

Tutor Translator

PC-DMIS™ Enhancement for Versions 3.5 and Above

by

Wilcox Associates, Inc.

<http://www.wilcoxassoc.com>

Last Updated: Aug 11, 2005

Table Of Contents

Tutor Translator for PC-DMIS	1
Introduction	1
Functionality.....	1
Architecture.....	1
PC-DMIS Settings.....	4
Part Program Structure.....	4
Tutor to PC-DMIS Translation User Interface.....	6
TutorPCDmis Translator options.....	7
Translation in Progress.....	8
Probe Mapping	8
Rules of Translation	10
PC-DMIS Program Translations	10
Program Declaration.....	10
Variable Declaration	12
Functions	13
Arithmetical Operators	13
Vector Operators	14
Relational and Logical Operators.....	14
Probe Management Commands.....	14
Load Qualification File	14
Save Probe File	15
Alignment Commands	16

Tutor Translator

Skew1 – Alignment Secondary Axis	16
Skew2 – Alignment Secondary Axis	17
Iskew - Theoretical Rotation	18
Preset – Origin Presetting.....	19
Use of Reference Systems	20
Use of Reference Planes.....	21
Measurement Commands	22
Measurement Commands with no PC-DMIS Equivalent	23
Measurement Commands that may have a PC-DMIS Equivalent	23
Measurement Command Conditions	23
Example Circle Command	24
Circle Measurement Commands	25
Cone Measurement Commands	27
Cylinder Measurement Command	28
Ellipse Measurement Commands.....	30
Line Measurement Commands.....	31
Middle Point Measurement Commands.....	33
Paraboloid Measurement Commands	34
Pick (Uncompensated Point) Measurement Commands	35
Plane Measurement Commands	35
Three Level Plane Measurement Commands.....	36
Shoulder (Compensated Point) Measurement Commands	37
Slot Measurement Commands	41

Table Of Contents

Sphere Measurement Commands	43
Thickness Measurement Commands	43
Torus Measurement Commands	44
Canned Circle Measurement Commands.....	45
Construction Commands	50
Construction Commands with no PC-DMIS Equivalent	51
Circle Construction Commands	51
Cone Construction Commands.....	57
Cylinder Construction Commands	59
Line Construction Commands.....	61
Middle Point Construction Commands.....	63
Paraboloid Construction Commands	64
Plane Construction Commands	64
Three Level Plane Construction Commands	65
Slot Construction Commands	68
Sphere Construction Commands.....	68
Thickness Construction Commands	70
Torus Construction Command.....	70
RIDPT Construction Command	71
Relationship Between Elements.....	71
Projection Commands	72
Intersection Commands	74
Middle Relation Commands.....	81

Tutor Translator

Distance Relation Commands	83
CIRCLE_BY_CONE Relation Command.....	85
Geometric Tolerances	86
Orientation Tolerances	87
Concentricity, Coaxiality, and Symmetry Tolerances.....	91
True Position Tolerances	96
APPROACH / NO_APPROACH Commands	107
PL3L_DISTANCE Command	107
Element Comment.....	107
Element Evaluation	108
DELAY Command.....	108
EMERGENCY Command.....	109
ASSIGNMENT Commands	109
INTVAR as an integer variable	109
REALVAR as an integer variable.....	109
BOOLVAR as an integer variable	110
STRINGVAR as an integer variable.....	110
THEO Command.....	111
Output Control	111
Output Basics	112
Output to Screen.....	113
Output to Printer	115
Input / Output on Serial Line	116

Table Of Contents

Output to File	116
Text File Management.....	119
Output Protocol and Output Format	122
Format Command.....	122
Block Commands.....	123
Dimension Mode Commands.....	124
THEO Commands	125
Input/Output Parameters	126
Loading/Saving Output Formats	126
Element Evaluation	127
Element Buffer Management.....	127
Movement Commands	130
Motion Control.....	131
Movement Parameters	132
System Functions.....	133
Date Function	133
Time Function	134
Type Function.....	134
Info Function.....	134
Current tip radius commands.....	134
Current reference system number command.....	135
Current Approach command.....	135
Current Block Number command.....	135

Tutor Translator

Current Current SelPl command	135
Current Mspeed command	135
Current Pspeed command.....	135
Current Fly Mode command	136
Ph9 present command.....	136
CMM Type command	136
Depth command	136
Theoretical value commands	136
Theoretical value management:.....	137
Play Function	137
Exec Function	138
SYSTEM_INFO Function.....	139
Read I/O channels command	140
Write I/O channels commands.....	140
GET_POSITION Command.....	140
GETDIR Command.....	140
INCR Command	141
DECR Command.....	141
Scanning Control.....	141
Flow Control	143
Unconditional	143
Conditional.....	143
Logical Functions	145

Table Of Contents

EOF Functions.....	145
EOL Functions	146
FOUND Functions	147
ENCODE Functions.....	150
DECODE Functions.....	155
OOT Function	160
CRITICAL Function.....	160
BADMEAS Function	161
Emergency Control.....	161
NO_EME_BDMS	161
EME_BDMS.....	162
Repetition of Commands.....	162
FOR Commands.....	162
LOOP Command	166
EXIT Command	166
EXIF Command	166
Internal Procedure Call.....	167
External Procedure Call	167
Procedures.....	168
Procedure Variables	168
Procedure Declaration	169
END Procedure Declaration.....	169
Core Functions	169

Tutor Translator

BIGGEST.....	169
Tutor Simulation Subroutines	170
Look Ahead Procedure.....	170
Problems in Developing the Tutor Translator	171
Theoretical value management:.....	171
Feature evaluation and output	172
Probe management	173
Limitations	173
Index	177

Tutor Translator for PC-DMIS

Introduction

The purpose of the Tutor Translator for PC-DMIS is to give clients using Tutor the ability to easily convert their Tutor part program into a part program for PC-DMIS.

The converted part programs will provide the same results of the Tutor part programs. Also, the file provides a base for learning the PC-DMIS language. This file is created to be easy to comprehend and modify.

Because of the notable difference of the structure of the two languages it is not possible to translate all the part programs and is therefore necessary to identify some limits to the performance of the translator. See the chapter "[Limitations](#)" for a list of the limitations.

Tutor will need to be executed for part programs that are unable to translate to PC-DMIS.

Functionality

Tutor Translator does not directly create a PC-DMIS part program, but instead creates one or more Basic Script files. Translation is performed in 2 main steps:

1. Translation of Tutor commands to BASIC commands that are stored in one or more basic script files
2. The resulting script files are automatically executed by BASIC interpreter that populate the current PC-DMIS file with commands.

Architecture

The Tutor-PC-DMIS translator is composed of three components:

[Basic script generator](#).

[Basic script utility subroutines collection file](#).

[PC-DMIS Tutor emulation subroutines collection](#).

BASIC script generator

The BASIC script generator is composed by several Borland Delphi Pascal DLLs that are derived from TutoRun compile and pretty printing modules.

The main activities of this component are:

1. Compiles Tutor source code.
2. Parses the compiled code to generate the basic script commands.

Tutor Translator

3. Splits the basic script commands in subroutines with at least 1000 lines of code.
4. Splits the basic script file into more script files if the number of subroutines is more than 6.
5. Extracts the utility subroutines collection from the Basic script and the subroutines that are recalled by the script.
6. Asks the operator for the data information (Option Dialog):
 - a. PC-DMIS probe file name to be used with translated part program.
 - b. Tutor-PC-DMIS probe map: this is a map that correlates the Tutor probe IDs (Head/Tip) with the PC-DMIS probe names.
 - c. The tip radius to be used in the evaluation of certain features.
 - d. Tutor True Position, Linearity and Form Error modes.
7. Warns the operator about commands that:
 - a. Cannot be translated.
 - b. Are partially translated.
 - c. The translate command could behave in an unexpected manner.
8. Record all warnings in log file.

BASIC script utility subroutines collection

The BasicUtil.bas file is supplied with PC-DMIS and contains predefined basic subroutines. The purpose of these subroutines is to avoid large code duplication or to create predefined sections of the PC-DMIS part program.

The subroutines in this module are:

1. BeginMainScript: Initializes the main script variables and creates some PC-DMIS Global variables.
2. BeginAuxScrip: Initializes the secondary scripts variables
3. MakeFunctions: creates function variables.
4. MakeProjCmd: Expands the translation of Tutor Projection commands.
5. MakeIntersCmd: Expands the translation of Tutor Intersection commands.
6. MakeMiddleCmd: Expands the translation of Tutor Middle commands.
7. MakeDistCmd: Expands the translation of Tutor Distance (2D/3D) commands.
8. MakeOrientTolCmd: Expands the translation of Tutor Orientation Geometrical Tolerance commands. Parallelism, Squareness and Angularity)
9. MakeConcentCmd: Expands the translation of Tutor Concentricity commands.
10. MakeCoaxCmd: Expands the translation of Tutor Coaxially commands.
11. MakeSymmCmd: Expands the translation of Tutor Symmetry commands.
12. MakeTPCommand: Expands the translation of Tutor True Position commands.(RFS, MMC1, MMC2, LMC1, LMC2)
13. AskFeaType: Asks the operator to choice a feature type. Used by subroutines from 4 to 11 know the type of the feature involved in the expanding command when it is not possible to retrieve it.
14. Make_Exec: Expands the translation of an external application execution.

Each created script will only include subroutines used in that script.

PC-DMIS Tutor emulation subroutines collection

Many Tutor functionalities are emulated with PC-DMIS subroutines. They are available within an auxiliary part program that is supplied with PC-DMIS.

Supplied subroutines are:

1. Probe management:
 - a. ERASE_PROBE_SUB: Probe 0 emulation
2. Reference management:
 - a. SAVE_REFSYS_SUB: Save Refsys file emulation
 - b. LOAS_REFSYS_SUB: Load Refsys file emulation. **This subroutine doesn't work in current PCDMIS release.**
3. Theoretical values management:
 - a. THEO_INIT: Initialize Tutor theoretical value table emulation
4. Output management:
 - a. INIT_FORMAT_TBL: Tutor output format table initialization
 - b. TUTOR_DEV_ENDIS: Enable/Disable output device emulation (Dy,NoDy,Prn/NoPrn,File/Nofile)
 - c. TUTOR_FORMAT: Emulates Tutor Format command
 - d. TUTOR_OUTPUT: Emulates Tutor evaluation output to enabled devices
 - e. TUTOR_OUTSTRING: Emulates Tutor string output to enabled devices
 - f. TUTOR_HEADER and TUTOR_LINEHEADER: Emulate Tutor Header Command
 - g. TUTOR_DY: Emulates Tutor DY (string) Command.
 - h. TUTOR_PRN: Emulates Tutor PRN (string) Command.
 - i. TUTOR_FILE: Emulates Tutor FILE (string) Command.
 - j. TUTOR_OPEN: Emulates Tutor OPEN (File) Command.
 - k. TUTOR_CLOSE: Emulates Tutor CLOSE (File) Command.
 - l. TFSCAN_WRITE: Emulates TFSCAN File write (.SNC file only)
 - m. LOAD_FORMAT: Emulates Tutor Load Format command. In current version it does nothing.
 - n. SAVE_FORMAT: Emulates Tutor Save Format command. In current version it does nothing.
 - o. TUTOR_TYPE: Emulates Tutor TYPE Command.
5. Element management:
 - a. LOAD_ELEMENT: Emulate Tutor Load Element command. In current version it does nothing.
 - b. SAVE_ELEMENT: Emulate Tutor Load Element command. In current version it does nothing.
 - c. TUTOR_IO: Emulates Tutor I/O Channels (SISTEM_INFO(IO,WORD1,WORD2))

PC-DMIS Settings

Some PCDMIS settings are required in order to properly run the translated part program:

Setup options:

The following Setup options must be set:

- Treat theo values as if stored in part coordinates.
- Ignore CAD to Part.

The following Setup options must be reset:

- Reset global settings when branching.

Edit Windows Layout settings:

The following Report settings must be set:

- Show Comments.

The following Report settings must be reset:

- Show Features
 - Show Alignments
 - Show Moves
 - Show Header / Footer
 - Show Screen Captures
-

Part Program Structure

The structure of a translated PC-DMIS part program is outlined below:

1. Machine Reference Systems Declaration

```
STARTUP      = ALIGNMENT/START, RECALL:, LIST= YES  
ALIGNMENT/END
```

2. Default Functions Definition

```
ASSIGN/E.OFF2ROT = FUNCTION((X,Y),IF(X==0,IF(Y>0,90,90),RAD2DEG(ATAN(Y/X))))  
  
ASSIGN/E.BIGGEST =  
FUNCTION((F),IF(ABS(F.I)>ABS(F.J),IF(ABS(F.K)>ABS(F.I),IF(F.K>0,E.ZPLUS,E.ZMINUS),  
IF(F.I>0,E.XPLUS,E.XMINUS)),IF(ABS(F.K)>ABS(F.J),IF(F.K>0,E.ZPLUS,E.ZMINUS),  
IF(F.J>0,E.YPLUS,E.YMINUS))))
```

```

ASSIGN/E.CALC_AXY = FUNCTION((V1,V2),
ANGLEBETWEEN(UNIT(MPOINT(V1.I,V1.J,0)),UNIT(MPOINT(V2.I,V2.J,0)))))

ASSIGN/E.CALC_AYZ = FUNCTION((V1,V2),
ANGLEBETWEEN(UNIT(MPOINT(0,V1.J,V1.K)),UNIT(MPOINT(0,V2.J,V2.K)))))

ASSIGN/E.CALC_AZX = FUNCTION((V1,V2),
ANGLEBETWEEN(UNIT(MPOINT(V1.I,0,V1.K)),UNIT(MPOINT(V2.I,0,V2.K)))))

ASSIGN/E.BOOLVAL = FUNCTION((BVAL),IF(BVAL<>0,"TRUE","FALSE"))

ASSIGN/E.PMS2DEG = FUNCTION((VAL),INT(VAL)+INT((VAL-INT(VAL))*100)/60+ (((VAL-
INT(VAL))*100)-INT((VAL-INT(VAL))*100))*100/3600)

ASSIGN/E.DMS2DEG = FUNCTION((VAL,ISDMS),IF((ISDMS),VAL,IF((VAL>=0),
E.PDMS2DEG(VAL), -E.PDMS2DEG(-VAL))))

```

3. Part Program Declaration

```

PROGRAM_PrgName = LABEL/
ASSIGN/E.WM1 = TUTORELEMENT (1)
ASSIGN/E.WM2 = TUTORELEMENT (1)
ASSIGN/E.MEMORY = TUTORELEMENT (300)
ASSIGN/E.MEMSIZE = 300
ASSIGN/E.SELPL=E.ZPLUS

```

4. Variables Declaration

```

. . .
ASSIGN/BoolVar = 0
ASSIGN/BoolVar = 1
ASSIGN/CoordVar = MPOINT(0,0,0)
. . .

```

5. Tutor Output Format Table Structure Initialization

```

ASSIGN/E.OFMT.ON = 1
CS3 =CALLSUB/INIT_FORMAT_TBL,:E.OFMT.,

```

6. Tutor Theoretical Table Structure Initialization

```

ASSIGN/E.TH.X.N = 0
=CALLSUB/THEO_INIT,TutorSub.prg:E.TH,,,

```

7. Probe Structures Initialization

ASSIGN/PROBE_SP=0

```

ASSIGN/E.PROBE_STACK=0
LOADPROBE/PROBEFILENAME
=CALLSUB/ ERASE_PROBE_SUB,TutorSub.PRG:0,0,E.TIPARRAY,
ASSIGN/E.TipArray[1,1] = "T1A0B0"
ASSIGN/E.TipArray[2,1] = "T1A45B0"
ASSIGN/E.TipArray[3,1] = "T1A90B0"
ASSIGN/E.TipArray[4,1] = "T1A90B45"
ASSIGN/E.TipArray[5,1] = "T1A90B90"
ASSIGN/E.TipArray[6,1] = "T1A45B45"
ASSIGN/E.TipArray[7,1] = "T1A-90B45"
. . .

```

Tutor Translator

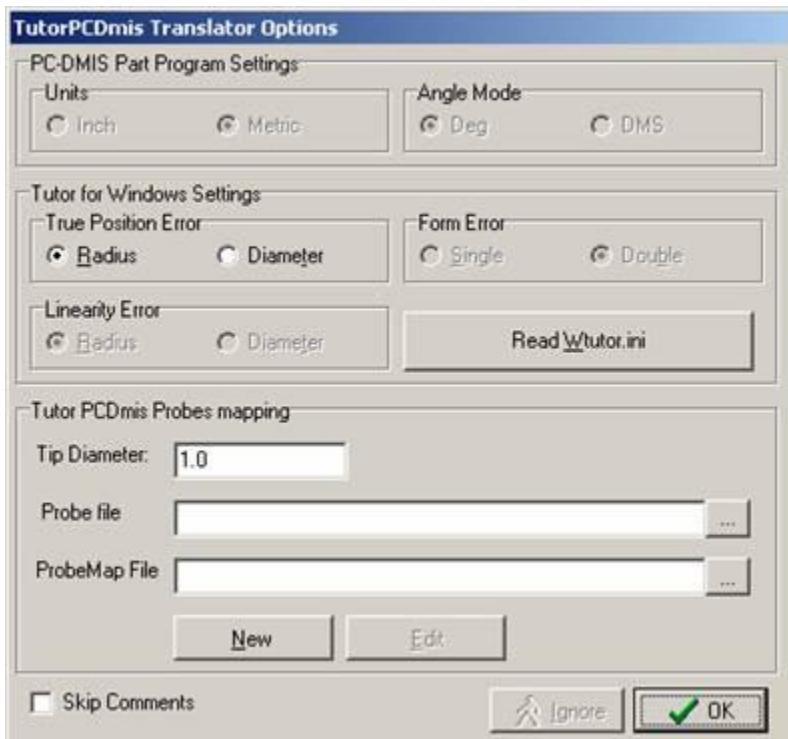
8. Alignments Structures Initialization

```
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN = 0
ASSIGN/E.REF_SP = 0
ASSIGN/E.ALIGNMENTS[ 0 ] = " STARTUP "
ASSIGN/E.ALIGNMENTS[ 1 ] = " STARTUP "
ASSIGN/E.ALIGNMENTS[ 2 ] = " STARTUP "
ASSIGN/E.ALIGNMENTS[ 3 ] = " STARTUP "
ASSIGN/E.ALIGNMENTS[ 4 ] = " STARTUP "
ASSIGN/E.ALIGNMENTS[ 5 ] = " STARTUP "
ASSIGN/E.ALIGNMENTS[ 6 ] = " STARTUP "
ASSIGN/E.ALIGNMENTS[ 7 ] = " STARTUP "
ASSIGN/E.ALIGNMENTS[ 8 ] = " STARTUP "
ASSIGN/E.ALIGNMENTS[ 9 ] = " STARTUP "
ASSIGN/E.ALIGNMENTS[ 10 ] = " STARTUP "
```

9. Part Program Body

Tutor to PC-DMIS Translation User Interface

The process of importing a Tutor part program is achieved through a series of three different dialog boxes. To start the process select **File | Import | Tutor...** This opens the TutorPCDmis translator Option dialog box.



TutorPCDmis Translator Options dialog box

TutorPCDmis Translator options

This dialog box allows you to determine how items in the Tutor file should be interpreted or displayed inside PC-DMIS. The following groups of options are available:

PC-DMIS Part Program Settings:

These options come from the PC-DMIS part program into which the Tutor file is being imported and cannot be modified:

Units – Shows the units of measurement (either **Inch** or **Metric**) of the PC-DMIS part program. Unlike Tutor for Windows, PC-DMIS does not allow you to change measurement units at execution time.

Angle Mode – Shows how angles are currently displayed (either **Deg** or **DMS**) in the PC-DMIS part program.

Tutor for Window Settings:

PC-DMIS uses Tutor for Windows configuration files to correctly translate Tutor files. Tutor for Windows configurations are stored in the **WTutor.ini** configuration file. When translation starts the wTutor.ini file is searched in the c:\Winnt folder.

The user can force translator to read another Tutor configuration file using the “**Read WTutor.ini**” button.

True Position Error - Select **Radius** if the importing part program requires the True Position error evaluated as radius, otherwise select **Diameter**.

Form Error - Select **Double** if the importing part program requires the Double Form error, otherwise select **Single**.

Linearity Error - Select **Radius** if the importing part program requires the Linearity error evaluated as radius, otherwise select **Diameter**.

Tutor PC-DMIS Probes Mapping:

Tip Diameter: Type the default tip diameter for the probe used by Tutor for Windows.

Probe File: Use the ... button to choose a PC-DMIS probe file (.PRB) to use.

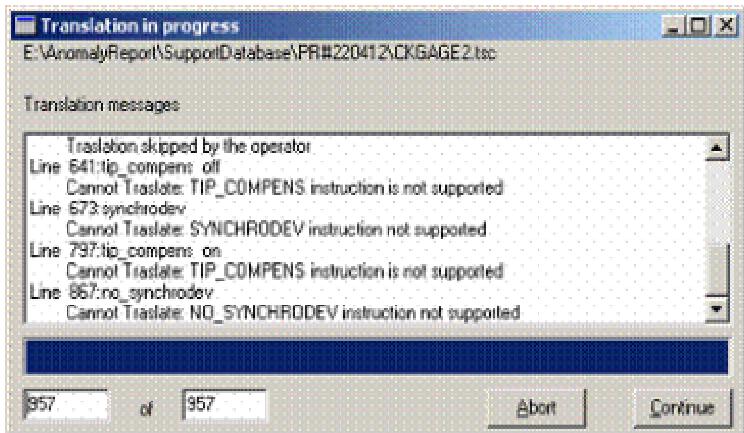
ProbeMap File: Use the ... button to choose an existing Tutor to PC-DMIS probe map file. If you don't have a probe map file, you can create one by clicking the **New** button. The **Edit** button allows you to change the currently selected probe map file listed in this box. This map file allows you to tell the translator what PC-DMIS tip angles it should use when translating probe commands from the Tutor file into PC-DMIS. For more information, see “[Probe Mapping](#)”.

Skip Comments: Select this check box if you don't want to import programmer comments found in the Tutor file.

Tutor Translator

Translation in Progress

The **Translation in progress** dialog box displays the import process. In the **Translation messages** list box all warning messages are given. Upon translation completion the operator can inspect the translation messages and decide to continue or to abort the importing session.



Translation in progress

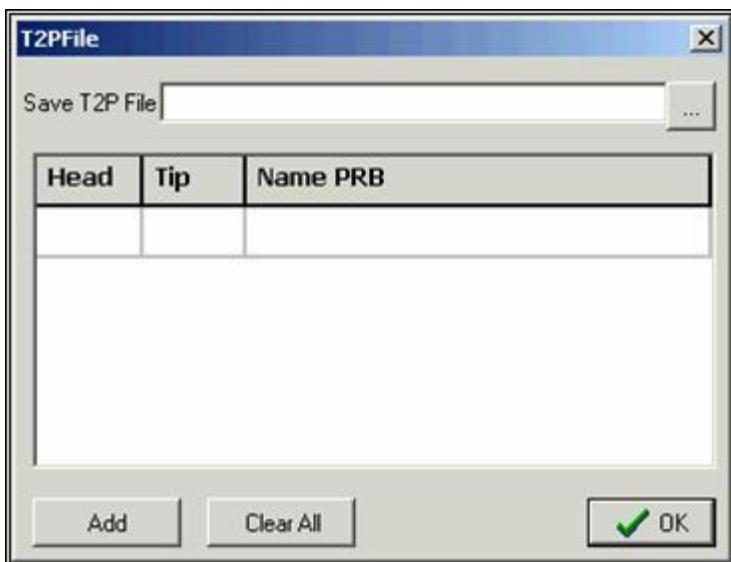
Translation messages are stored in a log file. The name of this file is the name of Tutor part program with .log extension and is located in the same folder of Tutor part program.

For example: When importing the file C:\WTutor\Prog\Part1029.tsc the resulting log file would be C:\WTutor\Prog\Part1029.log.

Probe Mapping

To properly translate programs from Tutor into PC-DMIS, you will need a probe map file that relates the Tutor file's probe definition with PC-DMIS probe definition. A Tutor's probe command is identified by two numbers that refer to a specific head and tip respectively. The T2P File dialog lets the operator to create or modify the probe map file.

To open the T2P dialog box click the **New** or **Edit** button in the **TutorPCDMIS Translator Options** dialog box.



T2PFile dialog box

The **Add** button allows the operator to specify more information adding a new row. For each row the following values are specified:

- **Head** - Identifies the probe head in Tutor (values 1 through 100).
- **Tip** - Identifies the probe head in Tutor (values 1 through 5).
- **Name PRB** - Contains the PC-DMIS probe tip angle to be used for the Head and Tip of that row.

For example, the probe map information in the following table tells the Tutor translator to use PC-DMIS's T5A30B30 probe tip angle when it encounters a probe command of 2,2 in the Tutor file.

Head	Tip	Name PRB
2	2	T5A30B30

The **Clear All** button clears all information from the dialog box.

The new or modified Tutor to PC-DMIS probe map file is saved by clicking the **OK** button.

Rules of Translation

Original Tutor commands are added inside of comments at the beginning of each translation block.

For example:

The Tutor command

PUSH_REF SYS translates to...

```
COMMENT/DOC, =====  
, = PUSH_REF SYS
```

If the command cannot be translated the comment is:

```
COMMENT/OPER, =====  
, = Unable to translate.  
, = PUSH_REF SYS
```

PC-DMIS Program Translations

Throughout this chapter examples are given of the PC-DMIS equivalent commands derived from the Tutor translation.

Program Declaration

The command PROGRAM PrgName[E.WM1, E.WM2] it is translated with a label and the declaration of two element variables in the first valid position of the part program after the initial command:

```
STARTUP = ALIGNMENT/START,RECALL:, LIST = YES  
          ALIGNMENT/END  
PROGRAM_PrgName = LABEL/  
ASSIGN/E.WM1 = TUTORELEMENT (1)  
ASSIGN/E.WM2 = TUTORELEMENT (1)
```



The names MEMORY,E.WM1 and E.WM2 can be replaced by others defined by the Tutor programmer.

Flat length initialization:

if the PC-DMIS part program is in metric mode.

```
ASSIGN/E.FLYLEN=30
```

if the PC-DMIS part program is in inch mode.

```
ASSIGN/E.FLYLEN=1.1811
```

Text file structure initialization:

```
ASSIGN/E.F0.PTR = "F0PTR"
ASSIGN/E.F0.NAME = "NONE"
ASSIGN/E.F0.OPEN = "NO"
ASSIGN/E.F1.PTR = "F1PTR"
ASSIGN/E.F1.NAME = "NONE"
ASSIGN/E.F1.OPEN = "NO"
ASSIGN/E.F2.PTR = "F2PTR"
ASSIGN/E.F2.NAME = "NONE"
ASSIGN/E.F2.OPEN = "NO"
ASSIGN/E.F3.PTR = "F3PTR"
ASSIGN/E.F3.NAME = "NONE"
ASSIGN/E.F3.OPEN = "NO"
ASSIGN/E.SELPL=E.ZPLUS
ASSIGN/E.BADMEAS=0
MODE/MANUAL
```



Every time the SELPL command is translated the value is store in the E.SELPL variable.

To simulate Tutor THEO functionality the TH table is declared and initialized:

```
ASSIGN/E.TH.X.N = 0
=CALLSUB/THEO_INIT,TutorSub.prg:E.TH,,,
```

To simulate Tutor Output Format table functionality the E.OFMT table is declared and initialized:

```
ASSIGN/E.OFMT.ON = 1
=CALLSUB/INIT_FORMAT_TBL,:E.OFMT,,
```

The command END_PROGRAM is translated with the declaration of the label, the command to print report if PRN command as been translated at least one time and the command to close already open file (if any):

```
END_PROGRAM= /LABEL
PRINT/REPORT,TO_FILE=OFF,AUTO=0,$
TO_PRINTER=ON,DRAFTMODE=OFF,$
PREVIOUS_RUNS=KEEP_INSTANCES
COMMENT/DOC,NO,===== Close All files=====
=CALLSUB/TUTOR_CLOSE,TutorSub.PRG: E.OFMT,0,,,
```

ENDSTAT translates to →

```
GOTO/END_PROGRAM
```

Variable Declaration

The declaration of variables does not exist in PC-DMIS. For improved readability, the declared variables are translated as an assigned value. Where the Tutor value in not suitable, the assigned value is "0" or {0,0,0}.

