FEAT/PARPLN,var_1,var_2,width**

or:
```
FA(lname)=FEAT/PARPLN,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/PARPLN,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/PARPLN,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z,p1x,p1y,p1z,i1,j1,k1,p2x,p2y,p2z,i2,j2,k2**
or: **POL,r,a,h,p1r,p1a,p1h,i1,j1,k1,p2r,p2a,p2h,i2,j2,k2**
or: **MIDPL,CART,x,y,z,i,j,k**
or: **MIDPL,POL,r,a,h,i,j,k**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **CART,x,y,z,p1x,p1y,p1z,i1,j1,k1,p2x,p2y,p2z,i2,j2,k2**
or: **POL,r,a,h,p1r,p1a,p1h,i1,j1,k1,p2r,p2a,p2h,i2,j2,k2**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

i,j,k is the direction vector of the center-plane.

i1,j1,k1
i2,j2,k2 are the direction vectors of the respective parallel planes, pointing away from the part's mass.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies internal parallel planes (that is, a slot).

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

MIDPL signifies that the center plane definition for this feature is to follow.

n is a positive integer that is the index value of the individual point.

OUTER	signifies external parallel planes (that is, a block).
p1r,p1a,p1h **p2r,p2a,p2h**	are the polar coordinates of the two points residing on opposite parallel planes.
p1x,p1y,p1z **p2x,p2y,p2z**	are the Cartesian coordinates of the two points residing on opposite parallel planes.
POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
r,a,h	are the polar coordinates of a point on the center-plane.
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data point(s).
width	is a positive real number representing the distance between the opposite parallel planes.
x,y,z	are the Cartesian coordinates of a point on the center-plane.
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/PARPLN statement is passed to the output file by execution of the OUTPUT statement.

Note 1: Used if a feature of size (width) is located with its center-plane; otherwise, it can be functionally equivalent to two planes with a distance between them. Refer to (Figure B.42 — A parallel plane feature).

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.89 FEAT/PATERN

Function: Defines a nominal pattern using previously defined features, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/PATERN,F(lname1) var_1 var_2**

Output Formats:

can be: **F(lname)=FEAT/PATERN,F(lname1) var_1 var_2**

or: **FA(lname)=FEAT/PATERN,FA(lname2) var_3 var_4**

or:
```
FA(lname)=FEAT/PATERN,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/PATERN,PTDATA,var_5 var_6**

or: **FA(lname)[n]=FEAT/PATERN,PTDATA,var_7,prbdiam var_8**

Where:

var_1 can be: **,F(lnamen)**

var_2 can be: **var_1 var_2**
or: **does not exist**

var_3 can be: **,FA(lnamen)**

var_4 can be: **var_3 var_4**
or: **does not exist**

var_5 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_6 can be: **,in,jn,kn**
or: **does not exist**

var_7 can be: **var_5**
or: **RAWDAT,rx1,ry1,rz1**

var_8 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

F(lname1) is the first feature nominal label in a list to be used in the pattern.

F(lnamen) is the nth feature nominal label in a list to be used in a pattern.

FA(lname2) is the measured first feature actual label in a list used in the pattern.

FA(lnamen) is the nth measured feature actual label in a list used in a pattern.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

PATERN signifies that the feature is a pattern.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data point(s).

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/PATERN statement is passed to the output file by execution of the OUTPUT statement.

Note 1: This feature definition is used in conjunction with the composite positional tolerance for patterns, for example, TOL/COMPOS. It can also be used in video systems when required; for example, to count intersections of an edge line with a pattern of features, or to measure several intersections of an edge line with a pattern of features. This capability is similar to a CMM's ca pability in automatic mode to measure a circle with n points.

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.90 FEAT/PLANE

Function: Defines a nominal plane or constructs an actual plane, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/PLANE,var_1,i,j,k**

or: **FA(lname)=FEAT/PLANE,var_1,i,j,k**

Output Formats:

can be: **F(lname)=FEAT/PLANE,var_1,i,j,k**

or: **FA(lname)=FEAT/PLANE,var_1,i,j,k**

or:
```
FA(lname)=FEAT/PLANE,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/PLANE,PTDATA,var_2 var_3**

or: **FA(lname)[n]=FEAT/PLANE,PTDATA,var_4,prbdiam var_5**

Where:

var_1 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_2 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_3 can be: **,in,jn,kn**
or: **does not exist**

var_4 can be: **var_2**
or: **RAWDAT,rx1,ry1,rz1**

var_5 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

i,j,k is the direction vector of the plane. For a FEAT/PLANE that represents a surface of mass, the plane's normal i,j,k points away from the mass. For all other cases, the plane's normal i,j,k may point in either direction depending on its intended use.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

PLANE signifies that the feature is a plane.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

r,a,h are the polar coordinates of a point on the plane.

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data point(s).

x,y,z are the Cartesian coordinates of a point on the plane.

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/PLANE statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.91 FEAT/POINT

Function: Defines a nominal point or constructs an actual point, and assigns to it a feature label.

Input Formats:

can be: **F(lname)=FEAT/POINT,var_1,i,j,k**

or: **FA(lname)=FEAT/POINT,var_1,i,j,k**

Output Formats:

can be: **F(lname)=FEAT/POINT,var_1,i,j,k**

or: **FA(lname)=FEAT/POINT,var_1,i,j,k**

or:
```
FA(lname)=FEAT/POINT,RAWDAT
/rx1,ry1,rz1
ENDAT
```

or: **F(lname)[n]=FEAT/POINT,PTDATA,var_2 var_3**

or: **FA(lname)[n]=FEAT/POINT,PTDATA,var_4,prbdiam var_5**

Where:

var_1 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_2 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_3 can be: **,in,jn,kn**
or: **does not exist**

var_4 can be: **var_2**
or: **RAWDAT,rx1,ry1,rz1**

var_5 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

i,j,k is a vector, normal to and pointing away from, the surface in which the point lies, that can be used for probe compensation.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

POINT signifies that the feature is a point.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

r,a,h are the polar coordinates of the point itself.

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1 are the Cartesian coordinates of raw data point.

x,y,z are the Cartesian coordinates of the point itself.

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/POINT statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.92 FEAT/RCTNGL

Function: Defines a nominal right rectangular prism or constructs an actual right rectangular prism, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/RCTNGL,var_1,var_2,i1,j1,k1,width1,i2,j2,k2,width2 $**
,i3,j3,k3,width3

or: **FA(lname)=FEAT/RCTNGL,var_1,var_2,i1,j1,k1,width1,i2,j2,k2,width2 $**
,i3,j3,k3,width3

Output Formats:

can be: **F(lname)=FEAT/RCTNGL,var_1,var_2,i1,j1,k1,width1,i2,j2,k2,width2 $**
,i3,j3,k3,width3

or: **FA(lname)=FEAT/RCTNGL,var_1,var_2,i1,j1,k1,width1,i2,j2,k2,width2 $**
,i3,j3,k3,width3

or:
```
FA(lname)=FEAT/RCTNGL,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/RCTNGL,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/RCTNGL,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **var_3**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

i1,j1,k1
i2,j2,k2
i3,j3,k3 are the direction vectors associated with the faces of the right rectangular prism.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the right rectangular prism is to be measured.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

OUTER signifies that the outside of the right rectangular prism is to be measured.

POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
r,a,h	are the polar coordinates of the center-point of the right rectangular prism.
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point (raw data) are to follow.
RCTNGL	signifies that the feature is a right rectangular prism.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data point(s).
width1 **width2** **width3**	are positive real numbers representing the widths of the right rectangular prism, each disposed equally about the center point, along the corresponding preceding i,j,k vector.
x,y,z	are the Cartesian coordinates of the center-point of the right rectangular prism.
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data point is output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/RCTNGL statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.93 FEAT/REVSURF

Function: Defines a nominal surface-of-revolution or constructs an actual surface-of-revolution, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/REVSURF,var_1,var_2,i,j,k,var_3**

or: **FA(lname)=FEAT/REVSURF,var_1,var_2,i,j,k,var_3**

Output Formats:

can be: **F(lname)=FEAT/REVSURF,var_1,var_2,i,j,k,var_3**

or: **FA(lname)=FEAT/REVSURF,var_1,var_2,i,j,k,var_3**

or:
```
FA(lname)=FEAT/REVSURF,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/REVSURF,PTDATA,var_6 var_7**

or: **FA(lname)[n]=FEAT/REVSURF,PTDATA,var_8,prbdiam var_9**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_3 can be: **CART,x1,y1,z1,x2,y2,z2 var_4**
or: **POL,r1,a1,h1,r2,a2,h2 var_5**

var_4 can be: **,xn,yn,zn var_4**
or: **does not exist**

var_5 can be: **,rn,an,hn var_5**
or: **does not exist**

var_6 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_7 can be: **,in,jn,kn**
or: **does not exist**

var_8 can be: **var_6**
or: **RAWDAT,rx1,ry1,rz1**

var_9 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

REVSURF signifies that the feature is a surface-of-revolution.

ENDAT signifies the end of raw data.

i,j,k is the direction vector associated with the surface-of-revolution and points along the surface-of-revolution's axis.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section (5.3.2.6).

INNER signifies that the inside of the surface-of-revolution is to be measured (that is, a hole).

ip,jp,kp is the probe compensation direction of the specified point. Refer to section (5.3.2.6).

lname	is an alphanumeric label name assigned to the feature.
n	is a positive integer that is the index value of the individual point.
OUTER	signifies that the outside of the surface-of-revolution is to be measured (that is, a boss).
POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
r,a,h	are the polar coordinates of any point on the surface-of-revolution 's axis.
r1,a1,h2 **r2,a2,h2** **rn,an,hn**	are the polar coordinates of a point belonging to a sequence list or points describing a 2D curve for the surface-of-revolution.
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data point(s).
x,y,z	are the Cartesian coordinates of any point on the surface-of-revolution's axis.
x1,y1,z1 **x2,y2,z2** **xn,yn,zn**	are the Cartesian coordinates of a point belonging to a sequence list or points describing a 2D curve for the surface-of-revolution.
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/REVSURF statement is passed to the output file by execution of the OUTPUT statement.

Note 1: Refer to (Figure B.94 — A surface-of-revolution feature) for a pictorial representation of a surface-of-revolution feature.

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.94 FEAT/SPHERE

Function: Defines a nominal sphere or constructs an actual sphere, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/SPHERE,var_1,var_2,diam var_3**

or: **FA(lname)=FEAT/SPHERE,var_1,var_2,diam**

Output Formats:

can be: **F(lname)=FEAT/SPHERE,var_1,var_2,diam var_3**

or: **FA(lname)=FEAT/SPHERE,var_1,var_2,diam**

or:
```
FA(lname)=FEAT/SPHERE,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/SPHERE,PTDATA,var_4 var_5**

or: **FA(lname)[n]=FEAT/SPHERE,PTDATA,var_6,prbdiam var_7**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_3 can be: **,i,j,k var_8**
or: **does not exist**

var_4 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_5 can be: **,in,jn,kn**
or: **does not exist**

var_6 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_7 can be: **,ip,jp,kp**
or: **does not exist**

var_8 can be: **,angle**
or: **does not exist**

angle is the latitude indicating the measurable portion of the sphere. A positive angle is towards the i,j,k. Refer to (Figure B.43 — A sphere feature).

CART signifies that Cartesian coordinates are to follow.

diam is a positive real number representing the diameter of the sphere.

ENDAT signifies the end of raw data.

i,j,k is the orientation of the plane where the semi-sphere lies or the orientation of the sphere stem. Refer to (Figure B.43 — A sphere feature).

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the sphere is to be measured.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n	is a positive integer that is the index value of the individual point.
OUTER	signifies that outside of the sphere is to be measured.
POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
r,a,h	are the polar coordinates of the center-point of the sphere.
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data point(s).
SPHERE	signifies that the feature is a sphere.
x,y,z	are the Cartesian coordinates of the center-point of the sphere.
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/SPHERE statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.95 FEAT/SPHRADSEGMNT

Function: Defines a feature nominal or constructs a feature actual spherical radial segment and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/SPHRADSEGMNT,var_1,var_2,rad,npi,npj,npk, $**
latstartang,latstopang,pmi,pmj,pmk,longstartang,longstopang

or: **FA(lname)=FEAT/SPHRADSEGMNT,var_1,var_2,rad,npi,npj,npk, $**
latstartang,latstopang,pmi,pmj,pmk,longstartang,longstopang

Output Formats:

can be: **F(lname)= FEAT/SPHRADSEGMNT,var_1,var_2,rad,npi,npj,npk, $**
latstartang,latstopang,pmi,pmj,pmk,longstartang,longstopang

or: **FA(lname)= FEAT/SPHRADSEGMNT,var_1,var_2,rad,npi,npj,npk, $**
latstartang,latstopang,pmi,pmj,pmk,longstartang,longstopang

or:
```
FA(lname)=FEAT/SPHRADSEGMNT,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/SPHRADSEGMNT,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/SPHRADSEGMNT,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,cx,cy,cz**
or: **POL,cr,ca,ch**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

cr,ca,ch are the polar coordinates of the center of the feature.

cx,cy,cz are the Cartesian coordinates of the center of the feature.

ENDAT signifies the end of raw data.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies internal radial segment feature.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

latstartang is a real number angle, specifying the lower limit of the latitude of the spherical radial segment.

latstopang is a real number angle, specifying the upper limit of the latitude of the spherical radial segment.

longstartang is a real number angle, specifying the lower limit of the longitude of the spherical radial segment.

longstopang is a real number angle, specifying the upper limit of the longitude of the spherical radial segment.

n is a positive integer that is the index value of the individual point.

npi,npj,npk is the north pole vector of the feature whose direction from the center of the sphere specifies the positive 90° direction for latitude.

pmi,pmj,pmk is the prime meridian vector of the feature.

OUTER signifies external radial segment feature.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

rad is a positive real number representing the radius of the sphere.

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1 are the Cartesian coordinates of raw data point(s).

rx2,ry2,rz2

...

SPHRADSEGMNT signifies that the feature is a spherical radial segment.

The location(s) and vectors(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/SPHRADSEGMNT statement is passed to the output file by execution of the OUTPUT statement.

Note 1: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

Note 2: 3D radial segment features are partial cylinder, cone, sphere, or torus features that cannot be considered as a feature-of-size. Therefore it will NOT use a size tolerance, most likely a TOL/RAD or a TOL/PROFS. This 3D radial segment feature is similar to a 2D FEAT/ARC as compared to a 2D FEAT/CIRCLE.

Note 3: The ambiguity in the determination of the included angle is resolved using the right-hand rule of rotation (refer to 5.3.6.2) where the positive axis points in a direction of the north pole vector and the rotation about this axis vector is from the start angle toward the end angle relative to the prime meridian vector.

6.96 FEAT/SYMPLN

Function: Defines a feature nominal or constructs a feature actual with opposite symmetric planes (slot with draft, block with draft) and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/SYMPLN,var_1,var_2,width**

or: **FA(lname)=FEAT/SYMPLN,var_1,var_2,width**

Output Formats:

can be: **F(lname)=FEAT/SYMPLN,var_1,var_2,width**

or: **FA(lname)=FEAT/SYMPLN,var_1,var_2,width**

or:
```
FA(lname)=FEAT/SYMPLN,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/SYMPLN,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/SYMPLN,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z,p1x,p1y,p1z,i1,j1,k1,p2x,p2y,p2z,i2,j2,k2**
or: **POL,r,a,h,p1r,p1a,p1h,i1,j1,k1,p2r,p2a,p2h,i2,j2,k2**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **var_3**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

ENDAT signifies the end of raw data.

i,j,k is the direction vector of the center-plane.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies internal symmetric planes (that is, a slot).

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

OUTER signifies external symmetric planes (that is, a block).

p1r,p1a,p1h
p2r,p2a,p2h are the polar coordinates of the two points residing on opposite symmetric planes.

p1x,p1y,p1z
p2x,p2y,p2z are the Cartesian coordinates of the two points residing on opposite symmetric planes.

POL	signifies that polar coordinates are to follow.
prbdiam	is a positive real number representing the actual probe diameter used to measure the specified point.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
r,a,h	are the polar coordinates of a point on the center-plane.
RAWDAT	signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.
rp,ap,hp	are the polar coordinates of the specified point.
rx1,ry1,rz1 **rx2,ry2,rz2** **...**	are the Cartesian coordinates of raw data point(s)
width	is a positive real number representing the distance between the opposite symmetric planes at the perpendicular to the center-plane located at x,y,z.
x,y,z	are the Cartesian coordinates of a point on the center-plane.
xp,yp,zp	are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/SYMPLN statement is passed to the output file by execution of the OUTPUT statement.

Note 1: Used if a feature of size (width) is located with its center-plane. Refer to (Figure B.93 — A symmetric plane feature).

Note 2: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.97 FEAT/TORRADSEGMNT

Function: Defines a feature nominal or constructs a feature actual toroidial radial segment and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/TORRADSEGMNT,var_1,var_2,majrad,minrad, $**
npi,npj,npk,latstartang,latstopang,pmi,pmj,pmk, $
longstartang,longstopang

or: **FA(lname)=FEAT/TORRADSEGMNT,var_1,var_2,majrad,minrad, $**
npi,npj,npk,latstartang,latstopang,pmi,pmj,pmk, $
longstartang,longstopang

Output Formats:

can be: **F(lname)=FEAT/TORRADSEGMNT,var_1,var_2,majrad,minrad, $**
npi,npj,npk,latstartang,latstopang,pmi,pmj,pmk, $
longstartang,longstopang

or: **FA(lname)=FEAT/TORRADSEGMNT,var_1,var_2,majrad,minrad, $**
npi,npj,npk,latstartang,latstopang,pmi,pmj,pmk, $
longstartang,longstopang

or: **FA(lname)=FEAT/TORRADSEGMNT,RAWDAT**
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT

or: **F(lname)[n]=FEAT/TORRADSEGMNT,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/TORRADSEGMNT,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,cx,cy,cz**
or: **POL,cr,ca,ch**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **var_4**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

cr,ca,ch are the polar coordinates of a center of the feature.

cx,cy,cz are the Cartesian coordinates of a center of the feature.

ENDAT signifies the end of raw data.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies internal radial segment feature.

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

latstartang is a real number angle, specifying the lower limit of the latitude of the toriodial radial segment.

latstopang is a real number angle, specifying the upper limit of the latitude of the toriodial radial segment.

longstartang is a real number angle, specifying the lower limit of the longitude of the toriodial radial segment.

longstopang is a real number angle, specifying the upper limit of the longitude of the toriodial radial segment.

n is a positive integer that is the index value of the individual point.

majrad is a positive real number representing the major radius of the torus.

minrad is a positive real number representing the minor radius of the torus.

npi,npj,npk is the north pole vector of the feature whose direction from the center of the torus specifies the positive 90° direction for latitude.

pmi,pmj,pmk is the prime meridian vector of the feature.

OUTER signifies external radial segment feature.

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data point(s).

TORRADSEGMNT signifies that the feature is a toroidial radial segment.

The location(s) and vectors(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/TORRADSEGMNT statement is passed to the output file by execution of the OUTPUT statement.

Note 1: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

Note 2: 3D radial segment features are partial cylinder, cone, sphere, or torus features that cannot be considered as a feature-of-size. Therefore it will NOT use a size tolerance, most likely a TOL/RAD or a TOL/PROFS. This 3D radial segment feature is similar to a 2D FEAT/ARC as compared to a 2D FEAT/CIRCLE.

Note 3: The ambiguity in the determination of the included angle is resolved using the right-hand rule of rotation (refer to 5.3.6.2) where the positive axis points in a direction of the north pole vector and the rotation about this axis vector is from the start angle toward the end angle relative to the prime meridian vector.

6.98 FEAT/TORUS

Function: Defines a nominal torus or constructs an actual torus, and assigns a label to it.

Input Formats:

can be: **F(lname)=FEAT/TORUS,var_1,var_2,i,j,k,diam1,diam2**

or: **FA(lname)=FEAT/TORUS,var_1,var_2,i,j,k,diam1,diam2**

Output Formats:

can be: **F(lname)=FEAT/TORUS,var_1,var_2,i,j,k,diam1,diam2**

or: **FA(lname)=FEAT/TORUS,var_1,var_2,i,j,k,diam1,diam2**

or:
```
FA(lname)=FEAT/TORUS,RAWDAT
/rx1,ry1,rz1
/rx2,ry2,rz2
/...
ENDAT
```

or: **F(lname)[n]=FEAT/TORUS,PTDATA,var_3 var_4**

or: **FA(lname)[n]=FEAT/TORUS,PTDATA,var_5,prbdiam var_6**

Where:

var_1 can be: **INNER**
or: **OUTER**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_3 can be: **CART,xp,yp,zp**
or: **POL,rp,ap,hp**

var_4 can be: **,in,jn,kn**
or: **does not exist**

var_5 can be: **var_3**
or: **RAWDAT,rx1,ry1,rz1**

var_6 can be: **,ip,jp,kp**
or: **does not exist**

CART signifies that Cartesian coordinates are to follow.

diam1 is a positive real number representing the major diameter of the torus, as measured through the center line of the minor diameter. Refer to (Figure B.44 — A torus feature).

diam2 is a positive real number representing the minor diameter of the torus. Refer to (Figure B.44 — A torus feature).

ENDAT signifies the end of raw data.

i,j,k is the direction vector of the plane in which the torus lies.

in,jn,kn is the nominal approach direction specified in the PTMEAS statement in programmed mode. Refer to section 5.3.2.6.

INNER signifies that the inside of the torus is to be measured (that is, an inner donut).

ip,jp,kp is the probe compensation direction of the specified point. Refer to section 5.3.2.6.

lname is an alphanumeric label name assigned to the feature.

n is a positive integer that is the index value of the individual point.

OUTER signifies that the outside of the torus is to be measured (that is, an outer donut).

POL signifies that polar coordinates are to follow.

prbdiam is a positive real number representing the actual probe diameter used to measure the specified point.

PTDATA signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).

r,a,h are the polar coordinates of the center-point of the torus.

RAWDAT signifies that the uncompensated, Cartesian coordinates of the measured point(s) (raw data) are to follow.

rp,ap,hp are the polar coordinates of the specified point.

rx1,ry1,rz1
rx2,ry2,rz2
... are the Cartesian coordinates of raw data point(s)

TORUS signifies that the feature is a torus.

x,y,z are the Cartesian coordinates of the center-point of the torus.

xp,yp,zp are the Cartesian coordinates of the specified point.

The location(s) and direction(s) are given relative to the current coordinate system.

For RAWDAT output, each data point is preceded with a slash and followed with a carriage return and line feed. The feature definition is terminated with the ENDAT statement. The data points are output in relation to the current coordinate system in effect when the feature was measured.

The FEAT/TORUS statement is passed to the output file by execution of the OUTPUT statement.

Note: The RAWDAT output format is used when the DME does not recognize the feature type and is given a measurement sequence to follow.

6.99 FEDRAT

Function: Used to set the velocities for measurements, path-directed moves, and rotary tables.

Input Formats:

can be: **FEDRAT/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **HEDMESVEL,var_2**
or: **HEDROTVEL,var_2**
or: **HEDSCNVEL,var_2**
or: **MESVEL,var_2**
or: **POSVEL,var_2**
or: **ROTVEL,var_3**
or: **SCNVEL,var_2**

var_2 can be: **MPM,n**
or: **MMPS,n**
or: **IPM,n**
or: **IPS,n**
or: **var_4**

var_3 can be: **RPM,n**
or: **var_4**

var_4 can be: **PCENT,n**
or: **HIGH**
or: **LOW**
or: **DEFALT**

DEFALT signifies the DME's default velocity.

HEDMESVEL signifies the sensor head’s measurement velocity.

HEDROTVEL signifies the sensor head’s rotational velocity.

HEDSCNVEL signifies the sensor head’s scanning velocity.

HIGH signifies the DME's internally stored high value.

IPM signifies that the value of 'n' is inches per minute.

IPS signifies that the value of 'n' is inches per second.

LOW signifies the DME's internally stored low value.

MESVEL signifies that the measurement velocity or the velocity of the sensor for measurement/contact moves is to be set.

MMPS signifies that the value of 'n' is millimeters per second.

MPM signifies that the value of 'n' is meters per minute.

n is a positive real number representing the velocity value.

PCENT signifies that the value of 'n' is the percent of maximum as a fraction less than or equal to 1.0 and greater than or equal to 0.01, for example, 0.75 means 75%.

POSVEL signifies that the positional velocity or the velocity of the sensor for path-directed moves is to be set.

ROTVEL signifies that the rotary table's rotational velocity is to be set.

RPM signifies that the value of 'n' is revolutions per minute.

SCNVEL signifies that the scanning velocity is to be set.

The following statements are interrelated in the use of rotary tables: ROTDEF, ROTSET, ROTAB, FEDRAT, ACLRAT, and CALIB.

6.100 FILDEF

Function: Defines a filter for use with a video DME, and assigns a label to it.

Input Formats:

Can be: **VF(lname)=FILDEF/CODE,n**

Output Formats:

Can be: **None**

Where:

CODE signifies that the filter is being defined with a numeric code.

lname is an alphanumeric label name assigned to the filter.

n is a positive integer value representing the previously coded filter.

The characterization file contains the various filters that are supported by the particular DME. Each of these filters is assigned an integer code in the characterization file.

Filters in this definition are considered to be physical filters. Mathematical filters are considered to be algorithms and are defined as such with the ALGDEF statement. The intent of this statement is to support those DMEs (particularly video devices), that maintain several alternate filters for use in the image acquisition or inspection process.

6.101 FILNAM

Function: Specifies an internal identification within the DMIS file.

Input Formats:

can be: **FILNAM/'text',version**

Output Formats:

can be: **FILNAM/'text',version**

Where:

'text' is a series of printable UTF8 characters, enclosed with apostrophes.

version defines the 'major.minor' revision of DMIS.

The format for 'major.minor' revision of DMIS will be XX.x, where XX is the major version number and x is the minor version number. The major version number shall be base 10 positive integer values, left padded with zeros. The minor version number shall be a single base 10 positive integer value or zero.

The FILNAM statement can only appear once in a DMIS input program and must be the first line in DMIS output. In the input program, the FILNAM statement must appear after the DISPLY statement. The FILNAM statement is considered an output header and will be output to all currently open DMIS output devices and subsequently opened DMIS output devices. User defined DMIS output devices are opened with the OPEN/DID(lname),FDATA,DMIS,OUTPUT... statement.

The FILNAM statement opens the STOR device if DISPLY/STOR has been selected.

The string of characters is preceded by an apostrophe and followed by an apostrophe. The two apostrophes define the limits of the string but are not characters of the string. When the string of characters must extend to another line, use a single dollar sign, '$', following the character data in the line. For example:

```
FILNAM/'041380385 is the part number this program inspects.',05.1
```

or:

```
FILNAM/'041380385 is the part number $
this program inspects.',05.1
```

Produce the following output:

```
FILNAM/'041380385 is the part number this program inspects.',05.1
```

When an apostrophe is required as part of the text, specify two apostrophes.

Example:

```
FILNAM/'041380385 is the part''s serial number intended for this $
special program.',05.1
```

The result of the above FILNAM input statement is the following output statement:

```
FILNAM/'041380385 is the part's serial number intended for this $
special program.',05.1
```

The FILNAM statement is passed to the output file when executed.

6.102 FINPOS

Function: Enables or disables the fine positioning feature. When enabled the carriage is positioned with high accuracy following the normal positioning move.

Input Formats:

can be: **FINPOS/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **ON**
or: **OFF**

OFF signifies that the fine positioning is to be disabled until the FINPOS/ON statement is issued.

ON signifies that the fine positioning is to be enabled until the FINPOS/OFF statement is issued.

6.103 FIXTID

Function: Defines the identification of a part holding fixture, and assigns a label to it.

Input Formats:

can be: **FI(lname)=FIXTID/'text'**

Output Formats:

can be: **FI(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the part holding fixture.

'text' is a text string, enclosed with apostrophes, that identifies the part holding fixture.

The FIXTID statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined part holding fixture label name.

6.104 FIXTSN

Function: Defines the identification of a part holding fixture's serial number, and assigns a label to it.

Input Formats:

can be: **FS(lname)=FIXTSN/'text'**

Output Formats:

can be: **FS(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the part holding fixture's serial number.

'text' is a text string, enclosed with apostrophes, that identifies the part holding fixture's serial number.

The FIXTSN statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined part holding fixture serial number label name.

6.105 FLY

Function: Sets or disables the FLY mode of the controller for continuous motion via points specified by the GOTO statement or implied by measurement approach points.

Input Formats:

can be: **FLY/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **radius**
or: **OFF**

OFF signifies that the FLY mode of the controller is to be disabled.

radius is a positive real number representing the maximum spherical radius around an intermediate point in which the sensor path can deviate from the intermediate point. This also signifies that the FLY mode of the controller is to be enabled.

Refer to (Figure 19 — FLY, radius example) and (Figure 20 — FLY mode example).

6.106 FROM

Function: Defines the home position to be used by the GOHOME statement.

Input Formats:

can be: **FROM/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **CART,x,y,z**
or: **CART,x1,y1,z1,RAM**
or: **DME**
or: **POL,r,a,h**
or: **POL,r1,a1,h1,RAM**
or: **SCALE**
or: **x,y,z**
or: **x1,y1,z1,RAM**

CART signifies that Cartesian coordinates are to follow.

DME signifies that the DME will move to a DME defined location.

POL signifies that polar coordinates are to follow.

r,a,h are the polar coordinates of the sensor position in the current coordinate system in the current working plane.

r1,a1,h1 are the polar coordinates of the ram position in the current coordinate system.

RAM signifies that the DME will move the physical sensor mounting point to the specified coordinate.

SCALE signifies that the DME will move to the scale reference origin.

x,y,z are the Cartesian coordinates of the sensor position in the current coordinate system.

x1,y1,z1 are the Cartesian coordinates of the ram position in the current coordinate system.

6.107 GEOALG

Function: Defines and sets a substitute feature data fitting algorithm for a particular feature type.

Input Formats:

can be: **GEOALG/var_1 var_2 var_3**

Output Formats:

can be: **GEOALG/var_1**

Where:

var_1 can be: **ARC,var_4**
or: **CIRCLE,var_5**
or: **CONE,var_5**
or: **CONRADSEGMNT,var_5**
or: **CPARLN,var_6**
or: **CYLNDR,var_5**
or: **CYLRADSEGMNT,var_5**
or: **ELLIPS,var_4**
or: **ELONGCYL,var_5**
or: **GCURVE,var_7**
or: **GSURF,var_8**
or: **LINE,var_4**
or: **OBJECT,var_6**
or: **PARPLN,var_5**
or: **PLANE,var_4**
or: **RCTNGL,var_4**
or: **REVSURF,var_9**
or: **SPHERE,var_5**
or: **SPHRADSEGMNT,var_5**
or: **SYMPLN,var_5**
or: **TORUS,var_5**
or: **TORRADSEGMNT,var_5**

var_2 can be: **,ELIMINATE,var_10**
or: **does not exist**

var_3 can be: **,FILTER,var_11**
or: **does not exist**

var_4 can be: **LSTSQR**
or: **MINMAX**
or: **DEFALT**
or: **EXTERN,var_12**

var_5 can be: **LSTSQR**
or: **MAXINS**
or: **MINCIR**
or: **MINMAX**
or: **DEFALT**
or: **EXTERN,var_12**

var_6 can be: **DEFALT**
or: **EXTERN,var_12**

var_7 can be: **LSTSQR**
or: **BSPLIN**
or: **MINMAX**
or: **DEFALT**
or: **EXTERN,var_12**

```
 var_8 can be:  LSTSQR
          or:  BEZIER
          or:  NURBS
          or:  MINMAX
          or:  DEFALT
          or:  EXTERN,var_12

 var_9 can be:  LSTSQR
          or:  BSPLIN
               DEFALT
               EXTERN,var_12

var_10 can be:  STDDEV_LIMIT,dev_val
          or:  OFF

var_11 can be:  LAMBDAC,var_13
          or:  CIRCULAR,var_13
          or:  OFF

var_12 can be:  DMIS,M(lname1) var_14
          or:  DME,'routine' var_14
          or:  SYS,'pathname' var_14

var_13 can be:  LOWPASS,wave_val,var_15
          or:  HIGHPASS,wave_val,var_15
          or:  BANDPASS,wave_val,wave_val,var_15

var_14 can be:  ,parameter var_16

var_15 can be:  GAUSS
          or:  2RC
          or:  SPLINE
          or:  RECFILT

var_16 can be:  var_14
          or:  does not exist
```

2RC signifies that a 2RC curve is used as the filter function.

ARC signifies that a circular arc substitute feature algorithm is to be defined.

BANDPASS signifies that a band pass filter is used.

BEZIER signifies a Bezier spline fit substitute feature algorithm.

BSPLIN signifies a B-spline fit substitute feature algorithm.

CIRCLE signifies that a circle substitute feature algorithm is to be defined.

CIRCULAR signifies that filtering is done by Undulations Per Revolution (UPR).

CONE signifies that a cone substitute feature algorithm is to be defined.

CONRADSEGMNT signifies that a conical radial segment substitute feature algorithm is to be defined.

CPARLN signifies that a closed-ended parallel feature algorithm is to be defined.

CYLNDR signifies that a cylinder substitute feature algorithm is to be defined.

CYLRADSEGMNT signifies that a cylindrical radial segment substitute feature algorithm is to be defined.

DEFALT signifies the system's default substitute feature algorithm.

dev_val is a real number representing the standard deviation multiplier to be applied as the limit for data spikes.

DME signifies a routine written in the DME language.

DMIS signifies a routine written in the DMIS language.

ELIMINATE	signifies that the parameters for elimination of spikes of measured data are to be defined. If point buffering is enabled when a feature actual is created with this in effect, the point buffer will contain only that individual point information that was used in creating the feature actual.
ELLIPS	signifies that an ellipse substitute feature algorithm is to be defined.
ELONGCYL	signifies that an elongated cylinder substitute feature algorithm is to be defined.
EXTERN	signifies a call to invoke the execution of an external routine. This external routine could consist of DMIS statements, DME specific statements, or any high-level language.
FILTER	signifies that the parameters for filtering of scanned data are to be defined. If point buffering is enabled when a feature actual is created with this in effect, the point buffer will contain only that individual point information that was used in creating the feature actual.
GAUSS	signifies that a Gaussian curve is used as a filter function.
GCURVE	signifies that a generic curve substitute feature algorithm is to be defined.
GSURF	signifies that a generic surface substitute feature algorithm is to be defined.
HIGHPASS	signifies that a high pass filter is used.
LAMBDAC	signifies that filtering is done by wavelength.
LINE	signifies that a line substitute feature algorithm is to be defined.
LOWPASS	signifies that a low pass filter is used.
LSTSQR	signifies a least squares fit substitute feature algorithm.
M(lname1)	is the label of a previously defined macro.
MAXINS	signifies a maximum size inscribed fit substitute feature algorithm.
MINCIR	signifies a minimum size circumscribed fit substitute feature algorithm.
MINMAX	signifies a min/max two sided fit substitute feature algorithm.
NURBS	signifies a Non-Uniform Rational B-Splines fit substitute feature algorithm.
OBJECT	signifies that a "user created" feature is to be defined.
OFF	signifies that filtering for the elimination of data spikes is to be disabled.
parameter	is the parameter that will be substituted in place of the called routine's parameter list. A parameter can be a value,(lname), or declared variable name.
PARPLN	signifies that a feature of linear size (slot, block) substitute feature algorithm is to be defined.
'pathname'	is the external routine's pathname, enclosed with apostrophes.
PLANE	signifies that a plane substitute feature algorithm is to be defined.
RCTNGL	signifies that a rectangle substitute feature algorithm is to be defined.
RECFILT	signifies that a rectangle is used as a filter function.
REVSURF	signifies that a surface-of-revolution algorithm is to be defined.
'routine'	is the external routine's name, enclosed with apostrophes.
SPHERE	signifies that a sphere substitute feature algorithm is to be defined.
SPHRADSEGMNT	signifies that a spherical radial segment substitute feature algorithm is to be defined.
SPLINE	signifies that a spline is used as a filter function.
STDDEV_LIMIT	signifies that the elimination parameters are defined by multiples of the standard deviation.
SYMPLN	signifies that a feature of linear size (slot, block) substitute feature algorithm is to be defined.
SYS	signifies a routine written in a system language (for example, C, Fortran, and Pascal).
TORRADSEGMNT	signifies that a toroidal radial segment substitute feature algorithm is to be defined.

TORUS signifies that a torus substitute feature algorithm is to be defined.

wave_val is a real number representing the wavelength for LamdaC filters or undulations per revolution for a circular filter.

The characterization file contains the various algorithms that are supported by the particular DMEs.

The active GEOALG determines the accessible individual point information. If point buffering is enabled when a feature actual is created all of the individual point information used to create that feature actual and only that individual point information used to create that feature actual will be accessible in the point buffer.

Refer to ISO 11562:1996 Geometrical Product Specifications (GPS) – Surface texture: Profile method – Metrological characteristics of phase correct filters.

For roundness, a minimum of 3600 points (10 points per degree) are required.

The GEOALG statement is passed to the output file when executed.

Note: The application of the minor word FILTER is only useful for scanned features.

6.108 GEOM

Function: Defines the CAD geometry data associated with features, and assigns a label to it.

Input Formats:

can be: **G(lname)=GEOM/var_1**

Output Formats:

can be: **G(lname)=GEOM/var_1**

Where:

var_1 can be: **DID(lname1)**
or: **G(lname2) var_2**
or: **NONE**

var_2 can be: **,ENTITY,'identity' var_3**
or: **,ENTITY,'identity',OFFSET,dist var_3**
or: **,OFFSET,dist**
or: **does not exist**

var_3 can be: **,'identity' var_3**
or: **,'identity',OFFSET,dist var_3**
or: **does not exist**

DID(lname1) is the device identification label assigned by the DEVICE statement to an opened system file or database containing the CAD data.

dist is a non-zero real number representing the distance that this geometry definition is offset from the surface of the base geometry.

ENTITY signifies that the geometry being defined is made up of the geometric entities in the base geometry associated with the next parameter.

filename is the name of the CAD file containing the part geometry, enclosed with apostrophes. This can be a filename and extension or it can be a full path specification.

G(lname2) is the label of the previously defined geometry on which this definition is based.

'identity' is the identity associated with the geometric entities in the base geometry, enclosed with apostrophes, which form this definition.

lname is an alphanumeric label name assigned to the geometry.

NONE signifies that geometric data is not defined. Any features associated with this geometry are defined by their nominals alone. This allows statements such as LOCATE to still reference many features by a G(lname) when no part geometry is used.

OFFSET signifies that the geometry being defined is offset by the next parameter away from the surface of the base geometry.

The GEOM statement is passed to the output file when it is executed.

Note: The purpose of this statement is to enable features, usually surface or edge points, to be associated with part model geometry. These associations are used in the MATDEF and FEAT/GEOM statements.

6.109 GOHOME

Function: Used to position the DME at the coordinates defined in the previous FROM statement.

Input Formats:

can be: **GOHOME**

Output Formats:

can be: **None**

6.110 GOTARG

Function: Initiates execution of a path-directed move and defines the endpoint or destination to which the sensor will travel.

Input Formats:

can be: **GOTARG/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **x,y,z**
or: **CART,x,y,z**
or: **POL,r,a,h**

CART signifies that Cartesian coordinates are to follow.

POL signifies that polar coordinates are to follow.

r,a,h are the polar coordinates of the endpoint to which the sensor will travel relative to the origin of the current coordinate system in the current working plane.

x,y,z are the Cartesian coordinates of the endpoint to which the sensor will travel relative to the origin of the current coordinate system.

This statement is followed by at least two GOTO statements which define the required moves to safely travel to the given endpoint. The GOTARG sequence, consisting of the GOTARG statement and two or more GOTO statement(s), is terminated with an ENDGO statement. The coordinates in the last GOTO statement in the sequence must always have the same coordinates as those in the GOTARG statement. Only GOTO statements can be used in a GOTARG…ENDGO block.

6.111 GOTO

Function: Executes a sensor move and defines the endpoint of the move.

Input Formats:

can be: **GOTO/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **x,y,z var_2**
or: **CART,x,y,z var_2**
or: **INCR,dist,i,j,k**
or: **ARC,x1,y1,z1,x,y,z**
or: **ARC,x1,y1,z1,x,y,z,var_3**
or: **POL,r,a,h var_2**
or: **var_4 var_5 var_5**
or: **var_3**

var_2 can be: **,SA(lname)**
or: **,var_6**
or: **,var_3**
or: **does not exist**

var_3 can be: **PCS,z3,y3,z4**
or: **HEADCS,z5,y4**
or: **HEADCS,z5,y4,z6**
or: **F(lname1) var_7**
or: **FA(lname2) var_7**
or: **VEC,i2,j2,k2 var_7**

var_4 can be: **XAXIS,x2**
or: **YAXIS,y2**
or: **ZAXIS,z2**

var_5 can be: **,var_4**
or: **does not exist**

var_6 can be: **SW(lname),'anglename',ang,var_6**
or: **SW(lname),'anglename',ang**

var_7 can be: **,FZ,dev**
or: **does not exist**

ang is the angle for the wrist axis identified by 'anglename'.

anglename is the name of the wrist angle to rotate, enclosed with apostrophes.

ARC signifies that the DME will perform a circular move which passes from the current location through the intermediate point and terminates at the endpoint.

CART signifies that Cartesian coordinates are to follow.

dev is the maximum allowed angular deviation between the target axis and the final sensor orientation. The format of 'dev' must be consistent with the current setting for angles, set with the UNITS statement.

dist is a real number representing the distance to be moved relative to the current sensor position.

F(lname1)
FA(lname2)
is the label of an oriented feature, the orientation of which to use for rotation of the wrist. The DME will choose the closest rotation angle for the wrist, as calculated from the nominal wrist definition.

If more than one angle is possible to get the sensor aligned to the feature, the rotation which creates the smallest rotation from the current wrist's angular position is performed. Refer to (Figure B.80 — SNSLCT, sensor selection illustration).

FZ
signifies that the value for the allowed angular deviation is to follow.

HEADCS
signifies that the orientation of the probe at the end of the move will be specified by two or three angles of rotation in probe head coordinates.

When specifying HEADCS, the number of angles of rotation specified will match the number of physical axes of the probe head. The angles are always specified as absolute values, local to the probe head coordinates.

i,j,k
is the direction vector to use for rotation of the wrist. The DME will choose the closest rotation angles for the wrist, as calculated from the nominal wrist definition.

If more than one angle is possible to get the sensor aligned to the vector, the rotation which creates the smallest rotation from the current wrist's angular position is performed. Refer to (Figure B.80 — SNSLCT, sensor selection illustration).

i2,j2,k2
is the direction vector along which the sensor will move.

INCR
signifies that the move will be relative to the present position.

PCS
signifies that the orientation of the probe at the end of the move will be specified by three angles of rotation in part coordinates.

PCS signifies that the axis system is rotated relative to the current part coordinate system: the axis system is rotated around Z by the first angle, then around the newly created Y axis, and finally around the newly created Z (which is the ZYZ Euler angle convention).

The resultant head orientation will be such that the active probe tip shaft (or the main direction of an optical sensor) is aligned with the Z axis pointing towards the sensor. The alignment of the tool with respect to the X and Y axes will be tool dependent, e.g. to align a laser plane.

POL
signifies that polar coordinates are to follow.

r,a,h
are the polar coordinates of the endpoint to which the sensor will travel relative to the origin of the current coordinate system in the current working plane.

SA(lname)
is the label of an actual sensor to be activated during the GOTO move.

SW(lname)
is the label of a wrist to be rotated during the GOTO move. The wrist must be attached to the specified (for S(lname) and SA(lname) formats) or current sensor (if no sensor is specified) via the SNSDEF statement.

VEC
signifies that a direction vector to use for rotation of the wrist is to follow.

x,y,z
are the Cartesian coordinates of the endpoint to which the sensor will travel relative to the origin of the current coordinate system.

x1,y1,z1
are the Cartesian coordinates of an intermediate point of a circular arc through which to travel.

x2
is the X coordinate to which the sensor will travel relative to the origin of the current coordinate system.

XAXIS
signifies motion in the X axis.

y2
is the Y coordinate to which the sensor will travel relative to the origin of the current coordinate system.

YAXIS
signifies motion in the Y axis.

z2 is the Z coordinate to which the sensor will travel relative to the origin of the current coordinate system.

z3,y3,z4 are the Euler angles of rotation which define the axis system of the probe at the end of the move. If a target position is not explicitly defined (for example, as an xyz position), then the current position is maintained.

z5,y4 are the angles of rotation of the 2 axis probe in probe coordinates at the end of the move. If a target position is not explicitly defined (for example, as an xyz position), then the current position is maintained.

z5,y4,z6 are the angles of rotation of the 3 axis probe in probe coordinates at the end of the move. If a target position is not explicitly defined (for example, as an xyz position), then the current position is maintained.

ZAXIS signifies motion in the Z axis.

All parameters must be included in the statement, or an error condition will result, for example, GOTO/,,1 is not valid.

Unless otherwise specified, the motion is linear between the current location and the point specified in the GOTO statement.

Single or multiple axis moves can be performed for clearance moves by using the form GOTO/XAXIS,20 or GOTO/ZAXIS,20,YAXIS,30. The DME will maintain the current position for the axis or axes not specified in the statement. Each axis can only be specified once.

Note: The use of SA(lname) in the GOTO statement allows for the simultaneous move of both the DME and sensor to the designated GOTO position.

6.112 GROUP

Function: Builds a sensor group from previously calibrated sensors, and assigns a label to it.

Input Formats:

can be: **GSA(lname)=GROUP/SA(lname1),SA(lname2) var_1**

Output Formats:

can be: **GSA(lname)=GROUP/SA(lname1),SA(lname2) var_1**

Where:

var_1 can be: **,SA(lname3) var_1**
or: **does not exist**

lname is an alphanumeric label name assigned to the sensor group.

SA(lname1)
SA(lname2)
SA(lname3) are the labels of previously calibrated sensors.

The GROUP statement is passed to the output file when executed.

6.113 IF

Function: Transfers the control of the program based upon the conditional execution of a logical expression. The expression can be comprised of previously defined variables, arithmetic expressions, or logical expressions but must evaluate to either .TRUE. or .FALSE.. Nested IF…ENDIF blocks are also supported.

Input Formats:

can be: **IF/(var_1)**

Output Formats:

can be: **None**

Synopsis:

IF/(var_1)
executable statement(s)
ELSE
executable statement(s)
ENDIF

Where:

var_1 can be: **varname**
or: **expr**

expr is any expression comprised of relational and/or logical expressions.

varname is any previously declared and assigned variable of type BOOL.

A logical IF must begin with an IF and end with an ENDIF statement. ELSE statements are optional; however, there may be several ELSE statements in a logical IF sequence when IF statements are nested. When IF statements are nested, care should be exercised to include corresponding ENDIF statements for every IF statement. JUMPTO statements can be used anywhere within a block IF to transfer control out of the block.

If the logical expression evaluates to .TRUE., control of the program is passed to the statement(s) following the IF statement and executed until an ELSE or ENDIF is encountered. If the logical expression is .FALSE., control of the program is passed to the statements following the next occurrence of ELSE. If an ENDIF is encountered before an ELSE, the conditional is ended.

6.114 INCLUD

Function: To include line-for-line external code into the DMIS program from an external file.

Input Formats:

can be: **INCLUD/var_1,'pathname'**

Output Formats:

can be: **None**

Where:

var_1 can be: **DMIS**
or: **DME**

DME signifies inclusion of native DME language statements.

DMIS signifies inclusion of DMIS statements.

'pathname' is a fully qualified pathname of the external file containing included code, enclosed with apostrophes.

For example, '/usr/dme/myincludefile'

The included statements will be inserted at the exact location of the include request. The included code shall not include DMISMN, DMISMD or ENDFIL statements.

6.115 Intrinsic functions

Function: Used to implement numeric functions, character functions, and system functions.

Input Formats:

can be: **var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **ABS(x)**
or: **ACOS(x)**
or: **ASIN(x)**
or: **ATAN(x)**
or: **ATAN2(y,x)**
or: **BADGT()**
or: **BADPT()**
or: **CHR(x)**
or: **CONCAT(str var_2)**
or: **COS(x)**
or: **DBLE(x)**
or: **DTOR(x)**
or: **ELEMNT(x,char,str)**
or: **EOF(DID(lname1))**
or: **EOLN(DID(lname1))**
or: **EXIST(var_3)**
or: **EXP(x)**
or: **INDX(str,sstr)**
or: **INT(x)**
or: **LEN(str)**
or: **LN(x)**
or: **LOG(x)**
or: **LWC(str)**
or: **MN(x var_4)**
or: **MOD(x,y)**
or: **MX(x var_4)**
or: **NINT(x)**
or: **ORD(x)**
or: **PTDATA(FA(lname2))**
or: **QTEMP(tempsns)**
or: **RAND(var_5)**
or: **RL(x)**
or: **RPT(str,n)**
or: **RTOD(x)**
or: **SCFEAT()**
or: **SCSNS()**
or: **SDATE()**
or: **SDATETIME()**
or: **SELAPSETIME(str1,str2,var_6)**
or: **SENSNOTOUCH()**
or: **SERROR()**
or: **SIGN(x,y)**
or: **SILTCH()**
or: **SIN(x)**
or: **SMODE()**
or: **SQRT(x)**
or: **STIME()**
or: **STR(x var_7)**
or: **SUBSTR(str,x var_8)**
or: **TAN(x)**
or: **TRIM(str)**
or: **UPC(str)**
or: **VAL(str)**
or: **VCART(x,y,z)**
or: **VCROSS(v1,v2)**
or: **VDOT(v1,v2)**
or: **VECX(v)**
or: **VECY(v)**
or: **VECZ(v)**
or: **VMAG(v)**
or: **VMCS(v)**
or: **VPCS(v)**
or: **VPOL(r,a,h)**
or: **VUNIT(v)**

var_2 can be: **,str var_9**

var_3 can be: **DID(lname2)**
or: **F(lname2)**
or: **FA(lname2)**
or: **D(lname2)**
or: **DA(lname2)**
or: **S(lname2)**
or: **SA(lname2)**
or: **does not exist**

var_4 can be: **,x var_10**

var_5 can be: **seed**
or: **does not exist**

```
 var_6 can be:  LONG
           or:  SHORT

 var_7 can be:  ,width:decimal
           or:  ,width
           or:  does not exist

 var_8 can be:  ,y
           or:  does not exist

 var_9 can be:  var_2
           or:  does not exist

var_10 can be:  var_4
           or:  does not exist
```

ABS(x) signifies a function that returns the absolute value of the numeric expression 'x' as a number of type INTGR, LONG, REAL or DOUBLE, depending on the numeric type of the expression 'x'.

ACOS(x) signifies a function that returns the arccosine of the numeric expression 'x' as a number of type DOUBLE. The expression 'x' must evaluate to a number between –1.0 and +1.0. The return value is the angle in radian units and is in the range 0 angle π.

ASIN(x) signifies a function that returns the arcsine of the numeric expression 'x' as a number of type DOUBLE. The expression 'x' must evaluate to a number between –1.0 and +1.0. The return value is the angle in radian units and is in the range -π / 2 angle π / 2.

ATAN(x) signifies a mathematical function that returns the arctangent of the numeric expression 'x' as a number of type DOUBLE. The expression 'x' must evaluate to a number between –1.0 and +1.0. The return value is the angle in radian units and is in the range - π / 2 < angle < π / 2.

ATAN2(y,x) signifies a function that returns the angle defined by the numeric expressions 'y' and 'x' as a number of type DOUBLE. The expressions 'y' and 'x' cannot simultaneously evaluate to 0. The numbers 'y' and 'x' are taken to represent a point in a Cartesian coordinate system and the angle returned is the angle between the positive X axis of the Cartesian coordinate system and a line from the origin (0,0) to the point (x, y). The return value is an angle in the range -π ≤ return value ≤ π.

BADGT() signifies a function that returns a value indicating the error status of the most recently executed GOTO statement as a type BOOL. The return value is .TRUE. if an unexpected sensor collision occurred and .FALSE. if no unexpected sensor collision occurred. This return value of this function is valid only if sensor collision polling has been enabled with the BADTST/ON statement and a GOTO statement has been previously executed.

BADPT() signifies a function that returns a value indicating the error status of the most recently executed point feature measurement as a type BOOL. The return value is .TRUE. if the point measurement failed and .FALSE. if the point measurement succeeded. This return value of this function is valid only if point measurement polling has been enabled with the BADTST/ON statement and a MEAS/POINT...ENDMES block has been previously executed.

CHR(x) signifies a function that returns the single ASCII character corresponding to the value of the integer numeric expression 'x' as a text string of type CHAR. The expression 'x' must evaluate to an integer in the range 1 to 255.

CONCAT(str var_2) signifies a function that returns the concatenation of two or more character expressions as a text string of type CHAR. Literal text strings are concatenated left to right without their enclosing quotation marks.

For example, CONCAT('MY','_','COMMENT') returns 'MY_COMMENT', not 'MY""_""COMMENT' or 'COMMENT_MY'. The length of the returned text string is the sum of the lengths of all the character expressions.

COS(x) signifies a function that returns the cosine of the angle represented by the numeric expression 'x' as a number of type DOUBLE. The angle represented by the numeric expression 'x' is interpreted in radian units.

DBLE(x) signifies a function that returns the value of the numeric expression 'x' as a number of type DOUBLE.

decimal is a positive integer representing the number of digits to the right of the decimal point.

DTOR(x) signifies a function that converts the angle in degrees represented by the numeric expression 'x' into a the angle in radians as a number of type DOUBLE.

ELEMNT(x,char,str) signifies a function that returns an element of a delimited list as a text string of type CHAR. The element number to be returned is defined by the integer numeric expression 'x'. The list delimiter is the single character text string defined by the character expression 'char'. And, the text string representing a delimited list is defined by the character expression 'str'. The integer numeric expression must evaluate to a positive, non-zero number. If the element defined by 'x' does not exist in the string specified by 'str' then an empty string is returned.

For example ELEMNT(5,';','POINT;12;2.3;14;+Z') would return the text string '+Z'.

To extract a member of a delimited list as a number use the VAL intrinsic function on the text string returned by ELEMNT.

For example VAL(ELEMNT(3,';','POINT;12;2.3;14;+Z')) would return the number 2.3.

EOF(DID(lname1)) signifies a function that returns the end of file (EOF) condition of the device defined by DID(lname1) as a value of type BOOL. The value .TRUE. is returned if an end of file condition exists and the value .FALSE. is returned if no end of file condition exits.

EOLN(DID(lname1)) signifies a function that returns the end of line condition of the device defined by DID(lname1) as a value of type BOOL. The value .TRUE. is returned if an end of line condition exists and the value .FALSE. is returned if no end of line condition exits.

EXIST(D(lname2)) signifies a function that returns the state of existence of the nominal coordinate system defined by D(lname2) as a value of type BOOL. .TRUE. is returned if the nominal coordinate system exists and .FALSE. is returned if the coordinate system does not exist.

EXIST(DA(lname2)) signifies a function that returns the state of existence of the actual coordinate system defined by DA(lname2) as a value of type BOOL. .TRUE. is returned if the actual coordinate system exists and .FALSE. is returned if the coordinate system does not exist.

EXIST(DID(lname2)) signifies a function that returns the state of existence of the device defined by DID(lname2) as a value of type BOOL. .TRUE. is returned if the device exists and .FALSE. is returned if the device does not exist.

EXIST(F(lname2)) signifies a function that returns the state of existence of the feature nominal defined by F(lname2) as a value of type BOOL. .TRUE. is returned if the feature nominal exists and .FALSE. is returned if the feature nominal does not exist.

EXIST(FA(lname2)) signifies a function that returns the state of existence of the feature actual defined by FA(lname2) as a value of type BOOL. .TRUE. is returned if the feature actual exists and .FALSE. is returned if the feature actual does not exist.

EXIST(S(lname2)) signifies a function that returns the state of existence of the sensor nominal defined by S(lname2) as a value of type BOOL. .TRUE. is returned if the sensor nominal exists and .FALSE. is returned if the sensor nominal does not exist.

EXIST(SA(lname2)) signifies a function that returns the state of existence of the sensor actual defined by SA(lname2) as a value of type BOOL. .TRUE. is returned if the sensor actual exists and .FALSE. is returned if the sensor actual does not exist.

EXP(x) signifies a function that returns the exponential value of the numeric expression 'x' as a number of type DOUBLE. The return value is e(=2.7182818...) raised to the power 'x'.

INDX(str,sstr) signifies a function that returns the location of the beginning of the first occurrence of the sub-string defined by the character expression 'sstr' within the text string defined by the character expression 'str' as a number of type INTGR. If the sub-string 'sstr' does not occur in the string 'str' then the number 0 is returned.

For example INDX('To be or not to be', 'be') will return the number 4.

INT(x) signifies a function that returns the integer portion of the number defined by the numeric expression 'x' as a number of type INTGR. The return value is determined by truncation, not by rounding.

LEN(str) signifies a function that returns the length of a text string defined by the character expression 'str' as a number of type INTGR. The length of a literal string does not include the enclosing apostrophes.

For example LEN('xyz') returns the number 3.

LN(x) signifies a function that returns the natural (base e=2.7182818...) logarithm of the numeric expression 'x' as a number of type DOUBLE. The numeric expression 'x' must evaluate to a number greater than 0.

LOG(x) signifies a function that returns the common (base 10) logarithm of the numeric expression 'x' as a number of type DOUBLE. The numeric expression 'x' must evaluate to a number greater than 0.

LONG signifies the elapsed time long format 'DDDD HH:mm:SS.sss', where DDDD is the number of days, HH is hours from 0 to 23, mm is minutes, SS is seconds, and sss is milliseconds.

LWC(str) signifies a function that returns the text string defined by the character expression 'str' converted to lowercase as a new text string of type CHAR.

MN(x var_4) signifies a function that returns the value of the smallest (most negative) number defined by the numeric expressions 'x' as a number of type DOUBLE.

MOD(x,y) signifies a function that returns the remainder of the numerical expression 'x' divided by the numerical expression 'y' as a number of type DOUBLE. The result is the same as that of x-(INT(DBLE(x)/DBLE(y))*y). The numerical expression 'y' must evaluate to a non-zero number.

MX(x var_4) signifies a function that returns the value of the largest (most positive) number defined by the numeric expressions 'x' as a number of type DOUBLE.

NINT(x) signifies a function that returns the nearest integer value to the numeric expression 'x' as a number of type INTGR.

ORD(x) signifies a function that returns the ordinal value of the character or Boolean expression 'x' as a number of type INTGR. When the expression 'x' is a Boolean expression, then 0 is returned if 'x' evaluates to .FALSE. and 1 is returned if 'x' evaluates to .TRUE.. When the expression 'x' is a character expression, the ASCII numeric value of the first character in the string defined by 'x' is returned.

PTDATA(FA(lname2)) signifies a function that returns the number of points in the point buffer that were used in the creation of the feature actual defined by FA(lname2) as a number of type INTGR. If a feature was created by definition, by construction other than by BF or TR, or while PTBUFF was off, the return value will be 0.

QTEMP(tempsns) signifies a function that returns the value from the temperature sensor defined by the character expression 'tempsns' in current temperature units as a number of type DOUBLE.

RAND() signifies a function that returns a random number between 0 and 1 as a number of type DOUBLE.

RL(x) signifies a function that returns the value of the numeric expression 'x' as a number of type REAL.

RPT(str,n)	signifies a function that returns a text string formed by repeating the text string defined by the character expression 'str' the number of times defined by the integer numeric expression 'n' as a new text string of type CHAR. The length of the returned text string is the product of the length of the string 'str' and the number 'n'. For example, RPT('-*',10) will return '-*-*-*-*-*-*-*-*-*-*'.
RTOD(x)	signifies a function that converts the angle in radians represented by the numeric expression 'x' into a the angle in degrees as a number of type DOUBLE.
SCFEAT()	signifies a function that returns the label name of the current feature being measured as a text string of type CHAR. If outside of a CALIB…ENDMES block, MEAS...ENDMES block or RMEAS…ENDMES block, a zero length string is returned.
SCSNS()	signifies a function that returns the label name of the current selected sensor as a text string of type CHAR.
SDATE()	signifies a function that returns the system's current date as a text string of type CHAR. The form of the returned string is 'YYYY/MM/DD' where YYYY is the year, MM is the number of the month, and DD is the day within the month.
SDATETIME()	signifies a function that returns the system's current date and time as a text string of type CHAR. The form of the returned string is 'YYYY/MM/DD HH:mm:SS.sss', where YYYY is the year, MM is the number of the month, DD is the day within the month, HH is the hour of the day between 0 and 23, mm is the minute between 0 and 59, SS is the second between 0 and 59, and sss is milliseconds.
seed	is an integer that serves as the "seed" of the random number generator. When seed does not exist the returned value is the next number in the sequence. The same seed will return the same sequence.
SELAPSETIME $ (str1,str2,var_6)	signifies a function which returns the elapsed time between a start time defined by the character expressions 'str1' and a stop time defined by the character expression 'str2' as a text string of type CHAR. The form of the text strings defined by the character expressions 'str1' and 'str2' is 'YYYY/MM/DD HH:mm:SS.sss', where YYYY is the year, MM is the number of the month, DD is the day within the month, HH is the hour of the day between 0 and 23, mm is the minute between 0 and 59, SS is the second between 0 and 59, and sss is milliseconds. The form of the returned string is determined by var_6.
SENSNOTOUCH()	signifies a function that returns the DME specific error code if no touch is detected as a number of type INTGR. DME specific error codes are specified in the characterization file.
SERROR()	signifies a function that returns the error condition when a DME error occurred as a text string of type CHAR. Return values can be 'ILLEGALTOUCH', 'NOTOUCH' or a DME specific error code represented as a string. DME specific error codes are specified in the characterization file.
SHORT	signifies the elapsed time short format 'HHHH:mm:SS.sss', where HHHH is hours from 0 to 9999, mm is minutes, SS is seconds, and sss is milliseconds.
SIGN(x,y)	signifies a function that returns the number defined by the numerical expression 'x' if the number defined by the numerical expression 'y' is zero or positive and the negative of the number defined by the numeric expression 'x' if the number defined by the numeric expression 'y' is negative as a number of type DOUBLE.
SILTCH()	signifies a function that returns the DME specific error code for an illegal touch as a number of type INTGR. DME specific error codes are specified in the characterization file.
SIN(x)	signifies a function that returns the sine of the angle represented by the numeric expression 'x' as a number of type DOUBLE. The angle represented by the numeric expression 'x' is interpreted in radian units.
SMODE()	signifies a function that returns the current system mode of operation as a text string of type CHAR. The return value will either be "AUTO", "PROG", or "MAN".

SQRT(x) signifies a function that returns the square root of the numeric expression 'x' as a number of type DOUBLE. The numeric expression 'x' must evaluate to a number greater than or equal to 0.

STIME() signifies a function that returns the current system time as a text string of type CHAR. The form or the returned string is 'HH:mm:SS', where HH is the hour from 0 to 23, mm is the minute from 0 to 59, and SS is the second from 0 to 59.

STR(x var_7) signifies a function that returns a character representation of the numeric expression 'x' as a text string of type CHAR. If no width or decimal places are specified as optional arguments then the returned string will have no spaces. Numeric expressions evaluating to an integer type will result in strings with no decimal places. Numeric expressions evaluating to a real number type will have a number of decimal places that are DME dependent. If the width is specified as an optional argument the returned string will have as many spaces to the left of the number as is necessary to achieve the specified width. If the number of characters needed to express the number exceeds the specified width then the same string is returned that would have been returned if the optional width argument had not been present. If both the width and decimal places are specified as optional arguments then the number will be displayed rounded to the specified number of decimal places with added zeros on the right as necessary. The returned string will have as many spaces to the left of the number as is necessary to achieve the specified width. If the number of characters needed to express the number with the specified number of decimal places exceeds the specified width then a string longer than the specified width is returned. In all cases negative numbers will appear with a negative sign to the immediate left of the first digit and positive number will appear with no sign. A decimal number less than 1 in absolute value will have a zero to the left of the decimal point, otherwise the first digit to the left of the decimal point will always be non-zero.

Refer to example (A.32 STR())

SUBSTR $ (str,x var_8) signifies a function that returns a portion of the string defined by the character expression 'str' beginning with the character at the position defined by the integer numeric expression 'x' and ending with the character at the position defined by the optional integer numeric expression 'y' or to the end of the string defined by the character expression 'str' if the optional argument 'y' is missing, as a text string of type CHAR. Both the expressions 'x' and 'y' must evaluate to positive, non-zero integer numbers. If the expression 'x' evaluates to a number greater than or equal to 'y' or if x evaluates to a number greater than the length of the string defined by 'str' then an empty string is returned. If 'y' evaluates to a number greater than the length of 'str' then the string starting at 'x' and going to the end of 'str' is returned as if 'y' had been omitted.

TAN(x) signifies a function that returns the tangent of the angle represented by the numeric expression 'x' as a number of type DOUBLE. The angle represented by the numeric expression 'x' is interpreted in radian units.

TRIM(str) signifies a function that returns the string defined by the character expression 'str' with all its leading

and trailing spaces stripped as a new text string of type CHAR.

UPC(str) signifies a function that returns the text string defined by the character expression 'str' converted to uppercase as a new text string of type CHAR.

VAL(str) signifies a function that converts a the string defined by the character expression 'str' representing a number into a number of type DOUBLE. The first non-blank character in the string must be a decimal digit (0-9), a plus or minus sign, or a decimal point. The remaining characters must be decimal digits or a decimal point and must form a valid literal number.

VCART(x,y,z) signifies a function which accepts three numeric expressions 'x', 'y' and 'z' defining x, y, and z Cartesian coordinates or components and returns a vector of type VECTOR.

VCROSS(v1,v2) signifies a function which returns the vector cross product of the vectors defined by the vector expressions 'v1 and 'v2' as a vector of type VECTOR.

VDOT(v1,v2) signifies a function which returns the vector dot product of the vectors defined by the vector expressions 'v1 and 'v2' as a number of type DOUBLE.

VECX(v) signifies a function returning the X component of the vector defined by the vector expression 'v' as a number of type DOUBLE.

VECY(v) signifies a function returning the Y component of the vector defined by the vector expression 'v' as a number of type DOUBLE.

VECZ(v) signifies a function returning the Z component of the vector defined by the vector expression 'v' as a number of type DOUBLE.

VMAG(v) signifies a function returning the magnitude (length) of the vector defined by the vector expression 'v' as a number of type DOUBLE.

VMCS(v) signifies a function which converts the vector defined by the vector expression 'v' from the current coordinate system to the DME machine coordinate system and returns a vector of type VECTOR.

VPCS(v) signifies a function which converts the vector defined by the vector expression 'v' from the DME machine coordinate system to the current coordinate system and returns a vector of type VECTOR.

VPOL(r,a,h) signifies a function which accepts three numeric expressions 'r', 'a' and 'h' defining r, a, and h polar coordinates or components in the current working plane and returns a vector of type VECTOR.

VUNIT(v) signifies a function which normalizes the length of the vector defined by the vector expression 'v' to unit length and returns a new vector as type VECTOR.

width is a positive integer representing the value of the total field width (including any negative sign).

The intrinsic functions are not major words. All return types depend on the argument types. Where required, argument types are converted in the precedence order INTGR to LONG to REAL to DOUBLE, using the functions RL() and DBLE(). Conversion to INTGR is performed using the function INT().

6.116 ITERAT

Function: To provide the capability of repeating a sequence of instructions in order to perform an alignment on a CMM. The syntax is based on a single or series of convergence conditions being met before a maximum number of iterations have occurred.

Input Formats:

can be: **varname=ITERAT/(jumptarget1),(jumptarget2),convergence,var_1 $**
,limit var_2 var_3

Output Formats:

can be: **ITERAT/(jumptarget1),(jumptarget2),act_converge,var_1,iterate_num**

Where:

var_1 can be: **ABSL**
or: **INCR**

var_2 can be: **,XAXIS**
or: **,YAXIS**
or: **,ZAXIS**
or: **,i,j,k**
or: **,NOM**

var_3 can be: **,FA(lname3) var_4**
or: **,FA(lname3)**

var_4 can be: **var_3**
or: **var_2 var_3**

(jumptarget1) is the iteration loop start point jumptarget.

(jumptarget2) is the failure mode jumptarget for notifying the user when the convergence condition(s) are not achieved.

ABSL signifies that the convergence represents the maximum (over all measured features) of the deviation between nominal and actual along the direction defined by var_2.

act_converge is a real number representing the actual convergence value for iterate_num.

convergence is a real number representing the convergence value that the conditional tests must achieve.

FA(lname3) is the measured alignment feature actual value to be tested.

i,j,k is the direction vector in which the DME is to test the convergence on the PCS for the features that follow.

INCR signifies that the convergence represents the maximum (over all measured features) of the difference between iterations, of the deviation between nominal and actual along the direction defined by var_2.

iterate_num is a positive integer representing the current iteration number.

limit is a positive integer representing the maximum value of iterations that will be attempted if the convergence value is not achieved. If the convergence is not achieved, the program branches to the program label defined in jumptarget2.

NOM signifies that the nominal vector of the feature specified by FA(lname3) is the direction vector in which the DME is to test the convergence on the PCS for the features that follow.

varname is the name of the previously declared variable or array element to which the actual convergence value is assigned.

XAXIS signifies that the DME is to test the convergence on the PCS X axis for the features that follow.

YAXIS signifies that the DME is to test the convergence on the PCS Y axis for the features that follow.

ZAXIS signifies that the DME is to test the convergence on the PCS Z axis for the features that follow.

The convergence value is a real number or real number variable and the limit must be a positive integer or positive integer variable.

The output format of this statement is generated for each iteration.

6.117 JUMPTO

Function: Used to transfer the control of the program to a labelled statement anywhere in the program.

Input Formats:

can be: **JUMPTO/(jumptarget)**

Output Formats:

can be: **None**

Where:

(jumptarget) is a label name enclosed in parentheses identifying a program line that contains only a label name enclosed in parentheses, with no label type, to which the program execution control will be transferred.

The JUMPTO statement breaks the sequential flow of execution and transfers control to the statement immediately following the jumptarget. The jumptarget may be before or after the JUMPTO statement, as long as it is in the same program unit.

6.118 KEYCHAR

Function: Defines a key characteristic that associates nominal feature(s) and nominal tolerance(s) with an optional key characteristic criticality designation and assigns a unique label to it.

Input Formats:

can be: **KC(lname)=KEYCHAR/ var_1 var_2**

Output Formats:

can be: **KC(lname)=KEYCHAR/ var_1 var_2**
or: **KCA(lname)=KEYCHAR/var_3 var_2**

Where:

var_1 can be: **F(lname1),T(lname2) var_4**
or: **F(lname3),F(lname6),T(lname4)**

var_2 can be: **,CRITICAL**
or: **,MAJOR**
or: **,MINOR**
or: **does not exist**

var_3 can be: **FA(lname1),TA(lname2) var_5**
or: **FA(lname3),FA(lname6),TA(lname4)**

var_4 can be: **,T(lname2) var_4**
or: **does not exist**

var_5 can be: **,TA(lname2) var_5**
or: **does not exist**

CRITICAL signifies a critical key characteristic designation.

F(lname1)
FA(lname1) is the label of the feature to be associated with the tolerance(s).

F(lname3)
FA(lname3) is the label of the first feature associated with a two-feature relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

F(lname6)
FA(lname6) is the label of the second feature associated with a two-feature relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

lname is an alphanumeric label name assigned to the key characteristic evaluation.

MAJOR signifies a major key characteristic designation.

MINOR signifies a minor key characteristic designation.

T(lname2)
TA(lname2) is the label of the tolerance associated with the feature.

T(lname4)
TA(lname4) is the label of the relationship tolerance, for example, TOL/ANGLB or TOL/DISTB, associated with the features.

Tolerances are generic, in that they have no pointers to features. Association between features and tolerances is provided for with the KEYCHAR, OUTPUT and EVAL statements. One or several tolerances can be associated with a feature. Two features are required to be associated with the angle between, TOL/ANGLB, and the distance between, TOL/DISTB, tolerances. The KEYCHAR statement defines a persistent key characteristic label that associates nominal tolerance(s) with feature(s) with an optional key characteristic criticality designation, before evaluation. Refer to Figure A.3 — Key Characteristics. The KEYCHAR feature(s) actuals and tolernace(s) can be evaluated via the EVAL or OUTPUT statements.

The KEYCHAR statement is passed to the output file by execution of the output statement.

6.119 LITDEF (input format 1)

Function: Defines the lighting arrangement to be used by a DME for taking measurements with a video device, and assigns a label to it.

Input Formats:

can be: **VL(lname)=LITDEF/var_1,i,j,k**

Output Formats:

can be: **None**

Where:

var_1 can be: **SURF**
or: **BACK**
or: **GRID**
or: **OBLQ**

BACK signifies projected light (back-lighting).

GRID signifies artificial contrast projection for automatic focusing.

i,j,k is the unit vector of the light axis in machine coordinates.

lname is an alphanumeric label name assigned to the light.

OBLQ signifies oblique reflected light.

SURF signifies normal reflected light.

Once the lighting requirements are defined, they are activated with the SNSET statement.

Note: The light definition is repeated with a new label if any of the parameters are adjustable.

For example:

```
VL(left_label)=LITDEF/OBLQ,-.707,0,-.707
VL(right_label)=LITDEF/OBLQ,.707,0,-.707
VL(top_label)=LITDEF/SURF,0,0,-1
```

6.120 LITDEF (input format 2)

Function: Defines the lighting arrangement to be used by a DME for taking measurements with a video device, and assigns a label to it.

Input Formats:

can be: **VL(lname)=LITDEF/STROBE,var_1,timeon,i,j,k**

Output Formats:

can be: **None**

Where:

var_1 can be: **CYCLE,value**
or: **TRIGER**

CYCLE signifies that the strobe lighting and image acquisition will be activated in cycles per minute as specified by 'value'.

i,j,k is the unit vector of the light axis in machine coordinates.

lname is an alphanumeric label name assigned to the light.

STROBE signifies strobe lighting, and image acquisition.

timeon is a positive real number representing the time-on duration in milliseconds.

TRIGER implies both strobe and image acquisition are activated by a trigger.

value is a real number representing the cycles per minute.

Note: Because STROBE is a function of both lighting and image acquisition, the timing conditions are critical. While the STROBE requirements are defined with this format of the LITDEF statement, the association between the light and the sensor is provided with the SNSET statement. The MEAS statement in conjunction with the parameters in the LITDEF statement invoke both lighting and image acquisition to coincide.

6.121 LOCATE

Function: Creates a part coordinate system that fits the nominals of a set of features to the actuals of a set of features and assigns a label to it.

Input Formats:

can be: **D(lname)=LOCATE/var_1 var_2 var_3**

Output Formats:

can be all: **D(lname)=LOCATE/var_1 var_2 var_3**
DA(lname)=LOCATE/TRMATX,a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3

Where:

var_1 can be: **XDIR,**
or: **YDIR,**
or: **ZDIR,**
or: **XYDIR,**
or: **YZDIR,**
or: **ZXDIR,**
or: **XYZDIR,**
or: **NOTRAN,**
or: **does not exist**

var_2 can be: **XAXIS,**
or: **YAXIS,**
or: **ZAXIS,**
or: **XYAXIS,**
or: **YZAXIS,**
or: **ZXAXIS,**
or: **XYZAXI,**
or: **NOROT,**
or: **does not exist**

var_3 can be: **FA(lname1) var_4**
or: **MA(lname2) var_4**
or: **DAT(lname3) var_4**

var_4 can be: **,FA(lname1) var_4**
or: **,MA(lname2) var_4**
or: **,DAT(lname3) var_4**
or: **does not exist**

a1,a2,a3, b1,b2,b3, c1,c2,c3, d1,d2,d3 satisfy the following transformation equations:

(a1)x + (b1)y + (c1)z + d1 = x'
(a2)x + (b2)y + (c2)z + d2 = y'
(a3)x + (b3)y + (c3)z + d3 = z'

Where: x', y', and z' are the coordinates in the current part coordinate system (after the transformation has been applied) of any point, and x, y, and z are the coordinates in the previous coordinate system of the same point.

DAT(lname3) is the label of a previously defined datum.

FA(lname1) is the label of a measured or constructed feature actual used to locate the part.

lname is an alphanumeric label name assigned to the new part coordinate system.

MA(lname2) is the label of a previously defined feature-feature mating.

NOROT signifies that no rotation is allowed.

NOTRAN signifies that no translation is allowed.

TRMATX signifies that the alignment and transformation information from the previous coordinate system is output.

XAXIS signifies that rotation about the X-axis is allowed.

XDIR signifies that translation along the X-axis is allowed.

XYAXIS signifies that rotation about the X and Y axes is allowed.

XYDIR signifies that translation in the XY plane is allowed.

XYZAXI signifies that rotation about all axes is allowed.

XYZDIR signifies that translation in all directions is allowed.

YAXIS signifies that rotation about the Y-axis is allowed.

YDIR signifies that translation along the Y-axis is allowed.

YZAXIS signifies that rotation about the Y and Z axes is allowed.

YZDIR signifies that translation in the YZ plane is allowed.

ZAXIS signifies that rotation about the Z-axis is allowed.

ZDIR signifies that translation along the Z-axis is allowed.

ZXAXIS signifies that rotation about the Z and X axes is allowed.

ZXDIR signifies that translation in the ZX plane is allowed.

If both var_1 and var_2 do not exist, then rotation about and translation along all axes is allowed.

The LOCATE statement is passed to the output file when executed.

6.122 LOTID

Function: Defines the identification of the part lot identifier, and assigns a label to it.

Input Formats:

Can be: **LI(lname)=LOTID/'text'**

Output Formats:

Can be: **LI(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the part lot.

'text' is a text string, enclosed with apostrophes, that identifies the part's lot.

The LOTID statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined part lot identifier label name.

6.123 MACRO

Function: Defines a macro routine, and assigns a label to it.

Input Formats:

can be: **M(lname)=MACRO var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **/varname var_2**
or: **/'lname1' var_2**
or: **does not exist**

var_2 can be: **,varname var_2**
or: **,'lname1' var_2**
or: **does not exist**

lname is an alphanumeric label name assigned to the macro being defined. This label name will be used by the CALL statement to invoke the macro.

'lname1' is a substitutional label name, enclosed with apostrophes.

varname is a substitutional variable name.

When the macro is called, the actual parameters in the CALL statement will be substituted for the arguments in the argument list. The argument list may be empty.

A macro argument used to substitute a label name is enclosed with apostrophes in the MACRO statement.

A call parameter used to substitute a label name is enclosed with parentheses in the CALL statement.

Substitutional argument and label name values are filled with the values of the corresponding parameters in the CALL statement calling the macro, and do not have to be unique within the program, but must be unique within the macro.

Variables used as parameters in a CALL statement are always passed by reference. This means the original variable is sent to the macro and if changed during execution, the variable will retain the changed value upon returning from the macro.

A macro is terminated with the ENDMAC statement, and executed with the CALL statement. Refer to clause 5.2.2.4 for further information.

6.124 MATDEF

Function: Specifies the parameters for a functional mating between a feature actual and a feature nominal or geometry definition and assigns a label to it.

Input Formats:

can be: **MA(lname)=MATDEF/var_1 var_2**

Output Formats:

can be: **MA(lname)=MATDEF/var_1 var_2**

Where:

var_1 can be: **F(lname1),FA(lname2) var_3**
or: **G(lname3),FA(lname2) var_3**

var_2 can be: **,PT2PT,var_4,var_5 var_6**
or: **,PT2LN,var_4,var_5 var_6**
or: **,PT2PL,var_4,var_5 var_6**
or: **,LN2LN,var_4,var_5 var_6**
or: **does not exist**

var_3 can be: **,FA(lname2) var_3**
or: **does not exist**

var_4 can be: **BF**
or: **FZ**

var_5 can be: **fitzon**
or: **loband,upband**

var_6 can be: **,MMC,T(lname4)**
or: **,LMC,T(lname4)**
or: **does not exist**

BF signifies that the deviation between mating features is to be minimized by the best fit method.

F(lname1) is the label of the feature nominal to which the feature actuals will be mated.

FA(lname2) is the label of the feature actual which is mated to the feature nominal or which is associated to the geometry data.

fitzon is a real number representing a fitting zone for deviations along the primary direction vector.

FZ signifies that the deviation for the mating features must be within the specified fit zone or band.

G(lname3) is the label of the geometry data to which the feature actuals will be associated. The nominals of these features are repositioned on the model geometry in the new alignment to be nearest to the actuals.

lname is an alphanumeric label name assigned to the mating definition.

LMC,T(lname4) signifies that least material condition is applied to the fit zone or band for the mating. T(lname4) is the label of a previously defined tolerance for the size of the actual mating feature.

LN2LN signifies that the features will be mated line to line so that the deviation between the features along the primary direction vector is significant.

loband is a real number representing the lower fit band for deviations along the primary direction vector.

MMC,T(lname4) signifies that maximum material condition is applied to the fit zone or band for the mating. T(lname4) is the label of a previously defined tolerance for the size of the actual mating feature.

PT2LN signifies that the features will be mated point to line so that the deviation between the features in a plane normal to the primary direction vector is significant.

PT2PL signifies that the features will be mated point to plane so that the deviation between the features along the primary direction is significant.

PT2PT signifies that the features will be mated point to point so that the deviation between the features in any direction is significant.

upband is a real number representing the upper fit band for deviations along the primary direction.

For LN2LN mating, the direction vector is taken to be the feature nominal vector crossed with the feature actual vector.

The value of fitzon, upband and loband define how close to nominal a feature must be held during a LOCATE part alignment. If set to zero, the feature is to be held exactly at nominal.

Refer to clauses 5.3.6.7.3, 5.3.6.7.5, (Figure 10 — Datum targets on a car door) and (Figure 12 — LOCATE and MATDEF on automotive glass).

The MATDEF statement is passed to the output file when executed.

6.125 MEAS

Function: Causes the DME to measure a feature.

Input Formats:

can be: **MEAS/var_1,F(lname1),n**

Output Formats:

can be: **None**

Where:

var_1 can be: **ARC**
or: **CIRCLE**
or: **CONE**
or: **CONRADSEGMNT**
or: **CPARLN**
or: **CYLNDR**
or: **CYLRADSEGMNT**
or: **EDGEPT**
or: **ELLIPS**
or: **ELONGCYL**
or: **GCURVE**
or: **GSURF**
or: **LINE**
or: **OBJECT**
or: **PARPLN**
or: **PLANE**
or: **POINT var_2**
or: **RCTNGL**
or: **REVSURF**
or: **SPHERE**
or: **SPHRADSEGMNT**
or: **SYMPLN**
or: **TORUS**
or: **TORRADSEGMNT**

var_2 can be: **,COMP,var_3**
or: **does not exist**

var_3 can be: **AXDIR**
or: **DME**
or: **FEAT,var_4**
or: **POL**
or: **SPH**
or: **VEC,i,j,k**

var_4 can be: **F(lname2)**
or: **FA(lname2)**
or: **G(lname3)**

ARC signifies that a circular arc is to be measured.

AXDIR signifies that probe compensation is to be applied along the nearest axis of the current coordinate system.

CIRCLE signifies that a circle is to be measured.

COMP signifies that probe compensation is to be applied as specified by var_3.

CONE signifies that a cone is to be measured.

CONRADSEGMNT signifies that a conical radial segment is to be measured.

CPARLN signifies that a centered parallel line is to be measured.

CYLNDR	signifies that a cylinder is to be measured.
CYLRADSEGMNT	signifies that a cylindrical radial segment is to be measured.
DME	signifies that probe compensation is to be applied using the DME's system algorihm.
EDGEPT	signifies that an edge point feature is to be measured.
ELLIPS	signifies that an ellipse is to be measured.
ELONGCYL	signifies that an elongated cylinder is to be measured.
F(lname1)	is the label of the feature nominal to be measured.
F(lname2) **FA(lname2)**	is the label of a feature which will be used to define the probe compensation direction. The probe compensation direction will be derived from the feature's nearest point to the raw data point.
FEAT	signifies that probe compensation is to be applied in a radial direction, in relation to the centroid of the following feature.
G(lname3)	is the label of a previously defined geometry data which will be used to define the probe compensation direction. The probe compensation direction will be derived from the previously defined geometry data's nearest point to the raw data point.
GCURVE	signifies that a generic curve is to be measured.
GSURF	signifies that a generic surface is to be measured.
i,j,k	is the compensation direction to be used. The direction is pointing away from material.
LINE	signifies that a line is to be measured.
n	is a positive integer value (except in the special case of point measurement where n may be zero) that represents the number of measurements to be taken in the measurement of the feature. It is only used when manual mode or automatic mode are in effect, and when scanning is not in effect. When scanning has been activated, the scanning parameters and DME's internal algorithms determine the number of measurement points.
OBJECT	signifies that an object is being measured.
PARPLN	signifies that a feature of linear size (slot, block) is to be measured.
PLANE	signifies that a plane is to be measured.
POINT	signifies that a point is to be measured.
POL	signifies that probe compensation is to be applied in a radial direction, in relation to the origin of the current coordinate system, in the current working plane.
RCTNGL	signifies that a right rectangular prism is to be measured.
REVSURF	signifies that a surface-of-revolution is to be measured.
SPH	signifies that probe compensation is to be applied in a radial direction, in relation to the origin of the current coordinate system.
SPHERE	signifies that a sphere is to be measured.
SPHRADSEGMNT	signifies that a spherical radial segment is to be measured.
SYMPLN	signifies that a feature of linear size (slot, block) is to be measured.
TORRADSEGMNT	signifies that a toroidal radial segment is to be measured.
TORUS	signifies that a torus is to be measured.
VEC	signifies that probe compensation is to be applied along the following vector.

In the case of measuring a point: when n is 1, the point defined by F(lname) is measured. When n is 0, the current sensor position is recorded as the point and assigned to the F(lname) name in the 'MEAS/POINT,F(lname),0' statement.

The DME will measure the feature with its own internal algorithm when automatic mode is active. When the DME lacks this capability, it will default to the next program mode defined by the MODE statement.

When programmed mode is active, the DME will follow the given PAMEAS, PTMEAS and GOTO statements to measure the feature.

If the feature type is not recognized by the DME, and if a measurement sequence is programmed, the DME will follow the measurement sequence and output raw data for the measurement result.

The minimum number of points required to measure each feature is:

ARC..... 3	CPARLN,OPEN..... 3	GSURF 3	REVSURF 6
CIRCLE 3	CYLRADSEGMNT... 6	LINE 2	SPHERE 4
CONE..... 6	EDGEPT.... 1 (exactly)	PARPLN..... 4	SPHRADSEGMNT... 4
CONRADSEGMNT.. 6	ELLIPS 5	PLANE..... 3	SYMPLN 6
CPARLN,FLAT..... 5	ELONGCYL 9	POINT.....1 (exactly)	TORUS 9
CPARLN,ROUND.... 5	GCURVE 2	RCTNGL 9	TORRADSEGMNT... 9

A minimum number of nine points are required to measure a right rectangular prism, RCTNGL: three points for the first plane (primary), two for the second (secondary) and one point for each plane thereafter. Refer to (Figure B.8 — Constructed rectangle).

6.126 MFGDEV

Function: Defines a manufacturing device, and assigns a label to it.

Input Formats:

can be: **MD(lname)=MFGDEV/'text'**

Output Formats:

can be: **MD(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the manufacturing device.

'text' is a text string, enclosed with apostrophes, that identifies the manufacturing device.

The MFGDEV statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined manufacturing device label name.

Note: This statement is associated with the CUTCOM and TOOLDF statements for applications to adjust the manufacturing process based on inspection results.

6.127 MODE

Function: Sets the mode in which the DME will execute the program.

Input Formats:

can be: **MODE/var_1**

Output Formats:

can be: **MODE = AUTO**
or: **MODE = PROG**
or: **MODE = MAN**

Where:

var_1 can be: **AUTO,var_2**
or: **PROG,MAN**
or: **MAN**

var_2 can be: **PROG,MAN**
or: **MAN**

AUTO signifies the DME will execute GOTARG and MEAS statements ignoring the given intermediate moves and measurements, and will use its own algorithms in their place.

MAN signifies the DME will be driven manually for the measurements and moves.

PROG signifies the DME will execute GOTARG and MEAS statements using the given intermediate moves and measurements.

The DME will execute the GOTARG and MEAS statements in the mode given by the first minor word. When the DME is unable to execute in this mode, it will default to the next minor word. When it is unable to execute in this mode, it will default to the last minor word. The decision to default to the next level is made by the DME. If the DME defaults to manual mode, a message will be output to the operator.

The DME tests each MEAS and GOTARG statement independently; if MODE/AUTO,PROG,MAN has been issued the DME can perform some MEAS and GOTARG statements in automatic mode, some in programmed mode, and others in manual mode.

The MODE statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the MODE statement.

6.128 OBTAIN

Function: Sets a variable equal to any parameter of a definition.

Input Formats:

can be: **varname=OBTAIN/var_1,n**

Output Formats:

can be: **None**

Where:

var_1	*can be:*	**CC(lname1)**	*or:*	**PN(lname1)**
	or:	**CI(lname1)**	*or:*	**PR(lname1)**
	or:	**CR(lname1)**	*or:*	**PS(lname1)**
	or:	**CS(lname1)**	*or:*	**PV(lname1)**
	or:	**D(lname1)**	*or:*	**Q(lname1)**
	or:	**DA(lname1)**	*or:*	**R(lname1)**
	or:	**DI(lname1)**	*or:*	**RM(lname1)**
	or:	**DID(lname1)**	*or:*	**RT(lname1)**
	or:	**DS(lname1)**	*or:*	**S(lname1)**
	or:	**DV(lname1)**	*or:*	**SA(lname1)**
	or:	**F(lname1)**	*or:*	**SG(lname1)**
	or:	**FA(lname1)**	*or:*	**SGS(lname1)**
	or:	**F(lname1)[p]**	*or:*	**SRA(lname1)**
	or:	**FA(lname1)[p]**	*or:*	**SS(lname1)**
	or:	**FI(lname1)**	*or:*	**SW(lname1)**
	or:	**FS(lname1)**	*or:*	**SX(lname1)**
	or:	**G(lname1)**	*or:*	**T(lname1)**
	or:	**GSA(lname1)**	*or:*	**TA(lname1)**
	or:	**KC(lname1)**	*or:*	**TH(lname1)**
	or:	**LI(lname1)**	*or:*	**TL(lname1)**
	or:	**MA(lname1)**	*or:*	**V(lname1)**
	or:	**MD(lname1)**	*or:*	**VA(lname1)**
	or:	**OP(lname1)**	*or:*	**VF(lname1)**
	or:	**P(lname1)**	*or:*	**VL(lname1)**
	or:	**PC(lname1)**	*or:*	**VW(lname1)**
	or:	**PL(lname1)**		

varname is the name of a previously declared variable to which the obtained value is assigned.

n is a positive integer that represents the ordinal value of the parameter to be retrieved from the definition list. The first argument after the '/' is ordinal value 1.

p is a positive integer that represents the subscript number identifying a specific individual point data from the programmed PTMEAS statements, within the MEAS...ENDMES block used to measure F(lname) as described in clause 5.3.2.6.

The type of variable must be compatible with the type of parameter. For the cases where the parameter is a minor word or a DMS angle then a variable of type CHAR is required. The value will be truncated if it exceeds the number of characters in the CHAR variable declaration.

Note: This statement allows any parameter following the '/' delimiter from any DMIS statement having a label to be obtained, and assigned to a variable name. Where tolerances apply, OBTAIN will retrieve the tolerance statement(s) in the last EVAL and/or OUTPUT statements, for that label name of TA label type.

For example:

```
EVAL/FA(lname1),T(lname1)
var=OBTAIN/TA(lname1),3
```

6.129 OPEN

Function: Opens a previously defined system device or file and establishes the connection's input/output attributes.

Input Formats:

can be: **OPEN/DID(lname1),var_1**

Output Formats:

can be: **OPEN/DID(lname1),var_1**

Where:

var_1 can be: **DIRECT,var_2**
or: **FDATA,var_3,OUTPUT var_4**
or: **CAD var_5**
or: **SNS var_6**
or: **PCS var_6**
or: **FEATUR var_6**
or: **RTAB var_6**
or: **DML,'version_no',var_8**

var_2 can be: **INPUT**
or: **OUTPUT var_4**

var_3 can be: **V(lname2)**
or: **DMIS**

var_4 can be: **,APPEND**
or: **,OVERWR**
or: **does not exist**

var_5 can be: **,STEP**
or: **,IGES**
or: **,VENDOR,'text'**
or: **does not exist**

var_6 can be: **,SNS var_7**
or: **,PCS var_7**
or: **,FEATUR var_7**
or: **,RTAB var_7**
or: **does not exist**

var_7 can be: **var_6**

var_8 can be: **INPUT**
or: **OUTPUT,var_9**

var_9 can be: **PTDATA,ON**
or: **PTDATA,OFF**

APPEND signifies that any information written to the specified output file is to be appended to the existing data in the file.

CAD signifies storage that CAD data will be saved to and recalled from.

DID(lname1) is the device identification label assigned by the DEVICE statement to a system device to be opened.

DIRECT signifies that the specified device is to be accessed only by the READ and WRITE statements.

DMIS signifies that the output will be in DMIS format.

DML signifies that data will be in DML format with respect to the current coordinate system.

FDATA signifies that a specified format for the data will be used.

FEATUR	signifies storage that feature information will be saved to and recalled from.
IGES	signifies that the CAD data is in IGES format.
INPUT	signifies that the specified file is to be open for input
OFF	signifies that the individual point data will not be output.
ON	signifies that the individual point data will be output.
OUTPUT	signifies that the specified file is to be open for output
OVERWR	signifies that any information written to the specified output file is to be written over the existing data in the file.
PCS	signifies storage that part coordinate system information will be saved to and recalled from.
PTDATA	signifies output of individual point data (from the PTMEAS statements (if any) specified in the feature measurement).
RTAB	signifies storage that rotary table information will be saved to and recalled from.
SNS	signifies storage that sensor information will be saved to and recalled from.
STEP	signifies that the CAD data is in STEP format.
'text'	is a series of printable UTF8 characters, enclosed in apostrophes. Description of the CAD system.
V(lname2)	is the label assigned to vendor specific output format, as specified by the VFORM statement.
VENDOR	signifies that the CAD data is in vendor specific format.
'version_no'	is the version number of the DML specification, enclosed with apostrophes, to be used to store information.

APPEND and OVERWR only apply to storage devices defined as STOR or INCR in the DEVICE statement.

Reading from an output device or writing to an input device generates an error.

There is no interaction between DISPLY devices and DID devices.

When a device is opened for FDATA output using the DMIS parameter, output will go to that device in DMIS format. When a device is opened for FDATA output using the V(lname) parameter, output will go to that device in vendor format.

If a device is opened for FDATA output in a carriage specific section, output for only that carriage will be output to the device. If the device is opened for FDATA output in a carriage common section, all output will go to that device. A device opened for FDATA output within a carriage specific section, must have been defined in a carriage specific section for that same carriage. A device opened for FDATA output in a carriage common section must have been defined in a carriage common section. For the sequence of output, refer to clause 5.4.5.7.

The OPEN statement is passed to the output file when executed.

6.130 OPERID

Function: Defines the identification of the operator running the DME, and assigns a label to it.

Input Formats:

can be: **OP(lname)=OPERID/'text'**

Output Formats:

can be: **OP(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the operator identification.

'text' is a text string, enclosed with apostrophes, that identifies the operator.

The OPERID statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined operator identification label name.

6.131 OUTPUT

Function: Causes evaluation and output of actuals or nominals, or a previously defined report.

Input Formats:

can be: **OUTPUT/var_1**

Output Formats:

can be: **OUTPUT/var_1**

Where:

var_1 can be: **FA(lname1) var_2 var_3**
or: **FA(lname2),var_4,TA(lname3) var_3**
or: **var_5,FA(lname4),TA(lname3) var_3**
or: **F(lname5) var_6 var_3**
or: **F(lname6),F(lname7),T(lname8) var_3**
or: **KC(lname9)**
or: **KCA(lname9)**
or: **R(lname10)**
or: **S(lname11)**
or: **SA(lname12) var_7**
or: **SE(lname13)**
or: **ST(lname14)**
or: **ST(lname14),SE(lname13),T(lname15)**
or: **ST(lname14),SE(lname13),TA(lname16)**
or: **T(lname15)**
or: **TA(lname16)**

var_2 can be: **,TA(lname16) var_2**
or: **does not exist**

var_3 can be: **,R(lname10)**
or: **does not exist**

var_4 can be: **F(lname7)**
or: **FA(lname4)**
or: **DAT(x)**

var_5 can be: **F(lname6)**
or: **DAT(x)**

var_6 can be: **,T(lname17) var_6**
or: **does not exist**

var_7 can be: **var_8 var_9**
or: **,CURENT**
or: **does not exist**

var_8 can be: **,'desc'**
or: **,tipnum**
or: **does not exist**

var_9 can be: **,SW(lname18),'anglename',ang var_10 var_9**
or: **does not exist**

var_10 can be: **,'anglename',ang var_10**
or: **does not exist**

ang is the angle of the wrist axis identified by 'anglename'.

'anglename' is a text string, enclosed with apostrophes, identifying the wrist angle for which a rotation angle applies.

CURENT signifies that the current selected sensor calibration information is to be output.

DAT(x) represents the datum associated with a measured feature in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

'desc' is a text string, enclosed with apostrophes, which is the tip description as defined in the SENSOR/MLTPRB statement.

F(lname5) is the label of the feature nominal for which output data will be output. If point buffering was enabled, with a PTBUFF/ON statement when this feature was measured, then optionally the F(lname)[n,m] format may be used to cause the output of PTDATA statements for individual nominal point data. If the F(lname)[n,m] format is used then no tolerance output will occur. Refer to clause 5.3.2.6 and the FEAT definitions in section 6.

F(lname6) is the label of the first feature nominal to be associated with a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB. When used in the OUTPUT/F(lname5), FA(lname2), TA(lname16) form, F(lname5) represents the feature nominal associated with a measured feature FA(lname2) in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

F(lname7) is the label of the second feature nominal to be associated with a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB. When used in the OUTPUT/FA(lname1), F(lname6), TA(lname16) form, F(lname6) represents the feature nominal associated with a measured feature FA(lname1) in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

FA(lname1) is the label of the measured feature for which output data will be output. If point buffering was enabled, with a PTBUFF/ON statement, when this feature was measured, then optionally the FA(lname)[n,m] format may be used to cause the output of PTDATA statements for individual measured point data. If the FA(lname)[n,m] format is used then no tolerance evaluation or tolerance output will occur. Refer to clause 5.3.2.6 and the FEAT/... statement definitions in section 6.

FA(lname2) is the label of the first measured feature to be associated with a relationship tolerance, for example TOL/ANGLB or TOL/DISTB. When used in the OUTPUT/FA(lname1), F(lname6), TA(lname16) form, FA(lname1) represents the measured feature associated with a feature nominal F(lname6) in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB. When used in the OUTPUT/FA(lname1), DAT(x), TA(lname16) form, FA(lname1) represents the feature nominal associated with a datum DAT(x) in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

FA(lname4) is the label of the second measured feature to be associated with a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB. When used in the OUTPUT/F(lname5), FA(lname2), TA(lname16) form, FA(lname2) represents the measured feature associated with a feature nominal F(lname5) in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB. When used in the OUTPUT/DAT(x), FA(lname2), TA(lname16) form, FA(lname2) represents the feature nominal associated with a datum DAT(x) in a relationship tolerance, for example, TOL/ANGLB or TOL/DISTB.

KC(lname9)
KCA(lname9) is the label of the key characteristic label defined in the KEYCHAR statement that has associated feature(s) and tolerance(s), and may also have a key characteristic designation of CRITICAL, MAJOR, or MINOR.

R(lname10) is the label of the report label identifying additional report information required, as defined in the REPORT statement.

S(lname11) is the label of the sensor nominal to output. Refer to the SNSDEF statement.

SA(lname12) is the label of the sensor actual to output. Refer to the SNSDEF statement.

SE(lname13) is the label of a previously constructed gauge.

ST(lname14) is the label of a previously constructed part.

SW(lname18) is label of the wrist for which rotation angles apply.

T(lname15) is the label of a nominal tolerance.

T(lname17) is the label of the nominal tolerance associated with the feature nominal, F(lname).

T(lname8) is the label of the nominal relationship tolerance, for example, TOL/ANGLB or TOL/DISTB, associated with the feature nominals.

TA(lname16) is the label of the actual tolerance.

TA(lname3) is the label of the actual relationship tolerance, for example, TOL/ANGLB or TOL/DISTB, associated with the measured features.

tipnum is a positive integer that represents the tip number as defined in the SENSOR/MLTPRB statement.

Tolerances are generic, in that they have no pointers to features. Association between features and tolerances is provided for with the KEYCHAR, OUTPUT and EVAL statements. One or several tolerances can be associated with a feature. Two features are required to be associated with the angle between, TOL/ANGLB, and the distance between, TOL/DISTB, tolerances.

When a KC(lname) is specified, the execution of the OUTPUT statement results in the key characteristic and the feature nominal(s) and tolerance nominals(s) specified in the KEYCHAR statement being passed to the output file. When a KCA(lname) is specified, the execution of the OUTPUT statement results in evaluating the feature(s) and tolerance(s) specified in the KEYCHAR statement and the passing of the key characteristic and the feature actual(s) and tolerance actual (s) specified in the KEYCHAR statement to the output file.

Because several features are required to be associated with the composite TOL/COMPOS, tolerance for pattern positioning, the features in the pattern definition must all be previously measured before executing the OUTPUT statement with the FEAT/PATERN, FA(lname), and the TOL/COMPOS, TA(lname).

Additional report information can be requested by specifying the appropriate R(lname10).

The DME will output results in the order in which the OUTPUT statements appear in the DMIS program.

The OUTPUT statement is passed to the output file when executed and precedes all other statements generated by the execution of the OUTPUT statement. Furthermore, the statements generated by the OUTPUT statement appear in the output file in the same order in which they are requested in the OUTPUT statement.

6.132 PAMEAS

Function: Causes path directed scanning measurement to be performed.

Input Formats:

can be: **PAMEAS/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **var_2**
or: **var_3**

var_2 can be: **var_10, var_4 var_5 var_6 var_7 var_8**
or: **var_10, var_4 var_5 var_6 var_7 var_8, var_2**

var_3 can be: **var_4 var_5 var_6 var_7 var_8**
or: **var_4 var_5 var_6 var_7 var_8, var_3**

var_4 can be: **P(lname)**

var_5 can be: **,i,j,k**
or: **does not exist**

var_6 can be: **,SELFCENTER,is,js,ks**
or: **,SELFCENTER,is,js,ks var_9**
or: **does not exist**

var_7 can be: **,REMOVE,COUNT,nb_pt1,nb_pt2**
or: **,REMOVE,DIST,dist1,dist2**
or: **,REMOVE,ALL**
or: **does not exist**

var_8 can be: **,ROTARY,RT(lname2)**
or: **does not exist**

var_9 can be: **,FORCE,force1**
or: **,DEFLECTION,deflection1**

var_10 can be: **DISTANCE,dist3,SCNVEL,var_11,PITCH,pitch1**
or: **DISTANCE,dist3,SCNVEL,var_11**
or: **DISTANCE,dist3,PITCH,pitch1**
or: **SCNVEL,var_11,PITCH,pitch1**
or: **DISTANCE,dist3**
or: **SCNVEL,var_11**
or: **PITCH,pitch1**
or: **NODATA**

var_11 can be: **MPM,n**
or: **MMPS,n**
or: **IPM,n**
or: **IPS,n**

ALL indicates that all data is to be removed from the scan.

COUNT indicates that point removal is based on numbers.

DEFLECTION signifies that the probes deflection is to be used.

deflection1 is the probe's deflection.

DIST indicates that point removal is based on distance.

dist1 is the distance from the first scan point in which points will be removed, this value may be zero.

dist2	is the distance from the last scan point in which points will be removed, this value may be zero.
dist3	is the sampling distance for the scan segment as specified by the UNITS statement.
DISTANCE	signifies that the following parameter represents the sampling distance to be used for the scan segment.
FORCE	signifies that the probing force is used.
force1	is the probing force in Newtons.
i,j,k	is the direction vector to use for approaching the starting point of the path.
IPM	signifies that the value of 'n' is inches per minute.
IPS	signifies that the value of 'n' is inches per second.
is,js,ks	is the vector used to define the probing force. This vector points away from the surface.
MMPS	signifies that the value of 'n' is millimeters per second.
MPM	signifies that the value of 'n' is meters per minute.
n	is a positive real number representing the velocity value.
nb_pt1	is a non negative integer representing the number of points to be removed at the beginning of the scan path, this value may be zero.
nb_pt2	is a non negative integer representing the number of points to be removed at the end of the scan path, this value may be zero.
NODATA	signifies no data will be returned from the following segment.
P(lname)	is the label of a previously defined scan path.
PITCH	signifies that the following parameter represents the distance along the center line between two successive crossovers of the center line of the scan segment.
pitch1	is the distance along the center line between two successive crossovers of the center line of the scan segment, as specified by the UNITS statement.
REMOVE	signifies that points are to be removed from the scan.
ROTARY	signifies that rotary table motion will occur during measurement with total updating of the current part coordinate system. The path definition will define the rotational parameters of the rotary table for direction of rotation and amount of rotation.
RT(lname2)	is the label of the previously defined rotary table that is to be used.
SCNVEL	signifies that the following parameters will define the speed to be used for the following scan segment.
SELFCENTER	signifies that scanning will be performed in a way that the sensor contacts the part in at least two points in a groove.

In a case where the PAMEAS statement and the defined path that is referenced are in conflict, this results in an error condition.

Where the PAMEAS statement contains more than one P(lname), the scan segments are contiguous, with a single approach move at the start and a single retract move at the end of the PAMEAS using the current SNSET/APPRCH and RETRCT values.

Where a measurement block contains multiple PAMEAS statements, the current SNSET/APPRCH and RETRCT values are to be applied at the beginning and end of each PAMEAS statement.

When a SCNVEL or DISTANCE parameter is specified for a scan segment, the value replaces the current value set by a FEDRAT/SCNVEL or SCNSET/DIST statement respectively.

6.133 PARTID

Function: Defines the identification of the part to be inspected, and assigns a label to it.

Input Formats:

can be: **PN(lname)=PARTID/'text'**

Output Formats:

can be: **PN(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the part identification.

'text' is a text string, enclosed with apostrophes, that identifies the part.

The PARTID statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined part identification label name.

6.134 PARTRV

Function: Defines the identification of the part's revision level, and assigns a label to it.

Input Formats:

can be: **PR(lname)=PARTRV/'text'**

Output Formats:

can be: **PR(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the parts revision level.

'text' is a text string, enclosed with apostrophes, that identifies the part's revision level.

The PARTRV statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined part revision level label name.

6.135 PARTSN

Function: Defines the identification of the part serial number, and assigns a label to it.

Input Formats:

can be: **PS(lname)=PARTSN/'text'**

Output Formats:

can be: **PS(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the part serial number.

'text' is a text string, enclosed with apostrophes, that identifies the part serial number.

The PARTSN statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined part serial number label name.

6.136 PATH

Function: Defines a path that will direct scanning measurement, and assigns a label to it.

Input Formats:

can be: **P(lname)=PATH/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **POINT,var_2,ip,jp,kp**
or: **ARC,var_2,ia,ja,ka,rad,ang1,ang2 var_3**
or: **UNKNOWN,xs,ys,zs,xd,yd,zd,xe,ye,ze var_4**
or: **LINE,BND,var_5,VEC,il,jl,kl**
or: **CURVE,var_6**
or: **HELICAL,var_2,iax,jax,kax,hel_rad,hel_ang1,hel_ang2 var_7,pitch var_8**
or: **SURFACE,var_9**

var_2 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_3 can be: **,is,js,ks**
or: **does not exist**

var_4 can be: **,io,jo,ko**
or: **does not exist**

var_5 can be: **CART,START,e1x,e1y,e1z var_10,END,e2x,e2y,e2z var_10**
or: **POL,e1r,e1a,e1h,e2r,e2a,e2h**

var_6 can be: **x1,y1,z1,i,j,k,x2,y2,z2,i,j,k var_11**
or: **PTDATA,x1,y1,z1,i,j,k var_12 var_13**
or: **PTDATA,x1,y1,z1,i,j,k,PCS,angz5,angy4,angz6 var_12 var_13**
or: **PTDATA,x1,y1,z1,i,j,k,HEADCS,angz7,angy5 var_14 var_15**
or: **PTDATA,x1,y1,z1,i,j,k,HEADCS,angz7,angy5,angz8 var_14 var_15**

var_7 can be: **,ih,jh,kh**
or: **does not exist**

var_8 can be: **,ang3**
or: **does not exist**

var_9 can be: **PTDATA,x1s,y1s,z1s,i1s,j1s,k1s,width,height var_16 var_17**
or: **PTDATA,x1s,y1s,z1s,i1s,j1s,k1s,width,height,PCS,angz5,angy4 $**
,angz6 var_16 var_17
or: **PTDATA,x1s,y1s,z1s,i1s,j1s,k1s,width,height var_18 var_19**
or: **PTDATA,x1s,y1s,z1s,i1s,j1s,k1s,width,height,HEADCS,angz7,angy5 $**
var_18 var_19
or: **PTDATA,x1s,y1s,z1s,i1s,j1s,k1s,width,height,HEADCS,angz7,angy5 $**
,angz8 var_18 var_19

var_10 can be: **,HEADCS,angz7,angy5**
or: **,HEADCS,angz7,angy5,angz8**
or: **,PCS,angz5,angy4,angz6**
or: **does not exist**

var_11 can be: **,xn,yn,zn,i,j,k var_11**
or: **does not exist**

var_12 can be: **,PTDATA,x1,y1,z1,i,j,k**
or: **,PTDATA,x1,y1,z1,i,j,k,PCS,angz5,angy4,angz6**

var_13 can be: **var_12 var_13**
or: **does not exist**

var_14 can be: **,PTDATA,x1,y1,z1,i,j,k**
or: **,PTDATA,x1,y1,z1,i,j,k,HEADCS,angz7,angy5**
or: **,PTDATA,x1,y1,z1,i,j,k,HEADCS,angz7,angy5,angz8**

var_15 can be: **var_14 var_15**
or: **does not exist**

Var_16 can be: **,PTDATA,x1s,y1s,z1s,i1s,j1s,k1s,width,height**
or: **,PTDATA,x1s,y1s,z1s,i1s,j1s,k1s,width,height,PCS,angz5,angy4,angz6**

var_17 can be: **var_16 var_17**
or: **does not exist**

var_18 can be: **,PTDATA,x1s,y1s,z1s,i1s,j1s,k1s,width,height**
or: **,PTDATA,x1s,y1s,z1s,i1s,j1s,k1s,width,height,HEADCS,angz7,angy5**
or: **,PTDATA,x1s,y1s,z1s,i1s,j1s,k1s,width,height,HEADCS,angz7,angy5 $**
,angz8

var_19 can be: **var_18 var_19**
or: **does not exist**

ang1 is the start angle and is determined by either the major axis of the current working plane or by the vector defined by the presence of var_3. The angle is expressed in current system units.

ang2 is the included angle of the circular arc with respect to ang1 and can be more than 360 degrees. The angle is expressed in current system units.

ang3 is a positive real number representing the included angle of the cone. The angle is expressed in current system units.

ARC signifies that a circular path, including circular arc or over-scan is defined. The scan is complete after the scan has completed the specified number of degrees.

BND signifies that a bounded line is to be defined.

CART signifies that Cartesian coordinates are to follow.

CURVE signifies that a 3D curve definition will follow.

The stop condition for the scan is specified by the SCNSET statement.

END signifies that the coordinates of the ending point follow.

e1r,e1a,e1h are the polar coordinates of the two endpoints of the line.

e2r,e2a,e2h

e1x,e1y,e1z are the Cartesian coordinates of the two endpoints of the line.

e2x,e2y,e2z

HEADCS signifies that the orientation of the probe at the specified position will be specified by two or three angles of rotation in probe head coordinates.

When specifying HEADCS, the number of angles of rotation specified will match the number of physical axes of the probe head. The angles are always specified as absolute values, local to the probe head coordinates.

height is the height of the scan at the cross-section.

hel_ang1 is an angle used in finding the starting point of the helix as described in Note 1.

hel_ang2 is the included angle of the helical path from the start point to the end point. If the included angle is positive, the helix is right-handed (i.e. as a point on the helix moves in the direction of the axis, it rotates counterclockwise, as viewed from farther up the axis). If the included angle is negative, the helix is left-handed (i.e., as the point moves in the direction of the axis, it rotates clockwise, as viewed from farther up the axis). The included angle must not be zero, but its absolute value can be more than 360 degrees.

hel_rad Is a positive real number representing the radius of the helix, at the center point.

HELICAL	signifies that a helical profile will be scanned.
	The starting point of the helix is determined as described in Note 1. The starting point is defined as a point on a plane perpendicular to the helix's axis, which passes through the helix center point, at the radius of the helix along the starting angle. The scan is complete after the scan has completed the specified number of degrees.
i,j,k	is the direction vector of the preceding coordinate.
ia,ja,ka	is the direction vector of the plane in which the circular arc lies.
iax,jax,kax	is the direction vector of the axis of the helix.
ih,jh,kh	is the starting direction vector of the helix in the plane defined by the center point and axis of the helix.
il,jl,kl	is the direction vector of the plane in which the line lies.
INNER	signifies that the inside of the circular arc is to be measured (that is, a fillet).
io,jo,ko	represent a vector defining the orientation plane of the unknown scan.
ip,jp,kp	is the direction vector of the point.
is,js,ks	is the starting direction vector within the plane of the circular arc.
i1s,j1s,k1s	is a vector normal to and pointing away from the surface in which the point lies.
LINE	signifies that a line definition will follow. The scan is complete after the scan has crossed the plane defined by the endpoint of the line and the direction of the line.
OUTER	signifies that outside of the circular arc is to be measured (that is, a round).
PCS	signifies the angles specifying the probe orientation at the specified position are in part coordinates.
	PCS signifies that the axis system is rotated relative to the current part coordinate system: the axis system is rotated around Z by the first angle, then around the newly created Y axis, and finally around the newly created Z (which is the ZYZ Euler angle convention).
	The resultant head orientation will be such that the active probe tip shaft (or the main direction of an optical sensor) is aligned with the Z axis pointing towards the sensor. The alignment of the tool with respect to the X and Y axes will be tool dependent, e.g. to align a laser plane.
pitch	is a positive real number representing the offset distance parallel or antiparallel to the axis of the helix per rotation.
POINT	signifies that a point feature is to be measured. The machine axis motion will only move along the i,j,k direction to maintain the sensor relationship to the part, either contact or non-contact.
POL	signifies that polar coordinates are to follow.
PTDATA	signifies that the following parameters are a position and a direction.
r,a,h	are the polar coordinates of;
	in the case of ARC, the coordinates of the center point of the circular arc.
	in the case of POINT, the coordinates of the point.
	in the case of HELICAL, the coordinates of the center point of the helix.
rad	is a positive real number representing the radius of the circular arc.
START	signifies that the coordinates of the starting point follow.
SURFACE	signifies that 3D surface information follows.
UNKNOWN	signifies that an unknown profile will be scanned. The stop condition for the scan is specified by the SCNSET statement.

VEC	signifies that a direction vector to use for rotation of the wrist is to follow.
width	is the width of the scan at the cross-section.
x,y,z	are the Cartesian coordinates of; in the case of ARC, the coordinates of the center point of the circular arc. in the case of POINT, the coordinates of the point. in the case of HELICAL, the coordinates of the center point of the helix.
x1,y1,z1 **x2,y2,z2** **xn,yn,zn**	signifies the following list of points must be interpolated by the DME into a curve passing through the points.
x1s,y1s,z1s **,i1s,j1s,k1s**	is the location and normal on the centre line of the scanned surface at which the width and height parameters are applied.
xs,ys,zs **xd,yd,zd** **xe,ye,ze**	signify starting point, direction point and end point to be used scanning an unknown profile.
angz5,angy4 **,angz6**	are the Euler angles of rotation which define the axis system of the probe at the specified position.
angz7,angy5	are the angles of rotation of a 2 axis probe at the specified position.
angz7,angy5, **angz8**	are the angles of rotation of a 3 axis probe at the specified position.

The defined path is the actual pattern of the data to be collected. All individual point information will lie on the defined path.

Note 1: The starting point of the helix is found as follows:

1) Let C (x,y,z) be the center of the helix. Let P be the plane defined by C and the axis of the helix (iax,jax,kax).

2) Construct a half line L1 starting at C and lying in plane P. If var_7 does not exist, L1 is the projection on P of the half line starting at C going in the direction of the major axis of the current working plane. If var_7 (ih,jh,kh) does exist, L1 is the projection on P of the half line starting at C going in the direction ih,jh,kh.

3) Construct a second half line L2 starting at C and lying in plane P. The direction of L2 is established by making the angle between L1 and L2 be hel_ang1 (positive counterclockwise, as viewed from above P).

4) The starting point of the helix is the point on L2 a distance hel_rad from C.

Refer to (Figure B.92 — A helical scan).

Note 2: if the orientation plane direction is not specified for an UNKNOWN scan the plane is determined by fitting through the start, direction and end points. If a plane cannot be determined from the three points (e.g. they are collinear) an error condition results.

Note 3: if the orientation plane direction is specified for an UNKNOWN scan and it does not match the plane is determined by fitting through the start, direction and end points, an error condition results.

6.137 PLANID

Function: Defines the identification of the inspection plan, and assigns a label to it.

Input Formats:

can be: **PL(lname)=PLANID/'text'**

Output Formats:

can be: **PL(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the inspection plan.

'text' is a text string, enclosed with apostrophes, that identifies the inspection plan.

The PLANID statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined inspection plan identification label name.

6.138 POP

Function: Restores a selected portion of the DMIS environment from an internal stack, and removes that information from the stack. The POP statement has no effect if the stack contains no items of the selected portion. Each configuration group (such as ALGOR or DME) is maintained on a separate stack, and popping one group has no effect on the stack for any other configuration group.

Input Formats:

can be: **POP/var_1 var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **ALGOR**
or: **DME**
or: **DATSET**
or: **REPORT**

var_2 can be: **,var_1 var_2**
or: **does not exist**

ALGOR signifies that the configuration established by the following commands is restored from the last push of this information.

GEOALG
PRCOMP
PTBUFF
TECOMP
WKPLAN

DATSET signifies that the coordinate system is restored from the last one pushed.

DME signifies that the configuration established by the following commands is restored from the last push of this information.

ACLRAT
BADTST
CRMODE
CROSCL
ERROR
FEDRAT
FINPOS
FROM
MODE
RAPID
SNSET

REPORT signifies that the configuration established by the following commands is restored from the last push of this information.

DECPL
UNITS
VFORM

Note: During the restoration of state information while executing the POP statement, if any data item from the stack is marked as un-initialized, no change is made to that item, it is left in the state that it was before the POP statement was executed.

6.139 PRCOMP

Function: Causes automatic probe compensation to be enabled or disabled.

Input Formats:

can be: **PRCOMP/var_1**

Output Formats:

can be: **PRCOMP/var_1**

Where:

var_1 can be: **ON**
or: **OFF**

OFF signifies that automatic probe compensation is disabled.

ON signifies that automatic probe compensation is enabled.

Raw point data is invoked when a feature is not supported by the DME.

When PRCOMP/ON is initiated, the i,j,k vector of each PTMEAS is required.

The PRCOMP statement is passed to the output file when executed.

6.140 PREVOP

Function: Defines the identification of the previous operation, and assigns a label to it.

Input Formats:

can be: **PV(lname)=PREVOP/'text'**

Output Formats:

can be: **PV(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the previous operation.

'text' is a text string, enclosed with apostrophes, that identifies the previous operation.

The PREVOP statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined previous operation identification label name.

Note: This statement is required to support the inspection of electrical components, but is not limited to applications in the electronics industry.

6.141 PROCID

Function: Defines the identification of the inspection procedure, and assigns a label to it.

Input Formats:

can be: **PC(lname)=PROCID/'text'**

Output Formats:

can be: **PC(lname)= 'text'**

Where:

lname is an alphanumeric label name assigned to the inspection procedure.

'text' is a text string, enclosed with apostrophes, that identifies the inspection procedure.

The PROCID statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined inspection procedure identification label name.

6.142 PROMPT

Function: To prompt the DME operator for one or more values to be assigned to high level variable(s).

Input Formats:

can be: **varname=PROMPT/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **'text' var_2**
or: **var_3**

var_2 can be: **,maxval var_5**
or: **does not exist**

var_3 can be: **BUTTON,'text',ret_val var_4**
or: **CHECK,'text',chk_var var_4**
or: **EDIT,edit_var var_2 var_4**
or: **GROUP,'text',group_var,'item','item' var_6 var_4**
or: **LIST,list_var,'item','item' var_6 var_4**
or: **PICTURE,'filename' var_7 var_4**
or: **PIXBTN,'filnam1' var_8,ret_val var_4**
or: **SOUND,'filename' var_4**
or: **TEXT,'text' var_4**
or: **TITLE,'text' var_4**

var_4 can be: **,var_3**
or: **does not exist**

var_5 can be: **,minval**
or: **does not exist**

var_6 can be: **,'item' var_9**
or: **does not exist**

var_7 can be: **,index**
or: **does not exist**

var_8 can be: **,'filnam2'**
or: **does not exist**

var_9 can be: **var_6**
or: **does not exist**

BUTTON signifies that a text pushbutton control will be displayed.

CHECK signifies that a single checkbox control will be displayed.

chk_var is the name for a high level BOOL DECL variable.

EDIT signifies that a single text edit control will be displayed.

edit_var is the name for a high level DECL variable or a QIS variable.

'filename' is the name of a data file of a supported type for the format, enclosed with apostrophes.

'filnam1' is the name of a file containing an image to be displayed on the button face, enclosed with apostrophes.

'filnam2' is the name of a file containing an image to display when the button is pressed, enclosed with apostrophes.

GROUP signifies a group of radio button controls will be displayed.

group_var is the name for a high level DECL variable or a QIS variable.

index	is a positive integer representing an offset that begins at one for multi-frame graphics
'item'	is a text string, enclosed with apostrophes, representing one of the items to display in the control.
LIST	signifies that selection list control will be displayed.
list_var	is the name for a high level DECL variable or a QIS variable.
maxval	is a real number representing the maximum string length if varname is CHAR, or maximum limit if varname is INTGR, LONG, REAL, or DOUBLE.
minval	is a real number representing the minimum limit if varname is INTGR, LONG, REAL, or DOUBLE.
PICTURE	signifies that a graphic image will be displayed.
PIXBTN	signifies that a graphic button control will be displayed.
ret_val	is the integer value passed to varname when the control is activated
SOUND	signifies that a sound file will be played.
TEXT	signifies that a static text string will be displayed.
'text'	is a text string to be associated with the control, consisting of printable UTF8 characters, enclosed with apostrophes.
TITLE	signifies that text for the title bar of the prompt window follows.
varname	for PROMPT/'text' format, is the name of the previously declared high level variable or array element, or a QIS variable, to receive the value of the edit field; for all other formats, is an integral type (INTGR or LONG), to receive the value of the button which dismisses the window.

The PROMPT statement operates in two modes.

If the PROMPT major word is not followed by a minor word, then a simple entry mechanism including the text prompt is presented to the operator by the DME. The appearance is DME dependent.

If the major word PROMPT/ is followed by one of the minor words in the definition of var_3, then the DME will construct a window containing the specified fields in the order in which they appear in the PROMPT statement. All items may appear anywhere in the PROMPT parameter list.

The window is dismissed when the operator presses one of the pushbuttons it contains, and the integer value of the button is placed in *varname*. The window must contain at least one pushbutton. If a pushbutton with a return value of zero is pressed, no variable other than *varname* will be populated with the dialog contents.

For BUTTON items, the DME will display a text pushbutton. *text* is a text string which visually identifies the pushbutton to the DME operator and appears on the face of the pushbutton. *ret_val* must be unique for each of the pushbuttons within a prompt. Pressing and releasing the pushbutton dismisses the prompt window, places the *ret_val* associated with the pressed pushbutton into *varname*, and causes the DME to populate all other variables referenced within the PROMPT statement.

For CHECK items, the DME will display a single checkbox. *text* is a text string which visually identifies the checkbox to the DME operator and appears immediately to the right of the checkbox. *chk_var* must be the name of a high level variable of type BOOL: the variable which this name references must exist, and will be populated with the selection state of the checkbox at the time that the window is dismissed.

For EDIT items, the DME will display a text edit field. *edit_var* must be the name of a high level variable of type INTGR, LONG, REAL, DOUBLE, or CHAR, or the name of a QIS variable: the variable which this name references must exist, and will be populated with the contents which the edit field contains at the time that the window is dismissed. If a numeric type, the optional *maxval* provides the highest numeric level, and the optional *minval* provides the lowest numeric value, to be accepted by the DME. The DME is responsible for testing these conditions, and will not dismiss the window until a proper value is entered. For CHAR or QIS types, the optional *maxval* is the maximum number of characters to accept from the operator, and the optional *minval* is an initial field value to be displayed.

Refer to clauses 5.2.1.2 and 5.2.2.7 for DECL and QIS variable definitions.

For GROUP items, the DME will display a group of radio controls with exclusive selection properties: exactly one item in the group must be selected at any time. The first item in the group will be selected when the prompt window is displayed. *text* is a text string which, if present, visually identifies the group to the DME operator and appears immediately prior to the first radio control. The value of *group_var* must be the name of a high level variable of type INTGR, LONG, or CHAR, or a QIS variable: the variable which this name references must exist, and will be populated with the selected item of the group at the time that the window is dismissed. If a numeric type, it will be populated with the one-based index of the selected item, based on ordering in the input statement; if a CHAR type, it will be populated with the item text of the selected item.

For LIST items, the DME will display either a collapsed (drop down) list box or an expanded list box. *list_var* must be the name of a high level variable of type INTGR, LONG, or CHAR, or a QIS variable: the variable which this name references must exist, and will be populated with the choice selected within the list at the time that the window is dismissed. If a numeric type, it will be populated with the one-based index within the list of the selected item; if a CHAR type, it will be populated with the item text of the selected item.

For PICTURE items, *filename* is the name of a graphics formatted data file. The types of supported graphics formats is DME specific. The optional *index* is a one-based offset into a multi-frame graphics file (such as an animated GIF file).

For PIXBTN items, *filnam1* is the name of a graphics formatted data file to display on the face of a picture pushbutton. The optional *filnam2* is the name of a graphics formatted data file to display when the button is depressed; if this value is not supplied, the DME will visually indicate the pressing of the button in a distinguishable manner utilizing the picture in *filnam1*. Pressing and releasing the button dismisses the prompt window, places the *ret_val* associated with the pressed button into *varname*, and causes the DME to populate all other variables referenced within the PROMPT statement.

For SOUND items, *filename* is the name of a sound formatted data file. The types of supported sound formats is DME specific.

For TEXT items, *text* is a text string which permits the DME to add descriptive comments or information to the window.

For TITLE item, *text* is the title to be displayed in the prompt window's title bar. Only one TITLE item may occur in a single PROMPT statement.

All QIS variables can be assigned the user's responses. When applying the PROMPT statement to any QIS variable, including one defined using QISDEF, the entered value will be assigned to the 'text' value of the QIS variable.

Refer to example A.25.1, A.25.2 and A.25.3.

6.143 PSTHRU

Function: Passes statements in the inspection program through to the output file without interpretation or execution.

Input Formats:

can be: **PSTHRU/var_1**

Output Formats:

can be: **PSTHRU/var_1**

Where:

var_1 can be: **COMAND,'command'**
or: **CONTIN**
or: **PAUSE**
or: **START**
or: **STOP**
or: **TRMATX,a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3**

a1,a2,a3, b1,b2,b3, c1,c2,c3, d1,d2,d3 satisfy the following transformation equations:

(a1)x + (b1)y + (c1)z + d1 = x'
(a2)x + (b2)y + (c2)z + d2 = y'
(a3)x + (b3)y + (c3)z + d3 = z'

Where: x', y', and z' are the coordinates in the current part coordinate system (after the transformation has been applied) of any point, and x, y, and z are the coordinates in the previous coordinate system of the same point.

COMAND signifies a command will be sent to the system receiving the DME output file.

'command' is the command, enclosed with apostrophes, to be sent to the receiving system.

CONTIN signifies that the system receiving the DME output file is to continue processing data. It has no effect on the control of the DME, except that the DME will pass the statement to the output file. The receiving system will ignore any data that are between the PSTHRU/PAUSE statement and the next occurrence of the PSTHRU/CONTIN statement.

PAUSE signifies that the system receiving the DME output file is to stop processing data. It has no effect on the control of the DME, except that the DME will pass the statement to the output file. The receiving system will ignore any data that are between the PSTHRU/PAUSE statement and the next occurrence of the PSTHRU/CONTIN statement.

START signifies the system receiving the DME output file is to start processing data. It has no effect on the control of the DME, except that the DME will pass the statement to the output file. The receiving system will ignore any data before the START statement is encountered in the DME output file.

STOP signifies the system receiving the DME output file is to stop processing data. It has no effect on the control of the DME, except that the DME will pass the statement to the output file. Everything following this statement will be ignored by the receiving system.

TRMATX signifies that the alignment and transformation information is being output.

The PSTHRU statement is passed to the output file when executed.

6.144 PTBUFF

Function: To turn on/off the saving of individual point information from the measurement points of higher level features (for example, cylinder, plane). The points captured can be accessed by using the higher level feature name and subscripted arrays as stated in clause 5.3.2.6.

Input Formats:

can be: **PTBUFF/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **ON**
or: **OFF**

OFF signifies that the capture of individual point information from higher level features is to be disabled.

ON signifies that the capture of individual point information from subsequent higher level features is to be enabled.

Regardless of the setting of probe compensation at measurement time, both compensated and uncompensated point data can be accessed depending on the setting of probe compensation at the time of output.

6.145 PTMEAS

Function: signifies that an automatic point measurement is to be performed.

Input Formats:

can be: **PTMEAS/var_1 var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **CART,x,y,z**
or: **POL,r,a,h**

var_2 can be: **,i,j,k**
or: **,i,j,k,var_3**
,var_3
does not exist

var_3 can be: **PCS,z1,y1,z2**
or: **PCS,z1,y1,z2,var_4**
or: **HEADCS,z3,y2**
or: **HEADCS,z3,y2,var_4**
or: **HEADCS,z3,y2,z4**
or: **HEADCS,z3,y2,z4,var_4**
or: **var_4**

var_4 can be: **HEADTOUCH**
or: **ALLAXESTOUCH**

ALLAXESTOUCH signifies the DME will take the touch using head rotation in conjunction with other DME axes.

CART signifies Cartesian coordinates are to follow.

HEADCS signifies that the orientation of the probe at the approach position will be specified by two or three angles of rotation in probe head coordinates.

When specifying HEADCS, the number of angles of rotation specified will match the number of physical axes of the probe head. The angles are always specified as absolute values, local to the probe head coordinates.

HEADTOUCH signifies the DME will take the touch as a head rotation only.

i,j,k is the direction vector pointing away from the surface of the feature used in making the measurement.

PCS signifies the angles specifying the probe orientation at the approach position are in part coordinates.

PCS signifies that the axis system is rotated relative to the current part coordinate system: the axis system is rotated around Z by the first angle, then around the newly created Y axis, and finally around the newly created Z (which is the ZYZ Euler angle convention).

The resultant head orientation will be such that the active probe tip shaft (or the main direction of an optical sensor) is aligned with the Z axis pointing towards the sensor. The alignment of the tool with respect to the X and Y axes will be tool dependent, e.g. to align a laser plane.

POL signifies that polar coordinates are to follow.

r,a,h are the nominal polar coordinates of the point to be measured.

x,y,z are the nominal Cartesian coordinates of the point to be measured.

z1,y1,z2 are the Euler angles of rotation which define the axis system of the probe at the approach position.

z3,y2 are the angles of rotation of a 2 axis probe at the approach position.

z3,y2,z4 are the angles of rotation of a 3 axis probe at the approach position.

When PRCOMP/ON is in effect, the direction vector must be specified in each PTMEAS statement.

The x,y,z point coordinates and the i,j,k vectors are given relative to the current coordinate system. The 'r' and 'a' values are given relative to the current working plane and coordinate system. The 'h' value is the perpendicular distance of the point in the working plane.

6.146 PUSH

Function: Saves a selected portion of the DMIS environment on an internal stack. Each configuration group (such as ALGOR or DME) is maintained on a separate stack, and pushing one group has no effect on the stack for any other configuration group.

Input Formats:

can be: **PUSH/var_1 var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **ALGOR**
or: **DME**
or: **DATSET**
or: **REPORT**

var_2 can be: **,var_1 var_2**
or: **does not exist**

ALGOR signifies that the current configuration established by the following commands is pushed:

GEOALG
PRCOMP
PTBUFF
TECOMP
WKPLAN

DATSET signifies that the current coordinate system is pushed.

DME signifies that the current configuration established by the following commands is pushed:

ACLRAT
BADTST
CRMODE
CROSCL
ERROR
FEDRAT
FINPOS
FROM
MODE
RAPID
SNSET

REPORT signifies that the current configuration established by the following commands is pushed:

DECPL
UNITS
VFORM

Information on the stack only exists while the current DMIS program is active.

Any implementation must support a stack depth of at least four (4).

If any state information is unknown and unavailable at the time the PUSH statement is executed, that fact is recorded with the pushed information.

6.147 QISDEF

Function: Defines a user specified QIS variable, and assigns a label to it.

Input Formats:

can be: **Q(lname)=QISDEF/'type' var_1**

Output Formats:

can be: **Q(lname)='type' var_1**

Where:

var_1 can be: **,'text'**
or: **does not exist**

lname is an alphanumeric label name assigned to the QIS variable.

'text' is a text string, enclosed with apostrophes, that identifies the type's value.

'type' is a string, enclosed with apostrophes, defining a user specific QIS entity.

The QISDEF statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined QIS variable definition label name.

Refer to clause 5.2.2.7 and example A.26.

6.148 RAPID

Function: Used to signify that the next move of the DME is to take place at the specified percentage of maximum speed. It is in effect for only one move.

Input Formats:

can be: **RAPID/speed**

Output Formats:

can be: **None**

Where:

speed is the percent of maximum machine speed as a fraction less than or equal to 1.0 and greater than or equal to 0.01, for example, 0.75 means 75%.

The RAPID statement only applies to the move statements GOTO, GOTARG, and GOHOME. The RAPID statement applies to the GOTARG...ENDGO block. All GOTO statements within the block will execute at the rapid speed.

6.149 READ

Function: Used to transfer data from a system device or file to the program.

Input Formats:

can be: **READ/DID(lname1),var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **variable:x:y var_2**
or: **variable:x var_2**
or: **variable var_2**

var_2 can be: **,var_1**
or: **does not exist**

DID(lname1) is the device identification label assigned by the DEVICE statement to a system device.

variable is the variable name.

x is a positive integer representing the total field width (including any negative sign).

y is a positive integer representing the number of digits to the right of the decimal point.

The variable name must be previously defined using the DECL statement. Input data fields in the file being read may be delimited by blanks or commas. Formatted files can be read by using the field width specified. If not specified the field width of the variable names are defined as follows:

Type	Field width	Format
INTGR	System default	var_n :x
LONG	System default	var_n :x
REAL	System default	var_n :x:y
DOUBLE	System default	var_n :x:y
BOOLEAN	System default	var_n
CHAR, n	Declared width	var_n :x

Refer to example A.27.

6.150 RECALL

Function: Enables data stored with the SAVE statement to be recalled.

Input Formats:

can be: **RECALL/var_1**

Output Formats:

can be: **RECALL/var_1**

or all: **RECALL/var_1 var_2**
DA(lname1)=RECALL/TRMATX,a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3

Where:

var_1 can be: **D(lname2) var_2**
or: **DA(lname1) var_2**
or: **S(lname3) var_2**
or: **SA(lname4) var_2**
or: **FA(lname5) var_2**
or: **RT(lname6) var_2**
or: **DML,DID(lname7),DA(lname8)**

var_2 can be: **,DID(lname7)**
or: **does not exist**

D(lname2) is the label of the part coordinate system that is to be recalled.

DA(lname1) is the label of the part coordinate system that is to be recalled.

DA(lname8) is the label of the part coordinate system to which the DML Common Space will be equated.

DID(lname7) is the device identification label, assigned by the DEVICE statement to a system device.

DML signifies the dimensional mark-up language will be recalled from the device specified.

FA(lname5) is the label of the feature actual that is to be recalled.

RT(lname6) is the label of the rotary table that is to be recalled.

S(lname3) is the label of the sensor that is to be recalled.

SA(lname4) is the label of the calibrated sensor that is to be recalled.

When the DID(lname7) is not specified, the information will be recalled from the DME's internal storage devices or media.

In the case of the part coordinate system, the RECALL statement reactivates the recalled datum set to become the current coordinate system.

In the case of the sensor, the RECALL statement recalls the previously saved calibration data for that sensor. It is not necessary to recall calibration data when utilizing different sensors providing the data resides in memory. It is acceptable to save calibration data to a mass storage device for subsequent recall, when the amount of memory is a concern.

A sensor may be recalled even if it has not been calibrated in the current part program. Sensors can be calibrated and saved in separate programs.

When applying the RECALL statement to a S(lname3) or a SA(lname4) that was created using the SNSDEF/BUILD statement, aligning a sensor to features or direction vectors is not possible when using wrists. To fully use the SNSLCT statement, the sensor must be defined prior to issuing SNSLCT statement. This also applies to multi-probe sensors.

When a datum set or part coordinate system is recalled, then the output format with the TRMATX is output.

The RECALL statement is passed to the output file when executed.

6.151 REFMNT

Function: Defines the relationship of a reference mounting point to the last sensor for a sensor mount component, and assigns a label to it. This function establishes a new sensor coordinate system for subsequent sensor components.

Input Formats:

can be: **RM(lname)=REFMNT/XVEC,xi,xj,xk,ZVEC,zi,zj,zk,MNTLEN,dx,dy,dz**

Output Formats:

can be: **None**

Where:

dx,dy,dz is the offset vector in the current sensor coordinates between the DME manufacturer's sensor reference point and the mount point on the last sensor component.

lname is an alphanumeric label name assigned to the reference mount point.

MNTLEN signifies the nominal distance in machine coordinates between the DME manufacturer's sensor reference point and the origin of the sensor coordinate system is to follow.

xi,xj,xk is the vector in the previous sensor coordinate system along which the reference mount coordinate system X axis is aligned.

XVEC signifies that the specification of the X axis of the reference mount coordinate system in the current sensor coordinate system is to follow.

zi,zj,zk is the vector in the current sensor coordinate system along which the sensor coordinate system Z axis is aligned.

ZVEC signifies that the specification of the Z axis of the reference mount coordinate system in the current sensor coordinate system is to follow.

6.152 REPORT

Function: Specifies additional information to be put in the DME output file, and assigns a label to it.

Input Formats:

can be: **R(lname)=REPORT/var_1 var_2 var_3**

Output Formats:

can be: **None**

Where:

var_1 can be:	**ALGOR**		*or:*	**OP(lname11)**
	or:	**CC(lname1)**	*or:*	**PC(lname12)**
	or:	**CI(lname2)**	*or:*	**PL(lname13)**
	or:	**CS(lname3)**	*or:*	**PN(lname14)**
	or:	**DATE**	*or:*	**PR(lname15)**
	or:	**DI(lname4)**	*or:*	**PS(lname16)**
	or:	**DS(lname5)**	*or:*	**PV(lname17)**
	or:	**DV(lname6)**	*or:*	**Q(lname18)**
	or:	**FI(lname7)**	*or:*	**TEMPC**
	or:	**FS(lname8)**	*or:*	**TEMPF**
	or:	**HUMID**	*or:*	**TEMPWC**
	or:	**LI(lname9)**	*or:*	**TEMPWF**
	or:	**MD(lname10)**	*or:*	**TIME**
	or:	**MODE**	*or:*	**TL(lname19)**
var_2 can be:	**,var_1 var_2**			
	or:	**does not exist**		
var_3 can be:	**,'text'**			
	or:	**does not exist**		

ALGOR signifies the DME will output the type of algorithm used in calculating the feature.

CC(lname1) is the label of the cutter compensation.

CI(lname2) is the label of the part holding clamp.

CS(lname3) is the label of the part holding clamp serial number.

DATE If used together with an F(lname) or FA(lname) in an OUTPUT statement, signifies the date at the end of the measurement of that feature; otherwise, signifies the current date.

DI(lname4) is the label of the DME.

DS(lname5) is the label of the DME software.

DV(lname6) is the label of the DME software version.

FI(lname7) is the label of the part holding fixture.

FS(lname8) is the label of the part holding fixture serial number.

HUMID If used together with an F(lname) or FA(lname) in an OUTPUT statement, signifies that the relative humidity at the end of the measurement of that feature; otherwise, signifies the current relative humidity.

lname is an alphanumeric label name assigned to the report.

LI(lname9) is the label of the part lot.

MD(lname10) is the label of the manufacturing device.

MODE signifies the DME will output the mode (AUTO, MAN, or PROG) in which the feature measurement was made.

OP(lname11) is the label of the DME operator.

PC(lname12) is the label of the inspection procedure.

PL(lname13) is the label of the inspection plan.

PN(lname14) is the label of the part.

PR(lname15) is the label of the part revision level.

PS(lname16) is the label of the part serial number.

PV(lname17) is the label of the previous operation.

Q(lname18) is the label of the QIS variable.

TEMPC If used together with an F(lname) or FA(lname) in an OUTPUT statement, signifies that the temperature, in degrees Celsius, at the end of the measurement of that feature; otherwise, signifies the current temperature, in degrees Celsius.

TEMPF If used together with an F(lname) or FA(lname) in an OUTPUT statement, signifies that the temperature, in degrees Fahrenheit, at the end of the measurement of that feature; otherwise, signifies the current temperature, in degrees Fahrenheit.

TEMPWC If used together with an F(lname) or FA(lname) in an OUTPUT statement, signifies that the temperature of the workpiece, in degrees Celsius, at the end of the measurement of that feature; otherwise, signifies the current temperature of the workpiece, in degrees Celsius.

TEMPWF If used together with an F(lname) or FA(lname) in an OUTPUT statement, signifies that the temperature of the workpiece, in degrees Fahrenheit, at the end of the measurement of that feature; otherwise, signifies the current temperature of the workpiece, in degrees Fahrenheit.

'text' is a text string, enclosed with apostrophes.

TIME If used together with an F(lname) or FA(lname) in an OUTPUT statement, signifies that the time at the end of the measurement of that feature; otherwise, signifies the current time.

TL(lname19) is the label of the tool.

The additional information requested in the R(lname) statement are those data which are relevant to the measurement but cannot be calculated from the feature and tolerance output data.

When DATE, TIME, HUMID, TEMPC, TEMPF, TEMPWC, or TEMPWF is specified during reporting of ANGLB and DISTB tolerances, these values will be output for both features. All items for the first feature will be output followed by all items of the second feature.

The REPORT statement does not trigger output by itself. The elements specified in the REPORT statement are output through the OUTPUT statement when the R(lname) is referenced.

The specific output format for those elements not having a defined output format in a DMIS statement definition is as follows:

ALGOR = 'text'	HUMID = real	TEMPF = real	TEMPWF = real
DATE = yyyy/mm/dd	TEMPC = real	TEMPWC = real	TIME = hh:mm:ss

6.153 RESUME

Function: Used to resume program execution after recovery from DME error.

Input Formats:

can be: **RESUME/var_1**

Output Formats:

can be: **RESUME/var_1**

Where:

var_1 can be: **(jumptarget)**
or: **CURENT**
or: **END**
or: **NEXT**
or: **START**
or: **STOP**

(jumptarget) signifies the label to which program execution is transferred.

CURENT signifies that program execution re-executes the current program statement, that is, the statement being executed when the error occurred.

END signifies that the program execution skips to the ENDMES statement of the current MEAS or RMEAS block. If program execution is not in a MEAS...ENDMES block or RMEAS…ENDMES block, execution is transferred to the next statement.

NEXT signifies the transfer of program execution to the next program statement.

START signifies the transfer program execution to the beginning of the MEAS...ENDMES block or RMEAS…ENDMES block in which the error occurred. If program execution is not in a MEAS...ENDMES block or RMEAS…ENDMES block, program execution re-executes the current program statement, that is, the statement being executed when the error occurred.

STOP signifies that the program execution be stopped.

On branching to the error recovery label, error handling by the ERROR statement is suspended until a RESUME statement.

The RESUME statement is passed to the output file when executed.

6.154 RMEAS (input format 1)

Function: Causes the DME to measure an ARC, CIRCLE, ELLIPS or OBJECT feature relative to another defined/constructed feature.

Input Formats:

can be: **RMEAS/var_1,F(lname1),n,var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **ARC**
or: **CIRCLE**
or: **ELLIPS**
or: **OBJECT**

var_2 can be: **FA(lname2)**
or: **VECBLD,r,n1**

ARC signifies that a circular arc is to be measured.

CIRCLE signifies that a circle is to be measured.

ELLIPS signifies that an ellipse is to be measured.

F(lname1) is the label of the previously defined feature nominal to be measured.

FA(lname2) (Adjusts Location and/or Orientation)

signifies that the nominal relationship between the reference feature actual, FA(lname2), and the feature nominal being measured, F(lname1), is to be maintained. FA(lname2) must be a previously measured or constructed plane-reducible or point-reducible feature.

Position and orientational deviation are applied when FA(lname2) is a plane reducible feature. Positional deviation only is applied when FA(lname2) is a point reducible feature.

Positional and orientational deviation are applied relative to FA(lname2).

The transformation between the original feature nominal being measured, F(lname1), and its adjusted nominal is used to transform all the subsequent motion and measurement associated with the feature.

Refer to (Figure B.48 — Relative measure / position & approach), (Figure B.49 — Relative measure / retaining relative position), (Figure B.50 — Relative measure / position), and (Figure B.51 — Relative measure / position & approach) for a pictorial representations of features being measured relative to a reference feature.

Refer to (Figure B.53 — RMEAS/CIRCLE with FA(lname2)) for an example of the FA(lname2) parameter when measuring a circle feature.

n is a positive integer representing the number of measurements to be taken in the measurement of the feature. It is only used when manual mode or automatic mode are in effect, and when scanning is not in effect. When scanning has been activated, the scanning parameters and DME's internal algorithms determine the number of measurement points.

n1 is a positive integer representing the number of sample points to be taken when VECBLD is used. The algorithm of the DME can determine the order and direction (clockwise or counter-clockwise around the feature to be measured) the sample points will be taken.

n1 cannot equal 2.

OBJECT signifies that an object is to be measured.

r is a positive real number representing the incremental value from the defining portion of the feature that the sample points will be taken. The DME will use its own algorithm to place the sample points within this "radius zone" around the feature

VECBLD signifies the sampling method to be used depending on the value of n1. These samples will be automatically taken by the DME to locate or locate and orient the feature prior to measuring it:

n1 = 1 (Adjusts Location)

A single point is automatically taken within a specified distance, 'r', from the defining portion of the feature. Positional deviation is applied along the nominal vector to establish the measurement plane. The depth specified in the SNSET statement is applied to this measurement plane. The nominal vector of the feature being measured is used as the approach vector.

n1 >= 3 (Adjusts Location and Orientation)

'n1' non-collinear points are automatically taken within a specified distance, 'r', from the defining portion of the feature. A measurement plane and normal vector are to be constructed from the sample points. Positional deviation along this normal vector and the orientational deviation are applied. The depth specified in the SNSET statement is applied to the constructed measurement plane. The normal vector is used as the approach vector for measuring the feature.

Refer to (Figure B.46 — Positional deviation) for an example of applying positional deviation when measuring a feature.
Refer to (Figure B.47 — Positional and orientational deviation) for an example applying positional deviation and orientational deviation when measuring a feature.

The RMEAS statement is usually followed by a series of statements, either PAMEAS, PTMEAS or GOTO statements. The RMEAS statement is terminated with an ENDMES statement. The DME will measure the feature with its own internal algorithm when automatic mode is active. When the DME lacks this capability, it will default to the next program level.

Refer to (Figure B.54 — RMEAS/CIRCLE with VECBLD,r,1 and with VECBLD,r,3) for an example of the VECBLD parameter when measuring a circle feature.

Refer to (Figure B.52 — RMEAS/ARC with and without a cylindrical probe) for an example of RMEAS/ARC with and without a cylindrical probe.

Refer to (Figure B.55 — RMEAS/CIRCLE with and without a cylindrical probe) for an example of RMEAS/CIRCLE with and without a cylindrical probe.

When programmed mode is active, the DME will follow the given PAMEAS, PTMEAS or GOTO statements to measure the feature.

If the DME does not recognize the specified feature type, and if a measurement sequence is programmed, the DME will follow the measurement sequence and output raw data for the measurement result.

A minimum of 3 points are required for measuring an ARC feature.
A minimum of 3 points are required for measuring a CIRCLE feature.
A minimum of 5 points are required for measuring an ELLIPS feature.

6.155 RMEAS (input format 2)

Function: Causes the DME to measure a CONE, CYLNDR, PARPLN, RCTNGL or SYMPLN feature relative to another defined/constructed feature.

Input Formats:

can be: **RMEAS/var_1,F(lname1),n,FA(lname2)**

Output Formats:

can be: **None**

Where:

var_1 can be: **CONE**
or: **CYLNDR**
or: **PARPLN**
or: **RCTNGL**
or: **SYMPLN**

CONE signifies that a cone is to be measured.

CYLNDR signifies that a cylinder is to be measured.

F(lname1) is the label of the previously defined feature nominal to be measured.

FA(lname2) (Adjusts Location and/or Orientation)

signifies that the nominal relationship between the reference feature actual, FA(lname2), and the feature nominal being measured, F(lname1), is to be maintained. FA(lname2) must be a previously measured or constructed plane-reducible or point-reducible feature.

Position and orientational deviation are applied when FA(lname2) is a plane reducible feature. Positional deviation only is applied when FA(lname2) is a point reducible feature.

Positional and orientational deviation are applied relative to FA(lname2).

The transformation between the original feature nominal being measured, F(lname1), and its adjusted nominal is used to transform all the subsequent motion and measurement associated with the feature.

Refer to (Figure B.48 — Relative measure / position & approach), (Figure B.49 — Relative measure / retaining relative position), (Figure B.50 — Relative measure / position), and (Figure B.51 — Relative measure / position & approach) for a pictorial representations of features being measured relative to a reference feature.

n is a positive integer representing the number of measurements to be taken in the measurement of the feature. It is only used when manual mode and automatic mode are in effect, and when scanning is not in effect. When scanning has been activated, the scanning parameters and DME's internal algorithms determine the number of measurement points.

PARPLN signifies that a feature of linear size (slot, block) is to be measured.

RCTNGL signifies that a right rectangular prism is to be measured.

SYMPLN signifies that a feature of linear size (slot, block) is to be measured.

Refer to (Figure B.46 — Positional deviation) for an example of applying positional deviation when measuring a feature.
Refer to (Figure B.47 — Positional and orientational deviation) for an example applying positional deviation and orientational deviation when measuring a feature.

The RMEAS statement is usually followed by a series of statements, either PAMEAS, PTMEAS or GOTO statements. The RMEAS statement is terminated with an ENDMES statement.

The DME will measure the feature with its own internal algorithm when automatic mode is active. When the DME lacks this capability, it will default to the next program level.

When programmed mode is active, the DME will follow the given PAMEAS, PTMEAS or GOTO statements to measure the feature.

If the DME does not recognize the specified feature type, and if a measurement sequence is programmed, the DME will follow the measurement sequence and output raw data for the measurement result.

A minimum of 6 points are required for measuring a CONE feature.
A minimum of 6 points are required for measuring a CYLNDR feature.
A minimum of 4 points are required for measuring a PARPLN feature.
A minimum of 9 points are required for measuring a RCTNGL feature.
A minimum of 4 points are required for measuring a SYMPLN feature.

6.156 RMEAS (input format 3)

Function: Causes the DME to measure a centered parallel line, CPARLN, feature relative to another defined/constructed feature.

Input Formats:

can be: **RMEAS/CPARLN,F(lname1),n,var_1 var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **FA(lname2)**
or: **VECBLD,r,n1**

var_2 can be: **,ORIENT**
or: **does not exist**

CPARLN signifies that a centered parallel line feature is to be measured. This implies that at least two opposite parallel lines that are of equal length.

F(lname1) is the label of the previously defined centered parallel line feature nominal to be measured.

FA(lname2) (Adjusts Location and Orientation)

signifies that the nominal relationship between the reference feature actual, FA(lname2), and the feature nominal being measured, F(lname1), is to be maintained. FA(lname2) must be a previously measured or constructed plane-reducible or point-reducible feature.

Position and orientational deviation are applied when FA(lname2) is a plane reducible feature. Positional deviation only is applied when FA(lname2) is a point reducible feature.

Positional and orientational deviation are applied relative to FA(lname2).

The transformation between the original feature nominal being measured, F(lname1), and its adjusted nominal is used to transform all the subsequent motion and measurement associated with the feature.

Refer to (Figure B.48 — Relative measure / position & approach), (Figure B.49 — Relative measure / retaining relative position), (Figure B.50 — Relative measure / position), and (Figure B.51 — Relative measure / position & approach) for a pictorial representations of features being measured relative to a reference feature.

Refer to (Figure B.56 — RMEAS/CPARLN with FA(lname2) and with the ORIENT parameter) for an example of the FA(lname2) parameter when measuring a centered parallel line feature.

n is a positive integer representing the number of measurements to be taken in the measurement of the feature. It is only used when manual mode and automatic mode are in effect, and when scanning is not in effect. When scanning has been activated, the scanning parameters and DME's internal algorithms determine the number of measurement points.

n1 is a positive integer representing the number of sample points to be taken when VECBLD is used. The algorithm of the DME can determine the order and direction (clockwise or counter-clockwise around the feature to be measured) the sample points will be taken.

n1 cannot equal 2.

ORIENT signifies that the DME should adjust the remaining measurement points after measuring the first two points on the long side of the centered parallel line feature.

r is a positive real number representing the incremental value from the defining portion of the feature that the sample points will be taken. The DME will use its own algorithm to place the sample points within this "radius zone" around the feature.

VECBLD signifies the sampling method to be used depending on the value of n1. These samples will be automatically taken by the DME to locate or locate and orient the feature prior to

measuring it:

n1 = 1 (Adjusts Location)

A single point is automatically taken within a specified distance, 'r', from the defining portion of the feature.

Positional deviation is applied along the nominal vector to establish the measurement plane. SNSET/DEPTH is applied to this measurement plane. The nominal vector of the feature being measured is used as the approach vector.

n1 >= 3 (Adjusts Location and Orientation)

'n1' non-collinear points are automatically taken within a specified distance, 'r', from the defining portion of the feature. A measurement plane and normal vector are to be constructed from the sample points. Positional deviation along this normal vector and the orientational deviation are applied. SNSET/DEPTH is applied to the constructed measurement plane. The normal vector is used as the approach vector for measuring the feature.

The first two points in the RMEAS block must be the inspection points on the long side of the centered parallel line feature.

Refer to (Figure B.46 — Positional deviation) for an example of applying positional deviation when measuring a feature.
Refer to (Figure B.47 — Positional and orientational deviation) for an example applying positional deviation and orientational deviation when measuring a feature.

Refer to (Figure B.57 — RMEAS/CPARLN with VECBLD,r,1 and with VECBLD,r,3) for an example of the VECBLD parameter when measuring a centered parallel line feature.

The RMEAS statement is usually followed by a series of statements, either PAMEAS, PTMEAS or GOTO statements. The RMEAS statement is terminated with an ENDMES statement.

The DME will measure the centered parallel line feature with its own internal algorithm when automatic mode is active. When the DME lacks this capability, it will default to the next program level.

When programmed mode is active, the DME will follow the given PAMEAS, PTMEAS or GOTO statements to measure the feature.

If the DME does not recognize the centered parallel line feature, and if a measurement sequence is programmed, the DME will follow the measurement sequence and output raw data for the measurement result.

A minimum of 5 points are required for measuring a round or flat ended centered parallel line feature.

A minimum of 3 points are required for measuring an open ended centered parallel line feature.

6.157 RMEAS (input format 4)

Function: Causes the DME to measure a GCURVE or LINE feature relative to another defined feature and/or by approaching along a specific axis.

Input Formats:

can be: **RMEAS/var_1,F(lname1),n,var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **GCURVE**
or: **LINE**

var_2 can be: **F(lname1) var_3**
or: **FA(lname2) var_3**
or: **VECBLD,r,n1**
or: **XAXIS**
or: **YAXIS**
or: **ZAXIS**

var_3 can be: **,XAXIS**
or: **,YAXIS**
or: **,ZAXIS**
or: **does not exist**

F(lname1) is the label of the previously defined feature nominal to be measured.

F(lname1) var_3 (Adjusts Location)

signifies that the entire GCURVE, F(lname1), is to be measured with successive points being relative to one another. In essence, F(lname1) is being measured relative to itself.

Positional deviation is applied relative to F(lname1).

FA(lname2) (Adjusts Location and/or Orientation)

signifies that the nominal relationship between the reference feature actual, FA(lname2), and the feature nominal being measured, F(lname1), is to be maintained. FA(lname2) must be a previously measured or constructed plane-reducible or point-reducible feature.

Position and orientational deviation are applied when FA(lname2) is a plane reducible feature. Positional deviation only is applied when FA(lname2) is a point reducible feature.

Positional and orientational deviation are applied relative to FA(lname2).

The transformation between the original feature nominal being measured, F(lname1), and its adjusted nominal is used to transform all the subsequent motion and measurement associated with the feature.

Refer to (Figure B.48 — Relative measure / position & approach), (Figure B.49 — Relative measure / retaining relative position), (Figure B.50 — Relative measure / position), and (Figure B.51 — Relative measure / position & approach) for a pictorial representations of features being measured relative to a reference feature.

GCURVE signifies that a generic curve is to be measured.

LINE signifies that a line is to be measured.

n is a positive integer representing the number of measurements to be taken in the measurement of the feature. It is only used when manual mode and automatic mode are in effect, and when scanning is not in effect. When scanning has been activated, the scanning parameters and DME's internal algorithms determine the number of measurement points.

n1 is a positive integer representing the number of sample points to be taken when VECBLD is used. The algorithm of the DME can determine the order and direction (clockwise or counter-clockwise around the feature to be measured) the sample points will be taken.

n1 cannot equal 2.

r is a positive real number representing the incremental value from the defining portion of the feature that the sample points will be taken. The DME will use its own algorithm to place the sample points within this "radius zone" around the feature.

VECBLD signifies the sampling method to be used depending on the value of n1. These samples will be automatically taken by the DME to locate or locate and orient the feature prior to measuring it:

n1 = 1 (Adjusts Location)

A single point is automatically taken within a specified distance, 'r', from the defining portion of the feature.
Positional deviation is applied along the nominal vector to establish the measurement plane. SNSET/DEPTH is applied to this measurement plane. The nominal vector of the feature being measured is used as the approach vector.

n1 >= 3 (Adjusts Location and Orientation)

'n1' non-collinear points are automatically taken within a specified distance, 'r', from the defining portion of the feature. A measurement plane and normal vector are to be constructed from the sample points. Positional deviation along this normal vector and the orientational deviation are applied. SNSET/DEPTH is applied to the constructed measurement plane.

VECBLD is applied to each point on the GCURVE feature.

XAXIS signifies that the probe is to approach the measurement points for the feature parallel to the current Part Coordinate System (PCS) X axis.

YAXIS signifies that the probe is to approach the measurement points for the feature parallel to the current Part Coordinate System (PCS) Y axis.

ZAXIS signifies that the probe is to approach the measurement points for the feature parallel to the current Part Coordinate System (PCS) Z axis.

When VECBLD with n1>=3 is used, orientational deviation is applied to the approach and probe compensation vectors.

When XAXIS, YAXIS, or ZAXIS is used, the DME will approach the measurement points for the feature along a vector parallel to the indicated axis.

The RMEAS statement is usually followed by a series of statements, either PAMEAS, PTMEAS or GOTO statements. The RMEAS statement is terminated with an ENDMES statement.

The DME will measure the feature with its own internal algorithm when automatic mode is active. When the DME lacks this capability, it will default to the next program level.

When programmed mode is active, the DME will follow the given PAMEAS, PTMEAS or GOTO statements to measure the feature.

If the DME does not recognize the specified feature type, and if a measurement sequence is programmed, the DME will follow the measurement sequence and output raw data for the measurement result.

A minimum of 2 points are required for measuring a GCURVE feature.
A minimum of 2 points are required for measuring a LINE feature.

Refer to (Figure B.46 — Positional deviation) for an example of applying positional deviation when measuring a feature.
Refer to (Figure B.47 — Positional and orientational deviation) for an example applying positional deviation and orientational deviation when measuring a feature.

6.158 RMEAS (input format 5)

Function: Causes the DME to measure a PLANE, GSURF, SPHERE, or TORUS feature relative to another defined feature and/or by approaching along a specific axis.

Input Formats:

can be: **RMEAS/var_1,F(lname1),n,var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **PLANE**
or: **GSURF**
or: **SPHERE**
or: **TORUS**

var_2 can be: **FA(lname2) var_3**
or: **XAXIS**
or: **YAXIS**
or: **ZAXIS**

var_3 can be: **,XAXIS**
or: **,YAXIS**
or: **,ZAXIS**
or: **does not exist**

F(lname1) is the label of the previously defined feature nominal to be measured.

FA(lname2) (Adjusts Location and Orientation)

signifies that the nominal relationship between the reference feature actual, FA(lname2), and the feature nominal being measured, F(lname1), is to be maintained. FA(lname2) must be a previously measured or constructed plane-reducible or point-reducible feature.

Position and orientational deviation are applied when FA(lname2) is a plane reducible feature. Positional deviation only is applied when FA(lname2) is a point reducible feature.

Positional and orientational deviation are applied relative to FA(lname2).

The adjusted nominal vector of the feature being measured, F(lname1), is used as the approach vector for measuring the feature.

Refer to (Figure B.48 — Relative measure / position & approach), (Figure B.49 — Relative measure / retaining relative position), (Figure B.50 — Relative measure / position), and (Figure B.51 — Relative measure / position & approach) for a pictorial representations of features being measured relative to a reference feature.

GSURF signifies that a generic surface is to be measured.

n is a positive integer representing the number of measurements to be taken in the measurement of the feature. It is only used when manual mode and automatic mode are in effect, and when scanning is not in effect. When scanning has been activated, the scanning parameters and DME's internal algorithms determine the number of measurement points.

PLANE signifies that a plane is to be measured.

SPHERE signifies that a sphere is to be measured.

TORUS signifies that a torus is to be measured.

XAXIS signifies that the probe is to approach the feature parallel to the current Part Coordinate System (PCS) X axis.

YAXIS signifies that the probe is to approach the feature parallel to the current Part Coordinate System (PCS) Y axis

ZAXIS signifies that the probe is to approach the feature parallel to the current Part Coordinate System (PCS) Z axis.

When XAXIS, YAXIS, or ZAXIS is used, the DME will approach the feature along a vector parallel to the indicated axis. The DME will also perform probe compensation along the nominal vector of the plane.

Refer to (Figure B.46 — Positional deviation) for an example of applying positional deviation when measuring a feature.
Refer to (Figure B.47 — Positional and orientational deviation) for an example applying positional deviation and orientational deviation when measuring a feature.

The RMEAS statement is usually followed by a series of statements, either PAMEAS, PTMEAS or GOTO statements. The RMEAS statement is terminated with an ENDMES statement.

The DME will measure the feature with its own internal algorithm when automatic mode is active. When the DME lacks this capability, it will default to the next program level.

When programmed mode is active, the DME will follow the given PAMEAS, PTMEAS or GOTO statements to measure the feature.

If the DME does not recognize the specified feature type, and if a measurement sequence is programmed, the DME will follow the measurement sequence and output raw data for the measurement result.

A minimum of 3 points are required for measuring a PLANE feature.
A minimum of 3 points are required for measuring a GSURF feature.
A minimum of 4 points are required for measuring a SPHERE feature.
A minimum of 9 points are required for measuring a TORUS feature.

6.159 RMEAS (input format 6)

Function: Causes the DME to measure a POINT feature relative to another defined/constructed feature and/or by approaching along a specific axis.

Input Formats:

can be: **RMEAS/POINT,F(lname1),n,var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **FA(lname2) var_2**
or: **VECBLD,r,n1**
or: **XAXIS**
or: **YAXIS**
or: **ZAXIS**

var_2 can be: **,XAXIS**
or: **,YAXIS**
or: **,ZAXIS**
or: **does not exist**

F(lname1) is the label of the previously defined point feature nominal to be measured.

FA(lname2) (Adjusts Location and/or Orientation)

signifies that the nominal relationship between the reference feature actual, FA(lname2), and the feature nominal being measured, F(lname1), is to be maintained. FA(lname2) must be a previously measured or constructed plane-reducible or point-reducible feature.

Position and orientational deviation are applied when FA(lname2) is a plane reducible feature. Positional deviation only is applied when FA(lname2) is a point reducible feature.

Positional and orientational deviation are applied relative to FA(lname2).

The adjusted nominal vector of the feature nominal being measured, F(lname1), is used as the approach vector for measuring the feature.

Refer to (Figure B.48 — Relative measure / position & approach), (Figure B.49 — Relative measure / retaining relative position), (Figure B.50 — Relative measure / position), and (Figure B.51 — Relative measure / position & approach) for a pictorial representations of features being measured relative to a reference feature.

n is a positive integer representing the number of measurements to be taken in the measurement of the feature. It is only used when manual mode and automatic mode are in effect, and when scanning is not in effect. When scanning has been activated, the scanning parameters and DME's internal algorithms determine the number of measurement points.

n1 is a positive integer representing the number of sample points to be taken when VECBLD is used. The algorithm of the DME can determine the order and direction (clockwise or counter-clockwise around the feature to be measured) the sample points will be taken.

n1 cannot equal 2.

POINT signifies that a point feature is to be measured.

r is a positive real number representing the incremental value from the defining portion of the feature that the sample points will be taken. The DME will use its own algorithm to place the sample points within this "radius zone" around the feature.

VECBLD signifies the sampling method to be used depending on the value of n1. These samples will be automatically taken by the DME to locate or locate and orient the feature prior to measuring it:

n1 >= 3 (Adjusts Location and Orientation)

'n1' non-collinear points are automatically taken within a specified distance, 'r', from the defining portion of the feature. A measurement plane and normal vector are to be constructed from the sample points. Positional deviation along this normal vector and the orientational deviation are applied.

Refer to (Figure B.65 — RMEAS/POINT with VECBLD,r,3 using a surface point) for an example of the VECBLD parameter when measuring a surface point feature.

XAXIS signifies that the probe is to approach the feature parallel to the current Part Coordinate System (PCS) X axis.

YAXIS signifies that the probe is to approach the feature parallel to the current Part Coordinate System (PCS) Y axis.

ZAXIS signifies that the probe is to approach the feature parallel to the current Part Coordinate System (PCS) Z axis.

When XAXIS, YAXIS, or ZAXIS is used, the DME will approach the point along a vector parallel to the indicated axis. The DME will also perform probe compensation along the nominal vector of the point.

The SNSET/DEPTH parameter is only applied when the VECBLD parameter is not specified.

Refer to (Figure B.63 — RMEAS/POINT with a specific axis using an edge point) for an example of measuring an edge point along a specified axis (XAXIS, YAXIS, or ZAXIS) and (Figure B.64 — RMEAS/POINT with a specific axis using a surface point) for an example of measuring a surface point along a specified axis.

Refer to (Figure B.46 — Positional deviation) for an example of applying positional deviation when measuring a feature.
Refer to (Figure B.47 — Positional and orientational deviation) for an example applying positional deviation and orientational deviation when measuring a feature.

Refer to (Figure B.66 — Edge point feature and RMEAS/POINT using a cylindrical probe) for a pictorial representation of an edge point feature and for an example of using a cylindrical probe when measuring an edge point feature.

If a cylindrical probe is used, meaning that an edge point is being measured, it is not valid to also have VECBLD as a parameter (it is not possible to take three points near an edge point). However, FA(lname2), XAXIS, YAXIS, and ZAXIS are valid parameters when using a cylindrical probe.

When VECBLD with n1>=3 is used, the DME will perform probe compensation along the constructed normal vector.

The RMEAS statement is usually followed by a series of statements, either PAMEAS, PTMEAS or GOTO statements. The RMEAS statement is terminated with an ENDMES statement.

The DME will measure the point with its own internal algorithm when automatic mode is active. When the DME lacks this capability, it will default to the next program level.

When programmed mode is active, the DME will follow the given PAMEAS, PTMEAS or GOTO statements to measure the feature.

In the case when n=0, the current sensor position is recorded as the point and assigned to the F(lname1) name.

If the DME does not recognize the point feature, and if a measurement sequence is programmed, the DME will follow the measurement sequence and output raw data for the measurement result.

A single point is required for measuring a POINT feature.

6.160 RMEAS (input format 7)

Function: Causes the DME to measure an EDGEPT feature relative to another feature/measurement and/or by approaching along a specific axis/plane.

Input Formats:

can be: **RMEAS/EDGEPT,F(lname1),n1,var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **FA(lname2) var_3**
or: **VECBLD,r,n2,edge_offset var_4 var_3**
or: **var_2**

var_2 can be: **XAXIS**
or: **YAXIS**
or: **ZAXIS**

var_3 can be: **,var_2**
or: **does not exist**

var_4 can be: **,XDIR**
or: **,YDIR**
or: **,ZDIR**
or: **does not exist**

edge_offset is a non-zero real number representing the distance in the opposite direction of the edge vector from the nominal edge point where the centroid of the VECBLD points or the single surface point is measured. A negative value for edge_offset indicates a positive distance equal to the absolute value of edge_offset along the direction of the edge vector.

EDGEPT signifies that an edge point feature is to be measured.

F(lname1) is the label of the previously defined edge point feature nominal to be measured.

FA(lname2) (Adjusts Location and/or Orientation)

signifies that the nominal relationship between the reference feature actual, FA(lname2), and the feature nominal being measured, F(lname1), is to be maintained. FA(lname2) must be a previously measured or constructed plane-reducible or point-reducible feature.

Position and orientational deviation are applied when FA(lname2) is a plane reducible feature. Positional deviation only is applied when FA(lname2) is a point reducible feature.

Positional and orientational deviation are applied relative to FA(lname2).

The adjusted nominal vector of the feature nominal being measured, F(lname1), is used as the approach vector for measuring the feature.

n1 is a positive integer representing the number of measurements to be taken and is always 1.

n2 is a positive integer representing the number of VECBLD measurements taken having the following values.

n2 = 1 adjusts the position of the feature nominal.
n2 >= 3 adjusts the position and orientation of the nominal.

r is a positive real number representing the radius around the point defined by the edge_offset and is only applicable when n2 >= 3 the edge on the surface that VECBLD measurements are taken.

VECBLD signifies the sampling method to be used depending on the value of n2. These samples will be automatically taken by the DME to locate or locate and orient the feature prior to measuring it.

n2 >= 3 (Adjusts Location and Orientation)

'n2' non-collinear points are automatically taken within a specified distance, 'r', from the defining portion of the feature. A measurement plane and normal vector are to be constructed from the sample points. Positional deviation along this normal vector and the orientational deviation are applied.

XAXIS signifies that the measurement approach direction is parallel to the current datum X-axis.

XDIR signifies that the VECBLD measurement approach direction is parallel to the current datum X-axis.

YAXIS signifies that the measurement approach direction is parallel to the current datum Y-axis.

YDIR signifies that the VECBLD measurement approach direction is parallel to the current datum Y-axis.

ZAXIS signifies that the measurement approach direction is parallel to the current datum Z-axis.

ZDIR signifies that the VECBLD measurement approach direction is parallel to the current datum Z-axis.

Refer to (Figure B.46 — Positional deviation) for an example of applying positional deviation when measuring a feature.
Refer to (Figure B.47 — Positional and orientational deviation) for an example applying positional deviation and orientational deviation when measuring a feature.

The RMEAS statement is usually followed by a series of statements, either PAMEAS, PTMEAS or GOTO statements. The RMEAS…ENDMES block is terminated with an ENDMES statement.

The DME will measure the edge point feature with its own internal algorithm when automatic mode is active. When the DME lacks this capability, it will default to the next program level.

SNSET/DEPTH is applied in the adjusted orientation direction of the surface direction as specified in the FEAT/EDGEPT statement.

When programmed mode is active, the DME will follow the given PAMEAS, PTMEAS or GOTO statements to measure the feature.

If the DME does not recognize the edge point feature, and if a measurement sequence is programmed, the DME will follow the measurement sequence and output raw data for the measurement result.

Note: Refer to (Figure B.59 — RMEAS/EDGEPT,F(edge),1,VECBLD,0,1,5), (Figure B.60 — RMEAS/EDGEPT,F(edge),1,VECBLD,0,1,5,-XAXIS), (Figure B.61 — RMEAS/EDGEPT,F(edge),1,VECBLD,0,1,5,-XAXIS,ZDIR), and (Figure B.63 — RMEAS/POINT with a specific axis using an edge point) for illustrations of an edge point feature and also for an examples of using a cylindrical probe when measuring an edge point feature.

6.161 ROTAB

Function: Explicitly controls the motion of a rotary table on a DME.

Input Formats:

can be: **ROTAB/RT(lname1),var_1**

Output Formats:

can be: **ROTAB/RT(lname1),var_1**

Where:

var_1 can be: **var_2,var_3,ang var_4**
or: **F(lname1),var_3,var_5**
or: **FA(lname1),var_3,var_5**

var_2 can be: **ABSL,var_5**
or: **INCR,var_6**

var_3 can be: **ROTTOT**
or: **ROTORG**
or: **ROTNUL**

var_4 can be: **,FZ,dev**
or: **does not exist**

var_5 can be: **CW**
or: **CCW**
or: **SHORT**

var_6 can be: **CW**
or: **CCW**

ABSL signifies that the rotation is to an absolute position.

ang is the amount of rotation in angular units of degrees:minutes:seconds, radians, or decimal form, as specified in the UNITS statement.

If var_2 uses INCR then 'ang' is the relative position at which to stop rotation.

If var_2 uses ABSL, then 'ang' is the absolute position at which to stop rotation.

CCW signifies counter-clockwise rotation.

CW signifies clockwise rotation.

dev is the maximum allowed absolute angular deviation between the target position and the final rotary table orientation in angular units of degrees:minutes:seconds, radians, or decimal form, as specified in the UNITS statement.

F(lname1) is the label of a feature nominal to be used in rotating the rotary table to a position such that the direction of F(lname) projected in the rotary table's plane will be parallel to current sensor's shank projected in the rotary table's plane.

FA(lname1) is the label of a feature actual to be used in rotating the rotary table to a position such that the direction of FA(lname) projected in the rotary table's plane will be parallel to current sensor's shank projected in the rotary table's plane.

FZ signifies that the value for the allowed angular deviation is to follow. The angular deviation only applies to the current statement.

INCR signifies that the rotation is to be from the current position.

ROTNUL signifies that no updating of the part's coordinate system is to occur. Feature geometry maintains its positions relative to the current coordinate system before and after the ROTAB statement.

ROTORG signifies that a partial updating of the part's coordinate system, including only the origin, will be executed with the desired rotation. The axis alignment will remain as established from the DATSET statement.

ROTTOT signifies that total updating of the part's coordinate system established by the DATSET statement (that is, datums, part origin, and axis alignment) will be executed with the desired rotation. Feature geometry rotates with the table/part as a result of the ROTAB statement.

RT(lname1) is the label of the previously defined rotary table that is to be used.

SHORT signifies that the rotary table will rotate to the desired position via the shortest direction.

The ROTAB statement is passed to the output file when executed.

Note 1: The following statements are interrelated in the use of rotary tables:

ROTDEF, ROTSET, ROTAB, FEDRAT, ACLRAT, and CALIB.

Note 2: For an illustration of part coordinate system updating, refer to (Figure B.67 — Part coordinate system updating following a rotation move).

Note 3: Refer to example A.28.

6.162 ROTATE

Function: Rotates a part coordinate system about an axis, and assigns a label to it.

Input Formats:

can be: **D(lname)=ROTATE/var_1,var_2**

Output Formats:

can be all: **D(lname)=ROTATE/var_1,var_2**
DA(lname)=ROTATE/TRMATX,a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3

Where:

var_1 can be: **XAXIS**
or: **YAXIS**
or: **ZAXIS**

var_2 can be: **ang**
or: **F(lname1),var_3**
or: **FA(lname2),var_3**
or: **DAT(x),var_3**

var_3 can be: **XDIR**
or: **-XDIR**
or: **YDIR**
or: **-YDIR**
or: **ZDIR**
or: **-ZDIR**

a1,a2,a3, b1,b2,b3, c1,c2,c3, d1,d2,d3 satisfy the following transformation equations:

(a1)x + (b1)y + (c1)z + d1 = x'
(a2)x + (b2)y + (c2)z + d2 = y'
(a3)x + (b3)y + (c3)z + d3 = z'

Where: x', y', and z' are the coordinates in the current part coordinate system (after the transformation has been applied) of any point, and x, y, and z are the coordinates in the previous coordinate system of the same point.

ang is the angle of rotation in current system units.

DAT(x) is the previously defined datum label to be aligned with the following axis direction for the new part coordinate system.

F(lname1) is the label of the feature nominal to be aligned with the following axis direction for the new part coordinate system.

FA(lname2) is the label of the measured feature actual to be aligned with the following axis direction for the new part coordinate system.

lname is an alphanumeric label name assigned to the new part coordinate system.

TRMATX signifies that the alignment and transformation information from the previous coordinate system is being output.

XAXIS signifies that the rotation occurs about the X axis.

XDIR signifies that the positive X direction is given by the direction vector of the preceding feature.

-XDIR signifies that the negative -X direction is given by the direction vector of the preceding feature.

YAXIS signifies that the rotation occurs about the Y axis.

YDIR signifies that the positive Y direction is given by the direction vector of the preceding feature.

-YDIR signifies that the negative -Y direction is given by the direction vector of the preceding feature.

ZAXIS signifies that the rotation occurs about the Z axis.

ZDIR signifies that the positive Z direction is given by the direction vector of the preceding feature.

-ZDIR signifies that the negative -Z direction is given by the direction vector of the preceding feature.

A SAVE statement must be issued prior to the ROTATE statement if the current part coordinate system is to be used again with the RECALL statement. The new part coordinate system is activated when the ROTATE statement is executed.

The ROTATE statement is passed to the output file at the time it is executed.

6.163 ROTDEF

Function: Defines a rotary table, and assigns a label to it.

Input Formats:

can be: **RT(lname)=ROTDEF/x,y,z,i,j,k var_1**

Output Formats:

can be: **RT(lname)=ROTDEF/x,y,z,i,j,k var_1**

Where:

var_1 can be: **,RT(lname1)**
or: **does not exist**

i,j,k is the i,j,k unit vector for the rotary table's axis of rotation in a direction away from the rotary table's workpiece locating surface.

lname is an alphanumeric label name assigned to the rotary table.

RT(lname1) is the label of another previously defined rotary table that will support the currently defined rotary table.

x,y,z are the x,y,z coordinates from the DME's zero position with respect to the rotary table's center point.

The rotary table's center point is defined as the point of intersection of the axis of rotation with the plane being used as the rotary table's workpiece locating surface (that is, the center point of rotation on the surface of the rotary table).

The x,y,z coordinates and i,j,k vectors are given relative to the machine coordinate system. For cascaded tables, where a master table has been specified, the x,y,z coordinates are defined at a master table index value of zero.

The ROTDEF statement is passed to the output file when executed.

Note 1: The following statements are interrelated in the use of rotary tables:

ROTDEF, ROTSET, ROTAB, FEDRAT, ACLRAT, and CALIB.

Note 2: Refer to A.28 and (Figure A.5 — Rotary table application sequence) for an illustration of a rotary table program applying the use of several related statements.

Note 3: For an illustration of a sensor rotation versus a rotary table rotation, refer to (Figure B.68 — Part probe orientation) .

Note 4: For an illustration of rotary table positioning, refer to (Figure B.69 — Rotary table positioning with respect to the machine coordinate system (M.C.S.) zero point).

6.164 ROTSET

Function: Resets the angular counter value for a rotary table.

Input Formats:

can be: **ROTSET/RT(lname1),val**

Output Formats:

can be: **ROTSET/RT(lname1),val**

Where:

RT(lname1) is the label of the rotary table to be reset.

val is the reset value ranging from –360.00 to 360.00 degrees, given in current system units.

The format of the value for angular measurement used must be consistent with the angular units established in the UNITS statement.

The ROTSET statement is passed to the output file when executed.

Note: The following statements are interrelated in the use of rotary tables:

ROTDEF, ROTSET, ROTAB, FEDRAT, ACLRAT, and CALIB.

6.165 SAVE

Function: Stores part coordinate system datum sets, sensor calibration data, rotary table data, measured feature actual data or DML for later recall.

Input Formats:

can be: **SAVE/var_1 var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **D(lname1)**
or: **DA(lname2)**
or: **S(lname3)**
or: **SA(lname4)**
or: **FA(lname5)**
or: **RT(lname6)**
or: **DML,DID(lname7),DA(lname8)**

var_2 can be: **,DID(lname7)**
or: **does not exist**

D(lname1) is the label of the part coordinate system that is to be saved.

DA(lname2) is the label of the part coordinate system that is to be saved.

DA(lname8) is the label of the part coordinate system in which the Common Space DML data will be saved.

DID(lname7) is the device identification label, assigned by the DEVICE statement to a system device.

DML signifies the dimensiional mark-up language will be saved to the device specified.

FA(lname5) is the label of the measurement feature actual that is to be saved.

RT(lname6) is the label of the rotary table that is to be saved.

S(lname3) is the label of the sensor that is to be saved. This only applies to programs written for ANSI/CAM-I 101 1990.

SA(lname4) is the label of the calibrated sensor that is to be saved.

When the DID(lname7) is not specified, the information will be saved in open system devices or files.

Note 1: When the DA(lname2), SA(lname4), FA(lname5), or RT(lname6) is required for later use in the input program, it is recalled with the RECALL statement.

Note 2: DID(lname7) is used to augment the internal DME storage by providing control for the persistent file storage of DA(lname2)s, SA(lname4)s, FA(lname5)s, and RT(lname6)s

6.166 SCNMOD

Function: Activates or deactivates scanning mode for subsequent feature measurements.

Input Formats:

can be: **SCNMOD/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **ON**
or: **OFF**

OFF signifies that scanning mode is to be disabled.

ON signifies that scanning mode is to be enabled.

The SCNMOD statement cannot be issued within a feature MEAS...ENDMES block.

6.167 SCNSET

Function: Specifies the sampling method and rate for scanning.

Input Formats:

can be: **SCNSET/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **DRAG,var_2 var_3**
or: **NONCON,var_2**
or: **PECK,var_2**
or: **STOP,var_4**
or: **VENDOR,var_5**

var_2 can be: **DIST,dist var_6**
or: **CHORD,chord var_7**
or: **TIME,time**
or: **ANGLE,ang**
or: **DEFALT**

var_3 can be: **,FORCE,force**
or: **,DEFLECTION,deflval**
or: **does not exist**

var_4 can be: **PLANE,i,j,k var_8**
or: **SPHERE,sphere_rad**

var_5 can be: **FORM**
or: **POS**
or: **SIZE**

var_6 can be: **,XAXIS**
or: **,YAXIS**
or: **,ZAXIS**
or: **does not exist**

var_7 can be: **,maxdist**
or: **does not exist**

var_8 can be: **,RADIUS,plane_rad var_9**
or: **var_9**

var_9 can be: **,COUNT,stop_count**
or: **does not exist**

ang is a real number representing the increment value of rotation of the rotary table for sampling.

ANGLE signifies that samples will be taken at rotary table angle increments of 'ang'.

chord is a positive real number representing the maximum chordal distance allowed when sampling. The value of 'chord' will be interpreted based on the current UNITS statement.

CHORD signifies that samples will be taken such that the maximum chordal deviation between the surface of the part and a straight line between any two consecutive points does not exceed 'chord'.

COUNT signifies that scanning measurements will stop when the stop plane has been crossed the number of times specified by the 'stop_count' value.

DEFALT signifies that samples will be taken using the default settings of the DME.

deflval is a positive real number representing the probe displacement during scanning.

DEFLECTION signifies that the displacement value for the probe follows.

dist is a positive real number representing the increment value or distance for sampling. The value of 'dist' will be interpreted based on the current UNITS statement.

DIST signifies that samples will be taken at distance increments of 'dist'. XAXIS, YAXIS or ZAXIS signify that distance increments are to be measured along the specified coordinate axis. If no coordinate axis is specified, distance will be measured along the surface of the part from one point to the next.

DRAG signifies the DME will use a drag routine while scanning.

FORCE signifies that the maximum contact force that the scanning head applies to the part follows.

force is a positive real number representing the maximum contact force that will be applied to the part by the scanning head in DME units.

FORM signifies that the scan will be done with DME controlled parameters that are optimised to deliver the minimum uncertainty for the form of scanned features.

i,j,k is the direction vector of the stop plane to be used. The gate must be crossed in a positive sense for the scan to stop.

maxdist is a positive real number representing the maximum allowable distance between measurement points when maximum chordal deviation has been specified.

NONCON signifies the DME will use a non-contact routine while scanning.

PECK signifies the DME will use a pecking routine while scanning.

PLANE signifies that a planar stop criteria is being used to stop scanning measurements. Scanning measurement will stop when the plane is crossed.

plane_rad is a non-negative real number that is the radius of the circular area on the stop plane.

POS signifies that the scan will be done with DME controlled parameters that are optimised to deliver the minimum uncertainty for the position of scanned features.

RADIUS signifies that the stop plane is bounded by a circular area on the stop plane and the radius of the circular area follows.

SIZE signifies that the scan will be done with DME controlled parameters that are optimised to deliver the minimum uncertainty for the size of scanned features.

SPHERE signifies that a spherical stop criteria is being used to stop scanning measuremetns.

sphere_rad is a non-negative real number that is the radius of the sphere in which scanning measuremetns will stop.

STOP signifies that the scan will stop when the parameter defined by var_4 is reached. Refer to (Figure B.90 — Stop plane when scannig).

The STOP minor word is only applicable to paths defined by the PATH / UNKNOWN statement.

stop_count is a positive nonzero integer value that is the number of times that the stop plane will be crossed before scanning measurement stops.

time is a positive real number representing the time increment in seconds between sampling. The value of 'time' will be interpreted based on the current UNITS statement.

TIME signifies that samples will be taken at time increments of 'n' seconds.

VENDOR signifies that velocity, acceleration and point density will be set vendor specific.

x,y,z is the location of the scan stop "gate".

XAXIS signifies that distance increments are to be measured along the X axis.

YAXIS signifies that distance increments are to be measured along the Y axis.

ZAXIS signifies that distance increments are to be measured along the Z axis.

The SCNSET statement cannot be issued within a feature MEAS...ENDMES block.

The SCNSET/VENDOR statement will overrule all previous SCNSET/… statements. The settings done by the vendor can however be modified after a SCNSET/VENDOR statement.

6.168 SELECT

Function: Executes statements according to the results of one or more equality tests.

Input Formats:

can be: **SELECT/arg**

Output Formats:

can be: **None**

Synopsis: SELECT/arg

```
    CASE/arg_1
      ...
    CASE/arg_N
      executable statement(s)
    ENDCAS

    CASE/arg_N+1
      ...
    CASE/arg_N+M
      executable statement(s)
    ENDCAS

      ...

    DFTCAS
      executable statement(s)
    ENDCAS

ENDSEL
```

Where:

arg is an integer numeric expression or character expression.

A SELECT...ENDSEL block starts with a SELECT statement and ends with an ENDSEL statement. Between the SELECT and the ENDSEL there are one or more sub-blocks and then an optional default block. Each sub-block has one or more CASE statements followed by zero to many executable statements followed by an ENDCAS statement. A default block has a DFTCAS statement followed by zero to many executable statements followed by an ENDCAS statement.

For the purpose of determining equality the literal argument of the CASE statement must be of the same data type as the argument of the SELECT statement. If the arguments are different, they are not equal. If both arguments are character strings, then the strings must be the same length and contain the same, case-sensitive characters in the same order to be equal. If both arguments are integers then they must be the same number to be equal.

The functioning of the SELECT...ENDSEL block is as follows.

The expression passed as the argument to the SELECT statement is evaluated once to determine an integer number or a character string. This integer or string (call it val) will be compared with the literal arguments of the CASE statements in the order given until the first one is found that is equal to val. If a CASE statement is found with an argument equal to val, no more comparisons are made and the executable statements in the sub-block containing the CASE statement are executed. If there are no statements in the sub-block, then no statements in the SELECT...ENDSEL block are executed. If no CASE statement is found with an argument equal to val and there is a default block, the executable statements in the default block are executed. If there are no statements in the default block, then no statements in the SELECT...ENDSEL block are executed. If there is no default block and val is not equal to any CASE argument, then no statements in the SELECT...ENDSEL block are executed. After executing the SELECT...ENDSEL block as just described, the execution is passed to the statement after the ENDSEL statement.

There is no limit to the number of CASE statements in a SELECT..ENDSEL block.

Note 1: If two or more CASE statements have the same literal value then only the first will become a starting point for statement execution and the statements following the other CASE statements may or may not be executed depending on the placement of ENDCAS statements.

Note 2: Refer to example A.29 and section 5.2.4.2.2.

6.169 SENSOR

Function: Defines a sensor component capable of making measurement, and assigns a label to it.

Input Formats:

can be: **SS(lname)=SENSOR/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **PROBE,dx,dy,dz,sni,snj,snk,diam var_2 var_3**
or: **MLTPRB,n var_4 var_3**
or: **VIDEO,dx,dy,dz,sni,snj,snk,sci,scj,sck,focal,mag,apert var_3**
or: **LASER,dx,dy,dz,sni,snj,snk,sci,scj,sck,power,shuter var_3**
or: **INFRED,dx,dy,dz,sni,snj,snk,sci,scj,sck,fovx,fovy,freq,dwell var_3**
or: **NONCON,dx,dy,dz,sni,snj,snk,sci,scj,sck,proben var_3**
or: **POINT,dx,dy,dz,sni,snj,snk,sci,scj,sck,fovn var_3**
or: **LINE,dx,dy,dz,sni,snj,snk,sci,scj,sck,fovn,fovc var_3**
or: **AREA,dx,dy,dz,sni,snj,snk,sci,scj,sck,fovc,fovo,fovn var_3**

var_2 can be: **,SPHERE**
or: **,CYLNDR,len1**
or: **,DISK,thkn**
or: **does not exist**

var_3 can be: **,'data_stor','data_list',var_5**
or: **does not exist**

var_4 can be: **,var_6,dx,dy,dz,sni,snj,snk,diam var_7**

var_5 can be: **'data_item'**
or: **index**

var_6 can be: **'desc'**
or: **tipnum**

var_7 can be: **var_4**
or: **does not exist**

apert is a real number representing the aperture setting.

AREA signifies a sensor that can measure along all sensor axis directions.

CYLNDR signifies that the probe tip being defined is cylindrical. It can be used when a feature is measured with the shaft of the probe. The 'tip center point' location for this type of probe is the axial and radial center of the usable probe geometry of length 'len1'. Refer to (Figure B.77 — Cylindrical and disk tips)

'data_item' is the name of a data item within the data list.

'data_list' is the name of a section within a DME Hardware Descriptor that contains one or more data items.

'data_store' is a file or database called a "DME Hardware Descriptor", that contains DME specific information that may be needed by a DMIS program. The DMIS program can get access to a specific item using three names (data_store, data_list, and data_item), which is compatible with the way items within operating system initialization files are accessed.

'desc' is the description of a stylus on a multi-tip sensor, enclosed with apostrophes. Each tip description on a multi-tip sensor must be unique.

diam is a positive real number representing the diameter of the sensor tip.

DISK signifies that the probe tip being defined is a disk. The "tip center point" location for this type probe is the axial and radial center of the usable probe geometry of length 'thkn'. Refer to (Figure B.77 — Cylindrical and disk tips)

dwell is a positive real number representing the dwell time value in seconds.

dx,dy,dz is the nominal offset from the mounting point for the sensor to the effective center of the sensor.

focal is a non-negative real number representing the focal distance.

fovc is a real number representing the field of view in the clocking direction of the sensor.

fovn is a real number representing the field of view in the nominal direction of the sensor.

fovo is a real number representing the field of view in the direction orthogonal to both the nominal and clocking directions of the sensor.

fovx is a real number representing the field of view size in the X axis direction.

fovy is a real number representing the field of view size in the Y axis direction.

freq is a positive real number representing the frequency value in megahertz.

index is a positive integer number used to select the n'th data item within a data list.

INFRED signifies that an infrared sensor is being defined.

lname is an alphanumeric label name assigned to the sensor.

LASER signifies that a laser sensor is being defined.

len1 is a positive real number representing the length of the cylindrical portion of the probe which will be used for measuring. Refer to (Figure B.77 — Cylindrical and disk tips)

LINE signifies a sensor that can measure along the sensor's normal and clocking axis directions.

mag is a real number representing the magnification of the lens in decimal form.

MLTPRB signifies that a sensor with multiple probes is being defined.

n is a positive integer representing the number of styli on a multi-tip probe.

NONCON signifies that a non-contact sensor is being defined. For example, an electrical capacitance probe having several sensors.

POINT signifies a sensor that can measure along the sensor's normal axis direction.

power is a real number representing the required power setting in watts.

PROBE signifies that a CMM probe is being defined.

proben is a positive integer representing the number of sensors, or probes, that this device provides.

sci,scj,sck is the sensor clocking direction in the current sensor coordinate system. The sensor clocking direction must be orthogonal to the sensor normal direction.

shuter is a real number representing the shutter speed in milliseconds.

sni,snj,snk is the sensor normal direction in the current sensor coordinate system.

SPHERE signifies that the probe tip being defined is spherical.

thkn is a positive real number representing the thickness of the disk probe. Refer to (Figure B.77 — Cylindrical and disk tips)

tipnum is a positive integer representing the number of a stylus on a multi-tip sensor. Each tip number on a multi-tip sensor must be unique.

VIDEO signifies that a video camera is being defined.

Refer to example A.30 and sub-clauses.

Note 1: Refer to example A.30.2 and (Figure A.7 — Multi-probes from multiple ports on the head) for illustration and sample code used to construct a sensor component.

Note 2: Refer to example A.30.4 and (Figure A.10 — Multi-probe from an extension & from a single port) for illustration and sample code used to define a sensor component using MLTPRB.

Note 3: Refer to example A.30.5 and (Figure A.11 — Sensor components in an automated changer rack) for illustration and sample code used to define a more complex sensor definition.

Note 4: The parameters in the SENSOR statement are primarily used to document and communicate the sensor definition to other applications such as offline programming and can be used for error checking.

6.170 SIMREQT

Function: Starts a simultaneous requirement block. The simultaneous requirement defined by a SIMREQT…ENDSIMREQT block is used to associate tolerances to be evaluated in a common datum reference frame.

Input Formats:

can be: **SR(lname)=SIMREQT/var_1**

Output Formats:

can be: **SR(lname)=SIMREQT/var_1**

Where:

var_1 can be: **FIRST**
or: **OPTIMAL**

FIRST signifies that any transformation matrix that satisfies the simultaneous requirement process will be calculated and used in the evaluation of tolerance actuals.

lname is alphanumeric label name assigned to the simultaneous requirement.

OPTIMAL signifies that the optimal transformation matrix that satisfies the simultaneous requirement process will be calculated and used in the evaluation of tolerance actuals.

The SIMREQT statement signifies the beginning of an association of several EVAL and/or OUTPUT statements containing features and tolerances with simultaneous requirements in a common datum reference frame and suppresses the execution of these statements until the corresponding ENDSIMREQT is executed.

The SIMREQT statement is passed to the output file when executed.

6.171 SNSDEF (input format 1)

Function: Defines a probe sensor used by the DME in making measurements, and assigns a label to it. Several formats are used to define sensor types.

Input Formats:

can be: **S(lname)=SNSDEF/PROBE,var_1,var_2 var_3**

Output Formats:

can be: **S(lname)=SNSDEF/PROBE,var_1,var_2 var_3**

or: **SA(lname)=SNSDEF/OFFSET,ox,oy,oz,sns_size,art_size, $**
art_form,art_label,ax,ay,az,datetime

Where:

var_1 can be: **FIXED**
or: **INDEX**

var_2 can be: **CART,dx,dy,dz,ti,tj,tk,diam**
or: **POL,tilt,rot,ti,tj,tk,len1,diam**
or: **VEC,i,j,k,ti,tj,tk,len1,diam**
or: **var_4,CART,rx,ry,rz,diam**
or: **var_4,VEC,ri,rj,rk,rlen,diam**

var_3 can be: **,SPHERE**
or: **,CYLNDR,len2**
or: **,DISK,thkn**
or: **does not exist**

var_4 can be: **S(lname1)**
or: **SA(lname2)**

art_form is a positive real number representing the measured form error of the artifact during sensor calibration.

art_label is the label of the artifact feature actual or feature nominal used for calibration and specified in the CALIB statement.

art_size is a positive real number representing the measured size of the artifact used for sensor calibration.

ax,ay,az are the Cartesian coordinates of the calibration artifact in machine coordinates.

CART signifies that the probe tip location is defined using Cartesian coordinates.

CYLNDR signifies the probe tip being defined is cylindrical. It can be used when a feature is measured with the shaft of the probe. The tip center point location for this type probe is the axial and radial center of the usable probe geometry of length 'len2'. Refer to (Figure B.77 — Cylindrical and disk tips)

datetime is the date and time at completion of the last calibration of the sensor in the format 'YYYY/MM/DD HH:mm:SS.sss', where YYYY is the year, MM is the number of the month, and DD is the day within the month, HH is hours from 0 to 23, mm is minutes, SS is seconds, and sss is milliseconds.

diam is a positive real number representing the diameter of the sensor tip.

DISK signifies that the probe tip being defined is a disk. The "tip center point" location for this type probe is the axial and radial center of the usable probe geometry of length 'thkn'. Refer to (Figure B.77 — Cylindrical and disk tips)

dx,dy,dz is the distance in x, y, and z in probe coordinates from the fixed sensor mount location for fixed probes, or the pivot/hinge point for indexable probes, to the probe tip center point. If using INDEX and the probe tip center is offset from the sensor axis, dx,dy,dz define the orientation of the sensor in the indexable head. Refer to (Figure B.71 — Indexable head definitions) and (Figure B.72 — Indexable head Cartesian coordinates) .

FIXED signifies that a fixed probe is being defined.

i,j,k is the direction vector in probe coordinates from the fixed sensor mount location for fixed probes, or the pivot/hinge point for indexable probes, to the probe tip center point. If using INDEX and the probe tip center is offset from the sensor axis, i,j,k define the orientation of the indexable head. Refer to (Figure B.71 — Indexable head definitions) and (Figure B.73 — Indexable head polar coordinates).

INDEX signifies that an indexable probe is being defined.

lname is an alphanumeric label name assigned to the sensor.

len1 is a positive real number representing the length from the fixed sensor mount location for fixed probes, or the pivot/hinge point for indexable probes, to the probe tip center point. If using INDEX and the probe tip center is offset from the sensor axis, len1 defines the theoretical point that can be used as the master tip in a subsequent SNSDEF statement to define the offset tip location using the same techniques as would be used to define tips for a probe cluster. Refer to (Figure B.73 — Indexable head polar coordinates) and (Figure B.74 — Indexable head vector orientation).

len2 is a positive real number representing the length of the cylindrical portion of the probe which will be used for measuring. Refer to (Figure B.77 — Cylindrical and disk tips).

OFFSET signifies the actual Cartesian coordinates in the machine coordinate system of the calibrated sensor as well as the sensor's actual diameter.

ox,oy,oz is the offset of the sensor center from the sensor mount point in the sensor mount coordinate system. Refer to (Figure B.78 — Offset tip outputs).

POL signifies that the probe tip location is defined using polar coordinates.

PROBE signifies that a CMM probe is being defined.

ri,rj,rk is the relative direction vector from the master probe tip center to the probe tip center being defined (in the same coordinate system as the master probe tip's coordinate system). Refer to (Figure B.76 — Offset tip vector).

rlen is a positive real number representing the distance along ri,rj,rk from the master probe tip center to the probe tip center being defined. Refer to (Figure B.76 — Offset tip vector).

rot is the sensor's angle of rotation from the probe coordinate system's X axis in the XY plane of the probe coordinate system. Refer to (Figure B.71 — Indexable head definitions) and (Figure B.73 — Indexable head polar coordinates).

rx,ry,rz is the relative distance from the master probe tip center to the probe tip center being defined (in the same coordinate system as the master probe tip's coordinate system). Refer to (Figure B.75 — Offset tip Cartesian).

S(lname1) is an alphanumeric label assigned to the sensor that this definition is relative to. This type of definition would be used in the case of a cluster or star probe, where the master probe tip would be a previously defined probe and the remainder of the cluster or star is defined relative to the master tip.

SA(lname) is the label of the previously calibrated sensor.

SA(lname2) is the label of the previously calibrated sensor that this definition is relative to. This type of definition would be used in the case of a cluster or star probe, where the master probe tip would be a previously defined probe and the remainder of the cluster or star is defined relative to the master tip.

sns_size is a positive real number representing the effective size of the sensor.

SPHERE signifies that the probe tip being defined is spherical.

thkn is a positive real number representing the thickness of the disk probe. Refer to (Figure B.77 — Cylindrical and disk tips)

ti,tj,tk is the unit vector of the sensor mount axis.

ti,tj,tk is a vector representing the centerline of the DME ram (probe major axis) and points toward the sensor mount.

The i,j,k vector represents the direction of the probe tip relative to the sensor mount. This vector always points from the sensor mount to the probe tip.

The ZVEC parameter of the SNSMNT statement will always equal ti,tj,tk. If the SNSMNT statement is used, the ti,tj,tk values in the SNSDEF statement must be the same as those in the ZVEC parameter. Refer to (Figure B.70 — Probe sensor illustration).

tilt is the sensor's angle from the probe coordinate system's Z axis. Refer to (Figure B.71 — Indexable head definitions) and (Figure B.73 — Indexable head polar coordinates).

VEC signifies that the probe tip location is defined using a direction vector and a length along the vector.

All probe positions are defined relative to the sensor mount in the probe coordinate system.

Refer to example A.30 and sub-clauses.

The SNSDEF/PROBE statement is passed to the output file by execution of the OUTPUT statement.

Note: The parameters in the SENSDEF(input format 1) through SENSDEF(input format 6) statements are primarily used to document and communicate the sensor definition to other applications such as offline programming and can be used for error checking.

6.172 SNSDEF (input format 2)

Function: Defines a video sensor used by the DME in making measurements, and assigns a label to it. Several formats are used to define sensor types.

Input Formats:

can be: **S(lname)=SNSDEF/VIDEO,var_1,var_2,focal,mag,apert**

Output Formats:

can be: **S(lname)=SNSDEF/VIDEO,var_1,var_2,focal,mag,apert**

or: **SA(lname)=SNSDEF/OFFSET,ox,oy,oz,sns_size,art_size, $**
art_form,art_label,ax,ay,az,datetime

Where:

var_1 can be: **FIXED**
or: **INDEX**

var_2 can be: **CART,dx,dy,dz,ti,tj,tk**
or: **POL,tilt,rot,ti,tj,tk**
or: **VEC,i,j,k,ti,tj,tk**

apert is a real number representing the aperture setting.

art_form is a positive real number representing the measured form error of the artifact during sensor calibration.

art_label is the label of the artifact feature actual or feature nominal used for calibration and specified in the CALIB statement.

art_size is a positive real number representing the measured size of the artifact used for sensor calibration.

ax,ay,az are the Cartesian coordinates of the calibration artifact in machine coordinates.

CART signifies that the sensor location is to be defined in Cartesian coordinates.

datetime is the date and time at completion of the last calibration of the sensor in the format 'YYYY/MM/DD HH:mm:SS.sss', where YYYY is the year, MM is the number of the month, and DD is the day within the month, HH is hours from 0 to 23, mm is minutes, SS is seconds, and sss is milliseconds.

dx,dy,dz are the distances in X, Y, and Z between the sensor's axis orientation point, and the sensor reference point in probe coordinates.

FIXED signifies that a fixed sensor is being defined.

focal is a positive real number representing the focal distance in current system units.

i,j,k is the unit vector of the optical axis.

INDEX signifies that an indexable sensor is being defined.

lname is an alphanumeric label name assigned to the sensor.

mag is a real number representing the magnification of the lens in decimal form.

OFFSET signifies the actual Cartesian coordinates in the machine coordinate system of the calibrated sensor as well as the sensor's actual diameter.

ox,oy,oz is the offset of the sensor center from the sensor mount point in the sensor mount coordinate system. Refer to (Figure B.78 — Offset tip outputs) .

POL signifies that the sensor location is to be defined in polar coordinates.

rot is the sensor's angle of rotation with respect to the probe coordinate system's positive X axis in the XY plane of the probe coordinate system.

sns_size is a positive real number representing the effective size of the sensor.

ti,tj,tk is the unit vector of the sensor mount socket's axis.

tilt is the sensor's angle of tilt with respect to the fully extended position where the tilt angle is zero.

VEC signifies that the sensor location is to be defined in unit vectors.

VIDEO signifies that a video camera is being defined.

The sensor definition is repeated with a new label if any of the parameters are adjustable. In the following example, a camera with a zoom lens requires a new label to be associated with each zoom position.

Example:

```
S(zoom_10x)=SNSDEF/VIDEO,FIXED,VEC,0,0,-1,0,0,-1,20,10.0,5.6
S(zoom_20x)=SNSDEF/VIDEO,FIXED,VEC,0,0,-1,0,0,-1,20,20.0,5.6
```

A camera with a motorized tiltable optical axis would require a label to be associated with each tilt position:

Example:

```
S(tilt_45)=SNSDEF/VIDEO,INDEX,VEC,.707,0,-.707,0,0,-1,20,20.0,5.6
S(tilt_30)=SNSDEF/VIDEO,INDEX,VEC,.500,0,-.500,0,0,-1,20,20.0,5.6
```

The SNSDEF/VIDEO statement is passed to the output file by execution of the OUTPUT statement.

Note 1: Additional VIDEO sensor settings can be defined with other DMIS statements:

DMIS statement	Sensor setting
LITDEF	Video lighting
WINDEF	Video viewing windows
FILDEF	Video filters
ALGDEF	Video algorithms

Note 2: These additional sensor settings, along with scaling, automatic focus, and intensity, are activated with the SNSET statement.

Note 3: Refer to example A.31.

Note 4: The parameters in the SENSDEF(input format 1) through SENSDEF(input format 6) statements are primarily used to document and communicate the sensor definition to other applications such as offline programming and can be used for error checking.

6.173 SNSDEF (input format 3)

Function: Defines a laser sensor used by the DME in making measurements, and assigns a label to it. Several formats are used to define sensor types.

Input Formats:

can be: **S(lname)=SNSDEF/LASER,var_1,var_2,power,shuter**

Output Formats:

can be: **S(lname)=SNSDEF/LASER,var_1,var_2,power,shuter**

or: **SA(lname)=SNSDEF/OFFSET,ox,oy,oz,sns_size,art_size, $ art_form,art_label,ax,ay,az,datetime**

Where:

var_1 can be: **FIXED**
or: **INDEX**

var_2 can be: **CART,dx,dy,dz,ti,tj,tk**
or: **POL,tilt,rot,ti,tj,tk**
or: **VEC,i,j,k,ti,tj,tk**

art_form is a positive real number representing the measured form error of the artifact during sensor calibration.

art_label is the label of the artifact feature actual or feature nominal used for calibration and specified in the CALIB statement.

art_size is a positive real number representing the measured size of the artifact used for sensor calibration.

ax,ay,az are the Cartesian coordinates of the calibration artifact in machine coordinates.

CART signifies that the sensor location is to be defined in Cartesian coordinates.

datetime is the date and time at completion of the last calibration of the sensor in the format 'YYYY/MM/DD HH:mm:SS.sss', where YYYY is the year, MM is the number of the month, and DD is the day within the month, HH is hours from 0 to 23, mm is minutes, SS is seconds, and sss is milliseconds.

dx,dy,dz are the distances in X, Y, and Z between the sensor's axis orientation point, and the sensor reference point in probe coordinates.

FIXED signifies that a fixed sensor is being defined.

i,j,k is the unit vector of the projected beam's axis.

INDEX signifies that an indexable sensor is being defined.

lname is an alphanumeric label name assigned to the sensor.

LASER signifies that a laser sensor is being defined.

OFFSET signifies the actual Cartesian coordinates in the machine coordinate system of the calibrated sensor as well as the sensor's actual diameter.

ox,oy,oz is the offset of the sensor center from the sensor mount point in the sensor mount coordinate system. Refer to (Figure B.78 — Offset tip outputs).

POL signifies that the sensor location is to be defined in polar coordinates.

power is a real number representing the required power setting in watts.

rot is the sensor's angle of rotation with respect to the probe coordinate system's positive X axis in the XY plane of the probe coordinate system.

shuter is a real number representing the shutter speed in milliseconds.

sns_size is a positive real number representing the effective size of the sensor.

ti,tj,tk is the unit vector of the sensor mount socket's axis.

tilt is the sensor's angle of tilt with respect to the fully extended position where the tilt angle is zero.

VEC signifies that the sensor location is to be defined in unit vectors.

The SNSDEF/LASER statement is passed to the output file by execution of the OUTPUT statement.

Note: The parameters in the SENSDEF(input format 1) through SENSDEF(input format 6) statements are primarily used to document and communicate the sensor definition to other applications such as offline programming and can be used for error checking.

6.174 SNSDEF (input format 4)

Function: Defines an infrared sensor used by the DME in making measurements, and assigns a label to it. Several formats are used to define sensor types.

Input Formats:

can be: **S(lname)=SNSDEF/INFRED,var_1,var_2,fovx,fovy,freq,dwell**

Output Formats:

can be: **S(lname)=SNSDEF/INFRED,var_1,var_2,fovx,fovy,freq,dwell**

or: **SA(lname)=SNSDEF/OFFSET,ox,oy,oz,sns_size,art_size, $ art_form,art_label,ax,ay,az,datetime**

Where:

var_1 can be: **FIXED**
or: **INDEX**

var_2 can be: **CART,dx,dy,dz,ti,tj,tk**
or: **POL,tilt,rot,ti,tj,tk**
or: **VEC,i,j,k,ti,tj,tk**

art_form is a positive real number representing the measured form error of the artifact during sensor calibration.

art_label is the label of the artifact feature actual or feature nominal used for calibration and specified in the CALIB statement.

art_size is a positive real number representing the measured size of the artifact used for sensor calibration.

ax,ay,az are the Cartesian coordinates of the calibration artifact in machine coordinates.