String variables are translated as an assigned value where the string contains only "blank" characters. The number of "blank" is the dimension of the declared variable. See the chart below.

Array variables are translated as an assigned value with the assignment of a default value to the first one and the last element of the array.

Element variables are translated as "TutorElement"

The following chart shows the Tutor command with the equivalent PC-DMIS translations:

INTEGER IntVar	→ ASSIGN/ E.IntVar = 0
INTEGER IntVar = 1	→ ASSIGN/ E.IntVar = 1
REAL RealVar	→ ASSIGN/ E.RealVar = 0
REAL RealVar =1.234567890	→ ASSIGN/ E.RealVar = 1.234567890
BOOLEAN BoolVar	→ ASSIGN/ E.BoolVar = 0
BOOLEAN BoolVar = False	→ ASSIGN/ E.BoolVar = 0
BOOLEAN BoolVar = True	→ ASSIGN/ E.BoolVar = 1
COORD CoordVar	→ ASSIGN/ E.CoordVar = MPOINT(0,0,0)
COORD CoordVar = {1.1, 2.2, 3.3}	→ ASSIGN/ E.CoordVar = MPOINT(1.1, 2.2, 3.3)
VECTOR VectVar	→ ASSIGN/ E.VectVar = MPOINT(0,0,0)
VECTOR VectVar = {4.4, 5.5, 6.6}	→ ASSIGN/ E.VectVar = MPOINT(4.4, 5.5, 6.6)
ELEMENT ElVar	→ ASSIGN/ E.ElVar = TutorElement(1)
ELEMENT ElVar = {10.1, 2.23, 34.8}	→ ASSIGN/ E.ElVar = TutorElement(1)
	→ ASSIGN/ E.ElVar.X = 10.1
	→ ASSIGN/ E.ElVar.Y = 2.23
	→ ASSIGN/ E.ElVar.Z = 34.8
STRING StringVar[20]	→ ASSIGN/ E.StringVar = "
INTEGER_ARRAY IntArray[50,20]	" → ASSIGN/ EIntArray[1,1] = 0 → ASSIGN/ EIntArray[50,20] = 0
REAL_ARRAY	→ ASSIGN/ E.RealArray[1,1,1] = 0
RealArray[10,20,30]	→ ASSIGN/ E.RealArray[10,20,30] = 0
BOOLEAN_ARRAY BoolArray[10]	→ ASSIGN/ E.BoolArray[1] = 0 → ASSIGN/ E.BoolArray[10] = 0
COORD_ARRAY	→ ASSIGN/ E.CoordArray[1,1] = MPOINT(0, 0, 0)
CoordArray[10,10]	→ ASSIGN/ E.CoordArray[10,10] = MPOINT(0, 0, 0)
VECTOR_ARRAY VectArray[3,3]	→ ASSIGN/ E.VectArray[1,1] = MPOINT(0, 0, 0) → ASSIGN/ E.VectArray[3,3] = MPOINT(0, 0, 0)

ELEMENT_ARRAY ElVar[300]	→ ASSIGN/ E.ElVar = TutorElement(300)
---------------------------	---------------------------------------



[Only the FIRST element array command is translated as follows:](#)

ELEMENT_ARRAY MEMORY[300] translates to →

```
ASSIGN/ E.MEMORY = TUTORELEMENT (300)
ASSIGN/ E.MEMSIZE = 300
```



Tutor COORD variable also stores the reference system active at the moment values are assigned to it. This functionality is not available in PC-DMIS.

Functions

The following chart shows Tutor Functions with the associated PC-DMIS Functions:

ABS(value)	→	ABS(value)
TRUNC(value)	→	INT(value)
ROUND(value)	→	ROUND(value)
SQR(value)	→	(value)^2
SQRT(value)	→	SQRT(value)
EXP(value)	→	EXP(value)
LN(value)	→	LN(value)
SIN(value)	→	SIN(value)
ARSIN(value)	→	ASIN(value)
COS(value)	→	COS(value)
ARCOS(value)	→	ACOS(value)
TAN(value)	→	TAN(value)
ARTAN(value)	→	ATAN(value)
MODULE(vector)	→	DOUBLE(vector)
NORM(vector)	→	UNIT(vector)

Values may be an expression.

Arithmetical Operators

The following chart shows Tutor Arithmetical Operators with the corresponding PC-DMIS Arithmetical Operators:

-value (minus sign)	→	-value
+ (sum)	→	+
- (subtraction)	→	-
* (multiplication)	→	*
/ (division)	→	/
DIV (value)	→	INT(dividend / divisor)
MOD(value)	→	%

Tutor Translator

Vector Operators

The following chart shows Tutor Vector Operators with the corresponding PC-DMIS Vector Operators:

-vector (minus sign)	→	- vector
+ (sum)	→	+
- (subtraction)	→	-
= {x,y,z}	→	= MPOINT(x,y,z)
SCAL	→	DOT
EXT	→	CROSS

Relational and Logical Operators

The following chart shows Tutor Relational and Logical Operators with the corresponding PC-DMIS Relational and Logical Operators:

Not	→	!
Eq	→	==
NE	→	<>
GT	→	>
LT	→	<
GE	→	>=
LE	→	<=
AND	→	AND
OR	→	OR

Probe Management Commands

Load Qualification File

The **Load Qualification File** command is translated to associate the Tutor Head-Tip with the PC-DMIS qualified tip. This association is made through Option Dialog Windows.

The PUSH_PROBE and POP_PROBE commands use the E.PROBE_SP variable for the stack index, while the E.PROBE_STACK defines the stack.

These variables are declared at the beginning of the program:

```
ASSIGN/ E.PROBE_SP=0
ASSIGN/ E.PROBE_STACK=0
LOADPROBE/ PROBENAME
=CALLSUB/ ERASE_PROBE_SUB,TutorSub.PRG:0,0, E.TIPARRAY, (This subroutine will erase the
TipArray )
ASSIGN/ E.TipArray[1,1] = "T1A0B0"
ASSIGN/ E.TipArray[2,1] = "T1A45B0"
```

```
ASSIGN/ E.TipArray[3,1] = "T1A90B0"
ASSIGN/ E.TipArray[4,1] = "T1A90B45"
ASSIGN/ E.TipArray[5,1] = "T1A90B90"
ASSIGN/ E.TipArray[6,1] = "T1A45B45"
ASSIGN/ E.TipArray[7,1] = "T1A-90B45"
```



If the Tip name is not known during translation, then the TipArray command is translated as
TipArray[1,1] = ""

Tutor loads a qualification file for the current Tip based on the last qualified Tip. Therefore, it is required to insert a selection command for the head and tip based on the qualification file and operator demand. Otherwise, values of 0,0 will be added to be modified by the operator upon execution error.

TIP/TipArray[Head,Tip] translates to → TIP/ E.TIPARRAY[2,1], . . .

Save Probe File

The **Save Probe File** command does not translate. The operator must create all necessary tip qualification files using PC-DMIS.

PROBE(Head,Tip) translates to →

```
ASSIGN/ E.CURTIP = E.TipArray[Head,Tip]
TIP/ E.CURTIP, SHANKIJK=..., ..., ..., ANGLE=0
```

PUSH_PROBE translates to →

```
ASSIGN/ E.PROBE_SP= E.PROBE_SP+1
ASSIGN/ E.PROBE_STACK[E.PROBE_SP]= PROBEDATA()
```

POP_PROBE translates to →

```
TIP/ E.PROBE_STACK[E.PROBE_SP]
ASSIGN/ E.PROBE_SP= E.PROBE_SP-1
```

PROBE0 translates to →

```
=CALLSUB/ ERASE_PROBE_SUB,TutorSub.PRG:Head,Tip, E.TIPARRAY
```

If Tip=0 all 5 Tips related to the Head are cancelled.

If Head=0 all the Tips of all the Heads are cancelled.

The following Tutor commands related to probe definitions are not translated to PC-DMIS:

- GAUGE
- CALIB
- QUALI
- PH9
- TOOL_OFFSET

Tutor Translator

- CALIB_DIAM
- TDIAM
- TIP_COMPENS

Alignment Commands

PC-DMIS can have a large number of alignments, while TUTOR can have at most 11. The 11-element array called E.ALIGNMENTS will correspond to the PC-DMIS alignments. All alignments are initialized to the STARTUP ID that is language dependent. In this document helpfile we refer to startup alignment using "STARTUP" keyword.

The E.CUR_ALIGN variable contains the **index** of the current reference system.

The PUSH_REF SYS and POP_REF SYS commands use the REF_SP variable for the stack index, while the E.REFSYS_STACK defines the **stack**.

These variables are declared at the beginning of the program:

```
ASSIGN/E.REF_SP=0
ASSIGN/E.REFSYS_STACK=0
ASSIGN/E.CUR_ALIGN=0
ASSIGN/E.REF_SP=0
ASSIGN/E.ALIGNMENTS[0] = "STARTUP"
ASSIGN/E.ALIGNMENTS[1] = " STARTUP "
ASSIGN/E.ALIGNMENTS[2] = " STARTUP "
ASSIGN/E.ALIGNMENTS[3] = " STARTUP "
ASSIGN/E.ALIGNMENTS[4] = " STARTUP "
ASSIGN/E.ALIGNMENTS[5] = " STARTUP "
ASSIGN/E.ALIGNMENTS[6] = " STARTUP "
ASSIGN/E.ALIGNMENTS[7] = " STARTUP "
ASSIGN/E.ALIGNMENTS[9] = " STARTUP "
ASSIGN/E.ALIGNMENTS[10] = " STARTUP "
```

Tutor allows the axis name on which to work to be omitted while PC-DMIS requires it. The following function is required to analyze the cosines of the element and return the appropriate keyword.

```
ASSIGN/E.BIGGEST =
FUNCTION((F),IF(ABS(F.I)>ABS(F.J),IF(ABS(F.K)>ABS(F.I),IF(F.K>0,E.ZPLUS,E.ZMINUS),
IF(F.I>0,E.XPLUS,E.XMINUS)),IF(ABS(F.K)>ABS(F.J),IF(F.K>0,E.ZPLUS,E.ZMINUS),
IF(F.J>0,E.YPLUS,E.YMINUS)))) Skew1 - Alignment Primary Axis
```

Skew1 – Alignment Secondary Axis

SKEW1 (MEMORY[1],1,0,XDIR) translates to →

```
A1 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[0], LIST= YES
ALIGNMENT/LEVEL,XPLUS, E.MEMORY[1].ID
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =1
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN]= "A1"
```

SKEW1 (1,0,XDIR) translates to →

```
A2 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[0], LIST= YES
ALIGNMENT/LEVEL,XPLUS, E.WM1.ID
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =1
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN]= "A2"
```

SKEW1 (3,2) translates to →

```
A3 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[2], LIST= YES
ALIGNMENT/LEVEL,BIGGEST(E.WM1.ID), E.WM1.ID
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =3
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN]= "A3"
```

SKEW1 (2) translates to →

```
A4 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[E.CUR_ALIGN], LIST= YES
ALIGNMENT/LEVEL,BIGGEST(E.WM1.ID), E.WM1.ID
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =2
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A4"
```



The Tutorelement array E.MEMORY[1] or E.WM1 must have an ID field created. This is done when the feature associated with E.MEMORY[1] is created.

XPLUS alternative keywords are XMINUS, YPLUS, YMINUS, ZPLUS, and ZMINUS.

The numbers of the source reference system, the reference destination system and the index of the element in the array memory can be a variables or expressions.

Skew2 – Alignment Secondary Axis

SKEW2 (E.MEMORY[2], 2, 1, Z, XDIR) translates to →

```
A5 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[1], LIST= YES
ALIGNMENT/ROTATE,XPLUS,TO,E.MEMORY[2].ID,ABOUT,ZPLUS
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =2
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A5"
```

SKEW2 (2, 1, Z, XDIR) translates to →

```
A6 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[1], LIST= YES
ALIGNMENT/ROTATE,XPLUS,TO,E.WM1.ID,ABOUT,ZPLUS
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =2
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A6"
```

SKEW2 (2, Z, XDIR) translates to →

```
A7 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[E.CUR_ALIGN], LIST= YES
ALIGNMENT/ROTATE,XPLUS,TO,E.WM1.ID,ABOUT,ZPLUS
ALIGNMENT/END
```

Tutor Translator

```
ASSIGN/E.CUR_ALIGN =2  
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A7"
```

SKEW2 (2, Z) translates to →

```
A8 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[E.CUR_ALIGN], LIST= YES  
ALIGNMENT/ROTATE,BIGGEST(E.WM1.ID),TO,E.WM1.ID,ABOUT,ZPLUS  
ALIGNMENT/END  
ASSIGN/E.CUR_ALIGN =2  
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A8"
```

Iskew - Theoretical Rotation

ISKEW (ROT, 30, Z, 3, 2) translates to →

```
A9 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[2], LIST= YES  
ALIGNMENT/ROTATE_OFFSET,30,ABOUT,ZPLUS  
ALIGNMENT/END  
ASSIGN/E.CUR_ALIGN =3  
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A9"
```

ISKEW (ROT, 30, Z, 4) translates to →

```
A10 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[E.CUR_ALIGN], LIST= YES  
ALIGNMENT/ROTATE_OFFSET,30,ABOUT,ZPLUS  
ASSIGN/E.CUR_ALIGN =4  
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A10"
```

 In PC-DMIS the theoretical rotation by offset does not exist so it is necessary to calculate the rotation angle. For this reason the function E.OFF3ROT is defined:

```
ASSIGN/E.OFF2ROT = FUNCTION((X,Y),IF(X==0,IF(Y>0,90,-  
90),E.RAD2DEG(ATAN(X/Y))))
```

ISKEW (OFS, OFF1,OFF2, Z, 5, 4) translates to →

```
A11 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[4], LIST= YES  
ALIGNMENT/ROTATE_OFFSET,-E.OFF2ROT(OFF1,OFF2),ABOUT,ZPLUS  
ALIGNMENT/END  
ASSIGN/E.CUR_ALIGN =5  
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A11"
```

ISKEW (OFS, OFF1,OFF2, Z, 6)

ISKEW (OFF1,OFF2, Z, 6) translates to →

```
A12 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[E.CUR_ALIGN], LIST= YES  
ALIGNMENT/ROTATE_OFFSET,-E.OFF2ROT(OFF1,OFF2),ABOUT,ZPLUS&  
ALIGNMENT/END  
ASSIGN/E.CUR_ALIGN =6  
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A12"
```

Preset – Origin Presetting

PRESET (E.MEMORY[3],X=10,Y=20,Z=30,1,1) translates to →

```
A13 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[1], LIST= YES
ALIGNMENT/TRANS,XAXIS,E.MEMORY[3].ID
ALIGNMENT/TRANS,YAXIS,E.MEMORY[3].ID
ALIGNMENT/TRANS,ZAXIS,E.MEMORY[3].ID
ALIGNMENT/TRANS_OFFSET,XAXIS,10
ALIGNMENT/TRANS_OFFSET,YAXIS,20
ALIGNMENT/TRANS_OFFSET,ZAXIS,30
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =1&
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A13"
```

PRESET (MEMORY[3],X=10,1,1) translates to →

```
A13 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[1], LIST= YES
ALIGNMENT/TRANS,XAXIS,E.MEMORY[3].ID
ALIGNMENT/TRANS_OFFSET,XAXIS,10
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =1
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A13"
```

PRESET (E.WM1,1) translates to →

```
A13 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[CURALIGN], LIST= YES
ALIGNMENT/TRANS,XAXIS,E.WM1.ID
ALIGNMENT/TRANS,YAXIS,E.WM1.ID
ALIGNMENT/TRANS,ZAXIS,E.WM1.ID
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =1
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A13"
```

PRESET (X=10,Y=20,Z=30,1,1) translates to →

```
A14 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[1], LIST= YES
ALIGNMENT/TRANS,XAXIS,E.WM1.ID
ALIGNMENT/TRANS,YAXIS,E.WM1.ID
ALIGNMENT/TRANS,ZAXIS,E.WM1.ID
ALIGNMENT/TRANS_OFFSET,XAXIS,10
ALIGNMENT/TRANS_OFFSET,YAXIS,20
ALIGNMENT/TRANS_OFFSET,ZAXIS,30
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =1
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A14"
```

PRESET (X=10,1,1) translates to →

```
A15 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[1], LIST= YES
ALIGNMENT/TRANS,XAXIS,E.WM1.ID
ALIGNMENT/TRANS_OFFSET,XAXIS,10
ALIGNMENT/END
ASSIGN/E.CUR_ALIGN =1
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A15"
```

Tutor Translator

PRESET (X=10,7) translates to →

```
A16 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[E.CUR_ALIGN], LIST= YES  
ALIGNMENT/TRANS,XAXIS,E.WM1.ID  
ALIGNMENT/TRANS,YAXIS,E.WM1.ID  
ALIGNMENT/TRANS,ZAXIS,E.WM1.ID  
ALIGNMENT/TRANS_OFFSET,XAXIS,10  
ALIGNMENT/END  
ASSIGN/E.CUR_ALIGN =7  
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A16"
```

PRESET (-X=10,Y=20,Z=30,1,1) translates to →

```
A17 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[1], LIST= YES  
ALIGNMENT/TRANS_OFFSET,XAXIS,10  
ALIGNMENT/TRANS_OFFSET,YAXIS,20  
ALIGNMENT/TRANS_OFFSET,ZAXIS,30  
ALIGNMENT/END  
ASSIGN/E.CUR_ALIGN =1  
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A17"
```

PRESET (-X=10,1,1) translates to →

```
A18 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[1], LIST= YES  
ALIGNMENT/TRANS_OFFSET,XAXIS,10  
ALIGNMENT/END  
ASSIGN/E.CUR_ALIGN =1  
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A18"
```

PRESET (-X=10,7) translates to →

```
A19 =ALIGNMENT/START,RECALL:E.ALIGNMENTS[E.CUR_ALIGN], LIST= YES  
ALIGNMENT/TRANS_OFFSET,XAXIS,10  
ALIGNMENT/END  
ASSIGN/E.CUR_ALIGN =7  
ASSIGN/E.ALIGNMENTS[E.CUR_ALIGN] = "A19"
```

Use of Reference Systems

PUSH_REF SYS translates to →

```
ASSIGN/E.REF_SP = E.REF_SP+1  
ASSIGN/E.REFSYS_STACK[E.REF_SP] = E.CUR_ALIGN
```

POP_REF SYS translates to →

```
ASSIGN/E.CUR_ALIGN = E.REFSYS_STACK[E.REF_SP]  
ASSIGN/E.REF_SP = E.REF_SP-1  
RECALL/ALIGNMENT,INTERNAL,E.ALIGNMENTS[E.CUR_ALIGN]
```

REFSYS 1 translates to →

```
RECALL/ALIGNMENT,INTERNAL, E.ALIGNMENTS[1]  
ASSIGN/E.CUR_ALIGN =1
```

REF0 4**REF0 4 DEL** translates to →

```
IF/E.CUR_ALIGN <> 4
ASSIGN/E.ALIGNMENTS[4] = "STARTUP"
END_IF/
```

REF0**REF0 DEL** translates to →

```
ASSIGN/E.CUR_ALIGN = 0
ASSIGN/E.ALIGNMENTS[0] = "STARTUP"
ASSIGN/E.ALIGNMENTS[1] = "STARTUP"
ASSIGN/E.ALIGNMENTS[2] = "STARTUP"
ASSIGN/E.ALIGNMENTS[3] = "STARTUP"
ASSIGN/E.ALIGNMENTS[4] = "STARTUP"
ASSIGN/E.ALIGNMENTS[5] = "STARTUP"
ASSIGN/E.ALIGNMENTS[6] = "STARTUP"
ASSIGN/E.ALIGNMENTS[7] = "STARTUP"
ASSIGN/E.ALIGNMENTS[8] = "STARTUP"
ASSIGN/E.ALIGNMENTS[9] = "STARTUP"
ASSIGN/E.ALIGNMENTS[10] = "STARTUP"
```

SAVE(REFSYS,"filename")**SAVE_REFSYS "filename"** translates to →

```
COMMENT/DOC,SAVE(REFSYS,"Filename")
=CALLSUB/SAVE_REFSYS_SUB, TutorSub.PRG:"Filename",E.ALIGNMENTS,E.CUR_ALIGN,
```

LOAD(REFSYS,"filename")**LOAD_REFSYS "filename"** translates to →

```
COMMENT/DOC,LOAD(REFSYS,"Filename")
=CALLSUB/LOAD_REFSYS_SUB, TutorSub.PRG:"Filename",E.ALIGNMENTS,E.CUR_ALIGN,
```

Use of Reference Planes**SELPL X** translates to →

```
WORKPLANE/ZPLUS
ASSIGN/E.SELPL = E.ZPLUS
```

SELPL Y translates to →

```
WORKPLANE/XPLUS
ASSIGN/E.SELPL = E.XPLUS
```

SELPL Z translates to →

```
WORKPLANE/YPLUS
ASSIGN/E.SELPL = E.YPLUS
```

Measurement Commands

The Tutor measurement command has the following format:

Melem(ResMem,HitsNum,RefSysNum[,scpec]) [SAVEPTS]

Listed below are the parameters for the above command:

- [] means optional field or element specific keyword.
- **Melem:** The element keyword MCIRCLE,MCONE
- **ResMem:** The destination memory. If omitted means E.WM1.
- **HitsNum:** The number of hits to be measured. Tutor allows variables but only commands with a known hit number are translated.
- **RefSysNum:** The reference system in which the element has to be measured. If omitted the current reference system is used. This may be a variable.
- **SAVEPTS:** Measured points are not discarded.

Measurement Commands that **don't** have equivalent commands in PC-DMIS but do have construction commands are translated:

- For each **Measured Point** in the path (see the "[Look Ahead Procedure](#)" chapter) a measure point command is added:

```
PNT1      =FEAT/POINT,RECT
THEO/152.003,236.002,140.001,0,0,-1
ACTL/152.003,236.002,140.003,0,0,-1
MEAS/POINT,1
MOVE/POINT,152.003,236.002,130.003
HIT/ BASIC,152.003,236.002,140.003,USE THEO = NO
ENDMEAS
```

- **Construction Feature Command** - This uses the previously measured points. This command requires the definition of the feature theoretical values. For more Information, see "[THEO Command](#)".

The following PC-DMIS commands are admitted inside the measurement path definition:

- TIP/T1A0B0 = Probe selection (Probe(head, tip))
- PREHIT / appr = Approach definition (DIST_APPROACH appr,)
- RETRACT / king = Retract definition. In Tutor this is referred to as the approach.
- CHECK/ depth = Depth definition (DIST_APPROACH, depth)

For the elements circle, cone, cylinder and sphere PC-DMIS requires the information if it is an external or internal element. Since this information is not available in Tutor, it is deduced, where possible, from the positioning and measured points.

Using the automation method, RecalculateNominals theoretical values of the feature are extracted from the Measured Path. The SAVEPTS keyword is not translated because PC-DMIS always stores measured hits.

Measurement Commands with no PC-DMIS Equivalent

Some Tutor commands have no equivalent PC-DMIS commands and are approximately translated. For example, fixed axis cones are translated as generic cones and A Paraboloid or a Torus is translated as a measured Points Set. A Comment/Oper command has been added in order to notify the operator about missing compatibility with the original Part Program. A Comment/Oper command has been added for ISO commands as well.

ISO Commands

Measurement commands that use the ISO algorithm don't have an equivalent PC-DMIS command. The commands are approximately translated similar to a command that uses a Least Square algorithm.



The Measurement Command examples given below do not represent all of the possible measurement commands, but do provide a representative list of examples. All Tutor commands are listed, but not all their possible parameter combinations are given.

Measurement Commands that may have a PC-DMIS Equivalent

Measurement Commands that **may** or **may not** have equivalent commands in PC-DMIS but do have construction commands are translated:

- If **RefSysNum** is defined the associated reference system is recalled:
RECALL/ALIGNMENT,INTERNAL, E.ALIGNMENTS[RefSysNum]
- If **RefSysNum** is defined the current reference system is recalled:
RECALL/ALIGNMENT,INTERNAL, E.ALIGNMENTS[E.CUR_ALIGN]
- **Copy Structure** - Copy the structure of E.WM1 from E.WM2.
ASSIGN/E.WM2 = E.WM1
- **Data Memorization** in the structure E.WM1.
- If the memory of destination is not E.WM1, data is stored also in the destination memory structure.
- **Dimension Command** and **True Position** computing if necessary
- **Profile Command** - Computing if necessary
- **Evaluation output**. - For more information, see [Element Evaluation](#).

Measurement Command Conditions

Measurement commands for PC-DMIS must contain the positioning points and measured points. Tutor programs should meet the followings conditions:

1. The Measurement Command has an associated PATH. The most frequent case is that the translator reads the Path and turns its sequence into a MOVE/POINT command for the positioning and HIT/BASIC command for the Measurements. HIT.BASIC example:

```
HIT/BASIC,9.6603,20.1643,-2.0198,0,-0.9659258,0.258819,9.6603,20.1643,-2.0198,USE THEO = NO.
```

1. The Measurement Command does not have an associated PATH. In this case the translator starts a Look Ahead procedure to seek the commands for positioning (move)

Tutor Translator

and of measurement (movetf). See the “[Look Ahead Procedure](#)” chapter. At the end of the search the followings conditions can happen:

- All positioning and measured points have been found. In this case the command is translated as suitable in the point 1.
- Not all of the positioning and measured points have been found. In this case the Measurement command is translated using the relearn hit command and the operator is notified:
HIT/BASIC, RELEARN

The HIT/BASIC, RELEARN command must be inserted as many times as the number of hits specified in the measurement command.

Example Circle Command

```
CIR1 = FEAT/CIRCLE,RECT,OUT,LEAST_SQR  
THEO/-22.9857,-46.012,-4,0,0,1,22.955  
ACTL/0,0,0,0,0,1,1  
MEAS/CIRCLE,4,WORKPLANE
```

<i>Move/Hit/Tip</i> <i>Prehit/Retract and Check</i> <i>Commands</i>	MOVE/POINT,-23,-46,12 MOVE/POINT,-15.5,-46,-4 TIP/TIPARRAY[2,1], SHANKIJK=0, 0.7071, 0.7071, ANGLE=0 HIT/BASIC,-11.5,-46,-4,-11.5,-46,-4,USE THEO = NO MOVE/POINT,-23,-37,-4 TIP/TIPARRAY[3,1], SHANKIJK=0, 1, 0, ANGLE=180 HIT/BASIC,-23,-34.5,-4,-23,-34.5,-4,USE THEO = NO MOVE/POINT,-31,-46,-4 TIP/TIPARRAY[4,1], SHANKIJK=-0.7071, 0.7071, 0, ANGLE=90 HIT/BASIC,-34.5,-46,-4,-34.5,-46,-4,USE THEO = NO MOVE/POINT,-23,-54,-4 PREHIT/ 0.032 RETRACT/ 0.032 HECK/ 0.1027,1 CLEARP/ZPLUS,12,XPLUS,12 MOVE/CLEARPLANE MOVE/CIRCULAR MOVE/INCREMENT,12,12,2 MOVE/ALL,12,23,32,TIP=TIPARRAY[2,1], SHANKIJK=0, 0.7071, 0.7071, ANGLE=0 MOVESPEED/ 20 TOUCHSPEED/ 2 SCANSPEED/ 2 WORKPLANE/ZPLUS PROBECOM/ON PROBECOM/OFF COMMENT/DOC,"This is a comment" TIP/TIPARRAY[1,1], SHANKIJK=0, 0, 1, ANGLE=0 HIT/BASIC,-23,-57.5,-4,-23,-57.5,-4,USE THEO = NO MOVE/POINT,-23,-46,12 ENDMEAS
---	---

Circle Measurement Commands

MCIR(MEMORY[10],4,1) translates to →

```

ASSIGN/E.BADMEAS=0
RECALL/ALIGNMENT, INTERNAL, E.ALIGNMENTS [1]
    CIR2 =FEAT/CIRCLE,RECT,OUT,LEAST_SQR
    THEO/0,0,0,0,0,1,1ACTL/0,0,0,0,0,1,1
    MEAS/CIRCLE, 4, WORKPLANE
...
...      move	hit/tip commands
...
ENDMEAS/

RECALL/ALIGNMENT, INTERNAL, E.ALIGNMENTS [E.CUR_ALIGN]
ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT("CIR2")
ASSIGN/E.MEMORY[10]=E.WM1
...
...      Evaluation commands.
...