CART signifies that the sensor location is to be defined in Cartesian coordinates.

datetime is the date and time at completion of the last calibration of the sensor in the format 'YYYY/MM/DD HH:mm:SS.sss', where YYYY is the year, MM is the number of the month, and DD is the day within the month, HH is hours from 0 to 23, mm is minutes, SS is seconds, and sss is milliseconds.

dwell is a positive real number representing the dwell time value in seconds.

dx,dy,dz are the distances in X, Y, and Z between the sensor's axis orientation point, and the sensor reference point in probe coordinates.

FIXED signifies that a fixed sensor is being defined.

fovx is a real number representing the field of view size in the X axis direction.

fovy is a real number representing the field of view size in the Y axis direction.

freq is a positive real number representing the frequency value in milliseconds.

i,j,k is the unit vector of the optical axis.

INDEX signifies that an indexable sensor is being defined.

INFRED signifies that an infrared sensor is being defined.

lname is an alphanumeric label name assigned to the sensor.

OFFSET signifies the actual Cartesian coordinates in the machine coordinate system of the calibrated sensor as well as the sensor's actual diameter.

ox,oy,oz is the offset of the sensor center from the sensor mount point in the sensor mount coordinate system. Refer to (Figure B.78 — Offset tip outputs).

POL signifies that the sensor location is to be defined in polar coordinates.

rot is the sensor's angle of rotation with respect to the probe coordinate system's positive X axis in the XY plane of the probe coordinate system.

sns_size is a positive real number representing the effective size of the sensor.

ti,tj,tk is the unit vector of the sensor mount socket's axis.

tilt is the sensor's angle of tilt with respect to the fully extended position where the tilt angle is zero.

VEC signifies that the sensor location is to be defined in unit vectors.

The SNSDEF/INFRED statement is passed to the output file by execution of the OUTPUT statement.

Note: The parameters in the SENSDEF(input format 1) through SENSDEF(input format 6) statements are primarily used to document and communicate the sensor definition to other applications such as offline programming and can be used for error checking.

6.175 SNSDEF (input format 5)

Function: Defines a non-contact sensor used by the DME in making measurements, and assigns a label to it. Several formats are used to define sensor types.

Input Formats:

can be: **S(lname)=SNSDEF/NONCON,var_1,var_2,proben**

Output Formats:

can be: **S(lname)=SNSDEF/NONCON,var_1,var_2,proben**

or: **SA(lname)=SNSDEF/OFFSET,ox,oy,oz,sns_size,art_size, $ art_form,art_label,ax,ay,az,datetime**

Where:

var_1 can be: **FIXED**
or: **INDEX**

var_2 can be: **CART,dx,dy,dz,ti,tj,tk**
or: **POL,tilt,rot,ti,tj,tk**
or: **VEC,i,j,k,ti,tj,tk**

art_form is a positive real number representing the measured form error of the artifact during sensor calibration.

art_label is the label of the artifact feature actual or feature nominal used for calibration and specified in the CALIB statement.

art_size is a positive real number representing the measured size of the artifact used for sensor calibration.

ax,ay,az are the Cartesian coordinates of the calibration artifact in machine coordinates.

CART signifies that the sensor location is to be defined in Cartesian coordinates.

datetime is the date and time at completion of the last calibration of the sensor in the format 'YYYY/MM/DD HH:mm:SS.sss', where YYYY is the year, MM is the number of the month, and DD is the day within the month, HH is hours from 0 to 23, mm is minutes, SS is seconds, and sss is milliseconds.

dx,dy,dz are the distances in X, Y, and Z between the sensor's axis orientation point, and the sensor reference point in probe coordinates.

FIXED signifies that a fixed sensor is being defined.

i,j,k is the unit vector of the sensor axis.

INDEX signifies that an indexable sensor is being defined.

lname is an alphanumeric label name assigned to the sensor.

NONCON signifies that a non-contact sensor is being defined. For example, an electrical capacitance probe having several sensors.

OFFSET signifies the actual Cartesian coordinates in the machine coordinate system of the calibrated sensor as well as the sensor's actual diameter.

ox,oy,oz is the offset of the sensor center from the sensor mount point in the sensor mount coordinate system. Refer to (Figure B.78 — Offset tip outputs).

POL signifies that the sensor location is to be defined in polar coordinates.

proben is a positive integer representing the number of sensors, or probes, that this device provides.

rot is the sensor's angle of rotation with respect to the probe coordinate system's positive X axis in the XY plane of the probe coordinate system.

sns_size is a positive real number representing the effective size of the sensor.

ti,tj,tk is the unit vector of the sensor mount socket's axis.

tilt is the sensor's angle of tilt with respect to the fully extended position where the tilt angle is zero.

VEC signifies that the sensor location is to be defined in unit vectors.

The SNSDEF/NONCON statement is passed to the output file by execution of the OUTPUT statement.

Note: The parameters in the SENSDEF(input format 1) through SENSDEF(input format 6) statements are primarily used to document and communicate the sensor definition to other applications such as offline programming and can be used for error checking.

6.176 SNSDEF (input format 6)

Function: Builds a sensor from sensor components (such as wrists, extensions, reference mounts, sensors, and sensor component groups) used by the DME in making measurements, and assigns a label to it.

Input Formats:

can be: **S(lname)=SNSDEF/BUILD var_1,var_2**

Output Formats:

can be: **S(lname)=SNSDEF/BUILD var_1,var_2**

or: **SA(lname)=SNSDEF/OFFSET,ox,oy,oz,sns_size,art_size, $ art_form,art_label,ax,ay,az,datetime**

Where:

var_1 can be: **,SG(lname1) var_3**
or: **,SW(lname2) var_4**
or: **,SX(lname3) var_3**
or: **,RM(lname4) var_3**

var_2 can be: **SS(lname5)**
or: **SGS(lname6)**

var_3 can be: **var_1**
or: **does not exist**

var_4 can be: **,'anglename',angle var_5**
or: **var_1**
or: **does not exist**

var_5 can be: **var_4**
or: **does not exist**

angle is the angle of rotation for the axis identified by 'anglename'.

'anglename' is the name identifying the angle of rotation, enclosed with apostrophes.

art_form is a positive real number representing the measured form error of the artifact during sensor calibration.

art_label is the label of the artifact feature actual or feature nominal used for calibration and specified in the CALIB statement.

art_size is a positive real number representing the measured size of the artifact used for sensor calibration.

ax,ay,az are the Cartesian coordinates of the calibration artifact in machine coordinates.

BUILD signifies that a sensor is to be defined by assembeling the named components.

datetime is the date and time at completion of the last calibration of the sensor in the format 'YYYY/MM/DD HH:mm:SS.sss', where YYYY is the year, MM is the number of the month, and DD is the day within the month, HH is hours from 0 to 23, mm is minutes, SS is seconds, and sss is milliseconds.

lname is an alphanumeric label name assigned to the sensor.

OFFSET signifies the actual Cartesian coordinates in the machine coordinate system of the calibrated sensor as well as the sensor's actual diameter.

ox,oy,oz is the offset of the sensor center from the sensor mount point in the sensor mount coordinate system. Refer to (Figure B.78 — Offset tip outputs).

RM(lname4) is the label of a previously defined sensor reference mount.

SG(lname1) is the label of a previously defined sensor component group.

SGS(lname6) is the label of a previously defined sensor component group with a sensor probe.

sns_size is a positive real number representing the effective size of the sensor.

SS(lname5) is the label of a previously defined sensor probe.

SW(lname2) is the label of a previously defined sensor wrist.

SX(lname3) is the label of a previously defined sensor extension.

The SNSDEF/BUILD statement is passed to the output file by execution of the OUTPUT statement.

Note 1: Refer to example A.30 and sub-clauses.

Note 2: Refer to example A.30.2 and (Figure A.7 — Multi-probes from multiple ports on the head) for illustration and sample code used to define a sensor using the SNSDEF/BUILD statement.

Note 3: The parameters in the SENSDEF(input format 1) through SENSDEF(input format 6) statements are primarily used to document and communicate the sensor definition to other applications such as offline programming and can be used for error checking.

6.177 SNSET

Function: Specifies and activates sensor settings used on a DME.

Input Formats:

can be: **SNSET/var_1 var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **APPRCH,dist1**
or: **RETRCT,dist1**
or: **SEARCH,dist1**
or: **CLRSRF,var_3**
or: **DEPTH,var_3**
or: **VA(lname1)**
or: **VF(lname2)**
or: **VL(lname3),intnsty**
or: **VW(lname4)**
or: **FOCUSY**
or: **FOCUSN**
or: **SCALEX,n**
or: **SCALEY,n**
or: **MINCON,level**

var_2 can be: **,var_1 var_2**
or: **does not exist**

var_3 can be: **F(lname5) var_4**
or: **FA(lname6) var_4**
or: **DAT(x) var_4**
or: **dist2**
or: **OFF**

var_4 can be: **,dist3**
or: **does not exist**

APPRCH signifies the approach distance. The distance that the probe will begin its measurement taking sequence to the feature being measured.

CLRSRF signifies the clearance distance away from the feature for path-directed moves.

When F(lname5), FA(lname6), or DAT(x) is used, it must reference a previously defined or measured plane that is parallel to the plane in which the feature to be measured lies. Also, CLRSRF and DEPTH are perpendicular to APPRCH, RETRCT, and SEARCH.

CLRSRF restricts the region of permitted movement for a DME during automatic mode. When a feature is measured in automatic mode, the DME will move to the clearance plane by the shortest path (if necessary). It will stay at the clearance plane until it is over the feature to be measured. When it is over the target feature, it can descend to perform the measurements. After the measurement is completed, if the next motion or feature measurement would cause the DME to violate the current clearance region, it must first move to the clearance plane.

DAT(x) is the label of a previously measured and assigned datum (plane) used as a clearance plane, or a depth measuring plane.

DEPTH signifies the distance the probe will penetrate into the feature along the feature's orientation vector (nominal or actual depending on the measurement method). Features to which DEPTH can be applied are as follows:

- Arc
- Circle
- Ellipse
- Centered parallel line

If a cylindrical probe is being used to measure a feature, DEPTH is applied along the probe orientation vector, rather than along the feature's orientation vector. Features to which this can be applied are those listed above in addition to the following:

- GCurve
- GSurf
- Line
- Point (edge)

Note that when F(lname5), FA(lname6), or DAT(x) is used, it must reference a previously defined/measured plane that is parallel to the plane in which the feature to be measured lies. Also, CLRSRF and DEPTH are perpendicular to APPRCH, RETRCT, and SEARCH.

Refer to (Figure B.58 — RMEAS/CPARLN with and without a cylindrical probe) and (Figure B.62 — RMEAS/ELLIPS with and without a cylindrical probe) for an illustration of DEPTH with and without a cylindrical probe.

dist1 is a positive real number representing the distance value assigned.

dist2 is a real number representing the distance value assigned from the feature nominal and is to be measured along its vector.

When utilizing CLRSRF and DEPTH on features such as arcs, circles, ellipses, and a centered parallel line that do not lie in a plane, it is important to note that the 'dist2' is relative to an imaginary tangent plane to one of the aforementioned features. Refer to (Figure B.83 — SENS or distance illustration). Caution should be exercised when dealing with piercing features that do not penetrate normal to the surface. In these cases, GOTO points should be used to guarantee access. Refer to (Figure B.84 — Surface penetration illustration).

dist3 is a real number representing the distance value from the parallel plane and is to be measured along the parallel plane's vector.

When utilizing CLRSRF and DEPTH on features such as arcs, circles, ellipses, and a centered parallel line that do not lie in a plane, it is important to note that F(lname5), FA(lname6), or DAT(x) should be a plane feature and dist3 is relative to this plane. Refer to (Figure B.83 — SENS or distance illustration) and (Figure B.84 — Surface penetration illustration).

F(lname5) is the label of a previously defined feature nominal (plane) used as a clearance plane, or a depth measuring plane.

FA(lname6) is the label of a previously measured or constructed feature actual (plane) used as a clearance plane, or a depth measuring plane.

FOCUSN signifies that automatic focusing is off.

FOCUSY signifies that automatic focusing is on.

intnsty is a real number representing the brightness intensity in percent, for example, 0.75 = 75%.

level is a positive real number representing the real numeric value of the minimum confidence level.

MINCON signifies that a minimum confidence level is being established. If a non-contact sensor cannot achieve a measurement with this minimum level of confidence, an error occurs.

n is a real number representing the value of the scale factor.

OFF signifies that the CLRSRF or DEPTH option is to be turned off.

RETRCT signifies the retract distance. The distance that a probe will retract after a triggering measurement.

SCALEX signifies that a scale factor is assigned to the image in X.

SCALEY signifies that a scale factor is assigned to the image in Y.

SEARCH signifies the search distance. This is the distance that a probe will continue its measurement taking sequence beyond the APPRCH distance. If no point is found, the DME will retract to the point where the search was initiated and flag an error.

VA(lname1) is the label of a previously defined DME specific algorithm.

VF(lname2) is the label of a previously defined video filter.

VL(lname3) is the label of a previously defined video light.

VW(lname4) is the label of a previously defined video window.

The DEPTH setting is used by a CMM when executing a measurement sequence. For features such as arcs, circles, ellipses, and centered parallel line features the nominal definition specifies the surface within which the feature lies. In automatic mode, the probe will penetrate this surface in a direction opposite the feature's orientation vector by the amount specified in the SNSET/DEPTH statement before the CMM executes its automatic measurement routine. In programmed mode, the PTMEAS x, y, z value will be offset by the DEPTH value in a direction opposite the feature's orientation vector. However, if in automatic mode and a cylindrical probe is being used to measure the feature, the probe will, in the case of arcs, circles, ellipses, and centered parallel line features, penetrate the surface within which the feature lies along the probe orientation vector by the amount specified in SNSET/DEPTH. If in programmed mode and a cylindrical probe is being used to measure the feature, the PTMEAS x, y, z value will be offset by the DEPTH value along the probe orientation vector. In the case of GCURVE, GSURF, LINE, and POINT features, the use of a cylindrical probe indicates that the measurement is to be taken with the shaft of the probe and the DEPTH value is to be applied along the probe orientation vector at the feature's approach point(s). The DEPTH distance can be a positive or negative value applied along either the feature orientation vector or the probe orientation vector depending on the currently active probe at measurement time.

APPRCH, RETRCT, SEARCH, CLRSRF, DEPTH, SCALEX, and SCALEY may only appear once within a single SNSET statement. FOCUSY and FOCUSN can only appear once within a single SNSET statement and are mutually exclusive.

SNSET/CLRSRF is only available for automatic mode. It is ignored during program and manual mode.

Refer to (Figure B.81 — Sensor setting illustration using a typical cross section of a hole) or (Figure B.82 — Sensor setting illustration of a typical clearance plane) for an illustration of the sensor settings.

Note: Caution should be exercised when dealing with piercing features that do not penetrate normal to the surface. In these cases, GOTO point(s) should be used to guarantee access.

6.178 SNSGRP

Function: Defines a component group ending with a sensor, and assigns a label to it.

Input Formats:

can be: **SGS(lname)=SNSGRP/BUILD var_1**

Output Formats:

can be: **SGS(lname)=SNSGRP/BUILD var_1**

Where:

var_1 can be: **var_2,SS(lname1)**
or: **var_2,SGS(lname2)**

var_2 can be: **,SG(lname3) var_3**
or: **,SW(lname4) var_3**
or: **,SX(lname5) var_3**
or: **,RM(lname6) var_3**

var_3 can be: **var_2**
or: **does not exist**

lname is an alphanumeric label name assigned to the sensor group.

RM(lname6) is the label of a previously defined sensor reference mount.

SG(lname3) is the label of a previously defined sensor component group.

SGS(lname2) is the label of a previously defined sensor component group with a sensor.

SS(lname1) is the label of a previously defined sensor.

SW(lname4) is the label of a previously defined sensor wrist.

SX(lname5) is the label of a previously defined sensor extension.

The SNSGRP statement is passed to the output file when executed.

Note: Refer to example A.30.1 for sample code used to build a sensor group using the SNSGRP/BUILD statement.

6.179 SNSLCT

Function: Selects the sensor(s) to be used for measurement.

Input Formats:

can be: **SNSLCT/var_1**

Output Formats:

can be: **SNSLCT/var_2**

Where:

var_1 can be: **GSA(lname1) var_3**
or: **var_4 var_5**

var_2 can be: **GSA(lname1)**
or: **var_4 var_6**

var_3 can be: **,F(lname2) var_7**
or: **,FA(lname3) var_7**
or: **,VEC,i,j,k var_7**

var_4 can be: **S(lname4) var_8**
or: **SA(lname5) var_8**
or: **S(lname6) var_9**
or: **SA(lname7) var_9**

var_5 can be: **var_10**
or: **,F(lname2) var_7**
or: **,FA(lname3) var_7**
or: **,VEC,i,j,k var_7**
or: **,PCS,z3,y3,z4**
or: **,HEADCS,z5,y4**
or: **,HEADCS,z5,y4,z6**

var_6 can be: **,SW(lname8),'anglename',ang var_7 var_12 var_6**
or: **does not exist**

var_7 can be: **,FZ,dev**
or: **does not exist**

var_8 can be: **,S(lname9) var_8**
or: **,SA(lname10) var_8**
or: **does not exist**

var_9 can be: **,'desc'**
or: **,tipnum**

var_10 can be: **,SW(lname8) var_11 var_10**
or: **does not exist**

var_11 can be: **,'anglename',ang var_13**
or: **,'anglename',F(lname2) var_7 var_13**
or: **,'anglename',FA(lname3) var_7 var_13**
or: **,'anglename',VEC,i,j,k var_7 var_13**

var_12 can be: **,'anglename',ang var_7 var_12**
or: **does not exist**

var_13 can be: **var_11**
or: **does not exist**

ang is the angle for the wrist axis identified by 'anglename'.

'anglename' is the name of the wrist angle to rotate, enclosed with apostrophes.

'desc' is a string, enclosed with apostrophes, that is the tip description as defined in the SENSOR/MLTPRB statement.

dev is the maximum allowed angular deviation between the target axis and the final sensor orientation in the input format and the actual deviation between the target axis and the final sensor orientation in the output format. The format of 'dev' must be consistent with the current setting for angles, set with the UNITS statement.

F(lname2)
FA(lname3) is the label of an oriented feature, the orientation of which to use for rotation of the wrist. The DME will choose the closest rotation angle for the wrist, as calculated from the nominal wrist definition.

If more than one angle is possible to get the sensor aligned to the feature, the rotation which creates the smallest rotation from the current wrist's angular position is performed. Refer to (Figure B.80 — SNSLCT, sensor selection illustration).

FZ signifies that the value for the allowed angular deviation is to follow.

GSA(lname1) is the label of a previously defined group of calibrated sensors.

HEADCS signifies that the orientation of the probe is specified by two or three angles of rotation in probe head coordinates.

i,j,k is the direction vector to use for rotation of the wrist. The DME will choose the closest rotation angles for the wrist, as calculated from the nominal wrist definition.

If more than one angle is possible to get the sensor aligned to the vector, the rotation which creates the smallest rotation from the current wrist's angular position is performed. Refer to (Figure B.80 — SNSLCT, sensor selection illustration).

PCS signifies that the orientation of the probe is specified by three angles of rotation in part coordinates.

S(lname4) is the label of a previously defined sensor.

S(lname6) is the label of a previously defined sensor with multiple tips.

S(lname9) is the label of a previously defined sensor(s) used for machine vision systems when more than one sensor is used to take measurements.

SA(lname5) is the label of the previously calibrated sensor.

SA(lname7) is the label of a previously calibrated sensor with multiple tips.

SA(lname10) is the label of a previously calibrated sensor(s) used for machine vision systems when more than one sensor is used to take measurements.

SW(lname8) is the label of a wrist to be rotated. The wrist must be attached to the sensor specified with S(lname) or SA(lname).

tipnum is a positive integer representing the tip number as defined in the SENSOR/MLTPRB statement.

VEC signifies that a direction vector to use for rotation of the wrist is to follow.

z3,y3,z4 are the Euler angles of rotation which define the axis system of the probe (see note below).

z5,y4 are the angles of rotation of the 2 axis probe in probe coordinates (as described below).

z5,y4,z6 are the angles of rotation of the 3 axis probe in probe coordinates (as described below).

PCS signifies that the axis system is rotated relative to the current part coordinate system: the axis system is rotated around Z by the first angle, then around the newly created Y axis, and finally around the newly created Z (which is the ZYZ Euler angle convention).

The resultant head orientation will be such that the active probe tip shaft (or the main direction of an optical sensor) is aligned with the Z axis pointing towards the sensor. The alignment of the tool with respect to the X and Y axes will be tool dependent, e.g. to align a laser plane.

When specifying HEADCS, the number of angles of rotation specified will match the number of physical axes of the probe head. The angles are always specified as absolute values, local to the probe head coordinates.

The SNSLCT statement is passed to the output file when executed.

Note: Instances of the simplest possible use of SNSLCT may be found in examples A.7, A.9, A.19.2, A.28, and A.31. Instances of adjusting wrists using SNSLCT may be found in examples A.30.1, A.30.5, and A.30.7. Instances of using SELECT to select a probe on a multiprobe sensor may be found in example A.30.4.

6.180 SNSMNT

Function: Defines the relationship of the sensor coordinate system to the machine coordinate system. This function establishes a sensor coordinate system for the SNSDEF statement.

Input Formats:

can be: **SNSMNT/XVEC,xi,xj,xk,ZVEC,zi,zj,zk,MNTLEN,dx,dy,dz**

Output Formats:

can be: **None**

Where:

dx,dy,dz is the distance in machine coordinates between the DME manufacturer's sensor reference point and the origin of the sensor coordinate system

MNTLEN signifies the nominal distance in machine coordinates between the DME manufacturer's sensor reference point and the origin of the sensor coordinate system is to follow.

xi,xj,xk is the vector in machine coordinates along which the sensor coordinate system X axis is aligned.

XVEC signifies that the specification of the X axis of the sensor coordinate system in the machine coordinate system is to follow.

zi,zj,zk is the vector in machine coordinates along which the sensor coordinate system Z axis is aligned.

ZVEC signifies that the specification of the Z axis of the sensor coordinate system in the machine coordinate system is to follow.

Note: Refer to (Figure B.79 — Sensor mount axis illustration), clause A.30 and sub-clauses.

6.181 TECOMP

Function: Causes the machine's and part's temperature compensation to be turned on or off at the DME. It can also specify the part's thermal expansion coefficient.

Input Formats:

can be: **TECOMP/var_1**

Output Formats:

can be: **TECOMP/var_1**

Where:

var_1 can be: **MACH,ON**
or: **MACH,OFF**
or: **PART,ON var_2,tmpexp var_3 var_4**
or: **PART,ON var_2,tmpexp var_3,ALL**
or: **PART,OFF**

var_2 can be: **,DA(lname)**
or: **,OFFSET,xoff,yoff,zoff**
or: **does not exist**

var_3 can be: **,tmpexpunc**
or: **does not exist**

var_4 can be: **,'tempsns' var_5**

var_5 can be: **var_4**
or: **does not exist**

ALL signifies that all part sensors are used.

DA(lname) is the coordinate system used as the thermal datum by the DME for temperature compensation.

MACH signifies that machine temperature compensation parameters are being set.

OFF signifies that temperature compensation is to be turned off at the DME when minor word MACH is used. It signifies that part temperature compensation is to be turned off at the DME when the minor word PART is used.

OFFSET signifies that three offset values follow.

ON signifies that temperature compensation is to be turned on at the DME when minor word MACH is used. It signifies that part temperature compensation is to be turned on at the DME when the minor word PART is used.

PART signifies that part temperature compensation parameters are being set.

tmpexp is a real number representing the thermal expansion coefficient of the part specified in current system temperature units.

tmpexpunc is a real number representing the uncertainty in the thermal expansion coefficient of the part. It is specified in current system temperature units and indicates the half-width of a symmetric zone about tmpexp within which the true thermal expansion coefficient is believed to lie.

tmpsns is a character string representing the valid part sensor name(s) to be used by the DME.

xoff is a real number representing the X offset from the current coordinate system origin.

yoff is a real number representing the Y offset from the current coordinate system origin.

zoff is a real number representing the Z offset from the current coordinate system origin.

The TECOMP statement is passed to the output file when executed.

6.182 TEXT

Function: Specifies various forms of text to be sent to the operator and/or, the output file.

Input Formats:

can be: **TEXT/var_1,'text'**

Output Formats:

can be: **TEXT/var_4,'text'**

Where:

var_1 can be: **OPER**
or: **OUTFIL**
or: **MAN**
or: **QUERY,(lname1),length,var_2,var_3**

var_2 can be: **ALPHA**
or: **NUMERIC**
or: **PRNTCHAR**

var_3 can be: **LEFT**
or: **RIGHT**

var_4 can be: **OUTFIL**
or: **RES,(lname1),length,var_2,var_3**

(lname1) is a label name enclosed in parentheses.

ALPHA signifies that the response should consist of all alpha characters.

PRNTCHAR signifies that the response will be any printable UTF8 characters.

LEFT signifies a left justified response.

length is a positive integer representing the field length allowed for the response. The field length is the same for both the TEXT/QUERY and the corresponding TEXT/RES statements.

MAN signifies that the message will be output to the operator, but only if the DME is in manual mode (if MODE/MAN has been issued or if the DME has automatically defaulted to manual during a CALIB, GOTARG, MEAS or RMEAS) and waits for operator acknowlegment.

NUMERIC signifies that the response should consist of character representations of real numeric data.

OPER signifies that the message will be printed on the screen during program execution and waits for operator acknowlegment. The message can be any printable UTF8 character.

OUTFIL signifies that the message will be inserted into the output file. The message is inserted in the output at the point at which the TEXT/OUTFIL statement occurred in the input program.

QUERY signifies that a response is required from the operator. The response will be output into the output file with the same label as the query.

The output format for the TEXT/QUERY statement is TEXT/RES.

RIGHT signifies a right justified response.

RES signifies the response to the TEXT/QUERY statement.

'text' is the text string consisting of printable UTF8 characters, enclosed with apostrophes.

Use two apostrophes, one before the one required, when an apostrophe is required within a text string.

Refer to clause 5.1.8.1 for use of the TEXT/QUERY and TEXT/RES statements.

The TEXT/OUTFIL and TEXT/RES statements are passed to the output file when executed.

Note: Refer to example A.32.

6.183 THLDEF

Function: Defines an automatic tool or sensor holder/changer in terms of the sensors it carries, and assigns a label to it.

Input Formats:

can be: **TH(lname)=THLDEF/var_1 var_2**

Output Formats:

can be: **None**

Where:

var_1 can be: **S(lname1),n**
or: **SS(lname2),n**
or: **SGS(lname3),n**
or: **SG(lname4),n**
or: **SW(lname5),n**
or: **SX(lname6),n**
or: **RM(lname7),n**

var_2 can be: **,var_1**
or: **does not exist**

lname is an alphanumeric label name assigned to the sensor holder being defined.

n is a positive integer identifying the sensor holder pocket which the component occupies.

RM(lname7) is the label of a previously defined sensor reference mount.

S(lname1) is the label of a previously defined sensor.

SG(lname4) is the label of a previously defined sensor component group not ending with a sensor.

SGS(lname3) is the label of a previously defined sensor component group ending with a sensor.

SS(lname2) is the label of a previously defined sensor component that is capable of making measurement.

SW(lname5) is the label of a previously defined sensor wrist.

SX(lname6) is the label of a previously defined sensor extension.

Note 1: This statement associates a set of sensors with a sensor holder. There are no restrictions on the type or number of sensors in the sensor holder.

Note 2: Refer to example A.30.6 and (Figure A.11 — Sensor components in an automated changer rack) for an illustration and sample code used to define a tool changer configuration and the sensors that it contains.

6.184 TOL/ANGL

Function: Specifies an angular tolerance, and assigns a label to it. This is a direct tolerance.

Input Formats:

can be: **T(lname)=TOL/ANGL,lotol,uptol**

Output Formats:

can be: **TA(lname)=TOL/ANGL,dev,var_1**

or: **T(lname)=TOL/ANGL,lotol,uptol**

Where:

var_1 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

ANGL signifies that the tolerance is an angular tolerance.

combinedunc is a real number representing the combined uncertainty of the measurement.

dev is a real number representing the arithmetic difference between the actual value and the nominal value. It is positive when the nominal is less than the actual, and negative when the nominal is greater than the actual.

INTOL signifies the actual is within tolerance, that is when uptol >= dev >= lotol with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lotol is the signed lower tolerance value assigned to the angle.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol is the signed upper tolerance value assigned to the angle.

The TOL/ANGL statement is passed to the output file by execution of the OUTPUT statement.

Note: If an angular tolerance such as 30 degrees +.5 degrees, -.4 degrees is given, then uptol = +.5 and lotol = -.4. Unsigned lotol and uptol values are assumed positive. Refer to (Figure B.1 —Angle tolerance description) . This tolerance is typically applied to a single feature such as FEAT/CONE.

6.185 TOL/ANGLB

Function: Specifies an angular tolerance for an angle of a given nominal size and assigns a label to it. This is a direct tolerance.

Input Formats:

can be: **T(lname)=TOL/ANGLB,ang,lotol,uptol var_1**

Output Formats:

can be: **TA(lname)=TOL/ANGLB,ang,var_2 var_1**

or: **T(lname)=TOL/ANGLB,ang,lotol,uptol**

Where:

var_1 can be: **,XYPLAN**
or: **,YZPLAN**
or: **,ZXPLAN**
or: **,VEC,i,j,k**
or: **does not exist**

var_2 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

ang is the nominal value of an angle in the input format and either the nominal value or the actual value in the output format.

ANGLB signifies the value and tolerance are applied to the angle between two features.

combinedunc is a real number representing the combined uncertainty of the measurement.

INTOL signifies the actual is within tolerance, that is when uptol >= (ang_{actual} – $ang_{nominal}$) >=lotol with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lotol is the signed lower tolerance value assigned to the angle.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol is the signed upper tolerance value assigned to the angle.

VEC,i,j,k signifies the angle between will be calculated in a plane with a normal specified by i,j,k.

XYPLAN signifies the angle between will be calculated in the XY plane of the current part coordinate system.

YZPLAN signifies the angle between will be calculated in the YZ plane of the current part coordinate system.

ZXPLAN signifies the angle between will be calculated in the ZX plane of the current part coordinate system.

The TOL/ANGLB statement is passed to the output file by execution of the OUTPUT statement.

Note 1: If an angle and tolerance such as 30 degrees +.5 degrees, -.4 degrees are given, then uptol = +.5 and lotol = -.4. Unsigned lotol and uptol values are assumed positive. Refer to (Figure B.1 —Angle tolerance description).

Note 2: Defines the nominal angle to be used between two line reducible or plane reducible features.

6.186 TOL/ANGLR

Function: Specifies an angularity tolerance, and assigns a label to it. This is an orientation tolerance.

Input Formats:

can be: **T(lname)=TOL/ANGLR,ang,tolzon var_1 var_2 var_3 var_4 var_5**

Output Formats:

can be: **TA(lname)=TOL/ANGLR,tolzon,var_6 var_1,lim var_2 var_3 var_4 var_5**

or: **T(lname)=TOL/ANGLR,ang,tolzon var_1 var_2 var_3 var_4 var_5**

Where:

var_1 can be: **,MMC var_7**
or: **,LMC var_7**
or: **,RFS**
or: **does not exist**

var_2 can be: **,DAT(x) var_1**
or: **,F(lname1)**
or: **,FA(lname2) var_1**

var_3 can be: **,DAT(x) var_1**
or: **,F(lname1)**
or: **,FA(lname2) var_1**
or: **does not exist**

var_4 can be: **,TANGPL**
or: **,PARPLN**
or: **does not exist**

var_5 can be: **,XAXIS**
or: **,YAXIS**
or: **,ZAXIS**
or: **,VEC,i,j,k**
or: **does not exist**

var_6 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_7 can be: **,MAX,maxtol**
or: **does not exist**

ang is the nominal value of an angle in the current primary datum plane relative to the specified datum.

ANGLR signifies angularity tolerance.

combinedunc is a real number representing the combined uncertainty of the measurement.

DAT(x) is a datum to be used, or used as a reference.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of the feature actual to be used as a reference.

i,j,k is the direction vector along which the view dependent parallel plane tolerance zone width is to be applied.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname	is an alphanumeric label name assigned to the tolerance.
lim	is a real number representing the sum of the tolerance zone plus any gain from MMC or LMC.
LMC	signifies that a least material condition is applied.
MAX	signifies that a maximum allowed bonus tolerance is to be used.
maxtol	is a real number representing the maximum allowed bonus tolerance value to be used.
meanerror	is a real number representing the mean error of the measurement.
MMC	signifies that a maximum material condition is applied.
OUTOL	signifies the actual is out of tolerance with no consideration made to measurement uncertainty.
PARPLN	signifies a tolerance zone defined by two parallel planes applied to axial features such as cylinders.
REQUNCERT	signifies that uncertainty evaluation data was requested but is not available.
RFS	signifies that a regardless of feature size condition is applied.
RULEINTOL	signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.
RULEOUTOL	signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.
RULEUNDET	signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.
RULEUNSUP	signifies that the specified decision rule is not supported and that the uncertainty data will follow.
TANGPL	signifies that a tangent plane is established by contacting the high points of a surface.
tolzon	is a positive real number representing the width of the tolerance zone defined by two parallel lines within which all points of the feature lie, or the distance between parallel planes within which the center plane of the feature lies, or the diameter of a cylindrical tolerance zone within which the axis of the feature lies. This is a specified value in the input format and either a specified or actual value in the output format.
UNCERT	signifies that no decision rule was stipulated and that the uncertainty data will follow.
VEC	signifies that the view dependent parallel plane tolerance zone width is to be applied along the vector specified with i,j,k.
XAXIS	signifies that the view dependent parallel plane tolerance zone width is to be applied along the X axis.
YAXIS	signifies that the view dependent parallel plane tolerance zone width is to be applied along the Y axis.
ZAXIS	signifies that the view dependent parallel plane tolerance zone width is to be applied along the Z axis.

When var_5 does not exist and the tolerance is being applied to a line-reducible feature, the tolerance zone is cylindrical.

MMC and LMC can be applied to a center plane or axis. Refer to (Figure B.1 —Angle tolerance description).

Projected tolerance zones per ASME Y14.5M-1994 can be supported by bounding the tolerance using the DMIS BOUND statement. Refer to clauses 5.3.2.7.2 and 5.3.2.7.6.

The TOL/ANGLR statement is passed to the output file by execution of the OUTPUT statement.

6.187 TOL/ANGLWRT

Function: Specifies an anglular tolerance on an angle of given nominal size with respect to a previously defined feature or datum and assigns a label to it. This is a direct tolerance.

Input Formats:

can be: **T(lname)=TOL/ANGLWRT,ang,lotol,uptol,var_1 var_2**

Output Formats:

can be: **TA(lname)=TOL/ANGLWRT,ang,var_3,var_1 var_2**

or: **T(lname)=TOL/ANGLWRT,ang,lotol,uptol,var_1 var_2**

Where:

var_1 can be: **DAT(x)**
or: **F(lname1)**
or: **FA(lname2)**

var_2 can be: **,XYPLAN**
or: **,YZPLAN**
or: **,ZXPLAN**
or: **,VEC,i,j,k**
or: **does not exist**

var_3 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

ang is the nominal value of an angle in the input format and the actual value in the output format.

ANGLWRT signifies the value and tolerance are applied to the angle between a feature with respect to a feature or datum.

combinedunc is a real number representing the combined uncertainty of the measurement.

F(lname1) is the label of a feature nominal to be used as a reference in which the distance with respect to will be calculated.

FA(lname2) is the label of a feature actual to be used as a reference in which the distance with respect to will be calculated.

INTOL signifies the actual is within tolerance, that is when uptol >= (angactual - angnominal) >=lotol with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lotol is the signed lower tolerance value assigned to the angle.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol is the signed upper tolerance value assigned to the angle.

VEC,i,j,k signifies the angle between will be calculated in a plane with a normal specified by i,j,k.

XYPLAN signifies the angle between will be calculated in the XY plane of the current part coordinate system.

YZPLAN signifies the angle between will be calculated in the YZ plane of the current part coordinate system.

ZXPLAN signifies the angle between will be calculated in the ZX plane of the current part coordinate system.

Unlike TOL/ANGLB, the use of the TOL/ANGLWRT statement is applied to a single feature.

The TOL/ANGLWRT statement is passed to the output file by execution of the OUTPUT statement.

Note 1: If an angle and tolerance such as 30 degrees +.5 degrees, -.4 degrees are given, then uptol = +.5 and lotol = -.4. Unsigned lotol and uptol values are assumed positive. Refer to (Figure B.1 —Angle tolerance description).

Note 2: Defines the nominal angle to be used between two line reducible or plane reducible features.

6.188 TOL/CIRLTY

Function: Specifies a circularity tolerance, and assigns a label to it. This is a tolerance of form.

Input Formats:

can be: **T(lname)=TOL/CIRLTY,tolzon**

Output Formats:

can be: **TA(lname)=TOL/CIRLTY,tolzon,var_1**

or: **T(lname)=TOL/CIRLTY,tolzon**

Where:

var_1 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

CIRLTY signifies circularity tolerance.

combinedunc is a real number representing the combined uncertainty of the measurement.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

tolzon is a positive real number representing the width of the tolerance zone bounded by two concentric circles within which all elements of the surface of the feature lie. This is a specified value in the input format and either a specified or actual value in the output format.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

The TOL/CIRLTY statement is passed to the output file by execution of the OUTPUT statement.

6.189 TOL/COMPOS

Function: Specifies a composite positional tolerance for use with patterns, and assigns a label to it. This is a location tolerance.

Input Formats:

can be: **T(lname)=TOL/COMPOS,PATERN,tolzon1 var_1,FEATUR,var_2**

Output Formats:

can be: **TA(lname)=TOL/COMPOS,PATERN,tolzon1,var_3,lim1,var_4 $**
,FEATUR,tolzon2,var_3,var_5

or: **T(lname)=TOL/COMPOS,PATERN,tolzon1 var_1,FEATUR,var_2**

Where:

var_1 can be: **,var_4**
or: **,var_6,var_4**

var_2 can be: **tolzon2**
or: **tolzon2,var_4**
or: **tolzon2,var_6**
or: **tolzon2,var_6,var_4**

var_3 can be: **var_8**
or: **var_8,var_6**

var_4 can be: **var_7**
or: **var_7,var_7**
or: **var_7,var_7,var_7**

var_5 can be: **lim2**
or: **lim2,var_4**

var_6 can be: **MMC**
or: **LMC**
or: **RFS**

var_7 can be: **DAT(x)**
or: **DAT(x),var_5**
or: **F(lname1)**
or: **FA(lname2)**
or: **FA(lname2),var_5**

var_8 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

combinedunc is a real number representing the combined uncertainty of the measurement.

COMPOS signifies a composite position tolerance for use with patterns.

DAT(x) is a datum to be used as a reference.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of a feature actual to be used as a reference.

FEATUR signifies the Feature Relating Tolerance Zone Framework (FRTZF).

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lim1 is a real number representing the sum of the tolerance zone plus any gain from MMC or LMC, for the Pattern locating Tolerance Zone Framework.

lim2 is a real number representing the sum of the tolerance zone plus any gain from MMC or LMC, for the Feature Relating Tolerance Zone Framework.

LMC signifies that a least material condition is applied.

meanerror is a real number representing the mean error of the measurement.

MMC signifies that a maximum material condition is applied.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

PATERN signifies the Pattern Locating Tolerance Zone Framework (PLTZF).

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RFS signifies that a regardless of feature size condition is applied.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

tolzon1 is a positive real number representing the size of the tolerance zone for the Pattern Locating Tolerance Zone Framework. This is the specified value in the input format and an actual value in the output format.

tolzon2 is a real number representing the size of the tolerance zone for the Feature Relating Tolerance Zone Framework. This is the specified value in the input format and an actual value in the output format.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

The feature being toleranced must be a pattern, for example, FEAT/PATERN.

The material condition modifiers in var_6 signify whether MMC or LMC is applied to the feature.

The TOL/COMPOS statement is passed to the output file by execution of the OUTPUT statement.

6.190 TOL/CONCEN

Function: Specifies a concentricity tolerance, and assigns a label to it. This is a location tolerance.

Input Formats:

can be: **T(lname)=TOL/CONCEN,tolzon,var_1**

Output Formats:

can be: **TA(lname)=TOL/CONCEN,tolzon,var_2,var_1**

or: **T(lname)=TOL/CONCEN,tolzon,var_1**

Where:

var_1 can be: **DAT(x)**
or: **F(lname1)**
or: **FA(lname2)**

var_2 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

combinedunc is a real number representing the combined uncertainty of the measurement.

CONCEN signifies concentricity tolerance.

DAT(x) is the datum to be used as a reference.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of a feature actual to be used as a reference.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

tolzon is a positive real number representing the diameter of the cylindrical tolerance zone in which the axis of the feature lies. This is a specified value in the input format and either a specified or actual value in the output format.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

The TOL/CONCEN statement is passed to the output file by execution of the OUTPUT statement

Note: A concentricity tolerance specifies a cylindrical (or spherical) tolerance zone whose datum axis (or center point) within which the feature's median points must lie. It applies on a regardless of feature size basis only.

6.191 TOL/CORTOL

Function: Specifies bilateral positional tolerancing of features in Cartesian or polar coordinates relative to the current coordinate system at the time of usage, and assigns a label to it.

Input Formats:

can be: **T(lname)=TOL/CORTOL,var_1**

Output Formats:

can be: **TA(lname)=TOL/CORTOL,var_2,dev,var_3**

or: **T(lname)=TOL/CORTOL,var_1**

Where:

var_1 can be: **XAXIS,lotol,uptol var_4**
or: **YAXIS,lotol,uptol var_4**
or: **ZAXIS,lotol,uptol var_4**
or: **RADIAL,lotol,uptol**
or: **ANGLE,lotol,uptol**

var_2 can be: **XAXIS var_4**
or: **YAXIS var_4**
or: **ZAXIS var_4**
or: **RADIAL**
or: **ANGLE**

var_3 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_4 can be: **,AXIAL,var_5**
or: **does not exist**

var_5 can be: **F(lname1) var_6**
or: **FA(lname1) var_6**

var_6 can be: **,dist**
or: **does not exist**

ANGLE signifies that the polar coordinate method is used to tolerance the angular position.

AXIAL signifies that the feature being toleranced must be capable of being reduced to a line/axial feature.

combinedunc is a real number representing the combined uncertainty of the measurement.

CORTOL signifies bilateral positional tolerancing.

dev is a real number representing the arithmetic difference between the actual value and the nominal value. It is positive when the nominal value is less than the actual value, and negative when the nominal value is greater than the actual value.

dist is the distance along the normal of qualifying nominal or actual feature (creating a parallel plane) that intersects the line/axial feature.

F(lname1) is the label of the previously defined plane reducible feature nominal that qualifies the feature by intersecting the feature being toleranced.

FA(lname1) is the label of the plane reducible feature actual that qualifies the feature by intersecting the feature being toleranced.

INTOL signifies the actual is within tolerance, that is when uptol >= dev >= lotol with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lotol is the signed lower tolerance value assigned to the angle.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance, that is when dev > uptol or dev < lotol with no consideration made to measurement uncertainty.

RADIAL signifies that the polar coordinate method is used to tolerance the radial position.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol is the signed upper tolerance value assigned to the angle.

XAXIS signifies that the rectangular coordinate method is to be used to tolerance the position along the X axis.

YAXIS signifies that the rectangular coordinate method is to be used to tolerance the position along the Y axis.

ZAXIS signifies that the rectangular coordinate method is to be used to tolerance the position along the Z axis.

When using the rectangular coordinate method, the tolerance zone can be square or rectangular. Note that the feature(s) being toleranced with this method must be defined with Cartesian coordinates. Likewise, when using the polar coordinate method, the feature(s) being toleranced with this method must be defined with polar coordinates.

The TOL/CORTOL statement is passed to the output file by execution of the OUTPUT statement.

6.192 TOL/CPROFL

Function: Specifies a composite profile tolerance of a line, and assigns a label to it. This is a profile tolerance.

Input Formats:

can be: **T(lname)=TOL/CPROFL,lotol1,var_1,lotol2,var_2**

Output Formats:

can be: **TA(lname)=TOL/CPROFL,var_6,var_4,var_8,var_4**

or: **T(lname)=TOL/CPROFL,lotol1,var_1,lotol2,var_2**

Where:

var_1 can be: **uptol1**
or: **uptol1 var_3**

var_2 can be: **uptol2**
or: **uptol2 var_3**

var_3 can be: **var_5**
or: **var_5,AVGDEV**
or: **var_5 var_5**
or: **var_5 var_5,AVGDEV**
or: **var_5 var_5 var_5**
or: **var_5 var_5 var_5,AVGDEV**

var_4 can be: **var_7**
or: **var_7 var_5**
or: **var_7 var_5 var_5**
or: **var_7 var_5 var_5 var_5**

var_5 can be: **,DAT(x)**
or: **,DAT(x) var_9**
or: **,F(lname1)**
or: **,FA(lname2)**
or: **,FA(lname2) var_9**

var_6 can be: **lotol1,uptol1**
or: **AVGDEV,dev1**

var_7 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_8 can be: **lotol2,uptol2**
or: **AVGDEV,dev2**

var_9 can be: **,MMC**
or: **,LMC**
or: **,RFS**

AVGDEV signifies that the average deviation is to be used for the measured feature deviation.

combinedunc is a real number representing the combined uncertainty of the measurement.

CPROFL signifies a composite profile tolerance.

DAT(x) is a datum to be used, or used as a reference.

dev1 is a real number representing the arithmetic difference between the actual value and the nominal value. It is positive when the nominal is less than the actual, and negative when the nominal is greater than the actual.

dev2 is a real number representing the arithmetic difference between the actual value and the nominal value. It is positive when the nominal is less than the actual, and negative when the nominal is greater than the actual.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of the feature actual to be used as a reference.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

LMC signifies that a least material condition is applied.

lotol1 is a real number representing the signed lower tolerance value, which lies inside of the part, for the upper composite profile segment. This is a specified value in the input format and either a specified or actual value in the output format.

lotol2 is a real number representing the signed lower tolerance value, which lies inside of the part, for the lower composite profile segment. This is a specified value in the input format and either a specified or actual value in the output format.

meanerror is a real number representing the mean error of the measurement.

MMC signifies that a maximum material condition is applied.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RFS signifies that a regardless of feature size condition is applied.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol1 is a real number representing the signed upper tolerance value, which lies inside of the part, for the upper composite profile segment. This is a specified value in the input format and either a specified or actual value in the output format.

uptol2 is a real number representing the signed upper tolerance value, which lies inside of the part, for the lower composite profile segment. This is a specified value in the input format and either a specified or actual value in the output format.

When AVGDEV is specified in the input format, "AVGDEV,dev" is required in the output format.

The TOL/CPROFL statement is passed to the output file by execution of the OUTPUT statement.

6.193 TOL/CPROFS

Function: Specifies a composite profile tolerance of a surface, and assigns a label to it. This is a profile tolerance.

Input Formats:

can be: **T(lname)=TOL/CPROFS,lotol1,var_1,lotol2,var_2**

Output Formats:

can be: **TA(lname)=TOL/CPROFS,var_6,var_4,var_8,var_4**

or: **T(lname)=TOL/CPROFS,lotol1,var_1,lotol2,var_2**

Where:

var_1 can be: **uptol1**
or: **uptol1 var_3**

var_2 can be: **uptol2**
or: **uptol2 var_3**

var_3 can be: **var_5**
or: **var_5,AVGDEV**
or: **var_5 var_5**
or: **var_5 var_5,AVGDEV**
or: **var_5 var_5 var_5**
or: **var_5 var_5 var_5,AVGDEV**

var_4 can be: **var_7**
or: **var_7 var_5**
or: **var_7 var_5 var_5**
or: **var_7 var_5 var_5 var_5**

var_5 can be: **,DAT(x)**
or: **,DAT(x) var_9**
or: **,F(lname1)**
or: **,FA(lname2)**
or: **,FA(lname2) var_9**

var_6 can be: **lotol1,uptol1**
or: **AVGDEV,dev1**

var_7 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_8 can be: **lotol2,uptol2**
or: **AVGDEV,dev2**

var_9 can be: **,MMC**
or: **,LMC**
or: **,RFS**

AVGDEV signifies that the average deviation is to be used for the measured feature deviation.

combinedunc is a real number representing the combined uncertainty of the measurement.

CPROFS signifies a composite profile tolerance.

DAT(x) is a datum to be used, or used as a reference.

dev1 is a real number representing the arithmetic difference between the actual value and the nominal value. It is positive when the nominal is less than the actual, and negative when the nominal is greater than the actual.

dev2 is a real number representing the arithmetic difference between the actual value and the nominal value. It is positive when the nominal is less than the actual, and negative when the nominal is greater than the actual.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of the feature actual to be used as a reference.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

LMC signifies that a least material condition is applied.

lotol1 is a real number representing the signed lower tolerance value, which lies inside of the part, for the upper composite profile segment. This is a specified value in the input format and either a specified or actual value in the output format.

lotol2 is a real number representing the signed lower tolerance value, which lies inside of the part, for the lower composite profile segment. This is a specified value in the input format and either a specified or actual value in the output format.

meanerror is a real number representing the mean error of the measurement.

MMC signifies that a maximum material condition is applied.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RFS signifies that a regardless of feature size condition is applied.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol1 is a real number representing the signed upper tolerance value, which lies inside of the part, for the upper composite profile segment. This is a specified value in the input format and either a specified or actual value in the output format.

uptol2 is a real number representing the signed upper tolerance value, which lies inside of the part, for the lower composite profile segment. This is a specified value in the input format and either a specified or actual value in the output format.

When AVGDEV is specified in the input format, "AVGDEV,dev" is required in the output format.

The TOL/CPROFS statement is passed to the output file by execution of the OUTPUT statement.

6.194 TOL/CRNOUT

Function: Specifies a circular runout tolerance, and assigns a label to it. This is a runout tolerance.

Input Formats:

can be: **T(lname)=TOL/CRNOUT,tolzon,var_1**

Output Formats:

can be: **TA(lname)=TOL/CRNOUT,tolzon,var_4,var_2**

or: **T(lname)=TOL/CRNOUT,tolzon,var_1**

Where:

var_1 can be: **DAT(x)**
or: **DAT(x),VEC,i,j,k**
or: **DAT(x) var_3**
or: **DAT(x) var_3,VEC,i,j,k**
or: **DAT(x) var_3 var_3**
or: **DAT(x) var_3 var_3,VEC,i,j,k**

var_2 can be: **DAT(x)**
or: **DAT(x) var_3**
or: **DAT(x) var_3 var_3**

var_3 can be: **,DAT(x)**
or: **,F(lname1)**
or: **,FA(lname2)**
or:

var_4 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

combinedunc is a real number representing the combined uncertainty of the measurement.

VEC,i,j,k signifies the specified direction for the tolerance zone.

CRNOUT signifies circular runout tolerance.

DAT(x) is the datum to be used as an axis, and can be in the form of (x-x) for constructed compound datums.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of the feature actual to be used as a reference.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

tolzon is a positive real number representing the width of the tolerance zone within which a single circular element, for example a circular cross section, lies. This is a specified value in the input format and an actual value in the output format.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

The TOL/CRNOUT statement is passed to the output file by execution of the OUTPUT statement.

Note: Circular runout is applied to circular features. The tolerance is applied independently at any circular cross section as the part is rotated 360 degrees. When applied to features with an axis, circular runout controls the cumulative variations of the circularity and coaxiality. When applied to a plane feature , which is perpendicular to the datum axis, circular runout controls wobble.

6.195 TOL/CYLCTY

Function: Specifies a cylindricity tolerance, and assigns a label to it. This is a tolerance of form.

Input Formats:

can be: **T(lname)=TOL/CYLCTY,tolzon**

Output Formats:

can be: **TA(lname)=TOL/CYLCTY,tolzon,var_1**

or: **T(lname)=TOL/CYLCTY,tolzon**

Where:

var_1 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

combinedunc is a real number representing the combined uncertainty of the measurement.

CYLCTY signifies cylindricity tolerance.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

tolzon is a positive real number representing the width of the tolerance zone bounded by two concentric cylinders within which all elements of the surface of the feature lie. This is a specified value in the input format and either a specified or actual value in the output format.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

The TOL/CYLCTY statement is passed to the output file by execution of the OUTPUT statement.

6.196 TOL/DIAM

Function: Specifies a diameter tolerance, and assigns a label to it. This is a direct tolerance.

Input Formats:

can be: **T(lname)=TOL/DIAM,lotol,uptol var_1 var_2**

Output Formats:

can be: **TA(lname)=TOL/DIAM,dev var_1,var_3 var_4**

or: **T(lname)=TOL/DIAM,lotol,uptol var_1 var_2**

Where:

var_1 can be: **,MAJOR**
or: **,MINOR**
or: **does not exist**

var_2 can be: **,AVG**
or: **,MINMAX**
or: **does not exist**

var_3 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_4 can be: **,MINMAX,mindev,maxdev**
or: **,AVG**
or: **does not exist**

AVG signifies that the average diameter is to be used for the measurement feature size. The average diameter is to be computed as the arithmetic average of several (more than 4) diametrical measurements or calculations.

combinedunc is a real number representing the combined uncertainty of the measurement.

dev is a real number representing the arithmetic difference between the actual value and the nominal value. It is positive when the nominal is less than the actual, and negative when the nominal is greater than the actual.

DIAM signifies that the tolerance is a diameter tolerance.

INTOL signifies the actual is within tolerance, that is when uptol >= dev >= lotol with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lotol is a real number representing the signed lower tolerance value applied to the diameter.

MAJOR signifies that the major diameter of a feature is to be toleranced.

maxdev is a real number representing the most positive deviation from nominal over all points on the measured feature.

meanerror is a real number representing the mean error of the measurement.

mindev is a real number representing the most negative deviation from nominal over all points on the measured feature.

MINMAX signifies that the evaluation tolerance will output the minimum and maximum deviations to the feature nominal.

MINOR signifies that the minor diameter of a feature is to be toleranced.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol is a real number representing the signed upper tolerance value applied to the diameter.

The TOL/DIAM statement is passed to the output file by execution of the OUTPUT statement.

Note 1: If a diameter tolerance such as 5, +1, -2 is given, then uptol = +1 and lotol = -2. Unsigned lotol and uptol values are assumed positive. A diameter tolerance can be applied to a circle, cylinder, ellipse, sphere or torus.

6.197 TOL/DISTB

Function: Specifies a distance and a tolerance and assigns a label to the tolerance. This is a direct tolerance.

Input Formats:

can be: **T(lname)=TOL/DISTB,var_1,var_2 var_3**

Output Formats:

can be: **TA(lname)=TOL/DISTB,var_4,var_1,var_2 var_3**

or: **T(lname)=TOL/DISTB,var_1,var_2 var_3**

Where:

var_1 can be: **NOMINL,dist,lotol,uptol**
or: **LIMIT,lolimt,uplimt**

var_2 can be: **XAXIS**
or: **YAXIS**
or: **ZAXIS**
or: **PT2PT**

var_3 can be: **,AVG**
or: **,MAX**
or: **,MIN**
or: **does not exist**

var_4 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

AVG signifies average or mean distance between two features.

combinedunc is a real number representing the combined uncertainty of the measurement.

dist is a positive real number representing the nominal distance in the input format and the actual distance in the output format.

DISTB signifies the value and tolerance are applied to the distance between two features.

INTOL signifies the actual is within tolerance, that is when var_1 is NOMINL: uptol >= ($dist_{actual}$ – $dist_{nominal}$) >= lotol, or when var_1 is LIMIT: uplimt >= $dist_{actual}$ >= lolimt with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

LIMIT signifies limit dimensioning.

lolimt is a real number representing the lower limit dimension in the nominal "T(lname)" format and the actual minimum distance in the actual "TA(lname)" format.

lotol is a real number representing the signed lower tolerance, or in the case where var_3 does not exist and the two features do not resolve to points then lotol is the signed lower tolerance in the nominal "T(lname)" format and is the most negative deviation from nominal in the actual "TA(lname)" format.

MAX signifies maximum distance between two features.

meanerror is a real number representing the mean error of the measurement.

MIN signifies minimum distance between two features.

NOMINL signifies plus and minus tolerancing.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

PT2PT signifies that the distance between is point to point, or feature to feature.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uplimt is a real number representing the upper limit dimension in the nominal "T(lname)" format and the actual maximum distance in the actual "TA(lname)" format.

uptol is a real number representing the signed upper tolerance, or in the case where var_3 does not exist and the two features do not resolve to points the uptol is the signed upper tolerance in the nominal "T(lname)" format and is the most positive deviation from nominal in the actual "TA(lname)" format.

XAXIS signifies that the distance between is along the X axis.

YAXIS signifies that the distance between is along the Y axis.

ZAXIS signifies that the distance between is along the Z axis.

In the case where var_3 exists or the two features do resolve to points, then lolimt and uplimt will have the same value in the actual "TA(lname)" format.

When tolerancing the distance between two features that resolve to points, var_3 is not required.

The TOL/DISTB statement is passed to the output file by execution of the OUTPUT statement.

Note 1: If a distance and tolerance such as 5 +1,-2 are given, then uptol = +1 and lotol = -2. Unsigned lotol and uptol values are assumed positive.

Note 2: For LIMIT format on output, lolimt is equal to uplimt.

6.198 TOL/DISTWRT

Function: Specifies a distance and a tolerance with respect to a previously defined feature or datum and assigns a label to them. This is a direct tolerance.

Input Formats:

can be: **T(lname)=TOL/DISTWRT,var_1,var_2,var_3 var_4**

Output Formats:

can be: **TA(lname)=TOL/DISTWRT,var_5,var_1,var_2,var_3 var_4**

or: **T(lname)=TOL/DISTWRT,var_1,var_2,var_3 var_4**

Where:

var_1 can be: **NOMINL,dist,lotol,uptol**
or: **LIMIT,lolimt,uplimt**

var_2 can be: **DAT(x)**
or: **F(lname1)**
or: **FA(lname2)**

var_3 can be: **XAXIS**
or: **YAXIS**
or: **ZAXIS**
or: **PT2PT**

var_4 can be: **,AVG**
or: **,MAX**
or: **,MIN**
or: **does not exist**

var_5 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

AVG signifies average or mean distance between two features.

combinedunc is a real number representing the combined uncertainty of the measurement.

DAT(x) is the datum to be used as a reference in which the distance with respect to will be calculated.

dist is a real number representing the nominal distance in the input format and the actual distance in the output format.

DISTWRT signifies the value and tolerance are applied to the distance between a feature with respect to a feature.

F(lname1) is the label of a feature nominal to be used as a reference in which the distance with respect to will be calculated.

FA(lname2) is the label of a feature actual to be used as a reference in which the distance with respect to will be calculated.

INTOL signifies the actual is within tolerance, that is when var_1 is NOMINL: uptol >= ($dist_{actual}$ – $dist_{nominal}$) >= lotol, or when var_1 is LIMIT: uplimt >= $dist_{actual}$ >= lolimt with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

LIMIT signifies limit dimensioning.

lolimt is a real number representing the lower limit dimension in the nominal "T(lname)" format and the actual minimum distance in the actual "TA(lname)" format.

lotol is a real number representing the signed lower tolerance, or in the case where var_3 does not exist and the two features do not resolve to points then lotol is the signed lower tolerance in the nominal "T(lname)" format and is the most negative deviation from nominal in the actual "TA(lname)" format.

MAX signifies maximum distance between two features.

meanerror is a real number representing the mean error of the measurement.

MIN signifies minimum distance between two features.

NOMINL signifies plus and minus tolerancing.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

PT2PT signifies that the distance between is point to point, or feature to feature.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uplimt is a real number representing the upper limit dimension in the nominal "T(lname)" format and the actual maximum distance in the actual "TA(lname)" format.

uptol is a real number representing the signed upper tolerance, or in the case where var_3 does not exist and the two features do not resolve to points the uptol is the signed upper tolerance in the nominal "T(lname)" format and is the most positive deviation from nominal in the actual "TA(lname)" format.

XAXIS signifies that the distance between is along the X axis.

YAXIS signifies that the distance between is along the Y axis.

ZAXIS signifies that the distance between is along the Z axis.

Unlike TOL/DISTB, the use of the TOL/DISTWRT statement is applied to a single feature.

In the case where var_4 exists or the reference feature or datum and the feature being toleranced both resolve to points, then lolimt and uplimt will have the same value in the actual "TA(lname)" format.

When tolerancing the distance between a reference feature or datum and the feature that both resolve to points, var_4 is not required.

The TOL/DISTWRT statement is passed to the output file by execution of the OUTPUT statement.

Note 1: If a distance and tolerance such as 5 +1,-2 are given, then uptol = +1 and lotol = -2. Unsigned lotol and uptol values are assumed positive.

Note 2: For LIMIT format on output, lolimt is equal to uplimt.

6.199 TOL/FLAT

Function: Specifies a flatness tolerance, and assigns a label to it. This is a tolerance of form.

Input Formats:

can be: **T(lname)=TOL/FLAT,var_1**

Output Formats:

can be: **TA(lname)=TOL/FLAT,tolzon,var_2 var_3**

or: **T(lname)=TOL/FLAT,var_1**

Where:

var_1 can be: **tolzon**
or: **tolzon,tolzon1,unit1,unit2**
or: **tolzon1,unit1,unit2**

var_2 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_3 can be: **,tolzon1,var_2**
or: **does not exist**

combinedunc is a real number representing the combined uncertainty of the measurement.

FLAT signifies flatness tolerance.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

tolzon is a positive real number representing the width of the tolerance zone defined by two parallel planes within which the surface of the feature lies. This is a specified value in the input format and either a specified or actual value in the output format.

tolzon1 is a positive real number representing the tolerance zone value applied by a per area basis. This is a specified value in the input format and either a specified or actual value in the output format.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

unit1,unit2 are positive real numbers representing the dimensions of the area over which the per unit tolerance zone is to be applied.

The TOL/FLAT statement is passed to the output file by execution of the OUTPUT statement.

6.200 TOL/GTOL

Function: Specifies the parameters associated with soft functional gauging, and assigns a label to it.

Input Formats:

can be: **T(lname)=TOL/GTOL,var_1,var_2,maxitr,minshf,minrot var_3**

Output Formats:

can be: **TA(lname)=TOL/GTOL,maxitr var_4,TRMATX,a1,a2,a3,b1,b2,b3,c1,c2 $**
,c3,d1,d2,d3 var_5

or: **T(lname)=TOL/GTOL,var_1,var_2,maxitr,minshf,minrot var_3**

Where:

var_1 can be: **XDIR**
or: **YDIR**
or: **ZDIR**
or: **XYDIR**
or: **YZDIR**
or: **ZXDIR**
or: **XYZDIR**
or: **NOTRAN**

var_2 can be: **XAXIS**
or: **YAXIS**
or: **ZAXIS**
or: **XYAXIS**
or: **YZAXIS**
or: **ZXAXIS**
or: **XYZAXI**
or: **NOROT**

var_3 can be: **,PERCNT**
or: **,INTFPT**
or: **does not exist**

var_4 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_5 can be: **,pcent**
or: **,INTFPT**

```
FA(lname1)[n1]=FEAT/var_6,PTDATA,CART,x1,y1,z1
FA(lname2)[n2]=FEAT/var_6,PTDATA,CART,x2,y2,z2
FA(lname3)[n3]=FEAT/var_6,PTDATA,CART,x3,y3,z3
...
ENDAT
```

or: **does not exist**

var_6 can be: **ARC**
or: **CIRCLE**
or: **CONE**
or: **CONRADSEGMNT**
or: **CYLRADSEGMNT**
or: **CYLNDR**
or: **ELLIPS**
or: **ELONGCYL**
or: **GCURVE**
or: **GSURF**
or: **LINE**
or: **PARPLN**
or: **PATERN**
or: **PLANE**
or: **POINT**
or: **RCTNGL**
or: **REVSURF**
or: **SPHERE**
or: **SPHRADSEGMNT**
or: **SYMPLN**
or: **TORRADSEGMNT**
or: **TORUS**

a1,a2,a3, b1,b2,b3, c1,c2,c3, d1,d2,d3 satisfy the following transformation equations:

(a1)x + (b1)y + (c1)z + d1 = x'
(a2)x + (b2)y + (c2)z + d2 = y'
(a3)x + (b3)y + (c3)z + d3 = z'

Where: x', y', and z' are the coordinates in the current part coordinate system (after the transformation has been applied) of any point, and x, y, and z are the coordinates in the previous coordinate system of the same point.

ARC signifies that an interference point occurred with a circular arc.

CIRCLE signifies that an interference point occurred with a circle.

combinedunc is a real number representing the combined uncertainty of the measurement.

CONE signifies that an interference point occurred with a cone.

CONRADSEGMNT signifies that an interference point occurred with a conical radial segment.

CYLNDR signifies that an interference point occurred with a cylinder.

CYLRADSEGMNT signifies that an interference point occurred with a cylindrical radial segment.

ELLIPS signifies that an interference point occurred with an ellipse.

ELONGCYL signifies that an interference point occurred with an elongated cylinder.

ENDAT signifies the end of the interference point output data.

FA(lname1)[n1] is the feature actual label of an interference point (x1,y1,z1) that follows on the next line. (lname1) is the feature label name and n1 is a positive integer representing the point number of the PTMEAS within the MEAS...ENDMES block that measured the data x1,y1,z1 (point index of the feature's individual point data).

FA(lname2)[n2] is the feature actual label of an interference point (x2,y2,z2) that follows on the next line. (lname2) is the feature label name and n2 is a positive integer representing the point number of the PTMEAS within the MEAS...ENDMES block that measured the data x2,y2,z2 (point index of the feature's individual point data).

FA(lname3)[n3] is the feature actual label of an interference point (x3,y3,z3) that follows on the next line. (lname3) is the feature label name and n3 is a positive integer representing the point number of the PTMEAS within the MEAS...ENDMES block that measured the data x3,y3,z3 (point index of the feature's individual point data).

GCURVE signifies that an interference point occurred with a generic curve.

GSURF signifies that an interference point occurred with a generic surface.

GTOL signifies soft functional gauge tolerance(parameters).

INTFPT signifies the desired output from the tolerance statement is to include the points of interference (if any) along with identifiers of which feature and which point within the feature.

INTOL signifies part is within tolerance (without interference) with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

LINE signifies that an interference point occurred with a line.

maxitr is a positive integer number that represents, on input, the maximum number of iterations to be performed by the DME soft gauge algorithms and, on output, represents the number of iterations that were actually performed.

meanerror is a real number representing the mean error of the measurement.

minrot is a real number representing the minimum amount of rotation, of the part data per iteration in the soft gauge software algorithm on the DME.

minshf is a real number representing the minimum amount of translation, of the part data per iteration in the soft gauge software algorithm on the DME.

NOROT signifies that the soft gauge cannot rotate the part data.

NOTRAN signifies that the soft gauge cannot translate the part data.

OUTOL signifies part is out of tolerance (interference) with no consideration made to measurement uncertainty.

PARPLN signifies that an interference point occurred with a feature of linear size (slot, block).

PATERN signifies that an interference point occurred with a pattern.

pcent is a positive integer number representing the percentage of points that are found to be non-interference points within the soft gauge. pcent is a percentage number (for example 80 would signify 80%). pcent is present in the output only when the PERCNT option is used on the input format.

PERCNT signifies the desired output from the tolerance statement is to return the percentage of good points found during the soft gauge algorithm application by the DME.

PLANE signifies that an interference point occurred with a plane.

POINT signifies that an interference point occurred with a point feature.

PTDATA signifies that point data follows.

RCTNGL signifies that an interference point occurred with a rectangle.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

REVSURF signifies that an interference point occurred with a surface-of-revolution.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

SPHERE signifies that an interference point occurred with a sphere.

SPHRADSEGMNT signifies that an interference point occurred with a spherical radial segment.

SYMPLN signifies that an interference point occurred with a feature of linear size (slot, block).

TORRADSEGMNT signifies that an interference point occurred with a toroidal radial segment.

TORUS signifies that an interference point occurred with a torus.

TRMATX signifies that the transformation matrix data follows.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

x1,y1,z1 is the x, y, and z coordinate data of the interference point.

x2,y2,z2 is the x, y, and z coordinate data of the interference point.

x3,y3,z3 is the x, y, and z coordinate data of the interference point.

XAXIS signifies that the soft gauge can rotate about the X axis.

XDIR signifies that the soft gauge can translate in the X direction.

XYAXIS signifies that the soft gauge can rotate about both the X and Y axes.

XYDIR signifies that the soft gauge can translate in both the X and Y direction.

XYZAXI signifies that the soft gauge can rotate about all the axes.

XYZDIR signifies that the soft gauge can translate in all directions.

YAXIS signifies that the soft gauge can rotate about the Y axis.

YDIR signifies that the soft gauge can translate in the Y direction.

YZAXIS signifies that the soft gauge can rotate about both the Y and Z axes.

YZDIR signifies that the soft gauge can translate in both the Y and Z direction.

ZAXIS signifies that the soft gauge can rotate about the Z axis.

ZDIR signifies that the soft gauge can translate in the Z direction.

ZXAXIS signifies that the soft gauge can rotate about both the Z and X axes.

ZXDIR signifies that the soft gauge can translate in both the Z and X direction.

The TOL/GTOL statement is passed to the output file by execution of the OUTPUT statement.

Note 1: Soft functional gauge applications require that a soft gauge be designed in advance of the programming process much like hard functional gauges are designed and fabricated. During the soft gauge design, plating tolerances must be built into the dimensions used.

Note 2: During part measurements, datums should be established according to the part drawing and both the part and the soft gauge definitions should be made from a common coordinate system.

Note 3: The degrees of freedom specified by var_1 and var_2 are dictated by the part drawing datums, dimensions and tolerances, and should be consistent with hard functional gauging practices.

Note 4: Other DMIS statements concerned with soft functional gauging are CONST/SGAGE,SE (lname), CONST/SPART,ST(lname), PTBUFF/ON, and OUTPUT/SE(lname),ST(lname),TA(lname).

6.201 TOL/PARLEL

Function: Specifies a parallelism tolerance, and assigns a label to it. This is an orientation tolerance.

Input Formats:

can be: **T(lname)=TOL/PARLEL,tolzon var_1 var_2 var_3 var_4 var_5**

Output Formats:

can be: **TA(lname)=TOL/PARLEL,tolzon,var_6 var_1,lim var_2 var_3 var_4 var_5**

or: **T(lname)=TOL/PARLEL,tolzon var_1 var_2 var_3 var_4 var_5**

Where:

var_1 can be: **,MMC var_7**
or: **,LMC var_7**
or: **,RFS**
or: **does not exist**

var_2 can be: **,DAT(x) var_1**
or: **,F(lname1)**
or: **,FA(lname2) var_1**

var_3 can be: **,DAT(x) var_1**
or: **,F(lname1)**
or: **,FA(lname2) var_1**
or: **does not exist**

var_4 can be: **,TANGPL**
or: **,PARPLN**
or: **does not exist**

var_5 can be: **,XAXIS**
or: **,YAXIS**
or: **,ZAXIS**
or: **,VEC,i,j,k**
or: **does not exist**

var_6 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_7 can be: **,MAX,maxtol**
or: **does not exist**

combinedunc is a real number representing the combined uncertainty of the measurement.

DAT(x) is the datum to be used as a reference.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of a feature actual to be used as a reference.

i,j,k is the direction vector along which the view dependent parallel plane tolerance zone width is to be applied.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lim is a real number representing the sum of the tolerance zone plus any gain from MMC or LMC.

LMC	signifies that a least material condition is applied.
MAX	signifies that a maximum allowed bonus tolerance is to be used.
maxtol	is a real number representing the maximum allowed bonus tolerance value to be used.
meanerror	is a real number representing the mean error of the measurement.
MMC	signifies that a maximum material condition is applied.
OUTOL	signifies the actual is out of tolerance with no consideration made to measurement uncertainty.
PARLEL	signifies parallelism tolerance.
PARPLN	signifies a tolerance zone defined by two parallel planes applied to axial features such as cylinders.
REQUNCERT	signifies that uncertainty evaluation data was requested but is not available.
RFS	signifies that a regardless of feature size condition is applied.
RULEINTOL	signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.
RULEOUTOL	signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.
RULEUNDET	signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.
RULEUNSUP	signifies that the specified decision rule is not supported and that the uncertainty data will follow.
TANGPL	signifies that a tangent plane is established by contacting the high points of a surface.
tolzon	is a positive real number representing the width of the tolerance zone defined by two parallel lines within which all points of the feature lie, or the distance between parallel planes within which the center plane of the feature lies, or the diameter of a cylindrical tolerance zone within which the axis of the feature lies. This is a specified value in the input format and either a specified or actual value in the output format.
UNCERT	signifies that no decision rule was stipulated and that the uncertainty data will follow.
VEC	signifies that the view dependent parallel plane tolerance zone width is to be applied along the vector specified with i,j,k.
XAXIS	signifies that the view dependent parallel plane tolerance zone width is to be applied along the X axis.
YAXIS	signifies that the view dependent parallel plane tolerance zone width is to be applied along the Y axis.
ZAXIS	signifies that the view dependent parallel plane tolerance zone width is to be applied along the Z axis.

When var_5 does not exist and the tolerance is being applied to a line-reducible feature, the tolerance zone is cylindrical.

The TOL/PARLEL statement is passed to the output file by execution of the OUTPUT statement.

Note: Projected tolerance zones per ASME Y14.5M-1994 can be supported by bounding the tolerance using the DMIS BOUND statement. Refer to clauses 5.3.2.7.2 and 5.3.2.7.6.

6.202 TOL/PERP

Function: Specifies a perpendicularity tolerance, and assigns a label to it. This is an orientation tolerance.

Input Formats:

can be: **T(lname)=TOL/PERP,tolzon var_1 var_2 var_3 var_4 var_5**

Output Formats:

can be: **TA(lname)=TOL/PERP,tolzon,var_6 var_1,lim var_2 var_3 var_4 var_5**

or: **T(lname)=TOL/PERP,tolzon var_1 var_2 var_3 var_4 var_5**

Where:

var_1 can be: **,MMC var_7**
or: **,LMC var_7**
or: **,RFS**
or: **does not exist**

var_2 can be: **,DAT(x) var_1**
or: **,F(lname1)**
or: **,FA(lname2) var_1**

var_3 can be: **,DAT(x) var_1**
or: **,F(lname1)**
or: **,FA(lname2) var_1**
or: **does not exist**

var_4 can be: **,TANGPL**
or: **,PARPLN**
or: **does not exist**

var_5 can be: **,XAXIS**
or: **,YAXIS**
or: **,ZAXIS**
or: **,VEC,i,j,k**
or: **does not exist**

var_6 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_7 can be: **,MAX,maxtol**
or: **does not exist**

combinedunc is a real number representing the combined uncertainty of the measurement.

DAT(x) is the datum to be used as a reference.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of a feature actual to be used as a reference.

i,j,k is the direction vector along which the view dependent parallel plane tolerance zone width is to be applied.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lim	is a real number representing the sum of the tolerance zone plus any gain from MMC or LMC.
LMC	signifies that a least material condition is applied.
MAX	signifies that a maximum allowed bonus tolerance is to be used.
maxtol	is a real number representing the maximum allowed bonus tolerance value to be used.
meanerror	is a real number representing the mean error of the measurement.
MMC	signifies that a maximum material condition is applied.
OUTOL	signifies the actual is out of tolerance with no consideration made to measurement uncertainty.
PARPLN	signifies a tolerance zone defined by two parallel planes applied to axial features such as cylinders.
PERP	signifies perpendicularity tolerance.
REQUNCERT	signifies that uncertainty evaluation data was requested but is not available.
RFS	signifies that a regardless of feature size condition is applied.
RULEINTOL	signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.
RULEOUTOL	signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.
RULEUNDET	signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.
RULEUNSUP	signifies that the specified decision rule is not supported and that the uncertainty data will follow.
TANGPL	signifies that a tangent plane is established by contacting the high points of a surface.
tolzon	is a positive real number representing the width of the tolerance zone defined by two parallel lines within which all points of the feature lie, or the distance between parallel planes within which the center plane of the feature lies, or the diameter of a cylindrical tolerance zone within which the axis of the feature lies. This is a specified value in the input format and either a specified or actual value in the output format.
UNCERT	signifies that no decision rule was stipulated and that the uncertainty data will follow.
VEC	signifies that the view dependent parallel plane tolerance zone width is to be applied along the vector specified with i,j,k.
XAXIS	signifies that the view dependent parallel plane tolerance zone width is to be applied along the X axis.
YAXIS	signifies that the view dependent parallel plane tolerance zone width is to be applied along the Y axis.
ZAXIS	signifies that the view dependent parallel plane tolerance zone width is to be applied along the Z axis.

When var_5 does not exist and the tolerance is being applied to a line-reducible feature, the tolerance zone is cylindrical.

The TOL/PERP statement is passed to the output file by execution of the OUTPUT statement.

Note: Projected tolerance zones per ASME Y14.5M-1994 can be supported by bounding the tolerance using the DMIS BOUND statement. Refer to clauses 5.3.2.7.2 and 5.3.2.7.6.

6.203 TOL/POS

Function: Specifies a position tolerance, and assigns a label to it. This is a location tolerance.

Input Formats:

can be: **T(lname)=TOL/POS,var_1,tolzon var_2 var_3 var_3 var_3 var_4**

Output Formats:

can be: **TA(lname)=TOL/POS,var_1,tolzon,var_5 var_6 var_3 var_3 var_3 var_4**

or: **T(lname)=TOL/POS,var_1,tolzon var_2 var_3 var_3 var_3 var_4**

Where:

var_1 can be: **2D**
or: **3D**

var_2 can be: **,MMC**
or: **,LMC**
or: **,RFS**
or: **does not exist**

var_3 can be: **,DAT(x) var_2**
or: **,F(lname1)**
or: **,FA(lname2) var_2**
or: **does not exist**

var_4 can be: **,XAXIS**
or: **,YAXIS**
or: **,ZAXIS**
or: **,RADIAL**
or: **,ANGLE**
or: **,VEC,i,j,k**
or: **does not exist**

var_5 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_6 can be: **,MMC,lim**
or: **,LMC,lim**
or: **,RFS**
or: **does not exist**

2D signifies a two dimensional tolerance zone, for example a circular zone or a zone described by two parallel lines.

3D signifies a three dimensional tolerance zone, for example a cylindrical zone, a spherical zone, or a zone described by two parallel planes.

ANGLE signifies that the polar coordinate method is used to tolerance the angular position.

combinedunc is a real number representing the combined uncertainty of the measurement.

DAT(x) is the datum to be used as a reference.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of a feature actual to be used as a reference.

i,j,k is the direction vector along which to apply the tolerance.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lim is a real number representing the sum of the tolerance zone plus any gain from MMC or LMC.

LMC signifies that a least material condition is applied.

meanerror is a real number representing the mean error of the measurement.

MMC signifies that a maximum material condition is applied.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

POS signifies a position tolerance.

RADIAL signifies that the polar coordinate method is used to tolerance the radial position.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RFS signifies that a regardless of feature size condition is applied.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

tolzon is a positive real number representing the size of the (2D) or (3D) tolerance zone. This is a specified value in the input format and either a specified or actual value in the output format.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

VEC signifies that a direction vector along which to apply the tolerance follows.

XAXIS signifies that the rectangular coordinate method is to be used to tolerance the position along the X axis.

YAXIS signifies that the rectangular coordinate method is to be used to tolerance the position along the Y axis.

ZAXIS signifies that the rectangular coordinate method is to be used to tolerance the position along the Z axis.

The TOL/POS statement is passed to the output file by execution of the OUTPUT statement.

Note 1: The material condition modifiers in var_2 signify whether MMC or LMC is applied to the feature of size. The material condition modifiers when used within var_3 signify to which datum(s), if any, MMC or LMC is applied.

Note 2: The presence of var_4 signifies that the tolerance is to be evaluated as a bi-directional positional tolerance per ASME Y14.5M-1994, clause 5.9.

Note 3: Projected tolerance zone per ASME Y14.5M-1994 can be supported by bounding the tolerance using the DMIS BOUND statement. Refer to clauses 5.3.2.7.2 and 5.3.2.7.6.

6.204 TOL/PROFL

Function: Specifies a profile of a line (curve) tolerance, and assigns a label to it. This is a profile tolerance.

Input Formats:

can be: **T(lname)=TOL/PROFL,lotol,uptol var_1 var_1 var_1 var_2**

Output Formats:

can be: **TA(lname)=TOL/PROFL,lotol,uptol,var_3 var_1 var_1 var_1 var_2**

or: **T(lname)=TOL/PROFL,lotol,uptol var_1 var_1 var_1 var_2**

Where:

var_1 can be: **,DAT(x) var_4**
or: **,F(lname1)**
or: **,FA(lname2) var_4**
or: **does not exist**

var_2 can be: **,XYPLAN**
or: **,ZXPLAN**
or: **,YZPLAN**
or: **,VEC,i,j,k**
or: **does not exist**

var_3 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_4 can be: **,MMC**
or: **,LMC**
or: **,RFS**
or: **does not exist**

combinedunc is a real number representing the combined uncertainty of the measurement.

DAT(x) is the datum to be used as a reference.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of a feature actual to be used as a reference.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

LMC signifies that a least material condition is applied.

lotol is a real number representing the signed lower tolerance value, which lies to the inside of the part (away from the normal of the feature). This is a specified value in the input format and either a specified or actual value in the output format.

meanerror is a real number representing the mean error of the measurement.

MMC signifies that a maximum material condition is applied.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

PROFL signifies that the tolerance is a line profile tolerance.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RFS signifies that a regardless of feature size condition is applied.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol is a real number representing the signed upper tolerance value, which lies to the outside of the part (along the normal of the feature). This is a specified value in the input format and either a specified or actual value in the output format.

VEC,i,j,k signifies that profile of a line is checked in a plane that is described by its normal vector.

XYPLAN signifies that profile of a line is checked in the XY plane.

YZPLAN signifies that profile of a line is checked in the YZ plane.

ZXPLAN signifies that profile of a line is checked in the ZX plane.

The TOL/PROFL statement is passed to the output file by execution of the OUTPUT statement.

Note: The profile of a line tolerance specifies a tolerance zone which is a two dimensional band, extending along the length of the feature. If the tolerance is unbounded, it is applied along the length of the feature. The tolerance may be bounded when it is desired to apply it to only a portion of the feature. If no material condition is supplied then regardless of feature size is assumed.

6.205 TOL/PROFP

Function: Specifies a profile tolerance applied to a point along the normal of the point's corresponding line(curve) or surface, and assigns a label to it. This is a profile tolerance.

Input Formats:

can be: **T(lname)=TOL/PROFP,lotol,uptol var_1 var_1 var_1**

Output Formats:

can be: **TA(lname)=TOL/PROFP,dev,var_2 var_1 var_1 var_1**

or: **T(lname)=TOL/PROFP,lotol,uptol var_1 var_1 var_1**

Where:

var_1 can be: **,DAT(x) var_3**
or: **,F(lname1)**
or: **,FA(lname2) var_3**
or: **does not exist**

var_2 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_3 can be: **,MMC**
or: **,LMC**
or: **,RFS**
or: **does not exist**

combinedunc is a real number representing the combined uncertainty of the measurement.

DAT(x) is the datum to be used as a reference.

dev is a real number representing the signed deviation from the nominal value along the normal vector, which establishes the positive direction of dev.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of a feature actual to be used as a reference.

INTOL signifies the actual is within tolerance, that is when uptol >= dev >= lotol with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

LMC signifies that a least material condition is applied.

lotol is a real number representing the signed lower tolerance value, which lies to the inside of the part. This is a specified value in the input format and either a specified or actual value in the output format.

meanerror is a real number representing the mean error of the measurement.

MMC signifies that a maximum material condition is applied.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

PROFP signifies that the tolerance is a line profile tolerance applied to a discreet point.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RFS signifies that a regardless of feature size condition is applied.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol is a real number representing the signed upper tolerance value, which lies to the outside of the part. This is a specified value in the input format and either a specified or actual value in the output format.

If the TOL/PROFP statement is applied to an FEAT/EDGEPT then the tolerance is applied along the edge vector.

The TOL/PROFP statement is passed to the output file by execution of the OUTPUT statement.

Note: The TOL/PROFP is an extension of the profile tolerance. It is used when an individual point on a line (curve) or surface is to be analyzed separately against a profile tolerance. The profile tolerance applies to a specific point that lies on a line (curve) or surface whereas the tolerance is along the normal of the corresponding line (curve) or surface at the location of the nominal point.

6.206 TOL/PROFS

Function: Specifies a profile of a surface tolerance, and assigns a label to it. This is a profile tolerance.

Input Formats:

can be: **T(lname)=TOL/PROFS,lotol,uptol var_1 var_1 var_1 var_2**

Output Formats:

can be: **TA(lname)=TOL/PROFS,var_3 var_4 var_1 var_1 var_1**

or: **T(lname)=TOL/PROFS,lotol,uptol var_1 var_1 var_1 var_2**

Where:

var_1 can be: **,DAT(x) var_5**
or: **,F(lname1)**
or: **,FA(lname2) var_5**
or: **does not exist**

var_2 can be: **,AVGDEV**
or: **does not exist**

var_3 can be: **lotol,uptol**
or: **AVGDEV,dev**

var_4 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_5 can be: **,MMC**
or: **,LMC**
or: **,RFS**
or: **does not exist**

AVGDEV signifies that the mean deviation (signified by 'dev' in the output format) is to be reported as the mean deviation instead of using the default maximum deviation from the measured feature's nominal definition.

combinedunc is a real number representing the combined uncertainty of the measurement.

DAT(x) is the datum to be used as a reference.

dev is a real number representing the signed deviation from the nominal value along the normal vector, which establishes the positive direction of dev.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of a feature actual to be used as a reference.

INTOL signifies the actual is within tolerance, that is when uptol >= dev >= lotol with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

LMC signifies that a least material condition is applied.

lotol is a real number representing the signed lower tolerance value, which lies to the inside of the part. This is a specified value in the input format and either a specified or actual value in the output format.

meanerror is a real number representing the mean error of the measurement.

MMC signifies that a maximum material condition is applied.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

PROFS signifies that the tolerance is a surface profile tolerance.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RFS signifies that a regardless of feature size condition is applied.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol is a real number representing the signed upper tolerance value, which lies to the outside of the part. This is a specified value in the input format and either a specified or actual value in the output format.

If the TOL/ PROFS statement is applied to an FEAT/EDGEPT then the tolerance is applied along the surface vector adjacent to the edge in which the edge point lies.

When AVGDEV is specified in the input format, "AVGDEV, dev" is required in the output format.

The TOL/PROFS statement is passed to the output file by execution of the OUTPUT statement.

Note: The profile of a surface tolerance specifies a three dimensional tolerance zone. As with the profile of a line tolerance, it may be unbounded or bounded.

6.207 TOL/RAD

Function: Specifies a radial tolerance, and assigns a label to it. This is a direct tolerance.

Input Formats:

can be: **T(lname)=TOL/RAD,lotol,uptol var_1 var_2**

Output Formats:

can be: **TA(lname)=TOL/RAD,dev var_3,var_4 var_1**

or: **T(lname)=TOL/RAD,lotol,uptol var_1 var_2**

Where:

var_1 can be: **,AVG**
or: **,CRAD**
or: **does not exist**

var_2 can be: **,MINMAX**
or: **does not exist**

var_3 can be: **,MINMAX,mindev,maxdev**
or: **does not exist**

var_4 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

AVG signifies that the average radius is to be used for the measured feature size. The average radius is to be computed as the arithmetic average of several (more than 4) radial measurements or calculations.

combinedunc is a real number representing the combined uncertainty of the measurement.

CRAD signifies that the radius is a controlled radius per ASME Y14.5M-1994.

dev is a real number representing the arithmetic difference between the actual value and the nominal value. It is positive when the nominal is less than the actual, and negative when the nominal is greater than the actual.

INTOL signifies the actual is within tolerance, that is when uptol >= dev >= lotol with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lotol is a real number representing the signed lower tolerance value applied to the radius.

maxdev is a real number representing the most positive deviation from nominal over all points on the measured feature.

meanerror is a real number representing the mean error of the measurement.

mindev is a real number representing the most negative deviation from nominal over all points on the measured feature.

MINMAX signifies that the evaluation tolerance will output the minimum and maximum deviations to the feature nominal.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

RAD signifies that the tolerance is a radial tolerance.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol is a real number representing the signed upper tolerance value applied to the radius.

The TOL/RAD statement is passed to the output file by execution of the OUTPUT statement.

Note: If a radius tolerance such as 5 +1, -2 is given, then uptol = +1 and lotol = -2. Unsigned lotol and uptol values are assumed positive. A radial tolerance can be applied to a circular arc, circle, cylindrical radial segment, spherical radial segment, toroidal radial segment, cylinder, sphere, and ellipse.

6.208 TOL/STRGHT

Function: Specifies a straightness tolerance, and assigns a label to it. This is a tolerance of form.

Input Formats:

can be: **T(lname)=TOL/STRGHT,var_1 var_2 var_3**

Output Formats:

can be: **TA(lname)=TOL/STRGHT,tolzon,var_4 var_5 var_2 var_3**

or: **T(lname)=TOL/STRGHT,var_1 var_2 var_3**

Where:

var_1 can be: **tolzon var_6**
or: **tolzon,unit**

var_2 can be: **,XAXIS**
or: **,YAXIS**
or: **,ZAXIS**
or: **does not exist**

var_3 can be: **,XDIR**
or: **,YDIR**
or: **,ZDIR**
or: **,VEC,i,j,k**
or: **does not exist**

var_4 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_5 can be: **,MMC,lim**
or: **,LMC,lim**
or: **,RFS**
or: **,tolzon1,var_4**
or: **does not exist**

var_6 can be: **,MMC**
or: **,LMC**
or: **,RFS**
or: **,tolzon1,unit**
or: **does not exist**

combinedunc is a real number representing the combined uncertainty of the measurement.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

lim is a real number representing the sum of the tolerance zone plus any gain from MMC or LMC.

LMC signifies that a least material condition is applied.

meanerror is a real number representing the mean error of the measurement.

MMC signifies that a maximum material condition is applied.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RFS signifies that a regardless of feature size condition is applied.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

STRGHT signifies straightness tolerance.

tolzon is a positive real number representing the width of the tolerance zone defined by two parallel lines within which all points of the feature lie, or the distance between parallel planes within which the center plane of the feature lies, or the diameter of a cylindrical tolerance zone within which the axis of the feature lies. This is a specified value in the input format and either a specified or actual value in the output format.

tolzon1 is a positive real number representing the tolerance zone applied by a per unit basis. This is a specified value in the input format and either a specified or actual value in the output format.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

unit is a positive real number representing the dimension of the length over which the per unit tolerance zone is to be applied.

VEC,i,j,k signifies that straightness is to be checked along the vector's direction.

XAXIS signifies that a planar tolerance zone perpendicular to X is to be used.

XDIR signifies that straightness is to be checked along the X axis.

YAXIS signifies that a planar tolerance zone perpendicular to Y is to be used.

YDIR signifies that straightness is to be checked along the Y axis.

ZAXIS signifies that a planar tolerance zone perpendicular to Z is to be used.

ZDIR signifies that straightness is to be checked along the Z axis.

The TOL/STRGHT statement is passed to the output file by execution of the OUTPUT statement.

MMC or LMC can only be applied to features that have a center-line, center-plane or axis.

If var_2 does not exist, the tolerance zone is cylindrical.

var_3 can only be used when STRGHT is applied to planes.

6.209 TOL/SYM

Function: Specifies a symmetry tolerance, and assigns a label to it. This is a location tolerance.

Input Formats:

can be: **T(lname)=TOL/SYM,tolzon,var_1 var_2**

Output Formats:

can be: **TA(lname)=TOL/SYM,tolzon,var_3,var_1 var_2**

or: **T(lname)=TOL/SYM,tolzon,var_1 var_2**

Where:

var_1 can be: **DAT(x) var_4**
or: **F(lname1)**
or: **FA(lname2) var_4**

var_2 can be: **,XDIR**
or: **,YDIR**
or: **,ZDIR**
or: **,VEC,i,j,k**
or: **does not exist**

var_3 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

var_4 can be: **,MMC**
or: **,LMC**
or: **,RFS**
or: **does not exist**

combinedunc is a real number representing the combined uncertainty of the measurement.

DAT(x) is the datum to be used as a reference.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of a feature actual to be used as a reference.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

LMC signifies that a least material condition is applied.

meanerror is a real number representing the mean error of the measurement.

MMC signifies that a maximum material condition is applied.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RFS signifies that a regardless of feature size condition is applied.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

SYM signifies a symmetry tolerance.

tolzon is a positive real number representing the width of the tolerance zone. This is a specified value in the input format and either a specified or actual value in the output format.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

VEC,i,j,k signifies that symmetry is to be checked along the vector's direction.

XDIR signifies that symmetry is to be checked along the X axis.

YDIR signifies that symmetry is to be checked along the Y axis.

ZDIR signifies that symmetry is to be checked along the Z axis.

The TOL/SYM statement is passed to the output file by execution of the OUTPUT statement.

6.210 TOL/TRNOUT

Function: Specifies a total runout tolerance, and assigns a label to it. This is a runout tolerance.

Input Formats:

can be: **T(lname)=TOL/TRNOUT,tolzon,DAT(x) var_1 var_1**

Output Formats:

can be: **TA(lname)=TOL/TRNOUT,tolzon,var_2,DAT(x) var_1 var_1**

or: **T(lname)=TOL/TRNOUT,tolzon,DAT(x) var_1 var_1**

Where:

var_1 can be: **,DAT(x)**
or: **,F(lname1)**
or: **,FA(lname2)**
or: **does not exist**

var_2 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

combinedunc is a real number representing the combined uncertainty of the measurement.

DAT(x) is the datum to be used as an axis and can be in the form of (x-x) for constructed compound datums.

F(lname1) is the label of a feature nominal to be used as a reference.

FA(lname2) is the label of a feature actual to be used as a reference.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

tolzon is a positive real number representing the width of the tolerance zone defined by two parallel lines within which all points of the feature lie, or the distance between parallel planes within which the center plane of the feature lies, or the diameter of a cylindrical tolerance zone within which the axis of the feature lies. This is a specified value in the input format and either a specified or actual value in the output format.

TRNOUT signifies total runout tolerance.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

The TOL/TRNOUT statement is passed to the output file by execution of the OUTPUT statement.

Note: Total runout provides composite control of an entire feature. The tolerance is applied simultaneously to all circular and profile measuring positions as the part is rotated 360 degrees. Where applied to surfaces constructed around a datum axis, total runout is used to control cumulative variations of circularity, straightness, coaxiality, angularity, taper, and profile of a surface. Where applied to surfaces constructed at right angles to a datum axis, total runout controls cumulative variations of perpendicularity (to detect wobble) and flatness (to detect concavity or convexity).

6.211 TOL/USETOL

Function: Used to identify a "user created" tolerance that may have definable characteristics in another file that the DME has the capability to measure, and assigns a label to it.

Input Formats:

can be: **T(lname)=TOL/USETOL,'text'**

or: **T(lname)=TOL/USETOL,parm var_1**

Output Formats:

can be: **TA(lname)=TOL/USETOL,var_2,'text'**

or: **T(lname)=TOL/USETOL,'text'**

or: **TA(lname)=TOL/USETOL,var_2,parm var_1**

or: **T(lname)=TOL/USETOL,parm var_1**

Where:

var_1 can be: **,parm var_1**

or: **does not exist**

var_2 can be: **INTOL**

or: **OUTOL**

or: **RULEINTOL,meanerror,combinedunc**

or: **RULEOUTOL,meanerror,combinedunc**

or: **RULEUNDET,meanerror,combinedunc**

or: **RULEUNSUP,meanerror,combinedunc**

or: **UNCERT,meanerror,combinedunc**

or: **REQUNCERT**

combinedunc is a real number representing the combined uncertainty of the measurement.

INTOL signifies the actual is within tolerance with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

meanerror is a real number representing the mean error of the measurement.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

parm is a value or variable of any DECL type, the meaning of which is determined by the implementation.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

'text' is any text required to identify the tolerance characteristics of the object, enclosed with apostrophes. These characteristics may be defined in the 'text', or be listed in a file identified by 'text'. There are examples of some of the uses for 'text' identified below as they apply to vision systems and electronics industry applications.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

USETOL signifies that the tolerance is user defined.

The TOL/USETOL statement is passed to the output file by execution of the OUTPUT statement

Note: The intent of this statement is to provide a means by which previously defined and coded tolerance requirements can be accessed for evaluation by a DME. The DME can, in turn, fully utilize its capabilities and functionality to perform the required evaluations. This statement is used in conjunction with the FEAT/OBJECT statement, explained above.

Some examples for 'text' include codes for the SRI Machine Vision attributes such as:

2-D MEASURE	3-D MEASURE	2-D & 3-D MEASURE
Similarity/Correlation	Similarity/Correlation	Principal moments of inertia
Presence/Absence	Presence/Absence	Second moments of inertia
Perimeter	Perimeter	Median point
Centroid	Centroid	Edge detection
Offset	Offset	
Area	Area	
Major axis	Volume	
Minor axis	Major axis	
	Minor axis	

Others may include: density, percent of void, or temperature. Additionally, 'text' can also be the name of a file that contains the required tolerance characteristics for the object.

6.212 TOL/WIDTH

Function: Specifies a linear size tolerance, and assigns a label to it. This is a direct tolerance.

Input Formats:

can be: **T(lname)=TOL/WIDTH,lotol,uptol var_1 var_2**

Output Formats:

can be: **TA(lname)=TOL/WIDTH,dev var_3,var_4 var_1**

or: **T(lname)=TOL/WIDTH,lotol,uptol var_1 var_2**

Where:

var_1 can be: **,i,j,k**
or: **,SHORT**
or: **,LONG**
or: **does not exist**

var_2 can be: **,MINMAX**
or: **does not exist**

var_3 can be: **,MINMAX,mindev,maxdev**
or: **does not exist**

var_4 can be: **INTOL**
or: **OUTOL**
or: **RULEINTOL,meanerror,combinedunc**
or: **RULEOUTOL,meanerror,combinedunc**
or: **RULEUNDET,meanerror,combinedunc**
or: **RULEUNSUP,meanerror,combinedunc**
or: **UNCERT,meanerror,combinedunc**
or: **REQUNCERT**

combinedunc is a real number representing the combined uncertainty of the measurement.

dev is a real number representing the arithmetic difference between the actual value and the nominal value. It is positive when the nominal is less than the actual and negative when the nominal is greater than the actual.

i,j,k is the direction vector along which the linear width is to be calculated.

INTOL signifies the actual is within tolerance, that is when uptol >= dev >= lotol with no consideration made to measurement uncertainty.

lname is an alphanumeric label name assigned to the tolerance.

LONG signifies that the WIDTH is toleranced along the CPARLN line length orientation vector.

lotol is a real number representing the signed lower tolerance value applied to the linear size (width).

maxdev is a real number representing the most positive deviation from nominal over all points on the measured feature.

meanerror is a real number representing the mean error of the measurement.

mindev is a real number representing the most negative deviation from nominal over all points on the measured feature.

MINMAX signifies that the evaluation tolerance will output the minimum and maximum deviations to the feature nominal.

OUTOL signifies the actual is out of tolerance with no consideration made to measurement uncertainty.

REQUNCERT signifies that uncertainty evaluation data was requested but is not available.

RULEINTOL signifies the actual is within tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEOUTOL signifies the actual is out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNDET signifies the actual cannot be said to be either within or out of tolerance according to the specified decision rule and that the uncertainty data will follow.

RULEUNSUP signifies that the specified decision rule is not supported and that the uncertainty data will follow.

SHORT signifies that the WIDTH is toleranced in the plane of the CPARLN feature normal perpendicular to the CPARLN line length orientation vector.

UNCERT signifies that no decision rule was stipulated and that the uncertainty data will follow.

uptol is a real number representing the signed upper tolerance value applied to the linear size (width).

WIDTH signifies that the tolerance is a linear size (width) tolerance.

The TOL/WIDTH statement is passed to the output file by execution of the OUTPUT statement.

Note: TOL/WIDTH may only be used with FEAT/CPARLN, FEAT/PARPLN, FEAT/RCTNGL or FEAT/SYMPLN. A var_1 is required for all cases except FEAT/PARPLN or FEAT/SYMPLN and is used to determine which of the possible widths in question to calculate. The nominal width must be contained in the feature definition.

6.213 TOOLDF

Function: Defines a tool used on a manufacturing device, and assigns a label to it.

Input Formats:

can be: **TL(lname)=TOOLDF/MD(lname1),'text'**

Output Formats:

can be: **TL(lname)= MD(lname1),'text'**

Where:

lname is an alphanumeric label name assigned to the tool.

MD(lname1) is the label of the manufacturing device with which this tool definition is associated.

'text' is a text string, enclosed with apostrophes, that identifies the manufacturing tool.

The TOOLDF statement is passed to the output file with execution of an OUTPUT statement, specifying a previously defined report that included the defined manufacturing device definition label name.

Note: This statement is associated with the CUTCOM and MFGDEV statements for applications to adjust the manufacturing process based on inspection results.

6.214 TRANS

Function: Translates a part coordinate system along an axis, and assigns a label to it.

Input Formats:

can be: **D(lname)=TRANS/var_1,var_2 var_3 var_3**

Output Formats:

can be all: **D(lname)=TRANS/var_1,var_4 var_5 var_5**
DA(lname)=TRANS/TRMATX,a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3

Where:

var_1 can be: **XORIG**
or: **YORIG**
or: **ZORIG**

var_2 can be: **var_6**
or: **var_7**

var_3 can be: **,var_1,var_2**
or: **does not exist**

var_4 can be: **var_6**
or: **var_8**

var_5 can be: **,var_1,var_4**
or: **does not exist**

var_6 can be: **value**
or: **F(lname1)**
or: **FA(lname2)**
or: **DAT(x)**

var_7 can be: **PRBRAD**
or: **-PRBRAD**

var_8 can be: **PRBRAD,proberad**
or: **-PRBRAD,proberad**

a1,a2,a3, b1,b2,b3, c1,c2,c3, d1,d2,d3 satisfy the following transformation equations:

(a1)x + (b1)y + (c1)z + d1 = x'
(a2)x + (b2)y + (c2)z + d2 = y'
(a3)x + (b3)y + (c3)z + d3 = z'

Where: x', y', and z' are the coordinates in the current part coordinate system (after the transformation has been applied) of any point, and x, y, and z are the coordinates in the previous coordinate system of the same point.

DAT(x) is the previously defined datum used to establish the origin.

F(lname1) is the label of the feature nominal used to establish the origin.

FA(lname2) is the label of the feature actual used to establish the origin.

lname is an alphanumeric label name assigned to the new part coordinate system.

PRBRAD signifies that the preceding origin component will be translated one half the probe diameter in the positive direction. The probe diameter is established by the DME during calibration.

-PRBRAD signifies that the preceding origin component will be translated one half the probe diameter in the negative direction. The probe diameter is established by the DME during calibration.

proberad is a positive real number representing the actual probe radius.

TRMATX signifies that the alignment and transformation information from the previous coordinate system is output.

value is a real number representing the distance the current coordinate system origin is to be translated. A positive value is a translation in the positive direction along the axis and, a negative value is a translation in the negative direction along the axis.

XORIG signifies that the current coordinate system origin is to be translated on the X axis if a value is given. Signifies the X coordinate of the origin is to be translated to the X coordinate of the following feature if F(lname1), FA(lname1), or DAT(x) is given.

YORIG signifies that the current coordinate system origin is to be translated on the Y axis if a value is given. Signifies the Y coordinate of the origin is to be translated to the Y coordinate of the following feature if F(lname1), FA(lname1), or DAT(x) is given.

ZORIG signifies that the current coordinate system origin is to be translated on the Z axis if a value is given. Signifies the Z coordinate of the origin is to be translated to the Z coordinate of the following feature if F(lname1), FA(lname1), or DAT(x) is given.

There are some implied limitations when translating to features. For example, it is not possible to translate the origin along an axis to a line that is parallel to the axis. Additionally, a limit of one translation along each axis is allowed for each TRANS statement. It is possible to translate the origin outside the travel limit of the DME.

A SAVE statement must be issued prior to the TRANS statement if the current part coordinate system is to be used again with the RECALL statement. The new part coordinate system is activated when the TRANS statement is executed. Refer to clause 5.3.6 and sub clauses.

The TRANS statement is passed to the output file at the time it is executed.

6.215 UNCERTALG

Function: Defines an uncertainty algorithm and assigns to it a label.

Input Formats:

can be: **U(lname)=UNCERTALG/var_1**

Output Formats:

can be: **U(lname)=UNCERTALG/var_1**
or: **UA(lname)=UNCERTALG/var_2**

Where:

var_1 can be: **ALGOR,n**
or: **'name' var_3**

var_2 can be: **SUPPORTED**
or: **UNSUPPORTED**

var_3 can be: **,DME var_4**
or: **,SENS var_4**
or: **var_4**

var_4 can be: **,parameter var_4**
or: **does not exist**

ALGOR signifies that the algorithm is being defined with a numeric code.

DME signifies that parameters specific to DME performance are to follow.

lname is an alphanumeric label name assigned to the algorithm.

n is a positive integer that represents a DME specific algorithm.

'name' is a text string, enclosed with apostrophes, identifying an algorithm name.

parameter can be a value or a variable.

SENS signifies that parameters specific to sensor performance are to follow.

SUPPORTED signifies that the DME supports the previously defined uncertainty algorithm.

UNSUPPORTED signifies that the DME does not support the previously defined uncertainty algorithm.

The UNCERTALG statement is passed to output when activated by the UNCERTSET statement.

Note: The characterization file contains the various uncertainty algorithms that are supported by the particular DME. Each of these algorithms is assigned an integer code in the characterization file.

6.216 UNCERTSET

Function: Activates an uncertainty algorithm and, optionally, a decision rule, or deactivates any that are currently activated.

Input Formats:

can be: **UNCERTSET/var_1**

Output Formats:

can be: **UNCERTSET/var_1**

Where:

var_1 can be: **ON,U(lname1),var_2**
or: **OFF**

var_2 can be: **DR(lname2)**
or: **NONE**

DR(lname2) is a label of a previously defined conformance decision rule.

NONE signifies that no decision rule is being activated.

OFF signifies that uncertainty algorithms and conformance decision rules are disabled.

ON signifies that the specified uncertainty algorithm and, optionally, the specified conformance decision rule are enabled.

U(lname1) is a label of a previously defined uncertainty algorithm.

The UNCERTSET statement is passed to output when executed. In addition, an UNCERTSET/ON statement will cause the corresponding UNCERTALG statement and, if appropriate, CNFRMRUL statement to be passed to output.

6.217 UNITS

Function: Specifies the units that will be active throughout the program.

Input Formats:

can be: **UNITS/var_1,var_2 var_3**

Output Formats:

can be: **UNITS/var_1,var_2 var_3**

Where:

var_1 can be: **MM**
or: **CM**
or: **METER**
or: **INCH**
or: **FEET**

var_2 can be: **ANGDEC**
or: **ANGDMS**
or: **ANGRAD**

var_3 can be: **,TEMPF**
or: **,TEMPC**
or: **does not exist**

ANGDEC signifies angles or angular tolerances in degree decimal form.

ANGDMS signifies angles or angular tolerances in degrees, minutes, and seconds with a colon ':' as a delimiter, for example: 04:03:47.00

ANGRAD signifies angles or angular tolerances in radian form.

CM signifies distance in centimeters.

FEET signifies distance in feet.

INCH signifies distance in inches.

METER signifies distance in meters.

MM signifies distance in millimeters.

TEMPC signifies temperature in degrees Celsius.

TEMPF signifies temperature in degrees Fahrenheit.

The UNITS statement can be issued multiple times in a program. Statements that explicitly specify a unit type, such as ACLRAT or FEDRAT, are not effected by the UNITS statement.

The UNITS statement is passed to the output file when executed.

6.218 VALUE

Function: Sets a DMIS variable equal to the value of a setting or property.

This function returns different data types, based on the particular information requested, so the result must be assigned to a previously declared variable of a proper type. For return values of type CHAR, if the length of the returned string is larger than the CHAR variable, the result will be truncated to the size of the variable.

Input Formats:

can be: **varname=VALUE/var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **ACLRAT var_2**
or: **BADTST**
or: **BOUND,var_5**
or: **CRMODE**
or: **CROSCL**
or: **CRSLCT**
or: **CZSLCT,lname1**
or: **DATSET**
or: **DEFLECTION**
or: **DMISMD,var_6**
or: **DMISMN,var_6**
or: **ERROR,var_7**
or: **FEDRAT var_8**
or: **FILNAM,var_6**
or: **FINPOS**
or: **GEOALG,var_11**
or: **GOTO,var_12**
or: **MODE**
or: **PRCOMP**
or: **PTBUFF**
or: **PTMEAS,var_12**
or: **SCNMOD**
or: **SNSET,var_13**
or: **SNSLCT**
or: **SNSMNT,var_14**
or: **TECOMP**
or: **UNITS,var_15**
or: **WKPLAN**
or: **FA(lname2),var_16**
or: **KC(lname3)**
or: **RT(lname4),var_18**
or: **SA(lname5) var_19**
or: **SW(lname6),var_20**
or: **TA(lname7),var_21**

var_2 can be: **,var_3**
or: **,var_4**

var_3 can be: **MESACL**
or: **POSACL**
or: **ROTACL**

var_4 can be: **MESACL,ACEL**
or: **POSACL,ACEL**
or: **ROTACL,ACEL**

var_5 can be: **F(lname8),COUNT**
or: **FA(lname9),COUNT**
or: **T(lname10),COUNT**
or: **TA(lname11),COUNT**
or: **F(lname8),bndnum**
or: **FA(lname9),bndnum**
or: **T(lname10),bndnum**
or: **TA(lname11),bndnum**

var_6 can be: **NAME**
or: **VERSION**

var_7 can be: **ERR**
or: **ERRMODE**

var_8 can be: **,var_9**
or: **,var_10**

```
 var_9 can be:  MESVEL
          or:  POSVEL
          or:  ROTVEL
          or:  SCNVEL

var_10 can be:  MESVEL,FEED
          or:  POSVEL,FEED
          or:  ROTVEL,FEED
          or:  SCNVEL,FEED

var_11 can be:  ARC
          or:  CIRCLE
          or:  CONE
          or:  CONRADSEGMNT
          or:  CPARLN
          or:  CYLNDR
          or:  CYLRADSEGMNT
          or:  ELLIPS
          or:  ELONGCYL
          or:  EDGEPT
          or:  GCURVE
          or:  GSURF
          or:  LINE
          or:  OBJECT
          or:  PARPLN
          or:  PLANE
          or:  RCTNGL
          or:  REVSURF
          or:  SPHERE
          or:  SPHRADSEGMNT
          or:  SYMPLN
          or:  TORRADSEGMNT
          or:  TORUS

var_12 can be:  XAXIS
          or:  YAXIS
          or:  ZAXIS
          or:  POS

var_13 can be:  APPRCH
          or:  RETRCT
          or:  SEARCH
          or:  CLRSRF
          or:  CLRSRF,DIST
          or:  DEPTH
          or:  DEPTH,DIST

var_14 can be:  XVEC
          or:  ZVEC
          or:  MNTLEN

var_15 can be:  DIST
          or:  ANGL
          or:  TEMP

var_16 can be:  PTDATA
          or:  SIZE var_17

var_17 can be:  ,sizenum
          or:  does not exist

var_18 can be:  ANGL,CW
          or:  ANGL,CCW
```

var_19 can be: **,'desc'**
or: **,tipnum**
or: **does not exist**

var_20 can be: **ANGLE,'anglename'**

var_21 can be: **var_22 var_23**

var_22 can be: **ACT**
or: **DEV**
or: **AMT**
or: **INTOL**
or: **OUTOL**

var_23 can be: **,tolnum**
or: **does not exist**

ACEL signifies that the function is to return the acceleration rate units indicated by var_4, as a type CHAR.

the value returned can be: 'MPMM'
or: 'MMPSS'
or: 'IPMM'
or: 'IPSS'
or: 'RPMM'
or: 'DEFALT'
or: 'HIGH'
or: 'LOW'
or: 'PCENT'.

ACLRAT signifies that the function is to return either the current acceleration rate or the current acceleration rate units.

When var_2 is var_3 the value returned is a type DOUBLE.

When var_2 is var_4 the value returned is a type CHAR.

ACT signifies that the function is to return the effective zone size (including any bonus) as a type REAL.

AMT signifies that the function is to return the amount of deviation beyond the tolerance limit as a type REAL.

ANGL,CW signifies that the function is to return the current angular setting in degrees, as a type REAL. The value returned is to be the absolute angle measured clockwise from 0.

ANGL,CCW signifies that the function is to return the current angular setting in degrees, as a type REAL. The value returned is to be the absolute angle measure counter-clockwise from 0.

ANGLE signifies that the function is to return the current angular setting of the named angle on the named wrist as a type REAL.

'anglename' is the name of a wrist angle, as it was defined in the WRIST statement.

APPRCH signifies that the function is to return the approach distance as a type DOUBLE.

ARC signifies that the current circular arc substitute feature algorithm is to be returned.

BADTST signifies that the function is to return the current setting of BADTST as a type BOOL (ON=.TRUE., OFF=.FALSE.).

bndnum is a positive integer number used to select the n'th bounding plane for the given feature or tolerance. The function is to return the label name of the n'th bounding plane or 'NONE', as a type CHAR.

BOUND signifies that the function is to return feature bounding information based on var_5.

CIRCLE signifies that the current circle substitute feature algorithm is to be returned.

CLRSRF	signifies that the function is to return the label name, as a type CHAR, of the clearance plane feature, or '' if no clearance plane feature is specified or if CLRSRF is OFF.
CLRSRF,DIST	signifies that the function is to return the distance as a type DOUBLE, or zero if CLRSRF is OFF.
CONE	signifies that the current cone substitute feature algorithm is to be returned.
CONRADSEGMNT	signifies that the current conical radial segment substitute feature algorithm is to be returned.
COUNT	signifies that the function is to return the number of bounding planes associated with the given feature or tolerance, as a type INTGR.
CPARLN	signifies that the current closed-ended parallel feature algorithm is to be returned.
CRMODE	signifies that the function is to return the current setting of DMIS statement execution, as one of ('SEQNTL', 'SIMUL', or 'SYNC') as a type CHAR.
CROSCL	signifies that the function is to return the current setting of CROSCL as a type BOOL (ON=.TRUE., OFF=.FALSE.).
CRSLCT	signifies that the function is to return the label name of the current carriage as a type CHAR: For a carriage common section, returns 'ALL' For a single carriage system returns '', if the carriage has not been defined with the CRGDEF statement.
CYLNDR	signifies that the current cylinder substitute feature algorithm is to be returned.
CYLRADSEGMNT	signifies that the current cylindrical radial segment substitute feature algorithm is to be returned.
CZSLCT	signifies that the function is to return the current setting of CZSLCT for the given label name as a type BOOL (ON=.TRUE., OFF=.FALSE.).
DATSET	signifies that the function is to return the label name of the current reference frame as a type CHAR.
DEFLECTION	signifies that the function is to return the value of the current probe displacement as a type DOUBLE. The function returns zero if the probe doesn't support displacement.
DEPTH	signifies that the function is to return the label name, as a type CHAR, of the depth plane feature, or '' if no depth plane feature is specified or DEPTH is OFF.
DEPTH,DIST	signifies that the function is to return the distance as a type DOUBLE, or zero if DEPTH is OFF.
'desc'	is a text string, enclosed with apostrophes, which is the tip description as defined in the SENSOR/MLTPRB statement.
DEV	signifies that the function is to return the tolerance deviation (actual-nominal) as a type REAL.
DMISMD	signifies that the function is to return the current DMIS module name or version as a type CHAR of the same length as that in the definition. Based on var_6.
DMISMN	signifies that the function is to return the current DMIS program name or version as a type CHAR of the same length as that in the definition. Based on var_6.

does not exist signifies that the function is to return various values dependent on the statement issued:

with ACLRAT;
returns the current measurement acceleration rate as a type DOUBLE, or zero for DEFALT, HIGH, LOW.

with FEDRAT;
returns the current measurement velocity as a type DOUBLE, or zero for DEFALT, HIGH, LOW.

with FA(lname),SIZE;
returns the first or only size component as a type DOUBLE.

with TA(lname);
to specify the first or only tolerance value as a type DOUBLE.

EDGEPT signifies that the current edge point substitute feature algorithm is to be returned.

ELLIPS signifies that the current ellipse substitute feature algorithm is to be returned.

ELONGCYL signifies that the current elongated cylinder substitute feature algorithm is to be returned.

ERR signifies that the function is to return the current handler as a type CHAR.

the value returned can be: 'label' - the statement label of an error handler.
or: 'OFF' - error handling is off.
or: 'AUTO' - error handling is automatic.

ERRMODE signifies that the function is to return the error handling type as a type CHAR.

the value returned can be: 'ALL'
or: 'ILLEGALTOUCH'
or: 'NOTOUCH'
or: 'errcode'
or: 'NONE'

ERROR signifies that current system error handeling information is to be returned based on the value of var_7.

F(lname8) is the label of the feature nominal, for which the bounding information will be returned.

FA(lname2) signifies that the function is to return the selected component of the feature actual as indicated by var_16.

FA(lname9) is the label of the feature actual, for which the bounding information will be returned.

FEDRAT signifies that the function is to return a feedrate value dependent on the value of var_8.

when var_8 is var_9 the value returned is a type DOUBLE.
when var_8 is var_10 the value returned is a type CHAR.

FILNAM signifies that the function is to return name or version of the current results file as a type CHAR of the same length as that in the definition. Based on var_6.

FINPOS signifies that the function is to return the current setting of FINPOS as a type BOOL (ON=.TRUE., OFF=.FALSE.).

GCURVE signifies that the current generic curve substitute feature algorithm is to be returned.

GEOALG signifies that the function is to return the current setting for the given feature type as a type CHAR.

the value returned can be: 'LSTSQR'
or: 'MINMAX'
or: 'MAXINS'
or: 'MINCIR'
or: 'BSPLN'
or: 'DEFALT'
or: 'EXTERN,DMIS, M(lname1), parameter list'
or: 'EXTERN,DME, routine, parameter list'
or: 'EXTERN,SYS, routine, parameter list'

GOTO signifies that the function is to return the current effective sensor position in PCS based on var_12.

GSURF signifies that the current generic surface substitute feature algorithm is to be returned.

INTOL signifies that the function is to return .TRUE. if the tolerance is within spec., else .FALSE. as a type BOOL.

KC(lname3) signifies that the function is to return the key characteristic's criticality designator as a type CHAR.

The value returned can be: 'CRITICAL'
or: 'MAJOR'
or: 'MINOR'
or: 'NONE'

lname1 is the label name of a collision zone.

LINE signifies that the current line substitute feature algorithm is to be returned.

MESACL signifies that the current setting for measurement acceleration rate or the current units for measurement acceleration rate is to be returned.

When var_2 is var_3 the value returned is a type DOUBLE.

When var_2 is var_4 the value returned is a type CHAR.

MESVEL signifies that the function is to return the current setting for measurement velocity as a type DOUBLE.

MESVEL,FEED signifies that the function is to return the current setting for measurement velocity units as a type CHAR.

the value returned can be: 'MPM'
or: 'MMPS'
or: 'IPM'
or: 'IPS'
or: 'RPM'
or: 'DEFALT'
or: 'HIGH'
or: 'LOW'
or: 'PCENT'

MNTLEN signifies that the function is to return the sensor mount offset as a type VECTOR in MCS.

MODE signifies that the function is to return the current DME execution mode as a type CHAR

the value returned can be: 'AUTO,PROG,MAN'
or: 'AUTO,MAN'
or: 'PROG,MAN'
or: 'MAN'

NAME signifies that the function is to return the filename, DMIS program name, or DMIS module name as a type CHAR of the same length as that in the definition.

OBJECT signifies that the current "user created" feature substitute feature algorithm is to be returned.

OUTOL signifies that the function is to return .TRUE. if the tolerance is out of spec, else .FALSE. as a type BOOL.

PARPLN signifies that the current feature of linear size (slot, block) substitute feature algorithm is to be returned.

PLANE signifies that the current plane substitute feature algorithm is to be returned.

POS signifies that the function is to return the current [x,y,z] location in PCS as a type VECTOR as indicated by var_12

POSACL signifies that the current setting for positioning acceleration rate or the current units for positioning acceleration rate is to be returned.

When var_2 is var_3 the value returned is a type DOUBLE.

When var_2 is var_4 the value returned is a type CHAR.

POSVEL signifies that the function is to return the current setting for positioning velocity as a type DOUBLE.

POSVEL,FEED signifies that the function is to return the current setting for positioning velocity units as a type CHAR.

the value returned can be: 'MPM'
or: 'MMPS'
or: 'IPM'
or: 'IPS'
or: 'RPM'
or: 'DEFALT'
or: 'HIGH'
or: 'LOW'
or: 'PCENT'

PRCOMP signifies that the function is to return the current setting of PRCOMP as a type BOOL (ON=.TRUE., OFF=.FALSE.).

PTBUFF signifies that the function is to return the current setting of PTBUFF as a type BOOL (ON=.TRUE., OFF=.FALSE.).

PTDATA signifies that the function is to return the number of sample points for this feature as a type INTGR.

PTMEAS signifies that the function is to return the actual location of the last measurement point in PCS based on var_12.

RCTNGL signifies that the current rectangle substitute feature algorithm is to be returned.

RETRCT signifies that the function is to return the retract distance as a type DOUBLE.

REVSURF signifies that the current surface-of-revolution substitute feature algorithm is to be returned.

ROTACL signifies that the current setting for rotary table acceleration rate or the current units for rotary table acceleration rate is to be returned.

When var_2 is var_3 the value returned is a type DOUBLE.

When var_2 is var_4 the value returned is a type CHAR.

ROTVEL signifies that the function is to return the current setting for rotary table velocity as a type DOUBLE.

ROTVEL,FEED signifies that the function is to return the current setting for rotary table velocity units as a type CHAR.

the value returned can be: 'MPM'
or: 'MMPS'
or: 'IPM'
or: 'IPS'
or: 'RPM'
or: 'DEFALT'
or: 'HIGH'
or: 'LOW'
or: 'PCENT'

RT(lname4) signifies that the function is to return the selected component of the rotary table as indicated by var_18.

SA(lname5) signifies that the function is to return the size of the selected component of a sensor actual as indicated by var_19 as a type REAL.

SCNMOD signifies that the function is to return the current setting of SCNMOD as a type BOOL (ON=.TRUE., OFF=.FALSE.).

SCNVEL signifies that the function is to return the current setting for scanning velocity as a type DOUBLE.

SCNVEL,FEED signifies that the function is to return the current setting for scanning velocity units as a type CHAR.

the value returned can be: 'MPM'
or: 'MMPS'
or: 'IPM'
or: 'IPS'
or: 'RPM'
or: 'DEFALT'
or: 'HIGH'
or: 'LOW'
or: 'PCENT'

SEARCH signifies that the function is to return the search distance as a type DOUBLE.

SIZE signifies that the function is to return the feature size (diameter, radius or angle) as a type REAL.

sizenum is a positive integer number used to select the n'th size component of a feature based on the order in the feature definition. The function is to return the n'th size component of a feature based on the order in the feature definition.

SNSET signifies that the function is to return the current specified sensor setting based on var_13.

SNSLCT signifies that the function is to return the label name of the currently active sensor as a type CHAR.

SNSMNT signifies that the function is to return the selected component of the sensor mount based on var_14.

SPHERE signifies that the current sphere substitute feature algorithm is to be returned.

SPHRADSEGMNT signifies that the current spherical radial segment substitute feature algorithm is to be returned.

SW(lname6) signifies that the function is to return the angle of a sensor wrist as a type REAL . The sensor wrist is identified by lname6, and the angle is identified by var_20. When DMS angle units are being used the value of the angle is in decimal degrees.

SYMPLN signifies that the current feature of linear size (slot, block) substitute feature algorithm is to be returned.

T(lname10) is the label of the tolerance nominal, for which the bounding information will be returned.

TA(lname11) is the label of the tolerance actual, for which the bounding information will be returned

TA(lname7) signifies that the function is to return the selected component of the tolerance actual as indicated by var_21.

TECOMP signifies that the function is to return the current setting of TECOMP as a type BOOL (ON=.TRUE., OFF=.FALSE.).

tipnum is a positive integer value of the tip number as defined in the SENSOR/MLTPRB statement.

tolnum is a positive integer number used to select the n'th specified tolerance based on the order in the tolerance definition. (ex. 1=lowtol, 2=uptol). If not specified the first value will be returned.

TORRADSEGMNT signifies that the current toroidal radial segment substitute feature algorithm is to be returned.

TORUS signifies that the current torus substitute feature algorithm is to be returned.

UNITS,ANGL signifies that the function is to return the current system setting for angular units as a type CHAR.

the value returned can be: 'ANGDEC'
or: 'ANGDMS'
or: 'ANGRAD'.

UNITS,DIST signifies that the function is to return the current system setting for linear units as a type CHAR.

the value returned can be: 'MM'
or: 'CM'
or: 'M'
or: 'INCH'
or: 'FEET'.

UNITS,TEMP signifies that the function is to return the current system setting for temperature units as a type CHAR.

the value returned can be: 'TEMPF'
or: 'TEMPC'.

varname is the name of the previously declared variable to which the value is assigned.

VERSION signifies that the function is to return the DMIS version number as a type CHAR of the same length as that in the definition, or 'NONE'.

WKPLAN signifies that the function is to return the current setting of WKPLAN as a type CHAR.

the value returned can be: 'XYPLAN'
or: 'YZPLAN'
or: 'ZXPLAN'

XAXIS signifies that the function is to return the current x location in PCS as a type REAL as indicated by var_12.

XVEC signifies that the function is to return the sensor coordinate system X axis as a type VECTOR in the DME's machine coordinate system.

YAXIS signifies that the function is to return the current y location in PCS as a type REAL as indicated by var_12.

ZAXIS signifies that the function is to return the current z location in PCS as a type REAL as indicated by var_12.

ZVEC signifies that the function is to return the sensor coordinate system Z axis as a type VECTOR in the DME's machine coordinate system.

Variables must be declared prior to using the VALUE statement with the DECL statement. The VALUE statement will evaluate the tolerance statement(s) in the last EVAL and/or OUTPUT statement.

The ' tolnum' parameter is typically used to gain access to either the first or second tolerance value for a composite tolerance (for example, COMPOS, CPROFS). If ' tolnum' is not specified the value returned is the first tolerance value.

6.219 VFORM

Function: Specifies the data content of vendor output, and assigns a label to it.

Input Formats:

can be: **V(lname)=VFORM/var_1 var_2**

Output Formats:

can be: **Activated with the DISPLY statement**

Where:

var_1 can be: **NOM var_3**
or: **ACT var_3**
or: **DEV var_3**
or: **AMT var_3**
or: **HIST var_3**
or: **PLOT var_3**
or: **STAT var_3**
or: **ALL**
or: **DME var_3**

var_2 can be: **,var_1 var_2**
or: **does not exist**

var_3 can be: **,'text'**
or: **does not exist**

ACT signifies that the actual measured values for features and tolerances will be output.

ALL signifies that all DME-supported output data is output.

AMT signifies the out of tolerance amount to be output. (= actual - (nominal + tolerance)).

DEV signifies that deviations from nominal conditions will be output.

DME signifies additional vendor specific instructions for formatting the report.

HIST signifies data content in histogram form.

lname is an alphanumeric label name assigned to the vendor output data specification.

NOM signifies that the nominals for features and tolerances will be output.

PLOT signifies data content in plot or graphics form.

STAT signifies data content in statistical form.

'text' is a DME specific vendor format string enclosed with apostrophes that is used to enable reporting capabilities that are not within the capability of the DMIS interface.

VFORM signifies that the vendor output data content is being specified.

The V(lname)=VFORM statement is used with the DISPLY statement.

6.220 WINDEF (input format 1)

Function: Defines a window used to limit the area of a video sensor's field of view, and assigns a label to it.

Input Formats:

can be: **VW(lname)=WINDEF/EDGELN,x,y,z,ang,len**

Output Formats:

can be: **None**

Where:

ang is the angle of the line relative to the positive X axis of the current part coordinate system. Note that it also provides the orientation for the scanning direction relative to the line's origin. Apply the right hand rule for sign convention.

EDGELN signifies that an edge line will be defined and used for measurements.

lname is an alphanumeric label name assigned to the window.

len is a positive real number representing the total length of the edge line equally disposed about the edge line's origin.

x,y,z are the X,Y,Z, coordinates of the center of the line. This point is referred to as the line's origin.

Note 1: A typical PTMEAS point would coincide with the edge line origin. Algorithms defined with the ALGDEF statement, and activated with the SNSET statement, can be used to specify applications for the edge line, for example counting intersections, or evaluating intersections, or both.

Note 2: Refer to (Figure B.85 — Edge line orientation and applications) for an illustration of the video edge line orientation and applications.

6.221 WINDEF (input format 2)

Function: Defines a window used to limit the area of a video sensor's field of view, and assigns a label to it.

Input Formats:

can be: **VW(lname)=WINDEF/BOX,x,y,z,dx,dy,ang**

Output Formats:

can be: **None**

Where:

ang is the angle of the box window axis relative to the positive X axis of the current part coordinate system. Note that it also provides the orientation for the scanning direction relative to the line's origin. Apply the right hand rule for sign convention.

BOX signifies that a rectangular shaped window is defined.

dx,dy are positive real numbers, where dx is the width of the box in the X direction and dy is the length of the box in the Y direction, equally disposed about the origin (center) of the box.

lname is an alphanumeric label name assigned to the window.

x,y,z are the X,Y,Z, coordinates for the origin of the box. The origin of the box is the center of the box.

Note 1: A typical PTMEAS point would coincide with the origin of the window. Algorithms defined with the ALGDEF statement, and activated with the SNSET statement, can be used to specify applications for the BOX window, for example, maximum or minimum points, and line fits, etc.

Note 2: Refer to (Figure B.86 — Video box window orientation and applications) and (Figure B.87 — Box window applications) for illustrations of the orientation and applications of the BOX window statements.

6.222 WKPLAN

Function: Used to explicitly declare or change a working plane.

Input Formats:

can be: **WKPLAN/var_1**

Output Formats:

can be: **WKPLAN/var_1**

Where:

var_1 can be: **XYPLAN**
or: **YZPLAN**
or: **ZXPLAN**

XYPLAN signifies the XY plane of the current part coordinate system is the working plane.

YZPLAN signifies the YZ plane of the current part coordinate system is the working plane.

ZXPLAN signifies the ZX plane of the current part coordinate system is the working plane.

This plane is required for determining polar coordinates and the start angle for FEAT/ARC (input format 1). It is in effect until a new working plane is changed. Refer to (Figure B.88 — The working plane) and (Figure B.89 — Work Plane change as a result of a new DATSET statement execution) .

The WKPLAN statement is passed to the output file when executed.

6.223 WRIST

Function: Defines an articulating wrist sensor component, and assigns a label to it.

Input Formats:

can be: **SW(lname)=WRIST/var_1,var_2 var_3,MNTLEN,ex,ey,ez var_4**

Output Formats:

can be: **None**

Where:

var_1 can be: **ROTCEN,tx,ty,tz,ai,aj,ak,di,dj,dk**

var_2 can be: **ANGLE,'anglename',var_5,var_6**

var_3 can be: **,var_1,var_2 var_3**
or: **does not exist**

var_4 can be: **,'data_store','data_list',var_7**
or: **does not exist**

var_5 can be: **begin,end**
or: **THRU**

var_6 can be: **step**
or: **CONTIN**

var_7 can be: **'data_item'**
or: **index**

ai,aj,ak is the axis of rotation in the current sensor coordinate system.

ANGLE signifies that rotation angle data is to follow.

'anglename' is the name identifying the angle of rotation, enclosed with apostrophes. Each angle name on a wrist must be unique.

begin is the beginning angle of rotation for the axis.

CONTIN signifies that the wrist can move to any angle in the specified range.

'data_item' is the name of a data item from a data list, enclosed with apostrophes.

'data_list' is the name of a section within a DME Hardware Descriptor that contains one or more data items, enclosed with apostrophes.

'data_store' is a file or database called a "DME Hardware Descriptor", that contains DME specific information that may be needed by a DMIS program, enclosed with apostrophes. The DMIS program can get access to a specific item using three names (data_store, data_list, and data_item), which is compatible with the way items within operating system initialization files are accessed.

di,dj,dk is the zero angle direction in the current sensor coordinate system.

end is the end angle of rotation for the axis.

ex,ey,ez is the offset from the final axis of rotation to the mounting point for the subsequent sensor component in the current sensor coordinate system.

index is a positive integer number used to specify the position of the data within the data list.

lname is an alphanumeric label name assigned to the wrist.

MNTLEN signifies that the offset from the final axis of rotation to the mounting point for the subsequent sensor component will follow.

ROTCEN signifies that rotation axis and center data is to follow.

step is the angular step of an indexable wrist.

THRU signifies that the wrist can move through any supported angle in either direction for the axis.

tx,ty,tz is the offset to the center of rotation in the current sensor coordinate system.

The axes of rotation must be described in the order of the kinematic chain from the point at which the wrist mounts to a previous sensor component to the point at which subsequent sensor components mount to the wrist. All rotation axis definitions are relative to the zero angle rotations of all previous rotation axes in the dynamical chain.

Note: Refer to A.30 and (Figure A.6 — Example wrist definition values) for an illustration of and sample syntax for wrist definitions.

6.224 WRITE

Function: Writes data from the program to a system device or file.

Input Formats:

can be: **WRITE/DID(lname1),var_1**

Output Formats:

can be: **None**

Where:

var_1 can be: **variable:x:y var_2**
or: **variable:x var_2**
or: **variable var_2**
or: **'text' var_2**

var_2 can be: **,var_1**
or: **does not exist**

DID(lname1) is the device identification label assigned by the DEVICE statement to a system device.

'text' is a printable text string, enclosed with apostrophes, sent to a system device.

variable is the variable name.

x is a positive integer representing the value of the total field width (including any negative sign).

y is a positive integer representing the number of digits to the right of the decimal point.

The variable name must be previously defined using the DECL statement. The completion of a DMIS WRITE statement concludes the current record or line with a CR (carriage return) and LF (line feed).

A variable field width is optional. Its effect depends on the data type of the variable to which it applies. If the field type is numeric and it exceeds the total field width 'x', then DMIS writes a field of asterisks to denote an overflow condition. Numbers are padded with leading spaces. If the field type is a character string and it exceeds the total field width 'x', then DMIS truncates to the first 'x' characters of the string. Character strings are left justified.

The field width of the variable names can be defined as follows:

Type	Field width	Format
INTGR	System default	var_n :x
LONG	System default	var_n :x
REAL	System default	var_n :x:y
DOUBLE	System default	var_n :x:y
BOOLEAN	System default	var_n
CHAR, n	Declared width	var_n :x

UTF8 codes can be included in the write statement to control output format by using the DMIS intrinsic function CHR(x). These include carriage return (CR), line feed (LF), and form feed (FF).

Note: Refer to example A.36 and (Figure A.12 — Results of writing to a line printer) for an illustration and sample code used to transfer data from the program to a system device or file.

6.225 XTERN

Function: Starts an external declaration definition block. The XTERN...ENDXTN external declaration block is used to declare programs, scripts, and macros that are used within the local DMIS program but are defined in other external files.

Input Formats:

can be: **XTERN**

Output Formats:

can be: **None**

The external declaration definition is terminated with an ENDXTN statement.

6.226 XTRACT

Function: Provides the ability to extract bounded DMIS features from DMIS FEAT/GCURVE or FEAT/GSURF features.

Input Formats:

can be: **XTRACT/F(lname1),FA(lname2)**

Output Formats:

can be: **None**

Where:

F(lname1) is the label of the previously defined bounded feature nominal (for example, FEAT/ARC, FEAT/LINE, FEAT/PLANE, FEAT/GCURVE or FEAT/GSURF, etc) to be extracted.

FA(lname2) is the label of the measured/scanned generic curve (FEAT/GCURVE) or generic surface (FEAT/GSURF) feature actual.

When F(lname1) is a point, the actual point extracted is the closest interpolated point on the curve or surface to the nominal definition.

This statement cannot be issued within a feature MEAS...ENDMES block. POINT, ARC, LINE and GCURVE features can be extracted from a GCURVE feature. POINT, PLANE, GSURF, CONRADSEGMNT, CYLRADSEGMNT, SPHRADSEGMNT,and TORRADSEGMNT, features can be extracted from a GSURF feature.

According to clause 5.3.2.6, if a feature has been measured with point buffering enabled, by a PTBUFF/ON statement, then its individual point data can be referenced by appending the [n] option to F(lname) or FA(lname), where n is the point index from the feature's programmed PTMEAS statements. When the index [n] option for specifying point data is used, the current probe compensation state shall determine whether probe compensation will be applied to the specified data points during the feature construction, as if the constructed feature were measured with that point data with the MEAS or RMEAS statements. If the best fit feature is constructed with point buffering enabled, by a PTBUFF/ON statement, then the individual point data is copied to the point buffer of the constructed feature and is then accessible with the [n] point data index as if it were a feature measured with the MEAS or RMEAS statements. Non-indexed features (specified with FA(lname2) without the [n] index option) and indexed feature point data (specified with FA(lname2)[n] or FA(lname2)[n,m]) cannot be mixed in a single XTRACT statement.

Annex A
(informative)

DMIS example code segments

Note: Some DMIS code may be left out of examples for brevity.

A.1 @ character, and use with a label.

```
DECL/INTGR,i
DECL/CHAR,64,tname,fname[2,5]
i=ASSIGN/3
tname=ASSIGN/CONCAT('tol',STR(i))
fname[1,1]=ASSIGN/CONCAT('feat',STR(i))
F(@fname[1,1])=FEAT/GSURF
T(@tname)=TOL/PROFS,-0.1,0.1
OUTPUT/F(@fname[1,1]),T(@tname)

$$ yields the following output:

OUTPUT/F(feat3),T(tol3)
F(feat3)=FEAT/GSURF
T(tol3)=TOL/PROFS,-0.1,0.1
```

A.2 ASSIGN

An example of the use of the ASSIGN statement is as follows:

```
DECL/GLOBAL,REAL,PI,Num[10]
DECL/GLOBAL,BOOL,Quit,Chek
DECL/GLOBAL,CHAR,6,EmplID
PI=ASSIGN/3.1416
EmplID=ASSIGN/'03548'
Num[1]=ASSIGN/2 * PI
$$ assigns 6.2832 to array Num[1]
Quit=ASSIGN/.FALSE.
Chek=ASSIGN/Num[1] .EQ. PI
$$ assigns .FALSE.  to Chek
```

A.3 BADTST

```
DECL/GLOBAL,REAL,XX
BADTST/ON
GOTO/10,10,20
IF/(BADGT())
        TEXT/OPER,'An error occurred during the CNC movement'
        TEXT/OPER,'Move the carriage in a clearance position'
        TEXT/OPER,'Press ENTER to go back into AUTO MODE'
        MODE/MAN
        TEXT/QUERY,(OPER_RES),2,PRNTCHAR,LEFT, 'Move the Probe'
        MODE/AUTO,PROG,MAN
ENDIF
XX=ASSIGN/10
(CHECKPOINT)
```

```
F(POI_1)=FEAT/POINT,CART,XX,10,10,0,0,1
MEAS/POINT,F(POI_1),1
PTMEAS/CART,XX,10,10
ENDMES
$$ check to see if the point feature was successfully measured.
$$ if not move over 5 in X and try again
IF/(BADPT())
        XX=ASSIGN/XX+5
        JUMPTO/(CHECKPOINT)
ENDIF
$$ The point feature was successfully measured
BADTST/OFF
```

A.4 CALL

A.4.1 Example 1 of CALL

Example of the external routine call are as follows:

```
XTERN
$$ Verifies that the /dmis/plane_fits and /dmis/datum_fits DMISMD file
$$ containing DMIS macros exists.  It also defines the DMISMD file search
$$ path for any called DMIS macros.
EXTFIL/DMIS,'/dmis/plane_fits'
EXTFIL/DMIS,'/dmis/datum_fits'
$$ Verifies that the DME file, which may be an execution file or it may
$$ contain DME routines, exists.  It also may define the DME file search
$$ path.  This is dependent upon DME vendor implementation.
EXTFIL/DME,'/user/subroutines'
EXTFIL/DME,'/dme/subroutines'
ENDXTN
CALL/EXTERN,DMIS,M(HighPtPlane),arg1
  executable statements
CALL/EXTERN,DME,'DME_SCRIPT',WAIT
  executable statements
CALL/EXTERN,SYS,'/user/cmm/algs/numcrunch.exe',CONT,A,B,(CIR1),7,'MMC_CIR1'
  executable statements
CALL/EXTERN,DME,'my_linkable_script'
```

The above example declares six external files and executes four external routines.

Macro HighPtPlane is a DMIS macro passing one parameter that should be defined in the file '/dmis/plane_fits' or '/dmis/datum_fits'. It will search for the macro in definition file order, (that is, search through '/dmis/plane_fits' first).

Routine DME_SCRIPT is a DME language subroutine that should be found in either '/user/subroutines' or '/dme/subroutines' file. The main program will wait until completion of this external. No parameters are passed to it.

The program '/user/cmm/algs/numcrunch' will be started. It has five parameters passed to it; two variables, one label, one number, and a character string.

The last external call deals with a user created DME routine that was linked with the DME's source. As a result, an EXTFIL definition would not be appropriate however this routine still requires an external call for execution.

A.4.2 Example 2 of CALL

```
$$
DMISMN/'ExternMacroFile',05.1
XTERN
EXTFIL/DMIS,'dmis/macros.dms'
EXTFIL/DMIS,'dmis/anothermacros.dms'
EXTFIL/DME,'dme/routines.dme'
ENDXTN
$$
$$ This calls the macro supermac that may be declared in the XTERN ...
$$ ENDXTN files
CALL/EXTERN,DMIS,M(supermac)
$$
$$ This runs everything that is contained in DMISMD
CALL/EXTERN,DMIS,'AnotherExternDMISMacros'
$$
$$ This calls a DME routine
CALL/EXTERN,DME,'supercrunch'
```

executable statements

```
ENDFIL
$$
```

=====File dmis/macros.dms======

```
DMISMD/'ExternDMISMacros',05.1
M(supermac)=MACRO/...
```

executable statements

```
ENDMAC
$$
M(hypermac)=MACRO/ ...
```

executable statements

```
ENDMAC
ENDFIL
$$
```

=====File dmis/anothermacros.dms======

```
DMISMD/'AnotherExternDMISMacros',05.1
```

executable statements

```
ENDFIL
$$
```

=====File dme/routines.dme======

```
Procedure supercrunch()
{
...
}
```

A.5 CLOSE

An example of how the CLOSE statement may be used to close the output file follows:

```
CLOSE/DID(sensor_dat),KEEP
```

A.6 CONST (input format 8)

```
F(PLA_1)=FEAT/PLANE,CART,100,100,100,0,0,1
F(POI_1)=FEAT/POINT,CART,200,200,110,0,0,1
F(POI_2)=FEAT/POINT,CART,240,240,130,0,0,1
F(POI_3)=FEAT/POINT,CART,300,300,80,0,0,1
MEAS/POINT,F(POI_1),1
ENDMES
MEAS/POINT,F(POI_2),1
ENDMES
MEAS/POINT,F(POI_3),1
ENDMES
CONST/PLANE,F(PLA_1),OFFSET,FA(POI_1),FA(POI_2),FA(POI_3)
```

A.7 CONST (input format 9) and CONST (input format 10)

The inspection program.

Refer to (Figure A.1 — The drawing requirements) and (Figure A.2 — The gauge).

```
DMISMN/'Soft Gauge Example',05.1
FILNAM/'Soft Gauge Example',05.1
DV(aversion) = DMESWV/'5,3,0,7'
UNITS/MM,ANGDEC
V(0)=VFORM/ALL
DISPLY/TERM,V(0),STOR,DMIS,V(0)
TECOMP/MACH,OFF
TECOMP/PART,OFF
SNSET/APPRCH,5
SNSET/CLRSRF,15
SNSET/DEPTH,0
D(0)=DATSET/MCS
MODE/MAN
T(0)=TOL/CORTOL,XAXIS,-0.1,0.1
T(1)=TOL/CORTOL,YAXIS,-0.1,0.1
T(2)=TOL/CORTOL,ZAXIS,-0.1,0.1
T(3)=TOL/DIAM,-0.1,0.1
S(0)=SNSDEF/PROBE,INDEX,POL,0,0,0,0,1,127,2
SNSLCT/SA(0)
MODE/AUTO,PROG,MAN
F(PL0)=FEAT/PLANE,CART,30,10,0,0,0,1
MEAS/PLANE,F(PL0),3
PTMEAS/CART,75,-15,0,0,0,1
PTMEAS/CART,25,30,0,0,0,1
PTMEAS/CART,-15,-10,0,0,0,1
ENDMES
SNSET/DEPTH,2
F(CR0)=FEAT/CIRCLE,INNER,CART,0,0,0,0,0,1,20
MEAS/CIRCLE,F(CR0),4
ENDMES
F(CR1)=FEAT/CIRCLE,INNER,CART,60,0,0,0,0,1,20
MEAS/CIRCLE,F(CR1),4
ENDMES
F(LN0)=FEAT/LINE,UNBND,CART,30,0,0,1,0,0,0,-1,0
CONST/LINE,F(LN0),BF,FA(CR0),FA(CR1)
DATDEF/FA(PL0),DAT(A)
DATDEF/FA(LN0),DAT(B-C)
DATDEF/FA(CR0),DAT(B)
DATDEF/FA(CR1),DAT(C)
```

```
D(Part)=DATSET/DAT(A),ZDIR,ZORIG,DAT(B-C),XDIR,DAT(B),XORIG,YORIG
F(CR2)=FEAT/CIRCLE,INNER,CART,30,0,0,0,0,1,12
MEAS/CIRCLE,F(CR2),4
ENDMES

$$ Define the Soft Gauge, note the nominals contain the gauging tolerance

F(PL0)=FEAT/PLANE,CART,30,10,0,0,0,1
F(CR0)=FEAT/CIRCLE,INNER,CART,0,0,0,0,0,1,19.95
F(CR1)=FEAT/CIRCLE,INNER,CART,60,0,0,0,0,1,11.8
F(CR2)=FEAT/CIRCLE,INNER,CART,30,0,0,0,0,1,19.95

CONST/SGAGE,SE(GAG1),F(PL0),F(CR0),F(CR1),F(CR2)

$$ Construct the Soft Part

CONST/SPART,ST(PART1),FA(PL0),FA(CR0),FA(CR1),FA(CR2)

$$ Define the Gauge Tolerance

T(TGAG1)=TOL/GTOL,XYDIR,ZAXIS,10,0,0,INTFPT

$$ Evaluate the tolerance

OUTPUT/ST(PART1),SE(GAG1),TA(TGAG1)

ENDFIL

$$ Example output

OUTPUT/ST(PART1),SE(GAG1),TA(TGAG1)
TA(TGAG1)=TOL/GTOL,10,OUTOL,TRMATX,x,y,z,i1,j1,k1,i2,j2,k2,i3,j3,k3
```

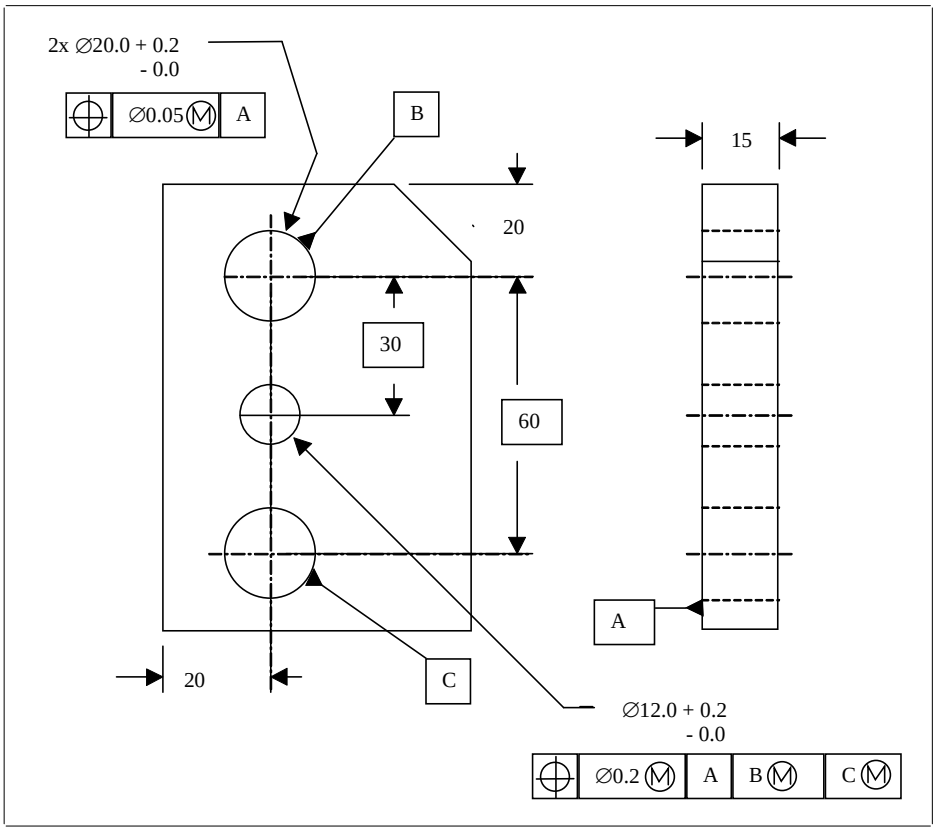


Figure A.1 — The drawing requirements

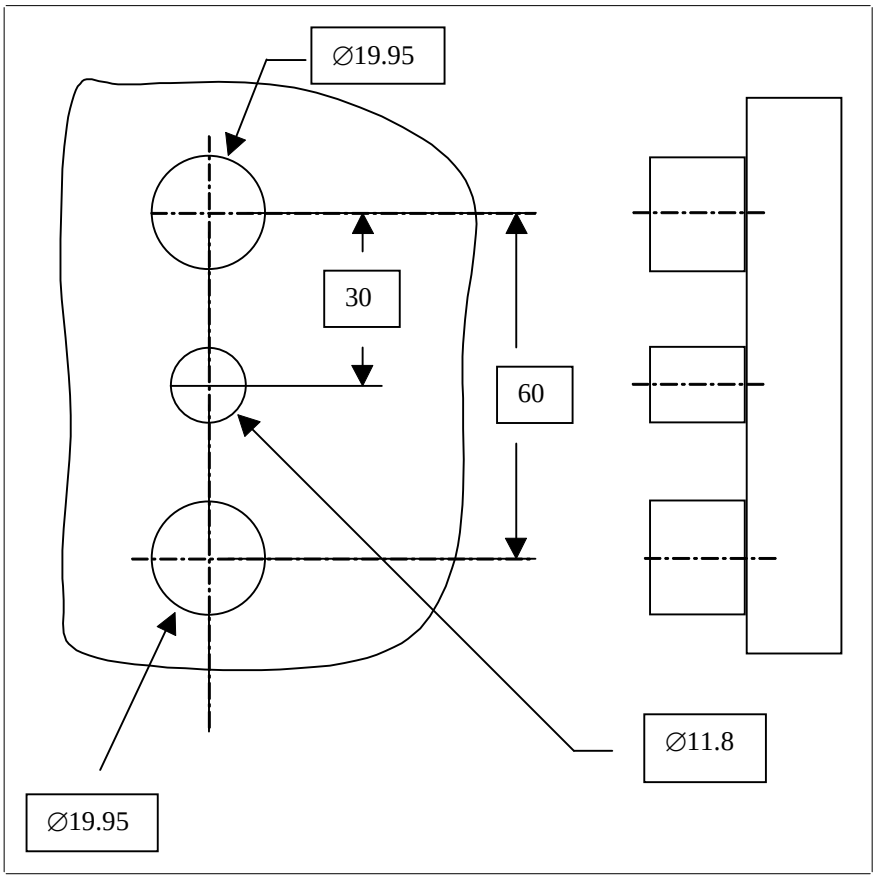


Figure A.2 — The gauge

A.8 CRMODE

A.8.1 CRMODE/SEQNTL

```
CRMODE/SEQNTL
CRSLCT/CR(ARM1)
MEAS/POINT,F(PT1),1
PTMEAS/CART,1,2,3,0,1,0
ENDMES
MEAS/POINT,F(PT3),1
PTMEAS/CART,2,2,3,0,1,0
ENDMES
CRSLCT/CR(ARM2)
MEAS/POINT,F(PT2),1
PTMEAS/CART,1,2,3,0,-1,0
ENDMES
MEAS/POINT,F(PT4),1
PTMEAS/CART,2,2,3,0,-1,0
ENDMES
```

Note: Points PT1 and PT3 are measured by ARM1, then points PT2 and PT4 are measured by ARM2.

A.8.2 CRMODE/SIMUL

```
CRMODE/SIMUL
```

```
CRSLCT/CR(ARM1)
MEAS/POINT,F(PT1),1
PTMEAS/CART,1,2,3,0,1,0
ENDMES
MEAS/POINT,F(PT3),1
PTMEAS/CART,2,2,3,0,1,0
ENDMES
CRSLCT/CR(ARM2)
MEAS/POINT,F(PT2),1
PTMEAS/CART,1,2,3,0,-1,0
ENDMES
MEAS/POINT,F(PT4),1
PTMEAS/CART,2,2,3,0,-1,0
ENDMES
```

Note: Points PT1 and PT3 are measured by ARM1 while points PT2 and PT4 are being measured by ARM2, without any type of synchronization.

A.8.3 CRMODE/SYNC

```
CRMODE/SYNC
CRSLCT/CR(ARM1)
MEAS/POINT,F(PT1),1
PTMEAS/CART,1,2,3,0,1,0
ENDMES
MEAS/POINT,F(PT3),1
PTMEAS/CART,2,2,3,0,1,0
ENDMES
CRSLCT/CR(ARM2)
MEAS/POINT,F(PT2),1
PTMEAS/CART,1,2,3,0,-1,0
ENDMES
MEAS/POINT,F(PT4),1
PTMEAS/CART,2,2,3,0,-1,0
ENDMES
```

Note: At the same time that ARM1 is measuring PT1, ARM2 is measuring PT2. The same logic is then applied to PT3 and PT4.

A.9 CZSLCT

```
$$ ************************************************
$$ Example DMIS program to illustrate multiple carriage handling.
$$ ************************************************
$$
$$ Common Elements
$$
cr(arm1)=crgdef/0,0,0,20,10,10,1,0,0,0,1,0,0,0,-1,0,1,0
cr(arm2)=crgdef/0,7,0,20,10,10,1,0,0,0,1,0,0,0,-1,0,-1,0
$$
crslct/all
v(v1)=vform/all
disply/print,v(v1),stor,dmis
filnam/'example1.dmo',05.1
units/inch,angdec
$$
$$ Sensor Definitions
$$
$$ Attach sensors to carriages
$$
```

```
crslct/cr(arm1)
s(probe_1)=snsdef/probe,fixed,cart,-4,0,0,-1,0,0,.1
snslct/s(probe_1)
$$
crslct/cr(arm2)
s(probe_2)=snsdef/probe,fixed,cart,-4,0,0,-1,0,0,.1
snslct/s(probe_2)
$$
$$ Example of 2 arms measuring same feature.
$$
crslct/all
f(plane_A)=feat/plane,cart,0,0,0,0,0,1
meas/plane,f(plane_A),4
crslct/cr(arm1)
goto/2,2,1
ptmeas/cart,2,2,0,0,0,1
goto/37,2,1
ptmeas/cart,37,2,0,0,0,1
goto/37,2,1
crslct/cr(arm2)
goto/2,23,1
ptmeas/cart,2,23,0,0,0,1
goto/37,23,1
ptmeas/cart,37,23,0,0,0,1
goto/37,23,1
endmes
output/fa(plane_A)
$$
$$ Example of collision zone selection
$$
crslct/all
f(cir_1)=feat/circle,inner,cart,35,6,-.5,0,0,1,7
cz(path_1)=czone
meas/circle,f(cir_1),4
crslct/cr(arm1)
goto/35,6,1
ptmeas/cart,38,6,-.5,-1,0,0
czslct/cz(path_1),on
ptmeas/cart,35,10,-.5,0,-1,0
ptmeas/cart,31,6,-.5,1,0,0
czslct/cz(path_1),off
ptmeas/cart,35,3,-.5,0,1,0
goto/35,6,1
endmes
t(pos_1)=tol/pos,2d,.005,rfs
t(diam_1)=tol/diam,-.001,.001
output/fa(cir_1),ta(pos_1),ta(diam_1)
crslct/all
f(cir_2)=feat/circle,inner,cart,35,15,-.5,0,0,1,7
meas/circle,f(cir_2),4
crslct/cr(arm2)
goto/35,15,1
ptmeas/cart,38,15,-.5,-1,0,0
ptmeas/cart,35,19,-.5,0,-1,0
ptmeas/cart,31,15,-.5,1,0,0
czslct/cz(path_1),on
ptmeas/cart,35,12,-.5,0,1,0
goto/35,12,1
goto/29,19,1
czslct/cz(path_1),off
endmes
```

```
output/fa(cir_2),ta(pos_1),ta(diam)
$$
endfil
```

A.10 DEVICE

An example of the use of the DEVICE statement to identify devices are as follows:

```
DID(sensor_dat)=DEVICE/STOR,'user/dmis/sensors/sensor52489'
DID(DRFsdata)=DEVICE/STOR,'user/dmis/drfs/partlO988'
DID(FloorPtr)=DEVICE/PRINT,'8'
DID(sys_term)=DEVICE/TERM,'petra:0.0'
DID(SummaryDat)=DEVICE/STOR,'/user/dmis/insp_summary/part81482'
DID(HeaderInfo)=DEVICE/STOR,'/user/dmis/header/IPPEX_header'
DID(Displayptr)=DEVICE/PRINT,'LPT1'
DID(data)=DEVICE/STOR,'gendata.dat '
DID(incdata)=DEVICE/INCR,'file???.inc'
```

A.11 DMISMN

Table A.1 — System file name example

(DMIS INPUT PROGRAM)	ANC101.DMI

`DMISMN/'ANC101 TEST PROGRAM',05.1`
`V(vendor)=VFORM/ALL`
`DISPLY/PRINT,DMIS,STOR,V(vendor)`
`FILNAM/'ANC101 Test Program Results',05.1`
`XTERN`
`EXTFIL/DMIS,'/usr/dme/mydmismods/9w3309.SUB'`
`ENDXTN`
`CALL/EXTERN,DMIS,' Bolt_Circle_Routine'`
`ENDFIL`	

(DMIS MODULE)	9W3309.SUB
`DMISMD/'Bolt_Circle_Routine',05.1` `ENDFIL`	
(DMIS OUTPUT FILE)	ANC101.DMO

`FILNAM/'ANC101 Test Program Results',05.1`
`CALL/EXTERN,DMIS,'Bolt_Circle_Routine'`
`DMISMD/'Bolt_Circle_Routine',05.1`
`ENDFIL`	

Note: This string of UTF8 characters must begin and end with an apostrophe. When the string of characters must extend to another line, use a single dollar sign, '$', at the end of the line. The DMISMN statement designates the beginning of the main input program and it must be the first line of executable code in the DMIS input program.

A.12 DO

An example of the use of the DO loop statement is as follows:

```
DECL/GLOBAL,INTGR,X_dim,A,I
DECL/GLOBAL,CHAR,10,featname
M(featdef)=MACRO/'dum_label',x,y,z
```

```
F(dum_label)=FEAT/CIRCLE,INNER,CART,x,y,z,0,0,1,10
ENDMAC
A=ASSIGN/3
X_dim=ASSIGN/18
$$
$$ To define circles aligned along the X direction
$$
DO/I,1,10,A
featname=ASSIGN/CONCAT('circ_',STR(I))
DO/X_dim,10,1,-1
CALL/M(featdef),featname,X_dim,2.0,0.0
ENDDO
ENDDO
```

A.13 EQUATE

```
$$ establish part coordinate system

D(PCS)=DATSET/DAT(A),ZDIR,ZORIG,DAT(B),XDIR,YORIG,DAT(C),XORIG

$$ measure features accessible to the current part orientation
$$ measure intermediate datum references and establish an intermediate
$$ coordinate system

D(D1)=DATSET/DAT(AA),ZDIR,ZORIG,DAT(AB),XDIR,YORIG,DAT(AC),XORIG

$$ move part to new position and/or orientation
$$ remeasure intermediate datum references and establish new coordinate system
$$ datums BA,BB,BC are made from the same physical features as AA,AB,AC

D(D2)=DATSET/DAT(BA),ZDIR,ZORIG,DAT(BB),XDIR,YORIG,DAT(BC),XORIG

$$ EQUATE the two coordinate systems.

EQUATE/DA(D2),DA(D1)
RECALL/DA(PCS)

$$ measure features now accessible, these will be measured with respect to
$$ original coordinate system.
$$ measure intermediate datum references and establish an intermediate
$$ coordinate system

D(D3)=DATSET/DAT(CA),ZDIR,ZORIG,DAT(CB),XDIR,YORIG,DAT(CC),XORIG

$$ move part to new position and/or orientation
$$ remeasure intermediate datum references and establish new coordinate system
$$ datums DA,DB,DC are made from the same physical features as CA,CB,CC

D(D4)=DATSET/DAT(DA),ZDIR,ZORIG,DAT(DB),XDIR,YORIG,DAT(DC),XORIG

$$ EQUATE the two coordinate systems

EQUATE/DA(D4),DA(D3)
RECALL/DA(PCS)

$$ measure features now accessible, these will be measured with respect to
$$ original coordinate system.
```

A.14 EVAL

```
EVAL/FA(cir1),T(siz1)
DECL/REAL,cirdev
cirdev=OBTAIN/TA(siz1),3
IF/(cirdev.LT.0.245)
MD(jig_bore)=MFGDEV/'6673 JIG BORE'
TL(bore)=TOOLDF/MD(jig_bore),'22470 TOOLNO BORE TOOL'
```

```
CC(bore1)=CUTCOM/MD(jig_bore),ADJUST,TL(bore),LEFT,XYPLAN,lotol1
ENDIF
GOHOME
```

The results obtained from EVAL will reflect the current coordinate system.

A.15 FEAT/GEOM

An example of the FEAT/GEOM statement applied in the manual free form measurement of 20 points on a part, referencing a CAD file for nominal information and constructing the nearest feature nominal on the part by using FA(PtFeat).

```
DECL/INTGR,I
DID(model1)=DEVICE/STOR,'C:\CAD_DATA\MYFILE.CAD'
OPEN/DID(model1),CAD,IGES
G(part)=GEOM/DID(model1)
F(FreeForm)=FEAT/GEOM,G(part),CART
F(PtFeat)=FEAT/POINT,CART,0,0,0,0,0,1
MODE/MAN
I=ASSIGN/1
T(ProfP1)=TOL/PROFP,-0.05,0.05
(LOOP)
MEAS/POINT,F(PtFeat),1
ENDMES
$$ Constructs nearest feature nominal on the part by using FA(PtFeat)
CONST/GEOM,F(FreeForm),NEARPT,FA(PtFeat)
OUTPUT/F(FreeForm)
OUTPUT/FA(FreeForm)
OUTPUT/FA(FreeForm),TA(ProfP1)
I=ASSIGN/I+1
if/(I.LE.20)
JUMPTO/(LOOP)
ENDIF
```

A.16 FEAT/GSURF

```
PRCOMP/ON
F(lname)=FEAT/GSURF
MEAS/GSURF,F(lname),5
PTMEAS/CART,1.23,0.14,0.00,0.00,0.00,1.00
PTMEAS/CART,1.41,3.65,-0.02,0.00,0.00,1.00
PTMEAS/CART,1.87,7.15,0.01,0.00,0.00,1.00
PTMEAS/CART,.23,0.86,0.03,0.00,0.00,1.00
PTMEAS/CART,.86,2.53,0.01,0.00,0.00,1.00
ENDMES
```

A.17 GEOM

executable statements

```
DID(model2)=DEVICE/STOR,'C:\CADATA\MYFILE.CAD'
OPEN/DID(model2),CAD,STEP
G(part)=GEOM/DID(model2)
$$ Create a new named entity by offsetting the part by 2.4
G(outside)=GEOM/G(part),OFFSET,2.4
$$ Create a new named entity in part
G(hump)=GEOM/G(part),ENTITY,'aabbcc'
$$ Create a new named entity by offsetting the entity by 2.4
```

```
G(ohump)=GEOM/G(part),ENTITY,'aabbcc',OFFSET,2.4
$$ Create a new named entity by offsetting the hump by -4.8
G(tab)=GEOM/G(hump),OFFSET,-4.8
  executable statements
F(PT0)=FEAT/POINT,CART,1.23,2.61,0.00,0.00,0.00,1.00
MA(mpt0)=MATDEF/G(outside),FA(PT0)
F(PT1)=FEAT/POINT,CART,4.31,8.62,0.00,0.00,0.00,1.00
MA(mpt1)=MATDEF/G(part),FA(PT1)
F(GS3)=FEAT/GSURF
MA(mgs3)=MATDEF/G(ohump),FA(GS3)
F(PT2)=FEAT/POINT,CART,21.19,34.53,1.76,0.00,0.00,-1.00
F(EP4)=FEAT/POINT,CART,25.31,41.45,1.76,0.00,0.00,-1.00
MA(mpt1ep4)=MATDEF/G(tab),FA(PT2),FA(EP4)
```

A.18 GOTARG

The GOTARG...ENDGO block provides structure for path-directed moves; however, it is optional.

Table A.2 — GOTARG structure for path-directed moves

Valid	Invalid
GOTARG/16,10,2	GOTARG/16,10,2
GOTO/1,1,1	ENDGO
GOTO/16,10,1	
GOTO/16,10,2	GOTARG/1,1,1
ENDGO	
GOTO/1,1,1	
GOTO/16,10,1	
GOTO/16,10,2	

A.19 IF and JUMPTO

A.19.1 Example 1

```
DECL/GLOBAL,CHAR,1,AnsrYN
DECL/GLOBAL,REAL,AnsR,A,B
DECL/GLOBAL,BOOL,DMEE
DMEE=ASSIGN/.TRUE.
IF/(DMEE)
JUMPTO/(outif)
ENDIF
  executable statements
IF/((AnsrYN .EQ. 'N').AND.(AnsR .GT. 0.1))
  executable statements
ENDIF
  executable statements
(outif)
F(lname)=FEAT/POINT,CART,25.31,41.45,1.76,0.00,0.00,-1.00
  executable statements
IF/(A .LE. B)
  executable statements
```

```
ELSE
IF/(A .LT. 3.0)
  executable statements
ELSE
JUMPTO/(endblk)
ENDIF
ENDIF
(endblk)
```

A.19.2 Example 2

```
DECL/GLOBAL,BOOL,bad
  executable statements
JUMPTO/(new_probe)
(restart)
  by-passed executable statements
(new_probe)
SNSLCT/S(next_probe)
(cylndr7)
F(CYL7)=FEAT/CYLNDR,INNER,CART,-1.00,1.00,0.0,0.00,0.00,1.00,0.375
MEAS/CYLNDR,F(CYL7),8
PTMEAS/CART,-1.188,1.00,-0.250,1.00,0.00,0.00
PTMEAS/CART,-0.813,1.00,-0.250,-1.00,0.00,0.00
PTMEAS/CART,-1.00,1.188,-0.250,0.00,1.00,0.00
PTMEAS/CART,-1.00,0.813,-0.250,0.00,-1.00,0.00
PTMEAS/CART,-1.188,1.00,-5.250,1.00,0.00,0.00
PTMEAS/CART,-0.813,1.00,-5.250,-1.00,0.00,0.00
PTMEAS/CART,-1.00,1.188,-5.250,0.00,1.00,0.00
PTMEAS/CART,-1.00,0.813,-5.250,0.00,-1.00,0.00
ENDMES
  executable statements
IF/(bad)
JUMPTO/(cylndr7)
ENDIF
```

A.20 ITERAT

```
DMISMN/'PDGS Sample DMIS 2007/05/17',05.1
V(STOR_LABEL)=VFORM/NOM,ACT,DEV,AMT
DISPLY/STOR,V(STOR_LABEL)
FILNAM/'DATUMS',05.1
MODE/MAN
PRCOMP/ON
UNITS/MM,ANGDEC
WKPLAN/XYPLAN
DECL/GLOBAL,DOUBLE,dConverg
F(LOCATOR_1)=FEAT/POINT,CART,4235.0,856.0854,1145.0,0.0024,0.9999,-0.0114
F(LOCATOR_2)=FEAT/POINT,CART,4147.0,848.6901,1187.0,0.0,1.0,0.0
F(LOCATOR_3)=FEAT/POINT,CART,4292.0,858.1987,1340.0,-0.0022,0.9999,0.0115
F(DAT_A)=FEAT/PLANE,CART,4235.0,856.0854,1145.0,0.0,1.0,0.0
F(SLOT_B)=FEAT/CPARLN,INNER,ROUND,CART,4160.0,860.6941,1156.0$
,0.0,1.0,0.0,0.0,0.0,1.0,16.0,13.0
F(CIR_C)=FEAT/CIRCLE,INNER,CART,4265.0049,863.6899,130.0,0.0,-1.0,0.0,13.0
F(DAT_B)=FEAT/LINE,UNBND,CART,4265.0049,863.6899,130.0,0.0,0.0,1.0,0.0,1.0,0.0
$$
(STARTING_POINT)
```

```
$$
MEAS/POINT,F(LOCATOR_1),1
ENDMES
MEAS/POINT,F(LOCATOR_2),1
ENDMES
MEAS/POINT,F(LOCATOR_3),1
ENDMES
MEAS/CIRCLE,F(CIR_C),4
ENDMES
MEAS/CPARLN,F(SLOT_B),5
ENDMES
CONST/PLANE,F(DAT_A),OFFSET,FA(LOCATOR_1),FA(LOCATOR_2),FA(LOCATOR_3)
DATDEF/FA(DAT_A),DAT(A)
CONST/LINE,F(DAT_B),OFFSET,FA(CIR_C),FA(SLOT_B)
DATDEF/FA(DAT_B),DAT(B)
DATDEF/FA(CIR_C),DAT(C)
D(orien_1)=DATSET/DAT(A),YDIR,YORIG
D(orien_2)=ROTATE/YAXIS,DAT(B),ZDIR
D(REF_SYS_1)=TRANS/XORIG,DAT(C),ZORIG,DAT(C)
SAVE/DA(REF_SYS_1)
MODE/AUTO,MAN
dConverg=ITERAT/(STARTING_POINT),(FAILED),0.05,ABSL,5,YAXIS,FA(LOCATOR_1),$
FA(LOCATOR_2),FA(LOCATOR_3),XAXIS,FA(CIR_C),FA(SLOT_B),ZAXIS,FA(CIR_C)
  executable statements
JUMPTO/(EOP)
$$
(FAILED)
TEXT/OPER,'Convergence failed!'
(EOP)
ENDFIL
```

A.21 KEYCHAR

Example of the use of KEYCHAR, TOL/COMPOS, TOL/DISTWRT, TOL/DISTB, and FEAT/PARPLN per Figure A.3 — Key Characteristics. Note the Key Characteristics are labeled within the pentagons.

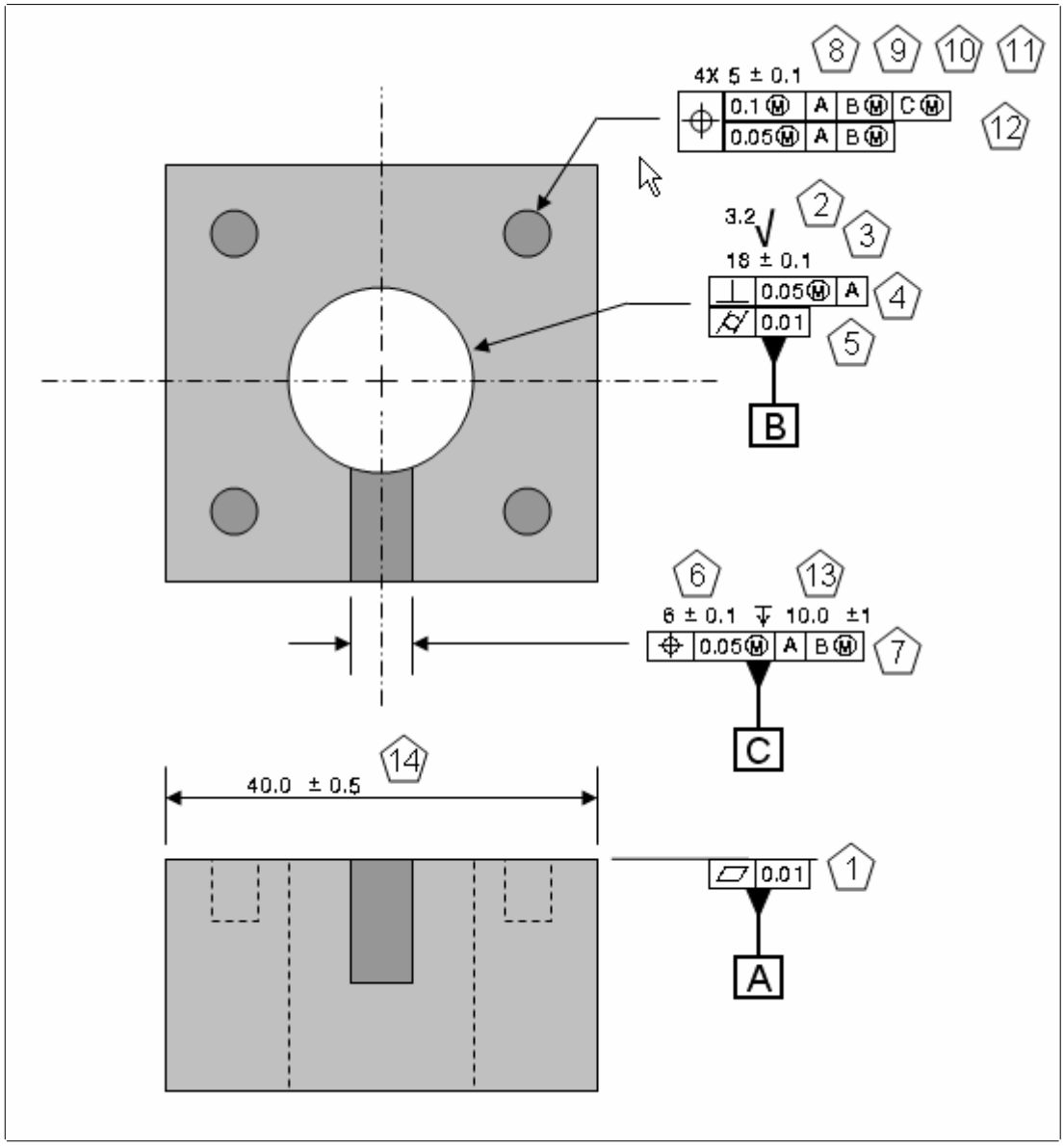


Figure A.3 — Key Characteristics

```
$$ Define, Measure, and Datum Top Plane
$$ an example of key characteristic definition and output
$$ after measurement
F(TOP)=FEAT/PLANE,INNER,CART,  0.00,  0.00,20.00,0.00,0.00,1.00
MEAS/PLANE,F(TOP),8
. . .
ENDMES
DATDEF/FA(Top),DAT(A)
$$ Tolerance, KeyChar, and Output Top Plane
T(Flat0.01) = TOL/FLAT, 0.01
KC(KC1) = KEYCHAR/F(Top), T(Flat0.01), MINOR
OUTPUT/KC(KC1)
$$ resulting output (4 lines):
$$  OUTOUT/KC(KC1)
$$ KC(KC1) = KEYCHAR/F(Top), T(Flat0.01), MINOR
$$  F(TOP)=FEAT/PLANE,INNER,CART,  0.00,  0.00,20.00,0.00,0.00,1.00
$$ T(Flat0.01) = TOL/FLAT, 0.01
```

```
OUTPUT/KCA(KC1)
$$ resulting output (4 lines):
$$ OUTOUT/KCA(KC1)
$$ KCA(KC1) = KEYCHAR/FA(Top), TA(Flat0.01), MINOR
$$ FA(TOP)=FEAT/PLANE,INNER,CART,  0.01,  0.02,20.01,0.03,0.00,0.99
$$ TA(Flat0.01) = TOL/FLAT, 0.003, INTOL

$$ Define PCS for DRF A
D(DRF_A) = DATSET/DAT(A),ZDIR,ZORIG

$$ Define 18mm Hole
$$ an example of key characteristic definition and nominal output
$$ before measurement followed by actual output after measurement
F(18Hole)=FEAT/CYLNDR,INNER,CART,  0.00,  0.00,0.00,0.00,0.00,1.00,18.00,20.00
$$ Tolerance, KeyChar, and Output 18mm Hole
T(Size+/-0.1) = TOL/DIAM,-0.1, 0.1
KC(KC3) = KEYCHAR/F(18Hole), T(Size+/-0.1), MINOR
T(Perp0.05wA) = TOL/PERP,0.05,RFS,DAT(A)
KC(KC4) = KEYCHAR/F(18Hole), T(Perp0. 05wA), MINOR
T(Cyl0.01) = TOL/CYLCTY,0.01
KC(KC5) = KEYCHAR/F(18Hole), T(Cyl0.01), MINOR
OUTPUT/KC(KC3)
OUTPUT/KC(KC4)
OUTPUT/KC(KC5)

$$ Now Measure, Datum and Output 18 mm Hole
MEAS/CYLNDR,F(18Hole),8
. . .
ENDMES
DATDEF/FA(18Hole),DAT(B)
OUTPUT/KCA(KC3)
OUTPUT/KCA(KC4)
OUTPUT/KCA(KC5)

$$ Define PCS for DRF AB
D(DRF_AB) = DATSET/DAT(A),ZDIR,ZORIG, DAT(B),XORIG, YORIG

$$ examples of key characteristic actual-only output

$$ Define, Measure, and Datum 6mmSlot
F(6Slot)=FEAT/PARPLN,INNER,MIDPL,CART,  0.00,7.00,15.00,  1.00,0.00,0.00,  6.00
MEAS/PARPLN,F(6Slot),8
. . .
ENDMES
DATDEF/FA(6Slot),DAT(C)
T(Width+/-0.1) = TOL/WIDTH,-0.1, 0.1
KC(KC6) = KEYCHAR /F(6Slot), T(Width+/-0.1), MINOR
OUTPUT/KCA(KC6)

T(Pos0.05wABm) = TOL/POS, 3D, 0.05,MMC, DAT(A), DAT(B),MMC
KC(KC7) = KEYCHAR/F(6Slot), T(Pos0.05wABm), MINOR
OUTPUT/KCA(KC7)

$$ Define and Measure Slot Bottom
F(SlotBot)=FEAT/PLANE,INNER,CART,  0.00,  0.00,10.00,0.00,0.00,1.00
MEAS/PLANE,F(SlotBot),4
. . .
ENDMES
T(Dist10+/-1) = TOL/DISTWRT,NOMINL,10.00,-1.0,1.0,DAT(A)
KC(KC13) = KEYCHAR/F(SlotBot), T(Dist10+/-1), MINOR
OUTPUT/KCA(KC13)
```

```
$$ Define PCS for DRF ABC
D(DRF_ABC) = DATSET/DAT(A),ZDIR,ZORIG, DAT(B),XORIG, YORIG, DAT(C),XDIR

$$ Define and Measure Bolt Holes
F(SW4Hole)=FEAT/CYLNDR,INNER,CART,-13.00,-13.00,14.00,0.00,0.00,1.00,4.00,6.00
MEAS/CYLNDR,F(SW4Hole),8
. . .
ENDMES
KC(KC8) = KEYCHAR/F(SW4Hole), T(Size+/-0.1), MINOR
OUTPUT/KCA(KC8)

F(SE4Hole)=FEAT/CYLNDR,INNER,CART,13.00,-13.00,14.00,0.00,0.00,1.00,4.00,6.00
MEAS/CYLNDR,F(SE4Hole),8
. . .
ENDMES
KC(KC9) = KEYCHAR/F(SE4Hole), T(Size+/-0.1), MINOR
OUTPUT/KCA(KC9)

F(NE4Hole)=FEAT/CYLNDR,INNER,CART,13.00,13.00,14.00,0.00,0.00,1.00,4.00,6.00
MEAS/CYLNDR,F(NE4Hole),8
. . .
ENDMES
KC(KC10) = KEYCHAR/F(NE4Hole), T(Size+/-0.1), MINOR
OUTPUT/KCA(KC10)

F(NW4Hole)=FEAT/CYLNDR,INNER,CART,-13.00,13.00,14.00,0.00,0.00,1.00,4.00,6.00
MEAS/CYLNDR,F(NW4Hole),8
. . .
ENDMES
KC(KC11) = KEYCHAR/F(NW4Hole), T(Size+/-0.1), MINOR
OUTPUT/KCA(KC11)

$$ Define Hole Pattern
F(HolePattern)=FEAT/PATERN, F(SW4Hole), F(SE4Hole), F(NE4Hole), F(NW4Hole)

$$ Construct Hole Pattern
CONST/ PATERN, F(HolePattern), BUILD
T(ComPos) = TOL/COMPOS, PATERN, 0.1,MMC, DAT(A),DAT(B),DAT(C), FEATUR,0.05,MMC,$
DAT(A),DAT(B)
KC(KC12) = KEYCHAR/F(HolePattern), T(ComPos), MAJOR
OUTPUT/KCA(KC12)

$$ Define, Measure, and Left & Right Planes
F(Left)=FEAT/PLANE,INNER,CART, -20.00, 0.00, 0.00,-1.00,0.00,0.00
MEAS/PLANE,F(Left),5
. . .
ENDMES
F(Right)=FEAT/PLANE,INNER,CART, 20.00, 0.00, 0.00, 1.00,0.00,0.00
MEAS/PLANE,F(Right),5
. . .
ENDMES
$$ Tolerance, KeyChar, and Output Left & Right Plane
T(Dist40.0+/-0.5) = TOL/DISTB, NOMINL, 40.0. -0.5, 0.5
KC(KC14) = KEYCHAR/F(Left), F(Right), T(Dist40.0+/-0.5), MINOR
OUTPUT/KCA(KC14)
```

A.22 Macro definition

A.22.1 An example of a macro definition without an argument list:

```
M(TEST)=MACRO
GOTO/10,10,10
GOTO/10,15,10
GOTO/15,15,10
GOTO/15,10,10
ENDMAC
CALL/M(TEST)
```

A.22.2 An example of a macro definition and a CAll statement follows:

```
FROM/0,0,2
MODE/AUTO,PROG,MAN
M(bolthole)=MACRO/x1,y1,x2,y2,'label_1','label_2'
GOTO/x1,y1,2
MEAS/CIRCLE,F(lname_1),4
ENDMES
GOTO/x1,y1,2
GOTO/x2,y2,2
MEAS/CIRCLE,F(lname_2),4
ENDMES
GOTO/x2,y2,2
GOHOME
ENDMAC
CALL/M(bolthole),1.0,1.5,-1.0,-1.5,(righthole),(lefthole)
```

A.23 OBTAIN

In the example below, the shutter speed from the LASER sensor definition is obtained and named shutsp. Note that the shutter speed is the eleventh parameter following the '/' delimiter.

```
S(Lazr1)=SNSDEF/LASER,FIXED,CART,2.031,1.973,.982,0,1,0,128,82
DECL/REAL,shutsp
DECL/CHAR,6,SENSOR_TYP
shutsp=OBTAIN/S(Lazr1),11
SENSOR_TYP=OBTAIN/S(Lazr1),1
```

Results in: SENSOR_TYP=LASER

A.24 OPEN

An example of the use of the OPEN statement to open devices are as follows:

```
OPEN/DID(sensor_data),SNS
OPEN/DID(DRFsdata),PCS
OPEN/DID(FloorPrinter),FDATA,DMIS,OUTPUT
OPEN/DID(system_term),DIRECT,OUTPUT
OPEN/DID(SummaryData),DIRECT,OUTPUT,OVERWR
OPEN/DID(Header_Info),DIRECT,INPUT
```

A.25 PROMPT, examples of numeric and alphanumeric input

A.25.1 Example 1 numeric input

```
DECL/DOUBLE,dHoleDia
$$
$$ The following input will …
$$ accept any numeric value
dHoleDia=PROMPT/'Hole diameter ? '
$$
$$ The following input will …
$$ limit input to less than 100
dHoleDia=PROMPT/'Hole diameter ? ',100
$$
$$ The following input will …
$$ limit to range 5-100
dHoleDia=PROMPT/'Hole diameter ? ',100,5
```

A.25.2 Example 2 alphanumeric input

```
DECL/CHAR,40,cName
$$
$$ The following input will …
$$ accept input up to 40 alphanumeric chars
cName=PROMPT/'Operator name ? '
$$
$$ The following input will …
$$ accept input up to 20 alphanumeric chars
cName=PROMPT/'Operator name ? ',20
```

A.25.3 Example 3 multiple input and playing a .wav file

```
$$ refer to (Figure A.4 — The PROMPT dialog box) below.
$$
DECL/INTGR,ok,model
DECL/CHAR,32,shift
ok=PROMPT/TITLE,'Enter inspection run information',    $
  TEXT,'Provide your operator ID, in the box below',   $
  EDIT,OP(oper_id),32,                                 $
  TEXT,'Pick the model of part to inspect',            $
   GROUP,'Model',model,'Short Block','Long Block','V6', $
         'Diesel',                                      $
  TEXT,'Select your shift',                            $
  LIST,shift,'First','Second','Third',                 $
  SOUND,'wakeup.wav',                                  $
  BUTTON,'OK',1,                                       $
  BUTTON,'Cancel',2
IF/(ok .EQ.  1)
  SELECT/model
   CASE/3
     DECL/CHAR,32,msg
      msg=ASSIGN/CONCAT('V6 Block during ',shift,' Shift')
     TEXT/OUTFIL,msg
     R(run)=REPORT/OP(oper_id)
$$Inspect the part here
   ENDCAS
  ENDSEL
ELSE
  TEXT/OUTFIL,'Run canceled by operator'
ENDIF
```

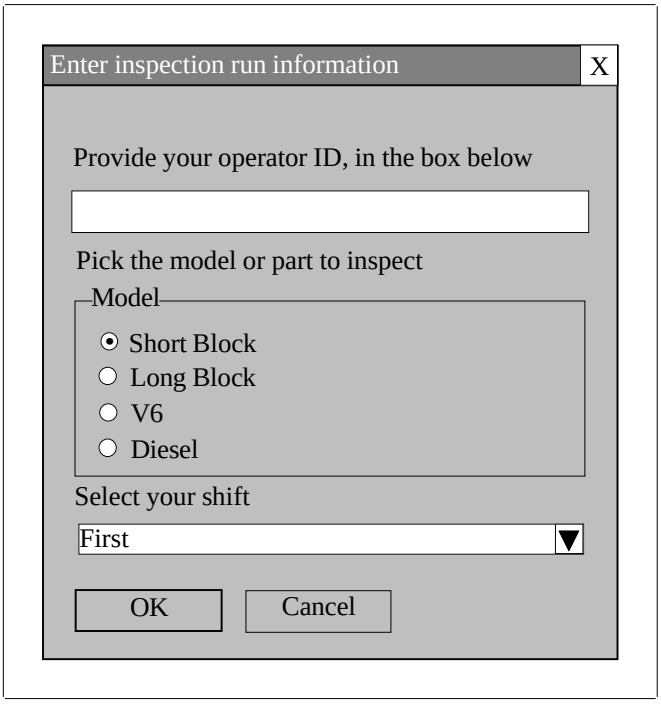


Figure A.4 — The PROMPT dialog box

A.26 QISDEF

```
Q(OperStep)=QISDEF/'OPSTEP','Operation 10'
R(rpt) = REPORT/Q(OperStep)
OUTPUT/R(rpt)
```

Produces the following in the DMIS output file:

```
Q(OperStep)=QISDEF/'OPSTEP','Operation 10'
```

A.27 READ examples of delimited and formatted

A.27.1 An example of the READ statement for a delimited file

```
DECL/DOUBLE,X[10],Y[10],Z[10],DIA,TOL_ZONE,P_TOL,M_TOL
DECL/INTGR,I
I=ASSIGN/2
READ/DID(DATA),X[i],Y[i],Z[i],Dia,Tol_Zone,P_tol,M_tol
```

A.27.2 An example of the READ statement for a formatted file

```
READ/DID(DATA),X[i]:8:4,Y[i]:8:4,Z[i]:8:4,Dia:8:4,Tol_Zone:6:4
```

A.28 ROTDEF rotary table definition

Refer to (Figure A.5 — Rotary table application sequence), below, for a graphical description of the rotary table and part being measured in the example program.

```
$$
$$ Declare variables
$$
```

```
DECL/CHAR,8,cir_lbl
DECL/INTGR,I,J
$$ Define rotary table relative to the DME home position.
$$
RT(ROTARY1)=ROTDEF/40.0,15.0,-10.0,0.0,0.0,1.0
$$
$$ Set some rotary table parameters.
$$
FEDRAT/ROTVEL,LOW
ACLRAT/ROTACL,LOW
ROTSET/RT(RTAB1),0
$$
$$ Select probe for calibration of rotary table.
$$
SNSLCT/S(PROBE1)
$$
$$ Translate from machine 0.0,0.0,0.0 to the center of the bung
$$
D(mytable)=TRANS/XORIG,40.0,YORIG,15.0,ZORIG,-10.0
$$
$$ Measure rotary table plane and level to it
$$
F(TABLE)=FEAT/PLANE,0,0,0,0,0,1
MEAS/PLANE,F(TABLE),4
PTMEAS/POL,200,0,0,0,0,1
PTMEAS/POL,200,90,0,0,0,1
PTMEAS/POL,200,180,0,0,0,1
PTMEAS/POL,200,270,0,0,0,1
ENDMES
DATDEF/FA(TABLE),DAT(A)
D(TABLE)=DATSET/DAT(A),ZDIR,ZORIG
$$
$$ Perform calibration of rotary table, with probe compensation off.
$$
PRCOMP/OFF
F(BUNG)=FEAT/CIRCLE,INNER,POL,0.0,0.0,0.0,0,0,1,1.000
MEAS/CIRCLE,F(BUNG),4
PTMEAS/POL,0.5,0.0,-0.25
PTMEAS/POL,0.5,90.0,-0.25
PTMEAS/POL,0.5,180.0,-0.25
PTMEAS/POL,0.5,270.0,-0.25
ENDMES
CALIB/RTAB,RT(ROTARY1),FA(TABLE),FA(BUNG)
ENDMES
$$
$$ Mount the part, a disk, on the rotary table and establish the
$$ orientation and alignment for the part coordinate system.
$$ Select hooked probe for inspection.
$$
SNSLCT/S(HOOKPROBE)
$$
$$ Position hooked probe for measurement taking.
$$
GOTO/POL,8.25,0.0,5.5
GOTO/POL,8.25,0.0,3.5
$$
$$ Now perform measurement on the 15.000 inch diameter at 3.5 inches
$$ up from the rotary table.
$$
F(cir_15)=FEAT/CIRCLE,OUTER,POL,0.0,0.0,3.5,0,0,1,4.5
I=ASSIGN/1
```

```
J=ASSIGN/0
(cir_loop)
cir_lbl=ASSIGN/CONCAT('cir_pt',STR(I))
F(@cir_lbl)=FEAT/POINT,POL,7.5,J,3.5,1,0,0
MEAS/POINT,F(@cir_lbl),1
ENDMES
GOTO/POL,8.25,0.0,3.5
ROTAB/RT(ROTARY1),INCR,CW,ROTTOT,20.0
I=ASSIGN/(I+1)
J=ASSIGN/(J+20)
IF/(J.LE.351)
JUMPTO/(cir_loop)
ENDIF
CONST/CIRCLE,F(cir_15),BF,FA(cir_pt1),FA(cir_pt2),FA(cir_pt3),FA(cir_pt4)$
,FA(cir_pt5),FA(cir_pt6),FA(cir_pt7),FA(cir_pt8),FA(cir_pt9),FA(cir_pt10)$
,FA(cir_pt11),FA(cir_pt12),FA(cir_pt13),FA(cir_pt14),FA(cir_pt15) $
,FA(cir_pt16),FA(cir_pt17),FA(cir_pt18)
  executable statements
```

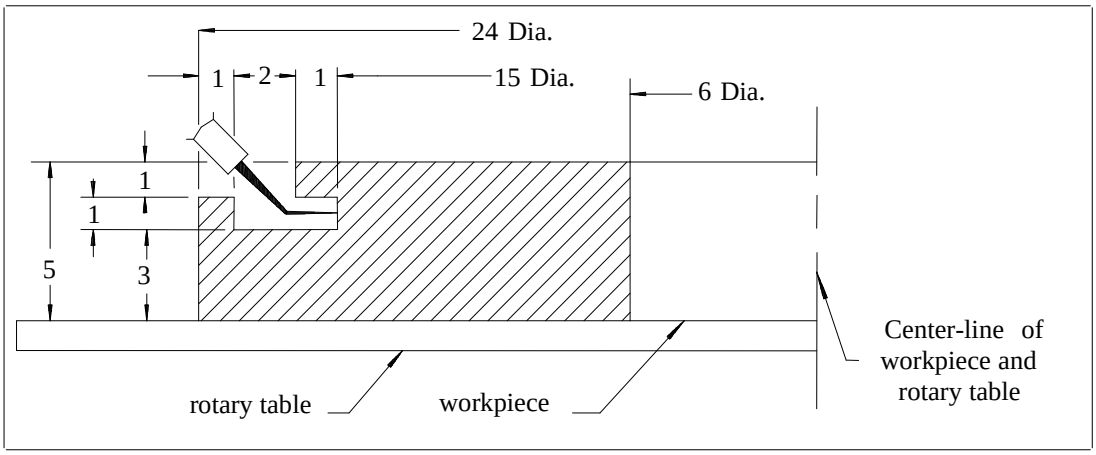


Figure A.5 — Rotary table application sequence

A.29 SELECT...CASE...ENDCAS...DFTCAS...ENDCAS...ENDSEL block example

```
DECL/GLOBAL,CHAR,12,gname
SELECT/gname
$$ If gname variable is equal to tyler or kelsey, then
$$ the following CASE block is executed.
CASE/'tyler'
CASE/'kelsey'
. . .
ENDCAS
$$ If gname variable is equal to DSC then the
$$ following CASE block is executed.
CASE/'DSC'
. . .
ENDCAS
$$ If gname variable is not equal to tyler, kelsey, or
$$ DSC then and only then the following default block is
$$ executed.
DFTCAS
. . .
ENDCAS
```

```
ENDSEL
```

A.30 Sensor, wrist, component group, and build examples

A.30.1 Example wrist, component, and sensor definition

A graphical example of a wrist definition statement is shown in (Figure A.6 — Example wrist definition values).

```
DMISMN/'Sample syntax for sensor definition',05.1
$$
$$ Defining this object:        PH10M   30mm from mount to rotation centre
$$                              45mm from rotation centre to joint surface
$$                              PAA1    32mm Autojoint
$$                              PEL1    50mm extension
$$                              TP200   43mm Touch probe
$$                              SE1     10mm Stylus extension
$$                              PS1R    20mm x 2mm Stylus
$$
$$
$$ Define the sensor mount orientation for a bridge machine
SNSMNT/XVEC,1,0,0,ZVEC,0,0,1,MNTLEN,0,0,0

$$ Define the wrist
SW(PH10)=WRIST/ROTCEN,0,0,-5,0,0,1,0,1,0,ANGLE,'RotAngle',-180,180,7.5,$
ROTCEN,0,0,-25,1,0,0,0,0,-1,ANGLE,'TiltAngle',0,105,7.5,$
MNTLEN,0,0,-45,'Renishaw','PH10M',1

$$ Define the components
$$      PAA1    32mm Autojoint
SX(PAA1)=EXTENS/0,0,-32
$$ or SX(PAA1)=EXTENS/VEC,0,0,-1,32
$$      PEL1    50mm extension
SX(PEL1)=EXTENS/0,0,-50
$$      TP200   43mm Touch probe
SX(TP200)=EXTENS/0,0,-43
$$      SE1     10mm Stylus extension
SX(SE1)=EXTENS/0,0,-10
$$      PS1R    20mm x 2mm Stylus
SS(PS1R)=SENSOR/PROBE,0,0,-20,0,0,-1,2
SS(Tool)=SENSOR/PROBE,0,0,-155,0,0,-1,2

$$ Assemble the components into a useable object
SGS(Tool1)=SNSGRP/BUILD,SX(PAA1),SX(PEL1),SX(TP200),SX(SE1),SS(PS1R)

$$ If using a tool changer, define where this object is located in
$$ the toolholder
THLDEF/SGS(Tool1),1

$$ Now turn these objects (Wrist PH10 and Tool1) into a sensor
S(Probe1)=SNSDEF/BUILD,SW(PH10),SGS(Tool1)

$$ Select the Sensor
SNSLCT/S(Probe1)

$$ Define the calibration reference feature in machine co-ordinates
F(Cal)=FEAT/SPHERE,OUTER,CART,250,250,-450,20,0,0,1

$$ Now CALIB the sensor
MODE/AUTO,MAN
CALIB/SENS,S(Probe1),F(Cal),9
```

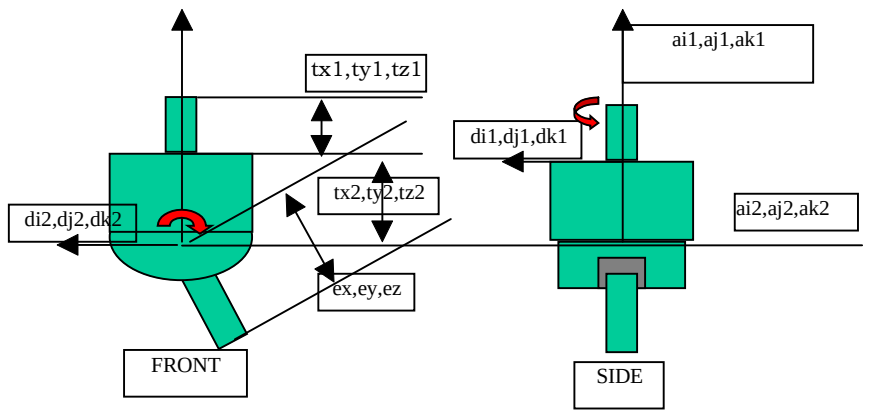
```
ENDMES
OUTPUT/SA(Probe1)
SAVE/SA(Probe1)

$$ Select the probe
SNSLCT/SA(Probe1)

$$ Define and measure a circle
F(CR1)=FEAT/CIRCLE,INNER,CART,100,200,-600,0,0,1,10
MEAS/CIRCLE,F(CR1),4
ENDMES

$$ Define and measure a circle
F(CR2)=FEAT/CIRCLE,INNER,CART,100,200,-600,0,-1,0,10

$$ Select the probe and orientate to feature to be measured A:90,B:90
SNSLCT/SA(Probe1),SW(PH10),'RotAngle',F(CR2),'TiltAngle',F(CR2)
MEAS/CIRCLE,F(CR2),4
ENDMES
ENDFIL
```




```
SW(P_2_1) = WRIST/ROTCEN,tx1,ty1,tz1, ai1,aj1,ak1, di1,dj1,dk1,$
ANGLE, 'RotAngle',-180,180,7.5, ROTCEN,tx2,ty2,tz2, ai2,aj2,ak2, di2,dj2,dk2, $
ANGLE,'TiltAngle',0,105,7.5, MNTLEN, ex,ey,ez ,'mfg_name','model_name',index
```

Figure A.6 — Example wrist definition values

A.30.2 Example component definitions

```
DECL/GLOBAL,DOUBLE,Dy,Dz
SNSMNT/XVEC,1,0,0,ZVEC,0,0,1,MNTLEN,0,0,0
$$ EXTENSION DEFINITIONS
SX(PX1)=EXTENS/VEC,0,0,-1,30
SX(PX3)=EXTENS/0,(Dy+30),Dz
SX(PX5)=EXTENS/0,-(Dy+30),Dz
$$ SENSOR DEFINITIONS
SS(PP1)=SENSOR/PROBE,0,0,-20,0,0,-1,3,SPHERE
SS(PP3)=SENSOR/PROBE,0,20,0,0,1,0,3,SPHERE
SS(PP5)=SENSOR/PROBE,0,-20,0,0,-1,0,3,SPHERE
$$ PROBE CONFIGURATION DEFINITIONS
S(P1)=SNSDEF/BUILD,SX(PX1),SS(PP1)
S(P3)=SNSDEF/BUILD,SX(PX3),SS(PP3)
```

```
S(P5)=SNSDEF/BUILD,SX(PX5),SS(PP5)
```

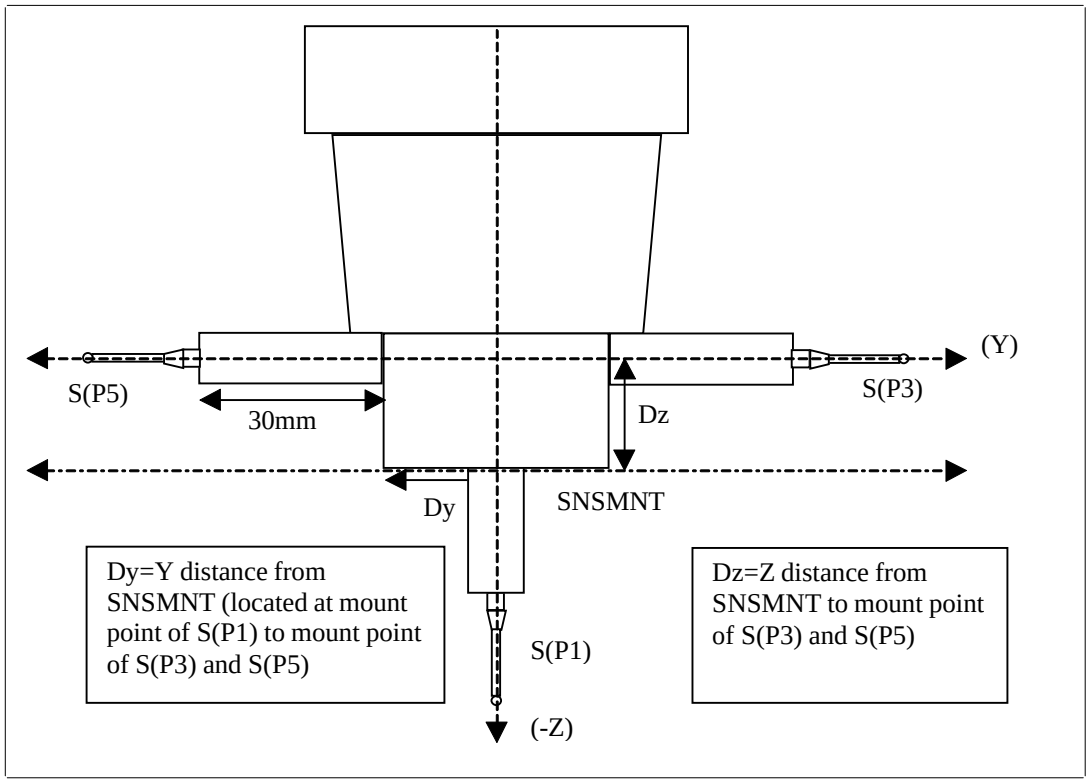


Figure A.7 — Multi-probes from multiple ports on the head

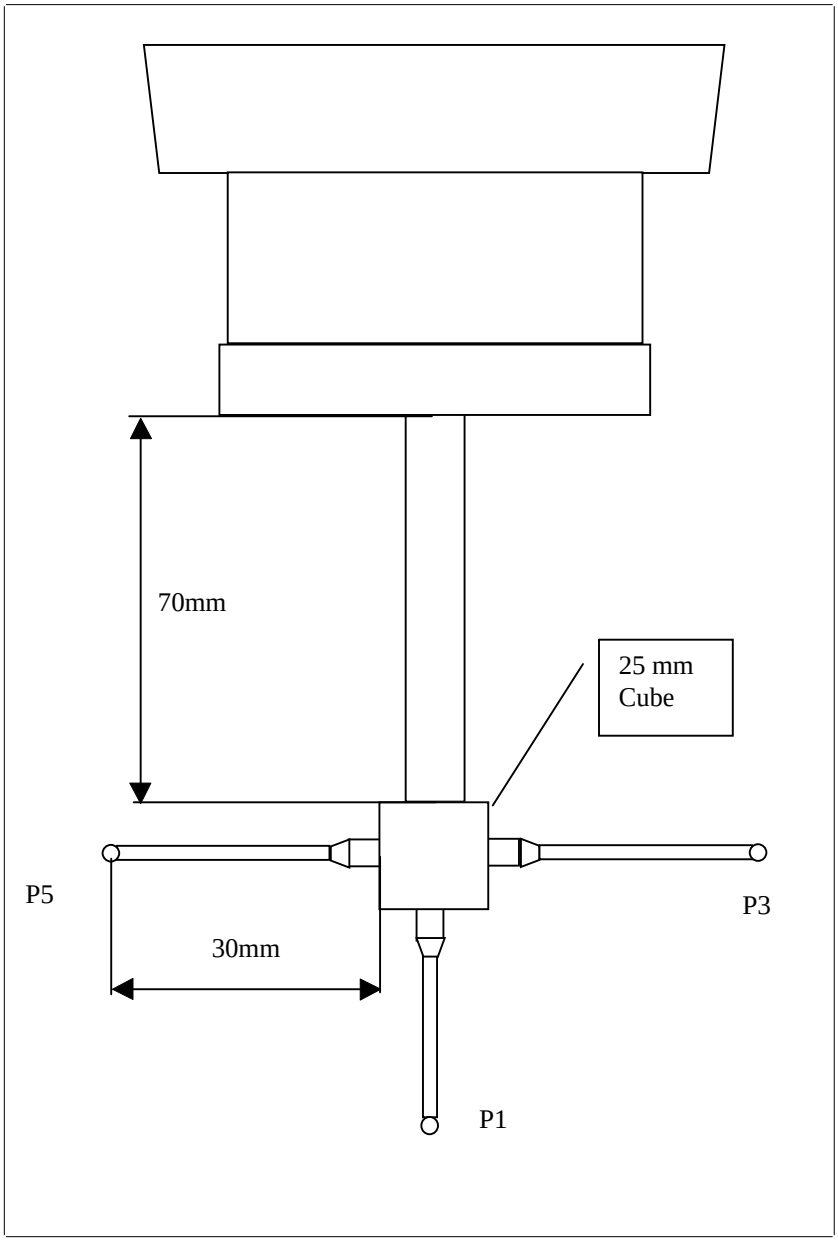


Figure A.8 — Multi-probes from multiple ports on a cube

A.30.3 Example component, group, and probe definitions

```
SNSMNT/XVEC,1,0,0,ZVEC,0,0,1,MNTLEN,0,0,0
$$ EXTENSION DEFINITIONS
SX(PX)=EXTENS/VEC,0,0,-1,70
SX(PX1)=EXTENS/0,0,-25
SX(PX3)=EXTENS/0,12.5,-12.5
SX(PX5)=EXTENS/0,-12.5,-12.5
$$ SENSOR DEFINITIONS
SS(PP1)=SENSOR/PROBE,0,0,-30,0,0,-1,3,SPHERE
SS(PP3)=SENSOR/PROBE,0,30,0,0,1,0,3,SPHERE
SS(PP5)=SENSOR/PROBE,0,-30,0,0,-1,0  ,3,SPHERE
$$ COMPONENT GROUPING
SG(PG1)=CMPNTGRP/BUILD,SX(PX),SX(PX1)
```

```
SG(PG3)=CMPNTGRP/BUILD,SX(PX),SX(PX3)
SG(PG5)=CMPNTGRP/BUILD,SX(PX),SX(PX5)
$$ PROBE CONFIGURATION DEFINITIONS
S(P1)=SNSDEF/BUILD,SG(PG1),SS(PP1)
S(P3)=SNSDEF/BUILD,SG(PG3),SS(PP3)
S(P5)=SNSDEF/BUILD,SG(PG5),SS(PP5)
```

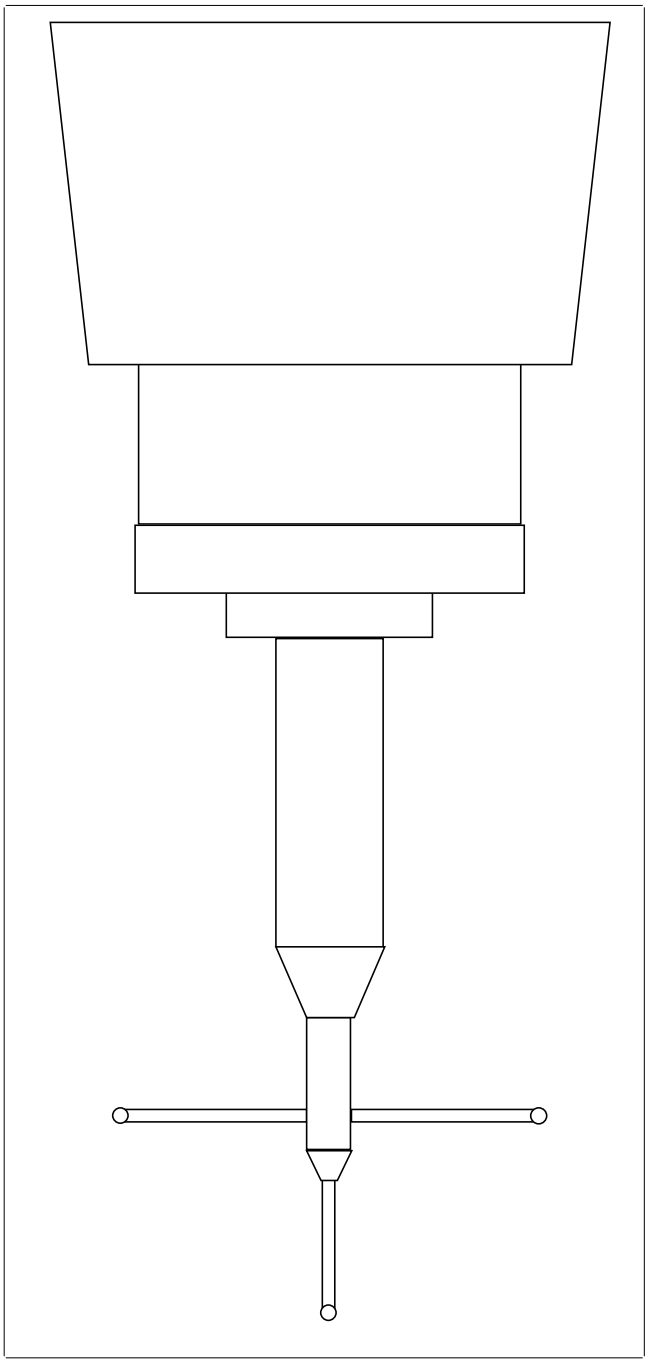


Figure A.9 — Multi-probes from multiple ports on a cube

A.30.4 Example component definition and calibration statements

```
SNSMNT/XVEC,1,0,0,ZVEC,0,0,1,MNTLEN,0,0,0
$$ EXTENSION DEFINITIONS
SX(PX1)=EXTENS/VEC,0,0,-1,75
```

```
$$ SENSOR DEFINITIONS
SS(STAR)=SENSOR/MLTPRB,3,1,0,0,-50,0,0,-1,3,$
3,0,25,-25,0,1,0,3,$
5,0,-25,-25,0,-1,0,3
$$  PROBE CONFIGURATION DEFINITIONS
S(P1)=SNSDEF/BUILD,SX(PX1),SS(STAR)
$$  CALIBRATION COMMANDS
CALIB/SENS,S(P1),FA(CALSPH),6
$$  SENSOR SELECT
SNSLCT/SA(P1),1
SNSLCT/SA(P1),3
SNSLCT/SA(P1),5
```

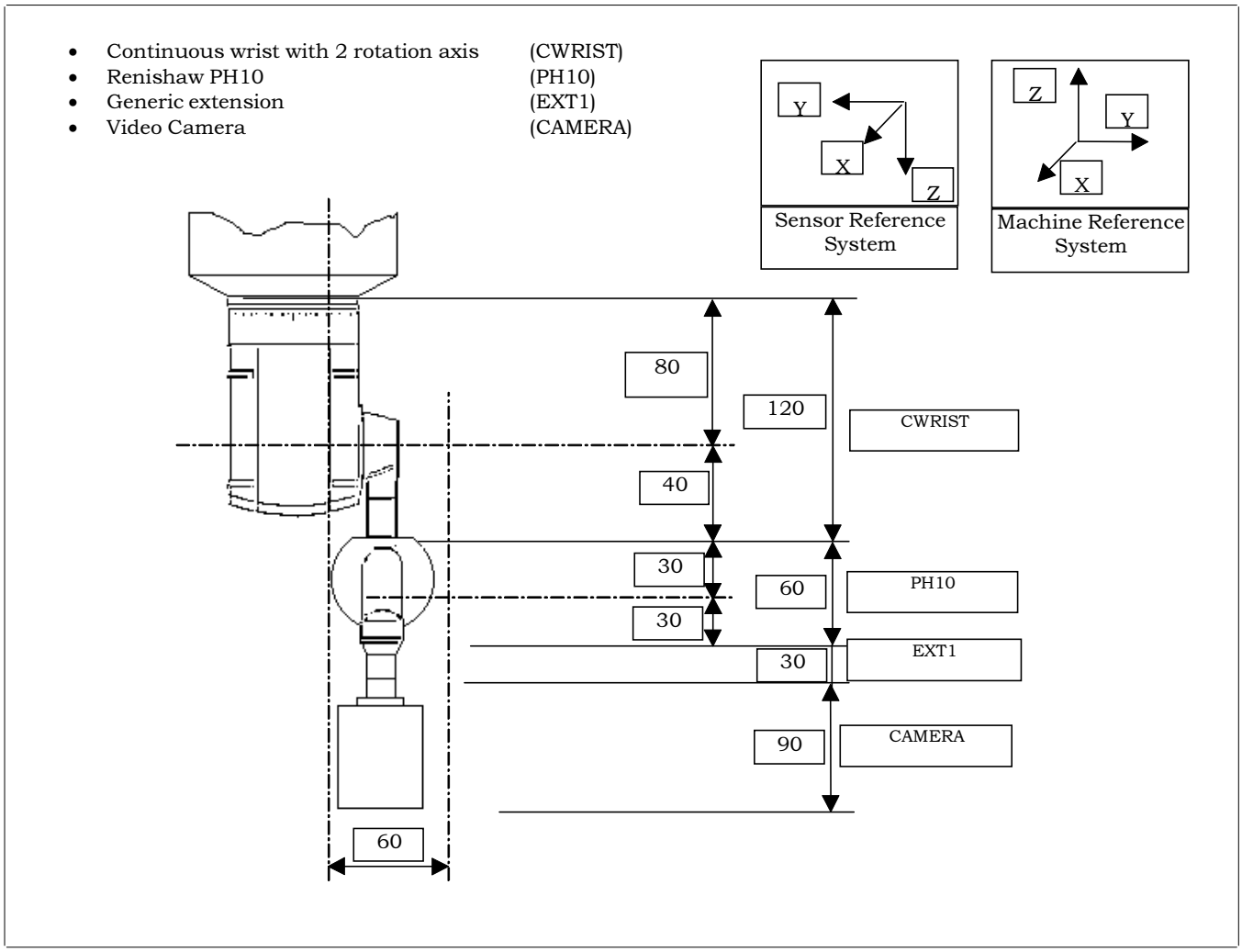


Figure A.10 — Multi-probe from an extension & from a single port

This example shows how to use DMIS statements for sensor definition in the above configuration:

A.30.5 Example complex wrist and component definitions

```
DMISMN/'Sample syntax for new sensor definition',05.1
$$ Define the sensor mount orientation
SNSMNT/XVEC,1,0,0,ZVEC,0,0,-1,MNTLEN,0,0,0
$$ Define the wrist CWRIST
SW(CWRIST)=WRIST/ROTCEN,0,0,0,0,0,1,1,0,0,ANGLE,'RotAngle',-180,180,CONTIN,$
  ROTCEN,0,-60,80,0,-1,0,0,0,1,ANGLE,'TiltAngle',-60,60,CONTIN,MNTLEN,0,0,40
```

```
$$ Define the wrist PH10
SW(PH10)=WRIST/ROTCEN,0,0,30,0,0,-1,1,0,0,ANGLE,'BAngle',-180,180,7.5,$
ROTCEN,0,0,25,0,1,0,0,0,1,ANGLE,'AAngle', $
0,105,7.5,MNTLEN,0,0,30
$$ Define the extension EXT1
SX(EXT1)=EXTENS/0,0,30
$$ Define the video sensor CAMERA
SS(CAMERA)=SENSOR/VIDEO,0,0,90,0,0,1,1,0,0,0,0,0,'SONY','CX1',1
$$ Assemble the components into a useable sensor
S(MYPROBE)=SNSDEF/BUILD,SW(CWRIST),SW(PH10),SX(EXT1),SS(CAMERA)
$$ Define the calibration reference feature in machine co-ordinates
F(GAUGE)=FEAT/OBJECT,200,200,-500,0,0,1
$$ Select the probe
SNSLCT/S(MYPROBE)
$$ Calibrate the sensor
MODE/AUTO,PROG,MAN
CALIB/SENS,S(MYPROBE),F(GAUGE),'SONY_CX1_CALIB_PROC'
ENDMES
OUTPUT/SA(MYPROBE)
SAVE/SA(MYPROBE)
$$ Select the probe
SNSLCT/SA(MYPROBE)
$$ Define and measure a circle
F(CR1)=FEAT/CIRCLE,INNER,CART,100,200,-600,0,0,1,10
MEAS/CIRCLE,F(CR1),6
ENDMES
$$ Select the probe and orientate to feature to be measured A:90,B:90
SNSLCT/SA(MYPROBE),SW(CWRIST),'RotAngle',F(CR2),'TiltAngle',F(CR2)
$$ Define and measure a circle
F(CR2)=FEAT/CIRCLE,INNER,CART,100,200,-600,0,-1,0,10
MEAS/CIRCLE,F(CR2),6
ENDMES
ENDFIL
```

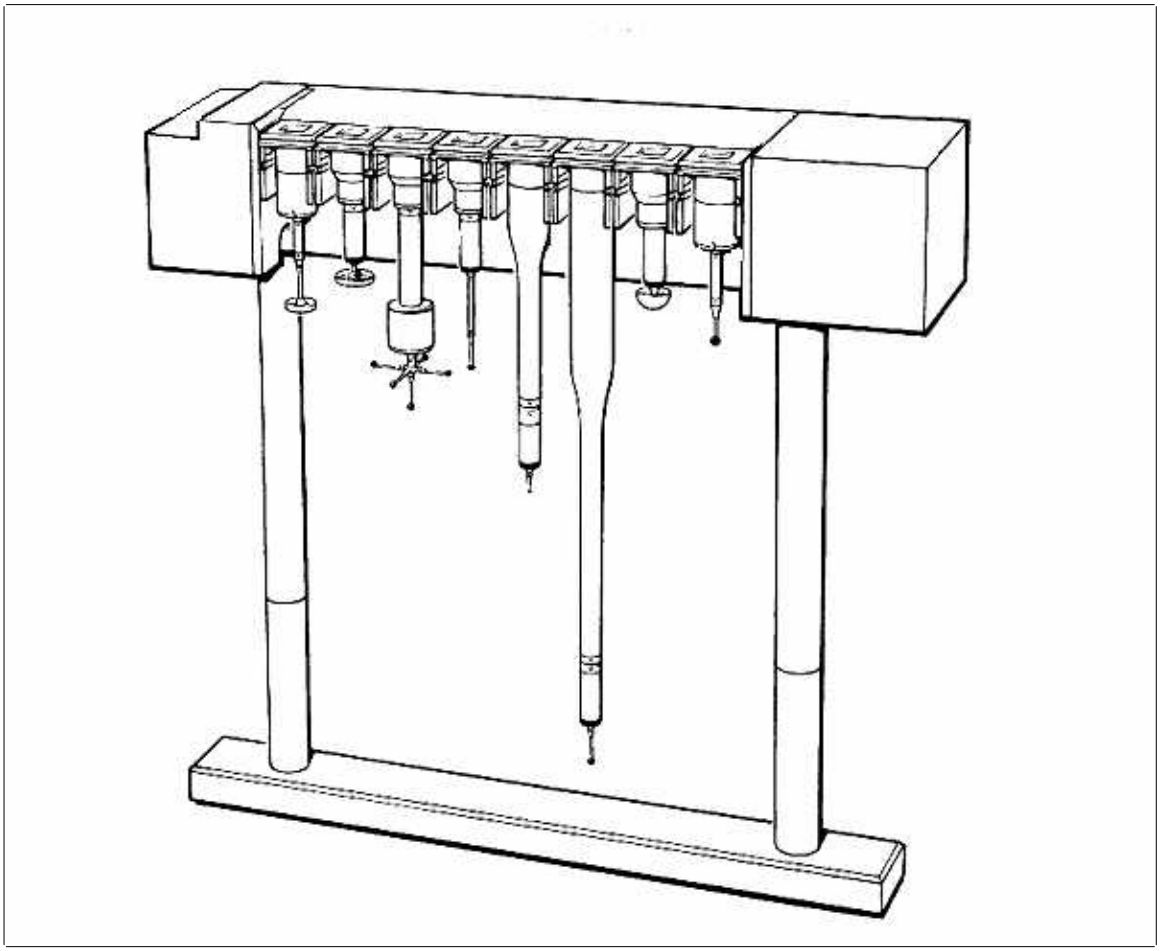


Figure A.11 — Sensor components in an automated changer rack

Description of the tools in an automated changer rack (Figure A.11 — Sensor components in an automated changer rack).