```

MCIR(4,C3D) translates to →

```

ASSIGN/E.BADMEAS=0
    CIR3 =FEAT/CIRCLE,RECT,IN,LEAST_SQR
    THEO/0,0,0,0,0,1,1
    ACTL/0,0,0,0,0,1,1
    MEAS/CIRCLE, 4, 3D
...
...      move	hit/tip commands
...
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT("CIR3")
...
...      Evaluation commands.
...
```

MCIR(ELEVVAR,4) CMIN translates to →

```

ASSIGN/E.BADMEAS=0
    CIR4 =FEAT/CIRCLE,RECT,IN,MIN_CIRCSC
    THEO/-0.098,0.0006,-9.9995,0.0099995,0,0.99995,10.31
    ACTL/-0.098,0.0006,-9.9995,0.0099995,0,0.99995,10.31
    MEAS/CIRCLE,3,WORKPLANE
...
...      move	hit/tip commands
...
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("CIR4")
ASSIGN/ELEVVAR = E.WM1

```

Tutor Translator

```
...
...      Evaluation commands.
...
```

MCIR(4) CMAX translates to →

```
ASSIGN/E.BADMEAS=0
CIR4    =FEAT/CIRCLE,RECT,IN, MAX_INSC
THEO/-0.098,0.0006,-9.9995,0.0099995,0,0.99995,10.31
ACTL/-0.098,0.0006,-9.9995,0.0099995,0,0.99995,10.31
MEAS/CIRCLE,3,WORKPLANE
...
...      move	hit/tip commands
...
ENDMEAS/
ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR4" )
...
...      Evaluation commands.
...
```

MCIR (MEMORY [10], 3) DM 12.93 translates to →

```
ASSIGN/E.BADMEAS=0
CIR5 = FEAT/CIRCLE, RECT, IN, FIXED_RAD
THEO/0.002,0.0006,10,0.0099995,0,0.99995,12.93
ACTL/0.002,0.0006,10,0.0099995,0,0.99995,12.93
MEAS/CIRCLE, 3, WORKPLANE
...
...      move	hit/tip commands
...
ENDMEAS/
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ( "CIR5" )
ASSIGN/E.MEMORY [10] = E.WM1
...
...      Evaluation commands.
...
```

Not Aligned Circle

PC-DMIS does not have a corresponding measured feature for a not aligned circle. This command is translated as two different PC-DMIS commands.

MCIR_NA(MEMORY[2],7) translates to →

```
ASSIGN/E.BADMEAS=0
PLN2    =FEAT/PLANE,RECT
THEO/1.8618,-1.1968,1.8671,-0.7975331,-0.6032664,-0.0032448
ACTL/1.8618,-1.1968,1.8671,-0.7975331,-0.6032664,-0.0032448
MEAS/PLANE,3
...
...      move	hit/tip commands. The number of hits is always 3.
...
```

```

ENDMEAS/

CIR7 =FEAT/CIRCLE,RECT,OUT,LEAST_SQR
THEO/2.0037,3.0008,0.12,-0.7975331,-0.6032664,-0.0032448,12.663
ACTL/2.0037,3.0008,0.12,-0.7975331,-0.6032664,-0.0032448,12.663
MEAS/CIRCLE,4,FEATURE=PLN2
...
... move/hit/tip commands. The number of hits is the number defined in the
MCIR_NA command minus the 3 hits of the plane.
...
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR7" )
ASSIGN/E.MEMORY[2]= E.WM1
...
... Evaluation commands.
...

```

Cone Measurement Commands

Tutor and PC-DMIS have different meanings for XYZ fields. Tutor views the XYZ fields as the vertex of the cone while PC-DMIS views the XYZ fields as the coordinates of the centroid of the first section circle, displayed in the THEO command. For Cone Measurements the WIDE keyword is ignored.

PC-DMIS THEO fields for cones are:

THEO/ XCoord, YCoord, ZCoord, I, J, K, Angle, Diameter1, Diameter2.

XCoord, YCoord and ZCoord are the coordinates of the centroid of the circle (section circle). Diameters1 and Diameter2 are not filled by the translator. Tutor Translator does not compute THEO values, but does insert HITS and calls the PC-DMIS automation command to compute the CONE values.

MCON(MEMORY [10], 7)
MCON(MEMORY [10], 7, FORM LSM)
MCON(MEMORY [10], 7, FORM LSM, WIDE)
MCON(MEMORY [10], 7, WIDE) translates to →

```

ASSIGN/E.BADMEAS=0
CON1 =FEAT/CONE,RECT,OUT,ANG
THEO/-22.9857,-46.012,-38.0277,0,0,1,20,0,12
ACTL/-22.9857,-46.012,-38.0277,0,0,1,20,12,0
MEAS/CONE,8
...
... move/hit/tip commands.
...
ENDMEAS/

```

ASSIGN/E.WM2=E.WM1

Tutor Translator

```
ASSIGN/E.WM1 = TUTORELEMENT( "CON1" )
ASSIGN/E.MEMORY[10]= E.WM1
...
...           Evaluation commands .
...
```

The Following Cone Measurement commands don't have equivalent commands in PC-DMIS.
See the chapter on "[Measurement Commands with no PC-DMIS Equivalent](#)" for more information.

MCONE(MEMORY[10],7,dirX, dirY,dirZ)
MCONE(MEMORY[10],7,dirX, dirY,dirZ,FORM LSM)
MCONE(MEMORY[10],7,dirX, dirY,dirZ,FORM LSM,WIDE)
MCONE(MEMORY[10],7,dirX, dirY,dirZ,WIDE) translates to →

```
ASSIGN/E.BADMEAS=0
  CON2 = FEAT/CONE,RECT,OUT,ANG
THEO/-22.9857,-46.012,-38.0277,E.DIRX,E.DIRY,E.DIRZ,20,0,12
ACTL/-22.9857,-46.012,-37.0277,0,0,1,0,0,0
MEAS/CONE,8
...
...           move/hit/tip commands .
...
ENDMEAS/
```

```
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CON2" )
ASSIGN/E.MEMORY[10]= E.WM1
...
...           Evaluation commands .
...
```

The following commands that use ISO algorithm don't have an equivalent PC-DMIS command.
See the chapter on "[Measurement Commands with no PC-DMIS Equivalent](#)" for more information.

MCONE(MEMORY [10] ,7, FORM ISO)
MCONE(MEMORY [10], 7, dirX, dirY, dirZ, FORM ISO)
MCONE(MEMORY [10], 7, dirX, dirY, dirZ, FORM ISO, WIDE)

Cylinder Measurement Command

PC-DMIS THEO fields for Cylinder are:

THEO/ XCoord, YCoord, ZCoord, I, J, K, Diameter, Length.
Length is not filled by the translator but computed using the CalculateNominals automation method.

MCYL (MEMORY [10], 7, SDM)

MCYL (MEMORY [10], 7, SDM, FORM LSM) translates to →

```

ASSIGN/E.BADMEAS=0
CYL1 =FEAT/CYLINDER,RECT,OUT,LEAST_SQR
THEO/0.4796,0.0006,-9.1832,0.5775835,0,0.8163316,23.3,0
ACTL/0.4796,0.0006,-9.1832,0.5775835,0,0.8163316,23.3,0
MEAS/CYLINDER,7
...
...      move	hit/tip commands
...
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CYL1" )
ASSIGN/E.MEMORY[10]= E.WM1
...
...      Evaluation commands .
...

```

The Following commands with fixed axes don't have equivalent commands in PC-DMIS. See the chapter on "[Measurement Commands with no PC-DMIS Equivalent](#)" for more information.

MCYL (MEMORY [10], 7, SDM, dirX, dirY, dirZ)

MCYL (MEMORY [10], 7, SDM, dirX, dirY, dirZ, FORM LSM) translates to →

```

ASSIGN/E.BADMEAS=0
CYL2 =FEAT/CYLINDER, RECT, OUT, LEAST_SQR
THEO/0.4796,0.0006,-9.1832,E.DIR           X,E.DIRY,E.DIRZ,0,0,0
ACTL/0.4796,0.0006,-9.1832,0.5775835,0,0.8163316,0,0,0
MEAS/CYLINDER,7
...
...      move	hit/tip commands
...
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CYL2" )
ASSIGN/E.MEMORY[10]= E.WM2
...
...      Evaluation commands .
...
```

The following commands that use ISO algorithm don't have an equivalent PC-DMIS command. See the chapter on "[Measurement Commands with no PC-DMIS Equivalent](#)" for more information.

MCYL (MEMORY [10],7, SDM, FORM ISO)

MCYL (MEMORY [10],7, SDM, dirX, dirY,dirZ, FORM ISO)

Tutor Translator

The command to measure a cylinder with 2 diameters (without SDM keyword) does not have equivalent commands on PC-DMIS and are not translated. A Comment/Oper is added in order to notify the operator about missing compatibility with the original Part Program.

Ellipse Measurement Commands

PC-DMIS does not have an equivalent measurement commands. Ellipses can only be constructed using measured points.

The command is translated as a measured SET command and a construction command.

MELLIPSE (MEMORY [3], 6) path PT1 where the path is:

```
P 1 1    152.003  236.002  130.003
M 1 1    152.003  236.002  140.003
P 1 1    52.001   118.002  130.007
M 1 1    152.001  118.002  130.007
P 1 1    152.004  100.030  120.003
M 1 1    152.004  96.001   120.001
P 1 1    152.002  116.003  103.004
M 1 1    152.002  116.003  96.003
P 1 1    152.003  236.004  104.000
M 1 1    152.003  236.004  96.002
P 1 1    152.003  248.001  118.003
M 1 1    152.003  258.001  118.003
```

translates to →

```
WORKPLANE/XPLUS
ASSIGN/E.BADMEAS=0
SCN1 = FEAT/SET,RECT
THEO/152.0023,176.6688,118.5883,0,0,1
ACTL/152.0023,176.6688,118.5883,0,0,1
MEAS/SET,6
MOVE/POINT,152.003,236.002,130.003
HIT/BASIC,152.003,236.002,151.514,152.003,236.002,151.514,USE THEO = NO
MOVE/POINT,152.001,118.002,130.007
HIT/BASIC,152.001,118.002,130.007,152.001,118.002,130.007,USE THEO = NO
MOVE/POINT,152.004,100.03,120.003
HIT/BASIC,152.002,96.001,120.001,152.002,96.001,120.001,USE THEO = NO
MOVE/POINT,152.002,116.003,103.004
HIT/BASIC,152.002,116.003,96.003,152.002,116.003,96.003,USE THEO = NO
MOVE/POINT,152.003,236.004,104
HIT/BASIC,152.003,236.004,96.002,152.003,236.004,96.002,USE THEO = NO
MOVE/POINT,152.003,248.001,118.003
HIT/BASIC,152.003,258.001,118.003,152.003,258.001,118.003,USE THEO = NO
ENDMEAS/
ELL1 = FEAT/ELLIPSE,RECT,IN
THEO/152.0021,176.3774,118.6179,0,0,1,166.668,0.0605,0,1,0
ACTL/152.0021,176.3774,118.6179,0,0,1,166.668,0.0605,0,1,0
```

```

CONSTR/ELLIPSE,BFRE,SCN1.HIT[1],SCN1.HIT[2],SCN1.HIT[3],SCN1.HIT[4],SCN1.HIT[5]
,SCN1.HIT[6],,

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "ELL1" )
ASSIGN/E.MEMORY[3]= E.WM1

...
... Evaluation commands.
...

```

Line Measurement Commands

The command Tutor MLINE command is translated with the equivalent PC-DMIS command:

```

ASSIGN/E.BADMEAS=0
LIN2 =FEAT/LINE, RECT, UNBND
THEO/xxx, yyy, zzz, i, j, k
ACTL/1, 0,-9.9995, 0, 0, 1
MEAS/LINE, 3, RefType

...
... Hits list
...
ENDMEAS/

```

Legend:

xxx = X coordinate	i = vector x component
yyy = Y coordinate	j = vector y component
zzz = Z coordinate	K = vector z component

The **refType** field is:

- WORKPLANE if the L2D Tutor keyword is specified.
- 3D if the L2D Tutor keyword is not specified.

The **Hits list** contains movement, hit, tip commands. The following HIT type may be used:

- HIT/BASIC,,,USE THEO = NO if the COMPENS vector is not specified.
- HIT/BASIC,,,USE THEO = YES if the COMPENS vector is specified. In this case the reversed COMPENS vector is used as HIT vector.



In this case the measurement behavior may be very different from Tutor. See the chapter on "[Measurement Commands with no PC-DMIS Equivalent](#)" for more information.

MLINE (MEMORY [18], 3, COMPENS 0.706038, 0.005531, 0.708153) path PT18 translates to →

```

ASSIGN/E.BADMEAS=0
LIN1 =FEAT/LINE, RECT, UNBND
THEO/1, 0,-9.9995, 0, 0, 1
ACTL/1, 0,-9.9995, 0, 0, 1

```

Tutor Translator

```
MEAS/LINE, 3, 3D
HIT/BASIC, 0, 0, -9.9995, -0.706038, -0.005531, -0.708153, USE THEO = YES
HIT/BASIC, 1, 0, -9.9995, -0.706038, -0.005531, -0.708153, USE THEO = YES
HIT/BASIC, -1.198, 0, -9.9995, -0.706038, -0.005531, -0.708153, USE THEO = YES
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT("LIN1")
ASSIGN/E.MEMORY [18] = E.WM1
...
... Evaluation commands.
...
```

MLINE (MEMORY [19], 3, L2D, COMPENS 0.706038, 0.005531, 0.708153) path PT19 translates to →

```
ASSIGN/E.BADMEAS=0
LIN2 =FEAT/LINE, RECT, UNBND
THEO/1, 0,-9.9995, 0, 0, 1
ACTL/1, 0,-9.9995, 0, 0, 1
MEAS/LINE, 3, WORKPLANE
HIT/BASIC, 0, 0, -9.9995, -0.706038, -0.005531, -0.708153, USE THEO = YES
HIT/BASIC, 1, 0, -9.9995, -0.706038, -0.005531, -0.708153, USE THEO = YES
HIT/BASIC, -1.198, 0, -9.9995, -0.706038, -0.005531, -0.708153, USE THEO = YES
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("LIN2")
ASSIGN/E.MEMORY [19] = E.WM1
...
... Evaluation commands.
...
```

MLINE (MEMORY [20], 3, L2D) path PT20 translates to →

```
ASSIGN/E.BADMEAS=0
LIN3 =FEAT/LINE, RECT, UNBND
THEO/1, 0,-9.9995, 0, 0, 1
ACTL/1, 0,-9.9995, 0, 0, 1
MEAS/LINE, 3, WORKPLANE
HIT/BASIC,0,0,-9.9995,0,-0.0002,-9.9909,USE THEO = NO
HIT/BASIC,1,0,-9.9995,1,-0.0002,-9.9909,USE THEO = NO
HIT/BASIC,-1.198,0,-9.9995,-1.198,-0.0002,-9.9909,USE THEO = NO
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT("LIN3")
ASSIGN/E.MEMORY [20] = E.WM1
...
... Evaluation commands.
```

...

MLINE (MEMORY [20], 3) path PT20 translates to →

```

ASSIGN/E.BADMEAS=0
LIN4 =FEAT/LINE, RECT, UNBND
THEO/1, 0,-9.9995, 0, 0, 1
ACTL/1, 0,-9.9995, 0, 0, 1
MEAS/LINE, 3, 3D
HIT/BASIC, 0, 0, -9.9995, USE THEO = NO
HIT/BASIC, 1, 0, -9.9995, USE THEO = NO
HIT/BASIC, -1.198, 0, -9.9995, USE THEO = NO
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("LIN4")
ASSIGN/E.MEMORY [21] = E.WM1
...
... Evaluation commands.
...

```

The following commands that use ISO algorithm don't have an equivalent PC-DMIS command. See the chapter on "[Measurement Commands with no PC-DMIS Equivalent](#)" for more information.

MLINE (MEMORY [23], 3, L2D, FORM ISO) path PT23

MLINE (MEMORY [24], 3, L2D, FORM ISO, COMPENS 0.089903, 0.704243, 0.704243) path PT24

MLINE (MEMORY [25], 3, FORM ISO, COMPENS 0.089903, 0.704243, 0.704243) path PT25

MLINE (MEMORY [26], 3, FORM ISO) path PT26

Middle Point Measurement Commands

PC-DMIS does not have an equivalent measurement command. Middle point can only be constructed using a set of measured points. The command is translated as a measure SET command and a construction command.

MMIDPT (MEMORY [27], 2) path PT27 translates to →

```

ASSIGN/E.BADMEAS=0
SCN1 = FEAT/SET,RECT
THEO/2.5,20,34.5,0,0,1
ACTL/2.5,20,34.5,-0.3582638,0,-0.9336204
MEAS/SET,2
MOVE/POINT,20,20,30
HIT/BASIC,10,20,39,10,20,39,USE THEO = NO
MOVE/POINT,20,30,30
MOVE/POINT,-15,30,30
MOVE/POINT,-15,20,30

```

Tutor Translator

```
HIT/BASIC,-5,20,30,-5,20,30,USE THEO = NO
MOVE/POINT,30,30,30
ENDMEAS/
    PNT12 = FEAT/POINT,RECT
    THEO/2.5,20,34.5,0,0,1
    ACTL/2.5,20,34.5,0,0,1
    CONSTR/POINT,MID,SCN1.HIT[1],SCN1.HIT[2]

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMNT( "PNT12" )
ASSIGN/E.MEMORY[20] = E.WM1
...
...           Evaluation commands .
...
```

Paraboloid Measurement Commands

PC-DMIS does not have **MPARAB** equivalent for measurement or construction commands. See the chapter on "[Measurement Commands with no PC-DMIS Equivalent](#)" for more information.

MPARAB(MEMORY[100],6) translates to →

```
... Warning message
...
ASSIGN/E.BADMEAS=0
    SCN1 = FEAT/SET,RECT
    THEO/152.0023,176.6688,118.5883,0,0,1
    ACTL/152.0023,176.6688,118.5883,0,0,1
    MEAS/SET,6
    MOVE/POINT,152.003,236.002,130.003
    HIT/BASIC,152.003,236.002,151.514,152.003,236.002,151.514,USE THEO = NO
    MOVE/POINT,152.001,118.002,130.007
    HIT/BASIC,152.001,118.002,130.007,152.001,118.002,130.007,USE THEO = NO
    MOVE/POINT,152.004,100.03,120.003
    HIT/BASIC,152.002,96.001,120.001,152.002,96.001,120.001,USE THEO = NO
    MOVE/POINT,152.002,116.003,103.004
    HIT/BASIC,152.002,116.003,96.003,152.002,116.003,96.003,USE THEO = NO
    MOVE/POINT,152.003,236.004,104
    HIT/BASIC,152.003,236.004,96.002,152.003,236.004,96.002,USE THEO = NO
    MOVE/POINT,152.003,248.001,118.003
    HIT/BASIC,152.003,258.001,118.003,152.003,258.001,118.003,USE THEO = NO
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "SCN1" )
ASSIGN/E.MEMORY[3]= E.WM1
...
...           Evaluation commands .
...
```

Pick (Uncompensated Point) Measurement Commands

PC-DMIS does not have **MPICK** equivalent measurement command. The command can be approximately translated as a measured point (compensated) and extracting uncompensated coordinates using the **RAWHIT** operator.

MPICK (MEMORY [10], 1) translates to →

```

ASSIGN/E.BADMEAS=0
  PNT14 = FEAT/POINT,RECT
  THEO/10,120,30,0,0,1
  ACTL/10,120,40,0,0,1
  MEAS/POINT,1
  MOVE/POINT,12,123,35
  HIT/BASIC,10,120,30,10,120,30,USE THEO = NO
  MOVE/POINT,12,123,350
  ENDMEAS/
  ASSIGN/COMP_POINT = PNT14.RAWHIT[1].XYZ
    PNT15 = GENERIC/POINT,DEPENDENT,RECT,$
    NOM/XYZ,PNT14.TX,PNT14.TY,PNT14.TZ,$
    MEAS/XYZ,COMP_POINT.X,COMP_POINT.Y,COMP_POINT.Z,$
    NOM/IJK,PNT14.TI,PNT14.TJ,PNT14.TZ,$
    MEAS/IJK,PNT14.I,PNT14.J,PNT14.K

  ASSIGN/E.WM2 = E.WM1
  ASSIGN/E.WM1 = TUTORELEMENT( "PNT15" )
  ASSIGN/E.WM1.TYPE = 115
  ASSIGN/E.MEMORY[10] = E.WM1
...
...           Evaluation commands.
...

```

Plane Measurement Commands

MPL (MEMORY [10], 3)

MPL (MEMORY [10], 3, FORM LSM) translates to →

```

ASSIGN/E.BADMEAS=0
  PLN3 = FEAT/PLANE, RECT
  THEO/1, 0,-9.9995, 0, 0, 1
  ACTL/1, 0,-9.9995, 0, 0, 1
  MEAS/PLANE, 3
...
...           move/hit/tip commands
...
ENDMEAS/

```

Tutor Translator

```
ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT("PLN3")
ASSIGN/E.MEMORY [10] = E.WM1
...
...           Evaluation commands .
...
```

The following command that uses ISO algorithm does not have an equivalent PC-DMIS command. See the chapter on "[Measurement Commands with no PC-DMIS Equivalent](#)" for more information.

MPL (MEMORY [10], 3, FORM ISO)

Three Level Plane Measurement Commands.

PC-DMIS does not have equivalent measurement command. A three level plane can only be constructed using a set of measured points. If the three distances have been omitted the values stored with the PL3L_DISTANCES command emulation is used.

MPL3L (MEMORY [10], 3, 10., 10., 15.) path PT27 translates to →

```
ASSIGN/E.BADMEAS=0
SCN1 = FEAT/SET,RECT
THEO/15,13.3333,10,0,0,1
ACTL/15,13.3333,10,0,0,1
MEAS/SET,3
MOVE/POINT,10,10,15
HIT/BASIC,10,10,10,10,10,10,USE THEO = NO
MOVE/POINT,20,10,15
HIT/BASIC,20,10,10,20,10,10,USE THEO = NO
MOVE/POINT,15,20,15
HIT/BASIC,15,20,10,15,20,10,USE THEO = NO
ENDMEAS/
PLN1 = FEAT/PLANE,RECT
THEO/15,7.5,-0.1036,0,-0.5,-0.8660254
ACTL/15,7.5,-0.1036,0,-0.5,-0.8660254
CONSTR/PLANE,OFFSET
ID = SCN1.HIT[1],SCN1.HIT[2],SCN1.HIT[3],,
OFFSET = 10,10,15

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT("PLN1")
ASSIGN/E.MEMORY[10] = E.WM1
...
...           Evaluation commands .
...
```

PL3L_DISTANCE (10.,10.,15.)

MPL3L (MEMORY [10], 3) path PT27 translates to →

```

ASSIGN/E.PL3L.D1 = 10.0
ASSIGN/E.PL3L.D2 = 10.0
ASSIGN/E.PL3L.D3 = 15.0
ASSIGN/E.BADMEAS=0
    SCN2 = FEAT/SET,RECT
    THEO/15,13.3333,10,0,0,1
    ACTL/0,0,0,0,0,1
    MEAS/SET,3
    MOVE/POINT,10,10,15
    HIT/BASIC,10,10,10,10,10,10,USE THEO = NO
    MOVE/POINT,20,10,15
    HIT/BASIC,20,10,10,20,10,10,USE THEO = NO
    MOVE/POINT,15,20,15
    HIT/BASIC,15,20,10,15,20,10,USE THEO = NO
ENDMEAS/
    PLN4 = FEAT/PLANE,RECT
    THEO/15,7.5,20.1036,0,-0.5,0.8660254
    ACTL/15,7.5,20.1036,0,-0.5,0.8660254
    CONSTR/PLANE,OFFSET
    ID = SCN2.HIT[1],SCN2.HIT[2],SCN2.HIT[3],,
    OFFSET = E.PL3L.D1,E.PL3L.D2,E.PL3L.D3

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PLN4" )
ASSIGN/E.MEMORY[10] = E.WM1
...
...
Evaluation commands.
...

```

Shoulder (Compensated Point) Measurement Commands

MSH (MEMORY [10], 1) with NO_APPROACH condition translates to →

```

ASSIGN/E.BADMEAS=0
    PNT1 =FEAT/POINT,RECT
    THEO/10,20,30,0,0,1
    ACTL/10,20,30,0,0,1
    MEAS/POINT,1
    MOVE/POINT,10,20,36
    HIT/BASIC,10,20,30,10,20,30,USE THEO = NO
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT1" )
ASSIGN/E.MEMORY[10] = E.WM1

```

Tutor Translator

```
...
...          Evaluation commands.
...
```

MSH (MEMORY [10], 1) with APPROACH condition translates to →

```
ASSIGN/E.BADMEAS=0
PNT2 =FEAT/POINT,RECT
THEO/10,20,30,E.APPR_VECT.I, E.APPR_VECT.J, E.APPR_VECT.K
ACTL/10,20,30,0,0,1
MEAS/POINT,1
HIT/BASIC,10,20,30, E.APPR_VECT.I, E.APPR_VECT.J, E.APPR_VECT.Z,10,20,30,USE
THEO = YES
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT2" )
ASSIGN/E.MEMORY[10] = E.WM1
...
...          Evaluation commands.
...
```

MSH (MEMORY [10], 1, COMPENS DIRX, DIRY, DIRZ) translates to →

```
ASSIGN/E.BADMEAS=0
PNT3 =FEAT/POINT,RECT
THEO/10,20,30,-DIRX,-DIRY,-DIRZ
ACTL/10,20,30,-0.2015971,-0.3528949,-0.9136869
MEAS/POINT,1
HIT/BASIC,10,20,30,-DIRX,- DIRY,- DIRZ,10,20,30,USE THEO = YES
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT3" )
ASSIGN/E.MEMORY[10] = E.WM1
...
...          Evaluation commands.
...
```

Shoulder commands were DIRX, DIRY and DIRZ may be either variables or immediate values

MSH (MEMORY [10], 1, COMPENSPV) translates to →

(approximately)

```
ASSIGN/E.BADMEAS=0
PNT4 =FEAT/POINT,RECT
THEO/12,34,56,0,0,1
ACTL/12,34,56,0,0,1
MEAS/POINT,1
```

```

MOVE/POINT,12,34,62
HIT/BASIC,12,34,56,12,34,56,USE THEO = NO
ENDMEAS/

ASSIGN/TIPRADIUS = PROBEDATA( "DIAM" )/2
ASSIGN/PICKPNT = PNT4.RAWHIT[1].XYZ
ASSIGN/COMP_VERS = UNIT(PICKPNT)
ASSIGN/COMP_POINT = PNT4.RAWHIT[1].XYZ + COMP_VERS*TIPRADIUS
PNT5 = GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,PNT4.TX,PNT4.TY,PNT4.TZ,$
MEAS/XYZ,COMP_POINT.X,COMP_POINT.Y,COMP_POINT.Z,$
NOM/IJK,PNT4.TI,PNT4.TJ,PNT4.TZ,$
MEAS/IJK,COMP_VERS.I,COMP_VERS.J,COMP_VERS.K

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT5" )
ASSIGN/E.MEMORY[10] = E.WM1
...
...           Evaluation commands.
...

```

MSH (MEMORY [10], 1, COMPENSNPV) translates to →

(approximately)

```

ASSIGN/E.BADMEAS=0
PNT6 = FEAT/POINT,RECT
THEO/12,34,56,0,0,1
ACTL/12,34,56,0,0,1
MEAS/POINT,1
MOVE/POINT,12,34,62
HIT/BASIC,12,34,56,12,34,56,USE THEO = NO
ENDMEAS/

ASSIGN/TIPRADIUS = PROBEDATA( "DIAM" )/2
ASSIGN/PICKPNT = PNT6.RAWHIT[1].XYZ
ASSIGN/COMP_VERS = UNIT(PICKPNT)
ASSIGN/COMP_POINT = PNT6.RAWHIT[1].XYZ - COMP_VERS*TIPRADIUS
PNT7 = GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,PNT6.TX,PNT6.TY,PNT6.TZ,$
MEAS/XYZ,COMP_POINT.X,COMP_POINT.Y,COMP_POINT.Z,$
NOM/IJK,PNT6.TI,PNT6.TJ,PNT6.TZ,$
MEAS/IJK,COMP_VERS.I,COMP_VERS.J,COMP_VERS.K

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT7" )
ASSIGN/E.MEMORY[10] = E.WM1
...
...           Evaluation commands.

```

Tutor Translator

MSH (MEMORY [10], 1, PV2D) translates to →

(approximately)

```
ASSIGN/E.BADMEAS=0
PNT8 =FEAT/POINT,RECT
THEO/12,34,56,0,0,1
ACTL/12,34,56,0,0,1
MEAS/POINT,1
MOVE/POINT,12,34,62
HIT/BASIC,12,34,56,12,34,56,USE THEO = NO
ENDMEAS/
PNT9 =GENERIC/POINT,DEPENDENT,RECT,$
    NOM/XYZ,PNT8.RAWHIT[1].X,PNT8.RAWHIT[1].Y,PNT8.RAWHIT[1].Z,$
        MEAS/XYZ,PNT8.RAWHIT[1].X,PNT8.RAWHIT[1].Y,PNT8.RAWHIT[1].Z,$
NOM/IJK,PNT8.I,PNT8.J,PNT8.K,$
MEAS/IJK,PNT8.I,PNT8.J,PNT8.K
PNT10 =FEAT/POINT,RECT
THEO/12,34,0,0,0,1
ACTL/12,34,0,0,0,1
CONSTR/POINT,PROJ,PNT9,

ASSIGN/TIPRADIUS = PROBEDATA( "DIAM" )/2
ASSIGN/COMP_VERS = UNIT(PNT10.XYZ)
ASSIGN/COMP_POINT = PNT10.XYZ + COMP_VERS*TIPRADIUS
PNT11 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,PNT10.TX,PNT10.TY,PNT10.TZ,$
    MEAS/XYZ,COMP_POINT.X,COMP_POINT.Y,COMP_POINT.Z,$
NOM/IJK,PNT10.TI,PNT10.TJ,PNT10.TZ,$
MEAS/IJK,COMP_VERS.I,COMP_VERS.J,COMP_VERS.K

ASSIGN/E.WM2 = E.WM2
ASSIGN/E.WM1 = TUTORELEMENT( "PNT11" )
ASSIGN/E.MEMORY[10] = E.WM1
...
...           Evaluation commands .
...
```

MSH (MEMORY [10], 1, NPV2D) translates to →

(approximately)

```
ASSIGN/E.BADMEAS=0
PNT12 =FEAT/POINT,RECT
THEO/12,34,56,0,0,1
ACTL/12,34,56,0,0,1
MEAS/POINT,1
MOVE/POINT,12,34,62
HIT/BASIC,12,34,56,12,34,56,USE THEO = NO
```

```

ENDMEAS/
PNT13 = GENERIC/POINT,DEPENDENT,RECT,$
    NOM/XYZ,PNT12.RAWHIT[1].X,PNT12.RAWHIT[1].Y,PNT12.RAWHIT[1].Z,$
        MEAS/XYZ,PNT12.RAWHIT[1].X,PNT12.RAWHIT[1].Y,PNT12.RAWHIT[1].Z,$
NOM/IJK,PNT12.I,PNT12.J,PNT12.K,$
MEAS/IJK,PNT12.I,PNT12.J,PNT12.K
PNT14 = FEAT/POINT,RECT
THEO/11.9953,33.9867,0,0,0,1
ACTL/11.9953,33.9867,0,0,0,1
CONSTR/POINT,PROJ,PNT13,

ASSIGN/TIPRADIUS = PROBEDATA( "DIAM" )/2
ASSIGN/COMP_VERS = UNIT(PNT14.XYZ)
ASSIGN/COMP_POINT = PNT14.XYZ + COMP_VERS*TIPRADIUS
PNT15 = GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,PNT14.TX,PNT14.TY,PNT14.TZ,$
MEAS/XYZ,COMP_POINT.X,COMP_POINT.Y,COMP_POINT.Z,$
NOM/IJK,PNT14.TI,PNT14.TJ,PNT14.TZ,$
MEAS/IJK,COMP_VERS.I,COMP_VERS.J,COMP_VERS.K

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT15" )
ASSIGN/E.MEMORY[10] = E.WM1
...
...           Evaluation commands .
...

```

Slot Measurement Commands

In PC-DMIS a slot can be measured using the slot auto feature. The rules by which hits are taken are completely different from Tutor, therefore it is not possible to use the path defined in the Tutor part program.

All positioning points before the first hit and after the last hit of the Tutor path are added as MOVE/POINT commands before and after the Auto Slot command.

An appropriate warning message (Comment/Oper) is added in order to alert the operator.

The translator uses the Tutor algorithm to compute the slot and use computed values as theoretical values X, Y, Z, CX, CY, CZ, DM (Length) and DM2 (Width) and leaves PC-DMIS to compute hits. This method requires the probe radius and working plane information. The working plane is determined using original hits. The probe radius is not determined during translation. Without the probe radius the slot is not compensated, so DM and DM2 are not meaningful.

	Forcing the Length and Width values to 0 will cause the operator to get an error message at execution time.
---	---

Tutor Translator

MSLOT (MEMORY [10], 6) translates to →

```
ASSIGN/E.BADMEAS=0
  SLT1 = AUTO/ROUND SLOT, SHOWALLPARAMS=YES, SHOWHITS=YES
  THEO/12,23,12,0,-0.7071068,-0.7071068,12,14,1,0,0
  ACTL/12,23,12,0,-0.7071068,-0.7071068,12,14,1,0,0
  TARG/12,23,12,0,-0.7071068,-0.7071068,1,0,0
  THEO_THICKNESS = 0, RECT, IN, ONERROR = NO ,$
  AUTO MOVE = YES, DISTANCE = 0, RMEAS = None, READ POS = NO, FIND HOLE = NO,
  REMEASURE = NO ,$
  INIT = 0, PERM = 0, SPACER = 0, DEPTH = 0, HITS = 6, MEAS ANGLE = 90
  MEAS/ROUND SLOT
  HIT/BASIC,17.2426,26,9,-0.7071068,-0.5,0.5,17.2426,26,9
  HIT/BASIC,19,23,12,-1,0,0,19,23,12
  HIT/BASIC,17.2426,20,15,-0.7071068,0.5,-0.5,17.2426,20,15
  HIT/BASIC,6.7574,20,15,0.7071068,0.5,-0.5,6.7574,20,15
  HIT/BASIC,5,23,12,1,0,0,5,23,12
  HIT/BASIC,6.7574,26,9,0.7071068,-0.5,0.5,6.7574,26,9
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT("SLT1")
ASSIGN/E.MEMORY[10] = E.WM1
...
...
      Evaluation commands.
...
```

Square Slot

MSQSLOT (MEMORY [10], 8) translates to →

```
ASSIGN/E.BADMEAS=0
  SLT2 = AUTO/SQUARE SLOT, SHOWALLPARAMS=YES, SHOWHITS=YES
  THEO/21,43,64,0,0.7071068,0.7071068,0,0,0,-0.7071068,0.7071068
  ACTL/21,43,64,0,0,1,0,1,0,0,1
  TARG/21,43,64,0,0,1,-1,0,0
  THEO_THICKNESS = 0, RECT, IN, ONERROR = NO ,$
  AUTO MOVE = NO, DISTANCE = 0, RMEAS = None, READ POS = NO, FIND HOLE = NO,
  REMEASURE = NO ,$
  INIT = 0, PERM = 0, SPACER = 0, DEPTH = 0 ,$
  WIDTH MINMAX = MINMAX, RADIUS = 0
  MEAS/SQUARE SLOT
  HIT/BASIC,21,43,63.6667,0,0,-1,21,42.9705,63.6667, USE THEO = YES
  HIT/BASIC,21,43,64.3333,0,0,-1,21,42.9705,64.3333, USE THEO = YES
  HIT/BASIC,21,43,64.3333,0,0,1,21,43.0295,64.3333, USE THEO = YES
  HIT/BASIC,21,43,63.6667,0,0,1,21,43.0295,63.6667, USE THEO = YES
  HIT/BASIC,21,43,63.5,0,0,1,20.9705,43,63.5, USE THEO = YES
  HIT/BASIC,21,43,64.5,0,0,-1,21.0295,43,64.5, USE THEO = YES
ENDMEAS/
```

```

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("SLT2")
ASSIGN/E.MEMORY [10] = E.WM1
...
...           Evaluation commands.
...

```

Sphere Measurement Commands

The following commands that use ISO algorithm don't have an equivalent PC-DMIS command. See the chapter on "[Measurement Commands with no PC-DMIS Equivalent](#)" for more information.

MSPH (MEMORY [10], 6)

MSPH (MEMORY [10], 6, FORM ISO) translates to →

```

ASSIGN/E.BADMEAS=0
SPH1 = FEAT/SPHERE,RECT,IN
THEO/10,20,30,0,0,1,23
ACTL/10,20,30,0,0,1,23
MEAS/SPHERE,5
...
...           move/hit/tip commands
...
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT("SPH1")
ASSIGN/E.MEMORY[10] = E.WM1
...
...           Evaluation commands.

```

Thickness Measurement Commands

MTHICHNS (MEMORY [10], 5) translates to →

```

ASSIGN/E.BADMEAS=0
PLN4 = FEAT/PLANE,RECT
THEO/0.2502,350,0.2,1,0,0
ACTL/0.2502,350,0.2,1,0,0
MEAS/PLANE,4
...
...           move/hit/tip commands for plane PLN1
...
ENDMEAS/
PNT1 = FEAT/POINT,RECT

```

Tutor Translator

```
THEO/23,340,0,0,0,1
ACTL/23,340,0,0,0,1
MEAS/POINT,1
...
...      move/hit/tip commands for point PNT1
...
ENDMEAS/
PNT2 = FEAT/POINT,RECT
THEO/0.2502,340,0,1,0,0
ACTL/0.2502,340,0,1,0,0
CONSTR/POINT,PROJ,PLN4,PNT1

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT1" )
ASSIGN/E.WM1.DS = DIST3D( {PNT1}, {PNT2} )
ASSIGN/E.MEMORY[10] = E.WM1
...
...      Evaluation commands.
...
```

Torus Measurement Commands

PC-DMIS does not have **MTORUS** equivalent for either measurement or construction commands. See the chapter on "[Measurement Commands with no PC-DMIS Equivalent](#)" for more information.

MTORUS (MEMORY[100],12) translates to →

```
...
...  Warning message
...
ASSIGN/E.BADMEAS=0
SCN1 = FEAT/SET,RECT
THEO/152.0023,176.6688,118.5883,0,0,1
ACTL/152.0023,176.6688,118.5883,0,0,1
MEAS/SET,6
MOVE/POINT,152.003,236.002,130.003
HIT/BASIC,152.003,236.002,151.514,152.003,236.002,151.514,USE THEO = NO
...
...      other move/hit/tip commands.
...
ENDMEAS/

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "SCN1" )
ASSIGN/E.MEMORY[3]= E.WM1
...
...      Evaluation commands.
```

Canned Circle Measurement Commands

The cycle_circle Tutor command is translated as an Autocircle command within PC-DMIS. The current workplane is used to set the Autofeature vector. The current workplane refers to the workplane at the time of translation and may differ from the workplane that used during part program execution.

CYCLE_CIRCLE (,8,X=63,Y=-78,Z=-3,DM=80)

CYCLE_CIRCLE (WM1,8,X=63,Y=-78,Z=-3,DM=80) translates to →

With current workplane = ZPLUS (Selpl y)

```

ASSIGN/E.BADMEAS = 0
CIR1 =AUTO/CIRCLE,SHOWALLPARAMS = YES,SHOWHITS = YES
THEO/63,-78,-3,0,0,1,80
ACTL/63,-78,-3,0,0,1,80
TARG/63,-78,-3,0,0,1
THEO_THICKNESS = 0,RECT,IN,STRAIGHT,LEAST_SQR,ONERROR = NO,$
AUTO MOVE = BOTH,DISTANCE = 0,RMEAS = None,None,None,$
READ POS = NO,FIND HOLE = NO,REMEASURE = NO,$
NUMHITS = 8,INIT = 1,PERM = 0,SPACER = 0,PITCH = 0,$
START ANG = 0,END ANG = 360,DEPTH = 0,$
ANGLE VEC = 1,0,0
MEAS/CIRCLE
HIT/BASIC,63,-78,-43,0,0,1,63,-78,-43
HIT/BASIC,63,-49.7157,-31.2843,0,-0.7071,0.7071,63,-49.7157,-31.2843
HIT/BASIC,63,-38,-3,0,-1,0,63,-38,-3
HIT/BASIC,63,-49.7157,25.2843,0,-0.7071,-0.7071,63,-49.7157,25.2843
HIT/BASIC,63,-78,37,0,0,-1,63,-78,37
HIT/BASIC,63,-106.2843,25.2843,0,0.7071,-0.7071,63,-106.2843,25.2843
HIT/BASIC,63,-118,-3,0,1,0,63,-118,-3
HIT/BASIC,63,-106.2843,-31.2843,0,0.7071,0.7071,63,-106.2843,-31.2843
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR1" )
...
...      Evaluation commands .
...

```

CYCLE_CIRCLE (WM2,6,X=0,Y=0,Z=-23,DM=30) translates to →

With current workplane = XPLUS (Selpl y)

```

ASSIGN/E.BADMEAS = 0
CIR3 =AUTO/CIRCLE,SHOWALLPARAMS = YES,SHOWHITS = YES
THEO/0,0,-23,1,0,0,30
ACTL/0,0,-23,1,0,0,30
TARG/0,0,-23,1,0,0

```

Tutor Translator

```
THEO_THICKNESS = 0,RECT,IN,STRAIGHT,LEAST_SQR,ONERROR = NO,$
AUTO MOVE = BOTH,DISTANCE = 0,RMEAS = None,None,None,$
READ POS = NO,FIND HOLE = NO,REMEASURE = NO,$
NUMHITS = 6,INIT = 1,PERM = 0,SPACER = 0,PITCH = 0,$
START ANG = 0,END ANG = 360,DEPTH = 0,$
ANGLE VEC = 0,0,-1
MEAS/CIRCLE
HIT/BASIC,0,0,-38,0,0,1,0,0,-38
HIT/BASIC,0,12.9904,-30.5,0,-0.866,0.5,0,12.9904,-30.5
HIT/BASIC,0,12.9904,-15.5,0,-0.866,-0.5,0,12.9904,-15.5
HIT/BASIC,0,0,-8,0,0,-1,0,0,-8
HIT/BASIC,0,-12.9904,-15.5,0,0.866,-0.5,0,-12.9904,-15.5
HIT/BASIC,0,-12.9904,-30.5,0,0.866,0.5,0,-12.9904,-30.5
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR3" )
ASSIGN/E.WM2 = E.WM1
...
... Evaluation commands .
...
```

CYCLE_CIRCLE (MEMORY[1],4,X=51,Y=0,Z=-13,DM=16) translates to →

With current workplane = XPLUS (Selpl y)

```
ASSIGN/E.BADMEAS = 0
CIR4 = AUTO/CIRCLE,SHOWALLPARAMS = YES,SHOWHITS = YES
THEO/51,0,-13,1,0,0,16
ACTL/51,0,-13,1,0,0,16
TARG/51,0,-13,1,0,0
THEO_THICKNESS = 0,RECT,IN,STRAIGHT,LEAST_SQR,ONERROR = NO,$
AUTO MOVE = BOTH,DISTANCE = 0,RMEAS = None,None,None,$
READ POS = NO,FIND HOLE = NO,REMEASURE = NO,$
NUMHITS = 4,INIT = 1,PERM = 0,SPACER = 0,PITCH = 0,$
START ANG = 0,END ANG = 360,DEPTH = 0,$
ANGLE VEC = 0,0,-1
MEAS/CIRCLE
HIT/BASIC,51,0,-21,0,0,1,51,0,-21
HIT/BASIC,51,8,-13,0,-1,0,51,8,-13
HIT/BASIC,51,0,-5,0,0,-1,51,0,-5
HIT/BASIC,51,-8,-13,0,1,0,51,-8,-13
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR4" )
ASSIGN/E.MEMORY[1] = E.WM1
...
... Evaluation commands .
```

CYCLE_CIRCLE (,8,Z=-3,DM=80,Y=-78,X=63) translates to →

With current workplane = YPLUS (Selpl z)

```

ASSIGN/E.BADMEAS = 0
CIR5 =AUTO/CIRCLE,SHOWALLPARAMS = YES,SHOWHITS = YES
THEO/63,-78,-3,0,1,0,80
ACTL/63,-78,-3,0,1,0,80
TARG/63,-78,-3,0,1,0
THEO_THICKNESS = 0,RECT,IN,STRAIGHT,LEAST_SQR,ONERROR = NO,$
AUTO MOVE = BOTH,DISTANCE = 0,RMEAS = None,None,None,$
READ POS = NO,FIND HOLE = NO,REMEASURE = NO,$
NUMHITS = 8,INIT = 1,PERM = 0,SPACER = 0,PITCH = 0,$
START ANG = 0,END ANG = 360,DEPTH = 0,$
ANGLE VEC = 0,0,-1
MEAS/CIRCLE
HIT/BASIC,63,-78,-43,0,0,1,63,-78,-43
HIT/BASIC,63,-49.7157,-31.2843,0,-0.7071,0.7071,63,-49.7157,-31.2843
HIT/BASIC,63,-38,-3,0,-1,0,63,-38,-3
HIT/BASIC,63,-49.7157,25.2843,0,-0.7071,-0.7071,63,-49.7157,25.2843
HIT/BASIC,63,-78,37,0,0,-1,63,-78,37
HIT/BASIC,63,-106.2843,25.2843,0,0.7071,-0.7071,63,-106.2843,25.2843
HIT/BASIC,63,-118,-3,0,1,0,63,-118,-3
HIT/BASIC,63,-106.2843,-31.2843,0,0.7071,0.7071,63,-106.2843,-31.2843
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR5" )

...
...      Evaluation commands .
...

```

CYCLE_CIRCLE (,NUM,X=XX,Y=2*YY,Z=ZZ,DM=50) translates to →

With current workplane = YPLUS (Selpl z)

```

ASSIGN/E.BADMEAS = 0
CIR6 =AUTO/CIRCLE,SHOWALLPARAMS = YES,SHOWHITS = YES
THEO/E.XX,2*E.YY,E.ZZ,0,1,0,50
ACTL/63,-78,-3,0,1,0,50
TARG/E.XX,2*E.YY,E.ZZ,0,1,0
THEO_THICKNESS = 0,RECT,IN,STRAIGHT,LEAST_SQR,ONERROR = NO,$
AUTO MOVE = BOTH,DISTANCE = 0,RMEAS = None,None,None,$
READ POS = NO,FIND HOLE = NO,REMEASURE = NO,$
NUMHITS = E.NUM,INIT = 1,PERM = 0,SPACER = 0,PITCH = 0,$
START ANG = 0,END ANG = 360,DEPTH = 0,$
ANGLE VEC = 0,0,-1
MEAS/CIRCLE
HIT/BASIC,0,0,-25,0,0,1,0,0,-25

```

Tutor Translator

```
HIT/BASIC,-25,0,0,1,0,0,-25,0,0
HIT/BASIC,0,0,25,0,0,-1,0,0,25
HIT/BASIC,25,0,0,-1,0,0,25,0,0
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR6" )
CS8      =CALLSUB/TUTOR_OUTPUT,TUTORSUB.PRG:E.WM1,E.OFMT,E.TH,"WM1",,
COMMENT/DOC,NO,=====
,=    selpl x
WORKPLANE/E.ZPLUS
ASSIGN/E.SELPL = E.ZPLUS
...
...      Evaluation commands .
...
```

CYCLE_CIRCLE (WM2,6,X=MEA[1]|X,Y=MEA[1]|Y,Z=-23,DM=30) translates to →

With current workplane = ZPLUS (Selpl x)

```
ASSIGN/E.BADMEAS = 0
CIR7  =AUTO/CIRCLE,SHOWALLPARAMS = YES,SHOWHITS = YES
THEO/E.MEA[1].X,E.MEA[1].Y,-23,0,0,1,30
ACTL/63,-78,-23,0,0,1,30
TARG/E.MEA[1].X,E.MEA[1].Y,-23,0,0,1
THEO_THICKNESS = 0,RECT,IN,STRAIGHT,LEAST_SQR,ONERROR = NO,$
AUTO MOVE = BOTH,DISTANCE = 0,RMEAS = None,None,None,$
READ POS = NO,FIND HOLE = NO,REMEASURE = NO,$
NUMHITS = 6,INIT = 1,PERM = 0,SPACER = 0,PITCH = 0,$
START ANG = 0,END ANG = 360,DEPTH = 0,$
ANGLE VEC = 1,0,0
MEAS/CIRCLE
HIT/BASIC,15,0,-23,-1,0,0,15,0,-23
HIT/BASIC,7.5,12.9904,-23,-0.5,-0.866,0,7.5,12.9904,-23
HIT/BASIC,-7.5,12.9904,-23,0.5,-0.866,0,-7.5,12.9904,-23
HIT/BASIC,-15,0,-23,1,0,0,-15,0,-23
HIT/BASIC,-7.5,-12.9904,-23,0.5,0.866,0,-7.5,-12.9904,-23
HIT/BASIC,7.5,-12.9904,-23,-0.5,0.866,0,7.5,-12.9904,-23
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR7" )
ASSIGN/E.WM2 = E.WM1
...
...      Evaluation commands .
...
```

CYCLE_CIRCLE (MEMORY[1],4,X=51,Y=0,Z=-13,DM=16,10.0,30.0) translates to →

With current workplane = ZPLUS (Selpl x)

```

ASSIGN/E.BADMEAS = 0
CIR8 =AUTO/CIRCLE,SHOWALLPARAMS = YES,SHOWHITS = YES
THEO/51,0,-13,0,0,1,16
ACTL/51,0,-13,0,0,1,16
TARG/51,0,-13,0,0,1
THEO_THICKNESS = 0,RECT,IN,STRAIGHT,LEAST_SQR,ONERROR = NO,$
AUTO MOVE = BOTH,DISTANCE = 0,RMEAS = None,None,None,$
READ POS = NO,FIND HOLE = NO,REMEASURE = NO,$
NUMHITS = 4,INIT = 1,PERM = 0,SPACER = 0,PITCH = 0,$
START ANG = 10,END ANG = 30,DEPTH = 0,$
ANGLE VEC = 1,0,0
MEAS/CIRCLE
HIT/BASIC,58.8785,1.3892,-13,-0.9848,-0.1736,0,58.8785,1.3892,-13
HIT/BASIC,58.6639,2.2944,-13,-0.958,-0.2868,0,58.6639,2.2944,-13
HIT/BASIC,58.3457,3.1686,-13,-0.9182,-0.3961,0,58.3457,3.1686,-13
HIT/BASIC,57.9282,4,-13,-0.866,-0.5,0,57.9282,4,-13
ENDMEAS/

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR8" )
ASSIGN/E.MEMORY[1] = E.WM1
...
...
Evaluation commands.
...

```

CYCLE_CIRCLE (MEMORY[1],4,X=51,Y=0,Z=-13,DM=16,ZZ,YY) translates to →

With current workplane = ZPLUS (Selpl x)

```

CIR9 =AUTO/CIRCLE,SHOWALLPARAMS = YES,SHOWHITS = YES
THEO/51,0,-13,0,0,1,16
ACTL/51,0,-13,0,0,1,16
TARG/51,0,-13,0,0,1
THEO_THICKNESS = 0,RECT,IN,STRAIGHT,LEAST_SQR,ONERROR = NO,$
AUTO MOVE = BOTH,DISTANCE = 0,RMEAS = None,None,None,$
READ POS = NO,FIND HOLE = NO,REMEASURE = NO,$
NUMHITS = 4,INIT = 1,PERM = 0,SPACER = 0,PITCH = 0,$
START ANG = E.ZZ,END ANG = E.YY,DEPTH = 0,$
ANGLE VEC = 1,0,0
MEAS/CIRCLE
HIT/BASIC,59,0,-13,-1,0,0,59,0,-13
HIT/BASIC,51,8,-13,0,-1,0,51,8,-13
HIT/BASIC,43,0,-13,1,0,0,43,0,-13
HIT/BASIC,51,-8,-13,0,1,0,51,-8,-13

```

Tutor Translator

```
ENDMEAS /  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT( "CIR9" )  
ASSIGN/E.MEMORY[1] = E.WM1  
...  
... Evaluation commands.  
...
```

Construction Commands

The Tutor construction command (indirect measure) has the following formats:

1. **Ielem (ResMem,PntNum,ElemNum,ElemSpec) [SAVEPTS]**
2. **Ielem (ResMem,Element1, Element2, Element3, , ElementN) [SAVEPTS]**
3. **Ielem (ResMem, PntNum, ScannedPointFile.SCN) [SAVEPTS]**



The third command in this list does not translate to PC-DMIS.

Listed below are the parameters for the above commands:

- []: Optional field or element specific keyword.
- **Ielem**: Element keyword - ICIR, ICONE
- **ResMem**: Destination memory - If omitted means E.WM1.
- **PntNum**: Number of points (or point class elements) to be processed. Tutor allows variables but only commands with known hits number are translated.
- **ElemNum**: Index in the memory array of the first of PntNum elements. This may be a variable.
- **ElemSpec**: Element specific parameter.
- **SAVEPTS**: Measured points are not discarded.
- **Element1, . . . , ElementN**: Elements to be processed.
- **ScannedPointFile.SCN**: File containing scansion points to be processed.

Construction commands that have equivalent command on PC-DMIS, are translated as:

- Insert **FORMAT** command. Translator takes care of FORMAT commands for each element. See the "[Format Command](#)" chapter.
- **Construction Commands** - This command requires the definition of the feature theoretical values. See the "[THEO Command](#)" chapter.
- If **RefSysNum** is defined the current reference system is recalled:
RECALL/ALIGNMENT,INTERNAL, E.ALIGNMENTS[E.CUR_ALIGN]
- **Copy Structure** - Copy the structure of E.WM1 from E.WM2.
ASSIGN/E.WM2 = E.WM1
- **Data Memorization** in the structure E.WM1.
- If the memory of destination is not E.WM1, data is stored also in the destination memory structure.
- **Dimension Command** and **True Position** computing if necessary
- **Profile Command** - Computing if necessary
- **Evaluation output**. - For more information, see [Element Evaluation](#).

Construction Commands with no PC-DMIS Equivalent

Cones are translated as generic constructed cones. A Comment/Oper command is added in order to notify the operator about missing compatibility with the original Part Program. A Comment/Oper command is added for ISO commands as well.

ISO Commands

Measurement commands that use the ISO algorithm don't have an equivalent PC-DMIS command. The commands are approximately translated similar to a command that uses a Least Square algorithm.



The Measurement Command examples given below do not represent all of the possible measurement commands, but do provide a representative list of examples. All Tutor commands are listed, but not all their possible parameter combinations are given.

Circle Construction Commands

ICIR (MEMORY [20], 4, MEMORY[101]) translates to →

```
CIR8 = FEAT/CIRCLE, RECT, OUT, LEAST_SQR
THEO/6989739.046,155.7689,121.503,0,0,1,13513201.6179
ACTL/6989739.2093,155.7689,121.4961,0,0,1,13513201.9335
CONSTR/CIRCLE,BF,E.MEMORY[101].ID,E.MEMORY[102].ID,E.MEMORY[103].ID,
E.MEMORY[104].ID,,

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR8" )
ASSIGN/E.MEMORY [ 20 ] = E.WM1
...
...           Evaluation commands .
...
```

ICIR (MEMORY [21], MEMORY [103], MEMORY [105], MEMORY [107], MEMORY [109]) translates to →

```
CIR9 = FEAT/CIRCLE, RECT, OUT, LEAST_SQR
THEO/18.0389,165.9559,64.0007,0,0,1,302.3213
ACTL/18.0389,165.9559,64.0155,0,0,1,302.3213
CONSTR/CIRCLE,BF,E.MEMORY[103].ID,E.MEMORY[105].ID,E.MEMORY[107].ID,E.MEMORY[10
9 ].ID,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ( "CIR9" )
ASSIGN/E.MEMORY [ 21 ] = TUTORELEMENT("CIR9" )

...
...           Evaluation commands .
...
```

3D Circle Command

PC-DMIS does not have the 3D circle construction. Tutor computes the 3D circle with the following steps:

1. Compute Best Fit plane.
2. Compute Best Fit cylinder forcing the plane axis as cylinder axis.
3. Project the cylinder on the plane.