Tool1:

Automated change adapter with integrated trigger probe
35mm styli extension
Disk styli (31mm len x 12.7mm dia x 2mm thk)

Tool2:

Automated change adapter
Trigger probe
Disk styli (4mm len x 35mm dia x 5mm thk)

Tool3:

Automated change adapter
50 mm extension
Trigger probe
5 way star stylus center (13mm len x 10mm dia)
Star styli (5 x 20mm len x 2mm rubies)

Tool4:

Automated change adapter
Trigger probe

50mm styli extension
Ruby styli (40mm len x 4mm dia)

Tool5:

Automated change adapter with 140mm extension
Trigger probe
Ruby styli (17mm len x 4mm dia)

Tool6:

Automated change adapter with 300mm extension
Trigger probe
Ruby styli (27mm len x 4mm dia)

Tool7:

Automated change adapter
Trigger probe
Hollow ball styli (17mm len x 30mm dia)

Tool8:

Automated change adapter with integrated trigger probe
35mm styli extension
ruby styli (27mm len x 4mm dia)

A.30.6 Example tool definition statements

```
$$ Define the sensor mounting to the CMM
SNSMNT/XVEC,0.0,0.0,1.0,ZVEC,0.0,1.0,0.0,MNTLEN,0.0,0.0,0.0
$$ Define the wrist
SW(WRIST10MQ)=WRIST/ROTCEN,0.0,0.0,-30.0,0.0,0.0,1.0,0.0,-1.0,0.0,$
ANGLE,'B',-180.0,180.0,7.5,$
ROTCEN,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,-1.0,$
ANGLE,'A',0.0,105.0,7.5,$
MNTLEN,0.0,0.0,-43.0
$$ Define the tool in station 1
SS(Tool1)=SENSOR/PROBE,0.0,0.0,-115.0,0.0,0.0,-1.0,12.7,DISK,2.0
$$ Define the tool in station 2
SS(Tool2)=SENSOR/PROBE,0.0,0.0,-72.0,0.0,0.0,-1.0,35,DISK,5
$$ Define the cluster probe in station 3
SS(Tool3)=SENSOR/MLTPRB,5,$
1,0.0,0.0,-154.0,0.0,0.0,-1.0,2.0, $
2,30.0,0.0,-134.0,1.0,0.0,0.0,2.0, $
3,0.0,30.0,-134.0,0.0,1.0,0.0,2.0, $
4,-30.0,0.0,-134.0,-1.0,0.0,0.0,2.0, $
5,0.0,-30.0,-134.0,0.0,-1.0,0.0,2.0
$$ Define the tool in station 4
SS(Tool4)=SENSOR/PROBE,0.0,0.0,-158.0,0.0,0.0,-1.0,4.0,SPHERE
$$ Define the tool in station 5
SS(Tool5)=SENSOR/PROBE,0.0,0.0,-195.0,0.0,0.0,-1.0,4.0
$$ Define the tool in station 6
SS(Tool6)=SENSOR/PROBE,0.0,0.0,-365.0,0.0,0.0,-1.0,4.0,SPHERE
$$ Define the tool in station 7
SS(Tool7)=SENSOR/PROBE,0.0,0.0,-85.0,0.0,0.0,-1.0,30.0
$$ Define the tool in station 8
SS(Tool8)=SENSOR/PROBE,0.0,0.0,-111.0,0.0,0.0,-1.0,4.0

$$ Define the location of the above tools in the automated changer rack
TH(Rack1)=THLDEF/SS(Tool1),1,SS(Tool2),2,SS(Tool3),3,SS(Tool4),4$
,SS(Tool5),5,SS(Tool6),6,SS(Tool7),7,SS(Tool8),8
$$ Assemble the wrist to tool1
```

```
S(Sens1)=SNSDEF/BUILD,SW(PH10MQ),SS(Tool1)
$$ Assemble the wrist to tool2
S(Sens2)=SNSDEF/BUILD,SW(PH10MQ),SS(Tool2)
$$ Assemble the wrist to tool3
S(Sens3)=SNSDEF/BUILD,SW(PH10MQ),SS(Tool3)
$$ Assemble the wrist to tool4
S(Sens4)=SNSDEF/BUILD,SW(PH10MQ),SS(Tool4)
$$ Assemble the wrist to tool5
S(Sens5)=SNSDEF/BUILD,SW(PH10MQ),SS(Tool5)
$$ Assemble the wrist to tool6
S(Sens6)=SNSDEF/BUILD,SW(PH10MQ),SS(Tool6)
$$ Assemble the wrist to tool7
S(Sens7)=SNSDEF/BUILD,SW(PH10MQ),SS(Tool7)
$$ Assemble the wrist to tool8
S(Sens8)=SNSDEF/BUILD,SW(PH10MQ),SS(Tool8)
```

A.30.7 Example sensor calibration statements

```
$$ Define calibration feature in machine coordinates
F(Cal)=FEAT/SPHERE,OUTER,CART,250,250,-400,20,0,0,1
$$ Now select and calibrate positions for each tool
$$ Calibrate the sensors for Sens1
MODE/AUTO,MAN
SNSLCT/S(Sens1),SW(PH10MQ),'A',0,'B',0
CALIB/SENS,S(Sens1),F(Cal),4
ENDMES
SNSLCT/S(Sens1),SW(PH10MQ),'A',90,'B',0
CALIB/SENS,S(Sens1),F(Cal),4
ENDMES
```

A.31 SNSDEF (input format 2)

Sensor settings, along with scaling, automatic focus, and intensity, are activated with the SNSET statement. The sensor is selected, with the SNSLCT statement prior to taking a measurement with the MEAS statement.

For example:

```
S(camera_10x)=SNSDEF/VIDEO,INDEX,VEC,0,0,-1,0,0,-1,20,10.0,5.6
VW(box_1)=WINDEF/BOX,12.314,6.713,3.179,3.5,5.3,47
VL(bright)=LITDEF/SURF,0,1,0
SNSET/VW(box_1),VL(bright),.75,FOCUSY
SNSLCT/S(camera_10x)
MEAS/CIRCLE,F(sm_hole),8
PTMEAS/...
```

A.32 STR()

Example for STR():

```
DECL/CHAR,20,str_var
DECL/DOUBLE,uval
DECL/INTGR,ival
uval = ASSIGN/3.14159265358979323
str_var = ASSIGN/STR(uval, 6:3)
$$ str_var will be ' 3.142'
str_var = ASSIGN/STR(uval)
$$ str_var will be '3.141592' number of decimals DME dependent
```

```
uval = ASSIGN/-1.23
str_var = ASSIGN/STR(uval, 6:3)
$$ str_var will be '-1.230'
str_var = ASSIGN/STR(uval)
$$ str_var will be '-1.230000' number of decimals DME dependent
ival = ASSIGN/-20
str_var = ASSIGN/STR(ival, 6:3)
$$ str_var will be '-20.000' note that a 7 character string is returned
str_var = ASSIGN/STR(ival)
$$ str_var will be '-20'
uval = ASSIGN/-23.23
str_var = ASSIGN/STR(uval, 6:3)
$$ str_var will be '-23.230' note that a 7 character string is
$$ returned with 3 decimal places
uval = ASSIGN/-0.03
str_var = ASSIGN/STR(uval, 6:3)
$$ str_var will be '-0.030'
```

A.33 TEXT

```
TEXT/OPER,'Use Paul''s setup instructions from the last job.'
```

This results in the following message sent to the screen:

```
Use Paul's setup instructions from the last job.
```

A.34 TOL/GTOL

Some examples of format variation and effect:

Common data for examples:

PTBUFF/ON is in effect

F(FEAT1) is a PLANE

F(FEAT3) is a CYLNDR

for GTOL,

input

Maximum number of iterations = 50 iterations

Minimum Shift = .1 MM

Minimum Rotation = .001 degrees

output

Actual number of iterations = 30

Part found to be = out of tolerance

found to interfere = 3 points

3 points out of 60 points represents 5% bad, 95% good points

A.34.1 Example 1, just GO/NOGO info desired:

input format:

```
T(SFGTOL1)=TOL/GTOL,XYDIR,ZAXIS,50,.1,.001
```

output format: (part out of tol.)

```
TA(SFGTOL1)=TOL/GTOL,30,OUTOL,TRMATX,1,0,0,0,1,0,0,0,1,.1,-.1,0
```

A.34.2 Example 2, GO/NOGO and percentage of good points data desired:

input format:

```
T(SFGTOL2)=TOL/GTOL,XYDIR,ZAXIS,50,.1,.001,PERCNT
```

output format: (part out of tol. in 5% of points)

```
TA(SFGTOL2)=TOL/GTOL,30,OUTOL,TRMATX,1,0,0,0,1,0,0,0,1,.1,-.1,0,95
```

A.34.3 Example 3, GO/NOGO and interference point/s info desired:

input format:

```
T(SFGTOL3)=TOL/GTOL,XYDIR,ZAXIS,50,.1,.001,INTFPT
```

output format: (part out of tol. in 3 places)

```
TA(SFGTOL3)=TOL/GTOL,30,OUTOL,TRMATX,1,0,0,0,1,0,0,0,1,.1,-.1,0,INTFPT
FA(FEAT1)[3]=FEAT/PLANE,PTDATA,CART,10.01133,11.00121,12.12345
FA(FEAT3)[2]=FEAT/CYLNDR,PTDATA,CART,9.999,8.888,6.5555
FA(FEAT3)[5]=FEAT/CYLNDR,PTDATA,CART,5.55555,4.44444,3.33333
ENDAT
```

A.35 VALUE

```
DECL/REAL,VAR
EVAL/FA(lname),T(lname)
VAR=VALUE/TA(lname),ACT
```

A.36 Vector variable values

```
$$ to measure a circle using an arbitrary number of sample points
UNITS/MM,ANGDEC
DECL/INTGR,loop,ptcnt
DECL/VECTOR,center,coord,ijk_vector
center=ASSIGN/VCART(50,100,10)
ijk_vector=ASSIGN/VCART(0,0,1)
ptcnt=PROMPT/'Total point count'
MEAS/CIRCLE,F(circ),ptcnt
  DO/loop,1,ptcnt
    coord=ASSIGN/center + VPOL(10,loop*360/ptcnt,0)
    ijk_vector=ASSIGN/VPOL(1,loop*360/ptcnt,0)
    PTMEAS/CART,VECX(coord),VECY(coord),VECZ(coord),VECX(ijk_vector), $
        VECY(ijk_vector),VECZ(ijk_vector)
  ENDDO
ENDMES
```

```
$$ to project the line onto the axis
DECL/VECTOR,lyne,axis,prj
lyne=ASSIGN/VCART(25,10,15)
axis=ASSIGN/VPOL(100,30,0)
$$ scale axis by ((dot product of lyne & axis) / (square of mag(axis)))
prj=ASSIGN/axis * VDOT(lyne,axis)/(VMAG(axis) ** 2)
```

A.37 WRITE

```
DECL/GLOBAL,REAL,RNom,RAct
DECL/GLOBAL,INTGR,sm_int,bg_int
```

```
DECL/GLOBAL,CHAR,6,MeasID
DECL/GLOBAL,REAL,Numl,Num2,Num3
. . .
MeasID=ASSIGN/'D51861'
RNom=ASSIGN/6.0
RAct=ASSIGN/5.871
sm_int=ASSIGN/44
bg_int=ASSIGN/32767
DID(meas_data)=DEVICE/STOR,'/usr/data/meas_dataO40893'
OPEN/DID(meas_data),DIRECT,OUTPUT,APPEND
WRITE/DID(meas_data),MeasID:3,'  ',RNom:4,'  ',RAct:4:1
produces   --->  D5l    6  5.9
                 123456789012345
WRITE/DID(meas_data),'The  MeasID  ',MeasID,'  actual  ',RAct:6:4
produces   --->  The MeasID D51861 actual 5.8710
                 12345678901234567890123456789012345
WRITE/DID(meas_data),sm_int:10
produces   --->          44
                 12345678901234567890
WRITE/DID(meas_data),bg_int:3
produces   --- > 327
                 123456
Numl=ASSIGN/10.4
Num2=ASSIGN/77.7
Num3=ASSIGN/4.4
DID(line_print)=DEVICE/PRINT,'Lab_Printer'
OPEN/DID(line_print),DIRECT,OUTPUT
WRITE/DID(line_print),Numl:4:1,' ',Num2:4:1,CHR(12),$
Num2:5:2,Num3:5:2,CHR(10),CHR(13),Numl:4:1
```

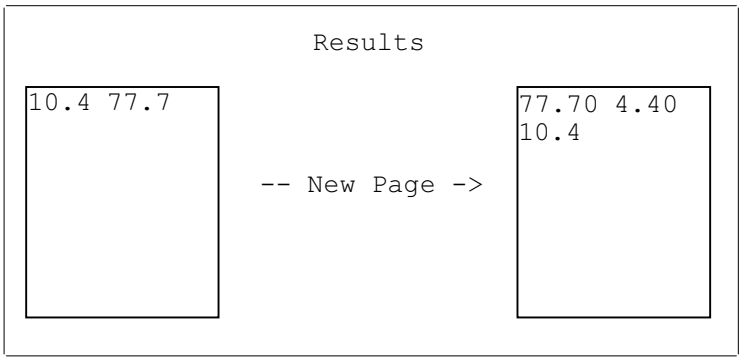


Figure A.12 — Results of writing to a line printer

Note: In the above example, CHR(12) causes a form feed or a new page, CHR(10) causes a line feed or new line, and CHR(13) causes a carriage return. A variable field width is optional. Its effect depends on the data type of the variable to which it applies. If the field type is numeric and it exceeds the total field width 'x', then DMIS writes a field of asterisks to denote an overflow condition. Numbers are padded with leading spaces. If the field type is a character string and it exceeds the total field with 'x', then DMIS truncates to the first 'x' characters of the string. Character strings are left justified. The system default is the output device's system default. Characters enclosed between apostrophes are displayed literally.

A.38 XTRACT

An example of the use of the XTRACT statement is as follows:

```
DMISMN/'PDGS NC-Inspection',05.1
V(STOR_LABEL)=VFORM/NOM,ACT,DEV,AMT
DISPLY/STOR,V(STOR_LABEL)
FILNAM/'Sample XTRACT file',05.1
MODE/PROG,MAN
PRCOMP/ON
UNITS/MM,ANGDEC
WKPLAN/XYPLAN
F(SEC1)=FEAT/GCURVE,CART,-15.8747,106.7010,0.0,0.1472,-0.9891,0.0
MEAS/GCURVE,F(SEC1),70
PTMEAS/CART,-00165.0000,000084.5127,000029.0204,-0.0343,00.0715,00.9969
PTMEAS/CART,-00159.9998,000085.2563,000029.1408,-0.0346,00.0693,00.9970
PTMEAS/CART,-00154.9997,000086.0002,000029.2644,-0.0350,00.0671,00.9971
PTMEAS/CART,-00149.9998,000086.7443,000029.3913,-0.0353,00.0649,00.9973
PTMEAS/CART,-00144.9999,000087.4886,000029.5214,-0.0356,00.0628,00.9974
. . .
. . .
PTMEAS/CART,-00044.9999,000102.3677,000032.8083,-0.0422,00.0190,00.9989
PTMEAS/CART,-00040.0000,000103.1118,000032.9839,00.0054,00.0175,00.9998
PTMEAS/CART,-00035.0023,000103.8554,000032.3061,00.2558,00.0184,00.9666
GOTO/-00029.3849,000104.6904,000039.5482
PTMEAS/CART,-00029.9962,000104.5996,000030.2039,00.5066,00.0180,00.8620
GOTO/-00023.2685,000105.6066,000036.2341
PTMEAS/CART,-00024.9993,000105.3431,000026.0185,00.7571,00.0157,00.6531
PTMEAS/CART,-00020.0000,000106.0882,000018.6736,00.8371,00.0116,00.5469
PTMEAS/CART,-00015.0054,000106.8323,000011.0184,00.8370,00.0076,00.5472
. . .
. . .
PTMEAS/CART,000170.0002,000134.3481,-00051.4293,-0.1511,-0.0693,00.9861
PTMEAS/CART,000175.0001,000135.0954,-00050.6092,-0.1514,-0.0715,00.9859
PTMEAS/CART,000180.0001,000135.8444,-00049.7856,-0.1517,-0.0736,00.9857
ENDMES
F(FLAT1)=FEAT/LINE,BND,CART,-165.5077,84.4372,29.0084,$
-40.9563,102.9693,32.9686,0.0,0.0,1.0
F(FLAT2)=FEAT/LINE,BND,CART,-23.3978,105.5819,23.8876,$
28.8941,113.3618,-55.8022,0.0,0.0,1.0
F(FLAT3)=FEAT/LINE,BND,CART,58.3218,117.7388,-68.8787,$
182.0318,136.1490,-49.4500,0.0,0.0,1.0
F(ARC_1)=FEAT/ARC,4POINT,INNER,$
-40.9563,102.9693,32.9686,-29.8176,104.6265,30.0978,$
-23.6305,105.5478,24.2439,-40.1407,103.0903,12.6875
F(ARC_2)=FEAT/ARC,4POINT,INNER,$
28.8941,113.3618,-55.8022,41.6952,115.2661,-66.6343,$
58.3218,117.7388,-68.8787,53.8309,117.0680,-38.7647
XTRACT/F(FLAT1),FA(SEC1)
XTRACT/F(FLAT2),FA(SEC1)
XTRACT/F(FLAT3),FA(SEC1)
XTRACT/F(ARC_1),FA(SEC1)
XTRACT/F(ARC_2),FA(SEC1)
OUTPUT/FA(FLAT1)
OUTPUT/FA(FLAT2)
OUTPUT/FA(FLAT3)
OUTPUT/FA(ARC_1)
OUTPUT/FA(ARC_2)
ENDFIL
```

Annex B
(informative)

Descriptive Figures

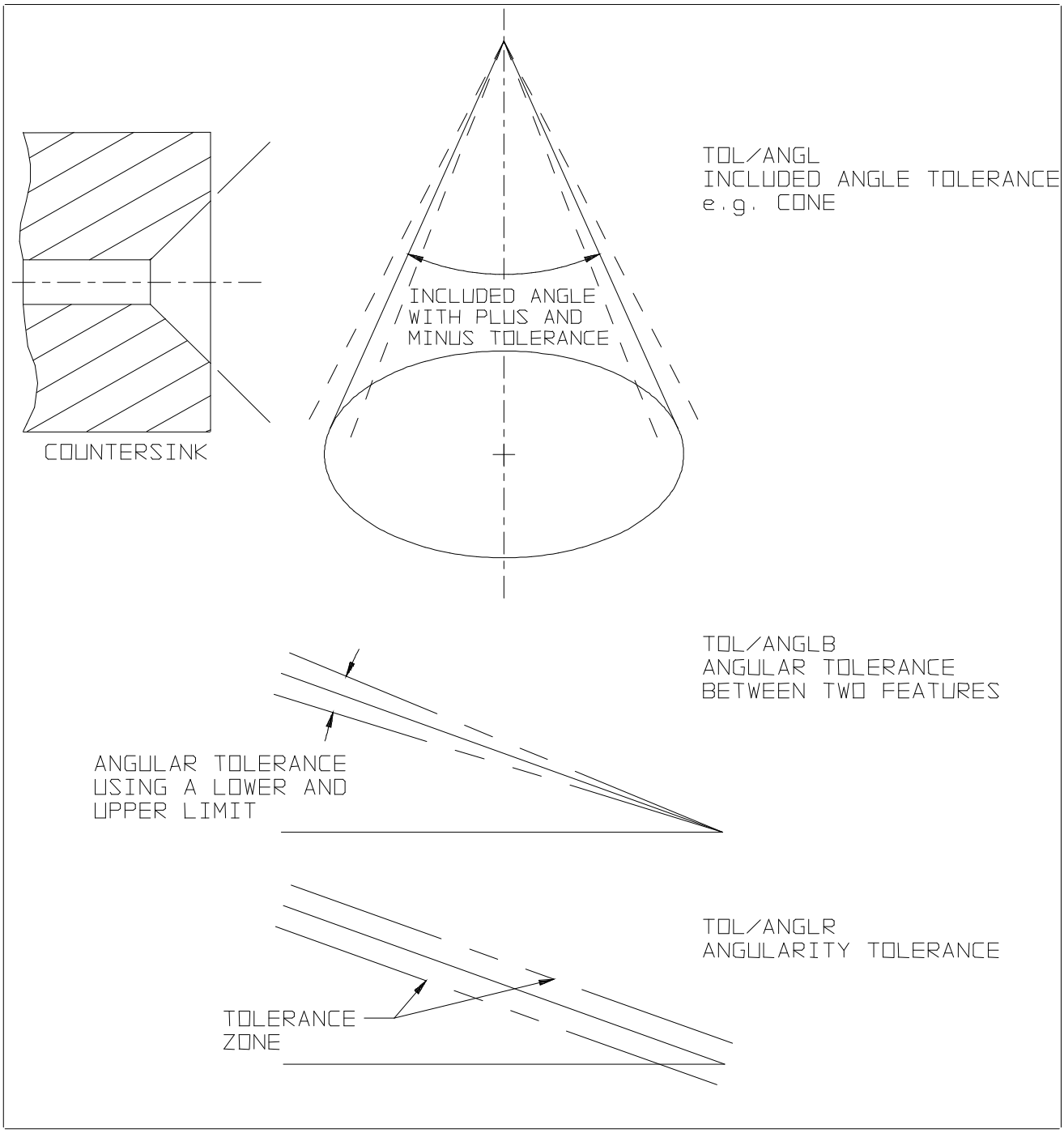


Figure B.1 —Angle tolerance description

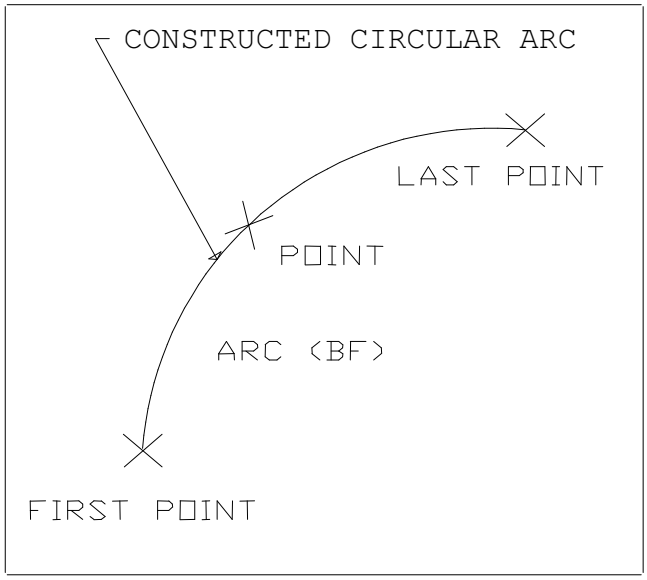


Figure B.2 — Constructed circular arc

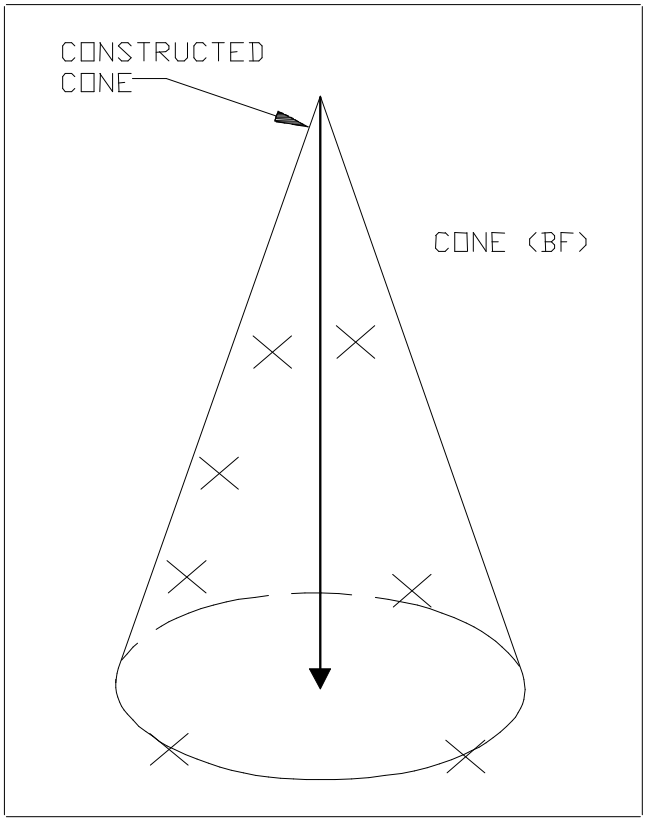


Figure B.3 — Constructed cone

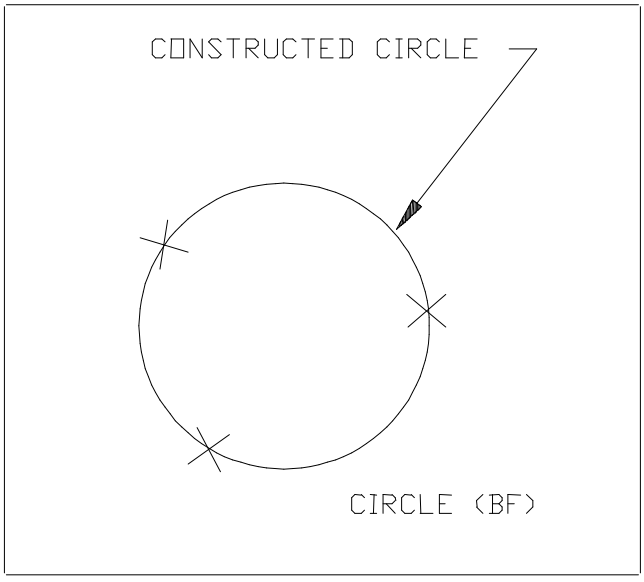


Figure B.4 — Constructed circle

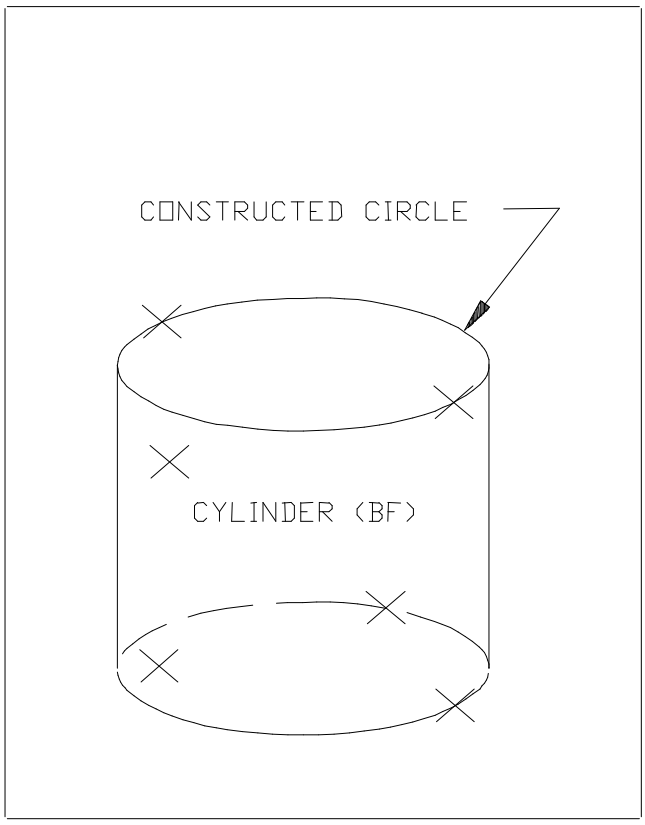


Figure B.5 — Constructed circle

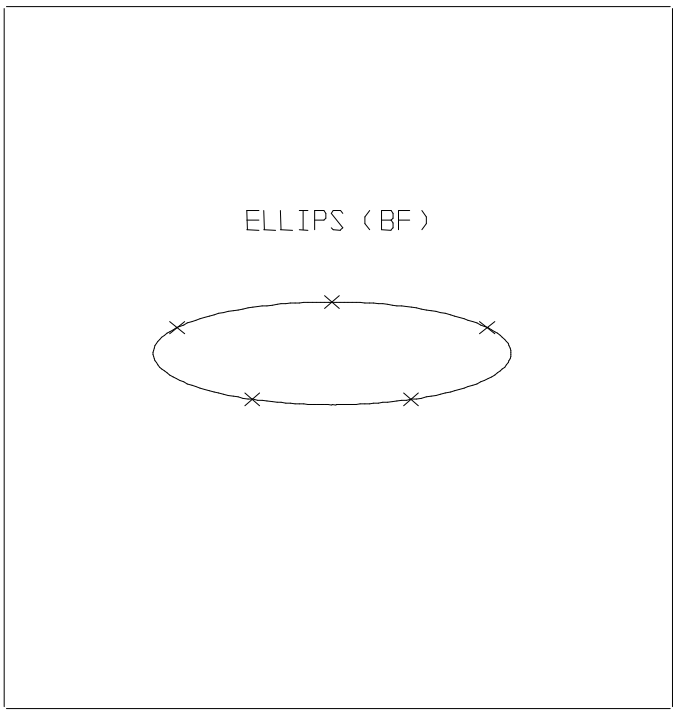


Figure B.6 — Constructed ellipse

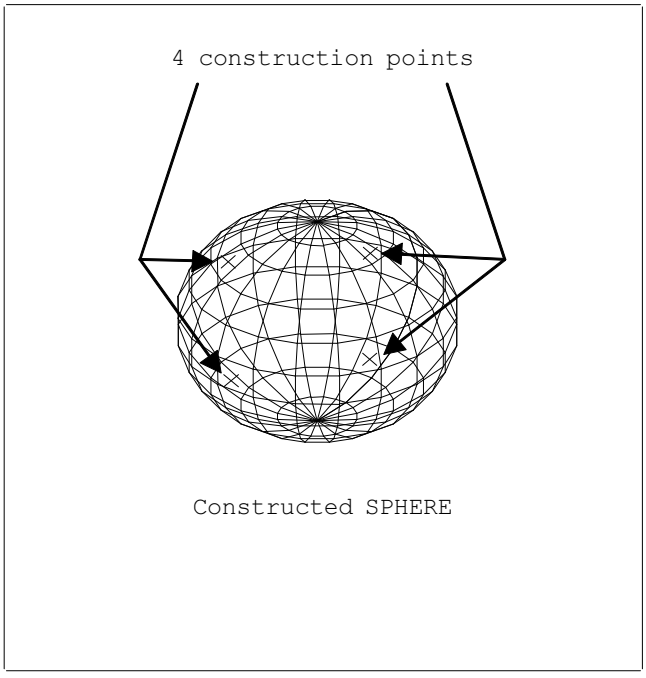


Figure B.7 — Constructed sphere

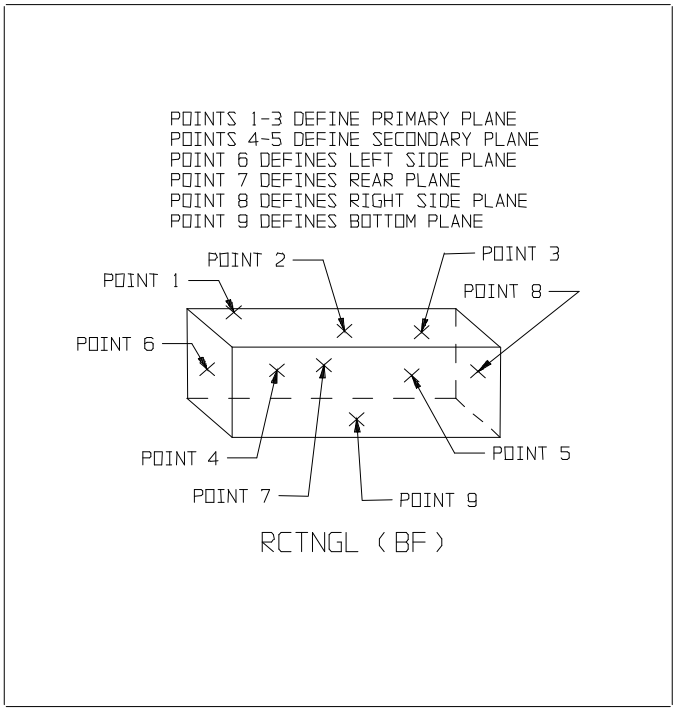


Figure B.8 — Constructed rectangle

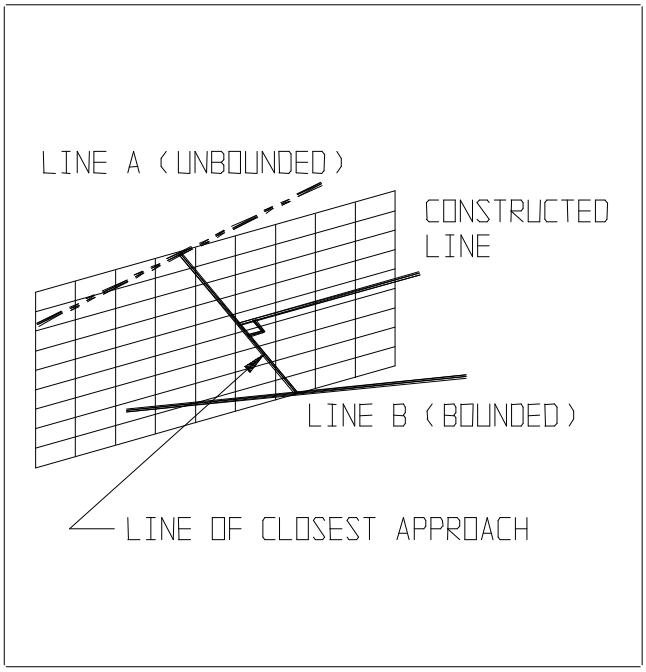


Figure B.9 — Constructed line

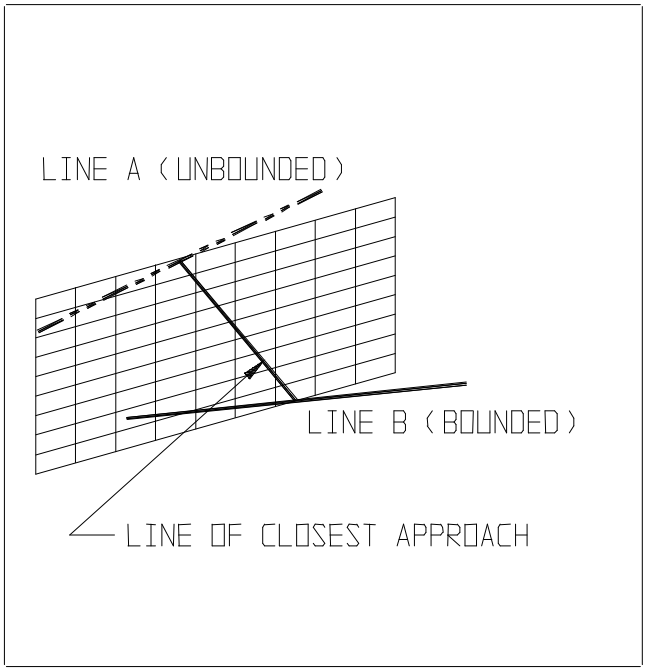


Figure B.10 — Constructed plane

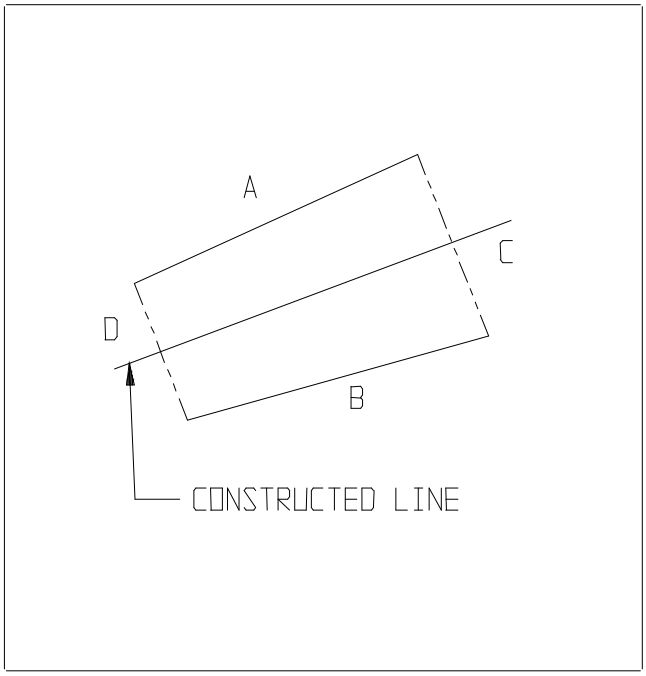


Figure B.11 — Constructed line

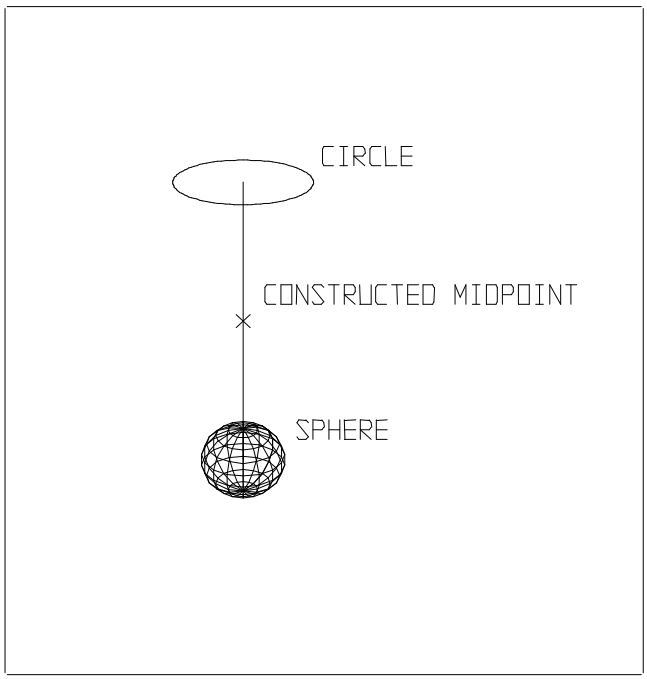


Figure B.12 — Constructed mid-point

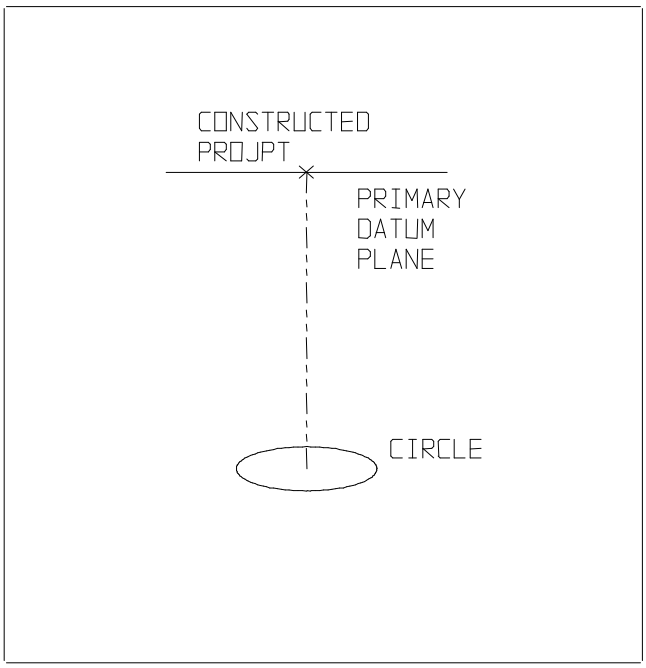


Figure B.13 — Constructed projected point

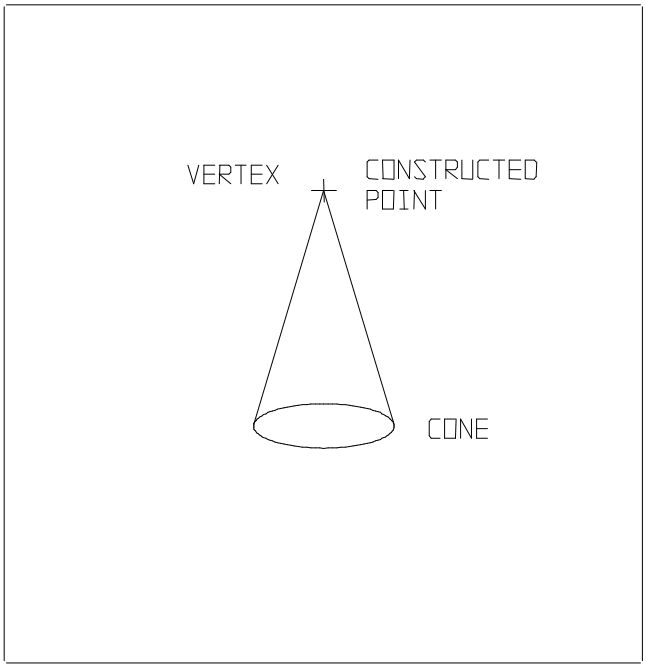


Figure B.14 — Constructed point

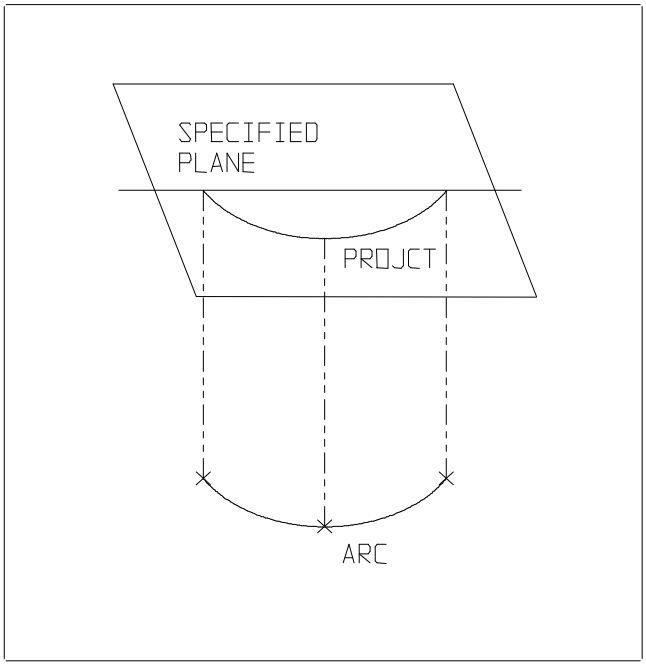


Figure B.15 — Constructed projected arc

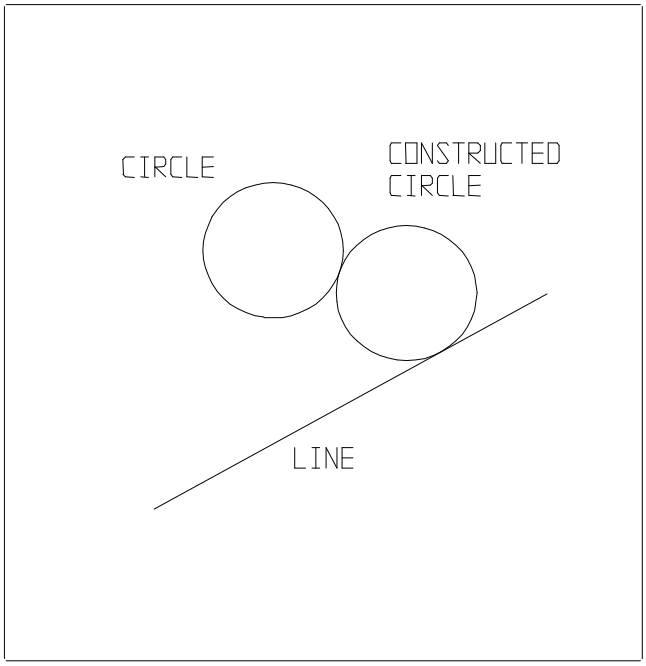


Figure B.16 — Constructed circle

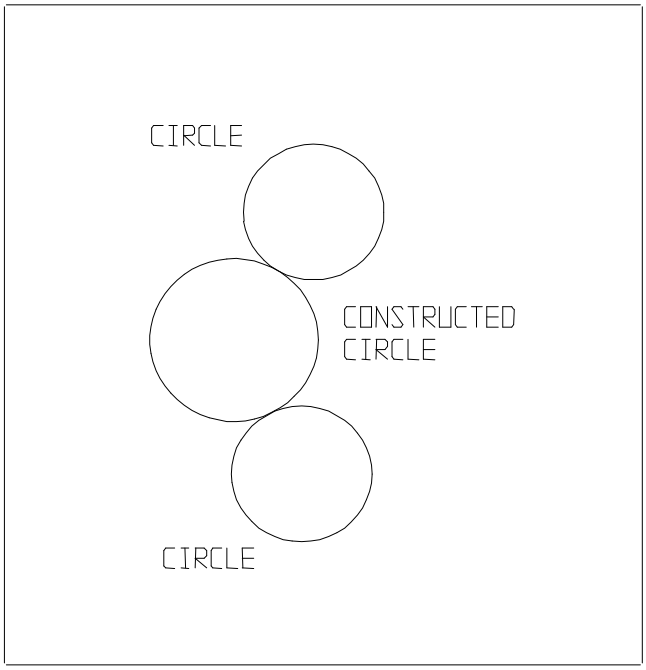


Figure B.17 — Constructed circle

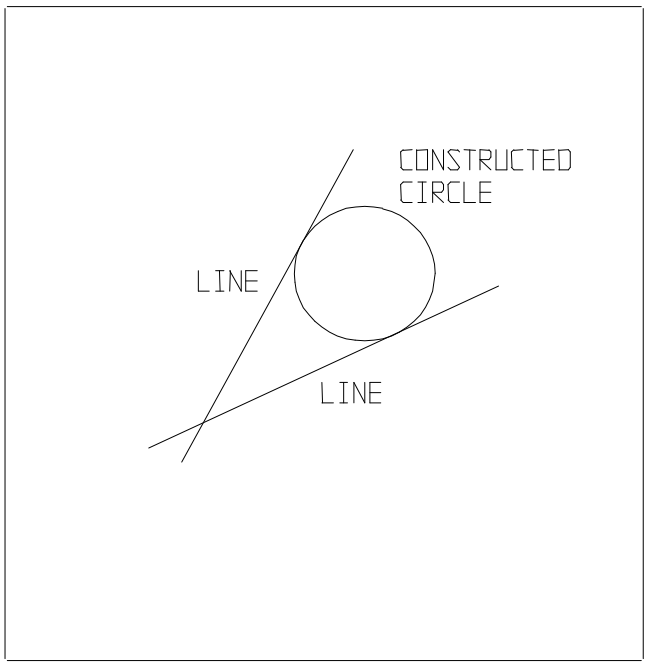


Figure B.18 — Constructed circle

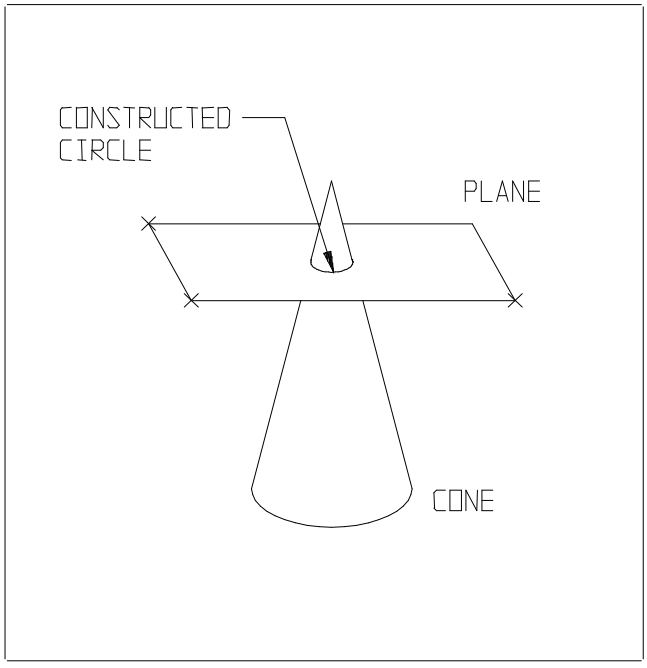


Figure B.19 — Constructed circle

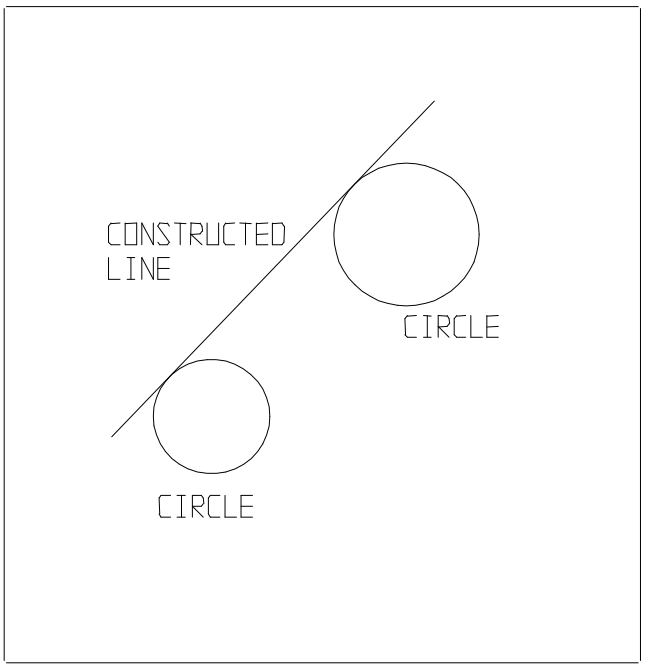


Figure B.20 — Constructed line

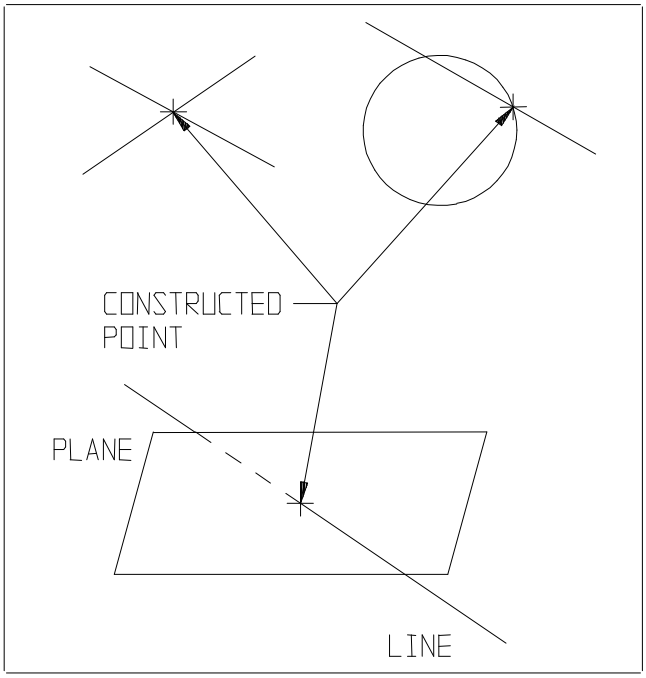


Figure B.21 — Constructed point

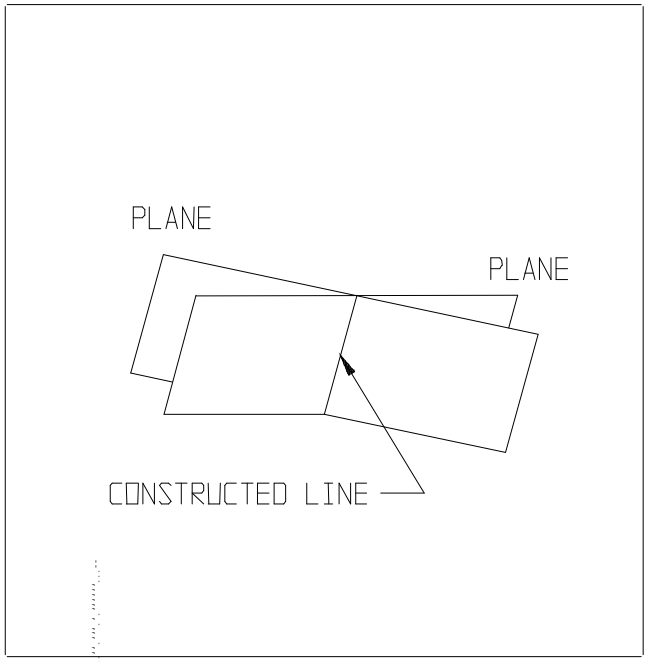


Figure B.22 — Constructed line

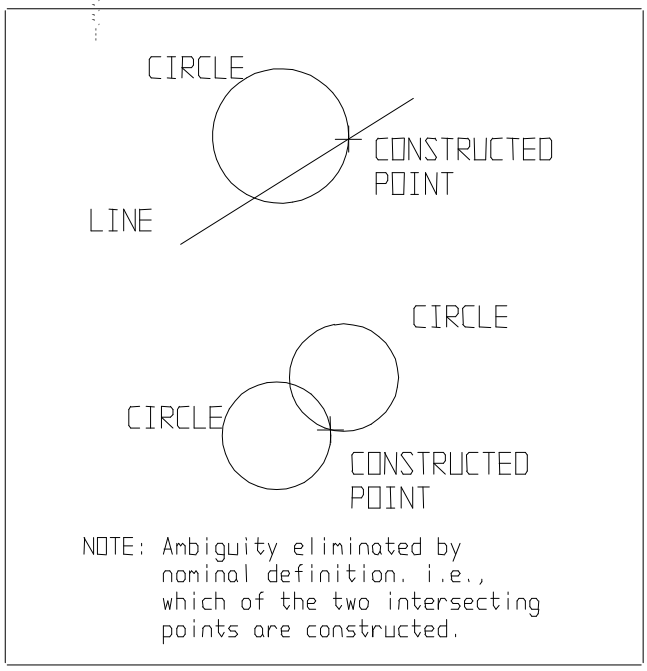


Figure B.23 — Constructed point

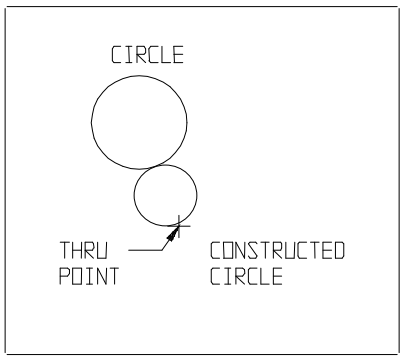


Figure B.24 — Constructed circle

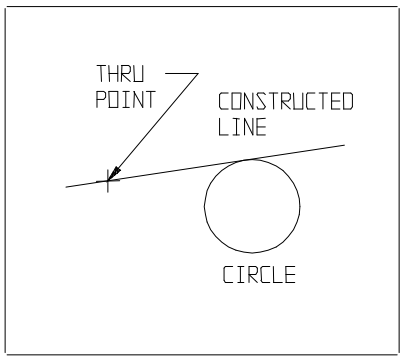


Figure B.25 — Constructed line

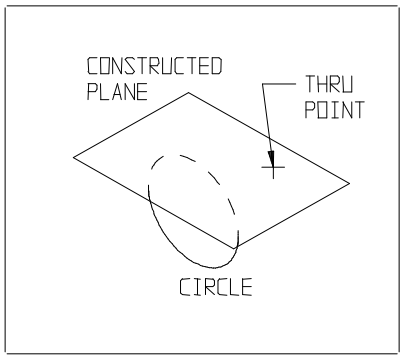


Figure B.26 — Constructed plane

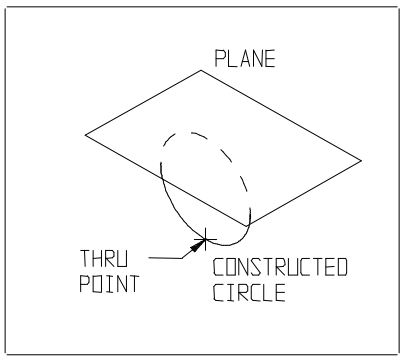


Figure B.27 — Constructed circle

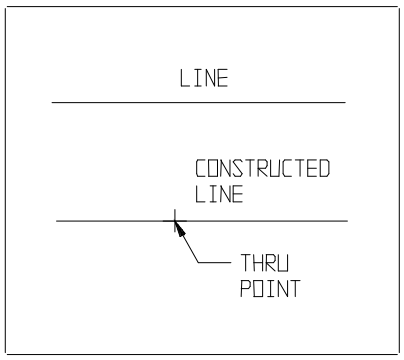


Figure B.28 — Constructed line

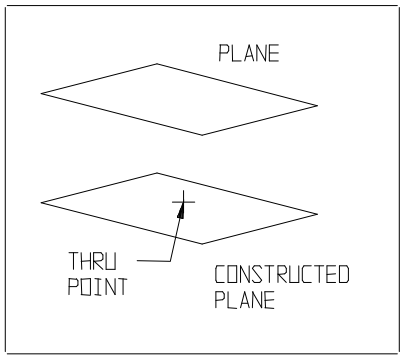


Figure B.29 — Constructed plane

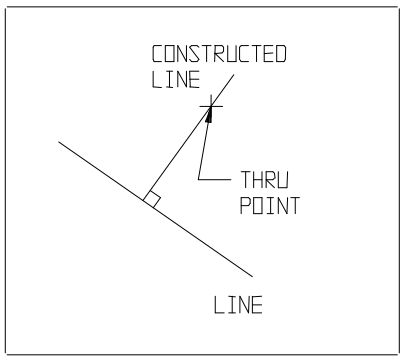


Figure B.30 — Constructed line

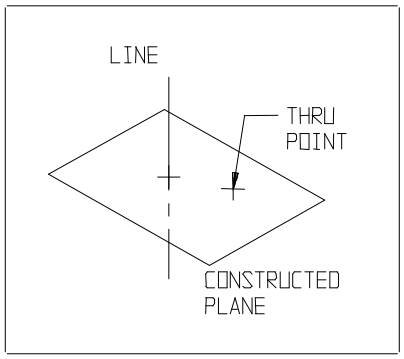


Figure B.31 — Constructed plane

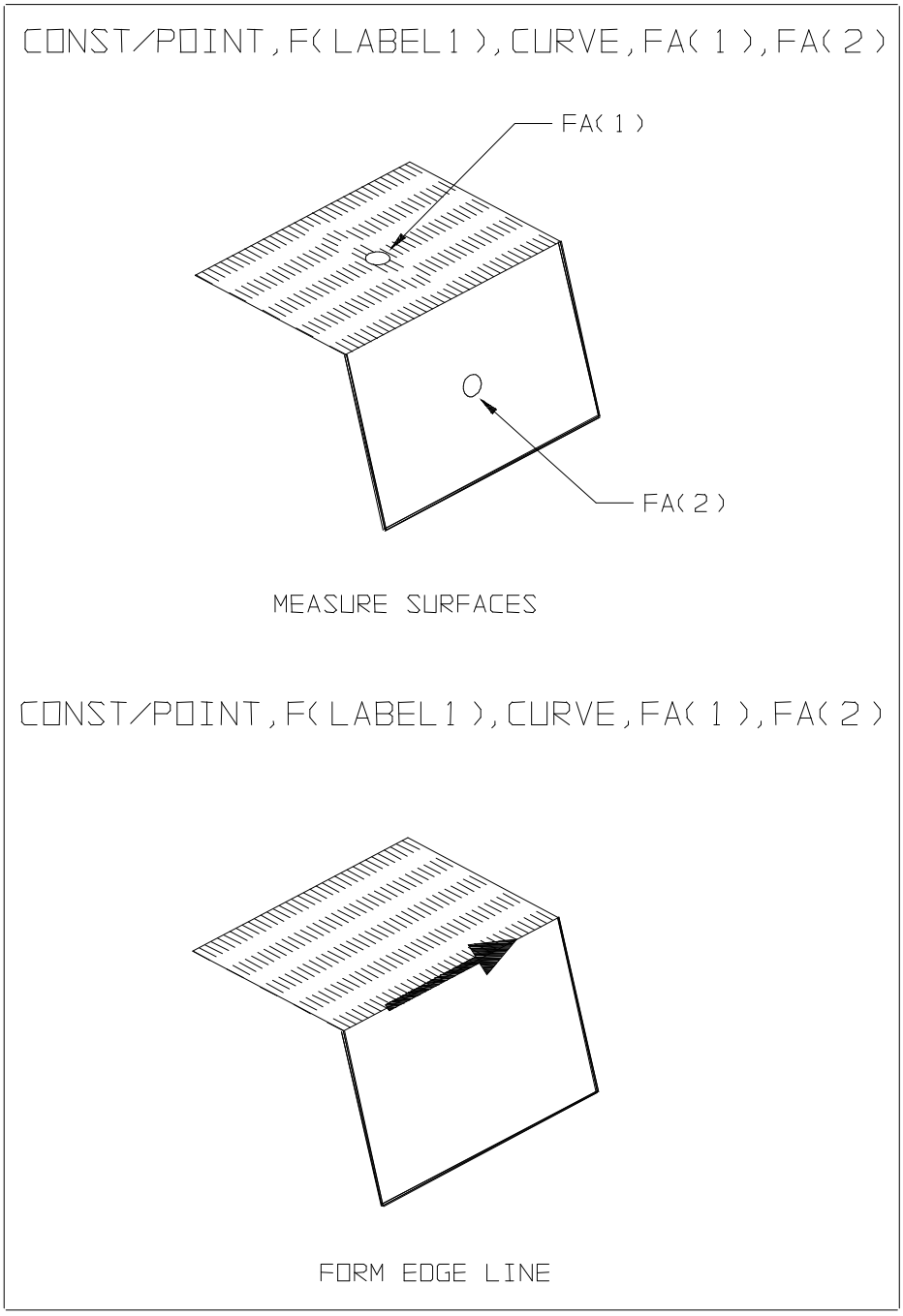


Figure B.32 — Construction of a point on a curve

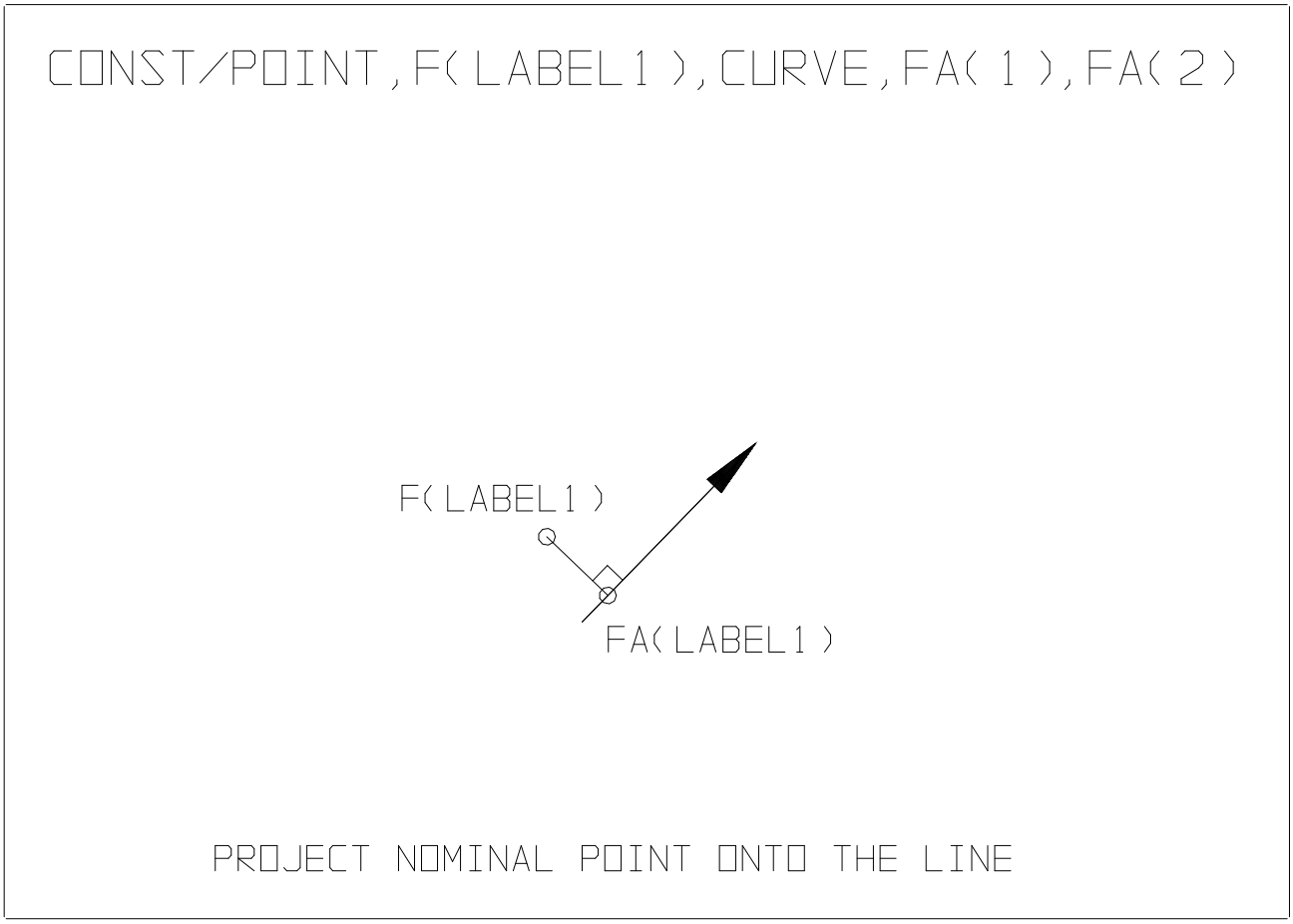


Figure B.33 — Construction of a point on a curve

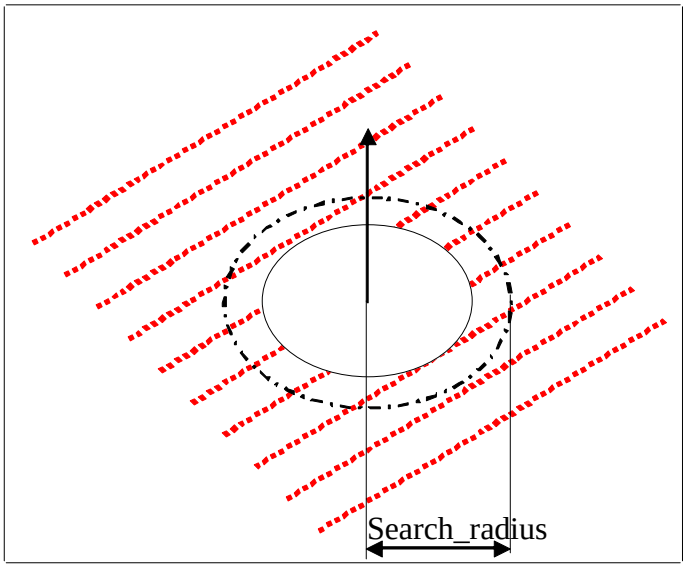


Figure B.34 — Construction, CONST/CIRCLE,... RETRIEVE

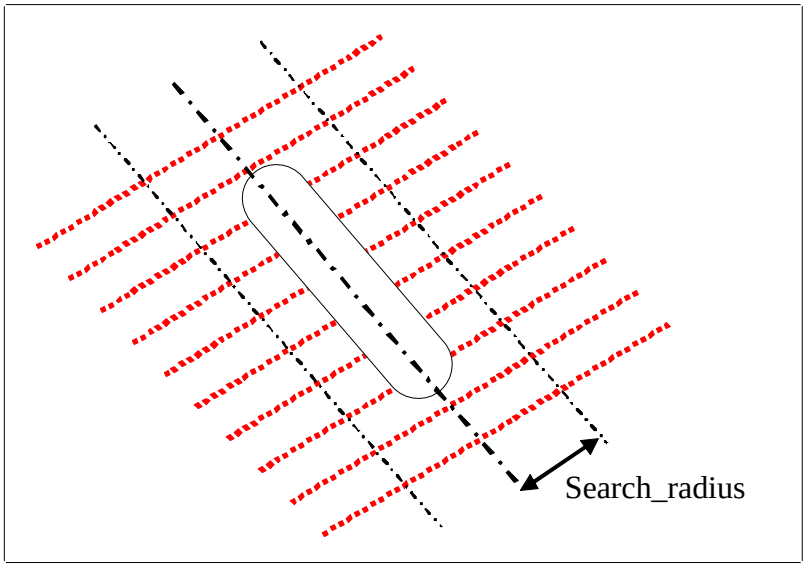


Figure B.35 — Construction, CONST/CPARLN,... RETRIEVE

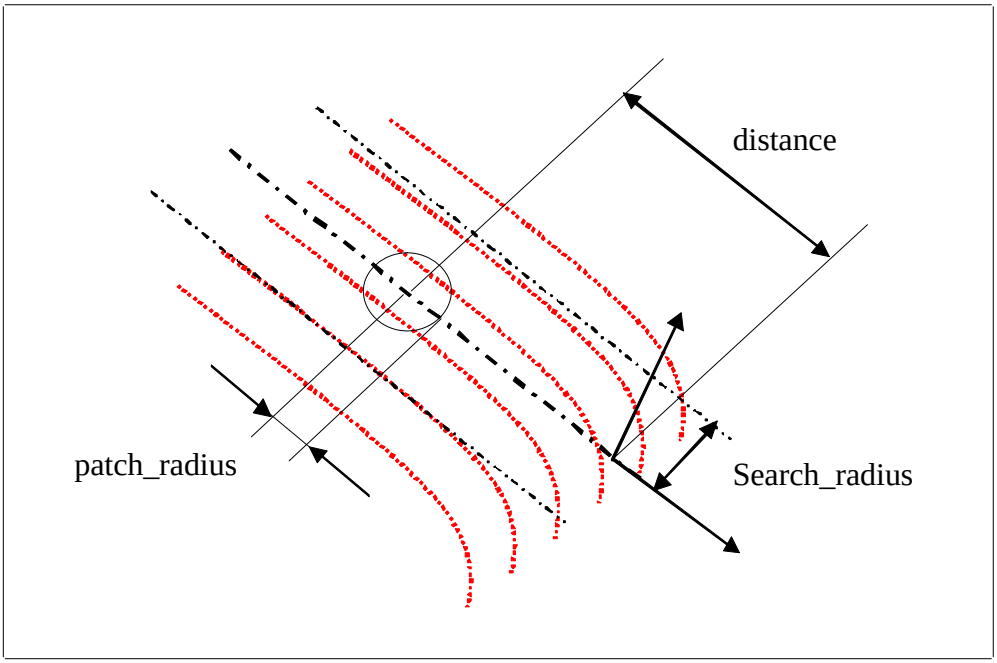


Figure B.36 — Construction, CONST/EDGEPT,... RETRIEVE

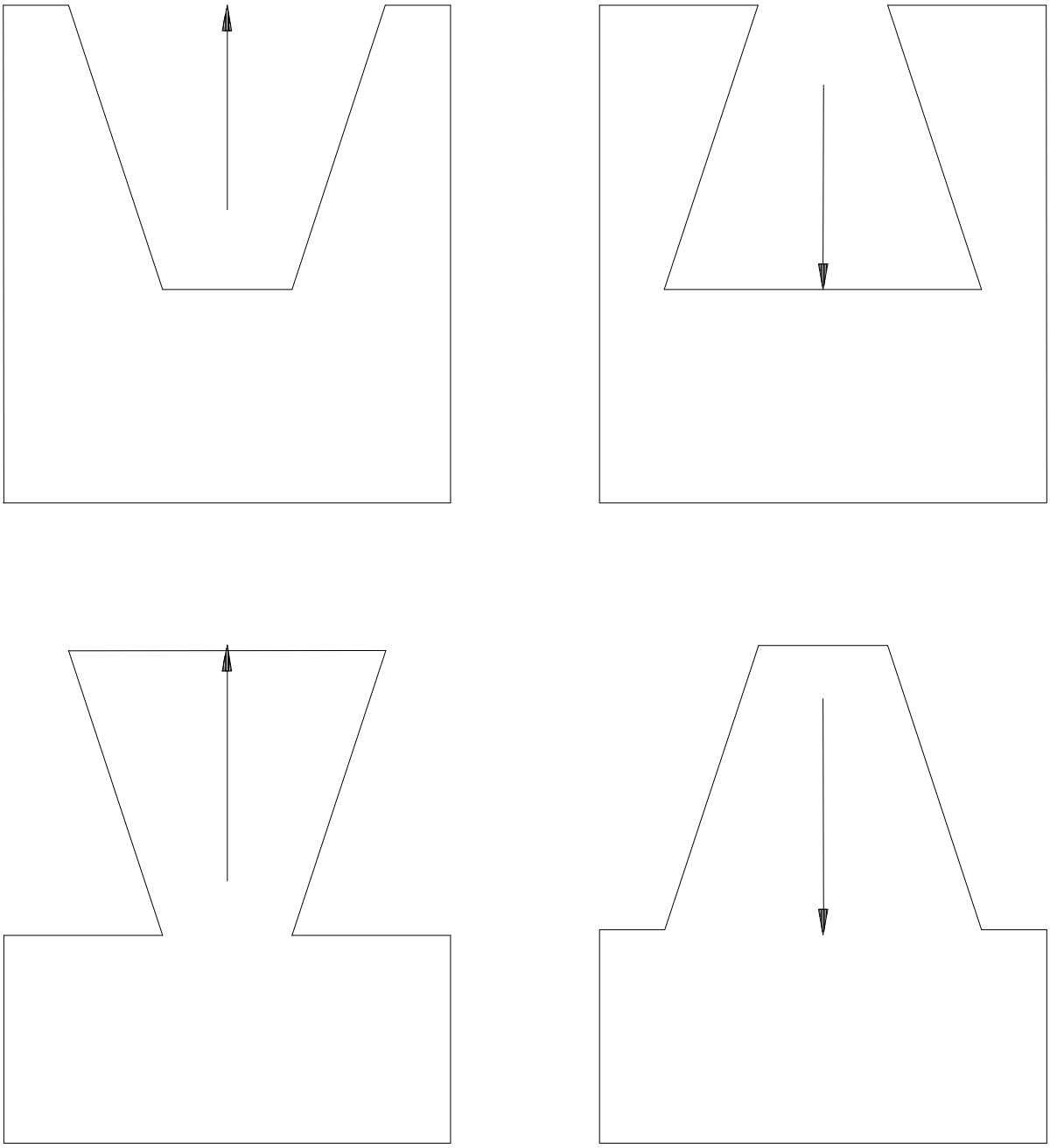


Figure B.37 — A cone feature, vector illustration

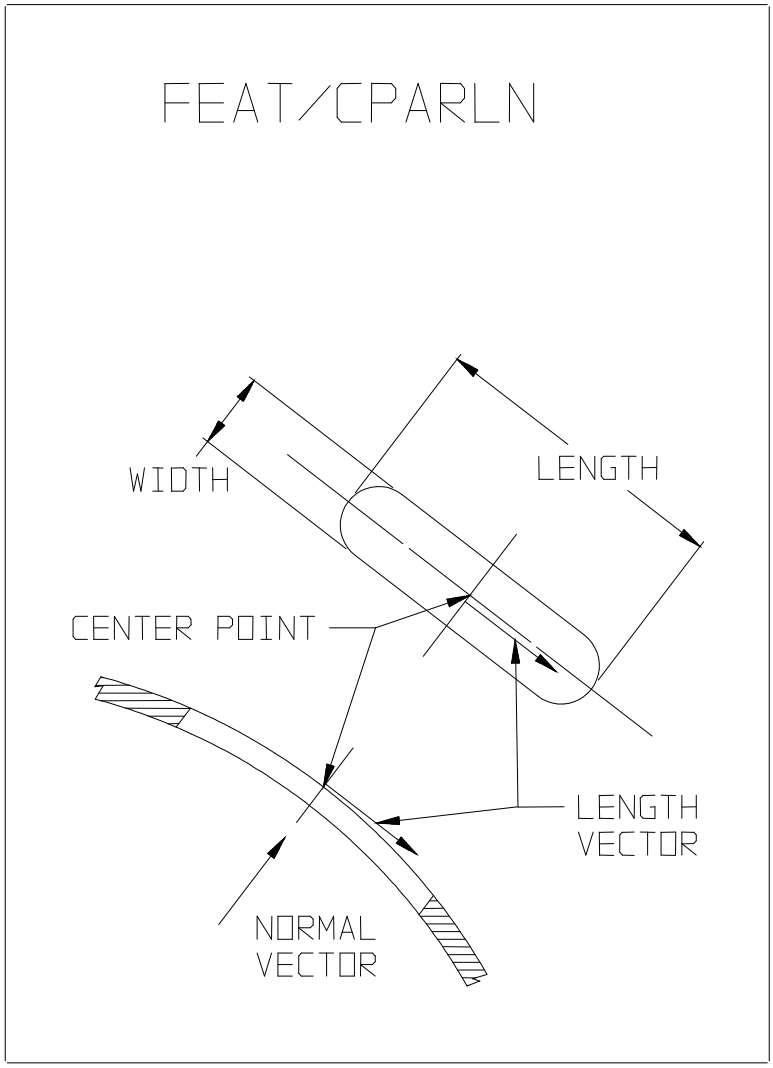


Figure B.38 — A centered parallel line feature

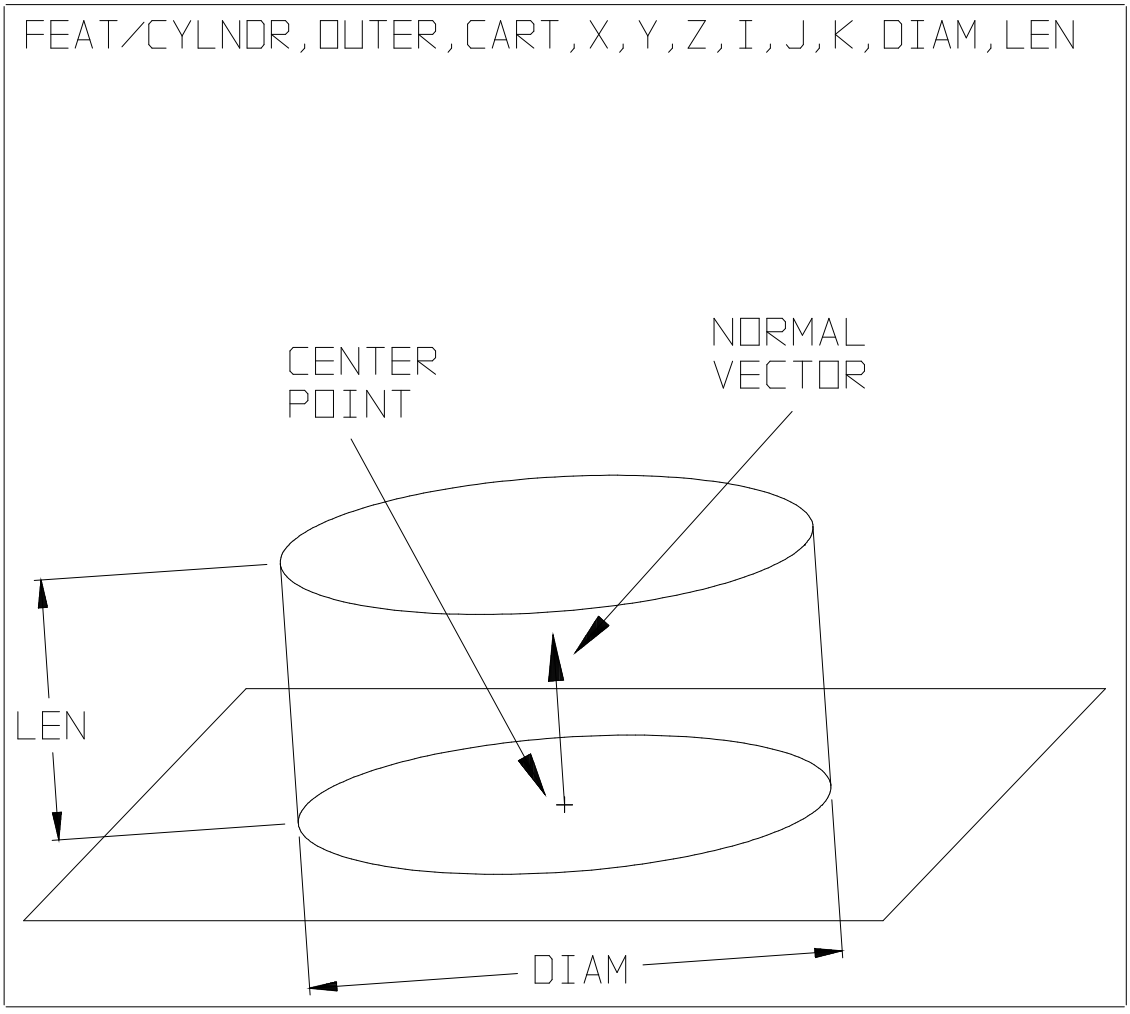


Figure B.39 — A cylinder feature with the 'len' parameter specified

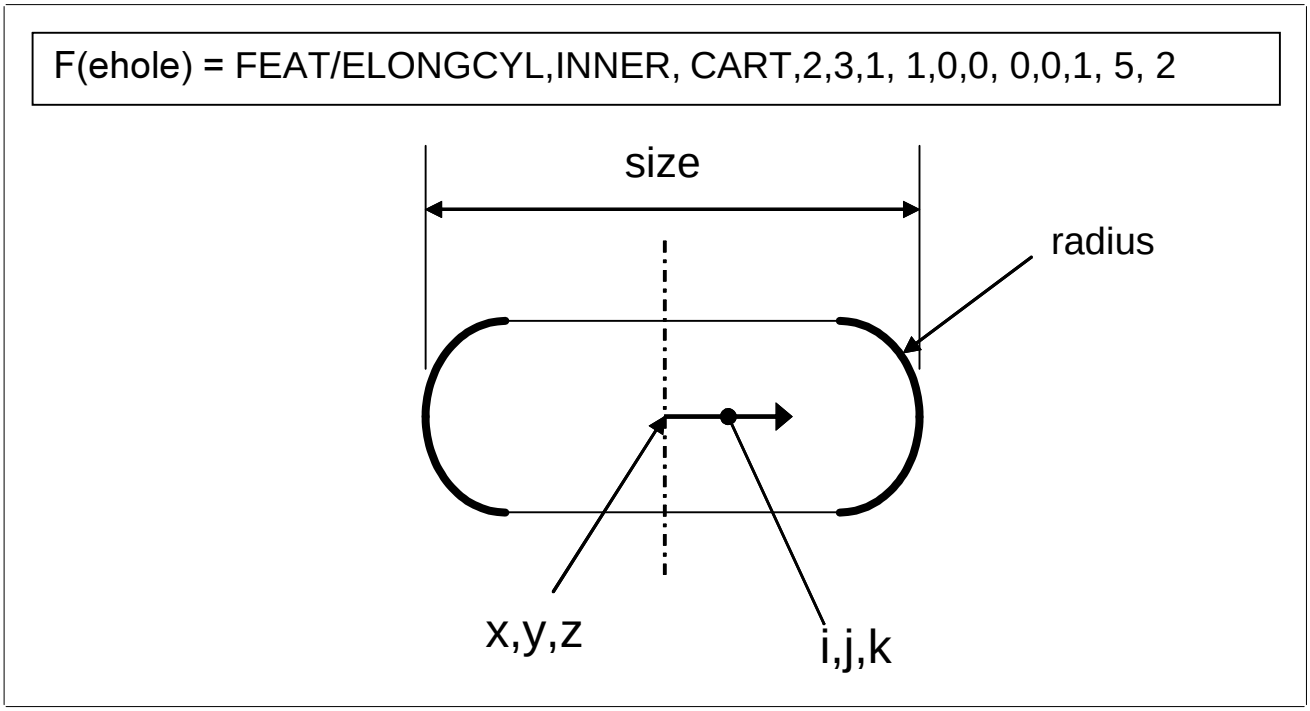


Figure B.40 — An elongated cylinder feature

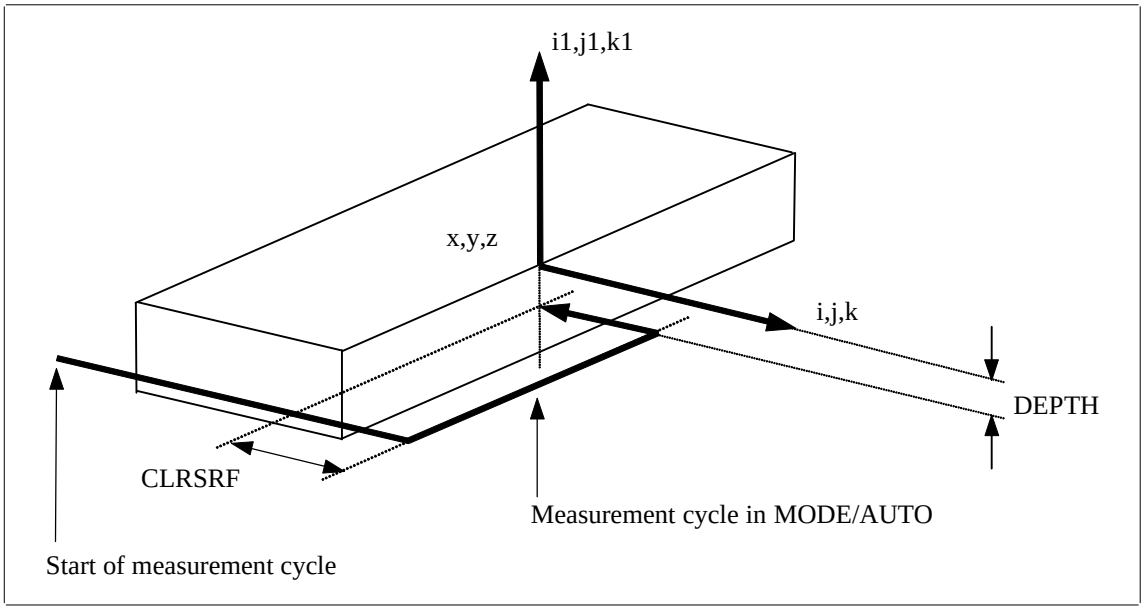


Figure B.41 — An edgepoint feature

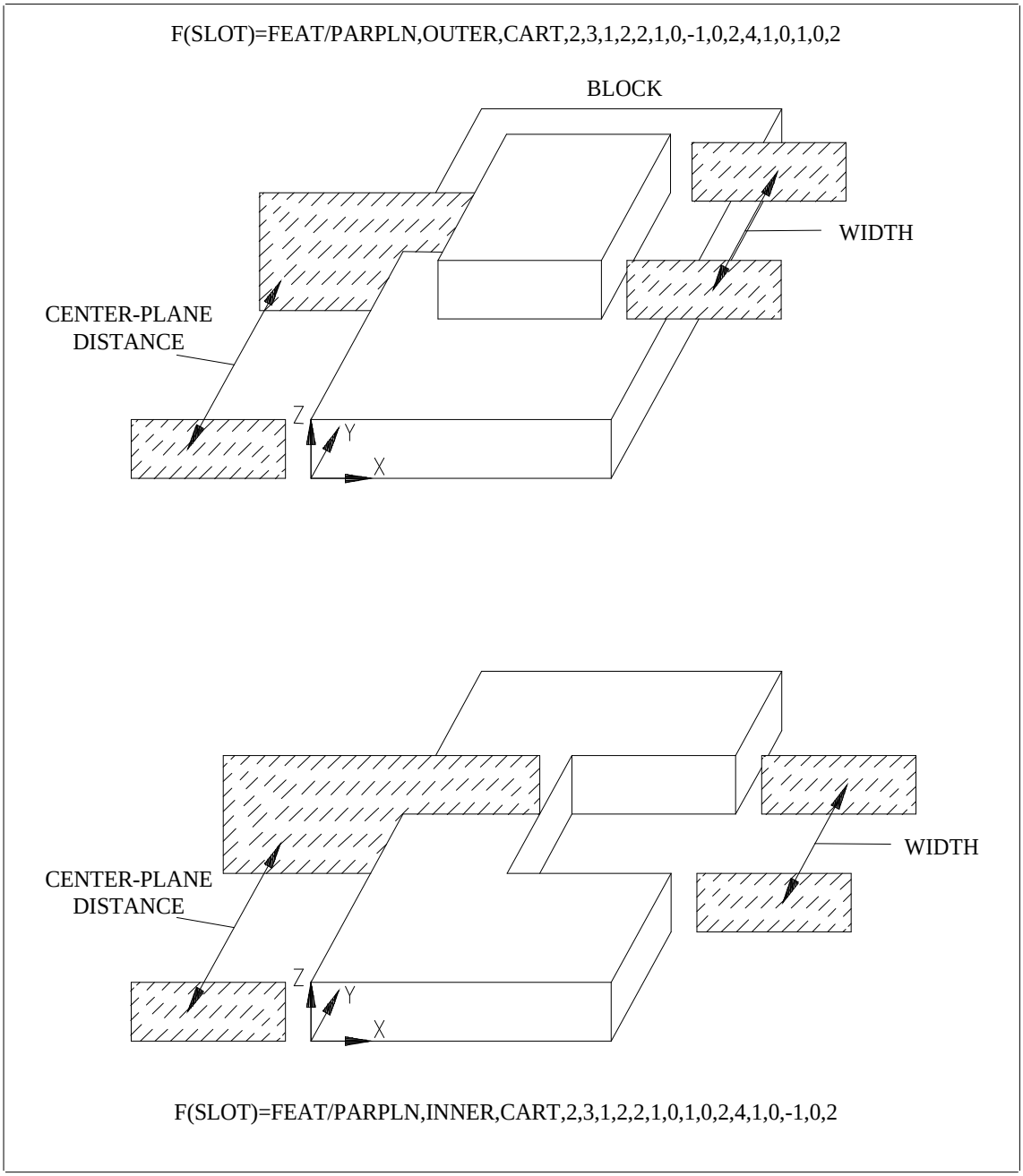


Figure B.42 — A parallel plane feature

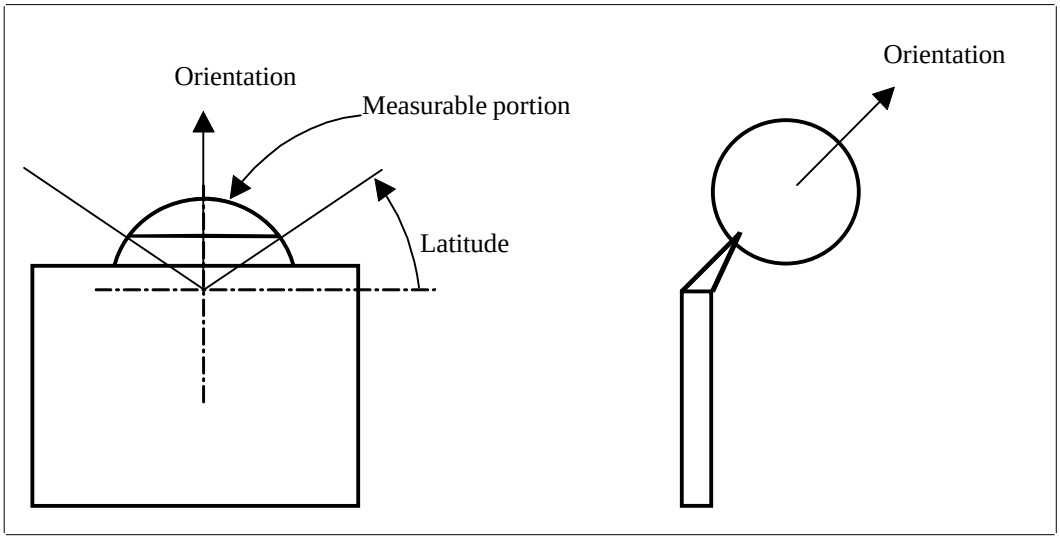


Figure B.43 — A sphere feature

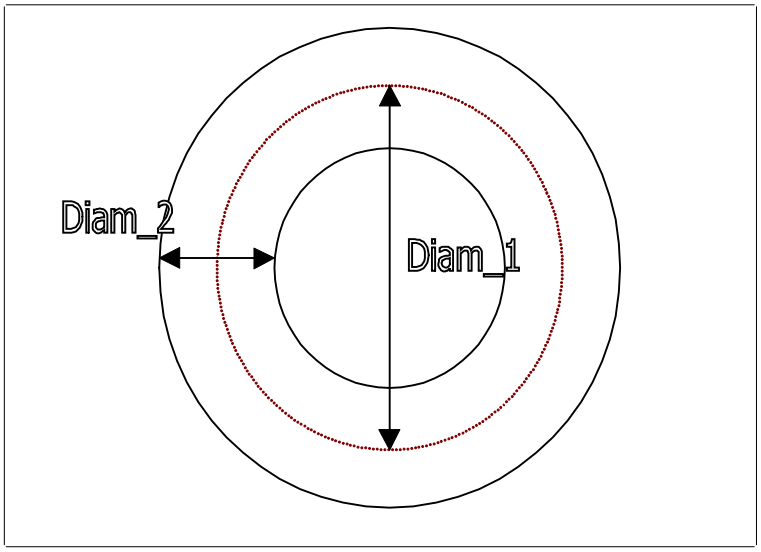


Figure B.44 — A torus feature

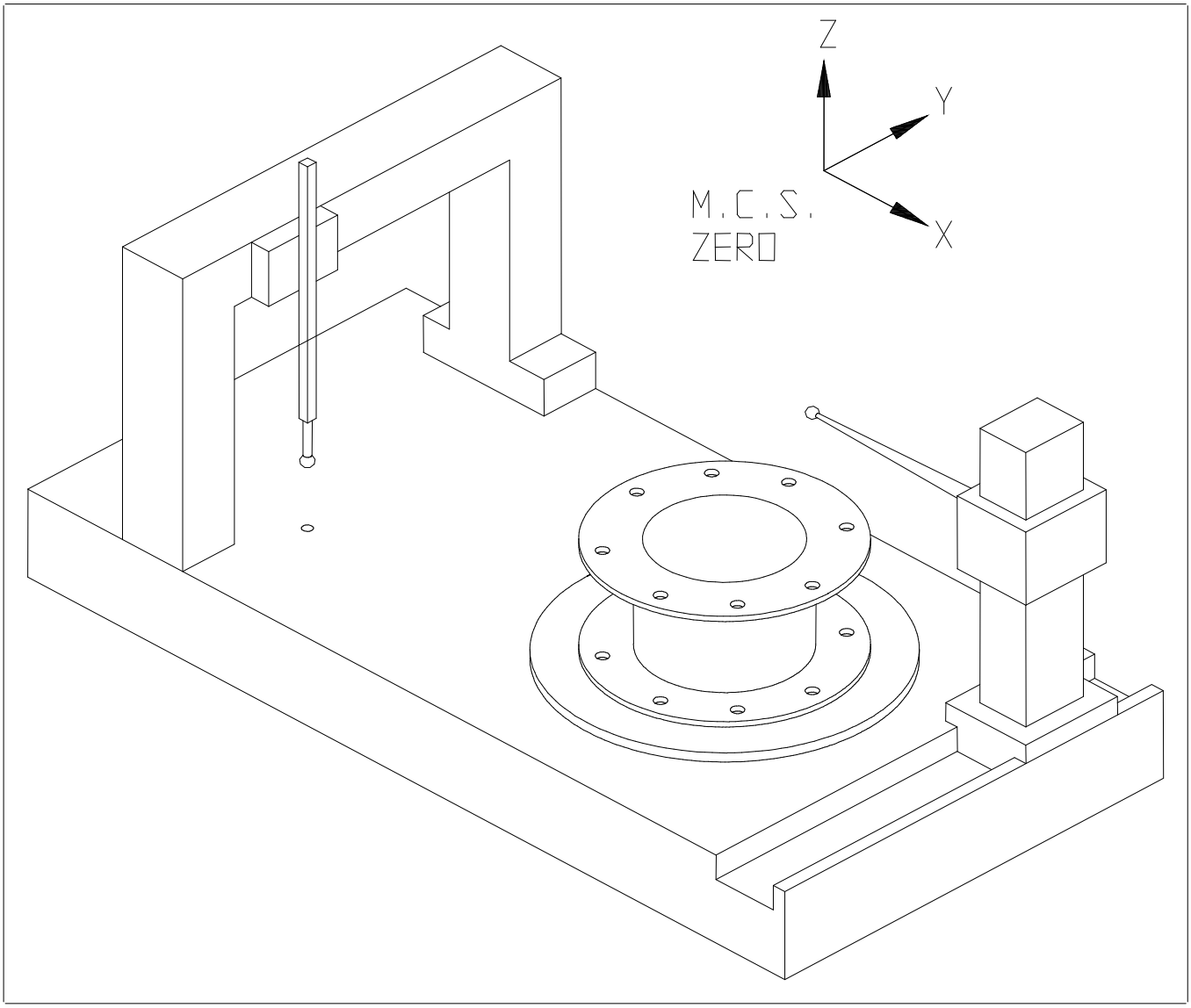


Figure B.45 — A dual system illustration

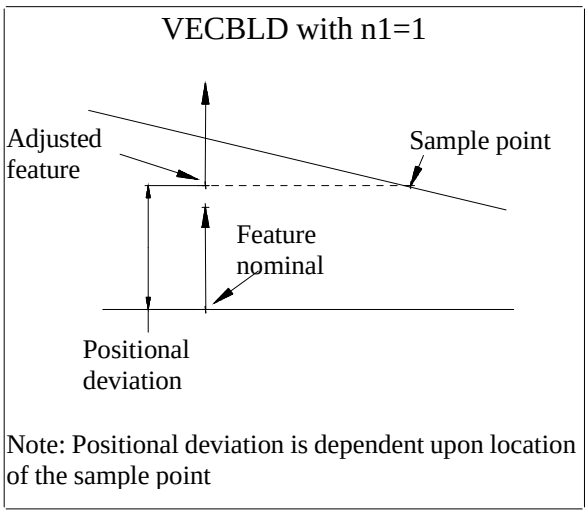


Figure B.46 — Positional deviation

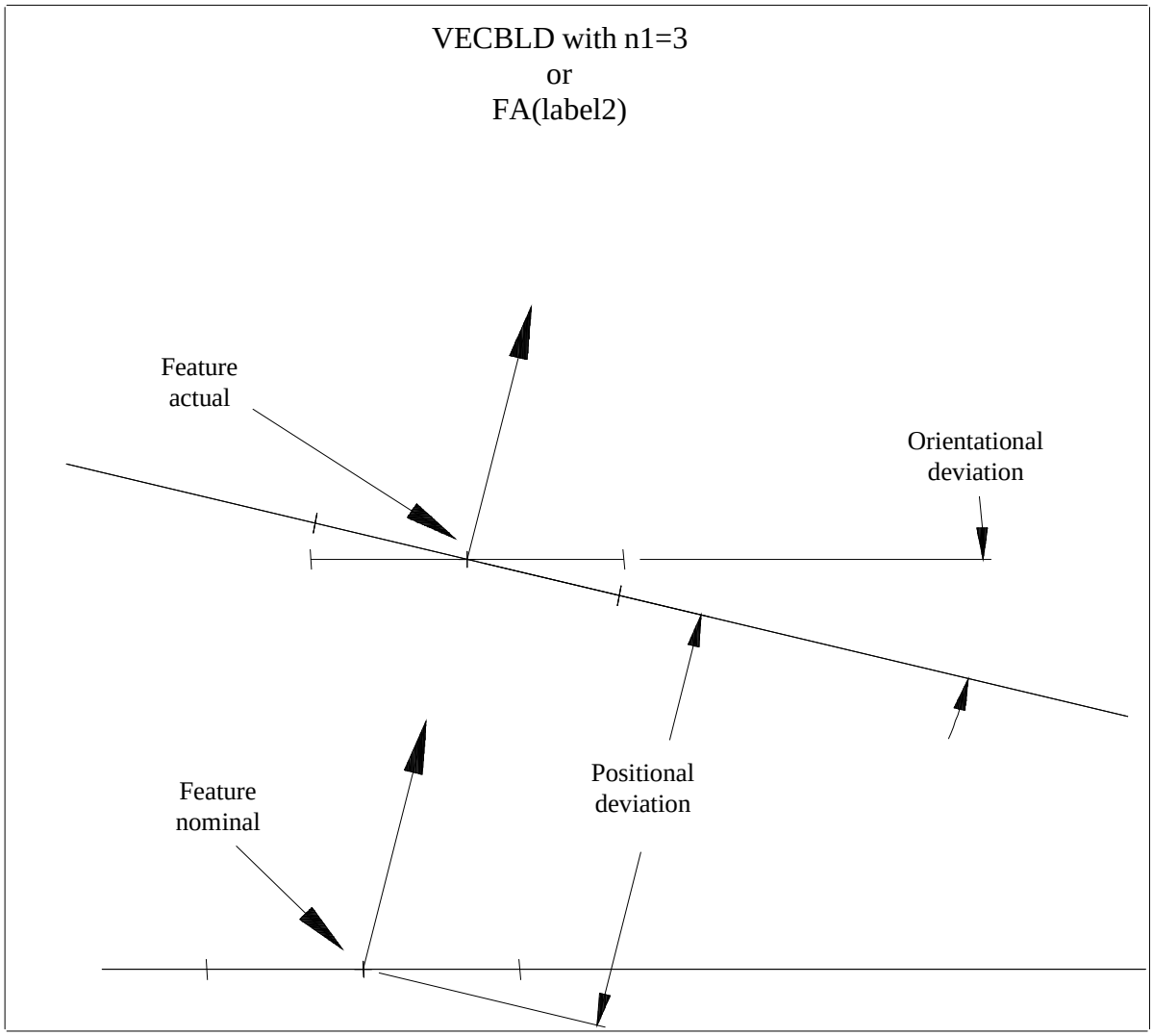
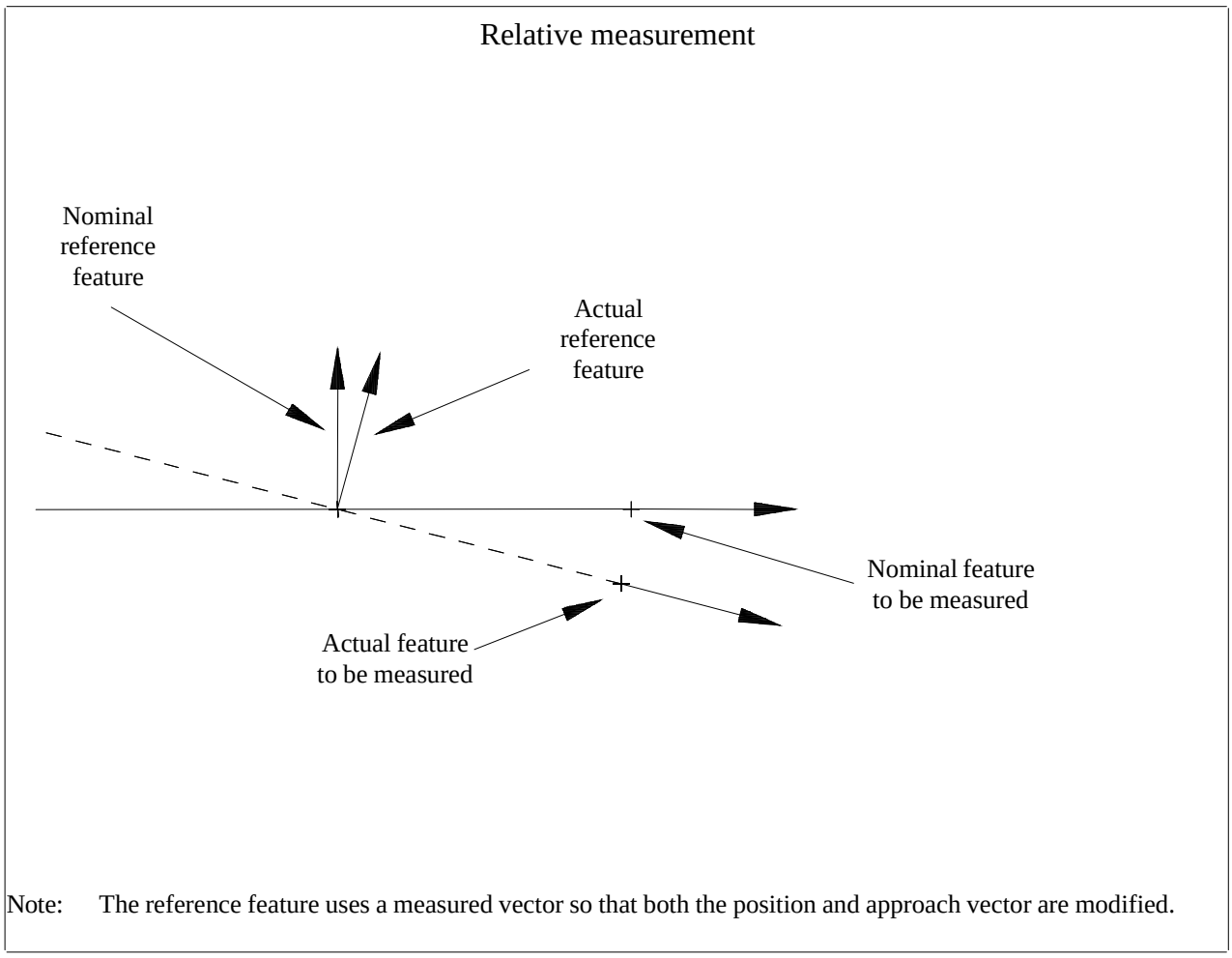


Figure B.47 — Positional and orientational deviation



Note: The reference feature uses a measured vector so that both the position and approach vector are modified.