In PC-DMIS the 3D circle can emulated approximately using the following steps:

1. Compute Best Fit plane.
2. Create a temporary alignment using plane vector: LEVEL ZPLUS
3. Set WORKPLANE/ZPLUS
4. Compute Best Fit circle.
5. Recall previous reference system
6. Set previous WORKPLANE/E.SELPL.
7. Evaluation

ICIR(MEMORY[22],3, MEMORY[108],C3D) translates to →

```
PLN5 =FEAT/PLANE, RECT
THEO/8.3333,16.6667,23.3333,0,-0.8944272,0.4472136
ACTL/8.3333,16.6667,23.3333,0,-
0.8944272,0.4472136CONSTR/PLANE,BF,E.MEMORY[108].ID,E.MEMORY[109].ID,E.MEMORY[1
10].ID,
A6 =ALIGNMENT/START,RECALL: E.ALIGNMENTS[E.CUR_ALIGN], LIST= YES
ALIGNMENT/LEVEL, ZPLUS, PLN5
ALIGNMENT/END
WORKPLANE/ZPLUS
CIR_1 =FEAT/CIRCLE,RECT,OUT,LEAST_SQR
THEO/2.5,19.0066,-4.4721,0,0,1,36.7423
ACTL/2.5,19.0066,-4.4677,0,0,1,36.7423
CONSTR/CIRCLE,BF,E.MEMORY[108].ID,E.MEMORY[109].ID,E.MEMORY[110].ID,,
RECALL/ALIGNMENT,INTERNAL,E.ALIGNMENTS[E.CUR_ALIGN]
WORKPLANE/E.SELPL
CIR10 =FEAT/CIRCLE,RECT,OUT
THEO/2.5,12.5,25,0,-0.8944272,0.4472136,22.6691
ACTL/2.5,12.5,25,0,-0.8944272,0.4472136,22.6691
CONSTR/CIRCLE,PROJ,CIR_1,PLN5

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR10" )
ASSIGN/E.MEMORY [ 22 ] = E.WM1
...
...           Evaluation commands .
...
```

ICIR (MEMORY [23], MEMORY [101], MEMORY [109], MEMORY [110], C3D) translates to →

```

PLN7 =FEAT/PLANE, RECT
THEO/52.3343,92.0007,66.6677,0,0.4538087,-0.8910991
ACTL/52.3343,92.0007,76.6677,0,0.4538087,-0.8910991
CONSTR/PLANE,BF,E.MEMORY[101].ID,E.MEMORY[109].ID,E.MEMORY[110].ID,,,
A7 =ALIGNMENT/START, RECALL: E.ALIGNMENTS[E.CUR_ALIGN] , LIST= YES
ALIGNMENT/LEVEL, ZPLUS, PLN7
ALIGNMENT/END
WORKPLANE/ZPLUS
CIR_2 =FEAT/CIRCLE, RECT, OUT, LEAST_SQR
THEO/2.5,-198.6235,-17.657,0,0,1,334.7111
ACTL/2.5,-198.6239,-17.648,0,0,1,334.7116
CONSTR/CIRCLE,BF,E.MEMORY[101].ID,E.MEMORY[109].ID,E.MEMORY[110].ID,,,
RECALL/ALIGNMENT, INTERNAL, E.ALIGNMENTS [E.CUR_ALIGN]
WORKPLANE/E.SELPL
CIR11 = FEAT/CIRCLE, RECT, OUT
THEO/2.5,168.9808,105.8712,0,0.4538087,-0.8910991,334.7116
ACTL/2.5,168.9808,105.8712,0,0.4538087,-0.8910991,334.7116
CONSTR/CIRCLE, PROJ, CIR_2, PLN7

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("CIR11")
ASSIGN/E.MEMORY [23] = E.WM1
...
... Evaluation commands.
...

```

ICIR (MEMORY [24], 3, MEMORY [105]) CMIN translates to →

```

CIR12 =FEAT/CIRCLE,RECT,OUT,MIN_CIRCSC
THEO/3.3333,5,5,1,0,0,14.1421
ACTL/3.3333,5,5,1,0,0,14.1421
CONSTR/CIRCLE,BF,E.MEMORY[105].ID,E.MEMORY[106].ID,E.MEMORY[107].ID,,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT("CIR12")
ASSIGN/E.MEMORY[24] = E.WM1
...
... Evaluation commands.
...
```

ICIR (MEMORY [25], MEMORY [115], MEMORY [106], MEMORY [109]) CMIN translates to →

```

CIR13 =FEAT/CIRCLE,RECT,OUT,MIN_CIRCSC
THEO/6.6667,10,20,1,0,0,44.7214
ACTL/6.6667,10,20,1,0,0,44.7214
CONSTR/CIRCLE,BF,E.MEMORY[115].ID,E.MEMORY[106].ID,E.MEMORY[109].ID,,
```

Tutor Translator

```
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR13" )
ASSIGN/E.MEMORY[25] = E.WM1
...
...
... Evaluation commands .
...
```

ICIR (MEMORY [26], 4, MEMORY [114]) CMAX translates to →

```
CIR14 =FEAT/CIRCLE,RECT,OUT,MAX_INSC
THEO/10,70,35,1,0,0,100.4988
ACTL/10,70,35,1,0,0,100.4988
CONSTR/CIRCLE,BF,E.MEMORY[114].ID,E.MEMORY[115].ID,E.MEMORY[116].ID,E.MEMORY[117].ID,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR14" )
ASSIGN/E.MEMORY[26] = E.WM1
...
...
... Evaluation commands .
...
```

ICIR (MEMORY [27], MEMORY[115],MEMOTY[114],MEMORY[116],MEMORY[107]) CMAX translates to →

```
CIR15 =FEAT/CIRCLE,RECT,OUT,MAX_INSC
THEO/10,70,6.6667,1,0,0,120.185
ACTL/10,70,6.6667,1,0,0,120.185
CONSTR/CIRCLE,BF,E.MEMORY[115].ID,E.MEMORY[114].ID,E.MEMORY[116].ID,E.MEMORY[107].ID,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR15" )
ASSIGN/E.MEMORY[27] = E.WM1
...
...
... Evaluation commands .
...
```

ICIR (MEMORY[28], 3, MEMORY [101]) DM 285.7785 translates to →

```
CIR16 =FEAT/CIRCLE,RECT,OUT,FIXED_RAD
THEO/104.6687,106.197,127.8452,1,0,0,285.7785
ACTL/104.6687,106.197,127.8452,1,0,0,285.7785
CONSTR/CIRCLE,BF,E.MEMORY[105].ID,E.MEMORY[106].ID,E.MEMORY[107].ID,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR16" )
ASSIGN/E.MEMORY[28] = E.WM1
```

```

...
      Evaluation commands.
...

```

ICIR (MEMORY [29], MEMORY [115], MEMORY [106], MEMORY [109]) DM 322.7269 translates to →

```

CIR17 =FEAT/CIRCLE,RECT,OUT,FIXED_RAD
THEO/50.7685,93.3268,170.6678,1,0,0,322.7269
ACTL/50.7685,93.3268,170.6678,1,0,0,322.7269
CONSTR/CIRCLE,BF,E.MEMORY[115].ID,E.MEMORY[106].ID,E.MEMORY[109].ID,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CIR17" )
ASSIGN/E.MEMORY[ 29 ] = E.WM1

...
      Evaluation commands.
...

```

Not Aligned Circle

PC-DMIS does not have the Not Aligned Circle construction. Tutor computes the Not-aligned circle with the following steps:

1. Using first 4 points computes Best Fit plane.
2. Compute Best Fit cylinder forcing the plane axis as cylinder axis.
3. Project the cylinder on the plane.

In PC-DMIS the Not-aligned circle can emulated approximately using the following steps:

1. Compute Best Fit plane.
2. Create a temporary alignment using plane vector: LEVEL ZPLUS
3. Set WORKPLANE/ZPLUS
4. Compute BF circle.
5. Projects the BF circle on the plane.
6. Recall previous reference system
7. Set previous WORKPLANE/E.SELPL.
8. Evaluation

ICIR_NA (MEMORY [28], 8, MEMORY [200]) translates to →

```

PLN8 =FEAT/PLANE, RECT
THEO/-8.6354,-10.9527,12.3279,0.5083415,-0.2241445,0.8314735
ACTL/-8.6354,-10.9527,12.3279,0.5083415,-0.2241445,0.8314735
CONSTR/PLANE,BF,E.MEMORY[200].ID,E.MEMORY[201].ID,E.MEMORY[202].ID,E.MEMORY[203]
].ID,,

A5 =ALIGNMENT/START, RECALL: E.ALIGNMENTS [E.CUR_ALIGN], LIST= YES
ALIGNMENT/LEVEL, ZPLUS, PLN8
ALIGNMENT/END
WORKPLANE/ZPLUS

CIR_6 =FEAT/CIRCLE, RECT, IN, LEAST_SQR
THEO/10, 10,-2, 0, 0,1,20
ACTL/10, 10,-2, 0, 0,1,20

```

Tutor Translator

```
CONSTR/CIRCLE,BF,E.MEMORY[204].ID,E.MEMORY[205].ID,E.MEMORY[206].ID,E.MEMORY[207].ID,,  
CIR11 =FEAT/CIRCLE, RECT, OUT  
THEO/10, 10,0,0,0,1,20  
ACTL/10, 10,0,0,0,1,20  
CONSTR/CIRCLE, PROJ, CIR_6, PLN8  
RECALL/ALIGNMENT, INTERNAL, E.ALIGNMENTS [E.CUR_ALIGN]  
WORKPLANE/E.SELPL  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT ("CIR11")  
ASSIGN/E.MEMORY [28] = E.WM1  
...  
... Evaluation commands.  
...
```

ICIR_NA (MEMORY [29], MEMORY [201], MEMORY [203], MEMORY [200], MEMORY [202], MEMORY [206], MEMORY [204], MEMORY [207], MEMORY [205]) translates to →

```
PLN9 =FEAT/PLANE, RECT  
THEO/8.6354,10.9527,12.3279,-0.5062175,0.223208,0.8330198  
ACTL/8.6354,10.9527,12.3279,-0.5062175,0.223208,0.8330198  
  
CONSTR/PLANE,BF,E.MEMORY[201].ID,E.MEMORY[203].ID,E.MEMORY[200].ID,E.MEMORY[202].ID,,  
A6 =ALIGNMENT/START, RECALL: E.ALIGNMENTS [E.CUR_ALIGN], LIST= YES  
ALIGNMENT/LEVEL, ZPLUS, PLN8  
ALIGNMENT/END  
WORKPLANE/ZPLUS  
CIR_7 =FEAT/CIRCLE, RECT, OUT, LEAST_SQR  
THEO/5.0322,11.4803,19.6313,-0.3310643,0.4461482,0.8314735,21.57  
ACTL/5.0322,11.4803,19.6313,-0.3310643,0.4461482,0.8314735,21.57  
CONSTR/CIRCLE,BF,E.MEMORY[206].ID,E.MEMORY[204].ID,E.MEMORY[207].ID,E.MEMORY[205].ID,,  
CIR18 =FEAT/CIRCLE, RECT, OUT  
THEO/3.2641,13.8814,20.8985,-0.5457286,0.7410799,0.3911277,21.57  
ACTL/13.9727,4.8518,8.3651,0.0000003,0.0000003,1,21.57  
CONSTR/CIRCLE, PROJ, CIR_7, PLN9  
RECALL/ALIGNMENT, INTERNAL, E.ALIGNMENTS [E.CUR_ALIGN]  
WORKPLANE/E.SELPL  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT ("CIR18")  
ASSIGN/E.MEMORY [29] = E.WM1  
...  
... Evaluation commands.  
...
```

Cone Construction Commands

Tutor and PC-DMIS have different meanings for XYZ fields. Tutor views the XYZ fields as the vertex of the cone while PC-DMIS views the XYZ fields as the coordinates of the centroid of the first section circle, displayed in the THEO command. For Cone Construction commands the WIDE keyword is ignored.

In Tutor the TutorElement X, Y and Z fields are the vertex of the cone.

To evaluate the Vertex of the cone is not possible to use the DIM/LOCATION evaluation but the TUTOR_OUTPUT subroutine call.

ICONE(MEMORY [10], 8, MEMORY [210])

ICONE(MEMORY [10], 8, MEMORY [210],FORM LSM)

ICONE(MEMORY [10], 8, MEMORY [210],FORM LSM, WIDE)

ICONE(MEMORY [10], 8, MEMORY [210],WIDE) translates to →

```

CON2 =FEAT/CONE,RECT,OUT,LENG
THEO/10.0002,10.0014,105.009,0.0004996,0.0004597,-
0.9999998,5.0095,7.9994,10.0018
ACTL/10.0002,10.0014,105.009,0.0004996,0.0004597,-
0.9999998,5.0095,7.9994,10.0018
CONSTR/CONE,BF,E.MEMORY[210].ID,E.MEMORY[211].ID,E.MEMORY[212].ID,E.MEMORY[213]
.ID,E.MEMORY[214].ID,E.MEMORY[215].ID,E.MEMORY[216].ID,E.MEMORY[217].ID,,

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CON2" )
ASSIGN/E.MEMORY[10]= E.WM1
...
...           Evaluation commands .
...
...
```

**ICONE(MEMORY [10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[215],
MEMORY[213], MEMORY[216], MEMORY[212])**

**ICONE(MEMORY [10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[215],
MEMORY[213], MEMORY[216], MEMORY[212],FORM LSM)**

**ICONE(MEMORY [10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[215],
MEMORY[213], MEMORY[216], MEMORY[212],FORM LSM, WIDE)**

**ICONE(MEMORY [10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[205],
MEMORY[213], MEMORY[216], MEMORY[212],WIDE) translates to →**

```

CON2 =FEAT/CONE,RECT,OUT,LENG
THEO/10.0002,10.0014,105.009,0.0004996,0.0004597,-
0.9999998,5.0095,7.9994,10.0018
```

Tutor Translator

```
ACTL/10.0002,10.0014,105.009,0.0004996,0.0004597,-  
0.9999998,5.0095,7.9994,10.0018  
CONSTR/CONE,BF,E.MEMORY[10].ID,E.MEMORY[211].ID,E.MEMORY[214].ID,E.MEMORY[210].  
ID,E.MEMORY[217].ID,E.MEMORY[215].ID,E.MEMORY[213].ID,E.MEMORY[216].ID,E.MEMORY  
[217].ID,,  
  
ASSIGN/E.WM2=E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT( "CON2" )  
ASSIGN/E.MEMORY[10]= E.WM1  
...  
... Evaluation commands.  
...
```



THEO fields are:

THEO/ XCoord, YCoord, ZCoord, I, J, K, Angle, Diameter1, Diameter2, where XCoord, YCoord and ZCoord are the coordinate of the centroid of the circle (section circle). Diameters 1 and 2 are not filled by translator.

The following Cone Construction commands don't have equivalent command on PC-DMIS so dirX, dirY, dirZ are ignored. See the section on "[Construction Commands with no PC-DMIS Equivalent](#)" for more information.

ICONE(MEMORY[10],8, MEMORY[210],dirX, dirY,dirZ)

ICONE(MEMORY[10],8, MEMORY[210],dirX, dirY,dirZ,FORM LSM)

ICONE(MEMORY[10],8, MEMORY[210],dirX, dirY,dirZ,FORM LSM,WIDE)

ICONE(MEMORY[10],8, MEMORY[210],dirX, dirY,dirZ,WIDE) translates to →

```
CON2 =FEAT/CONE,RECT,OUT,LENG  
THEO/10.0002,10.0014,105.009,0.0004996,0.0004597,-  
0.9999998,5.0095,7.9994,10.0018  
ACTL/10.0002,10.0014,105.009,0.0004996,0.0004597,-  
0.9999998,5.0095,7.9994,10.0018  
CONSTR/CONE,BF,E.MEMORY[210].ID,E.MEMORY[211].ID,E.MEMORY[212].ID,E.MEMORY[213].  
.ID,E.MEMORY[214].ID,E.MEMORY[215].ID,E.MEMORY[216].ID,E.MEMORY[217].ID,,  
  
ASSIGN/E.WM2=E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT( "CON2" )  
ASSIGN/E.MEMORY[10]= E.WM1  
...  
... Evaluation commands.  
...
```

**ICONE(MEMORY[10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[205],
MEMORY[213], MEMORY[216], MEMORY[212],dirX, dirY,dirZ)**

**ICONE(MEMORY[10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[205],
MEMORY[213], MEMORY[216], MEMORY[212],dirX, dirY,dirZ,FORM LSM)**

**ICONE(MEMORY[10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[205],
MEMORY[213], MEMORY[216], MEMORY[212],dirX, dirY, dirZ, FORM LSM, WIDE)**

**ICONE(MEMORY[10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[205],
MEMORY[213], MEMORY[216], MEMORY[212],dirX, dirY, dirZ, WIDE)** translates to →

```

CON2 =FEAT/CONE,RECT,OUT,LENG
THEO/10.0002,10.0014,105.009,0.0004996,0.0004597,-
0.9999998,5.0095,7.9994,10.0018
ACTL/10.0002,10.0014,105.009,0.0004996,0.0004597,-
0.9999998,5.0095,7.9994,10.0018
CONSTR/CONE,BF,E.MEMORY[210].ID,E.MEMORY[211].ID,E.MEMORY[212].ID,E.MEMORY[213]
.ID,E.MEMORY[214].ID,E.MEMORY[215].ID,E.MEMORY[216].ID,E.MEMORY[217].ID,,

ASSIGN/E.WM2=E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CON2" )
ASSIGN/E.MEMORY[10]= E.WM1
...
... Evaluation commands .
...

```

The following commands that use ISO algorithm don't have an equivalent PC-DMIS command.
See the section on "[Construction Commands with no PC-DMIS Equivalent](#)" for more information.

ICONE(MEMORY [10] ,8, MEMORY [210], FORM ISO)

ICONE(MEMORY [10], 8, MEMORY [210], dirX, dirY, dirZ, FORM ISO)

**ICONE(MEMORY [10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[205],
MEMORY[213], MEMORY[216], MEMORY[212], dirX, dirY, dirZ, FORM ISO, WIDE)**

**ICONE(MEMORY [10] , MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[205],
MEMORY[213], MEMORY[216], MEMORY[212], FORM ISO)**

**ICONE(MEMORY [10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[205],
MEMORY[213], MEMORY[216], MEMORY[212], dirX, dirY, dirZ, FORM ISO)**

**ICONE(MEMORY [10], MEMORY[211], MEMORY[214], MEMORY[210], MEMORY[217], MEMORY[205],
MEMORY[213], MEMORY[216], MEMORY[212], dirX, dirY, dirZ, FORM ISO, WIDE)**

Cylinder Construction Commands



THEO fields for Cylinder construction commands are:

THEO/ XCoord, YCoord, ZCoord, I, J, K, Diameter, Length. Length is not filled by translator.

ICYL (MEMORY[32], 8, MEMORY[220], SDM)

ICYL (MEMORY[32], 8, MEMORY[220], SDM, FORM LSM) translates to →

```

CYL2 =FEAT/CYLINDER, RECT, OUT, LEAST_SQR
THEO/10.0036,10.004,112.446,0.0000238,-0.0000024,1,10.0006,24.89

```

Tutor Translator

```
ACTL/10.0036,10.004,112.446,0.0000238,-0.0000024,1,10.0006,24.  
CONSTR/CYLINDER,BF,E.MEMORY[220].ID,E.MEMORY[221].ID,E.MEMORY[222].ID,  
E.MEMORY[223].ID,E.MEMORY[224].ID,E.MEMORY[225].ID,E.MEMORY[226].ID,  
E.MEMORY[227].ID,,
```

```
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT("CYL2")  
ASSIGN/E.MEMORY[32] = E.WM1  
...  
... Evaluation commands.  
...
```

**ICYL (MEMORY [32], MEMORY[222], MEMORY[220], MEMORY[221], MEMORY[225], MEMORY[227],
MEMORY[223], MEMORY[226], MEMORY[224], SDM)**

**ICYL (MEMORY [32], MEMORY[222], MEMORY[220], MEMORY[221], MEMORY[225], MEMORY[227],
MEMORY[223], MEMORY[226], MEMORY[224], SDM, FORM LSM)** translates to →

```
CYL3 =FEAT/CYLINDER,RECT,OUT,LEAST_SQR  
THEO/10.0036,10.004,112.4466,0.0000238,-0.0000024,1,10.0006,24.8889  
ACTL/10.0036,10.004,112.4466,0.0000238,-0.0000024,1,10.0006,24.8889  
CONSTR/CYLINDER,BF,E.MEMORY[222].ID,E.MEMORY[220].ID,E.MEMORY[221].ID,  
E.MEMORY[225].ID,E.MEMORY[227].ID,E.MEMORY[223].ID,E.MEMORY[226].ID,  
E.MEMORY[224].ID,,  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT("CYL2")  
ASSIGN/E.MEMORY[32] = TUTORELEMENT("CYL2")  
...  
... Evaluation commands.  
...
```

The following commands with fixed axis don't actually have equivalent command on PC-DMIS.
See the chapter on "[Construction Commands with no PC-DMIS Equivalent](#)" for more information.

ICYL (MEMORY [32], 8, MEMORY[220], SDM, dirX, dirY, dirZ)

ICYL (MEMORY [32], 8, MEMORY[220], SDM, dirX, dirY, dirZ, FORM LSM) translates to →

```
CYL2 =FEAT/CYLINDER, RECT, OUT, LEAST_SQR  
THEO/10.0036,10.004,112.446,0.0000238,-0.0000024,1,10.0006,24.89  
ACTL/10.0036,10.004,112.446,0.0000238,-0.0000024,1,10.0006,24.  
CONSTR/CYLINDER,BF,E.MEMORY[220].ID,E.MEMORY[221].ID,E.MEMORY[222].ID,  
E.MEMORY[223].ID,E.MEMORY[224].ID,E.MEMORY[225].ID,E.MEMORY[226].ID,  
E.MEMORY[227].ID,,  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT("CYL2")  
ASSIGN/E.MEMORY[32] = E.WM1  
...  
... Evaluation commands.  
...
```

**ICYL (MEMORY [32], MEMORY[222], MEMORY[220], MEMORY[221], MEMORY[225], MEMORY[227],
MEMORY[223], MEMORY[226], MEMORY[224], SDM,
dirX, dirY, dirZ)**

**ICYL (MEMORY [32], MEMORY[222], MEMORY[220], MEMORY[221], MEMORY[225], MEMORY[227],
MEMORY[223], MEMORY[226], MEMORY[224], SDM,
dirX, dirY, dirZ, FORM LSM)** translates to →

```

CYL3      = FEAT/CYLINDER,RECT,OUT,LEAST_SQR
THEO/10.0036,10.004,112.4466,0.0000238,-0.0000024,1,10.0006,24.8889
ACTL/10.0036,10.004,112.4466,0.0000238,-0.0000024,1,10.0006,24.8889
CONSTR/CYLINDER,BF,E.MEMORY[222].ID,E.MEMORY[220].ID,E.MEMORY[221].ID,
E.MEMORY[225].ID,E.MEMORY[227].ID,E.MEMORY[223].ID,E.MEMORY[226].ID,
E.MEMORY[224].ID,,
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "CYL2" )
ASSIGN/E.MEMORY[32] = E.WM1
...
...
Evaluation commands.
...

```

The following commands that use ISO algorithm don't have an equivalent PC-DMIS command.
See the chapter on "[Construction Commands with no PC-DMIS Equivalent](#)" for more information.

ICYL (MEMORY [10], 8, MEMORY[MemIndex], SDM, FORM ISO)

ICYL (MEMORY [10], 8, MEMORY[MemIndex], SDM, dirX, dirY, dirZ, FORM ISO)

ICYL (MEMORY [10], MEM1, MEM2,...MEMN, SDM, FORM ISO)

ICYL (MEMORY [10], MEM1, MEM2,...MEMN SDM, dirX, dirY, dirZ, FORM ISO) translates to →

The command to construct a 2 diameter cylinder (without SDM keyword) does not have equivalent command on PC-DMIS and is not translated. A Comment/Oper is added in order to notify the operator about missing compatibility with original Part Program.

Line Construction Commands

The command ILINE is translated with the equivalent PC-DMIS command:

```

LIN2 =FEAT/LINE, RECT, UNBND
THEO/xxx, yyy, zzz, i, j, k
ACTL/1, 0,-9.9995, 0, 0, 1
CONSTR/LINE,BF, RefType,E.MEMORY[103].ID,E.MEMORY[106].ID,,

```

Legend:	
xxx = X coordinate	I = vector x component
yyy = Y coordinate	J = vector y component
zzz = Z coordinate	K = vector z component

Tutor Translator

The **refType** field is:

- **2D** if the L2D Tutor keyword is specified.
- **3D** if the L2D Tutor keyword is not specified.



The commands that use FORM ISO algorithm actually don't have equivalent commands on PC-DMIS. See the chapter on "[Construction Commands with no PC-DMIS Equivalent](#)" for more information.

ILINE (MEMORY[36], 2, MEMORY[101]) translates to →

```
LIN1 =FEAT/LINE,RECT,UNBND  
THEO/152.003,236.002,151.514,-0.0000167,-0.9837928,-0.1793088  
ACTL/152.003,236.002,150.003,-0.0000254,-0.9964263,-0.0844669  
CONSTR/LINE,BF,3D,E.MEMORY[101].ID,E.MEMORY[102].ID,,  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT( "LIN1" )  
ASSIGN/E.MEMORY[36] = E.WM1  
  
...  
... Evaluation commands.  
...
```

ILINE (MEMORY [37], MEMORY[103],MEMORY[104]) translates to →

```
LIN2 =FEAT/LINE,RECT,UNBND  
THEO/152.002,96.001,120.001,-0.0000062,0.999924,-0.0123324  
ACTL/152.004,96.001,130.001,-0.0000062,0.999924,-0.0123324  
CONSTR/LINE,BF,3D,E.MEMORY[103].ID,E.MEMORY[106].ID,,  
  
ASSIGN/E.WM1 = TUTORELEMENT( "LIN2" )  
ASSIGN/E.MEMORY[37] = E.WM1  
  
...  
... Evaluation commands.  
...
```

ILINE (MEMORY [38], 2, MEMORY [101], L2D) translates to →

```
LIN3 =FEAT/LINE,RECT,UNBND  
THEO/152.003,236.002,151.514,0,-1,0  
ACTL/152.003,236.002,151.514,0,-1,0  
CONSTR/LINE,BF,2D,E.MEMORY[101].ID,E.MEMORY[102].ID,,  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT( "LIN3" )  
ASSIGN/E.MEMORY[38] = E.WM1  
  
...  
... Evaluation commands.  
...
```

ILINE (MEMORY[39],MEMORY[103],MEMORY[106],L2D) translates to →

```
LIN4 =FEAT/LINE,RECT,UNBND
THEO/152.004,-96.0004,131.515,0,0.999924,-0.0123318
ACTL/152.004,-96.0004,131.515,0,0.999924,-0.0123318
CONSTR/LINE,BF,2D,E.MEMORY[103].ID,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "LIN4" )
ASSIGN/E.MEMORY[ 39 ] = E.WM1
...
...
... Evaluation commands .
...
```

The following commands that use ISO algorithm don't have an equivalent PC-DMIS command.
See the chapter on "[Construction Commands with no PC-DMIS Equivalent](#)" for more information.