Figure B.48 — Relative measure / position & approach

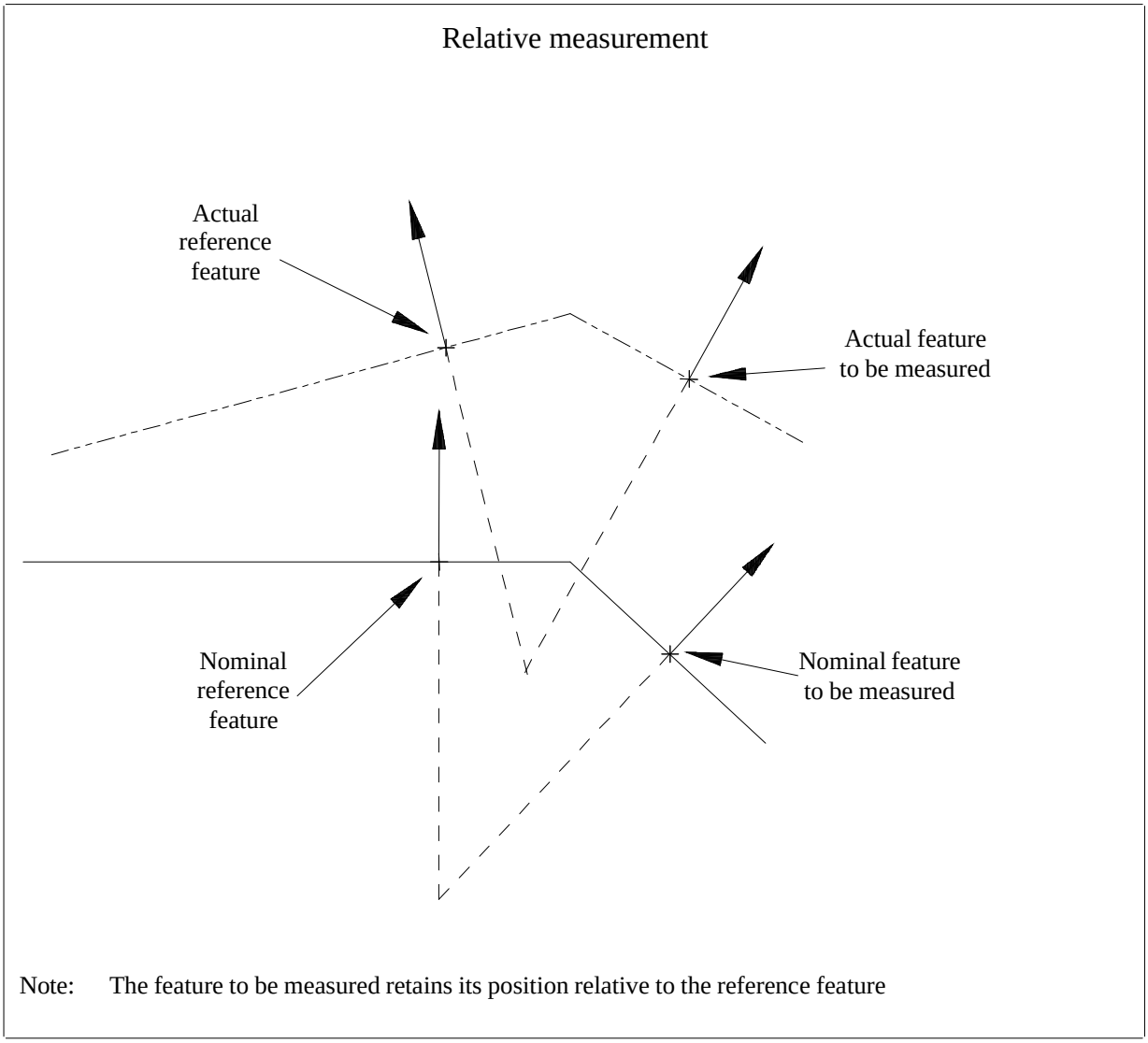
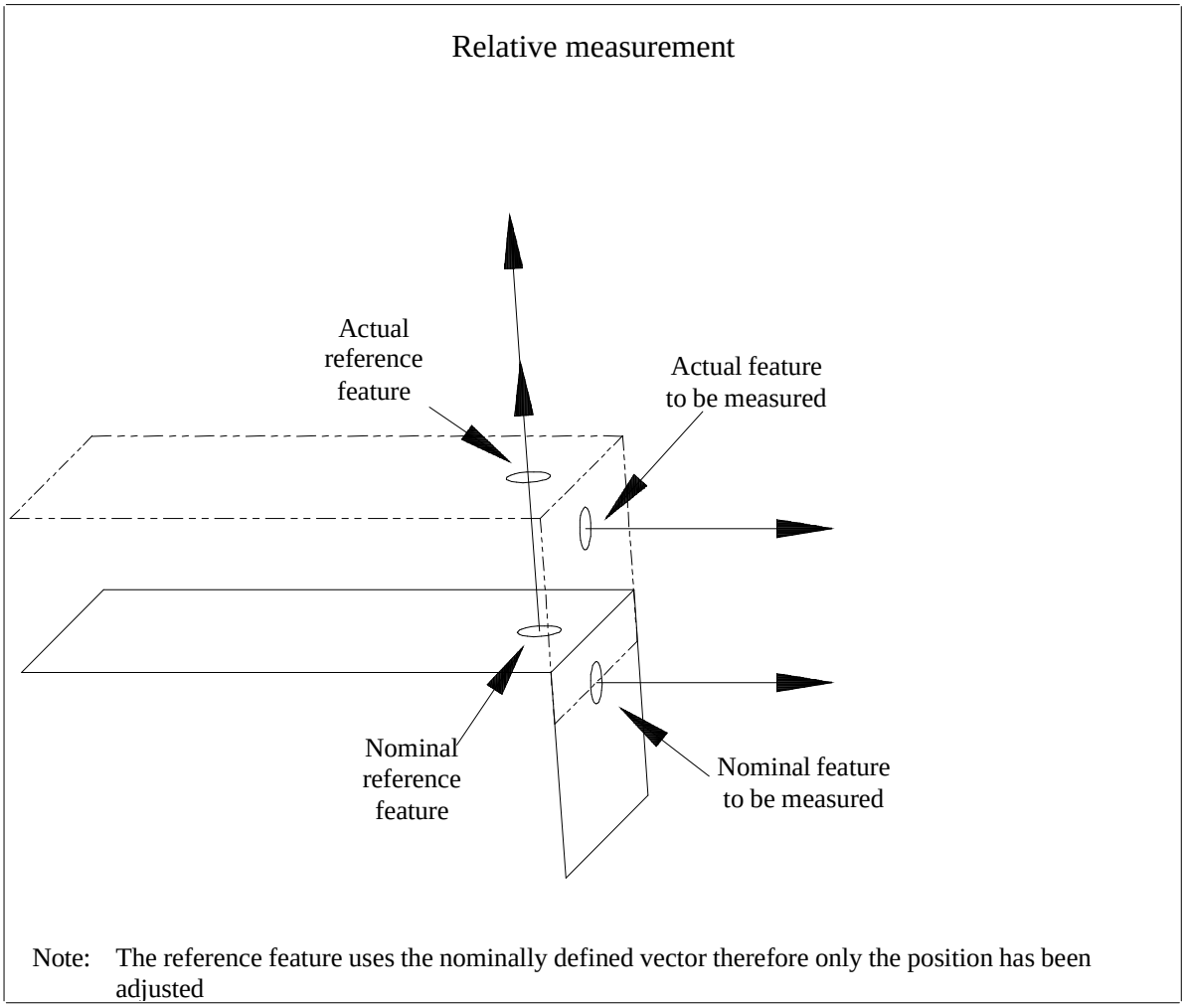
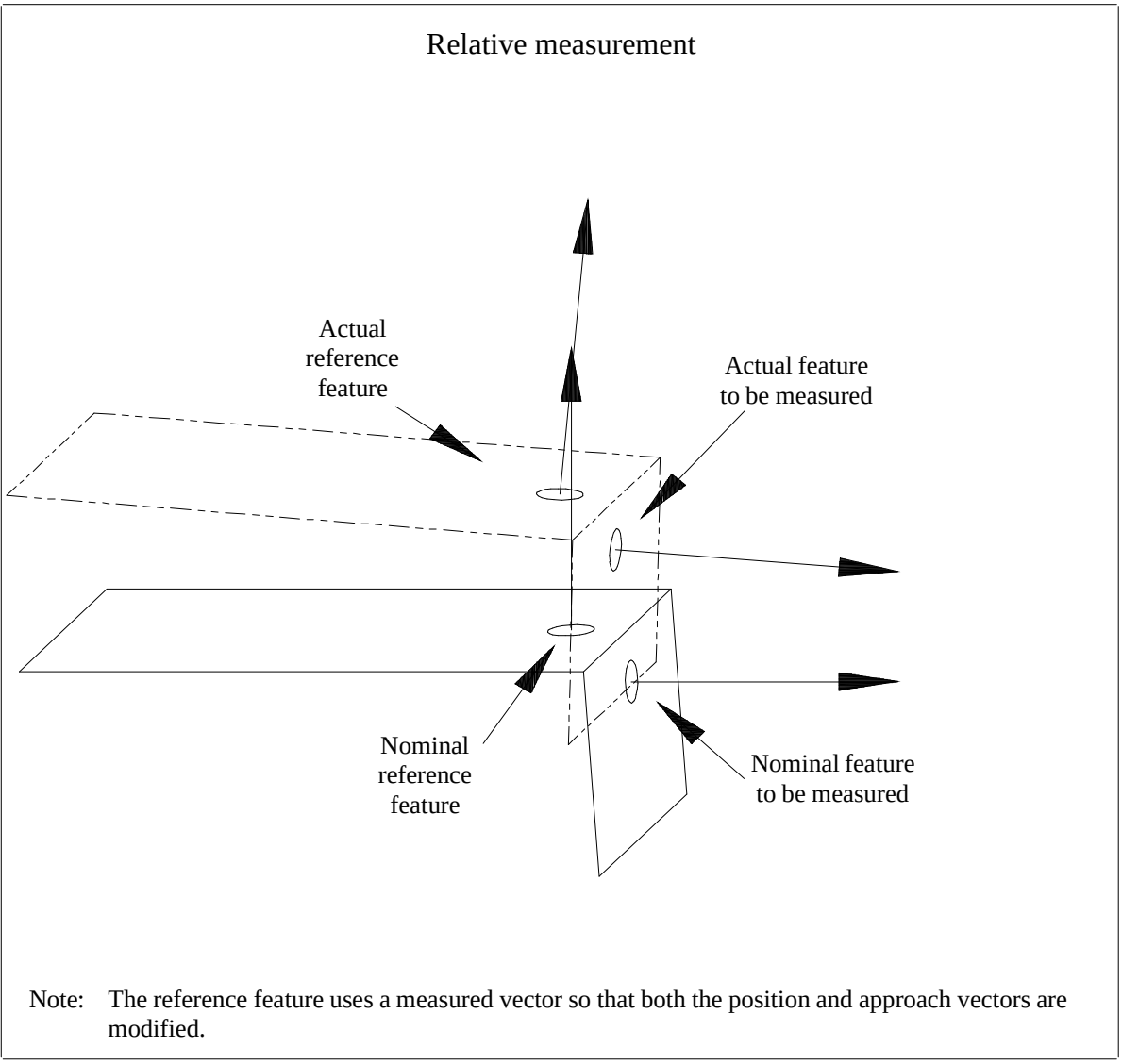


Figure B.49 — Relative measure / retaining relative position



Note: The reference feature uses the nominally defined vector therefore only the position has been adjusted

Figure B.50 — Relative measure / position



Note: The reference feature uses a measured vector so that both the position and approach vectors are modified.

Figure B.51 — Relative measure / position & approach

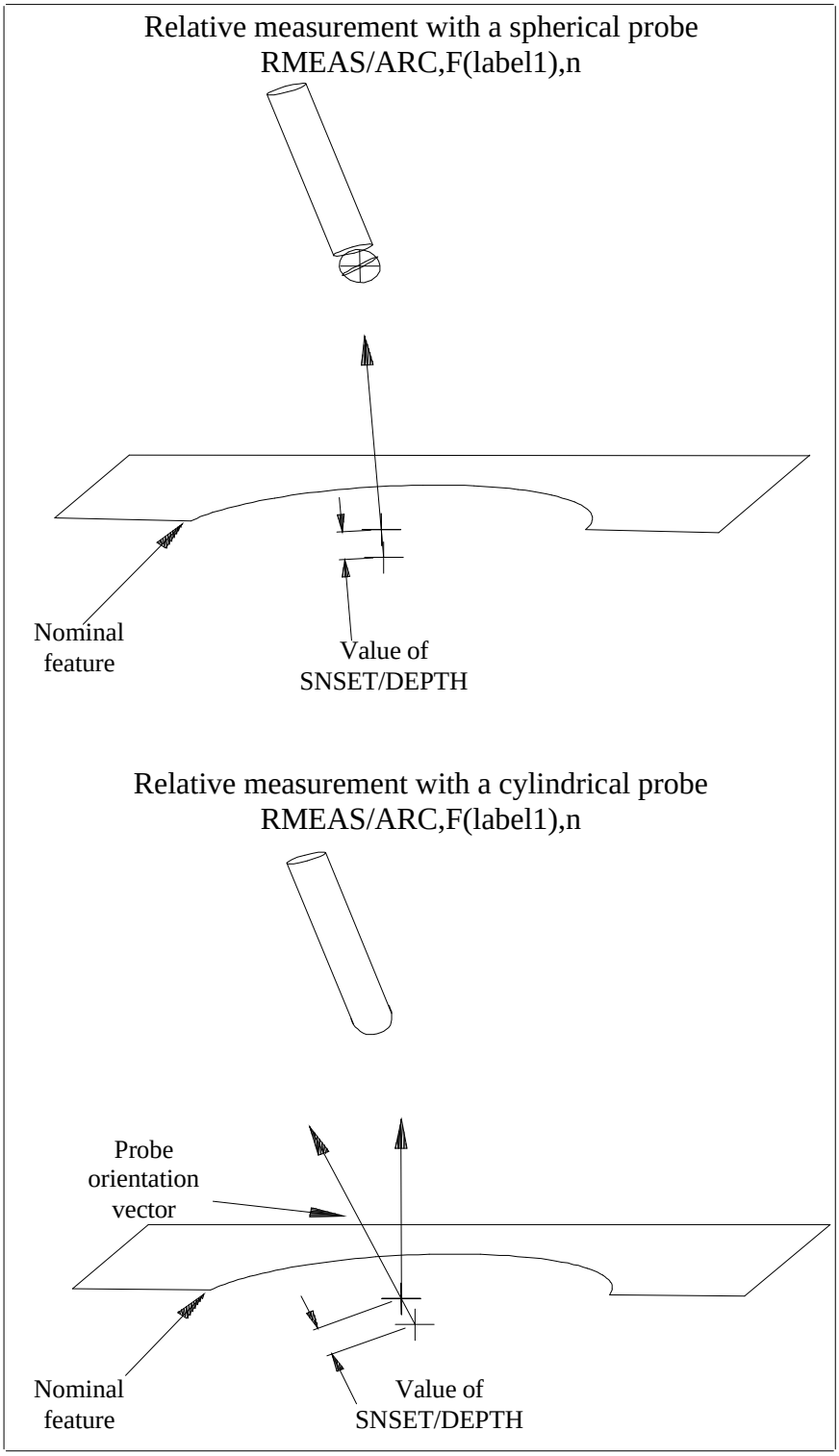
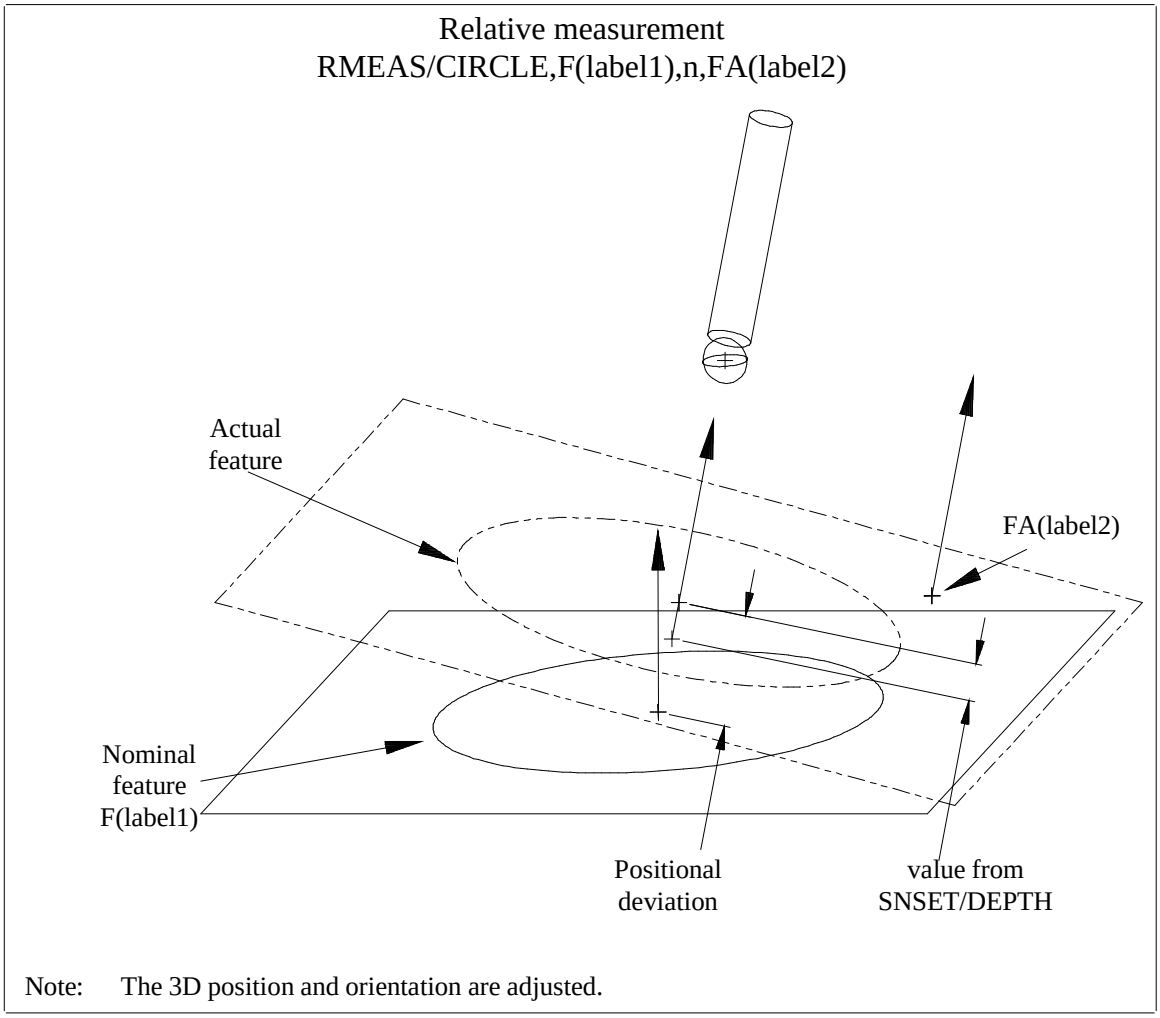


Figure B.52 — RMEAS/ARC with and without a cylindrical probe



Note: The 3D position and orientation are adjusted.

Figure B.53 — RMEAS/CIRCLE with FA(lname2)

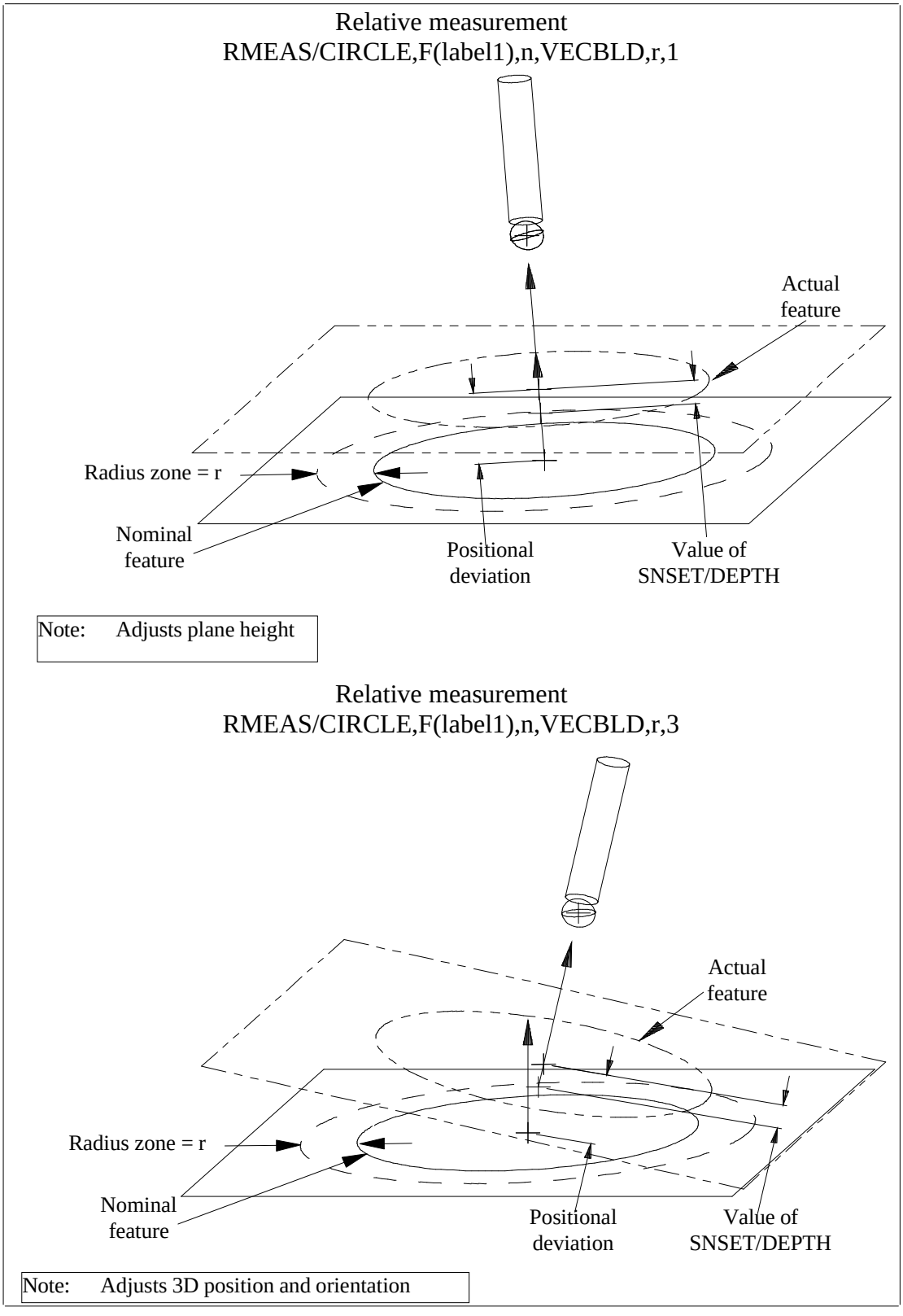


Figure B.54 — RMEAS/CIRCLE with VECBLD,r,1 and with VECBLD,r,3

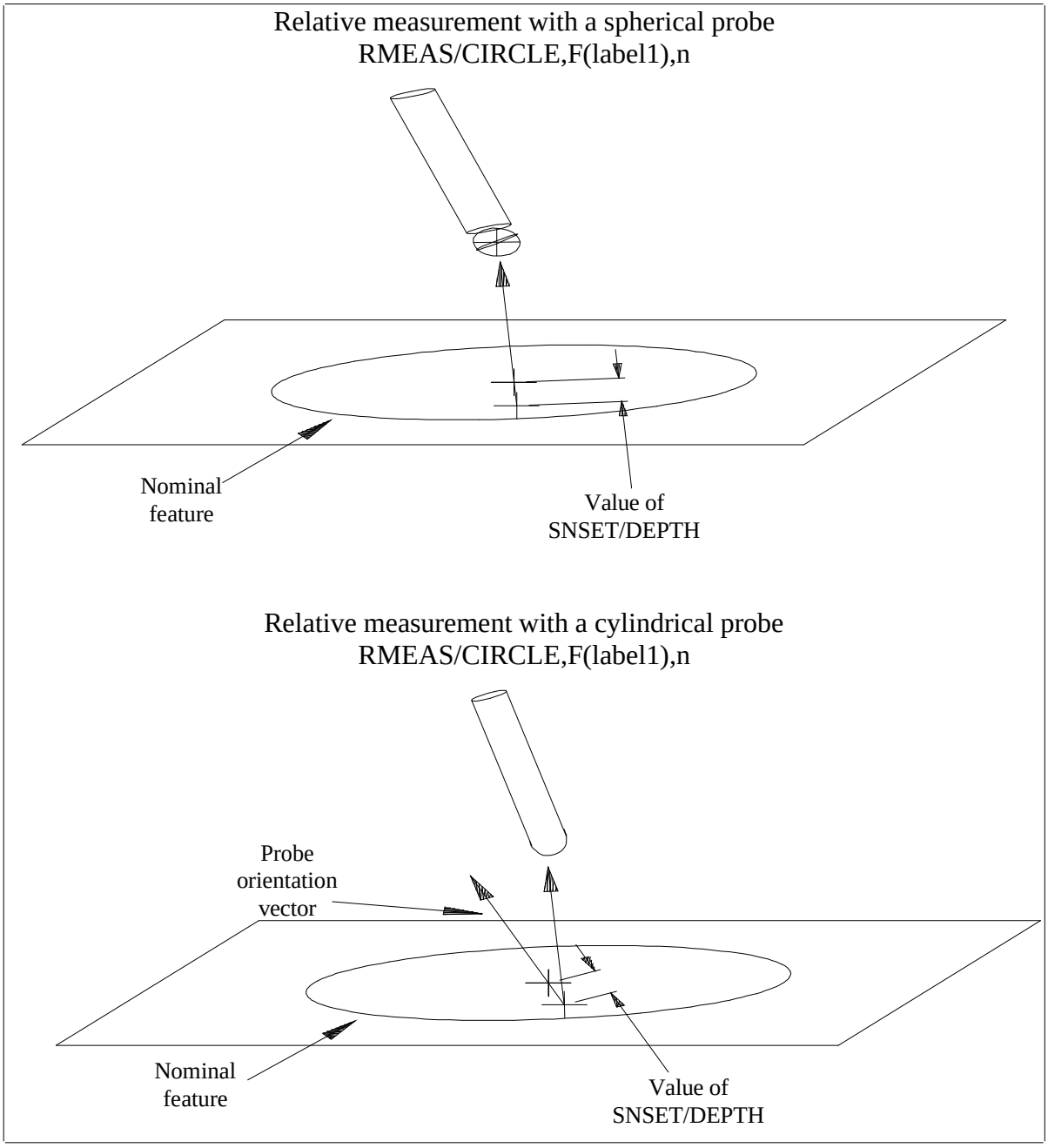


Figure B.55 — RMEAS/CIRCLE with and without a cylindrical probe

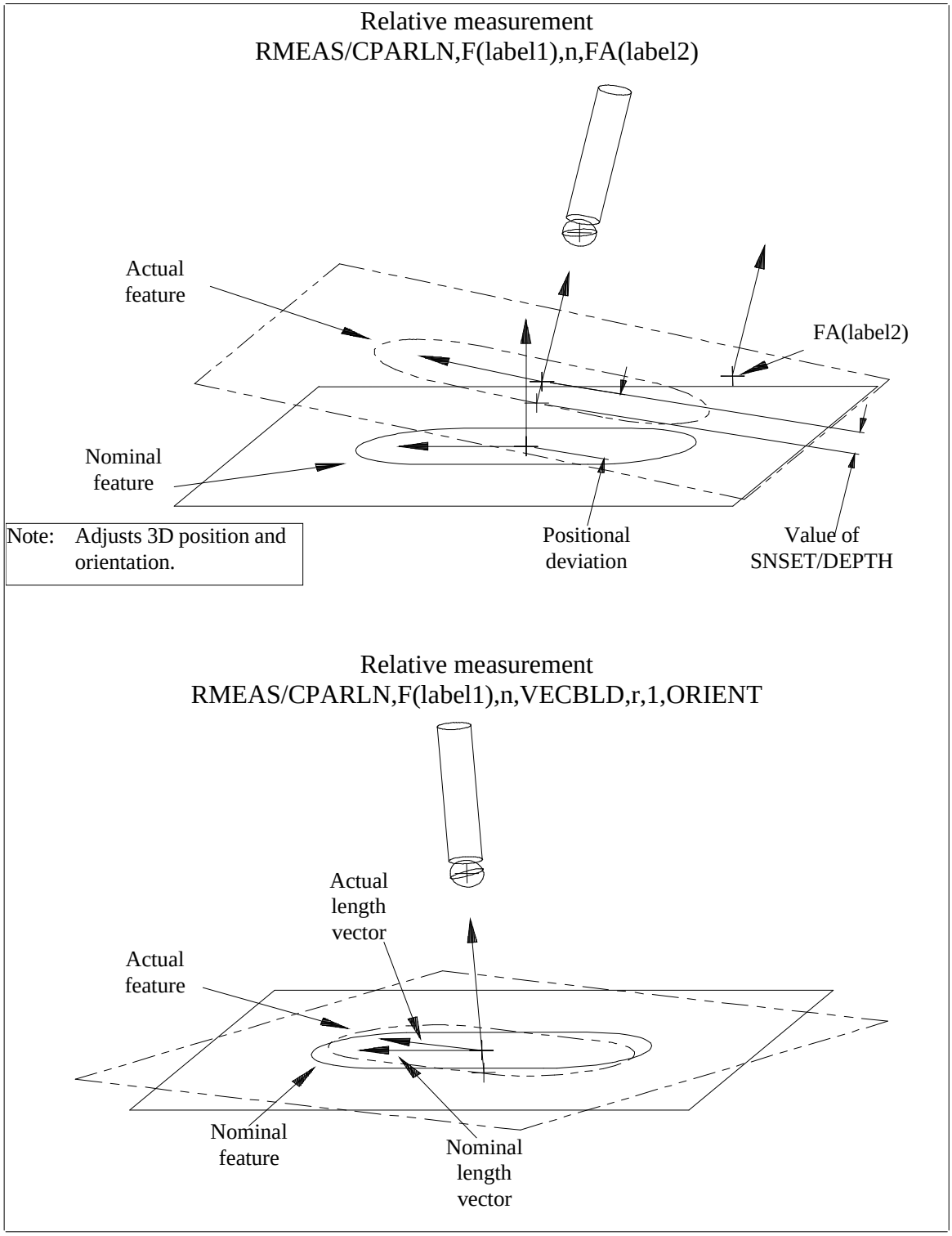


Figure B.56 — RMEAS/CPARLN with FA(lname2) and with the ORIENT parameter

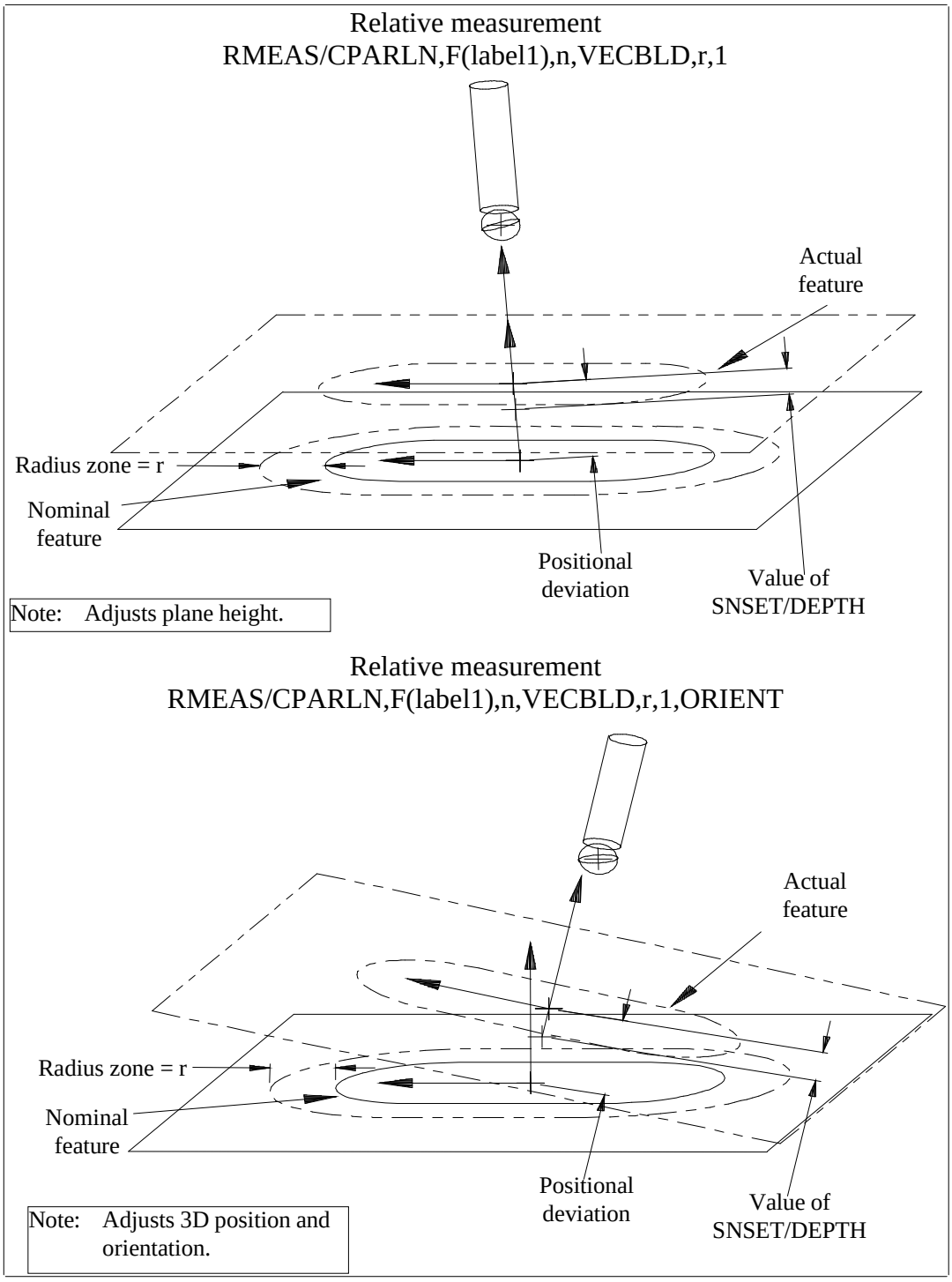


Figure B.57 — RMEAS/CPARLN with VECBLD,r,1 and with VECBLD,r,3

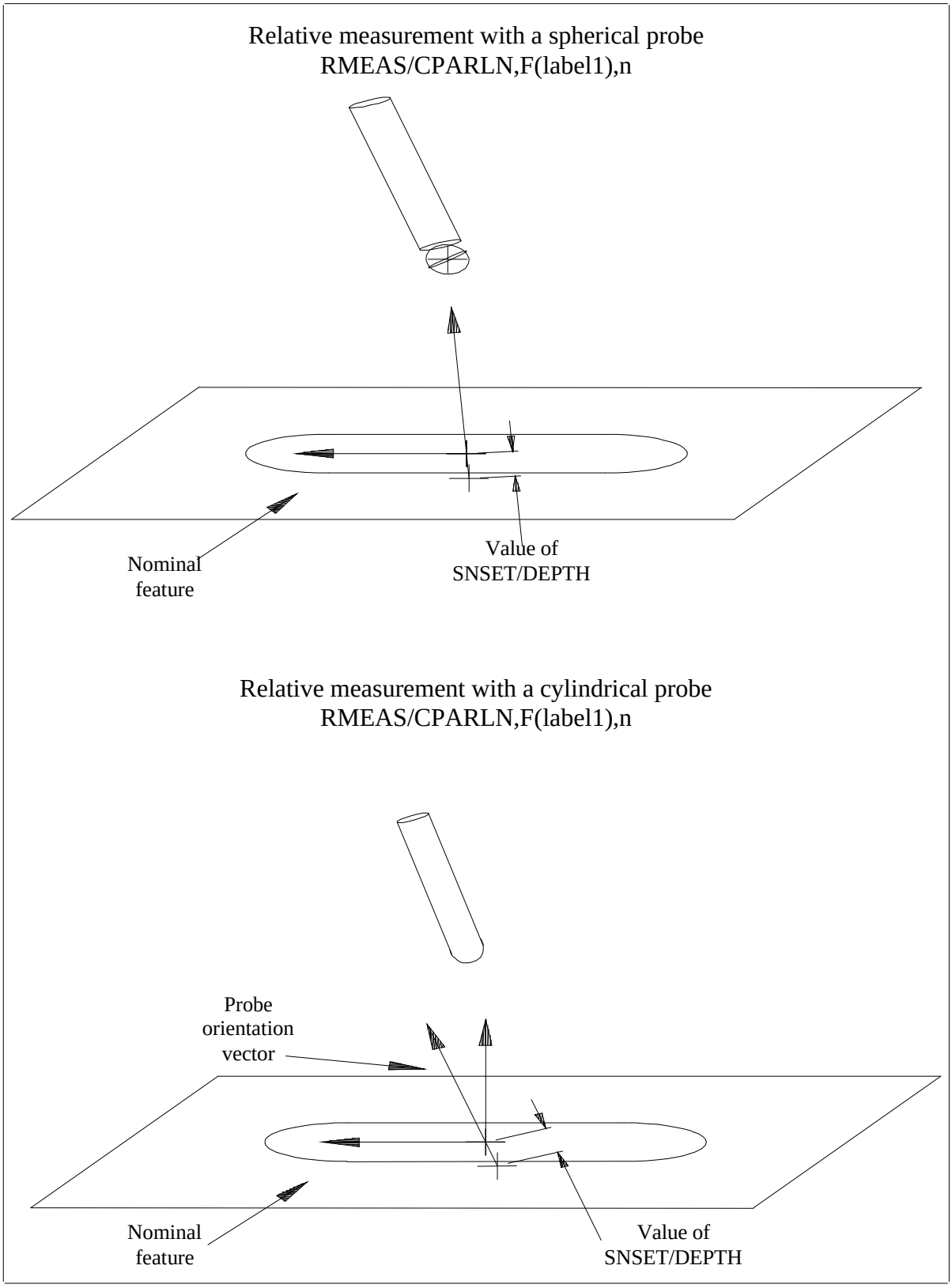


Figure B.58 — RMEAS/CPARLN with and without a cylindrical probe

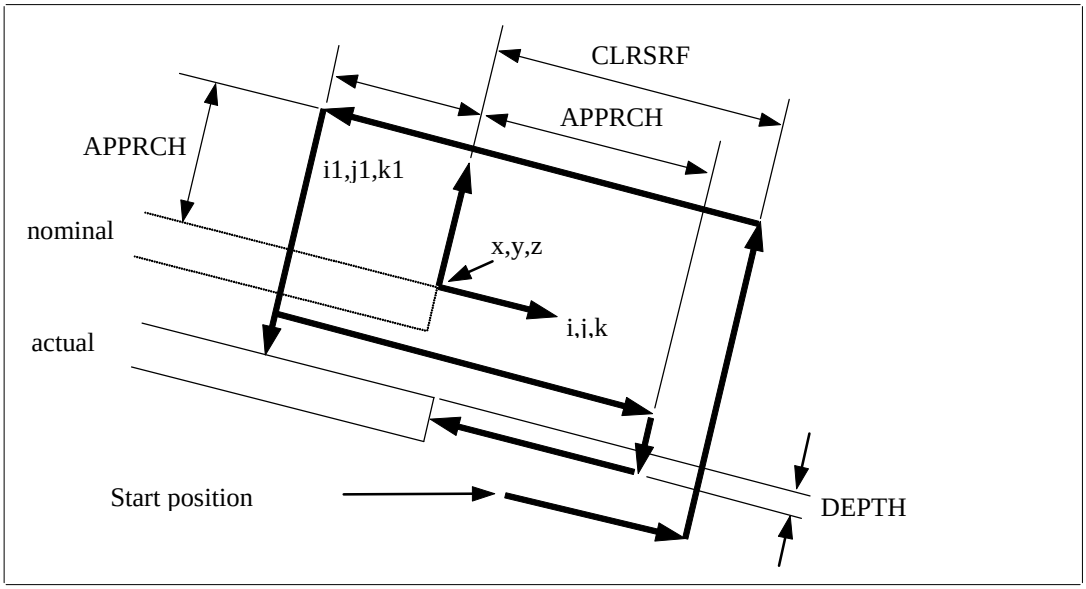


Figure B.59 — RMEAS/EDGEPT,F(edge),1,VECBLD,0,1,5

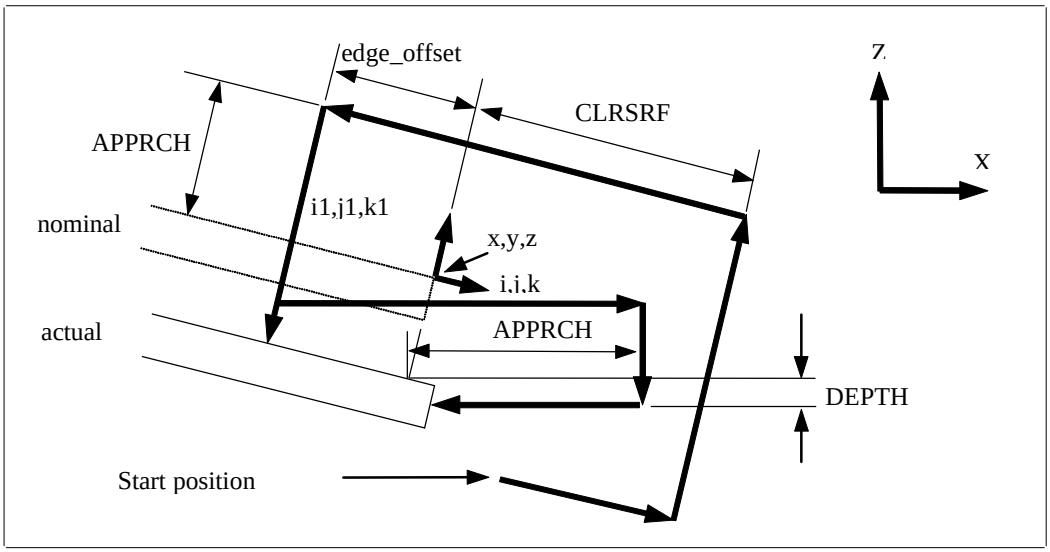


Figure B.60 — RMEAS/EDGEPT,F(edge),1,VECBLD,0,1,5,-XAXIS

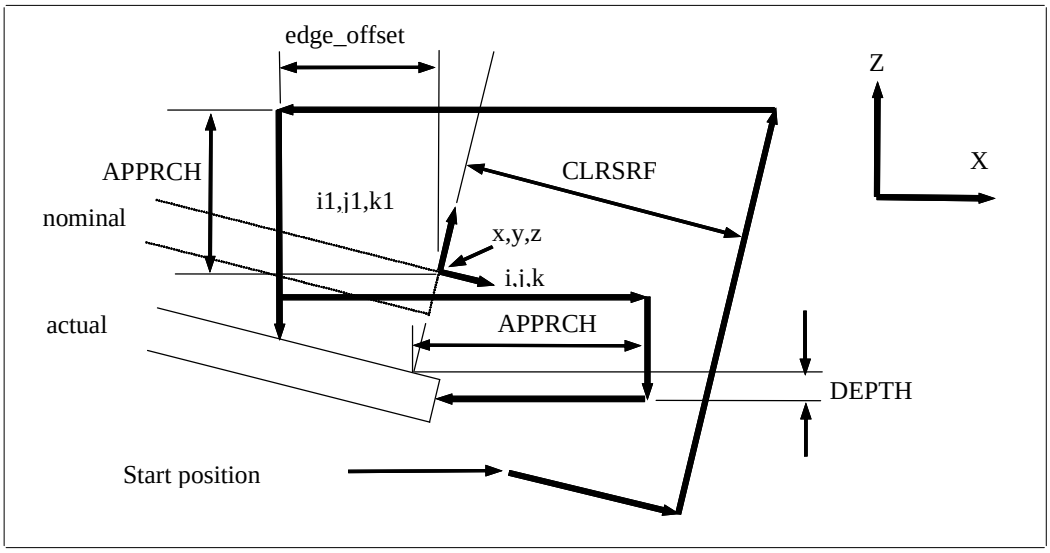


Figure B.61 — RMEAS/EDGEPT,F(edge),1,VECBLD,0,1,5,-XAXIS,ZDIR

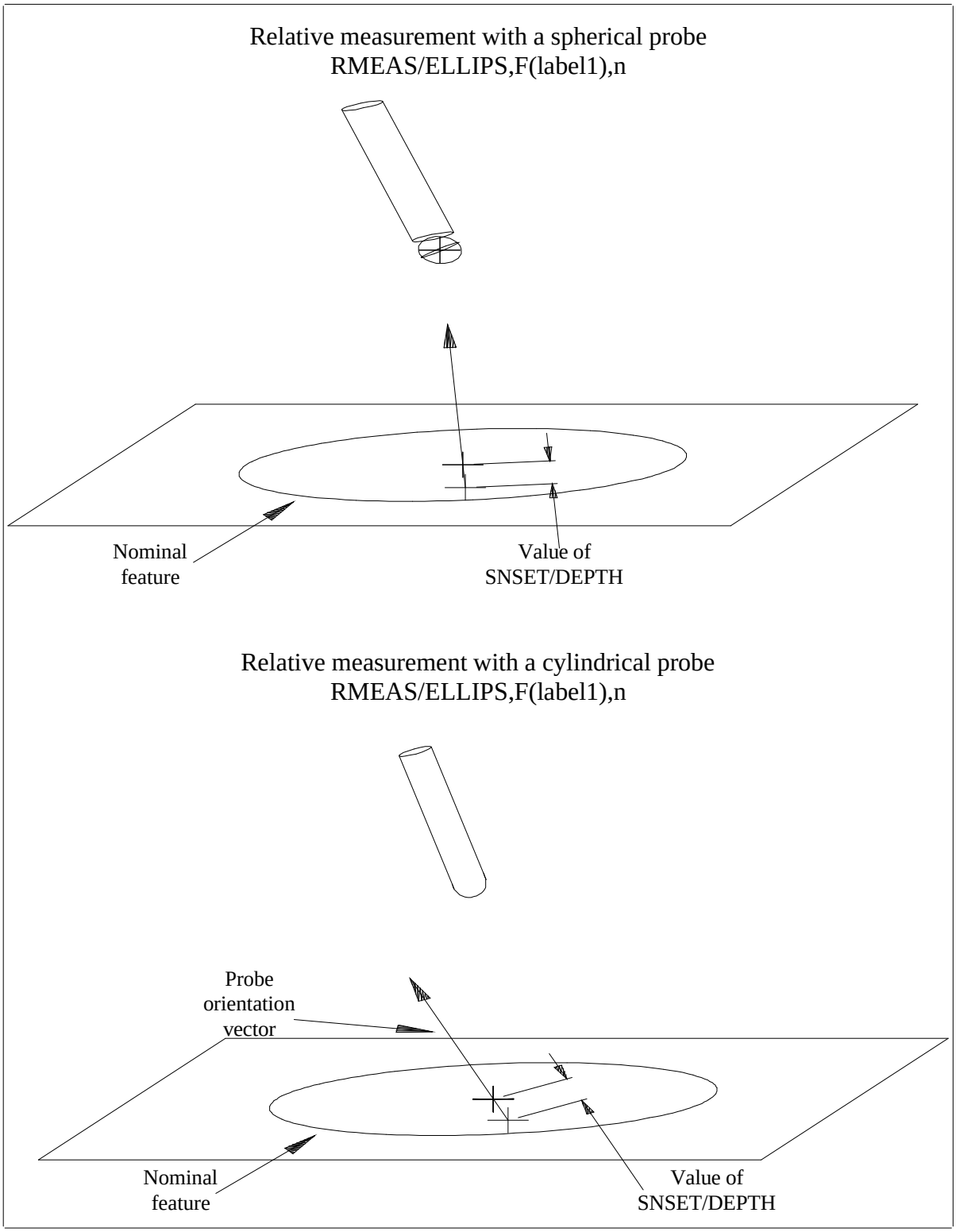


Figure B.62 — RMEAS/ELLIPS with and without a cylindrical probe

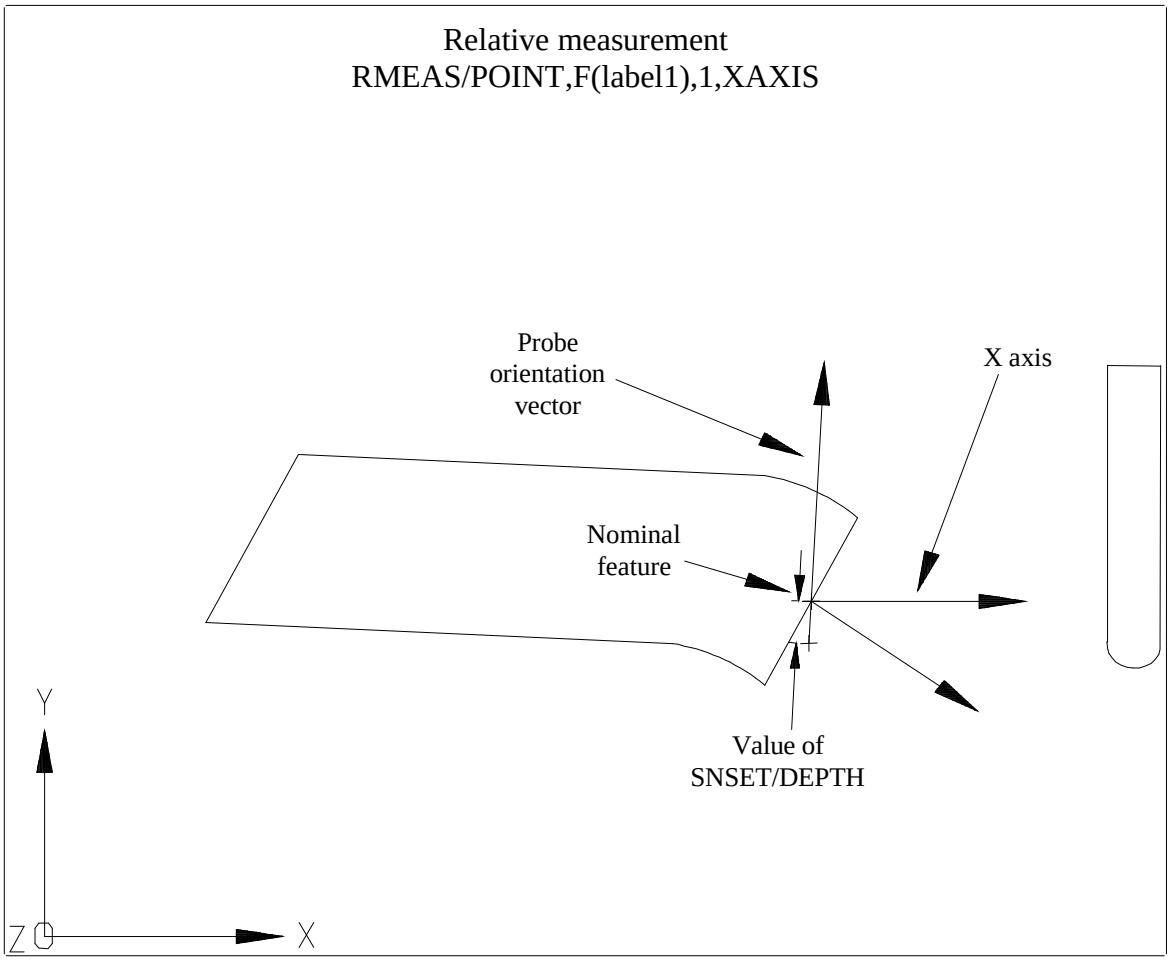


Figure B.63 — RMEAS/POINT with a specific axis using an edge point

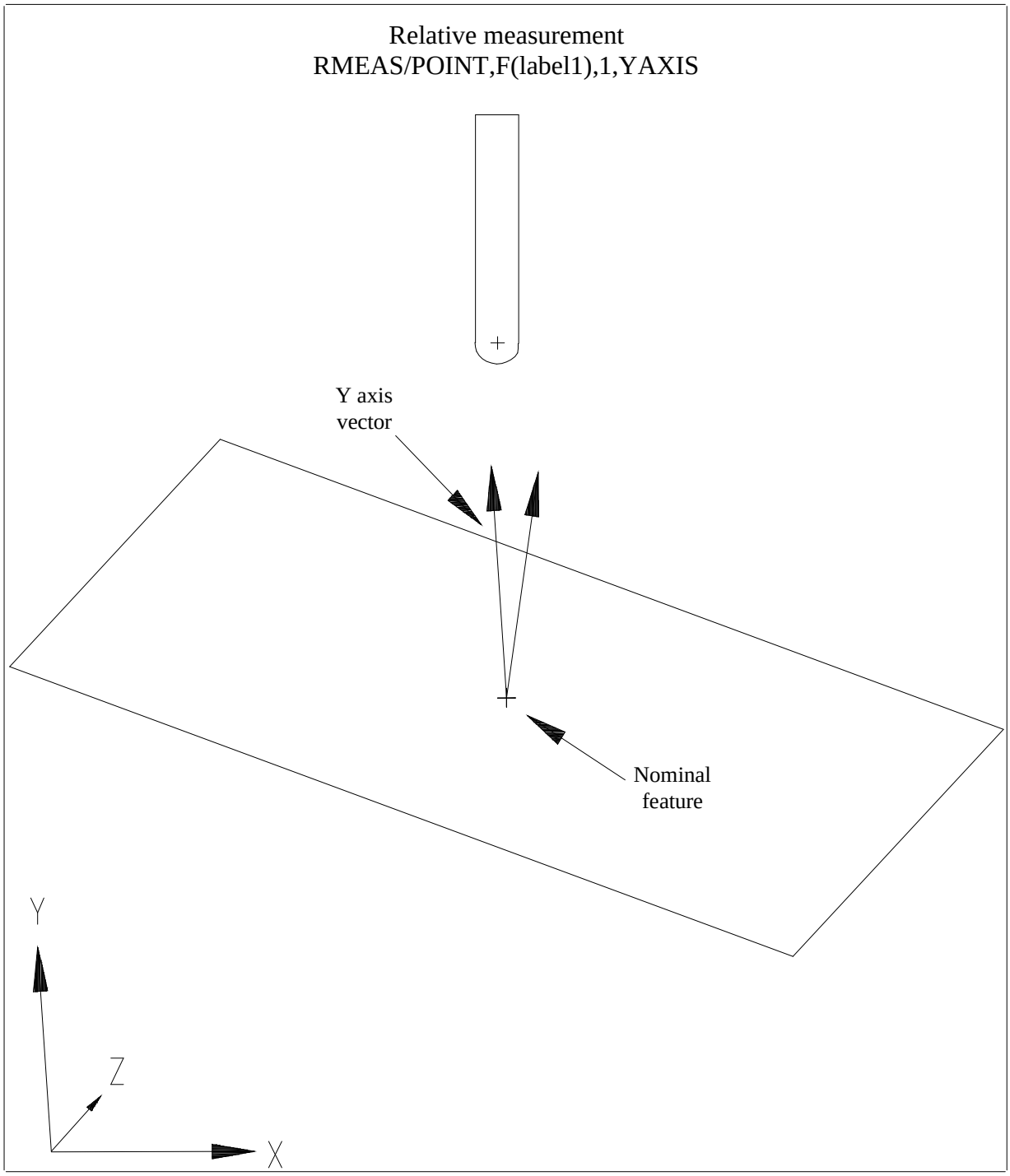


Figure B.64 — RMEAS/POINT with a specific axis using a surface point

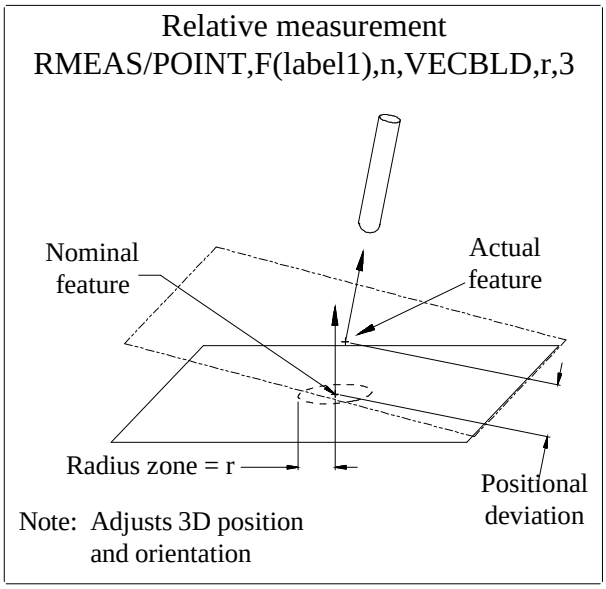


Figure B.65 — RMEAS/POINT with VECBLD,r,3 using a surface point

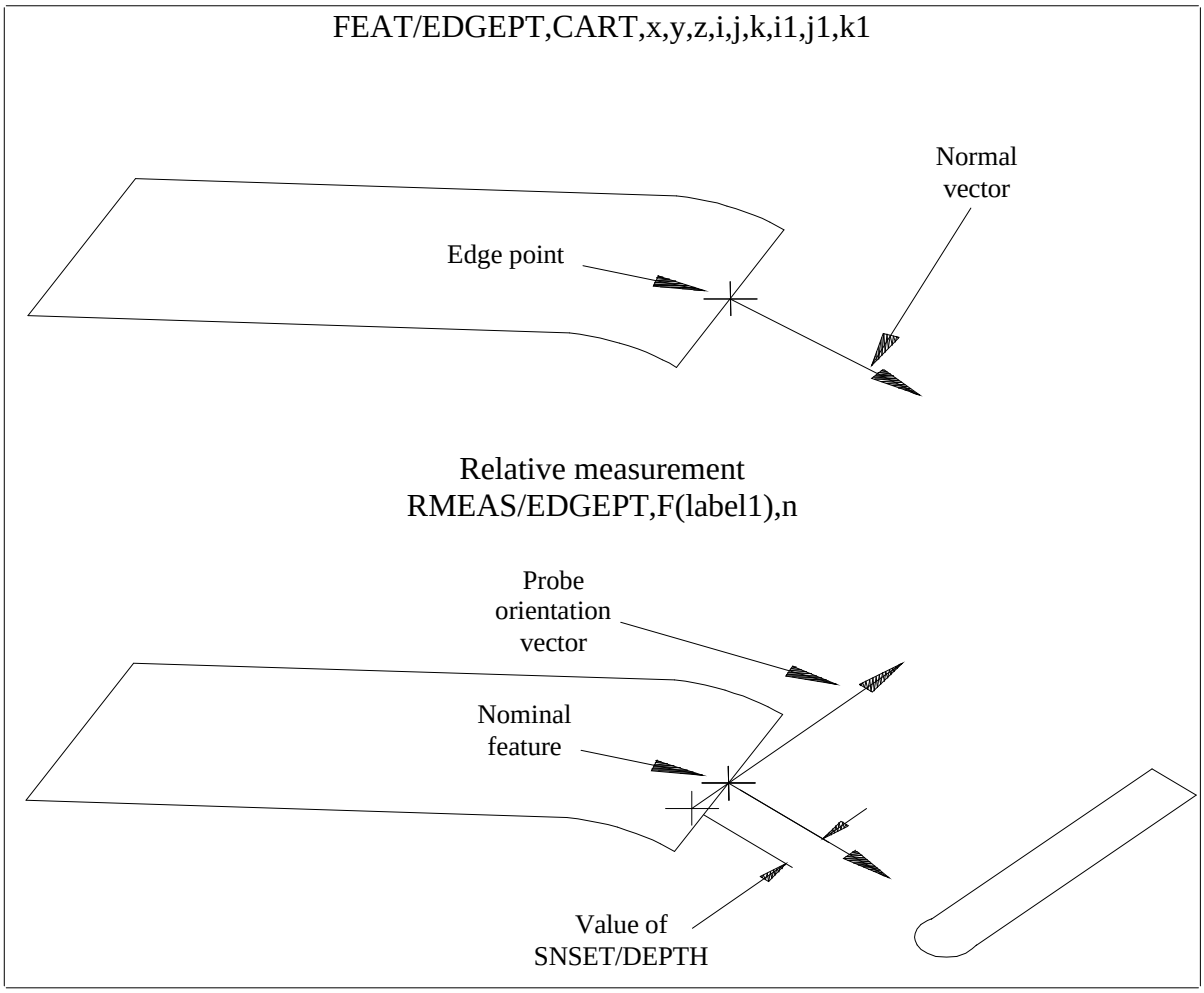


Figure B.66 — Edge point feature and RMEAS/POINT using a cylindrical probe

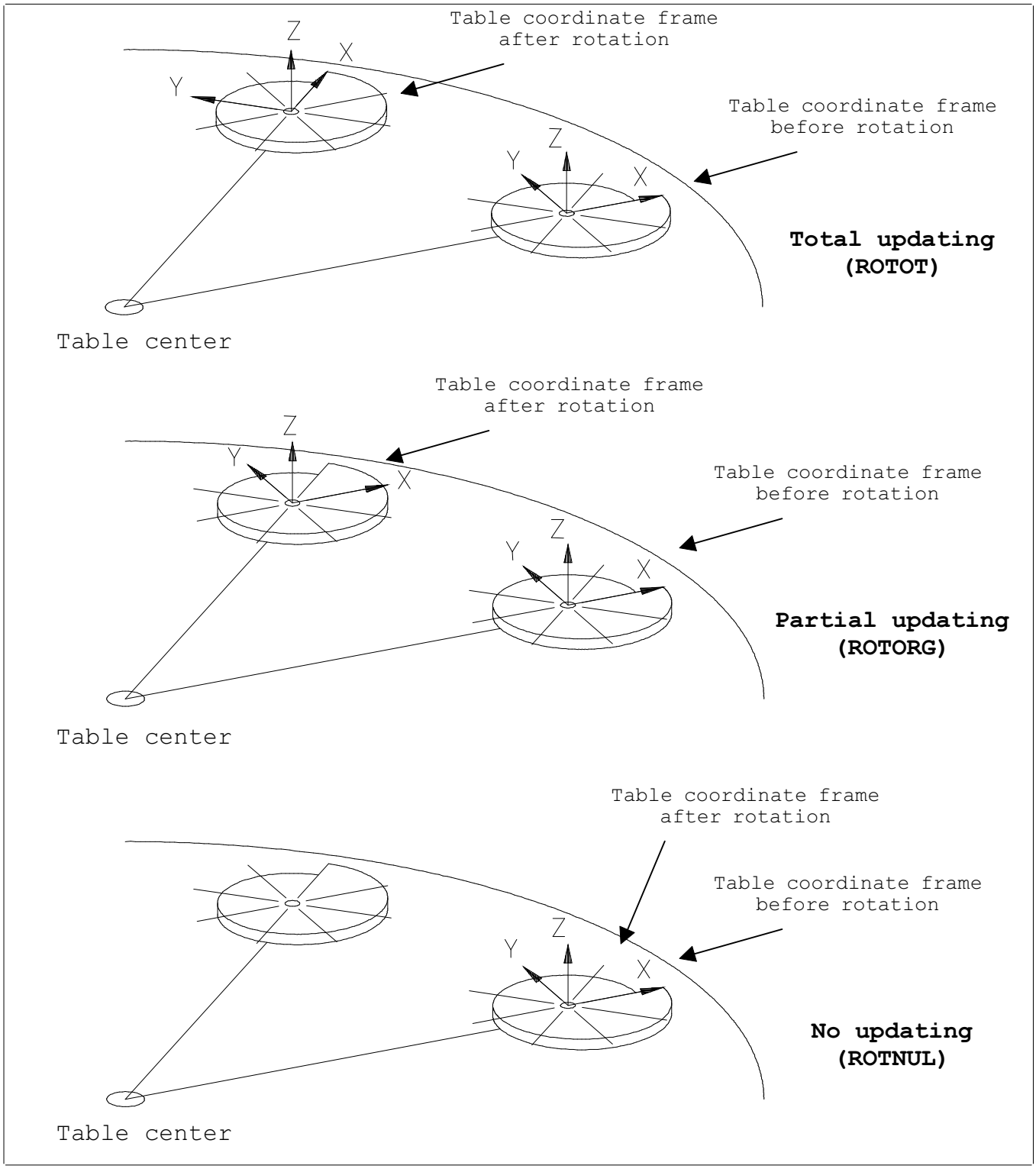


Figure B.67 — Part coordinate system updating following a rotation move

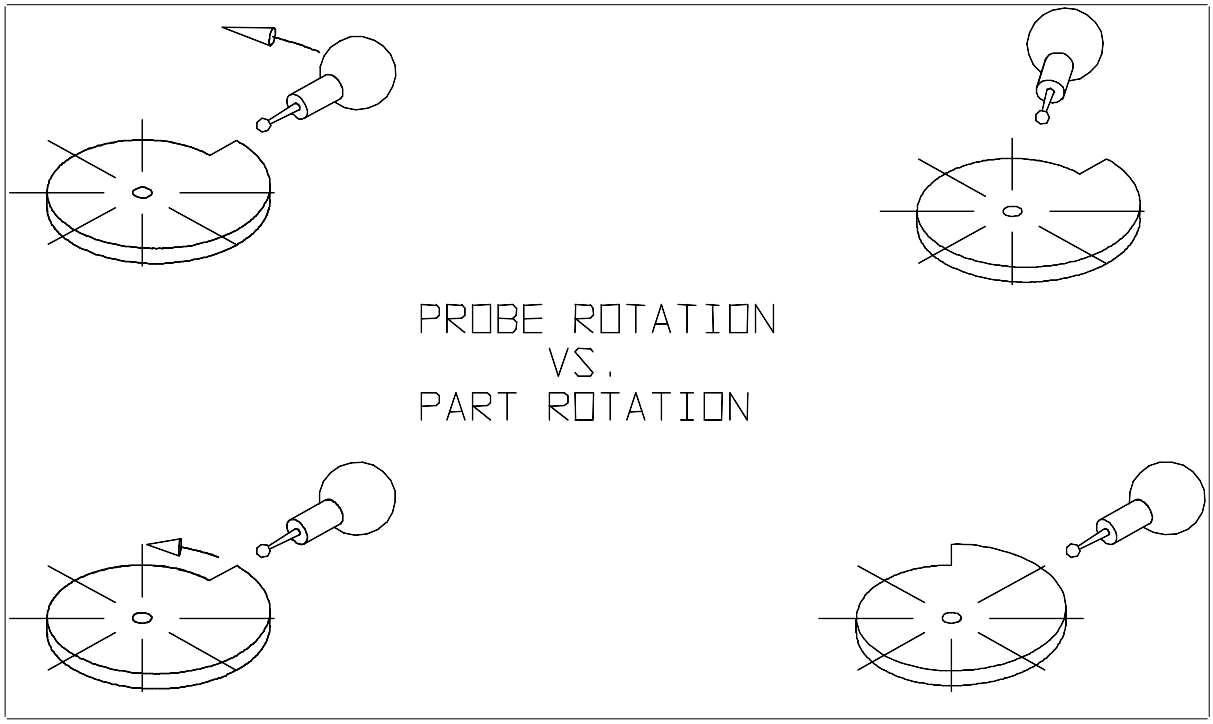


Figure B.68 — Part probe orientation

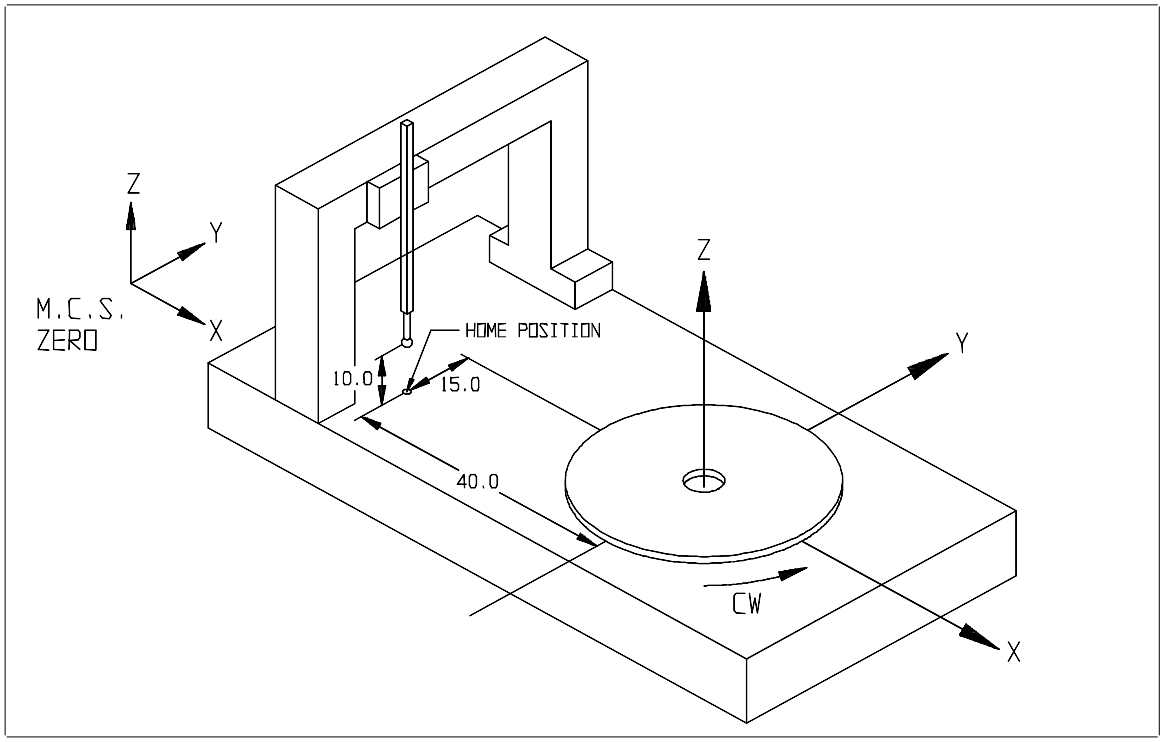


Figure B.69 — Rotary table positioning with respect to the machine coordinate system (M.C.S.) zero point

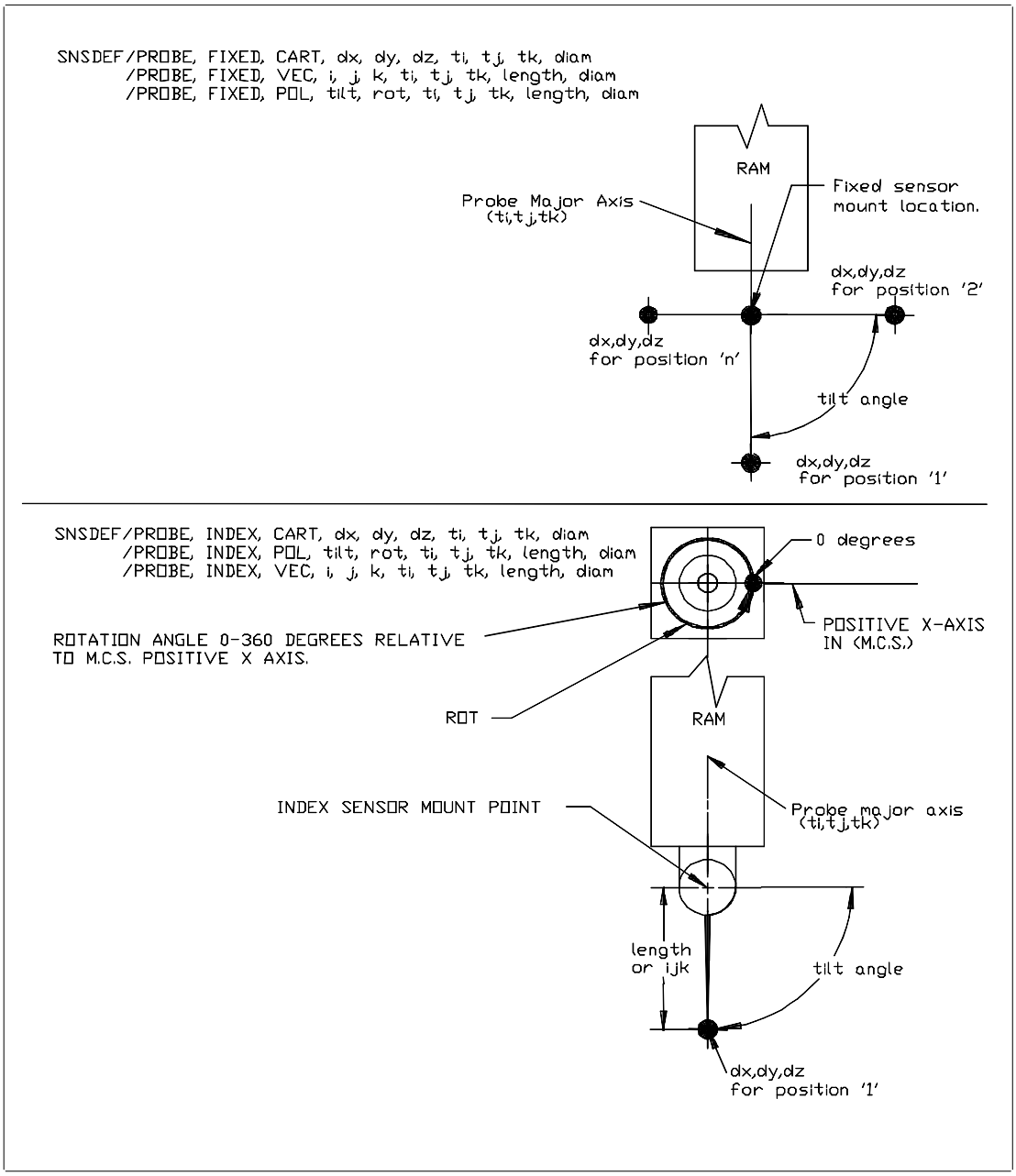


Figure B.70 — Probe sensor illustration

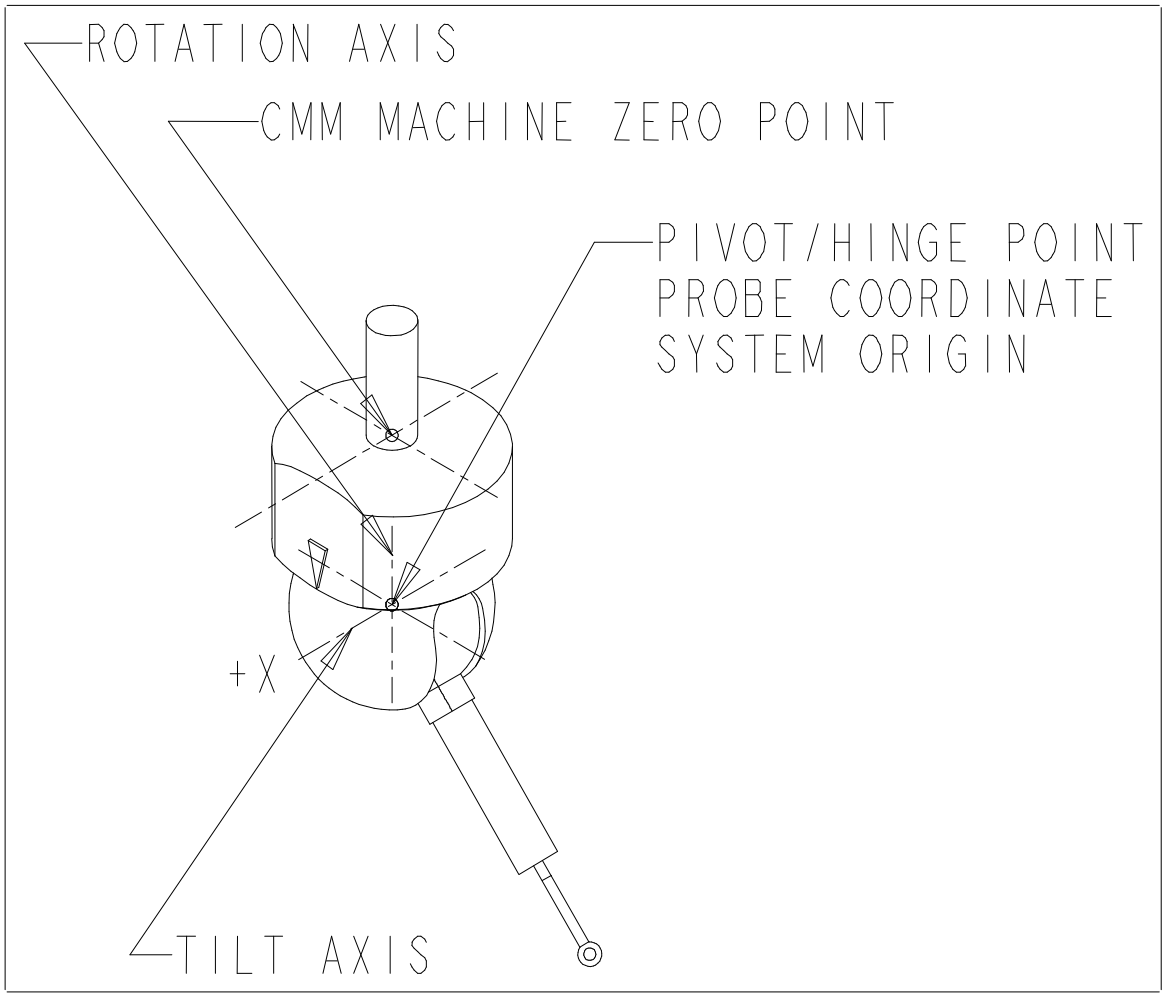


Figure B.71 — Indexable head definitions

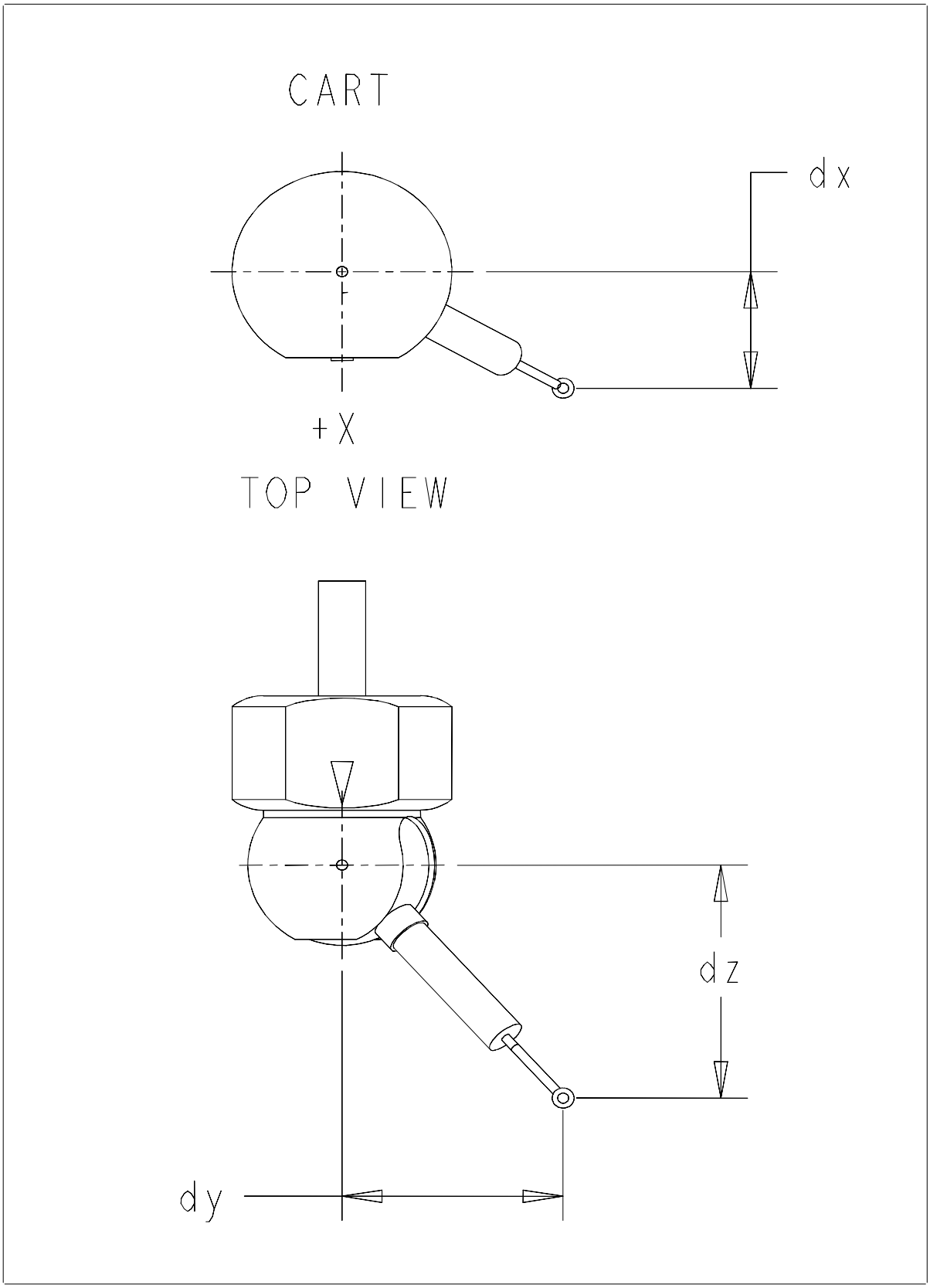


Figure B.72 — Indexable head Cartesian coordinates

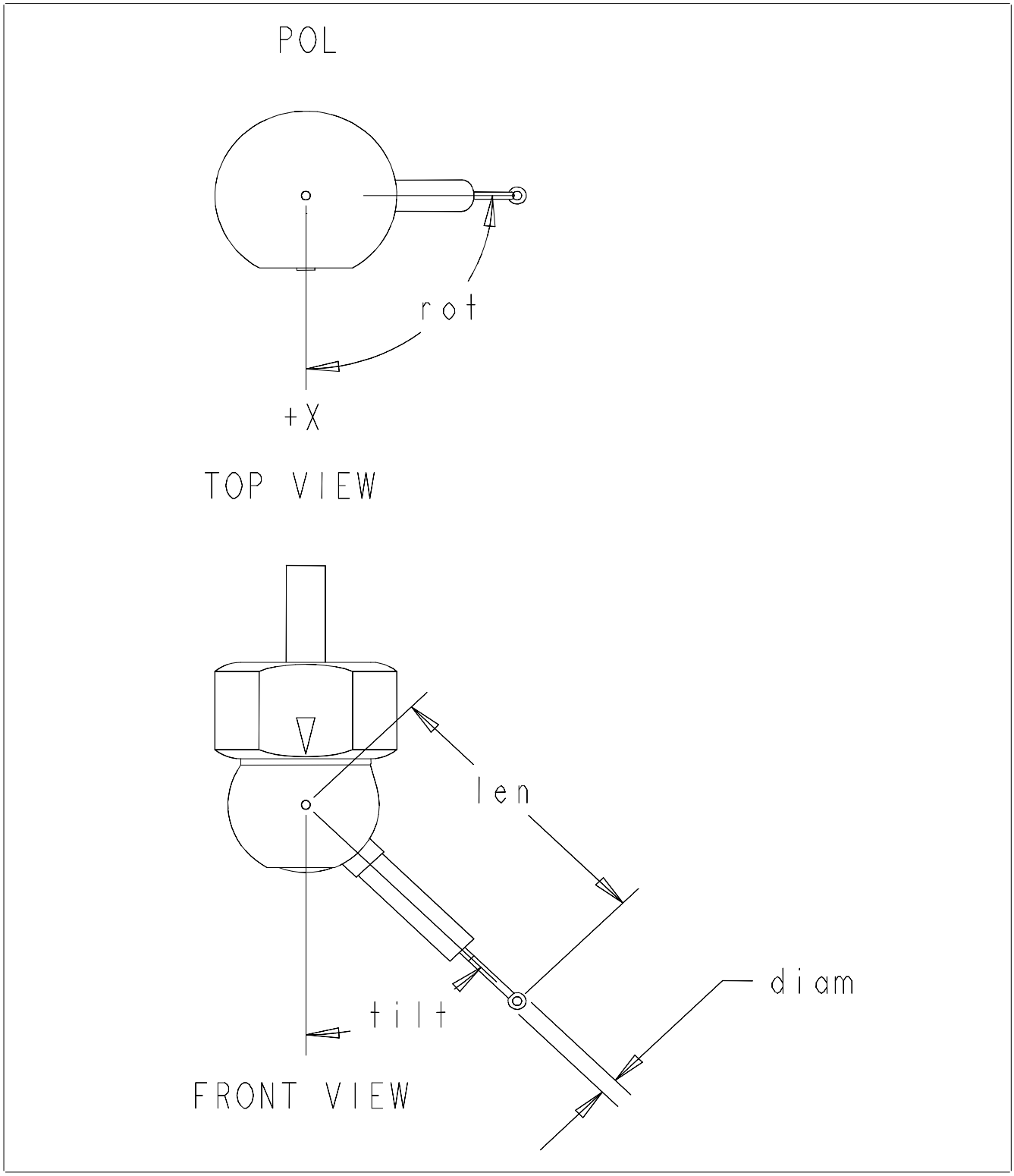


Figure B.73 — Indexable head polar coordinates

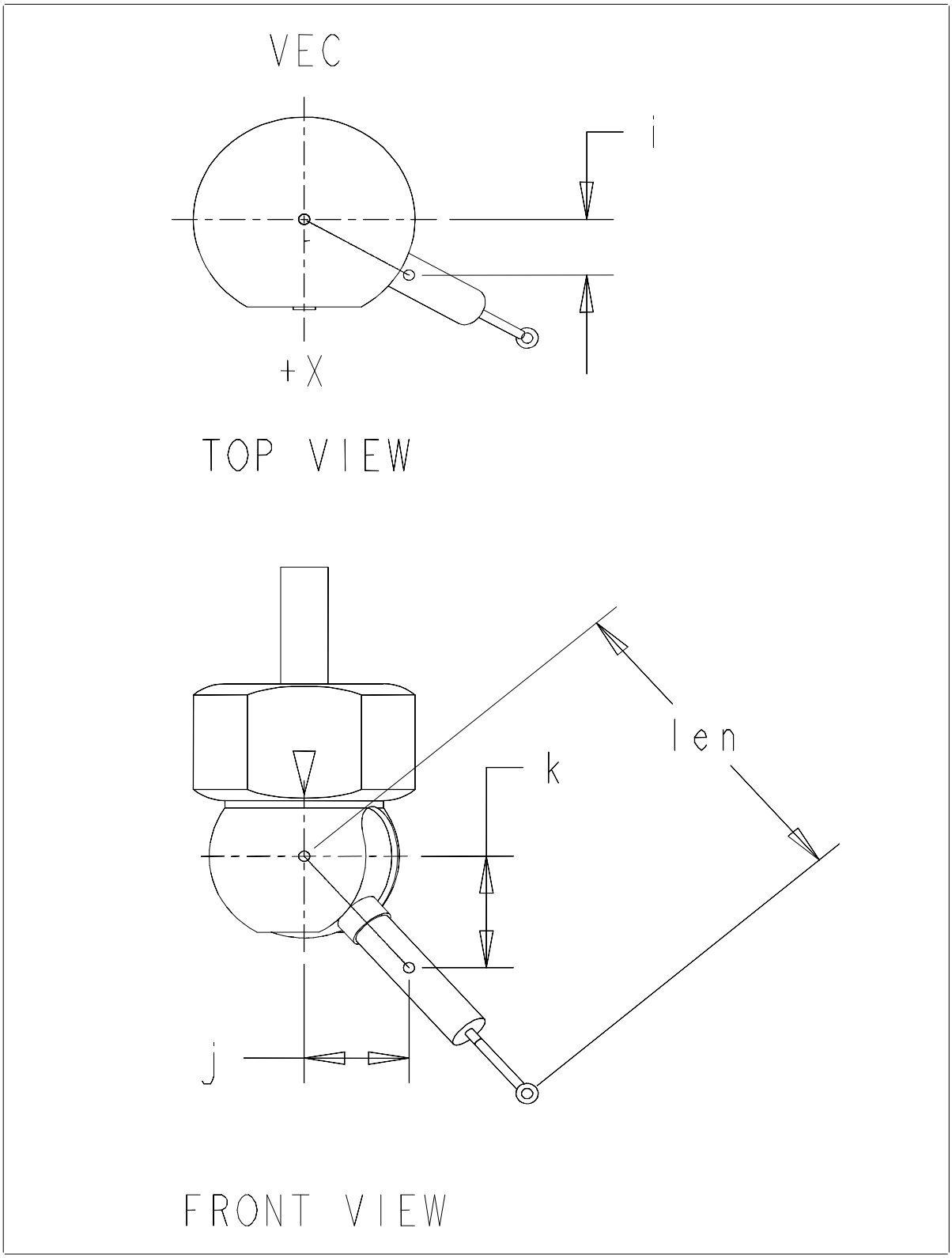


Figure B.74 — Indexable head vector orientation

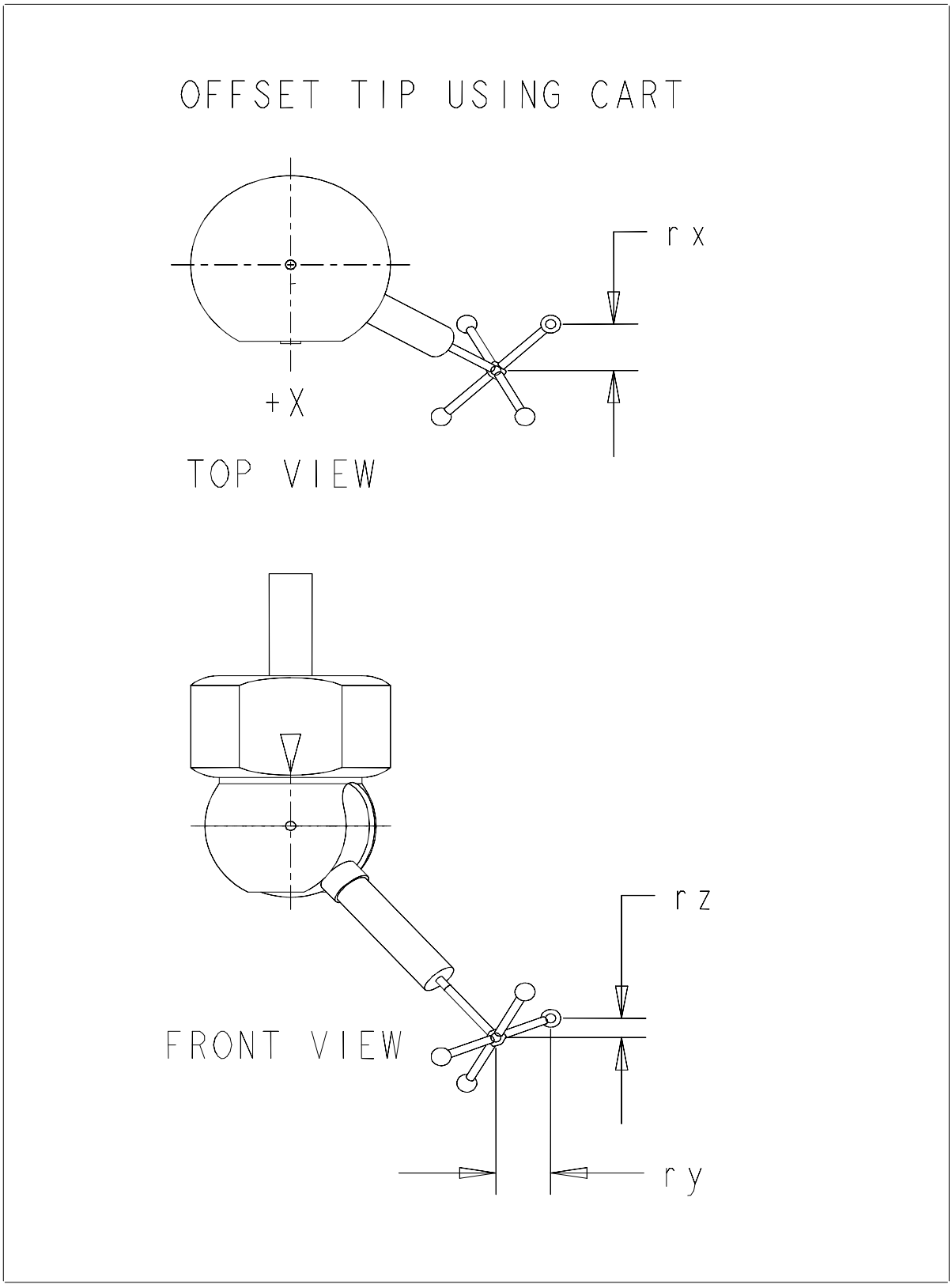


Figure B.75 — Offset tip Cartesian

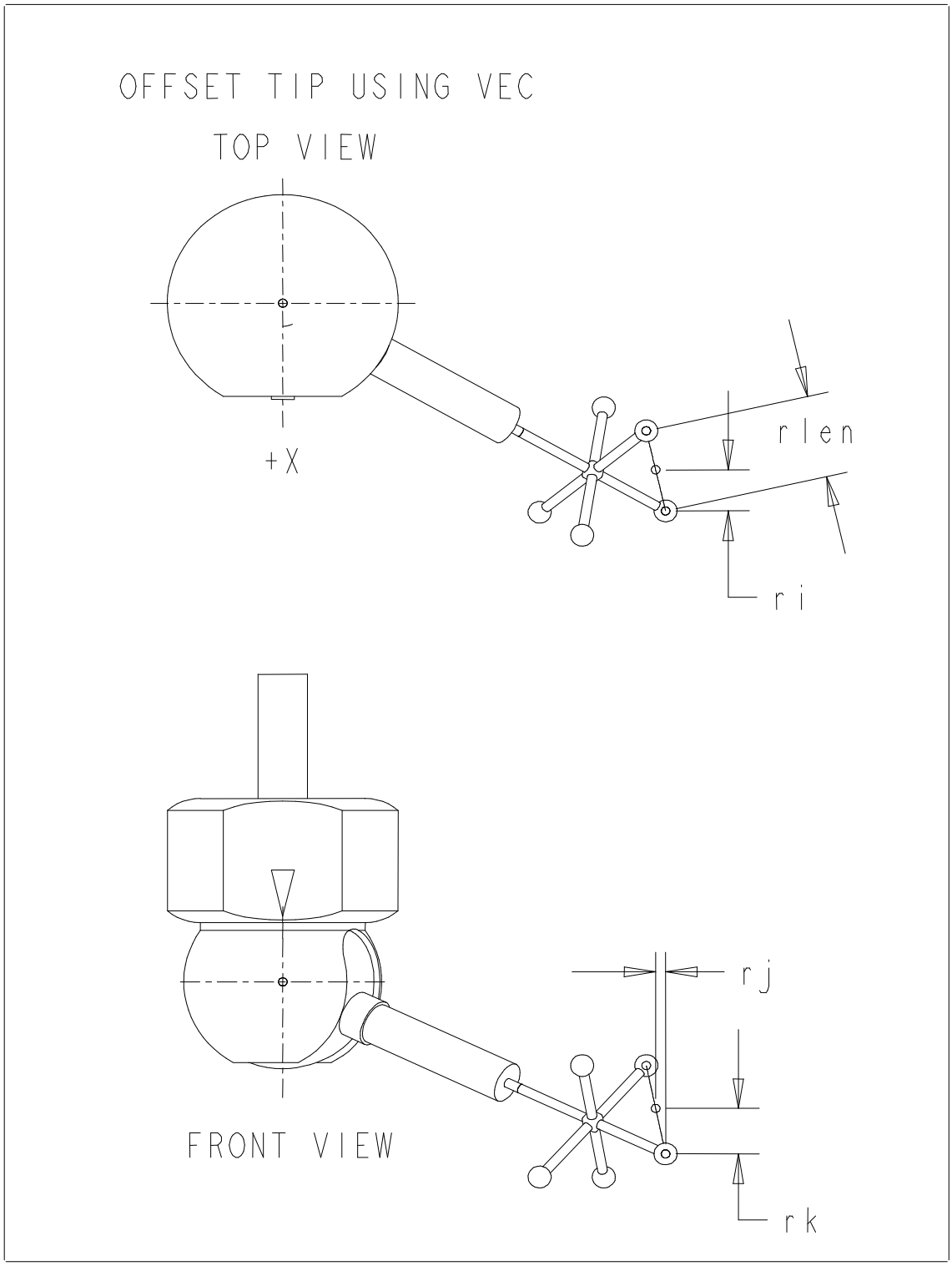


Figure B.76 — Offset tip vector

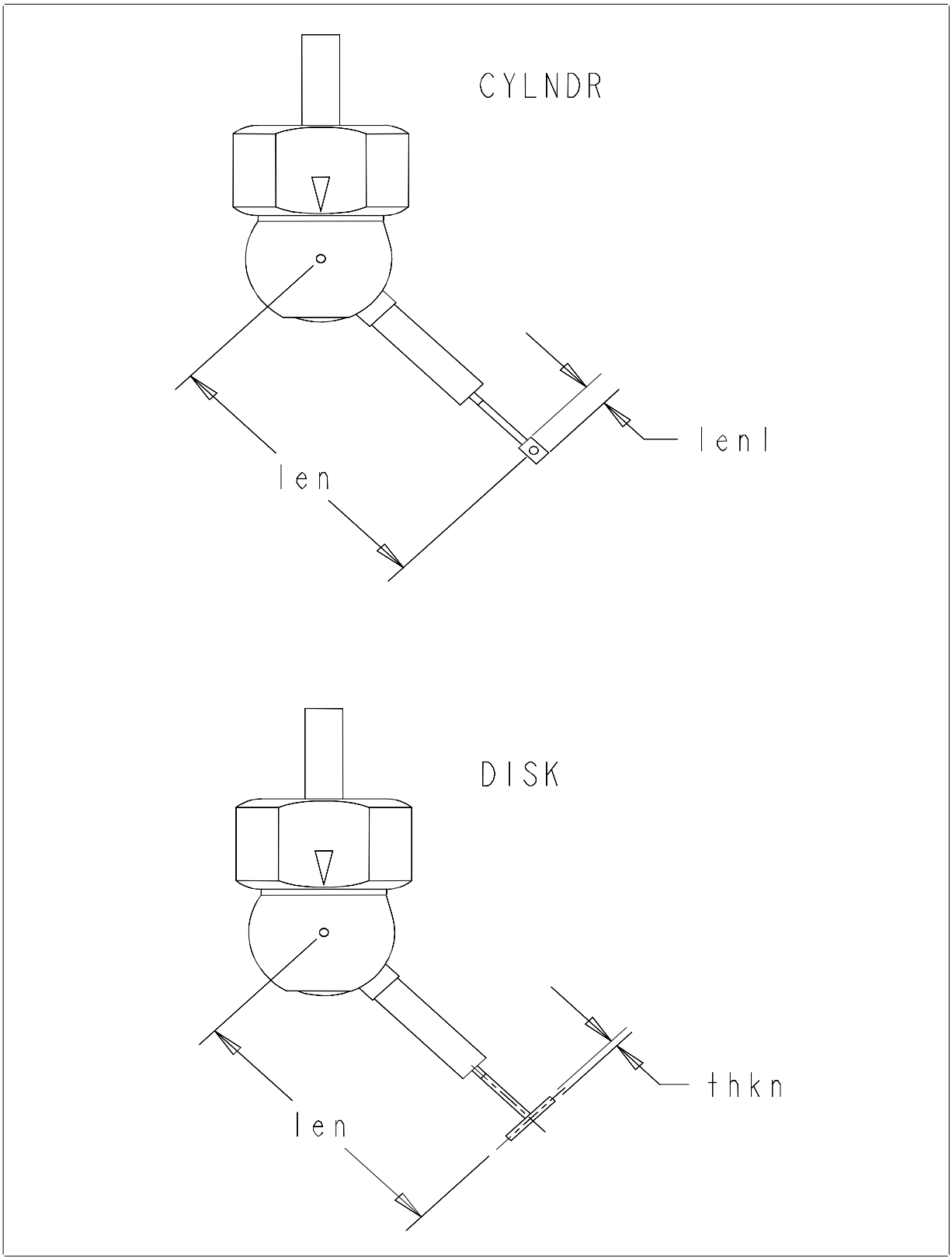


Figure B.77 — Cylindrical and disk tips

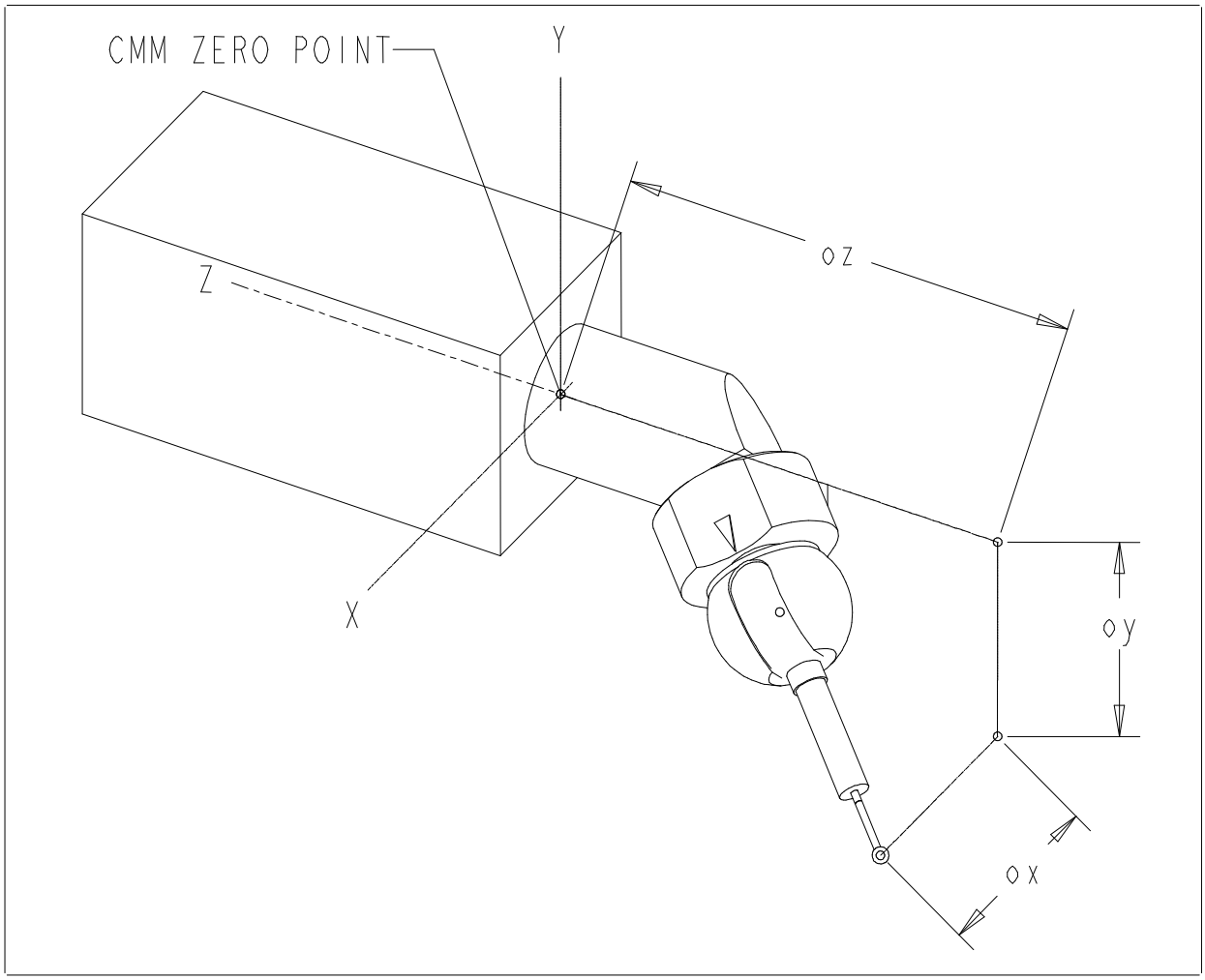


Figure B.78 — Offset tip outputs

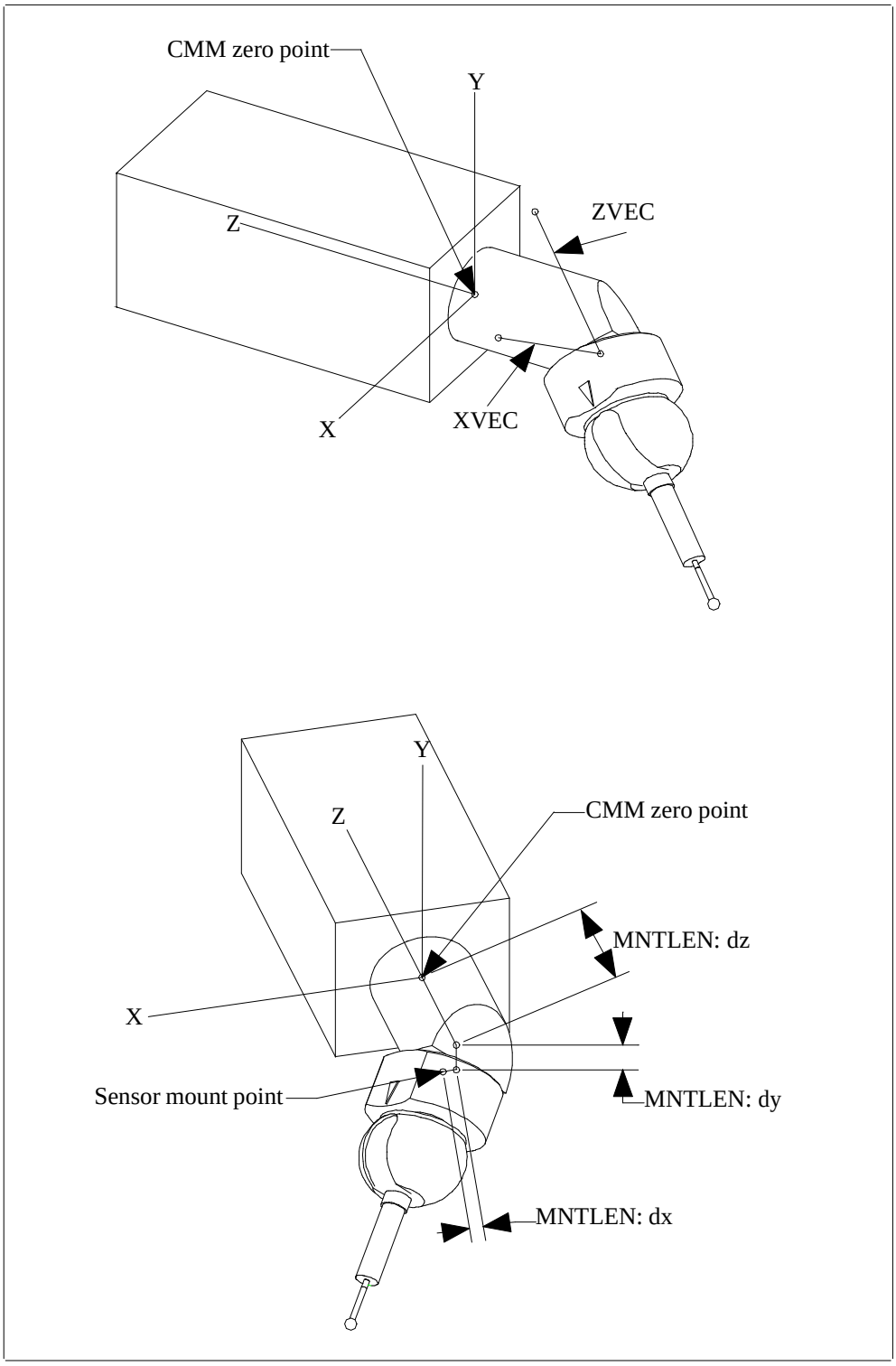


Figure B.79 — Sensor mount axis illustration

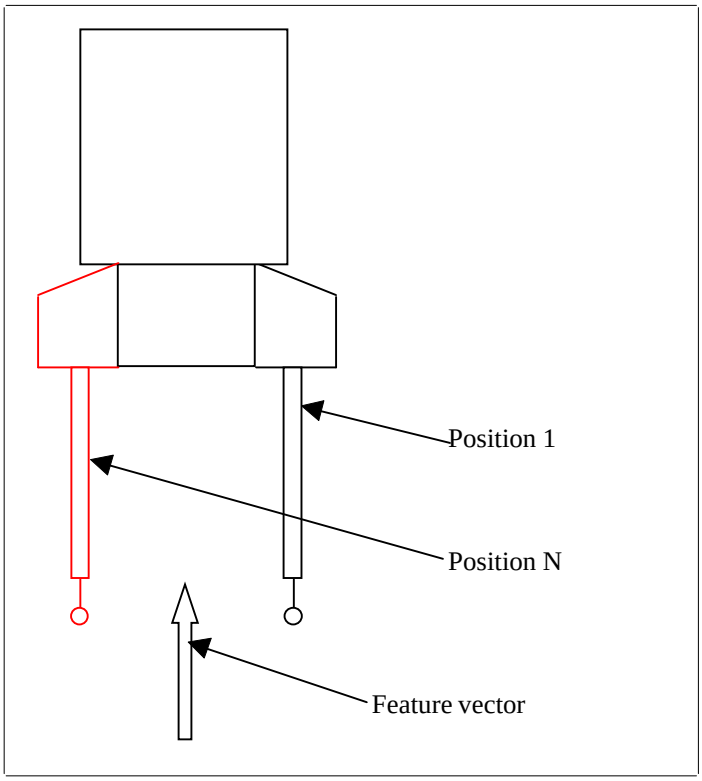


Figure B.80 — SNSLCT, sensor selection illustration

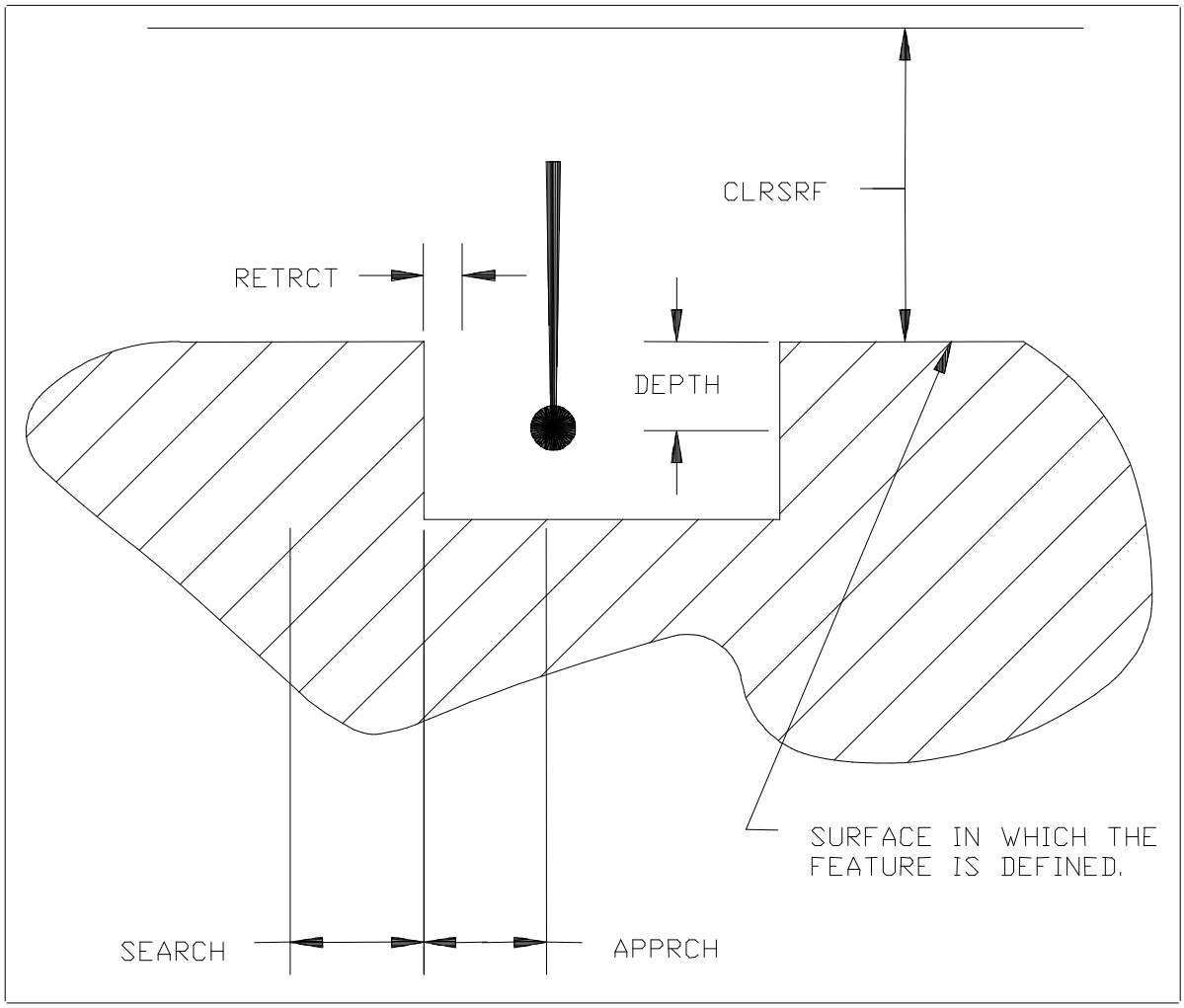


Figure B.81 — Sensor setting illustration using a typical cross section of a hole

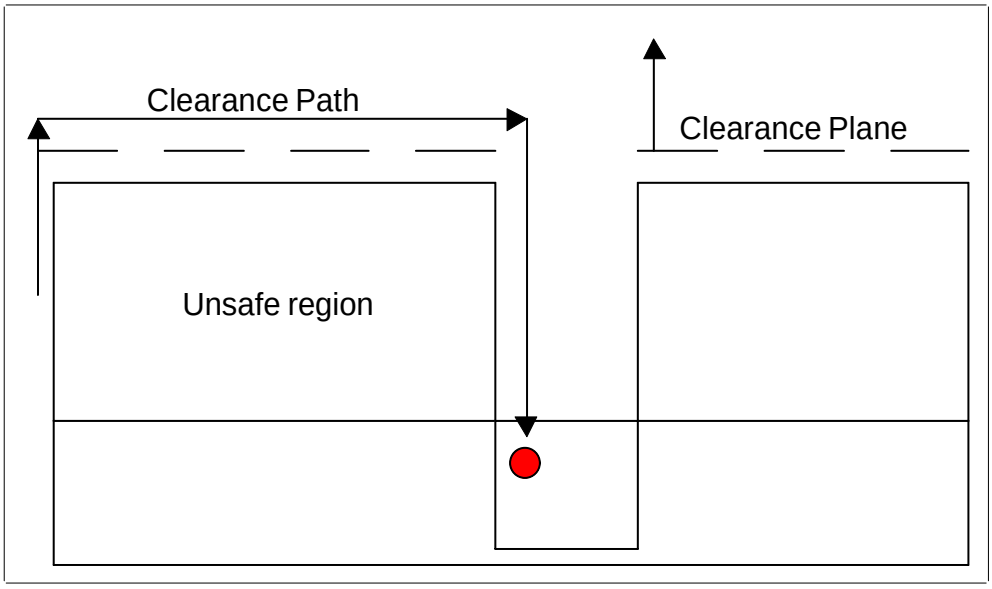
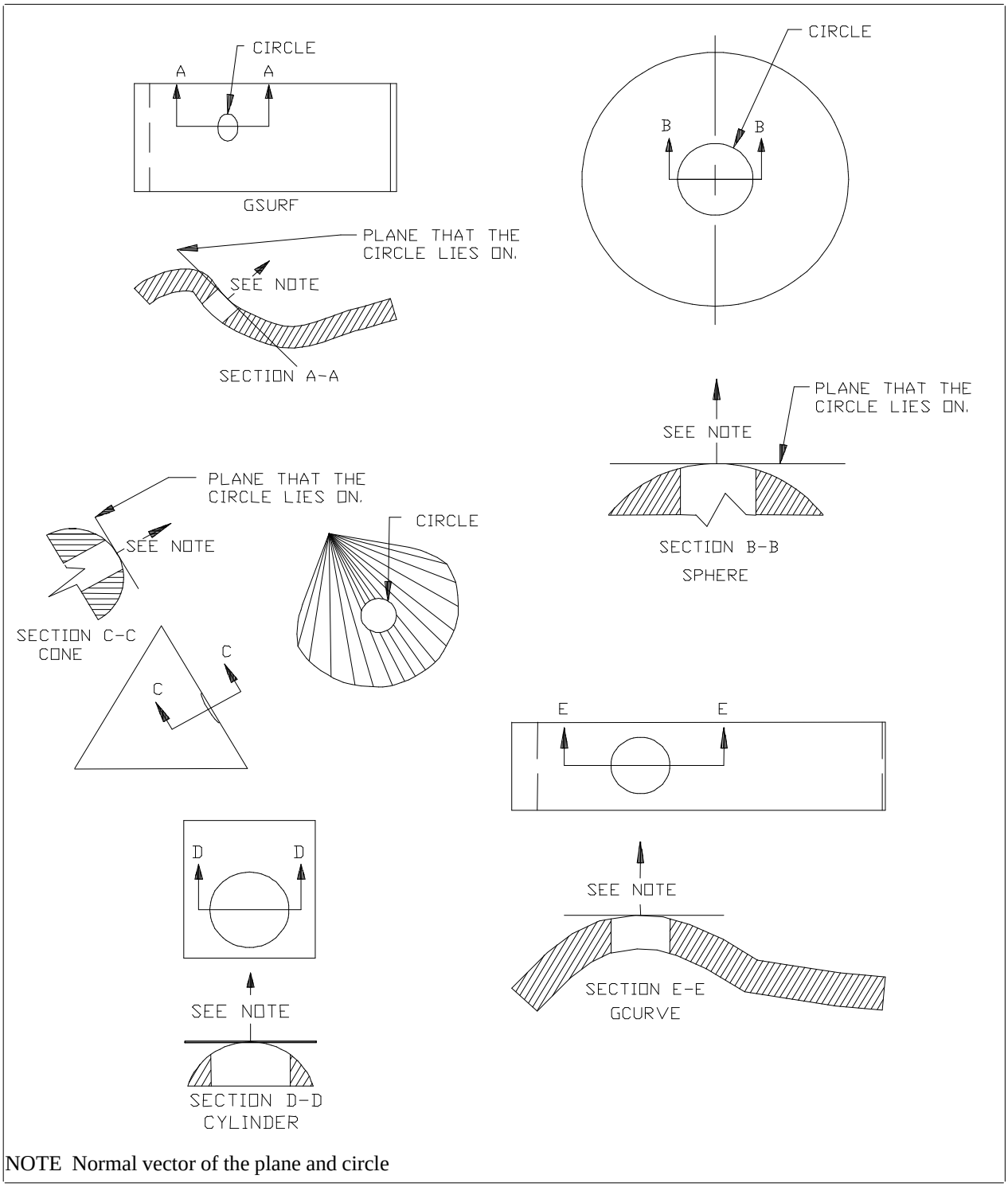


Figure B.82 — Sensor setting illustration of a typical clearance plane



NOTE Normal vector of the plane and circle

Figure B.83 — SENS or distance illustration

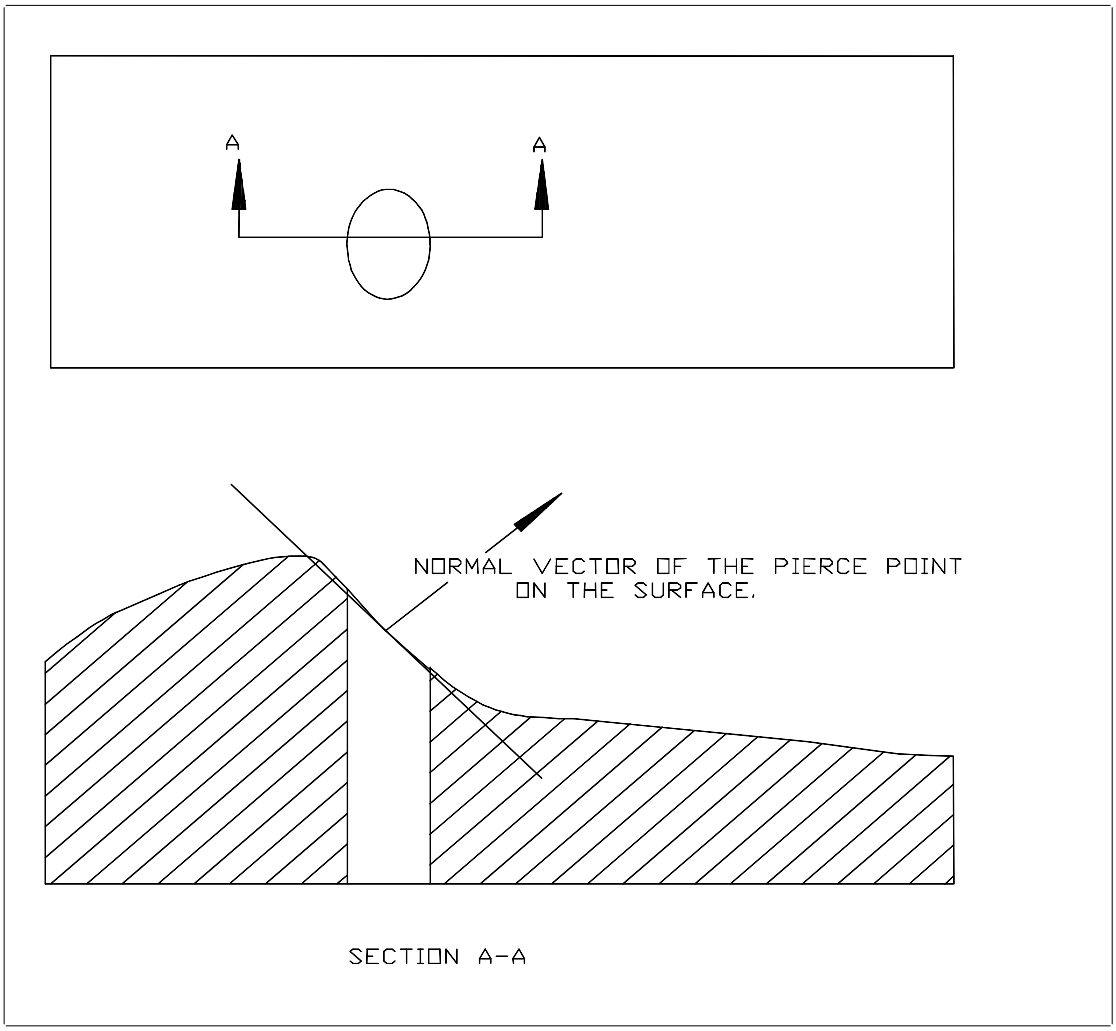


Figure B.84 — Surface penetration illustration

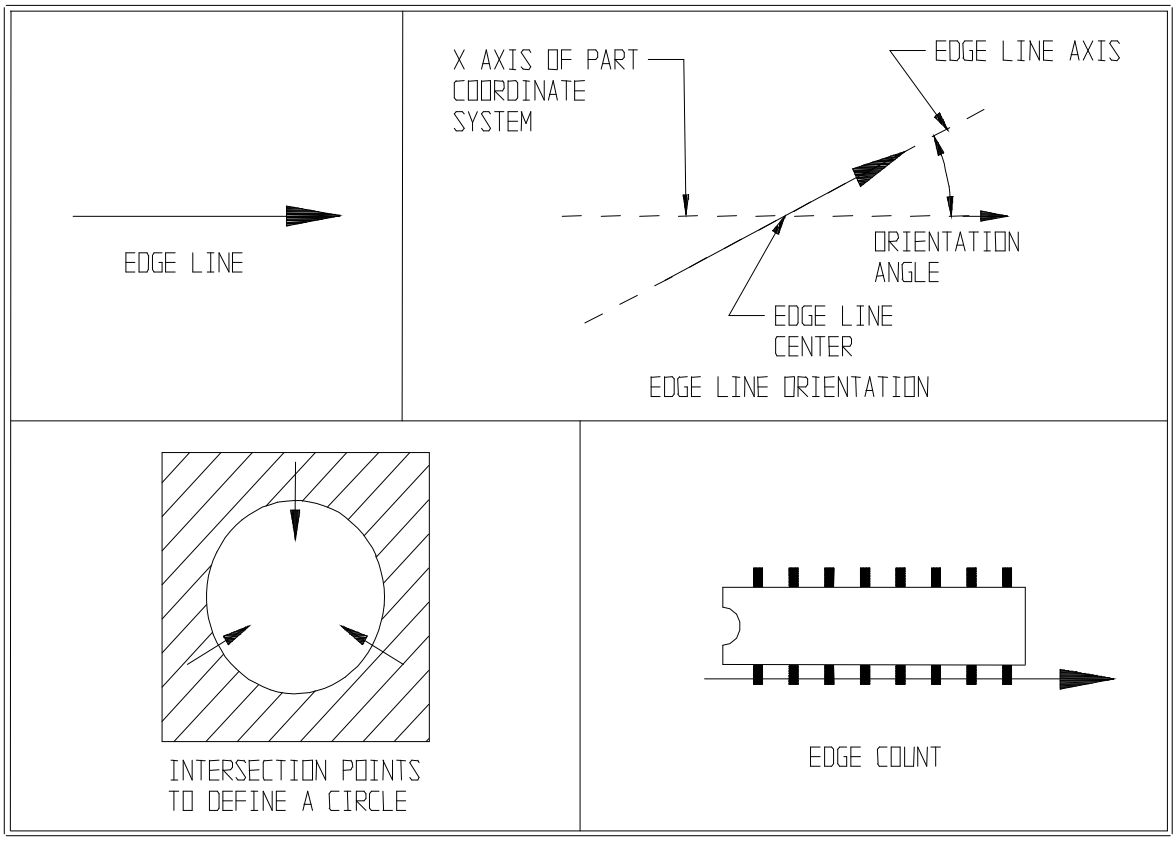


Figure B.85 — Edge line orientation and applications

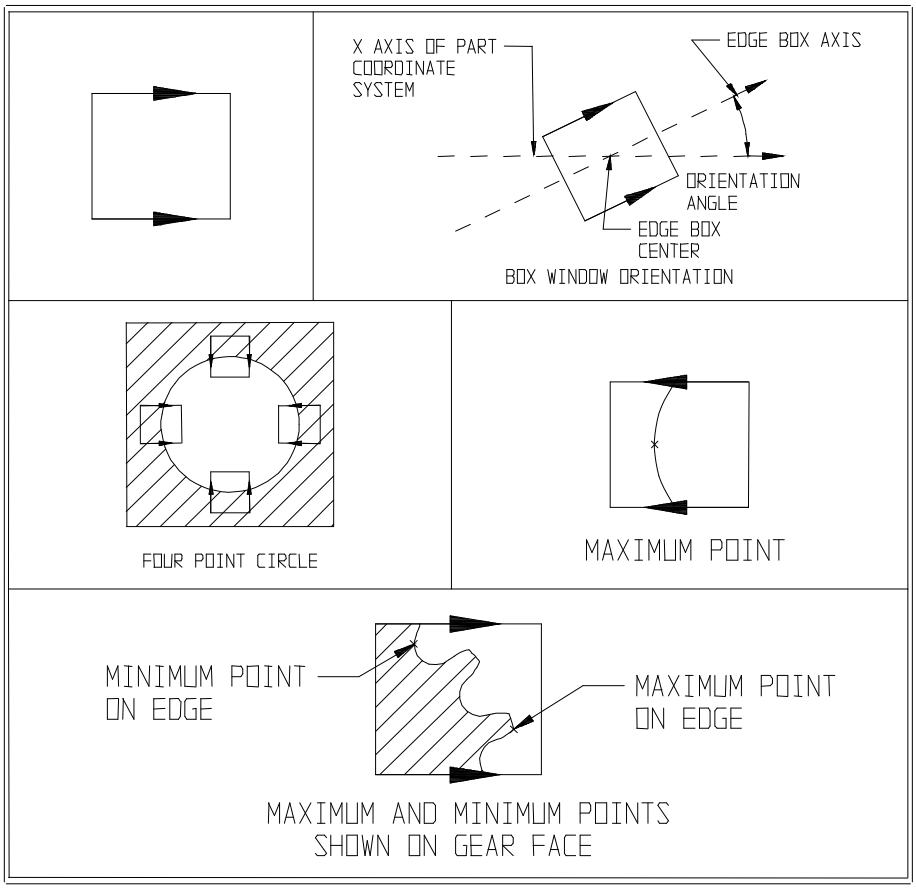


Figure B.86 — Video box window orientation and applications

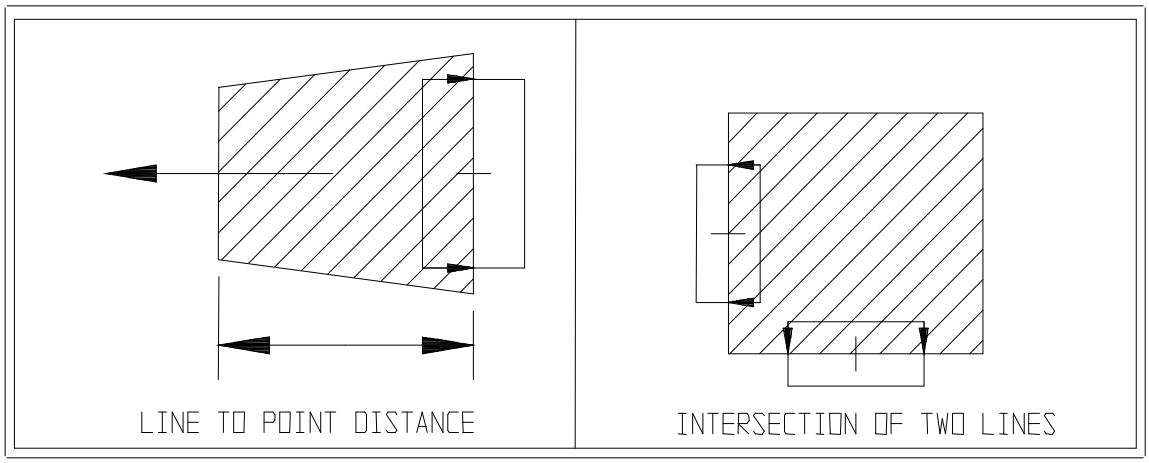


Figure B.87 — Box window applications

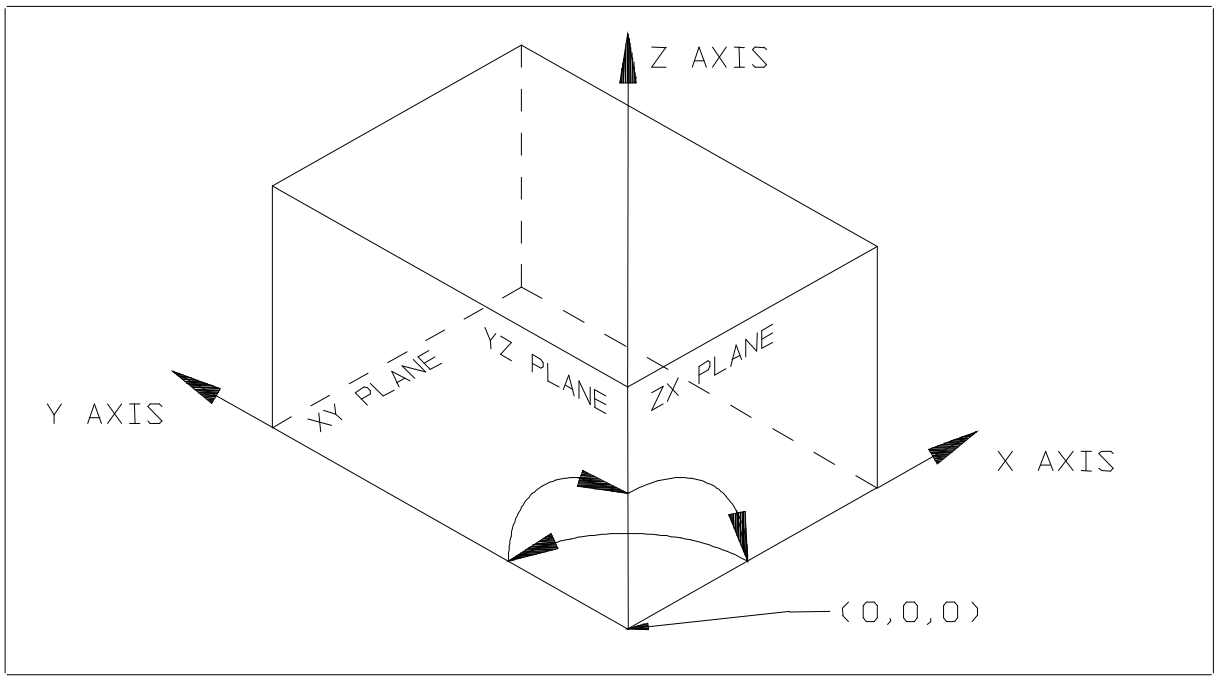


Figure B.88 — The working plane

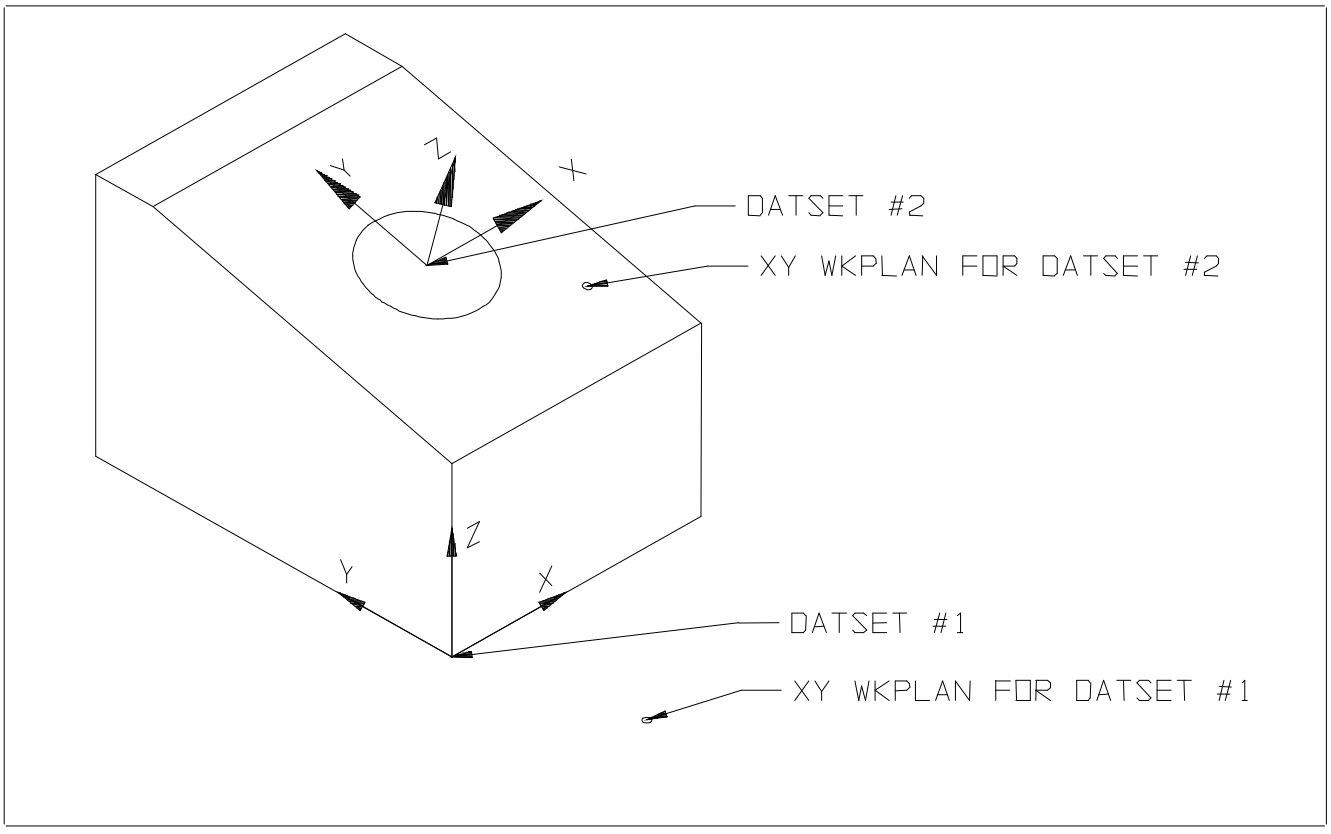
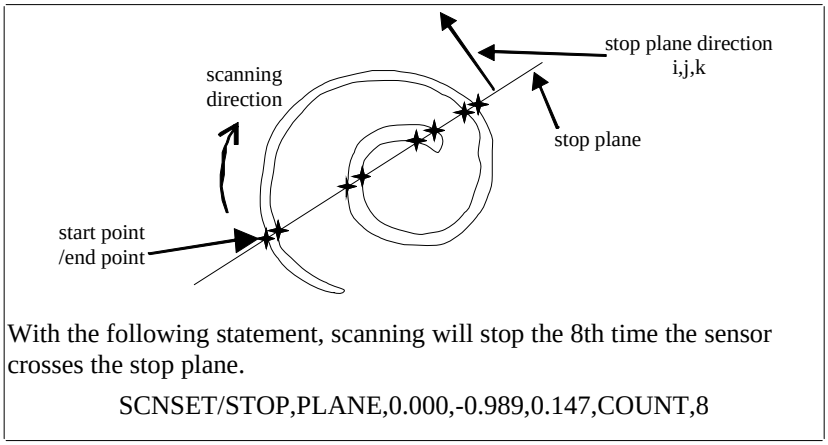


Figure B.89 — Work Plane change as a result of a new DATSET statement execution



With the following statement, scanning will stop the 8th time the sensor crosses the stop plane.

SCNSET/STOP,PLANE,0.000,-0.989,0.147,COUNT,8

Figure B.90 — Stop plane when scannig

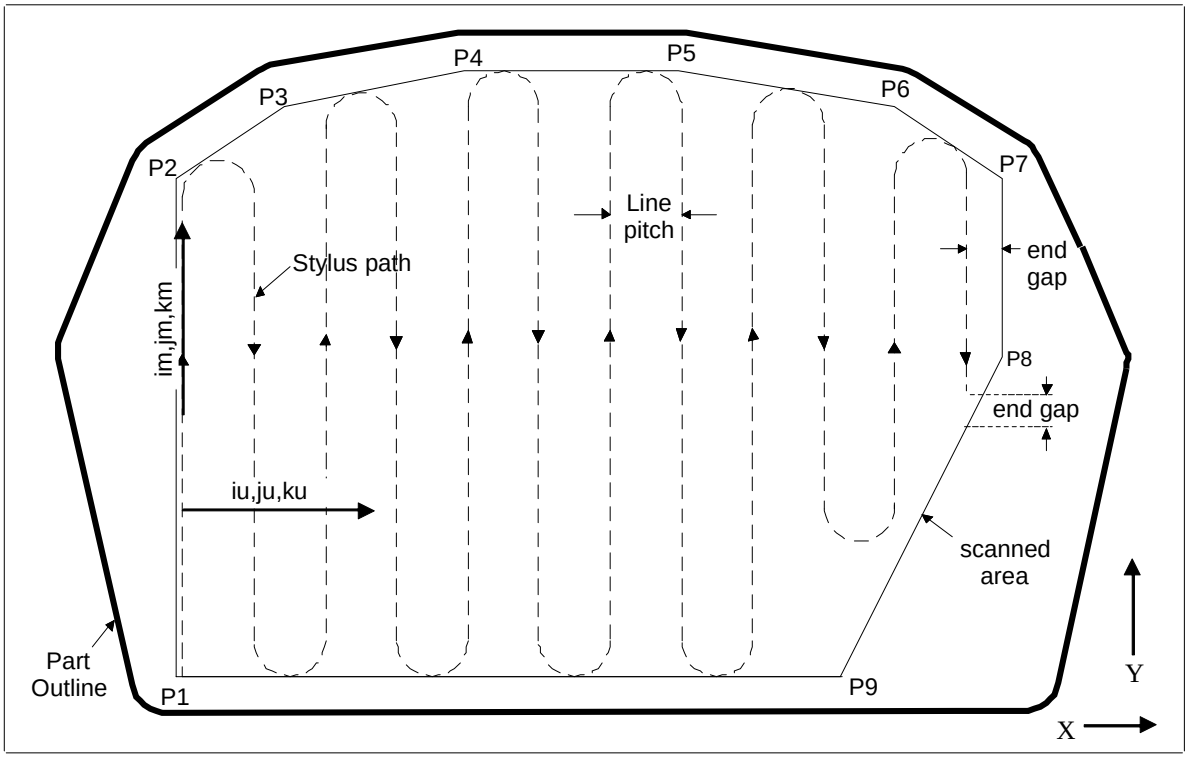


Figure B.91 — Scan path

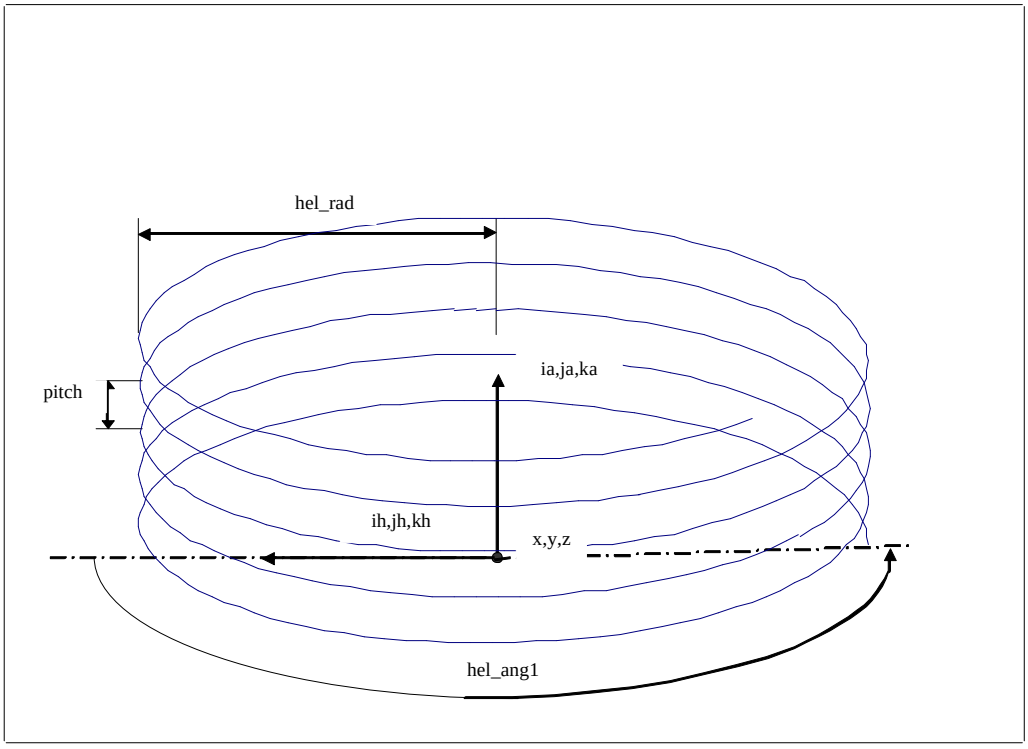


Figure B.92 — A helical scan

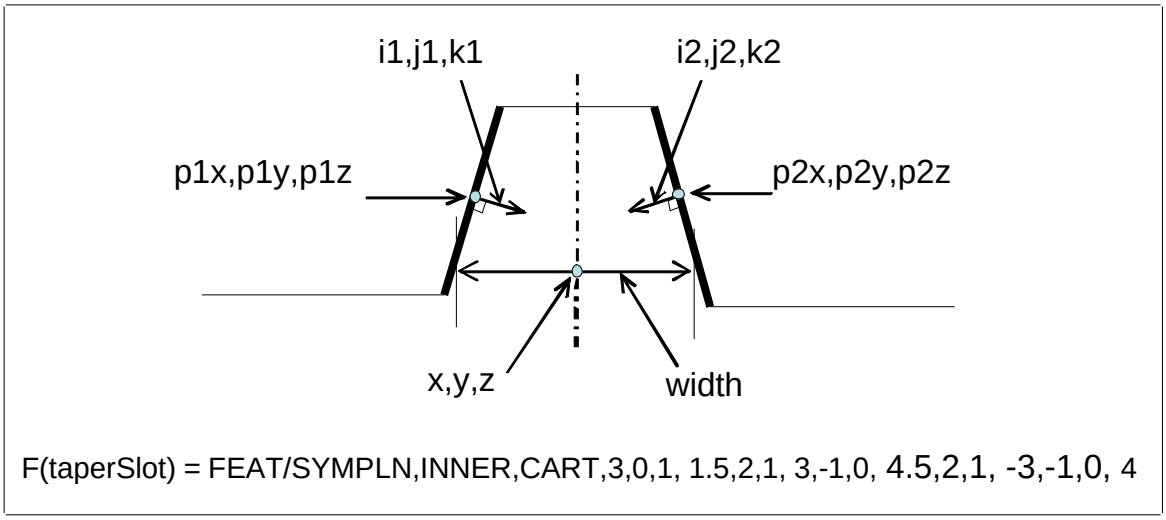


Figure B.93 — A symmetric plane feature

```
F(spindle) = FEAT/REVSURF,OUTER,CART $
,0,0,0,0,0,1, 6,0,-2,6,0,0,3,0,0.5,4,0,12,4.1,0 $
,12.5,3.6,0,13,2,1,13.5,2,0,14
```

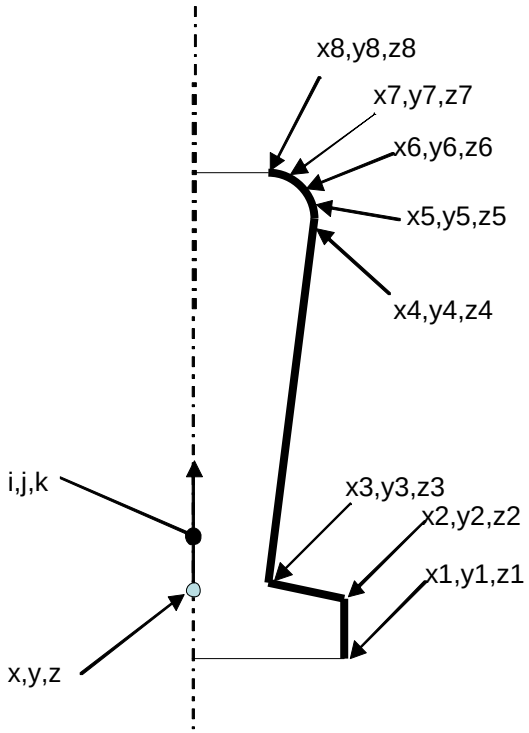


Figure B.94 — A surface-of-revolution feature

Annex C
(normative)