ILINE (MEMORY [22], 2, MEMORY[101] ,FORM ISO)

ILINE (MEMORY [23], MEMORY[102],MEMORY[105], FORM ISO)

ILINE (MEMORY [24], 2, MEMORY[101], L2D ,FORM ISO)

ILINE (MEMORY [24], MEMORY[102],MEMORY[105], L2D, FORM ISO)

Middle Point Construction Commands

The command is translated in the corresponding PD-DMIS command

IMIDPT (MEMORY[40], 2, MEMORY[101]) translates to →

```
PNT58 = FEAT/POINT,RECT
THEO/152.0015,177.001,135.0005,0.0099995,0,0.99995
ACTL/152.0015,177.001,135.0015,0.0099995,0,0.99995
CONSTR/POINT,MID,E.MEMORY[101].ID,E.MEMORY[102].ID

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT58" )
ASSIGN/E.MEMORY[ 40 ] = E.WM1
...
...
... Evaluation commands .
...
```

IMIDPT (MEMORY [41], MEMORY[103],MEMORY[106]) translates to →

```
PNT60 =FEAT/POINT,RECT
THEO/152.0035,177.001,119.002,0.0099995,0,0.99995
ACTL/152.0035,177.001,119.002,0.0099995,0,0.99995
CONSTR/POINT,MID,E.MEMORY[103].ID,E.MEMORY[106].ID
```

Tutor Translator

```
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT60" )
ASSIGN/E.MEMORY[41] = E.WM1
...
...
... Evaluation commands.
...
```

Paraboloid Construction Commands

PC-DMIS does not have **IPARAB** equivalent command. See the chapter on "[Construction Commands with no PC-DMIS Equivalent](#)" for more information.

Plane Construction Commands

IPL (MEMORY[42], 3, MEMORY[101])

IPL (MEMORY[42], 3, MEMORY[101], FORM LSM) translates to →

```
PLN11 =FEAT/PLANE,RECT
THEO/152.0023,150.001,130.0007,-0.9999998,0.000073,-0.0005604
ACTL/152.0023,150.001,130.0013,-0.9999998,0.000073,-0.0005604
CONSTR/PLANE,BF,E.MEMORY[101].ID,E.MEMORY[102].ID,E.MEMORY[103].ID,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PLN11" )
ASSIGN/E.MEMORY[42] = E.WM1
...
...
... Evaluation commands.
...
```

IPL (MEMORY [43], MEMORY[102],MEMORY[105],MEMORY[107])

IPL (MEMORY [43], MEMORY[102],MEMORY[105],MEMORY[107], FORM LSM) translates to →

```
PLN12 =FEAT/PLANE,RECT
THEO/104.6677,121.3347,78.6673,-0.7149675,0.1935771,0.6718255
ACTL/104.6677,121.3347,78.6673,-0.7149675,0.1935771,0.6718254
CONSTR/PLANE,BF,E.MEMORY[102].ID,E.MEMORY[105].ID,E.MEMORY[107].ID,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PLN12" )
ASSIGN/E.MEMORY[43] = E.WM1
...
...
... Evaluation commands.
...
```

The following commands that use ISO algorithm don't have an equivalent PC-DMIS command. See the chapter on "[Construction Commands with no PC-DMIS Equivalent](#)" for more information.

IPL (MEMORY [44], 3, MEMORY[101], FORM ISO)

IPL (MEMORY [43], MEMORY[102],MEMORY[105],MEMORY[107], FORM ISO)

Three Level Plane Construction Commands

If the three distances have been omitted the values stored with the PL3L_DISTANCES command emulation is used. The APPROACH vector is inverted.

PL3L_DISTANCE (-4.8887, 2.4481, 2.6532)

IPL3L(MEMORY[44],3, MEMORY[213], APPROACH 0, 0.70710678, 0.70710678) translates to →

```

ASSIGN/E.PL3L.D1 = -4.8887
ASSIGN/E.PL3L.D2 = 2.4481
ASSIGN/E.PL3L.D3 = 2.6532
ASSIGN/E.DIR = -UNIT (MPOINT (0,0.70710678,0.70710678))
PNT_1 =GENERIC/POINT, DEPENDENT, RECT,$
NOM/XYZ,E.MEMORY[213].ID.TX,E.MEMORY[213].ID.TY,E.MEMORY[213].ID.TZ,$
MEAS/XYZ,E.MEMORY[213].ID.X,E.MEMORY[213].ID.Y,E.MEMORY[213].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PNT_2 =GENERIC/POINT, DEPENDENT, RECT,$
NOM/XYZ,E.MEMORY[214].ID.TX,E.MEMORY[214].ID.TY,E.MEMORY[214].ID.TZ,$
MEAS/XYZ,E.MEMORY[214].ID.X,E.MEMORY[214].ID.Y,E.MEMORY[214].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PNT_3 =GENERIC/POINT, DEPENDENT, RECT,$
NOM/XYZ,E.MEMORY[215].ID.TX,E.MEMORY[215].ID.TY,E.MEMORY[215].ID.TZ,$
MEAS/XYZ,E.MEMORY[215].ID.X,E.MEMORY[215].ID.Y,E.MEMORY[215].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PLN16 =FEAT/PLANE, RECT
THEO/11.3151,9.6455,103.2726,0,-0.7071068,-0.7071068
ACTL/11.3151,9.6455,103.2726,-0.2899823,-0.3410351,-0.8942065
CONSTR/PLANE, OFFSET
ID =PNT_1, PNT_2, PNT_3,
OFFSET = E.PL3L.D1, E.PL3L.D2, E.PL3L.D3

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("PLN16")
ASSIGN/E.MEMORY [44] = E.WM1
...
... Evaluation commands .
...

```

Tutor Translator

**PL3L_DISTANCE (2.6532,-4.8887, 2.4481) IPL3L(MEMORY[45],MEMORY[215].ID,MEMORY[213].ID,
MEMORY[214].ID,APPROACH 0, 0.70710678, 0.70710678)**
translates to →

```
ASSIGN/E.PL3L.D1 = 2.6532
ASSIGN/E.PL3L.D2 = -4.8887
ASSIGN/E.PL3L.D3 = 2.4481
ASSIGN/E.DIR = -UNIT(MPOINT(0, 0.70710678, 0.70710678))
PNT_5 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,E.MEMORY[215].ID.TX,E.MEMORY[215].ID.TY,E.MEMORY[215].ID.TZ,$
MEAS/XYZ,E.MEMORY[215].ID.X,E.MEMORY[215].ID.Y,E.MEMORY[215].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PNT_3 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,E.MEMORY[213].ID.TX,E.MEMORY[213].ID.TY,E.MEMORY[213].ID.TZ,$
MEAS/XYZ,E.MEMORY[213].ID.X,E.MEMORY[213].ID.Y,E.MEMORY[213].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PNT_4 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,E.MEMORY[214].ID.TX,E.MEMORY[214].ID.TY,E.MEMORY[214].ID.TZ,$
MEAS/XYZ,E.MEMORY[214].ID.X,E.MEMORY[214].ID.Y,E.MEMORY[214].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PLN14 =FEAT/PLANE,RECT
THEO/8.002,5.002,68.3347,-0.2899823,-0.3410351,-0.8942065
ACTL/8.002,5.002,68.3347,0.8706348,0.4650446,-0.1604013
CONSTR/PLANE,OFFSET
ID =PNT_5,PNT_3,PNT_4,
OFFSET = 2.6532,-4.8887,2.4481

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT("PLN14")
ASSIGN/E.MEMORY[45] = E.WM1
...
... Evaluation commands.
...
```

**IPL3L (MEMORY [46], 3, MEMORY[213],-4.8887, 2.4481, 2.6532, APPROACH 0, 0.70710678,
0.70710678) translates to →**

```
ASSIGN/E.DIR = -UNIT (MPOINT (0,0.70710678,0.70710678))
PNT_9 =GENERIC/POINT, DEPENDENT, RECT,$
NOM/XYZ,E.MEMORY[213].ID.TX,E.MEMORY[213].ID.TY,E.MEMORY[213].ID.TZ,$
MEAS/XYZ,E.MEMORY[213].ID.X,E.MEMORY[213].ID.Y,E.MEMORY[213].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PNT_10 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,E.MEMORY[214].ID.TX,E.MEMORY[214].ID.TY,E.MEMORY[214].ID.TZ,$
MEAS/XYZ,E.MEMORY[214].ID.X,E.MEMORY[214].ID.Y,E.MEMORY[214].ID.Z,$
```

```

NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PNT_11 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,E.MEMORY[215].ID.TX,E.MEMORY[215].ID.TY,E.MEMORY[215].ID.TZ,$
MEAS/XYZ,E.MEMORY[215].ID.X,E.MEMORY[215].ID.Y,E.MEMORY[215].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PLN17 =FEAT/PLANE,RECT
THEO/11.3151,9.6455,103.2726,-0.2899823,-0.3410351,-0.8942065
ACTL/11.3151,9.6455,103.2726,-0.2899823,-0.3410351,-0.8942065
CONSTR/PLANE,OFFSET
ID =PNT_9,PNT_10,PNT_11,,,
OFFSET = -4.8887,2.4481,2.6532
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PLN17" )
ASSIGN/E.MEMORY[46] = E.WM1
...
...
Evaluation commands.
...

```

IPL3L(MEMORY[47],MEMORY[215].ID,MEMORY[213].ID,MEMORY[214].ID),2.6532,-4.8887,2.4481,APPROACH 0, 0.70710678, 0.70710678) translates to →

```

ASSIGN/E.DIR = -UNIT(MPOINT(0, 0.70710678, 0.70710678))
PNT_12 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,E.MEMORY[215].ID.TX,E.MEMORY[215].ID.TY,E.MEMORY[215].ID.TZ,$
MEAS/XYZ,E.MEMORY[215].ID.X,E.MEMORY[215].ID.Y,E.MEMORY[215].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PNT_13 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,E.MEMORY[213].ID.TX,E.MEMORY[213].ID.TY,E.MEMORY[213].ID.TZ,$
MEAS/XYZ,E.MEMORY[213].ID.X,E.MEMORY[213].ID.Y,E.MEMORY[213].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PNT_14 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,E.MEMORY[214].ID.TX,E.MEMORY[214].ID.TY,E.MEMORY[214].ID.TZ,$
MEAS/XYZ,E.MEMORY[214].ID.X,E.MEMORY[214].ID.Y,E.MEMORY[214].ID.Z,$
NOM/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z,$
MEAS/IJK,E.DIR.X,E.DIR.Y,E.DIR.Z
PLN18 =FEAT/PLANE,RECT
THEO/8.002,5.002,68.3347,-0.2899823,-0.3410351,-0.8942065
ACTL/8.002,5.002,68.3347,0.8706348,0.4650446,-0.1604013
CONSTR/PLANE,OFFSET
ID =PNT_12,PNT_13,PNT_14,,,
OFFSET = 2.6532,-4.8887,2.4481
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PLN18" )
ASSIGN/E.MEMORY[47] = E.WM1

```

Tutor Translator

```
...
...      Evaluation commands .
...
...
```

Slot Construction Commands

In PC-DMIS there is a command to construct a slot. Nevertheless the rules by which existing features can be used are completely different. Therefore it is not possible to use the same features from the Tutor part program.

It is not possible to translate the ISLOT command. As a partial solution the ISLOT command is translated to a constructed slot which construction features are undefined and theoretical values are forced to 0 unless theoretical values are available.(X, Y, Z, CX, CY, CZ, DM and DM2).

A Comment/Oper is added in order to notify the operator about missing compatibility with original Part Program.

Round Slot

ISLOT (MEMORY[10], 10, MEMORY[101]) ISLOT (MEMORY[10], 6, MEMORY[105])

**ISLOT (MEMORY[10], 6, MEMORY[101], MEMORY[107], MEMORY[119], MEMORY[111],
MEMORY[112],**

MEMORY[105] translates to →

```
...
...      Warning Messages
...
...      SLT1    =FEAT/SLOT,RECT,IN
THEO/0,0,0,0,0,0,0,0
ACTL/0,0,0,0,0,0,0,0
CONSTR/SLOT,ROUND SLOT,,
ASSIGN/E.WM2 = E.WM2
ASSIGN/E.WM1 = TUTORELEMENT( "SLT1" )
ASSIGN/E.MEMORY[10] = E.WM1
...
...      Evaluation commands .
...
...
```

Square Slot

It is not possible to translate the ISLOT command. There is no PC-DMIS command for Square Slot construction.

Sphere Construction Commands

The following commands that use ISO algorithm don't have an equivalent PC-DMIS command. See the chapter on "[Construction Commands with no PC-DMIS Equivalent](#)" for more information.

ISPH (MEMORY[10],5, MEMORY[101])

ISPH (MEMORY[10],5, MEMORY[101],FORM LSM)

ISPH (MEMORY[10],5, MEMORY[101],FORM ISO) translates to →

```
...
...           Warning Messages for FORM ISO
...
SPH1    =FEAT/SPHERE,RECT,OUT
THEO/10,20,30,0,0,1,22.9999
ACTL/10,20,30,0,0,1,22.9999
CONSTR/SPHERE,BF,E.MEMORY[101].ID,E.MEMORY[102].ID,E.MEMORY[103].ID,
E.MEMORY[104].ID,E.MEMORY[105].ID,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "SPH1" )
ASSIGN/E.MEMORY[10] = E.WM1

...
...           Evaluation commands.
...
```

ISPH (MEMORY[11], MEMORY[103], MEMORY[101], MEMORY[105], MEMORY[102], MEMORY[104])

**ISPH (MEMORY[11], MEMORY[103], MEMORY[101], MEMORY[105], MEMORY[102], MEMORY[104],
FORM LSM)**

**ISPH (MEMORY[11], MEMORY[103], MEMORY[101], MEMORY[105], MEMORY[102], MEMORY[104]
,FORM ISO) translates to →**

```
...
...           Warning Messages for FORM ISO
...
SPH2    =FEAT/SPHERE,RECT,OUT
THEO/10,20,30,0,0,1,22.9999
ACTL/10,20,30,0,0,1,22.9999
CONSTR/SPHERE,BF,E.MEMORY[103].ID,E.MEMORY[101].ID,E.MEMORY[105].ID,
E.MEMORY[102].ID,E.MEMORY[104].ID,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "SPH2" )
ASSIGN/E.MEMORY[11] = E.WM1

...
...           Evaluation commands.
...
```

Tutor Translator

Thickness Construction Commands

ITHICHNS(MEMORY[10],5, MEMORY[103]) translates to →

```
PLN3    =FEAT/PLANE,RECT
THEO/0.2199,350.0295,0.2001,1,0,0.0000002
ACTL/0.2199,350.0295,0.2001,1,0,0.0000002
CONSTR/PLANE,BF,E.MEMORY[103].ID, E.MEMORY[104].ID, E.MEMORY[105].ID,
E.MEMORY[106].ID,,
PNT7    =FEAT/POINT,RECT
THEO/0.2199,340,0,1,0,0.0000002
ACTL/0.2199,340,0,1,0,0.0000002
CONSTR/POINT,PROJ,PLN3,E.MEMORY[107].ID

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT7" )
ASSIGN/E.WM1.DS = DIST3D( {PNT7} ,{E.MEMORY[107].ID} )
ASSIGN/E.MEMORY[10] = E.WM1
...
...           Evaluation commands.
...
```

**ITHICHNS(MEMORY[10], MEMORY[207], MEMORY[200] , MEMORY[203] , MEMORY[210] ,
MEMORY[230])** translates to →

```
PLN4    =FEAT/PLANE,RECT
THEO/0.2199,350.0295,0.2001,1,0,0.0000002
ACTL/0.2199,350.0295,0.2001,1,0,0.0000002
CONSTR/PLANE,BF,E.MEMORY[207].ID, E.MEMORY[200].ID, E.MEMORY[203].ID,
E.MEMORY[210].ID,,
PNT8    =FEAT/POINT,RECT
THEO/0.2199,350.0295,0,0,0,1
ACTL/0.2199,340,0,1,0,0.0000002
CONSTR/POINT,PROJ,PLN4,E.MEMORY[230].ID

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT8" )
ASSIGN/E.WM1.DS = DIST3D( {PNT8} ,{E.MEMORY[230].ID} )
ASSIGN/E.MEMORY[10] = E.WM1
...
...           Evaluation commands.
...
```

Torus Construction Command

PC-DMIS does not have an equivalent **ITORUS** construction command. This command is not translated. A Comment/Oper is added in order to notify the operator about missing compatibility with original Part Program.

RIDPT Construction Command

This command reduces an element to a point element

RIDPT (MEMORY[10],MEMORY[100]) translates to →

```
PNT1 =FEAT/POINT,RECT
THEO/0.22,375.9418,-12.2665,0,0,1
ACTL/0.22,375.9418,-12.2665,0,0,1
CONSTR/POINT,CAST,E.MEMORY[100].ID

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT1" )
ASSIGN/E.MEMORY[10] = E.WM1
```

RIDPT (,MEMORY[101]) translates to →

```
PNT2 =FEAT/POINT,RECT
THEO/0.22,324.134,12.1668,0,0,1
ACTL/0.22,324.134,12.1668,0,0,1
CONSTR/POINT,CAST,E.MEMORY[101].ID

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT2" )
```

Relationship Between Elements

The Tutor relation command has the following formats:

RelType (ResMem[],Element1[]Element2[])

Listed below are the parameters for the above command:

- [] means optional field or element specific keyword.
- **RelType:** is the relation keyword:
 - PROJ = projection
 - INTER = intersection
 - MIDDLE = middle element
 - DIST_2D = distance in a plane
 - DIST_3D = distance in the space
- **ResMem:** is the destination memory. If omitted means E.WM1. If the relation generates a second element it is stored in E.WM2
- **Element1:** First element
- **Element2:** Second element

Projection Commands

Tutor allows only three kinds of projection commands:

1. A Point type feature (Point, Circle, Ellipse) projected onto a Line type feature (Line, Cone, Cylinder). The result is a Point by Rel feature.
2. A Point type feature (Point, Circle, Ellipse) projected onto a Plane. The result is a Point by Rel feature.
3. A Line type feature (Line, Cone, Cylinder) projected onto a Plane . The result is a Line by Rel feature.

Tutor Projection commands do not have the resulting feature types. During translation the type of the features involved in the computation may be unknown.

A PC-DMIS projection command is a feature construction command with a specific option. This means that at translation time the resulting feature type must be known. A Basic script Function checks the PC-DMIS features type and builds the appropriate Projection command. If PC-DMIS features type cannot be retrieved the operator is asked for the correct one. The **MakeProjCmd** function is inserted into the Basic script and called every time a projection is translated.

PROJ(MEMORY[40],MEMORY[100],MEMORY[102]) translates to →

If the resulting feature is a point:

```
PROJ1 =FEAT/POINT,RECT  
THEO/-87.7451,287.9767,-100.2316,0.5773503,0.5773503,0.5773503  
ACTL/-87.7451,287.9767,-100.2316,0.5773503,0.5773503,0.5773503  
CONSTR/POINT,PROJ,E.MEMORY[100].ID,E.MEMORY[102].ID  
  
ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK  
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT( "PRJ1_PNT1" )  
ASSIGN/E.WM1.TYPE = 116  
ASSIGN/E.WM1.DS = DIST3D( {E.MEMORY[100].ID},{E.MEMORY[102].ID} )  
ASSIGN/E.WM1.A = ANGLEBETWEEN( INT_V1,INT_V2 )  
ASSIGN/E.WM1.AXY = CALC_AXY( INT_V1,INT_V2 )  
ASSIGN/E.WM1.AYZ = CALC_AYZ( INT_V1,INT_V2 )  
ASSIGN/E.WM1.AZX = CALC_AZX( INT_V1,INT_V2 )  
ASSIGN/E.MEMORY[40] = E.WM1
```

If the resulting feature is a line:

```
PROJ1 =FEAT/LINE,RECT,UNBND  
THEO/106.2041,-1.7959,-4.4082,-0.4082483,-0.4082483,0.8164966  
ACTL/106.2041,-1.7959,-4.4082,-0.4082483,-0.4082483,0.8164966  
CONSTR/LINE,PROJ,E.MEMORY[100].ID,E.MEMORY[102].ID,1  
  
ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK  
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK  
ASSIGN/E.WM2 = E.WM1
```

```

ASSIGN/E.WM1 = TUTORELEMENT( "PRJ1_LIN1" )
ASSIGN/E.WM1.TYPE = 118
ASSIGN/E.WM1.DS = DIST3D( {E.MEMORY[100].ID}, {E.MEMORY[102].ID} )
ASSIGN/E.WM1.A = ANGLEBETWEEN( INT_V1, INT_V2 )
ASSIGN/E.WM1.AXY = CALC_AXY( INT_V1, INT_V2 )
ASSIGN/E.WM1.AYZ = CALC_AYZ( INT_V1, INT_V2 )
ASSIGN/E.WM1.AZX = CALC_AZX( INT_V1, INT_V2 )
ASSIGN/E.MEMORY[40] = E.WM1

```

PROJ(WM1,WM2) translates to →

If the resulting feature is a point:

```

PROJ1 =FEAT/POINT,RECT
THEO/-87.7451,287.9767,-100.2316,0.5773503,0.5773503,0.5773503
ACTL/-87.7451,287.9767,-100.2316,0.5773503,0.5773503,0.5773503
CONSTR/POINT,PROJ,E.WM1.ID,E.WM2.ID

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PRJ1_PNT1" )
ASSIGN/E.WM1.TYPE = 116
ASSIGN/E.WM1.DS = DIST3D( {E.MEMORY[100].ID}, {E.MEMORY[102].ID} )
ASSIGN/E.WM1.A = ANGLEBETWEEN( INT_V1, INT_V2 )
ASSIGN/E.WM1.AXY = CALC_AXY( INT_V1, INT_V2 )
ASSIGN/E.WM1.AYZ = CALC_AYZ( INT_V1, INT_V2 )
ASSIGN/E.WM1.AZX = CALC_AZX( INT_V1, INT_V2 )

```

If the resulting feature is a line:

```

PROJ1 =FEAT/LINE,RECT,UNBND
THEO/106.2041,-1.7959,-4.4082,-0.4082483,-0.4082483,0.8164966
ACTL/106.2041,-1.7959,-4.4082,-0.4082483,-0.4082483,0.8164966
CONSTR/LINE,PROJ,E.WM1.ID,E.WM2.ID,1

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PRJ1_LIN1" )
ASSIGN/E.WM1.TYPE = 118
ASSIGN/E.WM1.DS = DIST3D( {E.MEMORY[100].ID}, {E.MEMORY[102].ID} )
ASSIGN/E.WM1.A = ANGLEBETWEEN( INT_V1, INT_V2 )
ASSIGN/E.WM1.AXY = CALC_AXY( INT_V1, INT_V2 )
ASSIGN/E.WM1.AYZ = CALC_AYZ( INT_V1, INT_V2 )
ASSIGN/E.WM1.AZX = CALC_AZX( INT_V1, INT_V2 )

```

Intersection Commands

Tutor allows the following intersection commands:

- Circle-Circle. The results are 2 Point by Rel features (E.WM1 and E.WM2).
- Line-Line. The result is a Point by Rel feature.
- Plane-Plane. The result is a Line by Rel feature.
- Sphere-Sphere. The result is Circle by Rel feature.
- Sphere-Plane. The result is Circle by Rel feature.
- Circle-Line. The results are 2 Point by Rel features (E.WM1 and E.WM2).
- Plane-Line. The result is a Point by Rel feature.
- Cone-Plane. The result is Circle by Rel feature.
- Cylinder-Plane. The result is Point by Rel feature.
- Cylinder-Cylinder. The result is Point by Rel feature.
- Sphere-Line. The results are 2 Point by Rel features (E.WM1 and E.WM2).

Tutor Intersection commands do not have the resulting feature types. During translation the type of the features involved in the computation may be unknown.

A PC-DMIS Intersection command is a feature construction command with a specific option. This means that at translation time the resulting feature type must be known. A Basic script Function checks the PC-DMIS features type and build the appropriate Intersection command. If PC-DMIS features type cannot be retrieved the operator is asked for the correct one. The **MakelIntersCmd** function is inserted in the Basic script and called every time an intersection command is translated.



If the two elements don't intersect the intersection results are not computed and PC-DMIS displays an error message.

INTER(MEMORY[40],MEMORY[100],MEMORY[102]) translates to →

If the elements are two Circles:

```
INT1_PLN1 =FEAT/PLANE,RECT  
THEO/10,10,10,0,0,1  
ACTL/10,10,10,0,0,1  
CONSTR/PLANE,CAST,E.MEMORY[100].ID  
    INT1_PLN2 =FEAT/PLANE,RECT  
THEO/10,20,30,0,0,1  
ACTL/10,20,30,0,0,1  
CONSTR/PLANE,CAST,E.MEMORY[102].ID  
    INT1_PLN3 =FEAT/PLANE,RECT  
THEO/10,15,20,0,0,1  
ACTL/10,15,20,0,0,1  
CONSTR/PLANE,MID,INT1_PLN1,INT1_PLN2  
    INT1_CIR1 =FEAT/CIRCLE,RECT,OUT  
THEO/10,10,20,0,0,1,12  
ACTL/10,10,20,0,0,1,12  
CONSTR/CIRCLE,PROJ,E.MEMORY[100].ID,INT1_PLN3
```

```

INT1_CIR2 = FEAT/CIRCLE,RECT,OUT
THEO/10,20,20,0,0,1,12
ACTL/10,20,20,0,0,1,1
CONSTR/CIRCLE,PROJ,E.MEMORY[102].ID,INT1_PLN3
INT1_PNT1 = FEAT/POINT,RECT
THEO/6.6834,15,20,0,0,1
ACTL/6.6834,15,20,0,0,1
CONSTR/POINT,INT,INT1_CIR2,INT1_CIR1
INT1_PNT2 = FEAT/POINT,RECT
THEO/13.3166,15,20,0,0,1
ACTL/13.3166,15,20,0,0,1
CONSTR/POINT,INT,INT1_CIR1,INT1_CIR2

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_PNT1")
ASSIGN/E.WM1.TYPE = 116
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID},{E.MEMORY[102].ID})
ASSIGN/E.WM1.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM1.AXY = CALC_AXY(INT_V1,INT_V2)
ASSIGN/E.WM1.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM1.AZX = CALC_AZX(INT_V1,INT_V2)
ASSIGN/E.WM2 = TUTORELEMENT ("INT1_PNT2")
ASSIGN/E.WM2.TYPE = 116
ASSIGN/E.WM2.DS = DIST3D({E.MEMORY[100].ID},{E.MEMORY[102].ID})
ASSIGN/E.WM2.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM2.AXY = CALC_AXY(INT_V1,INT_V2)
ASSIGN/E.WM2.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM2.AZX = CALC_AZX(INT_V1,INT_V2)
ASSIGN/E.MEMORY[40] = E.WM1

```

If the elements are two Lines:

```

INT1_PNT1 = FEAT/POINT,RECT
THEO/7.5,12.5,5,0,0,1
ACTL/7.5,12.5,5,0,0,1
CONSTR/POINT,INT,E.MEMORY[100].ID,E.MEMORY[102].ID

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_PNT1")
ASSIGN/E.WM1.TYPE = 116
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID},{E.MEMORY[102].ID})
ASSIGN/E.WM1.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM1.AXY = CALC_AXY(INT_V1,INT_V2)
ASSIGN/E.WM1.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM1.AZX = CALC_AZX(INT_V1,INT_V2)
ASSIGN/E.MEMORY[40] = E.WM1

```

Tutor Translator

If the elements are two Planes:

```
INT1_LIN1 =FEAT/LINE,RECT,UNBND
THEO/-310,290.5,119.5,0,-0.7071068,0.7071068
ACTL/-310,290.5,119.5,0,-0.7071068,0.7071068
CONSTR/LINE,INTOF,E.MEMORY[100].ID,E.MEMORY[102].ID,1

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_LIN1")
ASSIGN/E.WM1.TYPE = 117
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID},{E.MEMORY[102].ID})
ASSIGN/E.WM1.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM1.AXY = CALC_AXY(INT_V1,INT_V2)
ASSIGN/E.WM1.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM1.AZX = CALC_AZX(INT_V1,INT_V2)
ASSIGN/E.MEMORY[40] = E.WM1
```