Standard characterization file

```
(*============================================================*)
(*                                                            *)
(* This file assumes line continuations, blank lines, and     *)
(* comments do not occur.  These should be removed by a       *)
(* preprocess.                                                *)
(*                                                            *)
(* The language used in this file to describe DMIS input      *)
(* conforms to the ISO 14977 standard for EBNF (Extended      *)
(* Backus-Naur Form), with the following conventions.         *)
(*                                                            *)
(* 1. Words spelled here with all upper case letters are DMIS  *)
(*    keywords.  The actual spelling of these words in an input*)
(*    file may use upper or lower case letters.  If a keyword  *)
(*    is not spelled as it appears, the spelling is given      *)
(*    here.  Otherwise, its spelling is not given.             *)
(*                                                            *)
(* 2. Words spelled here starting with an upper case letter    *)
(*    and containing lower case letters may be defined in      *)
(*    terms of characters alone but are not defined here.      *)
(*    They are listed separately below with dummy definitions. *)
(*                                                            *)
(* 3. Words spelled here starting with a lower case letter     *)
(*    are non-terminal symbols.  Every one of them appears     *)
(*    exactly once as the symbol being defined (i.e. the       *)
(*    symbol on the left side of the equal sign).              *)
(*                                                            *)
(* 4. The end of a line is represented by # .                  *)
(*                                                            *)
(* Meanings of the standard EBNF symbols used here are:        *)
(*   [  ]     ... optional expression (may be nested)          *)
(*   n*[  ]   ... optional expression repeated at most n times *)
(*   |        ... alternative expression                       *)
(*   (  )     ... group (the usual meaning of parentheses)     *)
(*   '  '     ... literal text (one or more characters)        *)
(*   =        ... assignment                                   *)
(*   ,        ... separates terms not otherwise separated      *)
(*   ;        ... end of statement                             *)
(*                                                            *)
(* Since EBNF and DMIS both use commas profusely, the DMIS     *)
(* comma is defined here as "c" rather than shown as ',' .     *)

(*============================================================*)

(*============================================================*)
(* starting symbol, dmisItem, and dmisItemList *)
(*============================================================*)

inputFile =
     dmisFirstStatement, [dmisItemList], endfilStm
   ;
```

```
dmisItem =
      dmisFreeStatement
    | dmisBlock
    ;

dmisItemList =
      [dmisItemList], dmisItem
    ;

(*=============================================================*)
(* EBNF requires that symbol names start with a letter, so     *)
(* DMIS minor words that do not start with letters have been   *)
(* given alternate names that are used throughout this file.   *)
(* Also, the DMIS minor word STDDEV_LIMIT is used without the  *)
(* underscore, since underscores are not allowed in EBNF       *)
(* symbol names.  The actual spellings of these words, as would *)
(* appear in a DMIS input file, are given immediately below.   *)
(*=============================================================*)

AND = '.', ('A'|'a'), ('N'|'n'), ('D'|'d'), '.'
    ;

EQ = '.', ('E'|'e'), ('Q'|'q'), '.'
    ;

FALSE = '.', ('F'|'f'), ('A'|'a'), ('L'|'l'), ('S'|'s'), ('E'|'e'), '.'
    ;

FOURPOINT = '4', ('P'|'p'), ('O'|'o'), ('I'|'i'), ('N'|'n'), ('T'|'t')
    ;

GE= '.', ('G'|'g'), ('E'|'e'), '.'
    ;

GT = '.', ('G'|'g'), ('T'|'t'), '.'
    ;

LE = '.', ('L'|'l'), ('E'|'e'), '.'
    ;

LT = '.', ('L'|'l'), ('T'|'t'), '.'
    ;

MINUSPRBRAD = '-', ('P'|'p'), ('R'|'r'), ('B'|'b'), ('R'|'r'),
     ('A'|'a'), ('D'|'d')
    ;

MINUSXDIR = '-', ('X'|'x'), ('D'|'d'), ('I'|'i'), ('R'|'r')
    ;

MINUSYDIR = '-', ('Y'|'y'), ('D'|'d'), ('I'|'i'), ('R'|'r')
    ;

MINUSZDIR = '-', ('Z'|'z'), ('D'|'d'), ('I'|'i'), ('R'|'r')
    ;

NE = '.', ('N'|'n'), ('E'|'e'), '.'
    ;
```

```
NOT = '.', ('N'|'n'), ('O'|'o'), ('T'|'t'), '.'
    ;

OR = '.', ('O'|'o'), ('R'|'r'), '.'
    ;

STDDEVLIMIT = ('S'|'s'), ('T'|'t'), ('D'|'d'), ('D'|'d'), ('E'|'e'), ('V'|'v'),
      '_', ('L'|'l'), ('I'|'i'), ('M'|'m'), ('I'|'i'), ('T'|'t')
    ;

THREED = '3', ('D'|'d')
    ;

TRUE = '.', ('T'|'t'), ('R'|'r'), ('U'|'u'), ('E'|'e'), '.'
    ;

TWOD = '2', ('D'|'d')
    ;

TWORC = '2', ('R'|'r'), ('C'|'c')
    ;

(*============================================================*)
(* undefined non-terminals *)
(*============================================================*)

(* This section provides dummy definitions for undefined non-terminals *)
(* so that the file may be parsed without error by EBNF parsers.       *)

BoolVarName = 'Not defined here'
    ;

CharString = 'Not defined here'
    ;

DeclVarName = 'Not defined here'
    ;

IntString = 'Not defined here'
    ;

IntVarName = 'Not defined here'
    ;

LblName = 'Not defined here'
    ;

MacroVarName = 'Not defined here'
    ;

RealString = 'Not defined here'
    ;

RealVarName = 'Not defined here'
    ;

StringVarName = 'Not defined here'
    ;

VectorVarName = 'Not defined here'
```

```
   ;

(*============================================================*)
(* DMIS first statement *)
(*============================================================*)

dmisFirstStatement =
      dmismdStm
    | dmismnStm
    ;

(*============================================================*)
(* DMIS block *)
(*============================================================*)

dmisBlock =
      calibMasterBlock (* FULL *)
    | calibRtabBlock (* FULL *)
    | calibSensBlock (* FULL *)
    | dmisOffBlock (* FULL *)
    | doBlock (* FULL *)
    | gotargBlock (* FULL *)
    | ifBlock (* FULL *)
    | macroBlock (* FULL *)
    | measBlock (* FULL *)
    | selectBlock (* FULL *)
    | simreqtBlock (* FULL *)
    | xternBlock (* FULL *)
    ;

(*============================================================*)
(* calibrate master sensor block *)
(*============================================================*)

calibMasterBlock =
       calibMasterStm, [calibSensBlockItemList], endmesStm
    ;

(*============================================================*)
(* calibrate rotary table block *)
(*============================================================*)

calibRtabBlock =
      calibRtabStm, [measBlockItemList], endmesStm
    ;

(*============================================================*)
(* calibrate sensor block *)
(*============================================================*)

calibSensBlock =
      calibSensStm, [calibSensBlockItemList], endmesStm
    ;

calibSensBlockItemList =
      [calibSensBlockItemList], calibSensBlockItem
    ;

calibSensBlockStatement =
      jumpStm
```

```
    | aclratStm
    | assignStm
    | badtstStm
    | czslctStm
    | dmehwStm
    | dmeswStm
    | errorStm
    | fedratStm
    | finposStm
    | fromStm
    | gohomeStm
    | gotoStm
    | includStm
    | jumptoStm
    | obtainStm
    | pameasStm
    | ptmeasStm
    | rapidStm
    | rotabStm
    | rotsetStm
    | snsetStm
    | textStm
    | valueStm
    | wkplanStm
    ;

calibSensBlockItem =
      calibSensBlockStatement
    | calibSensDoBlock
    | calibSensIfBlock
    | calibSensSelectBlock
    | gotargBlock
    ;

calibSensDoBlock =
      doStm, [calibSensBlockItemList], enddoStm
    ;

calibSensIfBlock =
      ifStm, [calibSensBlockItemList],
      [elseStm, [calibSensBlockItemList]], endifStm
    ;

calibSensSelectBlock =
      selectStm, calibSensCaseBlockList,
      [calibSensDefaultCaseBlock], endselStm
    ;

calibSensCaseBlockList =
      [calibSensCaseBlockList], calibSensCaseBlock
    ;

calibSensCaseBlock =
      caseList, [calibSensBlockItemList], endcasStm
    ;

calibSensDefaultCaseBlock =
      dftcasStm, [calibSensBlockItemList], endcasStm
    ;
```

```
(*============================================================*)
(* case block *)
(*============================================================*)

caseBlock =
      caseList, [dmisItemList], endcasStm
   ;

caseList =
      [caseList], caseStm
   ;

(*============================================================*)
(* default case block *)
(*============================================================*)

defaultCaseBlock =
      dftcasStm, [dmisItemList], endcasStm
   ;

(*============================================================*)
(* dmis off block *)
(*============================================================*)

dmisOffBlock =
      dmisOffStm, [noParseStmList], dmisOnStm
   ;

(*============================================================*)
(* do block *)
(*============================================================*)

doBlock =
      doStm, [dmisItemList], enddoStm
   ;

(*============================================================*)
(* gotarg block *)
(*============================================================*)

gotargBlock =
      gotargStm, gotoStm, gotoList, endgoStm
   ;

gotoList =
      [gotoList], gotoStm
   ;

(*============================================================*)
(* if block *)
(*============================================================*)

ifBlock =
      ifStm, [dmisItemList], [elseStm, [dmisItemList]], endifStm
   ;

(*============================================================*)
(* macro block *)
(*============================================================*)
```

```
macroBlock =
      macroStm, [noParseStmList], endmacStm
    ;

(*============================================================*)
(* measurement block *)
(*============================================================*)

measBlock =
      measStm, [measBlockItemList], endmesStm
    | rmeasStm, [measBlockItemList], endmesStm
    ;

measBlockItemList =
      [measBlockItemList], measBlockItem
    ;

measBlockStatement =
      calibSensBlockStatement
    | crslctStm
    | recallSensorStm
    | saveSensorStm
    | snslctStm
    ;

measBlockItem =
      measBlockStatement
    | measDoBlock
    | measIfBlock
    | measSelectBlock
    ;

measDoBlock =
      doStm, [measBlockItemList], enddoStm
    ;

measIfBlock =
      ifStm, [measBlockItemList], [elseStm, [measBlockItemList]], endifStm
    ;

measSelectBlock =
      selectStm, measCaseBlockList, [measDefaultCaseBlock], endselStm
    ;

measCaseBlockList =
      [measCaseBlockList], measCaseBlock
    ;

measCaseBlock =
      caseStm, [measBlockItemList], endcasStm
    ;

measDefaultCaseBlock =
      dftcasStm, [measBlockItemList], endcasStm
    ;

(*============================================================*)
(* select block *)
(*============================================================*)
```

```
selectBlock =
      selectStm, caseBlockList, [defaultCaseBlock], endselStm
    ;

caseBlockList =
      [caseBlockList], caseBlock
    ;

(*=============================================================*)
(* simultaneous requirement block *)
(*=============================================================*)

simreqtBlock =
      simreqtStm, [evalOrOutputList], endsimreqtStm
    ;

evalOrOutputList =
      [evalOrOutputList], evalOrOutput
    ;

evalOrOutput =
      evalStm
    | outputStm
    ;

(*=============================================================*)
(* xtern block *)
(*=============================================================*)

xternBlock =
      xternStm, [extfilList], endxtnStm
    ;

extfilList =
      [extfilList], extfilStm
    ;

(*=============================================================*)
(* DMIS input statements that do not start or end a block,     *)
(* except dmismdStm, dmismnStm, and endfilStm                  *)
(*=============================================================*)

dmisFreeStatement =
      jumpStm (* FULL *)
    | aclratStm (* FULL *)
    | algdefStm (* FULL *)
    | assignStm (* FULL *)
    | badtstStm (* FULL *)
    | boundStm (* FULL *)
    | callStm  (* FULL *)
    | clmpidStm (* FULL *)
    | clmpsnStm (* FULL *)
    | closeStm (* FULL *)
    | cmpntgrpStm (* FULL *)
    | cnfrmrulStm (* FULL *)
    | constStm (* FULL *)
    | crgdefStm (* FULL *)
    | crmodeStm (* FULL *)
    | crosclStm (* FULL *)
    | crslctStm (* FULL *)
```

```
| cutcomStm (* FULL *)
| czoneStm (* FULL *)
| czslctStm (* FULL *)
| datdefStm (* FULL *)
| datsetStm (* FULL *)
| dattrgdefStm (* FULL *)
| declStm (* FULL *)
| decplStm (* FULL *)
| deleteStm (* FULL *)
| deviceStm (* FULL *)
| displyStm (* FULL *)
| dmehwStm (* FULL *)
| dmeidStm (* FULL *)
| dmeswStm (* FULL *)
| dmeswiStm (* FULL *)
| dmeswvStm (* FULL *)
| dmisOnStm (* FULL *)
| equateStm (* FULL *)
| errorStm (* FULL *)
| evalStm (* FULL *)
| extensStm (* FULL *)
| featStm (* FULL *)
| fedratStm (* FULL *)
| fildefStm (* FULL *)
| filnamStm (* FULL *)
| finposStm (* FULL *)
| fixtidStm (* FULL *)
| fixtsnStm (* FULL *)
| flyStm (* FULL *)
| fromStm (* FULL *)
| geoalgStm (* FULL *)
| geomStm (* FULL *)
| gohomeStm (* FULL *)
| gotoStm (* FULL *)
| groupStm (* FULL *)
| includStm (* FULL *)
| iteratStm (* FULL *)
| jumptoStm (* FULL *)
| keycharStm (* FULL *)
| litdefStm (* FULL *)
| locateStm (* FULL *)
| lotidStm (* FULL *)
| matdefStm (* FULL *)
| mfgdevStm (* FULL *)
| modeStm (* FULL *)
| obtainStm (* FULL *)
| openStm (* FULL *)
| operidStm (* FULL *)
| outputStm (* FULL *)
| pameasStm (* FULL *)
| partidStm (* FULL *)
| partrvStm (* FULL *)
| partsnStm (* FULL *)
| pathStm (* FULL *)
| planidStm (* FULL *)
| popStm (* FULL *)
| prcompStm (* FULL *)
| prevopStm (* FULL *)
| procidStm (* FULL *)
| promptStm (* FULL *)
```

```
    | psthruStm (* FULL *)
    | ptbuffStm (* FULL *)
    | ptmeasStm (* FULL *)
    | pushStm (* FULL *)
    | qisdefStm (* FULL *)
    | rapidStm (* FULL *)
    | readStm (* FULL *)
    | recallStm (* FULL *)
    | refmntStm (* FULL *)
    | reportStm (* FULL *)
    | resumeStm (* FULL *)
    | rotabStm (* FULL *)
    | rotateStm (* FULL *)
    | rotdefStm (* FULL *)
    | rotsetStm (* FULL *)
    | saveStm (* FULL *)
    | scnmodStm (* FULL *)
    | scnsetStm (* FULL *)
    | sensorStm (* FULL *)
    | snsdefStm (* FULL *)
    | snsetStm (* FULL *)
    | snsgrpStm (* FULL *)
    | snslctStm (* FULL *)
    | snsmntStm (* FULL *)
    | tecompStm (* FULL *)
    | textStm (* FULL *)
    | thldefStm (* FULL *)
    | tolStm (* FULL *)
    | tooldfStm (* FULL *)
    | transStm (* FULL *)
    | uncertalgStm (* FULL *)
    | uncertsetStm (* FULL *)
    | unitsStm (* FULL *)
    | valueStm (* FULL *)
    | vformStm (* FULL *)
    | windefStm (* FULL *)
    | wkplanStm (* FULL *)
    | wristStm (* FULL *)
    | writeStm (* FULL *)
    | xtractStm (* FULL *)
    ;

(*=============================================================*)
(* comma *)
(*=============================================================*)

c = ','
    ;

(*=============================================================*)
(* base types, variables and functions *)
(*=============================================================*)

anyVal =
      stringVal
    | boolVal
    | rentVal
    | vectorVal
    ;
```

```
stringVal =
      stringConst
    | stringFunc
    | stringVar
    ;

boolVal =
      boolConst
    | boolExpr
    | boolFunc
    | boolVar
    ;

intVal =
      intConst
    | intExpr
    | intFunc
    | intVar
    ;

realVal =
      realConst
    | realExpr
    | realFunc
    | realVar
    ;

rentVal =
      realVal
    | intVal
    ;

vectorVal =
      vectorFunc
    | vectorExpr
    | vectorVar
    ;

stringConst =
      CharString
    ;

boolConst =
      TRUE
    | FALSE
    ;

intConst =
      IntString
    ;

realConst =
      RealString
    ;

stringFunc =
      stringFuncChr
    | stringFuncConcat
    | stringFuncElemnt
    | stringFuncLwc
```

```
    | stringFuncRpt
    | stringFuncScfeat
    | stringFuncScsns
    | stringFuncSdate
    | stringFuncSdatetime
    | stringFuncSelapsetime
    | stringFuncSerror
    | stringFuncSmode
    | stringFuncStime
    | stringFuncStr
    | stringFuncSubstr
    | stringFuncTrim
    | stringFuncUpc
    ;

boolFunc =
      boolFuncBadgt
    | boolFuncBadpt
    | boolFuncEof
    | boolFuncEoln
    | boolFuncExist
    ;

intFunc =
      intFuncAbs
    | intFuncIndx
    | intFuncInt
    | intFuncLen
    | intFuncMod
    | intFuncNint
    | intFuncOrd
    | intFuncPtdata
    | intFuncSensnotouch
    | intFuncSign
    | intFuncSiltch
    ;

realFunc =
      realFuncAbs
    | realFuncAcos
    | realFuncAsin
    | realFuncAtan
    | realFuncAtan2
    | realFuncCos
    | realFuncDble
    | realFuncDtor
    | realFuncExp
    | realFuncLn
    | realFuncLog
    | realFuncMn
    | realFuncMod
    | realFuncMx
    | realFuncQtemp
    | realFuncRand
    | realFuncRl
    | realFuncRtod
    | realFuncSign
    | realFuncSin
    | realFuncSqrt
    | realFuncTan
```

```
    | realFuncVal
    | realFuncVdot
    | realFuncVecX
    | realFuncVecY
    | realFuncVecZ
    | realFuncVmag
    ;

vectorFunc =
      vectorFuncVcart
    | vectorFuncVcross
    | vectorFuncVmcs
    | vectorFuncVpcs
    | vectorFuncVpol
    | vectorFuncVunit
    ;

boolExpr =
      '(', boolVal, ')'
    | stringRel
    | boolRel
    | intRel
    | realRel
    | vectorRel
    ;

stringRel =
      stringVal, EQ, stringVal
    | stringVal, NE, stringVal
    | stringVal, LT, stringVal
    | stringVal, LE, stringVal
    | stringVal, GT, stringVal
    | stringVal, GE, stringVal
    ;

boolRel =
      NOT, boolVal
    | boolVal, AND, boolVal
    | boolVal, OR, boolVal
    | boolVal, EQ, boolVal
    | boolVal, NE, boolVal
    ;

intRel =
      intVal, EQ, intVal
    | intVal, NE, intVal
    | intVal, LT, intVal
    | intVal, LE, intVal
    | intVal, GT, intVal
    | intVal, GE, intVal
    ;

intExpr =
      '(', intVal, ')'
    | intVal, '+', intVal
    | intVal, '-', intVal
    | intVal, '*', intVal
    | intVal, '/', intVal
    | intVal, '**', intVal
    | '-', intVal
```

```
    | '+', intVal
    ;

realRel =
      realRealRel
    | realIntRel
    | intRealRel
    ;

realRealRel =
      realVal, EQ, realVal
    | realVal, NE, realVal
    | realVal, LT, realVal
    | realVal, LE, realVal
    | realVal, GT, realVal
    | realVal, GE, realVal
    ;

intRealRel =
      intVal, EQ, realVal
    | intVal, NE, realVal
    | intVal, LT, realVal
    | intVal, LE, realVal
    | intVal, GT, realVal
    | intVal, GE, realVal
    ;

realIntRel =
      realVal, EQ, intVal
    | realVal, NE, intVal
    | realVal, LT, intVal
    | realVal, LE, intVal
    | realVal, GT, intVal
    | realVal, GE, intVal
    ;

vectorRel =
      vectorVal, EQ, vectorVal
    | vectorVal, NE, vectorVal
    ;

realExpr =
      '(', realVal, ')'
    | realRealExpr
    | realIntExpr
    | intRealExpr
    | '-', realVal
    | '+', realVal
    ;

realRealExpr =
      realVal, '+', realVal
    | realVal, '-', realVal
    | realVal, '*', realVal
    | realVal, '/', realVal
    | realVal, '**', realVal
    ;

realIntExpr =
      realVal, '+', intVal
```

```
    | realVal, '-', intVal
    | realVal, '*', intVal
    | realVal, '/', intVal
    | realVal, '**', intVal
    ;

intRealExpr =
      intVal, '+', realVal
    | intVal, '-', realVal
    | intVal, '*', realVal
    | intVal, '/', realVal
    | intVal, '**', realVal
    ;

vectorExpr =
      '(', vectorVal, ')'
    | vectorVal, '+', vectorVal
    | vectorVal, '-', vectorVal
    | vectorVal, '/', realVal
    | vectorVal, '/', intVal
    | vectorVal, '*', realVal
    | realVal  , '*', vectorVal
    | vectorVal, '*', intVal
    | intVal   , '*', vectorVal
    ;

boolVar =
      BoolVarName, [arrayIndex]
    ;

intVar =
      IntVarName, [arrayIndex]
    ;

realVar =
      RealVarName, [arrayIndex]
    ;

stringVar =
      StringVarName, [arrayIndex]
    ;

vectorVar =
      VectorVarName, [arrayIndex]
    ;

arrayIndex =
      '[', intList, ']'
    ;

stringFuncChr =
      CHR, '(', intVal, ')'
    ;

stringFuncConcat =
      CONCAT, '(', stringVal, c, stringList, ')'
    ;

stringFuncElemnt =
      ELEMNT, '(', intVal, c, stringVal, c, stringVal, ')'
```

```
    ;

stringFuncScfeat =
      SCFEAT, '(', ')'
    ;

stringFuncScsns =
      SCSNS, '(', ')'
    ;

stringFuncSdate =
      SDATE, '(', ')'
    ;

stringFuncSerror =
      SERROR, '(', ')'
    ;

stringFuncSmode =
      SMODE, '(', ')'
    ;

stringFuncStime =
      STIME, '(', ')'
    ;

stringFuncSdatetime =
      SDATETIME, '(', ')'
    ;

stringFuncRpt =
      RPT, '(', stringVal, c, intVal, ')'
    ;

stringFuncSelapsetime =
      SELAPSETIME, '(', stringVal, c, stringVal, c, strVar6, ')'
    ;

strVar6 =
      LONG
    | SHORT
    ;

stringFuncStr =
      STR, '(', rentVal, [strVar7], ')'
    ;

strVar7 =
      c, intVal, [':', intVal]
    ;

stringFuncLwc =
      LWC, '(', stringVal, ')'
    ;

stringFuncTrim =
      TRIM, '(', stringVal, ')'
    ;

stringFuncUpc =
```

```
      UPC, '(', stringVal, ')'
    ;

stringFuncSubstr =
      SUBSTR, '(', stringVal, c, intVal, [c, intVal], ')'
    ;

boolFuncBadgt =
      BADGT, '(', ')'
    ;

boolFuncExist =
      EXIST, '(', sLabel, ')'
    | EXIST, '(', saLabel, ')'
    | EXIST, '(', dLabel, ')'
    | EXIST, '(', daLabel, ')'
    | EXIST, '(', didLabel, ')'
    | EXIST, '(', fLabel, ')'
    | EXIST, '(', faLabel, ')'
    ;

boolFuncBadpt =
      BADPT, '(', ')'
    ;

boolFuncEof =
      EOF, '(', didLabel, ')'
    ;

boolFuncEoln =
      EOLN, '(', didLabel, ')'
    ;

intFuncAbs =
      ABS, '(', intVal, ')'
    ;

intFuncIndx =
      INDX, '(', stringVal, c, stringVal, ')'
    ;

intFuncInt =
      INT, '(', realVal, ')'
    ;

intFuncLen =
      LEN, '(', stringVal, ')'
    ;

intFuncMod =
      MOD, '(', intVal, c, intVal, ')'
    ;

intFuncNint =
      NINT, '(', realVal, ')'
    ;

intFuncOrd =
      ORD, '(', stringVal, ')'
    | ORD, '(', boolVal, ')'
```

```
    ;

intFuncPtdata =
      PTDATA, '(', faLabel, ')'
    ;

intFuncSensnotouch =
      SENSNOTOUCH, '(', ')'
    ;

intFuncSign =
      SIGN, '(', intVal, c, intVal, ')'
    ;

intFuncSiltch =
      SILTCH, '(', ')'
    ;

realFuncAbs =
      ABS, '(', realVal, ')'
    ;

realFuncCos =
      COS, '(', rentVal, ')'
    ;

realFuncDble =
      DBLE, '(', rentVal, ')'
    ;

realFuncDtor =
      DTOR, '(', rentVal, ')'
    ;

realFuncExp =
      EXP, '(', rentVal, ')'
    ;

realFuncLn =
      LN, '(', rentVal, ')'
    ;

realFuncLog =
      LOG, '(', rentVal, ')'
    ;

realFuncMn =
      MN, '(', rentVal, c, rentList, ')'
    ;

realFuncMod =
      MOD, '(', realVal, c, realVal, ')'
    ;

realFuncMx =
      MX, '(', rentVal, c, rentList, ')'
    ;

realFuncQtemp =
      QTEMP, '(', stringVal, ')'
```

```
    ;

realFuncRand =
      RAND, '(', [rentVal], ')'
    ;

realFuncRl =
      RL, '(', rentVal, ')'
    ;

realFuncRtod =
      RTOD, '(', rentVal, ')'
    ;

realFuncSign =
      SIGN, '(', realVal, c, realVal, ')'
    ;

realFuncSin =
      SIN, '(', rentVal, ')'
    ;

realFuncSqrt =
      SQRT, '(', rentVal, ')'
    ;

realFuncTan =
      TAN, '(', rentVal, ')'
    ;

realFuncVal =
      VAL, '(', stringVal, ')'
    ;

realFuncVdot =
      VDOT, '(', vectorVal, c, vectorVal, ')'
    ;

realFuncVecX =
      VECX, '(', vectorVal, ')'
    ;

realFuncVecY =
      VECY, '(', vectorVal, ')'
    ;

realFuncVecZ =
      VECZ, '(', vectorVal, ')'
    ;

realFuncVmag =
      VMAG, '(', vectorVal, ')'
    ;

realFuncAcos =
      ACOS, '(', rentVal, ')'
    ;

realFuncAsin =
      ASIN, '(', rentVal, ')'
```

```
   ;

realFuncAtan =
      ATAN, '(', rentVal, ')'
   ;

realFuncAtan2 =
      ATAN2, '(', rentVal, c, rentVal, ')'
   ;

vectorFuncVcart =
      VCART, '(', rentVal, c, rentVal, c, rentVal, ')'
   ;

vectorFuncVcross =
      VCROSS, '(', vectorVal, c, vectorVal, ')'
   ;

vectorFuncVmcs =
      VMCS, '(', vectorVal, ')'
   ;

vectorFuncVpcs =
      VPCS, '(', vectorVal, ')'
   ;

vectorFuncVpol =
      VPOL, '(', rentVal, c, angle, c, rentVal, ')'
   ;

vectorFuncVunit =
      VUNIT, '(', vectorVal, ')'
   ;

(*============================================================*)
(* label definitions *)
(*============================================================*)

labelNameConst =
      '(', LblName, ')'
   ;

labelName =
      labelNameConst
   | '(', '@', stringVar, ')'
   ;

datumLabel =
      dLabel
   | daLabel
   ;

featureLabel =
      fLabel
   | faLabel
   ;

qisLabel =
      ccLabel
   | ciLabel
```

```
    | csLabel
    | diLabel
    | dsLabel
    | dvLabel
    | fiLabel
    | fsLabel
    | liLabel
    | mdLabel
    | opLabel
    | pcLabel
    | plLabel
    | pnLabel
    | prLabel
    | psLabel
    | pvLabel
    | qLabel
    | tlLabel
    ;

sensorLabel =
      sLabel
    | saLabel
    ;

toleranceLabel =
      tLabel
    | taLabel
    ;

ccLabel =
      CC, labelName
    ;

ciLabel =
      CI, labelName
    ;

crLabel =
      CR, labelName
    ;

crLabelConst =
      CR, labelNameConst
    ;

csLabel =
      CS, labelName
    ;

czLabel =
      CZ, labelName
    ;

dLabel =
      D, labelName
    ;

daLabel =
      DA, labelName
    ;
```

```
datLabel =
      DAT, labelName
    ;

dattrgLabel =
      DATTRG, labelName
    ;

diLabel =
      DI, labelName
    ;

didLabel =
      DID, labelName
    ;

drLabel =
      DR, labelName
    ;

dsLabel =
      DS, labelName
    ;

dvLabel =
      DV, labelName
    ;

fLabel =
      F, labelName
    ;

faLabel =
      FA, labelName
    ;

fiLabel =
      FI, labelName
    ;

fsLabel =
      FS, labelName
    ;

gLabel =
      G, labelName
    ;

gsaLabel =
      GSA, labelName
    ;

kcLabel =
      KC, labelName
    ;

kcaLabel =
      KCA, labelName
    ;
```

```
liLabel =
      LI, labelName
    ;

mLabel =
      M, labelName
    ;

mLabelConst =
      M, labelNameConst
    ;

maLabel =
      MA, labelName
    ;

mdLabel =
      MD, labelName
    ;

opLabel =
      OP, labelName
    ;

pLabel =
      P, labelName
    ;

pcLabel =
      PC, labelName
    ;

plLabel =
      PL, labelName
    ;

pnLabel =
      PN, labelName
    ;

prLabel =
      PR, labelName
    ;

psLabel =
      PS, labelName
    ;

pvLabel =
      PV, labelName
    ;

qLabel =
      Q, labelName
    ;

rLabel =
      R, labelName
    ;
```

```
rmLabel =
      RM, labelName
    ;

rtLabel =
      RT, labelName
    ;

sLabel =
      S, labelName
    ;

saLabel =
      SA, labelName
    ;

seLabel =
      SE, labelName
    ;

sgLabel =
      SG, labelName
    ;

sgsLabel =
      SGS, labelName
    ;

srLabel =
      SR, labelName
    ;

sraLabel =
      SRA, labelName
    ;

ssLabel =
      SS, labelName
    ;

stLabel =
      ST, labelName
    ;

swLabel =
      SW, labelName
    ;

sxLabel =
      SX, labelName
    ;

tLabel =
      T, labelName
    ;

taLabel =
      TA, labelName
    ;
```

```
thLabel =
      TH, labelName
    ;

tlLabel =
      TL, labelName
    ;

uLabel =
      U, labelName
    ;

vLabel =
      V, labelName
    ;

vaLabel =
      VA, labelName
    ;

vfLabel =
      VF, labelName
    ;

vlLabel =
      VL, labelName
    ;

vwLabel =
      VW, labelName
    ;

jumpLabel =
      labelName
    ;

(*============================================================*)
(* miscellaneous definitions *)
(*============================================================*)

param =
      anyVal
    ;

paramList =
      [paramList, c], param
    ;

stringList =
      [stringList, c], stringVal
    ;

intList =
      [intList, c], intVal
    ;

leftRight =
      LEFT
    | RIGHT
```

```
    ;

rentList =
      [rentList, c], rentVal
    ;

featureList =
      [featureList, c], featureLabel
    ;

featureNominalList =
      [featureNominalList, c], fLabel
    ;

featureActualList =
      [featureActualList, c], faLabel
    ;

noParseStmList =
      [noParseStmList], noParseStm
    ;

dattrgList =
      [dattrgList, c], dattrgLabel
    ;

saLabelList =
      [saLabelList, c], saLabel
    ;

stackElementList =
      [stackElementList, c], stackElement
    ;

tLabelList =
      [tLabelList, c], tLabel
    ;

taLabelList =
      [taLabelList, c], taLabel
    ;

pointRange =
      '[', intVal, [c, intVal], ']'
    ;

indexedFeature =
      featureLabel, pointRange
    ;

indexedFeatureList =
      [indexedFeatureList, c], indexedFeature
    ;

versionTag =
      RealString
    ;

angle =
      rentVal
```

```
    | angleDms
    ;

angleDms =
      IntString, ':', IntString, [':', secondsString]
    ;

noParseStm =
      CharString, #
    ;

secondsString =
      RealString
    | IntString
    ;

triplet =
      rentVal, c, rentVal, c, rentVal
    ;

point =
      triplet
    ;

vector =
      triplet
    ;

matrix =
      triplet, c, triplet, c, triplet, c, triplet
    ;

rwVar =
      stringVar
    | boolVar
    | intVar
    | realVar
    | vectorVar
    ;

rwFormat =
      ':', intVal, [':', intVal ]
    ;

device =
      PRINT
    | TERM
    | COMM
    | STOR
    ;

state =
      ON
    | OFF
    ;

coordType =
      CART
    | POL
    ;
```

```
typePoint =
    coordType, c, point
    ;

axis =
      XAXIS
    | YAXIS
    | ZAXIS
    ;

orig =
      XORIG
    | YORIG
    | ZORIG
    ;

dir =
      posDir
    | negDir
    ;

posDir =
      XDIR
    | YDIR
    | ZDIR
    ;

negDir =
      MINUSXDIR
    | MINUSYDIR
    | MINUSZDIR
    ;

plan =
      XYPLAN
    | YZPLAN
    | ZXPLAN
    ;

matDir =
      INNER
    | OUTER
    ;

flatRoundOpen =
      FLAT
    | ROUND
    | OPEN
    ;

radiusSpec =
      MAJOR
    | MINOR
    ;

stackElement =
      ALGOR
    | DME
    | DATSET
```

```
    | REPORT
    ;

rotType =
      ROTTOT
    | ROTORG
    | ROTNUL
    ;

rotDir =
      rotOrient
    | SHORT
    ;

rotOrient =
      CW
    | CCW
    ;

rotAbs =
      ABSL, c, rotDir
    ;

rotIncr =
      INCR, c, rotOrient
    ;

(*=============================================================*)
(* DMIS statements *)
(*=============================================================*)

(*-------------------------------------------------------------*)
(* jump target *)
(*-------------------------------------------------------------*)

jumpStm =
      labelNameConst, #
    ;

(*-------------------------------------------------------------*)
(* ACLRAT *)
(*-------------------------------------------------------------*)

aclratStm =
      ACLRAT, '/', aclratMinor, #
    ;

aclratMinor =
      aclratMeas
    | aclratPos
    | aclratRot
    | aclratScan
    | aclratHedRot
    | aclratHedMeas
    | aclratHedScan
    ;

aclratMeas =
      MESACL, c, aclratLinSpec
    ;
```

```
aclratPos =
      POSACL, c, aclratLinSpec
    ;

aclratRot =
      ROTACL, c, aclratAngSpec
    ;

aclratScan =
      SCNACL, c, aclratLinSpec
    ;

aclratHedRot =
      HEDROTACL, c, aclratLinSpec
    ;

aclratHedMeas =
      HEDMESACL, c, aclratLinSpec
    ;

aclratHedScan =
      HEDSCNACL, c, aclratLinSpec
    ;

aclratAngSpec =
      aclratAngular
    | aclratDef
    ;

aclratLinSpec =
      aclratLinear
    | aclratDef
    ;

aclratLinear =
      MPMM, c, rentVal
    | MMPSS, c, rentVal
    | IPMM, c, rentVal
    | IPSS, c, rentVal
    ;

aclratAngular =
      RPMM, c, rentVal
    ;

aclratDef =
      PCENT, c, rentVal
    | HIGH
    | LOW
    | DEFALT
    ;

(*-------------------------------------------------------------*)
(* ALGDEF *)
(*-------------------------------------------------------------*)

algdefStm =
      vaLabel, '=', ALGDEF, '/', algdefMinor, #
    ;
```

```
algdefMinor =
      CODE, c, intVal
    | stringVal, [c, paramList]
    ;

(*-------------------------------------------------------------*)
(* ASSIGN *)
(*-------------------------------------------------------------*)

assignStm =
      assignString, #
    | assignBool, #
    | assignInt, #
    | assignReal, #
    | assignVector, #
    ;

assignString =
      stringVar, '=', ASSIGN, '/', stringVal
    ;

assignBool =
      boolVar, '=', ASSIGN, '/', boolVal
    ;

assignInt =
      intVar, '=', ASSIGN, '/', rentVal
    ;

assignReal =
      realVar, '=', ASSIGN, '/', rentVal
    ;

assignVector =
      vectorVar, '=', ASSIGN, '/', vectorVal
    ;

(*-------------------------------------------------------------*)
(* BADTST *)
(*-------------------------------------------------------------*)

badtstStm =
      BADTST, '/', state, #
    ;

(*-------------------------------------------------------------*)
(* BOUND *)
(*-------------------------------------------------------------*)

boundStm =
      BOUND, '/', boundMinor, #
    ;

boundMinor =
      boundFeat
    | boundTol
    ;

boundFeat =
```

```
      fLabel, c, featureList
   ;

boundTol =
      tLabel, c, featureList
   ;

(*-------------------------------------------------------------*)
(* CALIB *)
(*-------------------------------------------------------------*)

calibSensStm =
      CALIB, '/', calibSensMinor, #
   ;

calibRtabStm =
      CALIB, '/', calibRtabMinor, #
   ;

calibMasterStm =
      CALIB, '/', calibMasterMinor, #
   ;

calibSensMinor =
      SENS, c, sLabel, c, calibSensSpec
   | SENS, c, RECALIB
   ;

calibSensSpec =
      featureLabel, c, intVal
   | featureLabel, c, stringVal, [c, intVal]
   ;

calibRtabMinor =
      RTAB, c, rtLabel, c, calibRtabSpec
   | RTAB, c, RECALIB
   ;

calibRtabSpec =
      fLabel, c, intVal
   | faLabel, c, faLabel
   ;

calibMasterMinor =
      MASTER, c, stringVal
   ;

(*-------------------------------------------------------------*)
(* CALL *)
(*-------------------------------------------------------------*)

callStm =
      CALL, '/', callMinor, #
   ;

callMinor =
      callMacro
   | callModule
   | callRoutine
   | callProgram
```

```
    ;

callMacro =
      [EXTERN, c, DMIS, c], mLabel, [c, paramList]
    ;

callModule =
      EXTERN, c, DMIS, c, stringVal
    ;

callRoutine =
      EXTERN, c, DME, c, stringVal, [c, callType], [c, paramList]
    ;

callProgram =
      EXTERN, c, SYS, c, stringVal, [c, callType], [c, paramList]
    ;

callType =
      WAIT
    | CONT
    | ATTACH
    ;

(*-------------------------------------------------------------*)
(* CASE *)
(*-------------------------------------------------------------*)

caseStm =
       CASE, '/', [sign], intConst, #
     | CASE, '/', stringConst, #
     ;

plusSign =
      '+'
     ;

minusSign =
      '-'
     ;

sign =
       plusSign
     | minusSign
     ;

(*-------------------------------------------------------------*)
(* CLMPID *)
(*-------------------------------------------------------------*)

clmpidStm =
      ciLabel, '=', CLMPID, '/', stringVal, #
    ;

(*-------------------------------------------------------------*)
(* CLMPSN *)
(*-------------------------------------------------------------*)

clmpsnStm =
      csLabel, '=', CLMPSN, '/', stringVal, #
```

```
   ;

(*-------------------------------------------------------------*)
(* CLOSE *)
(*-------------------------------------------------------------*)

closeStm =
      CLOSE, '/', closeMinor, #
   ;

closeMinor =
      didLabel
    | didLabel, c, KEEP
    | didLabel, c, DELETE
    | didLabel, c, END
    ;

(*-------------------------------------------------------------*)
(* CMPNTGRP *)
(*-------------------------------------------------------------*)

cmpntgrpStm =
      sgLabel, '=', CMPNTGRP, '/', cmpntgrpMinor, #
   ;

cmpntgrpMinor =
      BUILD, c, cmpntgrpSpecList
   ;

cmpntgrpSpecList =
      [cmpntgrpSpecList, c], cmpntgrpSpec
   ;

cmpntgrpSpec =
      sgLabel
    | swLabel
    | sxLabel
    | rmLabel
    ;

(*-------------------------------------------------------------*)
(* CNFRMRUL *)
(*-------------------------------------------------------------*)

cnfrmrulStm =
      drLabel, '=', CNFRMRUL, '/', cnfrmrulVar1, #
   ;

cnfrmrulVar1 =
      RULE, c, intVal
    | stringVal, [c, paramList ]
    ;

(*-------------------------------------------------------------*)
(* CONST *)
(*-------------------------------------------------------------*)

constStm =
      CONST, '/', constMinor, #
   ;
```

```
constMinor =
      constArc
    | constCircle
    | constCompound
    | constCone
    | constCylndr
    | constCparln
    | constEllips
    | constEdgept
    | constGeom
    | constGcurve
    | constGsurf
    | constLine
    | constParpln
    | constPatern
    | constPlane
    | constPoint
    | constRctngl
    | constSgage
    | constSpart
    | constSphere
    | constSympln
    | constTorus
    ;

constArc =
      ARC, c, fLabel, c, bfConst
    | ARC, c, fLabel, c, projctConst
    | ARC, c, fLabel, c, trConst
    ;

constCircle =
      CIRCLE, c, fLabel, c, bfConst
    | CIRCLE, c, fLabel, c, coneConst
    | CIRCLE, c, fLabel, c, intofConst
    | CIRCLE, c, fLabel, c, projctConst
    | CIRCLE, c, fLabel, c, tantoConst
    | CIRCLE, c, fLabel, c, trConst
    | CIRCLE, c, fLabel, c, retrieve2
    ;

constCompound =
      COMPOUND, c, fLabel, c, BUILD
    ;

constCone =
      CONE, c, fLabel, c, bfConst
    | CONE, c, fLabel, c, trConst
    ;

constCparln =
      CPARLN, c, fLabel, c, bfConst
    | CPARLN, c, fLabel, c, projctConst
    | CPARLN, c, fLabel, c, trConst
    | CPARLN, c, fLabel, c, retrieve2
    ;

constCylndr =
      CYLNDR, c, fLabel, c, bfConst
```

```
    | CYLNDR, c, fLabel, c, trConst
    | CYLNDR, c, fLabel, c, retrieve1
    ;

constEllips =
      ELLIPS, c, fLabel, c, bfConst
    | ELLIPS, c, fLabel, c, intofConst
    | ELLIPS, c, fLabel, c, projctConst
    | ELLIPS, c, fLabel, c, trConst
    ;

constEdgept =
      EDGEPT, c, fLabel, c, retrieve4
    ;

constGeom =
      GEOM, c, fLabel, c, nearptConst
    ;

constGcurve =
      GCURVE, c, fLabel, c, bfConst
    | GCURVE, c, fLabel, c, projctConst
    | GCURVE, c, fLabel, c, trConst
    ;

constGsurf =
      GSURF, c, fLabel, c, bfConst
    | GSURF, c, fLabel, c, trConst
    ;

constLine =
      LINE, c, fLabel, c, bfConst
    | LINE, c, fLabel, c, intofConst
    | LINE, c, fLabel, c, midliConst
    | LINE, c, fLabel, c, offsetConst
    | LINE, c, fLabel, c, partoConst
    | LINE, c, fLabel, c, perptoConst
    | LINE, c, fLabel, c, projliConst
    | LINE, c, fLabel, c, tantoConst
    | LINE, c, fLabel, c, trConst
    ;

constParpln =
      PARPLN, c, fLabel, c, bfConst
    ;

constPatern =
      PATERN , c , fLabel , c , trConst
    | PATERN, c, fLabel, c, BUILD
    ;

constPlane =
      PLANE, c, fLabel, c, bfConst
    | PLANE, c, fLabel, c, midplConst
    | PLANE, c, fLabel, c, offsetConst
    | PLANE, c, fLabel, c, partoConst
    | PLANE, c, fLabel, c, perptoConst
    | PLANE, c, fLabel, c, tantoConstPlane
    | PLANE, c, fLabel, c, trConst
    ;
```

```
constPoint =
      POINT, c, fLabel, c, cogConst
    | POINT, c, fLabel, c, curveConst
    | POINT, c, fLabel, c, extremConst
    | POINT, c, fLabel, c, intofConst
    | POINT, c, fLabel, c, midptConst
    | POINT, c, fLabel, c, moveptConst
    | POINT, c, fLabel, c, pierceConst
    | POINT, c, fLabel, c, projptConst
    | POINT, c, fLabel, c, trConst
    | POINT, c, fLabel, c, vertexConst
    | POINT, c, fLabel, c, retrieve1
    ;

retrieve1 =
      RETRIEVE, c, rentVal, c, featureActualList
    ;

retrieve2 =
      RETRIEVE, c, rentVal, c, rentVal, c, featureActualList
    ;

retrieve4 =
      RETRIEVE, c, rentVal, c, rentVal, c, rentVal, c,
      rentVal, c, featureActualList
    ;

retrieve2b =
      RETRIEVE, c, rentVal, [c, rentVal, c, rentVal, c,
      rentVal], c, featureActualList
    ;

constRctngl =
      RCTNGL, c, fLabel, c, bfConst
    | RCTNGL, c, fLabel, c, trConst
    ;

constSgage =
      SGAGE, c, seLabel, c, sgageConst
    ;

constSpart =
      SPART, c, stLabel, c, spartConst
    ;

constSphere =
      SPHERE, c, fLabel, c, bfConst
    | SPHERE, c, fLabel, c, trConst
    | SPHERE, c, fLabel, c, retrieve2b
    ;

constSympln =
      SYMPLN, c, fLabel, c, bfConst
    ;

constTorus =
      TORUS, c, fLabel, c, bfConst
    | TORUS, c, fLabel, c, trConst
    ;
```

```
bfConst =
      BF, c, featureList
    | BF, c, indexedFeatureList
    ;

cogConst =
      COG, c, featureList
    ;

coneConst =
      CONE, c, DIAM, c, rentVal, c, faLabel
    | CONE, c, DIST, c, rentVal, c, faLabel
    ;

curveConst =
      CURVE, c, faLabel, c, featureLabel
    ;

extremConst =
      EXTREM, c, MIN, c, faLabel, c, extremConstDir
    | EXTREM, c, MAX, c, faLabel, c, extremConstDir
    ;

extremConstDir =
      extremConstAxial
    | extremConstVectorial
    | extremConstFeature
    | extremConstRadial
    ;

extremConstAxial =
      posDir
    ;

extremConstVectorial =
      VEC, c, vector
    ;

extremConstFeature =
      featureLabel
    ;

extremConstRadial =
      RADIAL
    ;

intofConst =
      INTOF, c, faLabel, c, featureLabel
    ;

midliConst =
      MIDLI, c, faLabel, c, featureLabel
    ;

midplConst =
      MIDPL, c, faLabel, c, featureLabel
    ;

midptConst =
```

```
      MIDPT, c, faLabel, c, featureLabel
    ;

moveptConst =
      MOVEPT, c, faLabel, c, vector
    | MOVEPT, c, faLabel, c, featureLabel, c, rentVal
    ;

nearptConst =
      NEARPT, c, faLabel
    ;

offsetConst =
      OFFSET, c, featureList
    ;

partoConst =
      PARTO, c, faLabel, c, THRU, c, featureLabel
    | PARTO, c, fLabel, c, THRU, c, faLabel
    ;

perptoConst =
      PERPTO, c, faLabel, c, THRU, c, featureLabel
    | PERPTO, c, fLabel, c, THRU, c, faLabel
    ;

pierceConst =
      PIERCE, c, faLabel, c, featureLabel
    ;

projctConst =
      PROJCT, c, faLabel, [c, featureLabel]
    ;

projliConst =
      PROJLI, c, faLabel, [c, featureLabel]
    ;

projptConst =
      PROJPT, c, faLabel, [c, featureLabel]
    ;

sgageConst =
      featureNominalList
    ;

spartConst =
      featureActualList
    ;

tantoConst =
      TANTO, c, faLabel, [c, THRU], c, featureLabel
    | TANTO, c, fLabel, c, THRU, c, faLabel
    ;

tantoConstPlane =
      TANTO, c, faLabel, c, THRU, c, featureLabel
    | TANTO, c, fLabel, c, THRU, c, faLabel
    ;
```

```
trConst =
      TR, c, faLabel, [c, datumLabel]
   ;

vertexConst =
      VERTEX, c, faLabel
   ;

(*-------------------------------------------------------------*)
(* CRGDEF *)
(*-------------------------------------------------------------*)

crgdefStm =
      crLabelConst, '=', CRGDEF, #
   | crLabelConst, '=', CRGDEF, '/', crgdefMinor, #
   ;

crgdefMinor =
      pointVec, c, vector, c, vector, c, vector, c, vector
   ;

(*-------------------------------------------------------------*)
(* CRMODE *)
(*-------------------------------------------------------------*)

crmodeStm =
      CRMODE, '/', crmodeMinor, #
   ;

crmodeMinor =
      SEQNTL
   | SIMUL
   | SYNC
   ;

(*-------------------------------------------------------------*)
(* CROSCL *)
(*-------------------------------------------------------------*)

crosclStm =
      CROSCL, '/', state, #
   ;

(*-------------------------------------------------------------*)
(* CRSLCT *)
(*-------------------------------------------------------------*)

crslctStm =
      CRSLCT, '/', crslctMinor, #
   ;

crslctMinor =
      crLabelConst
   | ALL
   ;

(*-------------------------------------------------------------*)
(* CUTCOM *)
(*-------------------------------------------------------------*)
```

```
cutcomStm =
      ccLabel, '=', CUTCOM, '/', cutcomMinor, #
    ;

cutcomMinor =
      cutcomAdjust
    | cutcomParam
    | cutcomMatrix
    | cutcomUserdf
    ;

cutcomAdjust =
      mdLabel, c, ADJUST, c, tlLabel, c, leftRight, c, plan, c, rentVal
    ;

cutcomParam =
      mdLabel, c, PARAM, c, pointVec
    ;

cutcomMatrix =
      mdLabel, c, MATRIX, c, matrix
    ;

cutcomUserdf =
      mdLabel, c, USERDF, c, stringVal
    ;

(*-------------------------------------------------------------*)
(* CZONE *)
(*-------------------------------------------------------------*)

czoneStm =
      czLabel, '=', CZONE, #
    ;

(*-------------------------------------------------------------*)
(* CZSLCT *)
(*-------------------------------------------------------------*)

czslctStm =
      CZSLCT, '/', czLabel, c, state, #
    ;

(*-------------------------------------------------------------*)
(* DATDEF *)
(*-------------------------------------------------------------*)

datdefStm =
      DATDEF, '/', datdefMinor, #
    ;

datdefMinor =
      featureLabel, c, datLabel
    | dattrgList, c, fLabel, c, datLabel
    ;

(*-------------------------------------------------------------*)
(* DATSET *)
(*-------------------------------------------------------------*)
```

```
datsetStm =
      dLabel, '=', DATSET, '/', datsetMinor, #
    ;

datsetMinor =
      datsetMcs
    | datsetDats
    | datsetMatrix
    | datsetDrf
    ;

datsetMcs =
      MCS
    ;

datsetMatrix =
      TRMATX, c, matrix
    ;

datsetDats =
      2*[datsetSpec, c], datsetSpec
    ;

datsetDrf =
      DRF, c, taLabel, [c, tier], c, dir, c, dir
    ;

tier =
      UPTIER
    | LOTIER
    ;

datsetSpec =
      datLabel, c, orig, 2*[c, orig]
    | datLabel, c, dir, 3*[c, orig]
    ;

(*-------------------------------------------------------------*)
(* DATTRG *)
(*-------------------------------------------------------------*)

dattrgdefStm =
      DATTRGDEF, '/', dattrgMinor, #
    ;

dattrgMinor =
      featureList, c, dattrgLabel
    ;

(*-------------------------------------------------------------*)
(* DECL *)
(*-------------------------------------------------------------*)

declStm =
      DECL, '/', declMinor, #
    ;

declMinor =
      [declScope, c], declType, c, declVarList
    ;
```

```
declVarList =
      [declVarList, c], declVar
    ;

declScope =
      COMMON
    | GLOBAL
    | LOCAL
    ;

declType =
      BOOL
    | INTGR
    | LONG
    | REAL
    | DOUBLE
    | CHAR, c, intVal
    | VECTOR
    ;

declVar =
      DeclVarName, ['[', declIndicesList, ']']
    ;

declIndicesList =
      [declIndicesList, c], intConst
    ;

(*-------------------------------------------------------------*)
(* DECPL *)
(*-------------------------------------------------------------*)

decplStm =
      DECPL, '/', decplMinor, #
    ;

decplMinor =
      decplAll
    | decplList
    ;

decplAll =
      ALL, c, decplNdigits
    ;

decplList =
      [decplList, c], decplSelection
    ;

decplNdigits =
      DEFALT
    | intVal
    ;

decplSelection =
      ANGLE, c, decplNdigits
    | DIST, c, decplNdigits
    | HUMID, c, decplNdigits
    | DEV, c, decplNdigits
```

```
    | TEMP, c, decplNdigits
    | VEC, c, decplNdigits
    ;

(*-------------------------------------------------------------*)
(* DELETE *)
(*-------------------------------------------------------------*)

deleteStm =
      DELETE, '/', deleteMinor, #
    ;

deleteMinor =
      deleteDatum
    | deleteSensor
    | deleteFeature
    | deleteRotaryTable
    | deleteAllSensors
    ;

deleteDatum =
      datumLabel, [c, didLabel]
    ;

deleteSensor =
      sensorLabel, [c, didLabel]
    ;

deleteFeature =
      faLabel, [c, didLabel]
    ;

deleteRotaryTable =
      rtLabel, [c, didLabel]
    ;

deleteAllSensors =
      ALLSA, [c, EXCEPT, c, saLabelList], [c, didLabel]
    ;

(*-------------------------------------------------------------*)
(* DEVICE *)
(*-------------------------------------------------------------*)

deviceStm =
      didLabel, '=', DEVICE, '/', deviceMinor, #
    ;

deviceMinor =
      device, c, stringVal
    | INCR, c, stringVal
    ;

(*-------------------------------------------------------------*)
(* DFTCAS *)
(*-------------------------------------------------------------*)

dftcasStm =
      DFTCAS, #
    ;
```

```
(*-------------------------------------------------------------*)
(* DISPLY *)
(*-------------------------------------------------------------*)

displyStm =
      DISPLY, '/', displyOff, #
    | DISPLY, '/', displySpecList, #
    ;

displyOff =
      OFF
    ;

displySpecList =
      [displySpecList, c], displySpecItem
    ;

displySpecItem =
      device, c, DMIS, [c, vLabel]
    | device, c, vLabel
    ;

(*-------------------------------------------------------------*)
(* DMEHW *)
(*-------------------------------------------------------------*)

dmehwStm =
      DMEHW, '/', dmehwMinor, #
    ;

dmehwMinor =
      CONTIN
    | PAUSE
    | SINGLE
    | AUTO
    | jointConf
    ;

jointConf =
      JOINTCONFIG, [c, rightyLefty], [c, aboveBelow], [c, flipNoFlip],
           [c, stringList]
    ;

rightyLefty =
      RIGHTY
    | LEFTY
    ;

aboveBelow =
      ABOVE
    | BELOW
    ;

flipNoFlip =
      FLIP
    | NOFLIP
    ;

(*-------------------------------------------------------------*)
```

```
(* DMEID *)
(*-------------------------------------------------------------*)

dmeidStm =
      diLabel, '=', DMEID, '/', stringVal, #
    ;

(*-------------------------------------------------------------*)
(* DMESW *)
(*-------------------------------------------------------------*)

dmeswStm =
      DMESW, '/', dmeswMinor, #
    ;

dmeswMinor =
      dmeswComand
    | dmeswDelay
    | dmeswSwitch
    ;

dmeswComand =
      COMAND, c, stringVal
    ;

dmeswDelay =
      DELAY, c, intVal
    ;

dmeswSwitch =
      CONTIN
    | PAUSE
    ;

(*-------------------------------------------------------------*)
(* DMESWI *)
(*-------------------------------------------------------------*)

dmeswiStm =
      dsLabel, '=', DMESWI, '/', stringVal, #
    ;

(*-------------------------------------------------------------*)
(* DMESWV *)
(*-------------------------------------------------------------*)

dmeswvStm =
      dvLabel, '=', DMESWV, '/', stringVal, #
    ;

(*-------------------------------------------------------------*)
(* DMIS OFF*)
(*-------------------------------------------------------------*)

dmisOffStm =
      DMIS, '/', OFF, #
    ;

(*-------------------------------------------------------------*)
(* DMIS ON*)
```

```
(*-------------------------------------------------------------*)

dmisOnStm =
      DMIS, '/', ON, #
    ;

(*-------------------------------------------------------------*)
(* DMISMD *)
(*-------------------------------------------------------------*)

dmismdStm =
      DMISMD, '/', stringConst, c, versionTag, [c, conformItemList], #
    ;

(*-------------------------------------------------------------*)
(* DMISMN *)
(*-------------------------------------------------------------*)

dmismnStm =
      DMISMN, '/', stringConst, c, versionTag, [c, conformItemList], #
    ;

conformItemList =
      [conformItemList, c], conformItem
    ;

conformItem =
      conformType, c, IntString
    ;

conformType =
      CT
    | FX
    | IP
    | MC
    | MU
    | PM
    | QI
    | RY
    | SF
    | TW
    ;

(*-------------------------------------------------------------*)
(* DO *)
(*-------------------------------------------------------------*)

doStm =
      DO, '/', doMinor, #
    ;

doMinor =
      intVar, c, intVal, c, intVal, [c, intVal]
    ;

(*-------------------------------------------------------------*)
(* ELSE *)
(*-------------------------------------------------------------*)

elseStm =
```

```
      ELSE, #
   ;

(*-------------------------------------------------------------*)
(* ENDCAS *)
(*-------------------------------------------------------------*)

endcasStm =
      ENDCAS, #
   ;

(*-------------------------------------------------------------*)
(* ENDDO *)
(*-------------------------------------------------------------*)

enddoStm =
      ENDDO, #
   ;

(*-------------------------------------------------------------*)
(* ENDFIL *)
(*-------------------------------------------------------------*)

endfilStm =
      ENDFIL, #
   ;

(*-------------------------------------------------------------*)
(* ENDGO *)
(*-------------------------------------------------------------*)

endgoStm =
      ENDGO, #
   ;

(*-------------------------------------------------------------*)
(* ENDIF *)
(*-------------------------------------------------------------*)

endifStm =
      ENDIF, #
   ;

(*-------------------------------------------------------------*)
(* ENDMAC *)
(*-------------------------------------------------------------*)

endmacStm =
      ENDMAC, #
   ;

(*-------------------------------------------------------------*)
(* ENDMES *)
(*-------------------------------------------------------------*)

endmesStm =
      ENDMES, #
   ;

(*-------------------------------------------------------------*)
```

```
(* ENDSEL *)
(*-------------------------------------------------------------*)

endselStm =
      ENDSEL, #
    ;

(*-------------------------------------------------------------*)
(* ENDSIMREQT *)
(*-------------------------------------------------------------*)

endsimreqtStm =
      ENDSIMREQT, #
    ;

(*-------------------------------------------------------------*)
(* ENDXTN *)
(*-------------------------------------------------------------*)

endxtnStm =
      ENDXTN, #
    ;

(*-------------------------------------------------------------*)
(* EQUATE *)
(*-------------------------------------------------------------*)

equateStm =
      EQUATE, '/', equateMinor, #
    ;

equateMinor =
      daLabel, c, daLabel
    | daLabel, c, CADCS, c, equateCadcs
    ;

equateCadcs =
      didLabel, c, matrix
    | didLabel, c, stringVal
    ;

(*-------------------------------------------------------------*)
(* ERROR *)
(*-------------------------------------------------------------*)

errorStm =
      ERROR, '/', errorMinor, #
    ;

errorMinor =
      jumpLabel, c, errorCondition
    | AUTO, c, errorCondition
    | OFF
    ;

errorCondition =
      ALL
    | ILLEGALTOUCH
    | NOTOUCH
    | intVal
```

```
   ;

(*-------------------------------------------------------------*)
(* EVAL *)
(*-------------------------------------------------------------*)

evalStm =
      EVAL, '/', evalMinor, #
   ;

evalMinor =
      evalFeat
   | evalDat
   | evalKeychar
   ;

evalFeat =
      faLabel, c, tLabelList
   | faLabel, c, fLabel, c, tLabel
   | fLabel, c, faLabel, c, tLabel
   | faLabel, c, faLabel, c, tLabel
   ;

evalDat =
      datLabel, c, faLabel, c, tLabel
   | faLabel, c, datLabel, c, tLabel
   ;

evalKeychar =
      kcLabel
   ;

(*-------------------------------------------------------------*)
(* EXTENS *)
(*-------------------------------------------------------------*)

extensStm =
      sxLabel, '=', EXTENS, '/', extensMinor, #
   ;

extensMinor =
      vector
   | VEC, c, vector, c, rentVal
   ;

(*-------------------------------------------------------------*)
(* EXTFIL *)
(*-------------------------------------------------------------*)

extfilStm =
      EXTFIL, '/', extfilMinor, #
   ;

extfilMinor =
      extfilDmis
   | extfilDme
   ;

extfilDmis =
      DMIS, c, stringVal
```

```
    ;

extfilDme =
      DME, c, stringVal
    ;

(*-------------------------------------------------------------*)
(* FEAT *)
(*-------------------------------------------------------------*)

featStm =
      fLabel,  '=', FEAT, '/', featMinor, #
    | faLabel, '=', FEAT, '/', featMinor, #
    | fLabel,  '=', FEAT, '/', featMinor1, #
    ;

featMinor =
      featArc
    | featCircle
    | featCone
    | featConradsegmnt
    | featCparln
    | featCylndr
    | featCylradsegmnt
    | featEdgept
    | featEllips
    | featElongcyl
    | featLine
    | featObject
    | featParpln
    | featPlane
    | featPoint
    | featRctngl
    | featRevsurf
    | featSphere
    | featSphradsegmnt
    | featSympln
    | featTorus
    | featTorradsegmnt
    ;

featMinor1 =
      featGcurve
    | featGeom
    | featGsurf
    | featPatern
    | featCompound
    ;

featArc =
      featArc1
    | featArc2
    ;

featArc1 =
      ARC, c, matDir, c, typePoint, c, vector, c, rentVal, c,
      angle, c, angle, [c, vector]
    ;

featArc2 =
```

```
      ARC, c, FOURPOINT, c, matDir, c, point, c, point, c, point, c, point
   ;

featCircle =
      CIRCLE, c, matDir, c, typePoint, c, vector, c, rentVal
   ;

featCone =
      CONE, c, matDir, c, typePoint, c, vector, c, angle
   ;

featConradsegmnt =
      CONRADSEGMNT, c, matDir, c, typePoint, c, point, c,
      rentVal, c, rentVal, c, vector, c, vector
   ;

featCparln =
      CPARLN, c, matDir, c, flatRoundOpen, c, typePoint, c,
      vector, c, vecRentRent
   ;

featCylndr =
      CYLNDR, c, matDir, c, typePoint, c, vector, c, rentVal, [c, rentVal]
   ;

featCylradsegmnt =
      CYLRADSEGMNT, c, matDir, c, typePoint, c, point, c,
      rentVal, c, vector, c, vector
   ;

featEllips =
      ELLIPS, c, matDir, c, typePoint, c, point, c,
      radiusSpec, c, vector, c, rentVal
   ;

featElongcyl =
      ELONGCYL, c, matDir, c, typePoint, c, vector, c,
      vecRentRent, [c, rentVal]
   ;

pointVec =
   point, c, vector
   ;

featGcurve =
      GCURVE, c, coordType, c, pointVec,
      [c, PTDATA, c, pointVec, c, pointVecList]
   ;

featGeom =
      GEOM, c, gLabel, c, coordType
   ;

featGsurf =
      GSURF, [c, coordType,
      [c, PTDATA, c, pointVec, c, pointVec, c, pointVecList]]
   ;

featLine =
      featLine1
```

```
    | featLine2
    ;

featLine1 =
      LINE, c, BND, c, typePoint, c, pointVec
    ;

featLine2 =
      LINE, c, UNBND, c, typePoint, c, vector, c, vector
    ;

featObject =
      OBJECT, c, paramList
    ;

featParpln =
      PARPLN, c, matDir, c, typePoint, c, pointVec, c, pointVec, c, rentVal
    | PARPLN, c, matDir, c, MIDPL, c, typePoint, c, vector, c, rentVal
    ;

featSympln =
      SYMPLN, c, matDir, c, typePoint, c, pointVec, c, pointVec, c, rentVal
    ;

featPatern =
      PATERN, c, fLabel, c, featureNominalList
    ;

featCompound =
      COMPOUND, c, AXIAL , c, typePoint, c, featCompoundEnd1
    | COMPOUND, c, PLANE , c, typePoint, c, featCompoundEnd1
    | COMPOUND, c, SPHERE, c, typePoint, c, featCompoundEnd2
    ;

featCompoundEnd1 =
      vector, c, fLabel, c, featureNominalList
    ;

featCompoundEnd2 =
      fLabel, c, featureNominalList
    ;

featRevsurf =
      REVSURF, c, matDir, c, typePoint, c, vector, c,
      coordType, c, point, c, pointList
    ;

pointList =
      [pointList, c], point
    ;

featPlane =
      PLANE, c, typePoint, c, vector
    ;

featPoint =
      POINT, c, typePoint, c, vector
    ;

featEdgept =
```

```
      EDGEPT, c, typePoint, c, vector, c, vector
   ;

featRctngl =
      RCTNGL, c, matDir, c, typePoint, c, vector, c, rentVal,
      c, vector, c, rentVal, c, vector, c, rentVal
   ;

featSphere =
      SPHERE, c, matDir, c, typePoint, c, rentVal, [c, vector, [c, angle]]
   ;

vecRentRent =
      vector, c, rentVal, c, rentVal
   ;

featSphradsegmnt =
      SPHRADSEGMNT, c, matDir, c, typePoint, c, rentVal, c,
      vecRentRent, c, vecRentRent
   ;

featTorus =
      TORUS, c, matDir, c, typePoint, c, vecRentRent
   ;

featTorradsegmnt =
      TORRADSEGMNT, c, matDir, c, typePoint, c, rentVal, c,
      rentVal, c, vecRentRent, c, vecRentRent
   ;

(*-------------------------------------------------------------*)
(* FEDRAT *)
(*-------------------------------------------------------------*)

fedratStm =
      FEDRAT, '/', fedratMinor, #
   ;

fedratMinor =
      fedratMeas
   | fedratPos
   | fedratRot
   | fedratScan
   | fedratHedRot
   | fedratHedMeas
   | fedratHedScan
   ;

fedratMeas =
      MESVEL, c, fedratLinSpec
   ;

fedratPos =
      POSVEL, c, fedratLinSpec
   ;

fedratRot =
      ROTVEL, c, fedratAngSpec
   ;
```

```
fedratScan =
      SCNVEL, c, fedratLinSpec
    ;

fedratHedRot =
      HEDROTVEL, c, fedratLinSpec
    ;

fedratHedMeas =
      HEDMESVEL, c, fedratLinSpec
    ;

fedratHedScan =
      HEDSCNVEL, c, fedratLinSpec
    ;

fedratAngSpec =
      fedratAngular
    | fedratDef
    ;

fedratLinSpec =
      fedratLinear
    | fedratDef
    ;

fedratLinear =
      MPM, c, rentVal
    | MMPS, c, rentVal
    | IPM, c, rentVal
    | IPS, c, rentVal
    ;

fedratAngular =
      RPM, c, rentVal
    ;

fedratDef =
      PCENT, c, rentVal
    | HIGH
    | LOW
    | DEFALT
    ;

(*-------------------------------------------------------------*)
(* FILDEF *)
(*-------------------------------------------------------------*)

fildefStm =
      vfLabel, '=', FILDEF, '/', fildefMinor, #
    ;

fildefMinor =
      CODE, c, intVal
    ;

(*-------------------------------------------------------------*)
(* FILNAM *)
(*-------------------------------------------------------------*)
```

```
filnamStm =
      FILNAM, '/', stringConst, c, versionTag, #
   ;

(*-------------------------------------------------------------*)
(* FINPOS *)
(*-------------------------------------------------------------*)

finposStm =
      FINPOS, '/', state, #
   ;

(*-------------------------------------------------------------*)
(* FIXTID *)
(*-------------------------------------------------------------*)

fixtidStm =
      fiLabel, '=', FIXTID, '/', stringVal, #
   ;

(*-------------------------------------------------------------*)
(* FIXTSN *)
(*-------------------------------------------------------------*)

fixtsnStm =
      fsLabel, '=', FIXTSN, '/', stringVal, #
   ;

(*-------------------------------------------------------------*)
(* FLY *)
(*-------------------------------------------------------------*)

flyStm =
      FLY, '/', flyMinor, #
   ;

flyMinor =
      OFF
   | rentVal
   ;

(*-------------------------------------------------------------*)
(* FROM *)
(*-------------------------------------------------------------*)

fromStm =
      FROM, '/', fromMinor, #
   ;

fromMinor =
      CART, c, point, [c, RAM]
   | DME
   | POL, c, point, [c, RAM]
   | SCALE
   | point, [c, RAM]
   ;

(*-------------------------------------------------------------*)
(* GEOALG *)
(*-------------------------------------------------------------*)
```

```
geoalgStm =
      GEOALG, '/', geoalgMinor, #
    ;

geoalgMinor =
      geoalgArc
    | geoalgCircle
    | geoalgCone
    | geoalgConradsegmnt
    | geoalgCparln
    | geoalgCylndr
    | geoalgCylradsegmnt
    | geoalgEllips
    | geoalgElongcyl
    | geoalgGcurve
    | geoalgGsurf
    | geoalgLine
    | geoalgObject
    | geoalgParpln
    | geoalgPlane
    | geoalgRctngl
    | geoalgRevsurf
    | geoalgSphere
    | geoalgSphradsegmnt
    | geoalgSympln
    | geoalgTorus
    | geoalgTorradsegmnt
    ;

geoalgArc =
      ARC, c, geoalgSpec1
    ;

geoalgCircle =
      CIRCLE, c, geoalgSpec2
    ;

geoalgCone =
      CONE, c, geoalgSpec2
    ;

geoalgConradsegmnt =
      CONRADSEGMNT, c, geoalgSpec2
    ;

geoalgCparln =
      CPARLN, c, geoalgSpec3
    ;

geoalgCylndr =
      CYLNDR, c, geoalgSpec2
    ;

geoalgCylradsegmnt =
      CYLRADSEGMNT, c, geoalgSpec2
    ;

geoalgEllips =
      ELLIPS, c, geoalgSpec1
```

```
    ;

geoalgElongcyl =
      ELONGCYL, c, geoalgSpec2
    ;

geoalgGcurve =
      GCURVE, c, geoalgSpec4
    ;

geoalgGsurf =
      GSURF, c, geoalgSpec5
    ;

geoalgLine =
      LINE, c, geoalgSpec1
    ;

geoalgObject =
      OBJECT, c, geoalgSpec3
    ;

geoalgParpln =
      PARPLN, c, geoalgSpec2
    ;

geoalgPlane =
      PLANE, c, geoalgSpec1
    ;

geoalgRctngl =
      RCTNGL, c, geoalgSpec1
    ;

geoalgRevsurf =
      REVSURF, c, geoalgSpec6
    ;

geoalgSphere =
      SPHERE, c, geoalgSpec2
    ;

geoalgSphradsegmnt =
      SPHRADSEGMNT, c, geoalgSpec2
    ;

geoalgSympln =
      SYMPLN, c, geoalgSpec2
    ;

geoalgTorus =
      TORUS, c, geoalgSpec2
    ;

geoalgTorradsegmnt =
      TORRADSEGMNT, c, geoalgSpec2
    ;

geoalgSpec1 =
      LSTSQR, [c, geoalgFilterSettings]
```

```
    | MINMAX, [c, geoalgFilterSettings]
    | DEFALT, [c, geoalgFilterSettings]
    | EXTERN, c, geoalgExternFunc, [c, paramList],
     [c, geoalgFilterSettings]
    ;

geoalgSpec2 =
      LSTSQR, [c, geoalgFilterSettings]
    | MINMAX, [c, geoalgFilterSettings]
    | MAXINS, [c, geoalgFilterSettings]
    | MINCIR, [c, geoalgFilterSettings]
    | DEFALT, [c, geoalgFilterSettings]
    | EXTERN, c, geoalgExternFunc, [c, paramList],
      [c, geoalgFilterSettings]
    ;

geoalgSpec3 =
      DEFALT, [c, geoalgFilterSettings]
    | EXTERN, c, geoalgExternFunc, [c, paramList],
      [c, geoalgFilterSettings]
    ;

geoalgSpec4 =
      LSTSQR, [c, geoalgFilterSettings]
    | MINMAX, [c, geoalgFilterSettings]
    | BSPLIN, [c, geoalgFilterSettings]
    | DEFALT, [c, geoalgFilterSettings]
    | EXTERN, c, geoalgExternFunc, [c, paramList],
      [c, geoalgFilterSettings]
    ;

geoalgSpec5 =
      LSTSQR, [c, geoalgFilterSettings]
    | MINMAX, [c, geoalgFilterSettings]
    | BEZIER, [c, geoalgFilterSettings]
    | NURBS, [c, geoalgFilterSettings]
    | DEFALT, [c, geoalgFilterSettings]
    | EXTERN, c, geoalgExternFunc, [c, paramList],
      [c, geoalgFilterSettings]
    ;

geoalgSpec6 =
      LSTSQR, [c, geoalgFilterSettings]
    | BSPLIN, [c, geoalgFilterSettings]
    | DEFALT, [c, geoalgFilterSettings]
    | EXTERN, c, geoalgExternFunc, [c, paramList],
      [c, geoalgFilterSettings]
    ;

geoalgExternFunc =
      DMIS, c, mLabel
    | DME, c, stringVal
    | SYS, c, stringVal
    ;

geoalgFilterSettings =
      [geoalgEliminate, c], geoalgFilter
    | geoalgEliminate
    ;
```

```
geoalgEliminate =
      ELIMINATE, c, STDDEVLIMIT, c, rentVal
    | ELIMINATE, c, OFF
    ;

geoalgFilter =
      FILTER, c, LAMBDAC, c, geoalgFilterType
    | FILTER, c, CIRCULAR, c, geoalgFilterType
    | FILTER, c, OFF
    ;

geoalgFilterType =
      LOWPASS, c, rentVal, c, geoalgFilterCurve
    | HIGHPASS, c, rentVal, c, geoalgFilterCurve
    | BANDPASS, c, rentVal, c, rentVal, c, geoalgFilterCurve
    ;

geoalgFilterCurve =
      GAUSS
    | TWORC
    | SPLINE
    | RECFILT
    ;

(*-------------------------------------------------------------*)
(* GEOM *)
(*-------------------------------------------------------------*)

geomStm =
      gLabel, '=', GEOM, '/', geomMinor, #
    ;

geomMinor =
      NONE
    | didLabel
    | gLabel, [c, OFFSET, c, rentVal]
    | gLabel, c, ENTITY, c, geomEntityList
    ;

geomEntityList =
       [geomEntityList, c], geomEntityItem
    ;

geomEntityItem =
      stringVal, [c, OFFSET, c, rentVal]
    ;

(*-------------------------------------------------------------*)
(* GOHOME *)
(*-------------------------------------------------------------*)

gohomeStm =
      GOHOME, #
    ;

(*-------------------------------------------------------------*)
(* GOTARG *)
(*-------------------------------------------------------------*)

gotargStm =
```

```
      GOTARG, '/', point, #
    | GOTARG, '/', CART, c, point, #
    | GOTARG, '/', POL, c, point, #
    ;

(*-------------------------------------------------------------*)
(* GOTO *)
(*-------------------------------------------------------------*)

gotoStm =
      GOTO, '/', gotoMinor, #
    ;

gotoMinor =
      gotoAbs
    | gotoRel
    | gotoArc
    | gotoAxis
    | probeOrient
    ;

gotoAxis =
      axis, c, rentVal, 2*[c, axis, c, rentVal]
    ;

gotoAbs =
      point
    | point, c, sensorMove
    | CART, c, point
    | CART, c, point, c, sensorMove
    | POL, c, point
    | POL, c, point, c, sensorMove
    ;

sensorMove =
      saLabel
    | gotoWristList
    | probeOrient
    ;

gotoRel =
      INCR, c, rentVal, c, vector
    ;

gotoArc =
      ARC, c, point, c, point, [c,  probeOrient]
    ;

probeOrient =
      featureLabel, [c, FZ, c, angle]
    | VEC, c, vector, [c, FZ, c, angle]
    | PCS, c, rentVal, c, rentVal, c, rentVal
    | HEADCS, c, rentVal, c, rentVal, [c, rentVal]
    ;

gotoWristList =
      [gotoWristList, c], gotoWristItem
    ;

gotoWristItem =
```

```
      swLabel, c, gotoWristAngleList
    ;

gotoWristAngleList =
     [gotoWristAngleList, c], gotoWristAngleItem
    ;

gotoWristAngleItem =
      stringVal, c, angle
    ;

(*-------------------------------------------------------------*)
(* GROUP *)
(*-------------------------------------------------------------*)

groupStm =
      gsaLabel, '=', GROUP, '/', saLabel, c, saLabelList, #
    ;

(*-----------------------------------------------------*)
(* IF *)
(*-----------------------------------------------------*)

ifStm =
      IF, '/', '(', boolVal, ')', #
    ;

(*-------------------------------------------------------------*)
(* INCLUD *)
(*-------------------------------------------------------------*)

includStm =
      INCLUD, '/', includMinor, #
    ;

includMinor =
      includDmis
    | includDme
    ;

includDmis =
      DMIS, c, stringConst
    ;

includDme =
      DME, c, stringConst
    ;

(*-------------------------------------------------------------*)
(* ITERAT *)
(*-------------------------------------------------------------*)

iteratStm =
      realVar, '=', ITERAT, '/', iteratMinor, #
    ;

iteratMinor =
      jumpLabel, c, jumpLabel, c, iteratConvergence, c,
      intVal, c, iteratCriterionList
    ;
```

```
iteratConvergence =
      rentVal, c, ABSL
    | rentVal, c, INCR
    ;

iteratCriterionList =
      [iteratCriterionList, c], iteratCriterionItem
    ;

iteratCriterionItem =
      iteratCriterionStart, c, featureActualList
    ;

iteratCriterionStart =
      axis
    | vector
    | NOM
    ;

(*-------------------------------------------------------------*)
(* JUMPTO *)
(*-------------------------------------------------------------*)

jumptoStm =
      JUMPTO, '/', jumpLabel, #
    ;

(*-------------------------------------------------------------*)
(* KEYCHAR *)
(*-------------------------------------------------------------*)

keycharStm =
      kcLabel, '=', KEYCHAR, '/', keycharMinor, #
    ;

keycharMinor =
      keycharOneFeature
    | keycharTwoFeatures
    ;

keycharOneFeature =
      fLabel, c, tLabelList, [c, criticality]
    ;

keycharTwoFeatures =
      fLabel, c, fLabel, c, tLabel, [c, criticality]
    ;

criticality =
      CRITICAL
    | MAJOR
    | MINOR
    ;

(*-------------------------------------------------------------*)
(* LITDEF *)
(*-------------------------------------------------------------*)

litdefStm =
```

```
      vlLabel, '=', LITDEF, '/', litdefMinor, #
   ;

litdefMinor =
      litdefLight
   | litdefStrobe
   ;

litdefLight =
      litdefLightType, c, vector
   ;

litdefLightType =
      SURF
   | BACK
   | GRID
   | OBLQ
   ;

litdefStrobe =
      STROBE, c, litdefStrobeType, c, rentVal, c, vector
   ;

litdefStrobeType =
      CYCLE, c, rentVal
   | TRIGER
   ;

(*-------------------------------------------------------------*)
(* LOCATE *)
(*-------------------------------------------------------------*)

locateStm =
      dLabel, '=', LOCATE, '/', locateMinor, #
   ;

locateMinor =
       [locateTransAllowed, c], [locateRotAllowed, c], locateLabelList
   ;

locateTransAllowed =
      XDIR
   | YDIR
   | ZDIR
   | XYDIR
   | YZDIR
   | ZXDIR
   | XYZDIR
   | NOTRAN
   ;

locateRotAllowed =
      XAXIS
   | YAXIS
   | ZAXIS
   | XYAXIS
   | YZAXIS
   | ZXAXIS
   | XYZAXI
   | NOROT
```

```
    ;

locateLabel =
      faLabel
    | maLabel
    | datLabel
    ;

locateLabelList =
      [locateLabelList, c], locateLabel
    ;

(*-------------------------------------------------------------*)
(* LOTID *)
(*-------------------------------------------------------------*)

lotidStm =
      liLabel, '=', LOTID, '/', stringVal, #
    ;

(*-------------------------------------------------------------*)
(* MACRO *)
(*-------------------------------------------------------------*)

macroStm =
      mLabelConst, '=', MACRO, ['/', macroParList], #
    ;

macroParList =
      [macroParList, c], macroPar
    ;

macroPar =
      MacroVarName
    | stringConst
    ;

(*-------------------------------------------------------------*)
(* MATDEF *)
(*-------------------------------------------------------------*)

matdefStm =
      maLabel, '=', MATDEF, '/', matdefMinor, #
    ;

matdefMinor =
      matdefFeat
    | matdefGeom
    ;

matdefFeat =
      fLabel, c, matdefSpec
    ;

matdefGeom =
      gLabel, c, matdefSpec
    ;

matdefSpec =
      faLabelList, [c, matdefType]
```

```
    ;

faLabelList =
      [faLabelList, c], faLabel
    ;

matdefType =
      matdefMating, c, matdefMethod, c, rentVal, [c, rentVal],
      [c, matdefMat]
    ;

matdefMating =
      PT2PT
    | PT2LN
    | PT2PL
    | LN2LN
    ;

matdefMethod =
      BF
    | FZ
    ;

matdefMat =
      MMC, c, tLabel
    | LMC, c, tLabel
    ;

(*-------------------------------------------------------------*)
(* MEAS *)
(*-------------------------------------------------------------*)

measStm =
      MEAS, '/', measGeotype, c, fLabel, c, intVal, #
    | MEAS, '/', POINT, c, fLabel, c, intVal, #
    | MEAS, '/', POINT, c, COMP, c, measVar2, c, fLabel, c, intVal, #
    ;

measGeotype =
      ARC
    | CIRCLE
    | CONE
    | CONRADSEGMNT
    | CPARLN
    | CYLNDR
    | CYLRADSEGMNT
    | EDGEPT
    | ELLIPS
    | ELONGCYL
    | GCURVE
    | GSURF
    | LINE
    | OBJECT
    | PARPLN
    | PLANE
    | REVSURF
    | RCTNGL
    | SPHERE
    | SPHRADSEGMNT
    | SYMPLN
```

```
    | TORUS
    | TORRADSEGMNT
    ;

measVar2 =
      AXDIR
    | DME
    | FEAT, c, measVar2Aux
    | POL
    | SPH
    | VEC, c, vector
    ;

measVar2Aux =
      faLabel
    | fLabel
    | gLabel
    ;

(*-------------------------------------------------------------*)
(* MFGDEV *)
(*-------------------------------------------------------------*)

mfgdevStm =
      mdLabel, '=', MFGDEV, '/', stringVal, #
    ;

(*-------------------------------------------------------------*)
(* MODE *)
(*-------------------------------------------------------------*)

modeStm =
      MODE, '/', [AUTO, c], [PROG, c], MAN, #
    ;

(*-------------------------------------------------------------*)
(* OBTAIN *)
(*-------------------------------------------------------------*)

obtainStm =
      boolVar, '=', OBTAIN, '/', obtainLabeled, #
    | stringVar, '=', OBTAIN, '/', obtainLabeled, #
    | intVar, '=', OBTAIN, '/', obtainLabeled, #
    | realVar, '=', OBTAIN, '/', obtainLabeled, #
    | realVar, '=', OBTAIN, '/', obtainPoint, #
    ;

obtainLabeled =
      datumLabel, c, intVal
    | featureLabel, c, intVal
    | sensorLabel, c, intVal
    | toleranceLabel, c, intVal
    | didLabel, c, intVal
    | qisLabel, c, intVal
    | crLabel, c, intVal
    | gLabel, c, intVal
    | gsaLabel, c, intVal
    | kcLabel, c, intVal
    | maLabel, c, intVal
    | pLabel, c, intVal
```

```
    | rLabel, c, intVal
    | rmLabel, c, intVal
    | rtLabel, c, intVal
    | sgLabel, c, intVal
    | sgsLabel, c, intVal
    | sraLabel, c, intVal
    | ssLabel, c, intVal
    | swLabel, c, intVal
    | sxLabel, c, intVal
    | thLabel, c, intVal
    | vLabel, c, intVal
    | vaLabel, c, intVal
    | vfLabel, c, intVal
    | vlLabel, c, intVal
    | vwLabel, c, intVal
    ;

obtainPoint =
      featureLabel, '[', intVal, ']', c, intVal
    ;

(*-------------------------------------------------------------*)
(* OPEN *)
(*-------------------------------------------------------------*)

openStm =
      OPEN, '/', openMinor, #
    ;

openMinor =
      didLabel, c, openDevice
    | didLabel, c, openFdata
    | didLabel, c, openCadfile
    | didLabel, c, storageSpecList
    | didLabel, c, openDMLfile
    ;

openDevice =
      DIRECT, c, inputAccess
    | DIRECT, c, outputAccess
    ;

inputAccess =
      INPUT
    ;

outputAccess =
      OUTPUT
    | OUTPUT, c, APPEND
    | OUTPUT, c, OVERWR
    ;

openFdata =
      FDATA, c, vLabel, c, outputAccess
    | FDATA, c, DMIS, c, outputAccess
    ;

openCadfile =
      CAD, [c, cadfileType]
    ;
```

```
cadfileType =
      STEP
    | IGES
    | VENDOR, c, stringVal
    ;

storageSpecList =
      [storageSpecList, c], storageSpec
    ;

storageSpec =
      SNS
    | PCS
    | FEATUR
    | RTAB
    ;

openDMLfile =
      DML, c, stringVal, c, INPUT
    | DML, c, stringVal, c, OUTPUT, c, PTDATA, c, state
    ;

(*-------------------------------------------------------------*)
(* OPERID *)
(*-------------------------------------------------------------*)

operidStm =
      opLabel, '=', OPERID, '/', stringVal, #
    ;

(*-------------------------------------------------------------*)
(* OUTPUT *)
(*-------------------------------------------------------------*)

outputStm =
      OUTPUT, '/', outputMinor, #
    ;

outputMinor =
      outputFeatData
    | outputConstData
    | outputKeycharData
    | outputSensorData
    | outputToleranceData
    | outputReportData
    ;

outputFeatData =
      outputFeat
    | outputDat
    ;

outputFeat =
      fLabel, [pointRange], [c, rLabel]
    | faLabel, [pointRange], [c, rLabel]
    | fLabel, c, faLabel, c, taLabel, [c, rLabel]
    | faLabel, c, faLabel, c, taLabel, [c, rLabel]
    | fLabel, c, fLabel, c, tLabel, [c, rLabel]
    | faLabel, c, fLabel, c, taLabel, [c, rLabel]
```

```
    | fLabel, [pointRange], c, tLabelList, [c, rLabel]
    | faLabel, [pointRange], c, taLabelList, [c, rLabel]
    ;

outputDat =
      datLabel, c, faLabel, c, taLabel, [c, rLabel]
    | faLabel, c, datLabel, c, taLabel, [c, rLabel]
    ;

outputReportData =
      rLabel
    ;

outputConstData =
      seLabel
    | stLabel
    | stLabel, c, seLabel, c, tLabel
    | stLabel, c, seLabel, c, taLabel
    ;

outputKeycharData =
      kcLabel
    | kcaLabel
    ;

outputToleranceData =
      tLabel
    | taLabel
    ;

outputSensorData =
      sLabel
    | saLabel, [c, outputSensorDesc]
    ;

outputSensorDesc =
      stringVal, [c, outputSensorWristList]
    | intVal, [c, outputSensorWristList]
    | outputSensorWristList
    | CURENT
    ;

outputSensorWristList =
      [outputSensorWristList, c], outputSensorWristItem
    ;

outputSensorWristItem =
      swLabel, c, outputSensorWristAngleList
    ;

outputSensorWristAngleList =
      [outputSensorWristAngleList,  c],  outputSensorWristAngle
    ;

outputSensorWristAngle =
      stringVal, c, angle
    ;

(*-------------------------------------------------------------*)
(* PAMEAS *)
```

```
(*-------------------------------------------------------------*)

pameasStm =
      PAMEAS, '/', pameasMinor, #
    ;

pameasMinor =
      pameasVar2List
    | pameasVar3List
    ;

pameasDetail =
       DISTANCE, c, rentVal, [c, SCNVEL, c, fedratLinear],
           [c, PITCH, c, rentVal]
     | SCNVEL, c, fedratLinear, [c, PITCH, c, rentVal]
     | PITCH, c, rentVal
     | NODATA
     | SCNVEL, c, fedratLinear, c, NODATA
     ;

pameasVar2List =
      [pameasVar2List, c], pameasVar2ListItem
    ;

pameasVar2ListItem =
      pameasDetail, c, pLabel, [c, vector], [c, SELFCENTER, c, vector,
            [c, forceOrDeflection]], [c, pameasRemove], [c, ROTARY, c, rtLabel]
    ;

pameasVar3List =
      [pameasVar3List, c], pameasVar3ListItem
    ;

pameasVar3ListItem =
      pLabel, [c, vector], [c, SELFCENTER, c, vector,
            [c, forceOrDeflection]], [c, pameasRemove], [c, ROTARY, c, rtLabel]
    ;

pameasRemove =
      REMOVE, c, COUNT, c, intVal, c, intVal
    | REMOVE, c, DIST, c, rentVal, c, rentVal
    | REMOVE, c, ALL
    ;

(*-------------------------------------------------------------*)
(* PARTID *)
(*-------------------------------------------------------------*)

partidStm =
      pnLabel, '=', PARTID, '/', stringVal, #
    ;

(*-------------------------------------------------------------*)
(* PARTRV *)
(*-------------------------------------------------------------*)

partrvStm =
      prLabel, '=', PARTRV, '/', stringVal, #
    ;
```

```
(*-------------------------------------------------------------*)
(* PARTSN *)
(*-------------------------------------------------------------*)

partsnStm =
      psLabel, '=', PARTSN, '/', stringVal, #
    ;

(*-------------------------------------------------------------*)
(* PATH *)
(*-------------------------------------------------------------*)

pathStm =
      pLabel, '=', PATH, '/', pathMinor, #
    ;

pathMinor =
      pathPoint
    | pathArc
    | pathCurve
    | pathHelical
    | pathLine
    | pathSurface
    | pathUnknown
    ;

pathPoint =
      POINT, c, typePoint, c, vector
    ;

pathArc =
      ARC, c, typePoint, c, vector, c, rentVal, c,
      angle, c, angle, [c, vector]
    ;

pathCurve =
      CURVE, c, pathCurvePtdata
    | CURVE, c, pathCurvePoints
    ;

pathCurvePtdata =
      curvePtdataPcs
    | curvePtdataHeadcs
    ;

curvePtdataPcs =
      PTDATA, c, pointVec, [c, PCS, c, triplet], c, curvePtdataPcsList
    ;

curvePtdataPcsList =
    [curvePtdataPcsList, c], curvePtdataPcsListItem
    ;

curvePtdataPcsListItem =
      PTDATA, c, pointVec, [c, PCS, c, triplet]
    ;

curvePtdataHeadcs =
      PTDATA, c, pointVec, c, HEADCS, c, rentVal, c, rentVal, [c, rentVal],
           c, curvePtdataHeadcsList
```

```
    ;

curvePtdataHeadcsList =
       [curvePtdataHeadcsList, c], curvePtdataHeadcsListItem
    ;

curvePtdataHeadcsListItem =
      PTDATA, c, pointVec, [c, HEADCS, c, rentVal, c, rentVal, [c, rentVal]]
    ;

pathCurvePoints =
      pointVec, c, pointVecList
    ;

pointVecList =
      [pointVecList, c], pointVec
    ;

pathHelical =
      HELICAL, c, typePoint, c, vector, c, rentVal, c,
      angle, c, angle, [c, vector], c, rentVal, [c, angle]
    ;

pathLine =
      LINE, c, BND, c, CART, c, START, c, point, c, [csSpec, c], END, c,
           point, c, [csSpec, c], VEC, c, vector
    | LINE, c, BND, c, POL, c, point, c, point, c, VEC, c, vector
    ;

csSpec =
      PCS, c, triplet
    | HEADCS, c, rentVal, c, rentVal, [c, rentVal]
    ;

pathSurface =
      SURFACE, c, surfPtdataList
    | SURFACE, c, surfPtdataList, c, PCS, c, triplet
    | SURFACE, c, surfPtdataList, c, PCS, c, triplet, c, surfPtdataPcsList
    | SURFACE, c, surfPtdataList, c, HEADCS, c,
           rentVal, c, rentVal, [c, rentVal]
    | SURFACE, c, surfPtdataList, c, HEADCS, c,
           rentVal, c, rentVal, [c, rentVal], c, surfPtdataHeadcsList
    ;

surfPtdataList =
      [surfPtdataList, c], surfPtdataListItem
    ;

surfPtdataListItem =
      PTDATA, c, pointVec, c, rentVal, c, rentVal
    ;

surfPtdataPcsList =
      [surfPtdataPcsList, c], surfPtdataPcsListItem
    ;

surfPtdataPcsListItem =
      PTDATA, c, pointVec, c, rentVal, c, rentVal, [c, PCS, c, triplet]
    ;
```

```
surfPtdataHeadcsList =
      [surfPtdataHeadcsList, c], surfPtdataHeadcsListItem
   ;

surfPtdataHeadcsListItem =
     PTDATA, c, pointVec, c, rentVal, c, rentVal,
          [c, HEADCS, c,  rentVal, c, rentVal, [c, rentVal]]
   ;

pathUnknown =
     UNKNOWN, c, point, c, point, c, point, [c, vector]
   ;


(*-------------------------------------------------------------*)
(* PLANID *)
(*-------------------------------------------------------------*)

planidStm =
     plLabel, '=', PLANID, '/', stringVal, #
   ;

(*-------------------------------------------------------------*)
(* POP *)
(*-------------------------------------------------------------*)

popStm =
     POP, '/', stackElementList, #
   ;

(*-------------------------------------------------------------*)
(* PRCOMP *)
(*-------------------------------------------------------------*)

prcompStm =
     PRCOMP, '/', state, #
   ;

(*-------------------------------------------------------------*)
(* PREVOP *)
(*-------------------------------------------------------------*)

prevopStm =
     pvLabel, '=', PREVOP, '/', stringVal, #
   ;

(*-------------------------------------------------------------*)
(* PROCID *)
(*-------------------------------------------------------------*)

procidStm =
     pcLabel, '=', PROCID, '/', stringVal, #
   ;

(*-------------------------------------------------------------*)
(* PROMPT *)
(*-------------------------------------------------------------*)

promptStm =
     boolVar, '=', PROMPT, '/', stringVal, #
```

```
    | stringVar, '=', PROMPT, '/', stringVal, [c, intVal], #
    | intVar , '=', PROMPT, '/', promptIntEnd, #
    | realVar, '=', PROMPT, '/', stringVal, [c, rentVal, [c, rentVal]], #
    | ccLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | ciLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | csLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | diLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | dsLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | dvLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | fiLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | fsLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | liLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | mdLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | opLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | pcLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | plLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | pnLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | prLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | psLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | pvLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    | qLabel , '=', PROMPT, '/', stringVal, [c, intVal], #
    | tlLabel, '=', PROMPT, '/', stringVal, [c, intVal], #
    ;

promptIntEnd =
      stringVal, [c, intVal, [c, intVal]]
    | promptItemList
    ;

promptItemList =
      [promptItemList, c], promptItem
    ;

promptItem =
      BUTTON, c, stringVal, c, intVal
    | CHECK, c, stringVal, c, boolVar
    | EDIT, c, promptVar, 2*[c, rentVal]
    | GROUP, c, stringVal, c, promptVar, c, stringVal, c, stringList
    | LIST, c, promptVar, c, stringVal, c, stringList
    | PICTURE, c, stringVal, [c, intVal]
    | PIXBTN, c, stringVal, [c, stringVal], c, intVal
    | SOUND, c, stringVal
    | TEXT, c, stringVal
    | TITLE, c, stringVal
    ;

promptVar =
      stringVar
    | boolVar
    | intVar
    | realVar
    | qisLabel
    ;

(*-------------------------------------------------------------*)
(* PSTHRU *)
(*-------------------------------------------------------------*)

psthruStm =
      PSTHRU, '/', psthruMinor, #
```

```
   ;

psthruMinor =
      COMAND, c, stringVal
    | CONTIN
    | PAUSE
    | START
    | STOP
    | TRMATX, c, matrix
    ;

(*-------------------------------------------------------------*)
(* PTBUFF *)
(*-------------------------------------------------------------*)

ptbuffStm =
      PTBUFF, '/', state, #
    ;

(*-------------------------------------------------------------*)
(* PTMEAS *)
(*-------------------------------------------------------------*)

ptmeasStm =
      PTMEAS, '/', typePoint, [c, vector], [c, ptmeasEnd], #
    ;

ptmeasEnd =
      PCS, c, triplet, [c, touchSpec]
    | HEADCS, c, rentVal, c, rentVal, [c, rentVal], [c, touchSpec]
    | touchSpec
    ;

touchSpec =
      HEADTOUCH
    | ALLAXESTOUCH
    ;

(*-------------------------------------------------------------*)
(* PUSH *)
(*-------------------------------------------------------------*)

pushStm =
      PUSH, '/', stackElementList, #
    ;

(*-------------------------------------------------------------*)
(* QISDEF *)
(*-------------------------------------------------------------*)

qisdefStm =
      qLabel, '=', QISDEF, '/', stringVal, [c, stringVal], #
    ;

(*-------------------------------------------------------------*)
(* RAPID *)
(*-------------------------------------------------------------*)

rapidStm =
      RAPID, '/', rentVal, #
```

```
   ;

(*-------------------------------------------------------------*)
(* READ *)
(*-------------------------------------------------------------*)

readStm =
      READ, '/', readMinor, #
   ;

readMinor =
      didLabel, c, readSpecList
   ;

readSpecList =
      [readSpecList, c], readSpec
   ;

readSpec =
      rwVar, [rwFormat]
   ;

(*-------------------------------------------------------------*)
(* RECALL *)
(*-------------------------------------------------------------*)

recallStm =
      recallDatumStm
   | recallSensorStm
   | recallFeatureStm
   | recallRotaryTableStm
   | recallDMLStm
   ;

recallDatumStm =
      RECALL, '/', datumLabel, [c, didLabel], #
   ;

recallSensorStm =
      RECALL, '/', sensorLabel, [c, didLabel], #
   ;

recallFeatureStm =
      RECALL, '/', faLabel, [c, didLabel], #
   ;

recallRotaryTableStm =
      RECALL, '/', rtLabel, [c, didLabel], #
   ;

recallDMLStm =
      RECALL, '/', DML, c, didLabel, c, daLabel, [c, didLabel], #
   ;

(*-------------------------------------------------------------*)
(* REFMNT *)
(*-------------------------------------------------------------*)

refmntStm =
      rmLabel, '=', REFMNT, '/', refmntMinor, #
```

```
   ;

refmntMinor =
      XVEC, c, vector, c, ZVEC, c, vector, c, MNTLEN, c, vector
   ;

(*-------------------------------------------------------------*)
(* REPORT *)
(*-------------------------------------------------------------*)

reportStm =
      rLabel, '=', REPORT, '/', reportItemList, [c, stringVal], #
   ;

reportItemList =
      [reportItemList, c], reportItem
   ;

reportItem =
      ALGOR
    | DATE
    | HUMID
    | MODE
    | TEMPC
    | TEMPF
    | TEMPWC
    | TEMPWF
    | TIME
    | qisLabel
   ;

(*-------------------------------------------------------------*)
(* RESUME *)
(*-------------------------------------------------------------*)

resumeStm =
      RESUME, '/', resumeMinor, #
   ;

resumeMinor =
      jumpLabel
    | CURENT
    | END
    | NEXT
    | START
    | STOP
   ;

(*-------------------------------------------------------------*)
(* RMEAS *)
(*-------------------------------------------------------------*)

rmeasStm =
      RMEAS, '/', rmeasMinor, #
   ;

rmeasMinor =
      rmeasArc
    | rmeasCircle
    | rmeasCone
```

```
    | rmeasCparln
    | rmeasCylndr
    | rmeasEdgept
    | rmeasEllips
    | rmeasGcurve
    | rmeasGsurf
    | rmeasLine
    | rmeasObject
    | rmeasParpln
    | rmeasPlane
    | rmeasPoint
    | rmeasRctngl
    | rmeasSphere
    | rmeasSympln
    | rmeasTorus
    ;

rmeasArc =
      ARC, c, fLabel, c, intVal, c, rmeasSpecFa
    | ARC, c, fLabel, c, intVal, c, rmeasSpecVecbld
    ;

rmeasCircle =
      CIRCLE, c, fLabel, c, intVal, c, rmeasSpecFa
    | CIRCLE, c, fLabel, c, intVal, c, rmeasSpecVecbld
    ;

rmeasCone =
      CONE, c, fLabel, c, intVal, c, rmeasSpecFa
    ;

rmeasCparln =
      CPARLN, c, fLabel, c, intVal, c, rmeasSpecFaOrient
    | CPARLN, c, fLabel, c, intVal, c, rmeasSpecVecbldOrient
    ;

rmeasCylndr =
      CYLNDR, c, fLabel, c, intVal, c, rmeasSpecFa
    ;

rmeasEdgept =
      EDGEPT, c, fLabel, c, intVal, c, rmeasSpecFaAxis
    | EDGEPT, c, fLabel, c, intVal, c, rmeasSpecVecbldEdgept
    ;

rmeasEllips =
      ELLIPS, c, fLabel, c, intVal, c, rmeasSpecFa
    | ELLIPS, c, fLabel, c, intVal, c, rmeasSpecVecbld
    ;

rmeasGcurve =
      GCURVE, c, fLabel, c, intVal, c, rmeasSpecFeatAxis
    | GCURVE, c, fLabel, c, intVal, c, rmeasSpecVecbld
    ;

rmeasGsurf =
      GSURF, c, fLabel, c, intVal, c, rmeasSpecFaAxis
    ;

rmeasLine =
```

```
      LINE, c, fLabel, c, intVal, c, rmeasSpecFeatAxis
    | LINE, c, fLabel, c, intVal, c, rmeasSpecVecbld
    ;

rmeasObject =
      OBJECT, c, fLabel, c, intVal, c, rmeasSpecFa
    | OBJECT, c, fLabel, c, intVal, c, rmeasSpecVecbld
    ;

rmeasParpln =
      PARPLN, c, fLabel, c, intVal, c, rmeasSpecFa
    ;

rmeasPlane =
      PLANE, c, fLabel, c, intVal, c, rmeasSpecFaAxis
    ;

rmeasPoint =
      POINT, c, fLabel, c, intVal, c, rmeasSpecFaAxis
    | POINT, c, fLabel, c, intVal, c, rmeasSpecVecbld
    ;

rmeasRctngl =
      RCTNGL, c, fLabel, c, intVal, c, rmeasSpecFa
    ;

rmeasSphere =
      SPHERE, c, fLabel, c, intVal, c, rmeasSpecFaAxis
    ;

rmeasSympln =
      SYMPLN, c, fLabel, c, intVal, c, rmeasSpecFa
    ;

rmeasTorus =
      TORUS, c, fLabel, c, intVal, c, rmeasSpecFaAxis
    ;

rmeasSpecFa =
      faLabel
    ;

rmeasSpecFaAxis =
      faLabel, [c, axis]
    | axis
    ;

rmeasSpecFaOrient =
      faLabel, [c, ORIENT]
    ;

rmeasSpecFeatAxis =
      featureLabel, [c, axis]
    | axis
    ;

rmeasSpecVecbld =
      VECBLD, c, rentVal, c, intVal
    ;
```

```
rmeasSpecVecbldOrient =
      rmeasSpecVecbld, [c, ORIENT]
    ;

rmeasSpecVecbldEdgept =
      rmeasSpecVecbld, c, rentVal, [c, posDir], [c, axis]
    ;

(*-------------------------------------------------------------*)
(* ROTAB *)
(*-------------------------------------------------------------*)

rotabStm =
      ROTAB, '/', rotabMinor, #
    ;

rotabMinor =
      rtLabel, c, rotAbs, c, rotType, c, angle, [c, FZ, c, angle]
    | rtLabel, c, rotIncr, c, rotType, c, angle, [c, FZ, c, angle]
    | rtLabel, c, featureLabel, c, rotType, c, rotDir
    ;

(*-------------------------------------------------------------*)
(* ROTATE *)
(*-------------------------------------------------------------*)

rotateStm =
      dLabel, '=', ROTATE, '/', rotateMinor, #
    ;

rotateMinor =
      rotateValue
    | rotateFeature
    | rotateDatum
    ;

rotateValue =
      axis, c, angle
    ;

rotateFeature =
      axis, c, featureLabel, c, dir
    ;

rotateDatum =
      axis, c, datLabel, c, dir
    ;

(*-------------------------------------------------------------*)
(* ROTDEF *)
(*-------------------------------------------------------------*)

rotdefStm =
      rtLabel, '=', ROTDEF, '/', rotdefMinor, #
    ;

rotdefMinor =
      pointVec, [c, rtLabel]
    ;
```

```
(*-------------------------------------------------------------*)
(* ROTSET *)
(*-------------------------------------------------------------*)

rotsetStm =
      ROTSET, '/', rtLabel, c, angle, #
   ;

(*-------------------------------------------------------------*)
(* SAVE *)
(*-------------------------------------------------------------*)

saveStm =
      saveDatumStm
   | saveSensorStm
   | saveFeatureStm
   | saveRotaryTableStm
   | saveDMLStm
   ;

saveDatumStm =
      SAVE, '/', datumLabel, [c, didLabel], #
   ;

saveSensorStm =
      SAVE, '/', sensorLabel, [c, didLabel], #
   ;

saveFeatureStm =
      SAVE, '/', faLabel, [c, didLabel], #
   ;

saveRotaryTableStm =
      SAVE, '/', rtLabel, [c, didLabel], #
   ;

saveDMLStm =
      SAVE, '/', DML, c, didLabel, c, daLabel, [c, didLabel], #
   ;

(*-------------------------------------------------------------*)
(* SCNMOD *)
(*-------------------------------------------------------------*)

scnmodStm =
      SCNMOD, '/', state, #
   ;

(*-------------------------------------------------------------*)
(* SCNSET *)
(*-------------------------------------------------------------*)

scnsetStm =
      SCNSET, '/', scnsetMinor, #
   ;

scnsetMinor =
      scnsetPeck
   | scnsetDrag
   | scnsetNoncon
```

```
    | scnsetStop
    | scnsetVendor
    ;

scnsetPeck =
      PECK, c, scnsetSample
    ;

scnsetDrag =
      DRAG, c, scnsetSampleDrag
    ;

scnsetNoncon =
      NONCON, c, scnsetSample
    ;

scnsetStop =
      scnsetStopPlane
    | scnsetStopSphere
    ;

scnsetStopPlane =
      STOP, c, PLANE, c, vector, [c, RADIUS, c, rentVal],
      [c, COUNT, c, intVal]
    ;

scnsetStopSphere =
      STOP, c, SPHERE, c, rentVal
    ;

scnsetVendor =
      VENDOR, c, FORM
    | VENDOR, c, POS
    | VENDOR, c, SIZE
    ;

scnsetSample =
      DIST, c, rentVal, [c, axis]
    | CHORD, c, rentVal, [c, rentVal]
    | TIME, c, rentVal
    | ANGLE, c, angle
    | DEFALT
    ;

scnsetSampleDrag =
      DIST, c, rentVal, [c, axis], [c, forceOrDeflection]
    | CHORD, c, rentVal, [c, rentVal], [c, forceOrDeflection]
    | TIME, c, rentVal, [c, forceOrDeflection]
    | ANGLE, c, angle, [c, forceOrDeflection]
    | DEFALT, [c, forceOrDeflection]
    ;

forceOrDeflection =
      FORCE, c, rentVal
    | DEFLECTION, c, rentVal
    ;

(*--------------------------------------------------------------*)
(* SELECT *)
(*--------------------------------------------------------------*)
```

```
selectStm =
      SELECT, '/', intVal, #
    | SELECT, '/', stringVal, #
    ;

(*-------------------------------------------------------------*)
(* SENSOR *)
(*-------------------------------------------------------------*)

sensorStm =
      ssLabel, '=', SENSOR, '/', sensorMinor, #
    ;

sensorMinor =
      sensorProbe
    | sensorMltprb
    | sensorVideo
    | sensorLaser
    | sensorInfred
    | sensorNoncon
    | sensorPoint
    | sensorLine
    | sensorArea
    ;

sensorProbe =
      PROBE, c, sensorProbeGeometry, [c, sensorProbeForm], [c, sensorItem]
    ;

sensorMltprb =
      MLTPRB, c, intVal, c, sensorMltprbItemList, [c, sensorItem]
    ;

sensorMltprbItemList =
      [sensorMltprbItemList, c], sensorMltprbItem
    ;

sensorVideo =
      VIDEO, c, sensorGeometry, c, rentVal, c, rentVal, c,
      rentVal, [c, sensorItem]
    ;

sensorLaser =
      LASER, c, sensorGeometry, c, rentVal, c, rentVal, [c, sensorItem]
    ;

sensorInfred =
      INFRED, c, sensorGeometry, c, rentVal, c, rentVal, c,
      rentVal, c, rentVal, [c, sensorItem]
    ;

sensorNoncon =
      NONCON, c, sensorGeometry, c, intVal, [c, sensorItem]
    ;

sensorPoint =
      POINT, c, sensorGeometry, c, rentVal, [c, sensorItem]
    ;
```

```
sensorLine =
      LINE, c, sensorGeometry, c, rentVal, c, rentVal, [c, sensorItem]
    ;

sensorArea =
      AREA, c, sensorGeometry, c, rentVal, c, rentVal,
      c, rentVal, [c, sensorItem]
    ;

sensorProbeGeometry =
      vector, c, vector, c, rentVal
    ;

sensorProbeForm =
      SPHERE
    | CYLNDR, c, rentVal
    | DISK, c, rentVal
    ;

sensorItem =
      stringVal, c, stringVal, c, stringVal
    | stringVal, c, stringVal, c, intVal
    ;

sensorMltprbItem =
      stringVal, c, sensorProbeGeometry
    | intVal, c, sensorProbeGeometry
    ;

sensorGeometry =
      vector, c, vector, c, vector
    ;

(*-------------------------------------------------------------*)
(* SIMREQT *)
(*-------------------------------------------------------------*)

simreqtStm =
      srLabel, '=', SIMREQT, '/', simreqtMinor, #
    ;

simreqtMinor =
      FIRST
    | OPTIMAL
    ;

(*-------------------------------------------------------------*)
(* SNSDEF *)
(*-------------------------------------------------------------*)

snsdefStm =
      sLabel, '=', SNSDEF, '/', snsdefMinor, #
    ;

snsdefMinor =
      snsdefProbe
    | snsdefVideo
    | snsdefLaser
    | snsdefInfred
    | snsdefNoncon
```

```
    | snsdefBuild
    ;

snsdefProbe =
      PROBE, c, snsdefType, c, snsdefProbeLocation, c,
      rentVal, [c, snsdefProbeForm]
    ;

snsdefProbeLocation =
      CART, c, vector, c, vector
    | POL, c, angle, c, angle, c, vector, c, rentVal
    | VEC, c, vector, c, vector, c, rentVal
    | sensorLabel, c, CART, c, vector
    | sensorLabel, c, VEC, c, vector, c, rentVal
    ;

snsdefProbeForm =
      SPHERE
    | CYLNDR, c, rentVal
    | DISK, c, rentVal
    ;

snsdefVideo =
      VIDEO, c, snsdefType, c, snsdefLocation, c, rentVal,
      c, rentVal, c, rentVal
    ;

snsdefLaser =
      LASER, c, snsdefType, c, snsdefLocation, c, rentVal, c, rentVal
    ;

snsdefInfred =
      INFRED, c, snsdefType, c, snsdefLocation, c, rentVal,
      c, rentVal, c, rentVal, c, rentVal
    ;

snsdefNoncon =
      NONCON, c, snsdefType, c, snsdefLocation, c, intVal
    ;

snsdefLocation =
      CART, c, vector, c, vector
    | POL, c, angle, c, angle, c, vector
    | VEC, c, vector, c, vector
    ;

snsdefType =
      FIXED
    | INDEX
    ;

snsdefBuildSensor =
      ssLabel
    | sgsLabel
    ;

snsdefBuild =
      BUILD, c, snsdefBuildItemList, c, snsdefBuildSensor
    ;
```

```
snsdefBuildItemList =
      [snsdefBuildItemList, c], snsdefBuildItem
    ;

snsdefBuildItem =
      sgLabel
    | swLabel, [c, snsdefWristAngleList]
    | sxLabel
    | rmLabel
    ;

snsdefWristAngleList =
      [snsdefWristAngleList, c], snsdefWristAngleItem
    ;

snsdefWristAngleItem =
      stringVal, c, angle
    ;

(*-------------------------------------------------------------*)
(* SNSET *)
(*-------------------------------------------------------------*)

snsetStm =
      SNSET, '/', snsetItemList, #
    ;

snsetItemList =
      [snsetItemList, c], snsetItem
    ;

snsetItem =
      snsetTypeAndValue
    | snsetType
    | snsetLabelAndValue
    | snsetLabel
    | snsetToggle
    | CLRSRF, c, snsetFeat, [c, rentVal]
    | DEPTH, c, snsetFeat, [c, rentVal]
    ;

snsetTypeAndValue =
      APPRCH, c, rentVal
    | CLRSRF, c, rentVal
    | DEPTH, c, rentVal
    | MINCON, c, rentVal
    | RETRCT, c, rentVal
    | SEARCH, c, rentVal
    | SCALEX, c, rentVal
    | SCALEY, c, rentVal
    ;

snsetType =
      FOCUSY
    | FOCUSN
    ;

snsetToggle =
      CLRSRF, c, OFF
    | DEPTH, c, OFF
```

```
   ;

snsetLabelAndValue =
      vlLabel, c, rentVal
   ;

snsetLabel =
      vaLabel
   | vfLabel
   | vwLabel
   ;

snsetFeat =
      fLabel
   | faLabel
   | datLabel
   ;

(*-------------------------------------------------------------*)
(* SNSGRP *)
(*-------------------------------------------------------------*)

snsgrpStm =
      sgsLabel, '=', SNSGRP, '/', snsgrpMinor, #
   ;

snsgrpMinor =
      BUILD, c, snsgrpItemList, c, ssLabel
   | BUILD, c, snsgrpItemList, c, sgsLabel
   ;

snsgrpItemList =
      [snsgrpItemList, c], snsgrpItem
   ;

snsgrpItem =
      sgLabel
   | swLabel
   | sxLabel
   | rmLabel
   ;

(*-------------------------------------------------------------*)
(* SNSLCT *)
(*-------------------------------------------------------------*)

snslctStm =
      SNSLCT, '/', snslctSensor, #
   | SNSLCT, '/', snslctGroup, #
   ;

snslctGroup =
      gsaLabel, c, featureLabel, [c, FZ, c, angle]
   | gsaLabel, c, VEC, c, vector, [c, FZ, c, angle]
   ;

snslctSensor =
      sensorLabel, [c, snslctTipData], [c, snslctData]
   | sensorLabel, c, sensorList, [c, snslctData]
   ;
```

```
snslctTipData =
      stringVal
    | intVal
    ;

sensorList =
      [sensorList, c], sensorLabel
    ;

snslctData =
      snslctWristList
    | probeOrient
    ;

snslctWristList =
      [snslctWristList, c], snslctWristItem
    ;

snslctWristItem =
      swLabel, c, snslctWristAngleList
    ;

snslctWristAngleList =
      [snslctWristAngleList, c], snslctWristAngleItem
    ;

snslctWristAngleItem =
      stringVal, c, angle
    | stringVal, c, featureLabel, [c, FZ, c, angle]
    | stringVal, c, VEC, c, vector, [c, FZ, c, angle]
    ;

(*-------------------------------------------------------------*)
(* SNSMNT *)
(*-------------------------------------------------------------*)

snsmntStm =
      SNSMNT, '/', snsmntMinor, #
    ;

snsmntMinor =
      XVEC, c, vector, c, ZVEC, c, vector, c,
      MNTLEN, c, vector
    ;

(*-------------------------------------------------------------*)
(* TECOMP *)
(*-------------------------------------------------------------*)

tecompStm =
      TECOMP, '/', tecompMinor, #
    ;

tecompMinor =
      MACH, c, state
    | PART, c, ON, [tecompVar2], c, rentVal, [c, rentVal], c, stringList
    | PART, c, ON, [tecompVar2], c, rentVal, [c, rentVal], c, ALL
    | PART, c, OFF
    ;
```

```
tecompVar2 =
      c, daLabel
    | c, OFFSET, c, rentVal, c, rentVal, c, rentVal
    ;

(*-------------------------------------------------------------*)
(* TEXT *)
(*-------------------------------------------------------------*)

textStm =
      TEXT, '/', textMinor, #
    ;

textMinor =
      textMan
    | textOper
    | textOutfil
    | textQuery
    ;

textMan =
      MAN, c, stringVal
    ;

textOper =
      OPER, c, stringVal
    ;

textOutfil =
      OUTFIL, c, stringVal
    ;

textQuery =
      QUERY, c, textQueryFormat, c, stringVal
    ;

textQueryFormat =
      labelName, c, intVal, c, textType, c, leftRight
    ;

textType =
      ALPHA
    | NUMERIC
    | PRNTCHAR
    ;

(*-------------------------------------------------------------*)
(* THLDEF *)
(*-------------------------------------------------------------*)

thldefStm =
      thLabel, '=', THLDEF, '/', thldefPocketList, #
    ;

thldefPocketList =
      [thldefPocketList, c], thldefPocket
    ;

thldefPocket =
```

```
      sLabel, c, intVal
    | ssLabel, c, intVal
    | sgsLabel, c, intVal
    | sgLabel, c, intVal
    | swLabel, c, intVal
    | sxLabel, c, intVal
    | rmLabel, c, intVal
    ;

(*-------------------------------------------------------------*)
(* TOL *)
(*-------------------------------------------------------------*)

tolStm =
      tLabel, '=', TOL, '/', tolMinor, #
    ;

tolMinor =
      tolAngl
    | tolAnglb
    | tolAnglr
    | tolAnglwrt
    | tolCirlty
    | tolCompos
    | tolConcen
    | tolCortol
    | tolCprofl
    | tolCprofs
    | tolCrnout
    | tolCylcty
    | tolDiam
    | tolDistb
    | tolDistwrt
    | tolFlat
    | tolGtol
    | tolParlel
    | tolPerp
    | tolPos
    | tolProfl
    | tolProfp
    | tolProfs
    | tolRad
    | tolStrght
    | tolSym
    | tolTrnout
    | tolUsetol
    | tolWidth
    ;

tolAngl =
      ANGL, c, angle, c, angle
    ;

tolAnglb =
      ANGLB, c, angle, c, angle, c, angle, [c, tolZoneDir2]
    ;

tolZoneDir2 =
      XYPLAN
    | YZPLAN
```

```
    | ZXPLAN
    | VEC, c, vector
    ;

tolAnglwrt =
      ANGLWRT, c, angle, c, angle, c, angle, c, tolFeature, [c, tolZoneDir2]
    ;

tolAnglr =
      ANGLR, c, angle, c, tolAppData
    ;

tolCirlty =
      CIRLTY, c, rentVal
    ;

tolCompos1 =
       rentVal, [c, tolMatCond], 2*[c, tolFeatureMat], c, tolFeatureMat
    ;

tolCompos2 =
      rentVal, [c, tolMatCond], 3*[c, tolFeatureMat]
    ;

tolCompos =
      COMPOS, c, PATERN, c, tolCompos1, c, FEATUR, c, tolCompos2
    ;

tolFeatureMat =
      datLabel, [c, tolMatCond]
    | faLabel, [c, tolMatCond]
    | fLabel
    ;

tolConcen =
      CONCEN, c, rentVal, c, tolFeature
    ;

tolCortol =
      CORTOL, c, axis, c, rentVal, c, rentVal, [c, tolCortolVar3]
    | CORTOL, c, RADIAL, c, rentVal, c, rentVal
    | CORTOL, c, ANGLE, c, rentVal, c, rentVal
    ;

tolCortolVar3 =
      AXIAL, c, featureLabel, [c, rentVal]
    ;

tolCprofl =
      CPROFL, c, rentVal, c, rentVal, c, tolCprofEnd
    ;

tolCprofs =
      CPROFS, c, rentVal, c, rentVal, c, tolCprofEnd
    ;

tolCprofEnd =
       3*[tolFeatureMat, c], [AVGDEV, c], rentVal, c, rentVal,
      3*[c, tolFeatureMat], [c, AVGDEV]
    ;
```

```
tolCrnout =
      CRNOUT, c, rentVal, c, datLabel, 2*[c, tolFeature],
      [c, VEC, c, vector]
    ;

tolCylcty =
      CYLCTY, c, rentVal
    ;

tolDiam =
      DIAM, c, rentVal, c, rentVal, [c, radiusSpec], [c, AVG]
    | DIAM, c, rentVal, c, rentVal, [c, radiusSpec], c, MINMAX
    ;

avgMaxMin =
      c, AVG
    | c, MAX
    | c, MIN
    ;

tolDistb =
      DISTB, c, tolDistbTol, c, tolDistbDir, [avgMaxMin]
    ;

tolDistbDir =
      XAXIS
    | YAXIS
    | ZAXIS
    | PT2PT
    ;

tolDistbTol =
      tolDistbLimit
    | tolDistbNominl
    ;

tolDistbLimit =
      LIMIT, c, rentVal, c, rentVal
    ;

tolDistbNominl =
      NOMINL, c, rentVal, c, rentVal, c, rentVal
    ;

tolDistwrt =
      DISTWRT, c, tolDistbTol, c, tolFeature, c, tolDistbDir, [avgMaxMin]
    ;

tolFlat =
      FLAT, c, rentVal, [c, rentVal, c, rentVal, [c, rentVal]]
    ;

tolGtol =
      GTOL, c, tolGtolTrans, c, tolGtolRot, c, intVal, c,
      angle, c, angle, [c, tolGtolSpec]
    ;

tolGtolTrans =
      XDIR
```

```
    | YDIR
    | ZDIR
    | XYDIR
    | YZDIR
    | ZXDIR
    | XYZDIR
    | NOTRAN
    ;

tolGtolRot =
      XAXIS
    | YAXIS
    | ZAXIS
    | XYAXIS
    | YZAXIS
    | ZXAXIS
    | XYZAXI
    | NOROT
    ;

tolGtolSpec =
      PERCNT
    | INTFPT
    ;

tolAppData =
     rentVal, [c, tolMaxMatCond], c, tolFeatMaxMat,
     [c, tolFeatMaxMat], [c, tolZoneType], [c, tolZoneDir]
    ;

tolFeatMaxMat =
      datLabel, [c, tolMaxMatCond]
    | faLabel, [c, tolMaxMatCond]
    | fLabel
    ;

tolMaxMatCond =
      LMC, [c, MAX, c, rentVal]
    | RFS
    | MMC, [c, MAX, c, rentVal]
    ;

tolZoneDir =
      XAXIS
    | YAXIS
    | ZAXIS
    | VEC, c, vector
    ;

tolParlel =
      PARLEL, c, tolAppData
    ;

tolPerp =
      PERP, c, tolAppData
    ;

tolPos =
      tolPosStart, tolPosEnd
    ;
```

```
tolPosStart =
      POS, c, TWOD, c, rentVal
    | POS, c, THREED, c, rentVal
    ;

tolPosEnd =
      [c, tolMatCond], 3*[c, tolFeatureMat], [c, tolPosMethod]
    ;

tolPosMethod =
      XAXIS
    | YAXIS
    | ZAXIS
    | RADIAL
    | ANGLE
    | VEC, c, vector
    ;

tolProfl =
      PROFL, c, rentVal, c, rentVal, 3*[c, tolFeatureMat], [c, tolZoneDir2]
    ;

tolProfp =
      PROFP, c, rentVal, c, rentVal, 3*[c, tolFeatureMat]
    ;

tolProfs =
      PROFS, c, rentVal, c, rentVal, 3*[c, tolFeatureMat], [c, AVGDEV]
    ;

tolRad =
      RAD, c, rentVal, c, rentVal, [c, tolRadSpec], [c, MINMAX]
    ;

tolRadSpec =
      AVG
    | CRAD
    ;

tolZoneDir3 =
      XDIR
    | YDIR
    | ZDIR
    | VEC, c, vector
    ;

tolStrght =
      STRGHT, c, rentVal, c, tolMatCond, [c, axis], [c, tolZoneDir3]
    | STRGHT, c, rentVal, 2*[c, rentVal], [c, axis], [c, tolZoneDir3]
    ;

tolSym =
      SYM, c, rentVal, c, tolFeatureMat, [c, tolZoneDir3]
    ;

tolTrnout =
      TRNOUT, c, rentVal, c, datLabel, 2*[c, tolFeature]
    ;
```

```
tolUsetol =
      USETOL, c, stringVal
    ;

tolWidth =
      WIDTH, c, rentVal, c, rentVal, [c, tolWidthDir], [c, MINMAX]
    ;

tolWidthDir =
      vector
    | SHORT
    | LONG
    ;

tolFeature =
      fLabel
    | faLabel
    | datLabel
    ;

tolMatCond =
      MMC
    | LMC
    | RFS
    ;

tolZoneType =
      TANGPL
    | PARPLN
    ;

(*-------------------------------------------------------------*)
(* TOOLDF *)
(*-------------------------------------------------------------*)

tooldfStm =
      tlLabel, '=', TOOLDF, '/', mdLabel, c, stringVal, #
    ;

(*-------------------------------------------------------------*)
(* TRANS *)
(*-------------------------------------------------------------*)

transStm =
      dLabel, '=', TRANS, '/', transMinor, 2*[c, transMinor], #
    ;

transMinor =
      transValue
    | transFeature
    | transDatum
    | transProbe
    ;

transValue =
      orig, c, rentVal
    ;

transFeature =
      orig, c, featureLabel
```

```
    ;

transDatum =
      orig, c, datLabel
    ;

transProbe =
      orig, c, MINUSPRBRAD
    | orig, c, PRBRAD
    ;

(*-------------------------------------------------------------*)
(* UNCERTALG *)
(*-------------------------------------------------------------*)

uncertalgStm =
      uLabel, '=', UNCERTALG, '/', uncertMinor, #
    ;

uncertMinor =
      ALGOR, c, intVal
    | stringVal, c, uncertVar3
    ;

uncertVar3 =
      DME, [c, paramList]
    | SENS, [c, paramList]
    | paramList
    ;

(*-------------------------------------------------------------*)
(* UNCERTSET *)
(*-------------------------------------------------------------*)

uncertsetStm =
      UNCERTSET, '/', uncertsetMinor, #
    ;

uncertsetMinor =
      ON, c, uLabel, c, uncertsetVar2
    | OFF
    ;

uncertsetVar2 =
      drLabel
    | NONE
    ;

(*-------------------------------------------------------------*)
(* UNITS *)
(*-------------------------------------------------------------*)

unitsStm =
      UNITS, '/', lengthUnit, c, angleUnit, [c, tempUnit], #
    ;

lengthUnit =
      MM
    | CM
    | METER
```

```
    | INCH
    | FEET
    ;

angleUnit =
      ANGDEC
    | ANGDMS
    | ANGRAD
    ;

tempUnit =
      TEMPF
    | TEMPC
    ;

(*-------------------------------------------------------------*)
(* VALUE *)
(*-------------------------------------------------------------*)

valueStm =
      boolVar, '=', VALUE, '/', valueMinorBool, #
    | stringVar, '=', VALUE, '/', valueMinorString, #
    | intVar, '=', VALUE, '/', valueMinorInt, #
    | realVar, '=', VALUE, '/', valueMinorReal, #
    | vectorVar, '=', VALUE, '/', valueMinorVector, #
    ;

valueMinorBool =
      valueBadtst
    | valueCroscl
    | valueCzslct
    | valueFinpos
    | valuePrcomp
    | valuePtbuff
    | valueScnmod
    | valueTaBool
    | valueTecomp
    ;

valueMinorString =
      valueAclratString
    | valueBoundString
    | valueCrmode
    | valueCrslct
    | valueDatset
    | valueDmismd
    | valueDmismn
    | valueError
    | valueFedratString
    | valueFilnam
    | valueGeoalg
    | valueKeychar
    | valueMode
    | valueSnsetString
    | valueSnslct
    | valueUnits
    | valueWkplan
    ;

valueMinorInt =
```

```
      valueBoundInt
    | valueFaInt
    ;

valueMinorReal =
      valueAclratReal
    | valueDeflection
    | valueFedratReal
    | valueGotoReal
    | valuePtmeasReal
    | valueSnsetReal
    | valueFaReal
    | valueRt
    | valueSa
    | valueSw
    | valueTaReal
    ;

valueMinorVector =
      valueGotoVector
    | valuePtmeasVector
    | valueSnsmnt
    ;

valueAclratReal =
      ACLRAT, c, MESACL
    | ACLRAT, c, POSACL
    | ACLRAT, c, ROTACL
    ;

valueAclratString =
      valueAclratReal, c, ACEL
    ;

valueBadtst =
      BADTST
    ;

valueBoundString =
      BOUND, c, featureLabel, c, intVal
    | BOUND, c, toleranceLabel, c, intVal
    ;

valueBoundInt =
      BOUND, c, featureLabel, c, COUNT
    | BOUND, c, toleranceLabel, c, COUNT
    ;

valueCrmode =
      CRMODE
    ;

valueCroscl =
      CROSCL
    ;

valueCrslct =
      CRSLCT
    ;
```

```
valueCzslct =
      CZSLCT, c, czLabel
    ;

valueDatset =
      DATSET
    ;

valueDeflection =
      DEFLECTION
    ;

valueDmismd =
      DMISMD, c, NAME
    | DMISMD, c, VERSION
    ;

valueDmismn =
      DMISMN, c, NAME
    | DMISMN, c, VERSION
    ;

valueError =
      ERROR, c, ERR
    | ERROR, c, ERRMODE
    ;

valueFedratReal =
      FEDRAT, c, MESVEL
    | FEDRAT, c, POSVEL
    | FEDRAT, c, ROTVEL
    | FEDRAT, c, SCNVEL
    ;

valueFedratString =
      valueFedratReal, c, FEED
    ;

valueFilnam =
      FILNAM, c, NAME
    | FILNAM, c, VERSION
    ;

valueFinpos =
      FINPOS
    ;

valueGeoalg =
      GEOALG, c, measGeotype
    ;

valueGotoReal =
      GOTO, c, axis
    ;

valueGotoVector =
      GOTO, c, POS
    ;

valueKeychar =
```

```
      kcLabel
    ;

valueMode =
      MODE
    ;

valuePrcomp =
      PRCOMP
    ;

valuePtbuff =
      PTBUFF
    ;

valuePtmeasReal =
      PTMEAS, c, axis
    ;

valuePtmeasVector =
      PTMEAS, c, POS
    ;

valueScnmod =
      SCNMOD
    ;

valueSnsetReal =
      SNSET, c, APPRCH
    | SNSET, c, RETRCT
    | SNSET, c, SEARCH
    | SNSET, c, CLRSRF, c, DIST
    | SNSET, c, DEPTH, c, DIST
    ;

valueSnsetString =
      SNSET, c, CLRSRF
    | SNSET, c, DEPTH
    ;

valueSnslct =
      SNSLCT
    ;

valueSnsmnt =
      SNSMNT, c, XVEC
    | SNSMNT, c, ZVEC
    | SNSMNT, c, MNTLEN
    ;

valueTecomp =
      TECOMP
    ;

valueUnits =
      UNITS, c, DIST
    | UNITS, c, ANGL
    | UNITS, c, TEMP
    ;
```

```
valueWkplan =
      WKPLAN
    ;

valueFaInt =
      faLabel, c, PTDATA
    ;

valueFaReal =
      faLabel, c, SIZE, [c, intVal]
    ;

valueRt =
      rtLabel, c, ANGL, c, CW
    | rtLabel, c, ANGL, c, CCW
    ;

valueSa =
      saLabel
    | saLabel, c, stringVal
    | saLabel, c, intVal
    ;

valueSw =
      swLabel, c, ANGLE, c, stringVal
    ;

valueTaBool =
      taLabel, c, INTOL, [c, intVal]
    | taLabel, c, OUTOL, [c, intVal]
    ;

valueTaReal =
      taLabel, c, ACT, [c, intVal]
    | taLabel, c, DEV, [c, intVal]
    | taLabel, c, AMT, [c, intVal]
    ;

(*-------------------------------------------------------------*)
(* VFORM *)
(*-------------------------------------------------------------*)

vformStm =
      vLabel, '=', VFORM, '/', vformItemList, #
    ;

vformItem =
      NOM, [c, stringVal]
    | ACT, [c, stringVal]
    | DEV, [c, stringVal]
    | AMT, [c, stringVal]
    | HIST, [c, stringVal]
    | PLOT, [c, stringVal]
    | STAT, [c, stringVal]
    | ALL
    | DME, [c, stringVal]
    ;

vformItemList =
      [vformItemList, c], vformItem
```

```
   ;

(*-------------------------------------------------------------*)
(* WINDEF *)
(*-------------------------------------------------------------*)

windefStm =
      vwLabel, '=', WINDEF, '/', windefMinor, #
   ;

windefMinor =
      windefEdgeln
   | windefBox
   ;

windefEdgeln =
      EDGELN, c, point, c, angle, c, rentVal
   ;

windefBox =
      BOX, c, point, c, rentVal, c, rentVal, c, angle
   ;

(*-------------------------------------------------------------*)
(* WKPLAN *)
(*-------------------------------------------------------------*)

wkplanStm =
      WKPLAN, '/', plan, #
   ;

(*-------------------------------------------------------------*)
(* WRIST *)
(*-------------------------------------------------------------*)

wristStm =
      swLabel, '=', WRIST, '/', wristMinor, #
   ;

wristMinor =
      wristList, c, wristMountOffset, [c, wristDataDesc]
   ;

wristList =
      [wristList, c], wristDef
   ;

wristDef =
      wristMountDef, c, wristRotDef
   ;

wristMountDef =
      ROTCEN, c, vector, c, vector, c, vector
   ;

wristRotDef =
      ANGLE, c, stringVal, c, wristRotLimit, c, wristRotStep
   ;

wristRotLimit =
```

```
      angle, c, angle
    | THRU
    ;

wristRotStep =
      angle
    | CONTIN
    ;

wristMountOffset =
      MNTLEN, c, vector
    ;

wristDataDesc =
      stringVal, c, stringVal, c, stringVal
    | stringVal, c, stringVal, c, intVal
    ;

(*-------------------------------------------------------------*)
(* WRITE *)
(*-------------------------------------------------------------*)

writeStm =
      WRITE, '/', writeMinor, #
    ;

writeMinor =
      didLabel, c, writeSpecList
    ;

writeSpecList =
      [writeSpecList, c], writeSpec
    ;

writeSpec =
      stringConst
    | stringFunc
    | rwVar, [rwFormat]
    ;

(*-------------------------------------------------------------*)
(* XTERN *)
(*-------------------------------------------------------------*)

xternStm =
      XTERN, #
    ;

(*-------------------------------------------------------------*)
(* XTRACT *)
(*-------------------------------------------------------------*)

xtractStm =
      XTRACT, '/', fLabel, c, faLabel, #
    ;

(*-------------------------------------------------------------*)
(*-------------------------------------------------------------*)
(*--- End of productions for the Input Section ----------------*)
(*-------------------------------------------------------------*)
```

```
(*-----------------------------------------------------------------*)
```

Annex D
(informative)

Characterization file extensions

D.1 Machine dependent parameters

This section lists machine dependent parameters. Because of the large variety of DMEs, the following format provides for a basic listing of classifications, capacities, measuring volumes, and other parameters. The supplier of the DME is urged to provide a full and complete description of any machine dependent parameters not otherwise noted elsewhere. These additional characteristics or attributes, are to be consistent with ANSI/ASME B89 standard terminology where appropriate, and the major words listed below. The descriptions following the major words listed below, are of an open format, that is, concise, and direct descriptions.

CHFIL2 signifies the beginning of the machine dependent parameters section of the characterization file.

DMEACC Indicates the accuracy capabilities of the DME.

Format: DMEACC/volumetric accuracy, (consistent with B-89 standards where appropriate).

DMEAXS Indicates the number and types of axes, and their range of motion.

Format: DMEAXS/XAXIS,number,range

/YAXIS,number,range

/ZAXIS,number,range

Note: the CRGDEF statement identifies the measuring volume and ram axis for an individual sensor carrying carriage. 'Number' is the quantity of carriages in each axis and 'Range' is the total measuring range of the carriages in that axis.

DMECAL Indicates the type of supported sensor calibration.

Format: DMECAL/feature,(sphere,cube,etc.),routine,

algorithm,automatic,etc.

DMELMT Indicates the DME's processing limitations not otherwise noted.

Format: DMELMT/program size, other stack sizes, buffer limits, maximum number of points for evaluating roundness, CIRLTY, or cylindricity, CYLCTY, or runout tolerances, etc.

DMEREP indicates the repeatability of the DME.

Format: DMEREP/repeatability, (consistent with B-89 standards where appropriate).

DMEROT Indicates the rotary table's part holding limitations.

Format: DMEROT/diameter,length,height,weight

DMESTR Indicates the DME's capability to store data.

Format: DMESTR/maximum number of features that can be measured and stored for evaluation with tolerances at the end of a program, (Provides for production efficiencies).

DMESWL Indicates the DME software language, and version.

Format: DMESWL/language,version

DMETYP Indicates the DME type or machine classification.

Format: DMETYP/CMM,INFRED,LASER,VIDEO,number of sensor carrying carriages,etc.

DMEVOL Indicates the acceptable part measuring volume of the DME.

Format: DMEVOL/length,width,height

Note: the CRGDEF statement identifies the measuring volume and ram axis for an individual sensor carrying carriage.

DMEWGT Indicates the part weight capacity for the DME.

Format: DMEWGT/weight capacity and units.

MEAS Indicates the modes available for each supported MEAS option and the maximum number of points (n) that can be used.

Format: MEAS/FEATUR

feature,[mode(s)],n

.

.

.

ENDAT

Example: MEAS/FEATUR

ARC,[AUTO,PROG,MAN],100

CIRCLE,[PROG,MAN],1000

ENDAT

RAPID Indicates the minimum percentage of the DME's maximum speed.

Format: RAPID/PERCENTAGE

Percentage is represented in a decimal format.

ENDCH2 signifies the end of the machine dependent parameters section of the characterization file.

D.2 User defined options

This final section of the characterization file is provided to list additional characteristics of the DME's capabilities. Some examples may include several of the high level language extensions or the user defined options described in the standard. Users are urged to format these user defined characteristics in a manner similar to the format of the characterization file.

CHFIL3 signifies the beginning of the user defined options section of the characterization file.

(Supported user defined options.)

ENDCH3 signifies the end of the user defined options section of the characterization file.

Annex E
(informative)

Scanning reference

E.1 Introduction

Because of the major changes in DMIS to maximize the functionallity of scanning systems the previous examples of scanning measurement are no longer valid. The DMIS National Standards Committee is in the process of developing examples of scanning measurement with the new DMIS statements, but these are not complete at this time.

The DMIS National Standards Committee would welcome any examples that can be provided. Examples must be provided in either a plain text file or Microsoft Word (version 6.0/95 or later) format, any figures must be provided as Microsoft Word (version 6.0/95 or later) pictures. Examples should include descriptive text, valid DMIS code (where applicable) and descriptive figures wherever it will clarify the example.

Annex F
(informative)

Tolerance application

F.1 Application of TOL/ANGLB and TOL/ANGLWRT

Valid feature combinations to be used with TOL/ANGLB and TOL/ANGLWRT are shown in Table F.1 — Feature combinations for angle tolerances.

When another combination is desired, the features must be constructed.

The following example demonstrates a method to evaluate the angle between two tori axes by construction of the axes.

EXAMPLE

```
$$ Executable code
MODE/AUTO,PROG,MAN
F(tor1)=FEAT/TORUS,OUTER,CART,x,y,z,I,j,k,20,5
MEAS/TORUS,F(tor1),8
ENDMES
F(tor2)=FEAT/TORUS,OUTER,CART,x1,y1,z1,I1,j1,k1,30,7
MEAS/TORUS,F(tor2),8
ENDMES
$$ Define and construct the lines
F(li1)=FEAT/LINE,UNBND,CART,x,y,z,I,j,k,j,I,k
CONST/LINE,F(li1),MIDLI,FA(tor1),FA(tor1)
F(li2)=FEAT/LINE,UNBND,CART,x1,y1,z1,I1,j1,k1,j1,I1,k1
CONST/LINE,F(li2),MIDLI,FA(tor2),FA(tor2)
T(an0)=TOL/ANGLB,ang_inc,-.5,.5
OUTPUT/FA(li1),FA(li2),TA(an0)
$$ more executable code
```

F.2 Application of TOL/DISTB and TOL/DISTWRT

Valid feature combinations to be used with TOL/DISTB and TOL/DISTWRT are shown in Table F.2 — Feature combinations for distance tolerances.

When another combination is desired, the features must be constructed.

The following example demonstrates a method to evaluate the distance between two tori axes by construction of the axes.

EXAMPLE

```
$$ Executable code
MODE/AUTO,PROG,MAN
F(tor1)=FEAT/TORUS,OUTER,CART,x,y,z,I,j,k,20,5
MEAS/TORUS,F(tor1),8
ENDMES
F(tor2)=FEAT/TORUS,OUTER,CART,x1,y1,z1,I1,j1,k1,30,7
MEAS/TORUS,F(tor2),8
ENDMES
```

```
$$ Define and construct the lines
F(li1)=FEAT/LINE,UNBND,CART,x,y,z,I,j,k,j,I,k
CONST/LINE,F(li1),MIDLI,FA(tor1),FA(tor1)
F(li2)=FEAT/LINE,UNBND,CART,x1,y1,z1,I1,j1,k1,j1,I1,k1
CONST/LINE,F(li2),MIDLI,FA(tor2),FA(tor2)
T(di0)=TOL/DISTB,NOMINL,dist_b,-.5,.5
OUTPUT/FA(li1),FA(li2),TA(di0)
$$ more executable code
```

F.3 Tolerance application

Valid combinations of feature / tolerance application are shown in Table F.3 — Tolerance application.

As more examples of feature / tolerance application become available they will be included here.

F.3.1 Feature combinations for angle tolerances

Table F.1 — Feature combinations for angle tolerances

Aplicable to TOL/ANGLB and TOL/ANGLWRT		CPARLN	CYLNDER	CYLNDER-BND	CYLRADSEGMNT	ELLIPS	ELONGCYL	LINE	LINE-BND	PARPLN	PLANE	SYMPLN	TORRADSEGMNT	TORUS	REVSURF
		Line	Axis	Axis	Axis	Line	Plane	Line	Line	Plane	Plane	Plane	Plane	Plane	Axis
CPARLN	Line	LN-LN													
CYLNDER	Axis	LN-LN	LN-LN												
CYLNDER-BND	Axis	LN-LN	LN-LN	LN-LN											
CYLRADSEGMNT	Axis	LN-LN	LN-LN	LN-LN	LN-LN										
ELLIPS	Line	LN-LN	LN-LN	LN-LN	LN-LN	LN-LN									
ELONGCYL	Plane	PL-LN	PL-LN	PL-LN	PL-LN	PL-LN	PL-PL								
LINE	Line	LN-LN	LN-LN	LN-LN	LN-LN	LN-LN	PL-LN	LN-LN							
LINE-BND	Line	LN-LN	LN-LN	LN-LN	LN-LN	LN-LN	PL-LN	LN-LN	LN-LN						
PARPLN	Plane	PL-LN	PL-LN	PL-LN	PL-LN	PL-LN	PL-PL	PL-LN	PL-LN	PL-PL					
PLANE	Plane	PL-LN	PL-LN	PL-LN	PL-LN	PL-LN	PL-PL	PL-LN	PL-LN	PL-PL	PL-PL				
SYMPLN	Plane	PL-LN	PL-LN	PL-LN	PL-LN	PL-LN	PL-PL	PL-LN	PL-LN	PL-PL	PL-PL	PL-PL			
TORRADSEGMNT	Plane	PL-LN	PL-LN	PL-LN	PL-LN	PL-LN	PL-PL	PL-LN	PL-LN	PL-PL	PL-PL	PL-PL	PL-PL		
TORUS	Plane	PL-LN	PL-LN	PL-LN	PL-LN	PL-LN	PL-PL	PL-LN	PL-LN	PL-PL	PL-PL	PL-PL	PL-PL	PL-PL	
REVSURF	Axis	LN-LN	LN-LN	LN-LN	LN-LN	LN-LN	PL-LN	LN-LN	LN-LN	PL-LN	PL-LN	PL-LN	PL-LN	PL-PL	LN-LN

F.3.2 Feature combinations for distance tolerances

Table F.2 — Feature combinations for distance tolerances

Aplicable to TOL/DISTB and TOL/DISTWRT		ARC	CIRCLE	CPARLN	CYLNDER	CYLNDER-BND	CYLRADSEGMNT	EDGEPT	ELLIPS	ELONGCYL	LINE	LINE-BND	PARPLN	PLANE	POINT	REVSURF	RCTNGL	SPHERE	SPHRADSEGMNT	SYMPLN	TORUS	TORRADSEGMNT
		CtrPt	CtrPt	CtrPt	Axis	Axis	Axis	Point	CtrPt	CtrPl	Line	Line	Plane	Plane	Point	Axis	CtrPt	CtrPt	CtrPt	CtrPl	CtrPt	CtrPt
ARC	CtrPt	PT-PT																				
CIRCLE	CtrPt	PT-PT	PT-PT																			
CPARLN	CtrPt	PT-PT	PT-PT	PT-PT																		
CYLNDER	Axis	LN-PT	LN-PT	LN-PT	LN-LN																	
CYLNDER-BND	Axis	LN-PT	LN-PT	LN-PT	LN-LN	LN-LN																
CYLRADSEGMNT	Axis	LN-PT	LN-PT	LN-PT	LN-LN	LN-LN	LN-LN															
EDGEPT	Point	PT-PT	PT-PT	PT-PT	PT-LN	PT-LN	LN-PT	PT-PT														
ELLIPS	CtrPt	PT-PT	PT-PT	PT-PT	PT-LN	PT-LN	LN-PT	PT-PT	PT-PT													
ELONGCYL	CtrPl	PL-PT	PL-PT	PL-PT	PL-LN	PL-LN	PL-LN	PL-PT	PL-PT	PL-PL												
LINE	Line	LN-PT	LN-PT	LN-PT	LN-LN	LN-LN	LN-LN	LN-PT	LN-PT	PL-LN	LN-LN											
LINE-BND	Line	LN-PT	LN-PT	LN-PT	LN-LN	LN-LN	LN-LN	LN-PT	LN-PT	PL-LN	LN-LN	LN-LN										
PARPLN	Plane	PL-PT	PL-PT	PL-PT	PL-LN	PL-LN	PL-LN	PL-PT	PL-PT	PL-PL	PL-LN	PL-LN	PL-PL									
PLANE	Plane	PL-PT	PL-PT	PL-PT	PL-LN	PL-LN	PL-LN	PL-PT	PL-PT	PL-PL	PL-LN	PL-LN	PL-PL	PL-PL								
POINT	Point	PT-PT	PT-PT	PT-PT	PT-LN	PT-LN	LN-PT	PT-PT	PT-PT	PL-PT	PT-LN	PT-LN	PT-PL	PT-PL	PT-PT							
REVSURF	Axis	LN-PT	LN-PT	LN-PT	LN-LN	LN-LN	LN-LN	LN-PT	LN-PT	PL-LN	LN-LN	LN-LN	PL-LN	PL-LN	PT-LN	LN-LN						
RCTNGL	CtrPt	PT-PT	PT-PT	PT-PT	PT-LN	PT-LN	LN-PT	PT-PT	PT-PT	PL-PT	PT-LN	PT-LN	PT-PL	PT-PL	PT-PT	PT-LN	PT-PT					
SPHERE	CtrPt	PT-PT	PT-PT	PT-PT	PT-LN	PT-LN	LN-PT	PT-PT	PT-PT	PL-PT	PT-LN	PT-LN	PT-PL	PT-PL	PT-PT	PT-LN	PT-PT	PT-PT				
SPHRADSEGMTN	CtrPt	PT-PT	PT-PT	PT-PT	PT-LN	PT-LN	LN-PT	PT-PT	PT-PT	PL-PT	PT-LN	PT-LN	PT-PL	PT-PL	PT-PT	PT-LN	PT-PT	PT-PT	PT-PT			
SYMPLN	CtrPl	PL-PT	PL-PT	PL-PT	PL-LN	PL-LN	PL-LN	PL-PT	PL-PT	PL-PL	PL-LN	PL-LN	PL-PL	PL-PL	PL-PT	PL-LN	PT-PT	PT-PT	PT-PT	PL-PL		

TORUS	CtrPt	PT-PT	PT-PT	PT-PT	PT-LN	PT-LN	LN-PT	PT-PT	PT-PT	PL-PT	PT-LN	PT-LN	PT-PL	PT-PL	PT-PT	PT-LN	PT-PT	PT-PT	PT-PT	PL-PT	PT-PT	
TORRADSEGMNT	CtrPt	PT-PT	PT-PT	PT-PT	PT-LN	PT-LN	LN-PT	PT-PT	PT-PT	PL-PT	PT-LN	PT-LN	PT-PL	PT-PL	PT-PT	PT-LN	PT-PT	PT-PT	PT-PT	PL-PT	PT-PT	PT-PT

F.3.3 Tolerance application

Table F.3 — Tolerance application

	TOL/ANGL	TOL/ANGLB	TOL/ANGLWRT	TOL/ANGLR	TOL/CIRLTY	TOL/COMPOS	TOL/CONCEN	TOL/CORTOL	TOL/CPROFS	TOL/CRNOUT	TOL/CYLCTY	TOL/DIAM	TOL/DISTB	TOL/DISTWRT	TOL/FLAT	TOL/PARLEL	TOL/PERP	TOL/POS,2D	TOL/POS,3D	TOL/PROFL	TOL/PROFP	TOL/PROFS	TOL/RAD	TOL/STRGHT	TOL/SYM	TOL/TRNOUT	TOL/WIDTH
MEAS/ARC								XCP	x				XCP	XCP				XCP	XCP	x	XCP		x				
MEAS/CIRCLE					x		x	XCP	x	x		x	XCP	XCP				XCP	XCP	x	XCP						
MEAS/CONE	x	x	x	XCL	XCS		x	XV	x	XCS			x2	x2		XCL	XCL	XCL	XCL	X2D	XV	x		XCL		x	
MEAS/CONRADSEGMNT	x	x	x	XCL	XCS		x	XV	x	XCS			x2	x2		XCL	XCL	XCL	XCL	X2D	XV	x		XCL		x	
MEAS/CPARLN								x	x				x2	x2				XCP	XCP	x	XCP						x
MEAS/CYLRADSEGMNT		x	x	XCL	XCS		x		x	XCS	x		x1	x1		XCL	XCL	XCL	XCL	X2D		x		XCL		x	
MEAS/CYLNDR		x	x	XCL	XCS		x		x	XCS	x	x	x1	x1		XCL	XCL	XCL	XCL	X2D		x		XCL		x	
MEAS/EDGEPT								x	x				x	x				x	x		x	x					
MEAS/ELLIPS									x			x						x	x	x	XCP						
MEAS/ELONGCYL		x	x	x		x			x				x	x		x	x	x	x	x	XCP						x
MEAS/GCURVE									x											x							
MEAS/GSURF									x											X2D		x					
MEAS/LINE		x	x	x			x		x				x1	x1	x	x	x	x	x	x				x			
MEAS/PARPLN		x	x	x					x				x1	x1		x	x		x	X2D		x		XCL	x		x
MEAS/PATERN						x												x	x								
MEAS/PLANE		x	x	x					x	x			x1	x1	x	x	x		x	X2D		x		XCL		x	
MEAS/POINT								x	x				x	x				x	x		x						
MEAS/RCTNGL								x	x				x	x				x	x								x

	TOL/ANGL	TOL/ANGLB	TOL/ANGLWRT	TOL/ANGLR	TOL/CIRLTY	TOL/COMPOS	TOL/CONCEN	TOL/CORTOL	TOL/CPROFS	TOL/CRNOUT	TOL/CYLCTY	TOL/DIAM	TOL/DISTB	TOL/DISTWRT	TOL/FLAT	TOL/PARLEL	TOL/PERP	TOL/POS,2D	TOL/POS,3D	TOL/PROFL	TOL/PROFP	TOL/PROFS	TOL/RAD	TOL/STRGHT	TOL/SYM	TOL/TRNOUT	TOL/WIDTH
MEAS/REVSURF		x	x	XCL			x		x	XCS			x1	x1		XCL	XCL	XCL	XCL	X2D		x		XCL		x	
MEAS/SPHERE					x (sphericity)		x	XCP	x	XCS		x	XCP	XCP				x	x	X2D	XCP	x				x	
MEAS/SPHRADSEGMNT					x (sphericity)		x	XCP	x	XCS			XCP	XCP				x	x	X2D	XCP	x				x	
MEAS/SYMPLN		x	x	x					x				x1	x1		x	x		x	X2D		x		XCL	x		x
MEAS/TORUS							x	x	x	XECS		x	X3CP	X3CP				XCP	XCP	X2D	XCP	x				x	
MEAS/TORRADSEGMNT							x	x	x	XECS			X3CP	X3CP				XCP	XCP	X2D	XCP	x				x	

This table covers only tolerances as defined by ASME Y14.5 and ISO 1101 plus their applicable features

For TOL / DISTB and TOL/DISTWRT refer to (Table F.2 — Feature combinations for distance tolerances).

x1: Refer to (Table F.2 — Feature combinations for distance tolerances) for details.

x2: Refer to (Table F.2 — Feature combinations for distance tolerances) for details.

x3: Refer to (Table F.2 — Feature combinations for distance tolerances) for details.

XCP = CenterPoint
XCS = Crosssections
XCL = CenterLines
XV = Vertexonly
XECS = Cross section parallel to equator
CL = Centerline
CP = Center point
X2D = Planar Crosssections

Note 1: This table does not yet include possible datums.

Note 2:This list defines combinations of features and tolerances in order to avoid ambiguities when exchanging DMIS files.

Note 3: This list defines best practice combinations of features and tolerances after reviewing ISO and ASME standards

	Note 4: TOL/CPROFS is under review by the committee and is subject to further refinement
TOL/ANGL	
TOL/ANGLB	
TOL/ANGLWRT	
TOL/ANGLR	
TOL/CIRLTY	
TOL/COMPOS	
TOL/CONCEN	
TOL/CORTOL	
TOL/CPROFS	
TOL/CRNOUT	
TOL/CYLCTY	
TOL/DIAM	
TOL/DISTB	
TOL/DISTWRT	
TOL/FLAT	
TOL/PARLEL	
TOL/PERP	
TOL/POS,2D	
TOL/POS,3D	
TOL/PROFL	
TOL/PROFP	
TOL/PROFS	
TOL/RAD	
TOL/STRGHT	
TOL/SYM	
TOL/TRNOUT	
TOL/WIDTH	

Annex G
(informative)

Deleted statements

G.1 Introduction

This annex presents a listing of DMIS statements that have been removed from the standard. Prior to this version when the committee voted to remove a statement from the DMIS standard it was listed in Forward as a significant change with text stating that it would be removed in the next major release of the standard. This creates ambiguity when new statements conflict with statements that will be removed, but are still in the standard until the next major release.

G.1.1 Listing order

Statements will be listed in the order that they are removed.

G.2 DNSC Oak Ridge, Tenn. U.S.A. August 1998

Statements removed at the DNSC meeting in Oak Ridge, Tenn. U.S.A. August 1998.

G.2.1 RADIUS

For the statements TOL/CORTOL and TOL/POS the minor word RADIUS has been replaced with RADIAL. The minor word RADIUS was a typographical error. The RADIUS and RADIAL minor words have the same definition in this case.

G.3 DNSC Auburn Hills, Mich. U.S.A. March 1999

Statements removed at the DNSC meeting in Auburn Hills, Mich. U.S.A. March 1999.

G.3.1 EXTFIL

The SYS minor word has been removed for the next major release of this standard.

G.4 DNSC Markham, Ont. Canada July 1999

Statements removed at the DNSC meeting in Markham, Ontario Canada July 1999.

G.4.1 SNSDEF (input format 6) Defining an XRAY sensor

As part of the sensor discussion it was decided that the XRAY sensor type would be removed in the next version after version 04.0.

G.5 DNSC Orlando, Florida February 2003

Statements removed at the DNSC meeting in Orlando, Florida U.S.A. February 2003.

G.5.1 SCAN

The SCAN statement, formerly 6.151 is no longer needed in the DMIS language. A more defined method of scanning has been accepted that uses a path directed scan. The new scanning method utilizes the PATH statement to define a scanning path and the PAMEAS statement to perform the scan.

G.5.2 SCNPLN

The SCNPLN statement, formerly 6.153 is no longer needed in the DMIS language. With a path directed scan the scanning plane is incorporated into the definition of the path.

G.6 DNSC Brighton, Mich. U.S.A. October 2003

Statements removed at the DNSC meeting in Brighton, Michigan U.S.A. October 2003.

G.6.1 PATERN

As part of adding the new statement FEAT/COMPOUND and CONST(Input Format 14) the minor word PATERN was removed from the MEAS and CONST(Input Format 1) statements.

G.7 DNSC Arlington, Texas U.S.A. April 2004

Statements removed at the DNSC meeting in Arlington, Texas U.S.A. April 2004.

G.7.1 CALL

The SPAWN minor word has been removed from the CALL statement for the next major release of this standard.

G.8 DSC Troy, Michigan U.S.A. April 2009

Statements removed at the DSC meeting in Troy, Michigan U.S.A. April 2009.

G.8.1 GECOMP

The GECOMP statement no longer has value and has been removed from the standard.

Index of statements

Index of figures

Index of tables

Index of statements by type

Branching and looping statements

Carriage statements

Datum statements

Feature statements

Feature construction statements

File and machine parameter statements

In process verification / quality information system statements

Input / output statements

Macro statements

Measurement statements

Miscellaneous statements

Motion statements

Program flow statements

Rotary table statements

Sensor statements

Scanning statements

Tolerance statements

Variable statements

ICS 25.040.40

Price based on 478 pages