If the elements are two Spheres:

```
INT1_LIN1 =FEAT/LINE,RECT,UNBND
THEO/20,20,20,0.5773503,0.5773503,0.5773503
ACTL/20,20,20,0.5773503,0.5773503,0.5773503
CONSTR/LINE,BF,3D,E.MEMORY[100].ID,E.MEMORY[102].ID,,
INT1_A1 =ALIGNMENT/START, RECALL:E.ALIGNMENTS[E.CUR_ALIGN], LIST= YES
ALIGNMENT/LEVEL,XPLUS,INT1_LIN1
ALIGNMENT/END
WORKPLANE/ZPLUS
INT1_CIR1 =FEAT/CIRCLE,RECT,OUT
THEO/23.094,8.165,-14.1421,0.5773503,-0.4082483,0.7071068,20
ACTL/34.641,0,0,0,0,1,20
CONSTR/CIRCLE,PROJ,E.MEMORY[100].ID,
INT1_CIR2 =FEAT/CIRCLE,RECT,OUT
THEO/34.641,12.2474,-21.2132,0.5773503,-0.4082483,0.7071068,20
ACTL/51.9615,0,0,0,0,1,20
CONSTR/CIRCLE,PROJ,E.MEMORY[102].ID,
WORKPLANE/YPLUS
INT1_CIR3 =FEAT/CIRCLE,RECT,OUT
THEO/23.094,-16.3299,0,0.5773503,0.8164966,0,20
ACTL/34.641,0,0,0,1,0,20
CONSTR/CIRCLE,PROJ,E.MEMORY[100].ID,
INT1_CIR4 =FEAT/CIRCLE,RECT,OUT
THEO/34.641,-24.4949,0,0.5773503,0.8164966,0,20
ACTL/51.9615,0,0,0,1,0,20
CONSTR/CIRCLE,PROJ,E.MEMORY[102].ID,
INT1_PNT1 =FEAT/POINT,RECT
THEO/28.8675,4.0825,-21.2132,0.5773503,-0.4082483,0.7071068
ACTL/43.3013,-5,0,0,0,0,1
CONSTR/POINT,INT,INT1_CIR1,INT1_CIR2
```

```

INT1_PNT2 = FEAT/POINT,RECT
THEO/28.8675,16.3299,-14.1421,0.5773503,-0.4082483,0.7071068
ACTL/43.3013,5,0,0,0,1
CONSTR/POINT,INT,INT1_CIR2,INT1_CIR1
INT1_PNT3 = FEAT/POINT,RECT
THEO/28.8675,-20.4124,7.0711,0.5773503,0.8164966,0
ACTL/43.3013,0,5,0,1,0
CONSTR/POINT,INT,INT1_CIR3,INT1_CIR4
INT1_PNT4 = FEAT/POINT,RECT
THEO/28.8675,-20.4124,-7.0711,0.5773503,0.8164966,0
ACTL/43.3013,0,-5,0,1,0
CONSTR/POINT,INT,INT1_CIR4,INT1_CIR3
WORKPLANE/XPLUS
INT1_CIR5 = FEAT/CIRCLE,RECT,OUT,LEAST_SQR
THEO/28.8675,0,0,1,0,0,43.2049
ACTL/28.8675,0,0,1,0,0,10
CONSTR/CIRCLE,BF,INT1_PNT1,INT1_PNT3,INT1_PNT4,INT1_PNT2,,,
RECALL/ALIGNMENT,INTERNAL,E.ALIGNMENTS[E.CUR_ALIGN]
WORKPLANE/E.SELPL

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_CIR5")
ASSIGN/E.WM1.TYPE = 117
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID},{E.MEMORY[102].ID})
ASSIGN/E.WM1.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM1.AXY = CALC_AXY(INT_V1,INT_V2)
ASSIGN/E.WM1.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM1.AZX = CALC_AZX(INT_V1,INT_V2)
ASSIGN/E.MEMORY[40] = E.WM1

```

If the elements are a Sphere and a Plane:

```

INT1_CIR1 = FEAT/CIRCLE,RECT,OUT
THEO/33.3333,33.3333,33.3333,0.5773503,0.5773503,0.5773503,16.3299
ACTL/33.3333,33.3333,33.3333,0.5773503,0.5773503,0.5773503,16.3299
CONSTR/CIRCLE,INTOF,E.MEMORY[100].ID,E.MEMORY[102].ID

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_CIR1")
ASSIGN/E.WM1.TYPE = 117
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID},{E.MEMORY[102].ID})
ASSIGN/E.WM1.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM1.AXY = CALC_AXY(INT_V1,INT_V2)
ASSIGN/E.WM1.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM1.AZX = CALC_AZX(INT_V1,INT_V2)
ASSIGN/E.MEMORY[40] = E.WM1

```

Tutor Translator

If the elements are a Circle and a Line or a Sphere and a Line:

```
INT1_PNT1 = FEAT/POINT,RECT
THEO/5.7574,5.7574,10,0.7071068,0.7071068,0
ACTL/5.7574,5.7574,10,0.7071068,0.7071068,0
CONSTR/POINT,PIERCE,E.MEMORY[100].ID,E.MEMORY[102].ID

ASSIGN/E.WM2 = TUTORELEMENT ("INT1_PNT1")
ASSIGN/E.WM2.TYPE = 116
ASSIGN/E.WM2.DS = DIST3D({E.MEMORY[126].ID},{E.MEMORY[120].ID})
ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM2.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM2.AXY = CALC_AXY(INT_V1,INT_V2)
ASSIGN/E.WM2.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM2.AZX = CALC_AZX(INT_V1,INT_V2)

INT1_LIN1 = FEAT/LINE,RECT,UNBND
THEO/20,20,10,-0.7071068,-0.7071068,0
ACTL/15,15,10,-0.7071068,-0.7071068,0
CONSTR/LINE,REV,E.MEMORY[100].ID,14.1421
INT1_PNT2 = FEAT/POINT,RECT
THEO/14.2426,14.2426,10,-0.7071068,-0.7071068,0
ACTL/14.2426,14.2426,10,-0.7071068,-0.7071068,0
CONSTR/POINT,PIERCE,INT1_LIN1,E.MEMORY[107].ID

ASSIGN/E.WM1 = TUTORELEMENT ("INT1_PNT2")
ASSIGN/E.WM1.TYPE = 116
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID},{E.MEMORY[102].ID})
ASSIGN/E.WM1.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM1.AXY = CALC_AXY(INT_V1,INT_V2)
ASSIGN/E.WM1.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM1.AZX = CALC_AZX(INT_V1,INT_V2)
ASSIGN/E.MEMORY[40] = E.WM1
```

If the elements are a Plane and a Line:

```
INT1_PNT1 = FEAT/POINT,RECT
THEO/400,400,10,0.7071068,0.7071068,0
ACTL/400,400,10,0.7071068,0.7071068,0
CONSTR/POINT,PIERCE,E.MEMORY[100].ID,E.MEMORY[102].ID

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_PNT1")
ASSIGN/E.WM1.TYPE = 116
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID},{E.MEMORY[102].ID})
ASSIGN/E.WM1.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM1.AXY = CALC_AXY(INT_V1,INT_V2)
```

```

ASSIGN/E.WM1.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM1.AZX = CALC_AZX(INT_V1,INT_V2)
ASSIGN/E.MEMORY[40] = E.WM1

```

If the elements are a Cone and a Plane becomes:

```

INT1_CIR1 = FEAT/CIRCLE,RECT,OUT
THEO/110,2,408,0,0,1,0
ACTL/110,2,408,0,0,1,0
CONSTR/CIRCLE,INTOF,E.MEMORY[100].ID,E.MEMORY[102].ID

```

```

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_CIR1")
ASSIGN/E.WM1.TYPE = 117
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID},{E.MEMORY[102].ID})
ASSIGN/E.WM1.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM1.AXY = CALC_AX(Y(INT_V1,INT_V2)
ASSIGN/E.WM1.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM1.AZX = CALC_AZX(INT_V1,INT_V2)
ASSIGN/E.MEMORY[40] = E.WM1

```

If the elements are a Cylinder and a Plane or Two Cylinders:

```

INT1_PNT1 = FEAT/POINT,RECT
THEO/10.8,21.8,12.98,0,0,1
ACTL/10.8,21.8,12.98,0,0,1
CONSTR/POINT,INT,E.MEMORY[100].ID,E.MEMORY[102].ID

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_PNT1")
ASSIGN/E.WM1.TYPE = 116
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID},{E.MEMORY[102].ID})
ASSIGN/E.WM1.A = ANGLEBETWEEN(INT_V1,INT_V2)
ASSIGN/E.WM1.AXY = CALC_AX(Y(INT_V1,INT_V2)
ASSIGN/E.WM1.AYZ = CALC_AYZ(INT_V1,INT_V2)
ASSIGN/E.WM1.AZX = CALC_AZX(INT_V1,INT_V2)
ASSIGN/E.MEMORY[40] = E.WM1

```

Commands with line-line intersection

INTER(MEMORY[40],MEMORY[100],MEMORY[102],"X")
 translates to →

```

INT1_PNT1 = FEAT/POINT,RECT
THEO/7.5,12.5,5,0,0,1
ACTL/7.5,12.5,5,0,0,1

```

Tutor Translator

```
CONSTR/POINT, INT, E.MEMORY[100].ID, E.MEMORY[101].ID

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_PNT1")
ASSIGN/E.WM1.TYPE = 116
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID}, {E.MEMORY[102].ID})
ASSIGN/E.WM1.AXY = CALC_AX(Y( INT_V1, INT_V2)
ASSIGN/E.WM1.AYZ = CALC_AZ(Y( INT_V1, INT_V2)
ASSIGN/E.WM1.AZX = CALC_AX(Z( INT_V1, INT_V2)
ASSIGN/INT_V1.I = 0
ASSIGN/INT_V2.I = 0
ASSIGN/E.WM1.A = ANGLEBETWEEN(UNIT( INT_V1), UNIT( INT_V2))
ASSIGN/E.MEMORY[40] = E.WM1
```

INTER(MEMORY[40],MEMORY[100],MEMORY[102],"Y")
translates to →

```
INT1_PNT1 = FEAT/POINT, RECT
THEO/7.5,12.5,5,0,0,1
ACTL/7.5,12.5,5,0,0,1
CONSTR/POINT, INT, E.MEMORY[100].ID, E.MEMORY[101].ID

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_PNT1")
ASSIGN/E.WM1.TYPE = 116
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID}, {E.MEMORY[102].ID})
ASSIGN/E.WM1.AXY = CALC_AX(Y( INT_V1, INT_V2)
ASSIGN/E.WM1.AYZ = CALC_AZ(Y( INT_V1, INT_V2)
ASSIGN/E.WM1.AZX = CALC_AX(Z( INT_V1, INT_V2)
ASSIGN/INT_V1.J = 0
ASSIGN/INT_V2.J = 0
ASSIGN/E.WM1.A = ANGLEBETWEEN(UNIT( INT_V1), UNIT( INT_V2))
ASSIGN/E.MEMORY[40] = E.WM1
```

INTER(MEMORY[40],MEMORY[100],MEMORY[102],"Z")
translates to →

```
INT1_PNT1 = FEAT/POINT, RECT
THEO/7.5,12.5,5,0,0,1
ACTL/7.5,12.5,5,0,0,1
CONSTR/POINT, INT, E.MEMORY[100].ID, E.MEMORY[101].ID

ASSIGN/INT_V1 = E.MEMORY[100].ID.IJK
ASSIGN/INT_V2 = E.MEMORY[102].ID.IJK
ASSIGN/E.WM1 = TUTORELEMENT ("INT1_PNT1")
ASSIGN/E.WM1.TYPE = 116
ASSIGN/E.WM1.DS = DIST3D({E.MEMORY[100].ID}, {E.MEMORY[102].ID})
ASSIGN/E.WM1.AXY = CALC_AX(Y( INT_V1, INT_V2)
```

```

ASSIGN/E.WM1.AYZ = CALC_AYZ(INT_V1, INT_V2)
ASSIGN/E.WM1.AZX = CALC_AZX(INT_V1, INT_V2)
ASSIGN/INT_V1.K = 0
ASSIGN/INT_V2.K = 0
ASSIGN/E.WM1.A = ANGLEBETWEEN(UNIT(INT_V1), UNIT(INT_V2))
ASSIGN/E.MEMORY[40] = E.WM1

```

Middle Relation Commands

Tutor allows the following middle relation commands:

- Point-Point. The results is a Point by Rel feature
- Line-Line. The result is a Line by Rel feature.
- Plane-Plane. The result is a Plane by Rel feature.
- Point-Line. The result is Point by Rel feature.
- Point-Plane. The result is Point by Rel feature.
- Plane-Line. The result is a Line by Rel feature.

Where:

- Point means Class Point features : Point, Circle and Sphere.
- Line means Class Line feature: Line, Cylinder and Cone.

Tutor Middle commands do not have the resulting feature types. During translation the type of the features involved in the computation may be unknown.

A PC-DMIS middle command is a feature construction command with a specific option. This means that at translation time the resulting feature type must be known. A Basic script Function checks the PC-DMIS features type and builds the appropriate Middle command. If PC-DMIS features type cannot be retrieved the operator is asked for the correct one. The **MakeMiddleCmd** function is inserted in the Basic script and called every time a Middle command is translated.



If the two elements don't intersect the intersection results are not computed and PC-DMIS displays an error message.

MIDDLE(MEMORY[120],MEMORY[100],MEMORY[101])
translates to →

If the elements are two class Point features:

```

MID1_PNT1 = FEAT/POINT,RECT
THEO/10,10,20,0,0,1
ACTL/10,10,20,0,0,1
CONSTR/POINT,MID,E.MEMORY[100].ID,E.MEMORY[101].ID

ASSIGN/E.WM1 = TUTORELEMENT ("MID1_PNT1")
ASSIGN/E.WM1.TYPE = 116
ASSIGN/E.MEMORY[120] = E.WM1

```

Tutor Translator

If the elements are two class Line features:

```
MID1_LIN1 =FEAT/LINE,RECT,UNBND  
THEO/10,0,10,0,0,1  
ACTL/10,0,10,0,0,1  
CONSTR/LINE,MID,E.MEMORY[100].ID,E.MEMORY[101].ID,0  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT ("MID1_LIN1")  
ASSIGN/E.WM1.TYPE = 118  
ASSIGN/E.MEMORY[120] = E.WM1
```

If the elements are two Plane features:

```
MID1_PLN1 =FEAT/LINE,RECT,UNBND  
THEO/10,0,10,0,0,1  
ACTL/10,0,10,0,0,1  
CONSTR/LINE,MID,E.MEMORY[100].ID,E.MEMORY[101].ID,0  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = TUTORELEMENT ("MID1_PLN1")  
ASSIGN/E.WM1.TYPE = 119  
ASSIGN/E.MEMORY[120] = E.WM1
```

If the elements are class Point and class Line features:

```
MID1_PNT1 =FEAT/POINT,RECT  
THEO/10,10,20,0,0,1  
ACTL/10,10,20,0,0,1  
CONSTR/POINT,MID,E.MEMORY[100].ID,E.MEMORY[101].ID  
  
ASSIGN/E.WM1 = TUTORELEMENT ("MID1_PNT1")  
ASSIGN/E.WM1.TYPE = 116  
ASSIGN/E.MEMORY[120] = E.WM1
```

If the elements are class Point and class Plane features:

```
MID1_PNT1 =FEAT/POINT,RECT  
THEO/10,10,20,0,0,1  
ACTL/10,10,20,0,0,1  
CONSTR/POINT,MID,E.MEMORY[100].ID,E.MEMORY[101].ID  
  
ASSIGN/E.WM1 = TUTORELEMENT ("MID1_PNT1")  
ASSIGN/E.WM1.TYPE = 116  
ASSIGN/E.MEMORY[120] = E.WM1
```

If the elements are class Plane and class Line features:

```

MID1_LIN1 =FEAT/LINE,RECT,UNBND
THEO/10,10,10,0.7071068,0.7071068,0
ACTL/10,10,10,0.7071068,0.7071068,0
CONSTR/LINE,PROJ, E.MEMORY[100].ID, E.MEMORY[101].ID,28.2843
MID1_LIN2 =FEAT/LINE,RECT,UNBND
THEO/10,10,30,0,0,1
ACTL/10,10,30,0,0,1
CONSTR/LINE,MID, E.MEMORY[100].ID,MID8_LIN1,28.2843

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "MID8_LIN2" )

```

Distance Relation Commands

Tutor allows the following distance relation commands:

- [Point-Point 2D](#). The result is a Line by Rel feature
- [Point-Line 2D](#). The result is Line by Rel feature.
- [Point-Point 3D](#). The result is a Line by Rel feature
- [Point-Line 3D](#). The result is Line by Rel feature.
- [Line-Line3D](#). The result is a Line by Rel feature.
- [Point-Plane3D](#). The result is Point by Rel feature.

Where:

- Point means Class Point features : Point, Circle and Sphere.
- Line means Class Line feature: Line, Cylinder and Cone.

 For Line-Point 2D, Line-Point 3D, Plane-Point 3D the PC-DMIS point of the resulting line differs from Tutor one. The correct value could be computed subtracting the Start Point coordinates from the Relation Line End Point Coordinates. However the value of Polar angle and polar radius must be recalculated.)

Tutor dist_2d and dist_3d commands do not have the resulting feature type. During translation the type of the features involved in the computation may be unknown.

A PC-DMIS perpendicular line feature construction command with a specific option is used.. A Basic script Function checks the PC-DMIS features type and builds the appropriate perpendicular command. If PC-DMIS features type cannot be retrieved the operator is asked for the correct one. The **MakeDistanceCmd** function is inserted in the Basic script and called every time a Distance command is translated.

Tutor Translator

DIST_2D (MEMORY[120],MEMORY[100],MEMORY[101])
translates to →

If the elements are two class Point features:

```
DIST1_LIN1 =FEAT/LINE,RECT,UNBND
THEO/10,15,10,0,-1,0
ACTL/10,15,10,0,-1,0
CONSTR/LINE,BF,2D,E.MEMORY[100].ID,E.MEMORY[101].ID,,

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("DIST1_LIN1")
ASSIGN/E.WM1.DS = DIST1_LIN1.LENGTH
ASSIGN/E.WM1.TYPE = 118
ASSIGN/E.MEMORY[120] = E.WM1
```

If the elements are class Point and class Line features:

```
COMMENT/DOC
=====
, Warning: Rel line's application point may differ from tutor.
=====

DIST1_LIN1 =FEAT/LINE,RECT,UNBND
THEO/10,10,0,0.7071068,0.7071068,0
ACTL/10,10,0,0.7071068,0.7071068,0
CONSTR/LINE,PROJ,E.MEMORY[100].ID,,0

DIST1_PNT1 =FEAT/POINT,RECT
THEO/10,50,0,0,0,1
ACTL/10,50,0,0,0,1
CONSTR/POINT,PROJ,E.MEMORY[101].ID,
DIST1_LIN2 =FEAT/LINE,RECT,UNBND
THEO/30,30,0,-0.7071068,0.7071068,0
ACTL/30,30,0,-0.7071068,0.7071068,0
CONSTR/LINE,PRTO,DIST1_LIN1,DIST1_PNT1,28.2843
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("DIST1_LIN2")
ASSIGN/E.WM1.DS = DIST1_LIN2.LENGTH
ASSIGN/E.MEMORY[120] = E.WM1
```

DIST_3D (MEMORY[120],MEMORY[100],MEMORY[101])
translates to →

If the elements are two class Point features:

```
DIST1_LIN1 =FEAT/LINE,RECT,UNBND
THEO/36.6667,36.6667,36.6667,-0.8164966,0.4082483,0.4082483
ACTL/36.6667,36.6667,36.6667,-0.8164966,0.4082483,0.4082483
CONSTR/LINE,PRTO,E.MEMORY[100].ID,E.MEMORY[100].ID,32.6599

ASSIGN/E.WM2 = E.WM1
```

```

ASSIGN/E.WM1 = TUTORELEMENT ("DIST1_LIN1")
ASSIGN/E.WM1.DS = DIST1_LIN1.LENGTH
ASSIGN/E.MEMORY[120] = E.WM1

```

If the elements are class Point and class Line features or class Point and class Plane features:

```

COMMENT/DOC
=====
, Warning: Rel line's application point may differ from tutor.
=====
DIST1_LIN1 =FEAT/LINE,RECT,UNBND
THEO/36.6667,36.6667,36.6667,-0.8164966,0.4082483,0.4082483
ACTL/36.6667,36.6667,36.6667,-0.8164966,0.4082483,0.4082483
CONSTR/LINE,PRTO,E.MEMORY[100].ID,E.MEMORY[100].ID,32.6599

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("DIST1_LIN1")
ASSIGN/E.WM1.DS = DIST1_LIN1.LENGTH
ASSIGN/E.MEMORY[120] = E.WM1

```

If the elements are two class LINE features:

```

DIST1_PNT1 =FEAT/POINT,RECT
THEO/10,10,20,0.7071068,0.7071068,0
ACTL/10,10,20,0.7071068,0.7071068,0
CONSTR/POINT,INT,E.MEMORY[100].ID,E.MEMORY[101].ID
DIST1_LIN1 =FEAT/LINE,RECT,UNBND
THEO/10,10,10,0,0,1
ACTL/10,10,10,0,0,1
CONSTR/LINE,PRTO,E.MEMORY[100].ID,DIST1_PNT1,10

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT ("DIST1_LIN1")
ASSIGN/E.WM1.DS = DIST1_LIN1.LENGTH * 2
ASSIGN/E.MEMORY[120] = E.WM1

```

CIRCLE_BY_CONE Relation Command

This command performs the intersection of the specified cone with a plane perpendicular to its axis. The format for this command is:

CIRCLE_BY_CONE (resmem, elmem1, dim=value):

- **dim** represents two (alternative) dimensions
- **H** is the distance between the intersection plane and the vertex of the cone. The diameter (DM) of the intersection circle is calculated.
- **D** is the diameter of the intersection circle. The distance (H) between the intersection plane and the cone vertex is calculated
- **value** can be a real number, a real variable or a real expression.

Tutor Translator

The resulting element is a Pick.

CIRCLE_BY_CONE (MEMORY[120],MEMORY[100],H=3)
translates to →

```
ASSIGN/H = 3
ASSIGN/D = 2*(H*TAN(DEG2RAD(E.MEMORY[100].ID.ANGLE/2)))
ASSIGN/PCIR = E.MEMORY[100].ID.XYZ + E.MEMORY[100].ID.IJK*H
PNT2 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,PCIR.X,PCIR.Y,PCIR.Z,$
MEAS/XYZ,PCIR.X,PCIR.Y,PCIR.Z,$
NOM/IJK,E.MEMORY[100].ID.I,E.MEMORY[100].ID.J,E.MEMORY[100].ID.K,$
MEAS/IJK,E.MEMORY[100].ID.I,E.MEMORY[100].ID.J,E.MEMORY[100].ID.K

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT2" )
ASSIGN/E.WM1.TYPE = 115
ASSIGN/E.WM1.DM = D
ASSIGN/E,MEMORY[120] = E.WM1
```

CIRCLE_BY_CONE (MEMORY[120],MEMORY[100],D=1.75)
translates to →

```
ASSIGN/D = 1.75
ASSIGN/H = D/(2*TAN(DEG2RAD(CON2.ANGLE/2)))
ASSIGN/PCIR = E.MEMORY[100].ID.XYZ + E.MEMORY[100].ID.IJK*H
PNT3 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,PCIR.X,PCIR.Y,PCIR.Z,$
MEAS/XYZ,PCIR.X,PCIR.Y,PCIR.Z,$
NOM/IJK,E.MEMORY[100].ID.I,E.MEMORY[100].ID.J,E.MEMORY[100].ID.K,$
MEAS/IJK,E.MEMORY[100].ID.I,E.MEMORY[100].ID.J,E.MEMORY[100].ID.K

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT3" )
ASSIGN/E.WM1.TYPE = 115
ASSIGN/E.WM1.DM = D
ASSIGN/E.WM1.DS = H
ASSIGN/E,MEMORY[120] = E.WM1
```

Geometric Tolerances

Tutor computes three Geometric Tolerance categories:

1. [Orientation Tolerances](#) - [Parallelism](#), [Perpendicularity](#) and [Angularity](#)
2. [Concentricity](#), [Coaxiality](#) and [Symmetry](#) Tolerances
3. [True Position tolerances](#)



In cases where the EXTENDLENGTH value is used, ensure that the value is 2 times the Tutor dist value.

Orientation Tolerances

The Tutor Orientation Tolerances have the following command formats for [Parallelism](#) (PARAL), [Perpendicularity](#) (PERPEN) and [Angularity](#) (ANGOL):

- PARAL or PERPEN or ANGOL (elchk, eldat)
- PARAL or PERPEN or ANGOL (> eldest, elchk, eldat)
- PARAL or PERPEN or ANGOL DIST(elchk, eldat, dist)
- PARAL or PERPEN or ANGOL DIST(> eldest, elchk, eldat, dist)
- PARAL or PERPEN or ANGOL BOUND (elchk, eldat, boundaries)
- PARAL or PERPEN or ANGOL BOUND (> eldest, elchk, eldat, boundaries)

The arguments for geometric tolerance commands are as follows:

- **eldest** - Memory that will contain the result. If omitted, the result will be written in the memory of the CHECKED element.
- **elchk** - Memory in which the CHECKED element is located. It may be an element variable, the element_array variable, or a working storage variable (E.WM1 or E.WM2). If omitted, the E.WM1 is used.
- **eldat** - Memory in which the DATUM element is held. It can be an element variable, the element_array variable, or a working storage variable (E.WM1 or E.WM2). If omitted, the E.WM1 is used.
- **angle** - The theoretical angle between the two elements, in degrees and thousandths (for the angularity check only)
- **dist** - The distance from the checked element baricenter (+and -) which bind the length or area of the element.
- **boundaries** - Represents a sequence of two or four elements (lines or measured planes constructed or defined theoretically) which bind the length or area of the checked element.



PC-DMIS does not support Boundary elements. The commands that use boundaries and the following Tutor DOS commands are not translated: PARAL (eldat), PERPEN (eldat), & ANGOL (eldat, angle)

Tutor allows the following parallelism, perpendicularity, and Angularity commands:

- Checked Line – Datum Line.
- Checked Line – Datum Plane.
- Checked Plane – Datum Line.
- Checked Plane – Datum Plane.

Where: Line means Class Line feature: Line, Cylinder and Cone.

Tutor Translator

Parallelism

PARAL(MEMORY[120], MEMORY[123]) translates to →

```
DIM PARAL1 = PARALLELISM OF LINE E.MEMORY[120].ID,RFS TO CYLINDER  
E.MEMORY[123].ID,RFS EXTENDLENGTH=0.0000 UNITS=IN ,  
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE  
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL  
M 0.0000 .0100 0.0000 10.0000 5.0000 5.0000 10.0000 9.9900 ----->  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.MEMORY[120].TP = PARAL1.MEAS  
ASSIGN/E.MEMORY[120].GTOL = 2  
ASSIGN/E.WM1 = E.MEMORY[120]
```

PARAL(>MEMORY[130], MEMORY[120], MEMORY[123])
translates to →

```
DIM PARAL2= PARALLELISM OF LINE E.MEMORY[120].ID,RFS TO CYLINDER  
E.MEMORY[123].ID,RFS EXTENDLENGTH=0.0000 UNITS=IN ,  
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE  
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL  
M 0.0000 .0100 0.0000 10.0000 5.0000 5.0000 10.0000 9.9900 ----->  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[120]  
ASSIGN/E.WM1.TP = PARAL2.MEAS  
ASSIGN/E.WM1.GTOL = 2  
ASSIGN/E.MEMORY[130] = E.WM1
```

PARAL DIST(MEMORY[120], MEMORY[123],6.) translates to →

```
DIM PARAL3= PARALLELISM OF LINE E.MEMORY[120].ID,RFS TO CYLINDER  
E.MEMORY[123].ID,RFS EXTENDLENGTH=12.0000 UNITS=IN ,  
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE  
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL  
M 0.0000 0.0100 0.0000 8.4853 4.2426 4.2426 8.4853 8.4753 ----->  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.MEMORY[120].TP = PARAL3.MEAS  
ASSIGN/E.MEMORY[120].GTOL = 2  
ASSIGN/E.WM1 = E.MEMORY[120]
```

PARAL DIST(MEMORY[120], MEMORY[123], DISTVAR)
translates to →

Where the variable DISTVAR contains the dist value.

```
DIM PARAL5= PARALLELISM OF LINE E.MEMORY[120].ID,RFS TO CYLINDER  
E.MEMORY[123].ID,RFS EXTENDLENGTH=DISTVAR*2 UNITS=IN ,  
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
```

```
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 8.4853 4.2426 4.2426 8.4853 8.4753 ----->
```

```
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[120].TP = PARAL5.MEAS
ASSIGN/E.MEMORY[120].GTOL = 2
ASSIGN/E.WM1 = E.MEMORY[120]
```

⚠ Warning: For commands where the checked element is a plane, the datum element is a line and the DIST is required the following comment is written in place of the tolerance:

```
COMMENT/DOC,=====
,WARNING: PARALLELISM BETWEEN PLANE AND
,LINE USING PROJECTION, DISTANCE IS NOT ALLOWED
=====
```

Perpendicularity

PERPEN(MEMORY[120], MEMORY[123]) translates to →

```
DIM PERP1= PERPENDICULARITY OF LINE E.MEMORY[120].ID,RFS TO CYLINDER
E.MEMORY[123].ID,RFS EXTENDLENGTH=0.0000 UNITS=IN ,$
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 .0100 0.0000 10.0000 5.0000 5.0000 10.0000 9.9900 ----->

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[120].TP = PERP1.MEAS
ASSIGN/E.MEMORY[120].GTOL = 3
ASSIGN/E.WM1 = E.MEMORY[120]
```

PERPEN(>MEMORY[130], MEMORY[120], MEMORY[123])
translates to →

```
DIM PERP2= PERPENDICULARITY OF LINE E.MEMORY[120].ID,RFS TO CYLINDER
E.MEMORY[123].ID,RFS EXTENDLENGTH=0.0000 UNITS=IN ,$
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 .0100 0.0000 10.0000 5.0000 5.0000 10.0000 9.9900 ----->

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = E.MEMORY[120]
ASSIGN/E.WM1.TP = PERP2.MEAS
ASSIGN/E.WM1.GTOL = 3
ASSIGN/E.MEMORY[130] = E.WM1
```

PERPEN DIST(MEMORY[120], MEMORY[123],6.) translates to →

```
DIM PERP3= PERPENDICULARITY OF LINE E.MEMORY[120].ID,RFS TO CYLINDER
E.MEMORY[123].ID,RFS EXTENDLENGTH=12.0000 UNITS=IN ,$
```

Tutor Translator

```
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 8.4853 4.2426 4.2426 8.4853 8.4753 ----->

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[120].TP = PERP3.MEAS
ASSIGN/E.MEMORY[120].GTOL = 3
ASSIGN/E.WM1 = E.MEMORY[120]
```

PERPEN DIST(MEMORY[120], MEMORY[123], DISTVAR)
translates to →

Where the variable DISTVAR contains the dist value.

```
DIM PERP5= PERPENDICULARITY OF LINE E.MEMORY[120].ID,RFS TO CYLINDER
E.MEMORY[123].ID,RFS EXTENDLENGTH=DISTVAR*2 UNITS=IN,$
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 8.4853 4.2426 4.2426 8.4853 8.4753 ----->
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[120].TP = PERP5.MEAS
ASSIGN/E.MEMORY[120].GTOL = 3
ASSIGN/E.WM1 = E.MEMORY[120]
```

⚠ Warning: For commands where the checked element is a plane, the datum element is a line and the DIST is required the following comment is written in place of the tolerance:

```
COMMENT/DOC,=====
,WARNING: PERPENDICULARITY BETWEEN PLANE AND
,LINE USING PROJECTION, DISTANCE IS NOT ALLOWED
=====
```

Angularity

ANGOL(MEMORY[120], MEMORY[123]) translates to →

```
DIM ANGLRTY1 =ANGULARITY FROM LINE E.MEMORY[120].ID TO CYLINDER
E.MEMORY[123].ID EXTENDLENGTH=0.0000 ANG=0.1000 UNITS=IN,$
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 9.9825 4.9913 4.9913 9.9825 9.9725 ----->

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[120].TP = ANGLRTY1.MEAS
ASSIGN/E.MEMORY[120].GTOL = 4
ASSIGN/E.WM1 = E.MEMORY[120]
```

ANGOL(>MEMORY[130], MEMORY[120], MEMORY[123],0.1) translates to →

```
DIM ANGLRTY2 =ANGULARITY FROM LINE E.MEMORY[120].ID TO CYLINDER
E.MEMORY[123].ID EXTENDLENGTH=0.0000 ANG=0.1000 UNITS=IN,$
```

```
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 9.9825 4.9913 4.9913 9.9825 9.9725 ----->
```

```
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = E.MEMORY[120]
ASSIGN/E.WM1.TP = ANGLRTY2.MEAS
ASSIGN/E.WM1.GTOL = 4
ASSIGN/E.MEMORY[130] = E.WM1
```

ANGOL DIST(MEMORY[120], MEMORY[123],0.1,6.) translates to →

```
DIM ANGLRTY3 =ANGULARITY FROM LINE E.MEMORY[120].ID TO CYLINDER
E.MEMORY[123].ID EXTENDLENGTH=12.0000 ANG=0.1000 UNITS=IN ,$
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 8.4705 4.2352 4.2352 8.4705 8.4605 ----->
```

```
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[120].TP = ANGLRTY3.MEAS
ASSIGN/E.MEMORY[120].GTOL = 4
ASSIGN/E.WM1 = E.MEMORY[120]]
```

ANGOL DIST(MEMORY[120], MEMORY[123],0.1, DISTVAR) translates to →

Where the variable DISTVAR contains the dist value.

```
DIM ANGLRTY5 =ANGULARITY FROM LINE E.MEMORY[120].ID TO CYLINDER
E.MEMORY[123].ID EXTENDLENGTH=DISTVAR*2 ANG=0.1000 UNITS=IN ,$
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 8.4705 4.2352 4.2352 8.4705 8.4605 ----->

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[120].TP = ANGLRTY5.MEAS
ASSIGN/E.MEMORY[120].GTOL = 4
ASSIGN/E.WM1 = E.MEMORY[120]
```

⚠ Warning: For commands where the checked element is a plane, the datum element is a line and the DIST is required the following comment is written in place of the tolerance:

```
COMMENT/DOC,=====
,WARNING: ANGULARITY BETWEEN PLANE AND
,LINE USING PROJECTION, DISTANCE IS NOT ALLOWED
=====
```

Concentricity, Coaxiality, and Symmetry Tolerances

The Tutor Concentricity, Coaxiality and Symmetry Tolerances have the following command formats for [Concentricity](#) (CONC), [Coaxiality](#) (COAX), & [Symmetry](#) (SYMM):

Tutor Translator

- GEO_TOL (CONC or COAX or SYMM, elchk, eldat)
- GEO_TOL (> eldest, CONC or COAX or SYMM, elchk, eldat)
- GEO_TOL (SYMM, elchk, MIDDLE, eldat1, eldat2)
- GEO_TOL (> eldest, SYMM, elchk, MIDDLE, eldat1, eldat2)
- GEO_TOL DIST (COAX, elchk, eldat, dist)
- GEO_TOL DIST (> eldest, COAX, elchk, eldat, dist)
- GEO_TOL BOUND (COAX or SYMM, elchk, eldat, boundaries)
- GEO_TOL BOUND (> eldest, COAX or SYMM, elchk, eldat, boundaries)
- GEO_TOL BOUND (SYMM, elchk, MIDDLE, eldat1, eldat2, boundaries)
- GEO_TOL BOUND (> eldest, SYMM, elchk, MIDDLE, eldat1, eldat2, boundaries)

Where:

- **eldest** - Memory that will contain the result. If omitted, the result will be written in the memory of the CHECKED element.
- **elchk** - Memory in which the CHECKED element is located. It may be an element variable, the element_array variable, or a working storage variable (E.WM1 or E.WM2). If omitted, the E.WM1 is used.
- **eldat** - Memory in which the DATUM element is held. It can be an element variable, the element_array variable, or a working storage variable (E.WM1 or E.WM2). If omitted, the E.WM1 is used.
- **angle** - The theoretical angle between the two elements, in degrees and thousandths (for the angularity check only)
- **dist** - The distance from the checked element baricenter (+and -) which bind the length or area of the element.
- **boundaries** - Represents a sequence of two or four elements (lines or measured planes constructed or defined theoretically) which bind the length or area of the checked element.
- **MIDDLE** – This keyword is used if the DATUM element in a symmetry check is the middle element between the two specified elements (eldat1, eldat2)
- **boundaries** - Represents a sequence of two or four elements (lines or measured planes constructed or defined theoretically) which bind the length or area of the checked element.

⚠ Warning: PC-DMIS does not support Boundary elements. The commands that use boundaries are not translated. The following Tutor DOS format commands are not translated: GEO_TOL (CONC or COAX or SYMM, eldat), & GEO_TOL (SYMM_C, elres, eldat, elcheck, R1, R2).

Concentricity

Tutor allows the following concentricity command:

- Checked Circle – Datum Circle.

GEO_TOL(CONC,MEMORY[133], MEMORY[134]) translates to →

```
DIM CONCEN1=CONCENTRICITY FROM CIRCLE E.MEMORY[133].ID TO CIRCLE  
E.MEMORY[134].ID UNITS=IN ,$  
GRAPH=OFF TEXT=OFF MULT=10.00 OUTPUT=NONE  
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL  
M 0.0000 0.0100 0.0000 2.0000 1.0000 0.0000 2.0000 1.9900 ----->
```

```

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[133].TP = CONCEN1.MEAS
ASSIGN/E.MEMORY[133].GTOL = 5
ASSIGN/E.WM1 = E.MEMORY[133]

```

GEO_TOL(>MEMORY[200],CONC,MEMORY[133], MEMORY[134]) translates to →

```

DIM CONCEN1=CONCENTRICITY FROM CIRCLE E.MEMORY[133].ID TO CIRCLE
E.MEMORY[134].ID UNITS=IN ,$
GRAPH=OFF TEXT=OFF MULT=10.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 2.0000 1.0000 0.0000 2.0000 1.9900 ----->

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = E.MEMORY[133]
ASSIGN/E.WM1.TP = CONCEN1.MEAS
ASSIGN/E.WM1.GTOL = 5
ASSIGN/E.MEMORY[200] = E.WM1

```

Coaxiality

Tutor allows the following coaxiality command:

- Checked Line – Datum Line.

where:

- Line means Class Line feature: Line, Cylinder and Cone.

GEO_TOL(COAX,MEMORY[123],MEMORY[124]) translates to →

```

DIM COAX1=COAXIALITY FROM CYLINDER E.MEMORY[123].ID TO CYLINDER
E.MEMORY[123].ID EXTENDLENGTH=0.0000 UNITS=IN ,$
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 #----->

ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[123].TP = COAX1.MEAS
ASSIGN/E.MEMORY[123].GTOL = 6
ASSIGN/E.WM1 = E.MEMORY[123]

```

GEO_TOL(>MEMORY[200],COAX,MEMORY[123],MEMORY[124]) translates to →

```

DIM COAX1=COAXIALITY FROM CYLINDER E.MEMORY[123].ID TO CYLINDER
E.MEMORY[123].ID EXTENDLENGTH=0.0000 UNITS=IN ,$
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 #----->

```

Tutor Translator

```
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = E.MEMORY[123]
ASSIGN/E.WM1.TP = COAX1.MEAS
ASSIGN/E.WM1.GTOL = 6
ASSIGN/E.MEMORY[200] = E.WM1
```

GEO_TOL DIST(COAX,MEMORY[123],MEMORY[124],6.) translates to →

```
DIM COAX1=COAXIALITY FROM CYLINDER E.MEMORY[123].ID TO CYLINDER
E.MEMORY[123].ID EXTENDLENGTH=12.0000 UNITS=IN ,$
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 0.0000 0.0000 0.0000 0.0000 #-----
```

```
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[123].TP = COAX1.MEAS
ASSIGN/E.MEMORY[123].GTOL = 6
ASSIGN/E.WM1 = E.MEMORY[123]
```

GEO_TOL DIST(COAX,MEMORY[123],MEMORY[124], DISTVAR) translates to →

Where the variable DISTVAR contains the dist value.

```
DIM COAX1=COAXIALITY FROM CYLINDER E.MEMORY[123].ID TO CYLINDER
E.MEMORY[123].ID EXTENDLENGTH= DISTVAR*2 UNITS=IN ,$
GRAPH=OFF TEXT=OFF MULT=0.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 0.0000 0.0000 0.0000 0.0000 #-----
```



```
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[123].TP = COAX1.MEAS
ASSIGN/E.MEMORY[123].GTOL = 6
ASSIGN/E.WM1 = E.MEMORY[123]
```

Symmetry

Tutor allows the following symmetry commands:

- Checked Line – Datum Line.
- Checked Line – Datum Plane.
- Checked Plane – Datum Line.
- Checked Plane – Datum Plane.
- Checked Line – Datum Middle Line.
- Checked Line – Datum Middle Plane.
- Checked Plane – Datum Middle Line.
- Checked Plane – Datum Middle Plane.

where:

- Line means Class Line feature: Line, Cylinder and Cone.

GEO_TOL(SYMM,MEMORY[122], MEMORY[123]) translates to →

```
DIM SYM1= SYMMETRY OF CONE E.MEMORY[122].ID TO CYLINDER E.MEMORY[123].ID
UNITS=IN ,$ GRAPH=OFF TEXT=OFF MULT=10.00 OUTPUT=None
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 #-----
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[122].TP = SYM1.MEAS
ASSIGN/E.MEMORY[122].GTOL = 7
ASSIGN/E.WM1 = E.MEMORY[122]
```

GEO_TOL(>MEMORY[200],SYMM,MEMORY[122], MEMORY[123]) translates to →

```
DIM SYM1= SYMMETRY OF CONE E.MEMORY[122].ID TO CYLINDER E.MEMORY[123].ID
UNITS=IN ,$ GRAPH=OFF TEXT=OFF MULT=10.00 OUTPUT=None
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 #-----
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = E.MEMORY[122]
ASSIGN/E.WM1.TP = SYM1.MEAS
ASSIGN/E.WM1.GTOL = 7
ASSIGN/E.MEMORY[200] = E.WM1
```

GEO_TOL(SYMM,MEMORY[122],MIDDLE, MEMORY[123], MEMORY[124]) translates to →

```
SYM1_LIN1 =FEAT/LINE,RECT,UNBND
THEO/68,13,14,0.8880738,0.3250576,0.3250576
ACTL/68,13,14,0.8880738,0.3250576,0.3250576
CONSTR/LINE,MID,E.MEMORY[123].ID,E.MEMORY[124].ID,112
DIM SYM1= SYMMETRY OF CONE E.MEMORY[122].ID TO LINE SYM1_LIN1 UNITS=IN ,
GRAPH=OFF TEXT=OFF MULT=10.00 OUTPUT=None
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 #-----
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.MEMORY[122].TP = SYM1.MEAS
ASSIGN/E.MEMORY[122].GTOL = 7
ASSIGN/E.WM1 = E.MEMORY[122]
```

GEO_TOL(>MEMORY[200],SYMM,MEMORY[122],MIDDLE, MEMORY[123], MEMORY[124]) translates to →

```
SYM1_LIN1 =FEAT/LINE,RECT,UNBND
THEO/68,13,14,0.8880738,0.3250576,0.3250576
```

Tutor Translator

```
ACTL/68,13,14,0.8880738,0.3250576,0.3250576
CONSTR/LINE,MID,E.MEMORY[123].ID,E.MEMORY[124].ID,112
DIM SYM1= SYMMETRY OF CONE E.MEMORY[122].ID TO LINE SYM1_LIN1 UNITS=IN ,$
GRAPH=OFF TEXT=OFF MULT=10.00 OUTPUT=NONE
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
M 0.0000 0.0100 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 #-----
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = E.MEMORY[122]
ASSIGN/E.WM1.TP = SYM1.MEAS
ASSIGN/E.WM1.GTOL = 7
ASSIGN/E.MEMORY[200] = E.WM1
```

True Position Tolerances

The Tutor True Position Tolerances have the following command formats:

1. Checked element only:
 - GEO_TOL (TP, elchk, in/out,cond)
 - GEO_TOL (> eldest, TP, elchk, in/out,cond)
2. Checked and Datum elements:
 - GEO_TOL (TP, elchk, in/out, eldat, in/out, cond, DM=dm, UDM=udm, LDM=ldm)
 - GEO_TOL (> eldest, TP, elchk, in/out, eldat, in/out, cond, DM=dm, UDM=udm, LDM=ldm)

where:

- **eldest** - Memory that will contain the result. If omitted, the result will be written in the memory of the CHECKED element.
- **elchk** - Memory in which the CHECKED element is located. It may be an element variable, the element_array variable, or a working storage variable (E.WM1 or E.WM2). If omitted, the E.WM1 is used.
- **eldat** - Memory in which the DATUM element is held. It can be an element variable, the element_array variable, or a working storage variable (E.WM1 or E.WM2). If omitted, the E.WM1 is used.
- **in/out** – These keywords are used for CHECKED element and, if present, for the DATUM element:
 - INNER for internal circles.
 - OUTER for external circles.
- **Cond** - This keyword is used to select material condition to be applied:
 - RFS: Regardless Feature Size. Checked element only.
 - MMC1: Maximum Material Condition. Checked element only.
 - LMC1: Minimum Material Condition. Checked element only.
 - MMC2: Maximum Material Condition. Checked and Datum.
 - LMC2: Minimum Material Condition. Checked and Datum.
- **dm, udm and ldm** are respectively the diameter and the upper and lower tolerances for the DATUM element (to be assigned to the DM, UDM and LDM variables).

Tutor only allows TP tolerance on circles (bosses and holes)

 The following Tutor DOS format commands are not translated: MMC1 or LMC1 (eltyp) & MMC2 or LMC2 (eltyp, elmem, DM=value, UDM=value, PIN_CHECKED)

 The resulting PC-DMIS commands depend on the current value of E.SELPL variable and of the selection made for "True Position Error Mode" in the configuration window.

SELPL X

GEO_TOL(TP,MEMORY[101],INNER,RFS)

GEO_TOL(TP,MEMORY[101],OUTER,RFS) translates to →

If True Position Error Mode is radius (IsTPRadius = True)

```

ASSIGN/E.SELPL = E.ZPLUS
DIM TP1= TRUE POSITION OF CIRCLE E.MEMORY[101].ID  UNITS=MM ,$ GRAPH=OFF
TEXT=OFF  MULT=1.00  OUTPUT=NONE  DEV PERPEN CENTERLINE=OFF  DISPLAY=RADIUS
AX  NOMINAL    MEAS      +TOL      -TOL      BONUS     DEV      DEVANG    OUTTOL
X   TH.X.N     10.000                -0.001
Y   TH.Y.N     10.000                -0.002
DF  TH.DM.N    6.000    TH.DM.U   -TH.DM.L      -5.780      5.580 <-----
TP  RFS          TH.TP.U           0.000     0.002    116.565  0.000 #-----
END OF DIMENSION TP1

ASSIGN/E.MEMORY[101].TP = TP1.TP.DEV/2
ASSIGN/E.MEMORY[101].BONUS = TP1.TP.BTOL/2
ASSIGN/E.MEMORY[101].GTOL = 1

```

If True Position Error Mode is diameter (IsTPRadius = False)

```

ASSIGN/E.SELPL = E.ZPLUS
DIM TP1= TRUE POSITION OF CIRCLE E.MEMORY[101].ID  UNITS=MM ,$ GRAPH=OFF
TEXT=OFF  MULT=1.00  OUTPUT=NONE  DEV PERPEN CENTERLINE=OFF  DISPLAY=DIAMETER
AX  NOMINAL    MEAS      +TOL      -TOL      BONUS     DEV      DEVANG    OUTTOL
X   TH.X.N     10.000                -0.001
Y   TH.Y.N     10.000                -0.002
DF  TH.DM.N    6.000    TH.DM.U   -TH.DM.L      -5.780      5.580 <-----
TP  RFS          TH.TP.U           0.000     0.002    116.565  0.000 #-----
END OF DIMENSION TP1

ASSIGN/E.MEMORY[101].TP = TP1.TP.DEV
ASSIGN/E.MEMORY[101].BONUS = TP1.TP.BTOL
ASSIGN/E.MEMORY[101].GTOL = 1

```

Tutor Translator

SELPL X

GEO_TOL(>MEMORY[201],TP,MEMORY[101],INNER, RFS)

GEO_TOL(>MEMORY[201],TP,MEMORY[101],OUTER, RFS) translates to →

If True Position Error Mode is radius (IsTPRadius = True)

```
DIM LOC2= TRUE POSITION OF CIRCLE E.MEMORY[101].ID UNITS=MM ,$  
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF  
DISPLAY=RADIUS  
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL  
X TH.X.N 10.000 -0.001  
Y TH.Y.N 10.000 -0.002  
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L -5.780 5.580 <-----  
TP RFS TH.TP.U 0.000 0.002 116.565 0.000 #-----  
END OF DIMENSION LOC2  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[101]  
ASSIGN/E.WM1.TP = TP2.TP.DEV/2  
ASSIGN/E.WM1.BONUS = TP2.TP.BTOL/2  
ASSIGN/E.WM1.GTOL = 1  
ASSIGN/E.MEMORY[201] = E.WM1
```

If True Position Error Mode is diameter (IsTPRadius = False)

```
DIM LOC2= TRUE POSITION OF CIRCLE E.MEMORY[101].ID UNITS=MM ,$  
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF  
DISPLAY=DIAMETER  
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL  
X TH.X.N 10.000 -0.001  
Y TH.Y.N 10.000 -0.002  
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L -5.780 5.580 <-----  
TP RFS TH.TP.U 0.000 0.002 116.565 0.000 #-----  
END OF DIMENSION LOC2  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[101]  
ASSIGN/E.WM1.TP = TP2.TP.DEV  
ASSIGN/E.WM1.BONUS = TP2.TP.BTOL  
ASSIGN/E.WM1.GTOL = 1  
ASSIGN/E.MEMORY[201] = E.WM1
```

SELPL X**GEO_TOL(TP,MEMORY[102],INNER, MMC1)****GEO_TOL(TP,MEMORY[102],OUTER, MMC1)** translates to →*If True Position Error Mode is radius (IsTPRadius = True)*

```

DIM TP3= TRUE POSITION OF CIRCLE E.MEMORY[102].ID UNITS=MM ,$
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF
DISPLAY=RADIUS
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL
X TH.X.N 10.000
Z TH.Z.N 30.000
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L 0.800 -6.420 6.020 <-----
TP MMC TH.TP.U 0.800 30.000 180.000 29.000 ----->
END OF DIMENSION TP3

ASSIGN/E.MEMORY[102].TP = TP3.TP.DEV/2
ASSIGN/E.MEMORY[102].BONUS = TP3.TP.BTOL/2
ASSIGN/E.MEMORY[102].GTOL = 1

```

If True Position Error Mode is diameter (IsTPRadius = False)

```

DIM TP3= TRUE POSITION OF CIRCLE E.MEMORY[102].ID UNITS=MM ,$
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF
DISPLAY=DIAMETER
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL
X TH.X.N 10.000
Z TH.Z.N 30.000
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L 0.800 -6.420 6.020 <-----
TP MMC TH.TP.U 0.800 30.000 180.000 29.000 ----->
END OF DIMENSION TP3

ASSIGN/E.MEMORY[102].TP = TP3.TP.DEV
ASSIGN/E.MEMORY[102].BONUS = TP3.TP.BTOL
ASSIGN/E.MEMORY[102].GTOL = 1

```

Tutor Translator

SELPL X

GEO_TOL(>MEMORY[201],TP,MEMORY[102],INNER, MMC1)

GEO_TOL(>MEMORY[201],TP,MEMORY[102],OUTER, MMC1) translates to →

If True Position Error Mode is radius (IsTPRadius = True)

```
DIM TP4= TRUE POSITION OF CIRCLE E.MEMORY[102].ID UNITS=MM ,$  
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF  
DISPLAY=RADIUS  
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL  
X TH.X.N 10.000 -0.010  
Z TH.Z.N 30.000 30.000  
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L 0.800 -6.420 6.020 <-----  
TP MMC TH.TP.U 0.800 30.000 180.000 29.000 ----->  
END OF DIMENSION TP4  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[102]  
ASSIGN/E.WM1.TP = TP4.TP.DEV/2  
ASSIGN/E.WM1.BONUS = TP4.TP.BTOL/2  
ASSIGN/E.WM1.GTOL = 1  
ASSIGN/E.MEMORY[201] = E.WM1
```

If True Position Error Mode is diameter (IsTPRadius = False)

```
DIM TP4= TRUE POSITION OF CIRCLE E.MEMORY[102].ID UNITS=MM ,$  
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF  
DISPLAY=DIAMETER  
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL  
X TH.X.N 10.000 -0.010  
Z TH.Z.N 30.000 30.000  
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L 0.800 -6.420 6.020 <-----  
TP MMC TH.TP.U 0.800 30.000 180.000 29.000 ----->  
END OF DIMENSION TP4  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[102]  
ASSIGN/E.WM1.TP = TP4.TP.DEV  
ASSIGN/E.WM1.BONUS = TP4.TP.BTOL  
ASSIGN/E.WM1.GTOL = 1  
ASSIGN/E.MEMORY[201] = E.WM1
```

SELPL X

GEO_TOL(TP,MEMORY[103],INNER, LMC1)

GEO_TOL(TP,MEMORY[103],OUTER, LMC1) translates to →

If True Position Error Mode is radius (IsTPRadius = True)

```
DIM TP5= TRUE POSITION OF CIRCLE E.MEMORY[103].ID UNITS=MM ,$
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF
DISPLAY=RADIUS
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL
Y TH.Y.N 10.000
Z TH.Z.N 10.000
DF TH.DM.N 5.850 TH.DM.U -TH.DM.L 0.800 -6.570 6.170 <-----
TP LMC TH.TP.U 0.800 13.932 -90.000 12.932 ----->
END OF DIMENSION TP5

ASSIGN/E.MEMORY[103].TP = TP5.TP.DEV/2
ASSIGN/E.MEMORY[103].BONUS = TP5.TP.BTOL/2
ASSIGN/E.MEMORY[103].GTOL = 1
```

If True Position Error Mode is radius (IsTPRadius = True)

```
DIM TP5= TRUE POSITION OF CIRCLE E.MEMORY[103].ID UNITS=MM ,$
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF
DISPLAY=DIAMETER
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL
Y TH.Y.N 10.000
Z TH.Z.N 10.000
DF TH.DM.N 5.850 TH.DM.U -TH.DM.L 0.800 -6.570 6.170 <-----
TP LMC TH.TP.U 0.800 13.932 -90.000 12.932 ----->
END OF DIMENSION TP5

ASSIGN/E.MEMORY[103].TP = TP5.TP.DEV
ASSIGN/E.MEMORY[103].BONUS = TP5.TP.BTOL
ASSIGN/E.MEMORY[103].GTOL = 1
```

Tutor Translator

SELPL X

GEO_TOL(>MEMORY[203],TP,MEMORY[103],INNER, LMC1)

GEO_TOL(>MEMORY[203],TP,MEMORY[103],OUTER, LMC1) translates to →

If True Position Error Mode is radius (IsTPRadius = True)

```
DIM TP6= TRUE POSITION OF CIRCLE E.MEMORY[103].ID UNITS=MM ,$  
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF  
DISPLAY=RADIUS  
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL  
Y TH.Y.N 10.000 -9.700  
Z TH.Z.N 10.000 10.000  
DF TH.DM.N 5.850 TH.DM.U -TH.DM.L 0.800 -6.570 6.170 <----  
TP LMC TH.TP.U 0.800 13.932 -90.000 12.932 ---->  
END OF DIMENSION TP6  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[103]  
ASSIGN/E.WM1.TP = TP6.TP.DEV/2  
ASSIGN/E.WM1.BONUS = TP6.TP.BTOL/2  
ASSIGN/E.WM1.GTOL = 1  
ASSIGN/E.MEMORY[203] = E.WM1
```

If True Position Error Mode is diameter (IsTPRadius = False)

```
DIM TP6= TRUE POSITION OF CIRCLE E.MEMORY[103].ID UNITS=MM ,$  
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF  
DISPLAY=DIAMETER  
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL  
Y TH.Y.N 10.000 -9.700  
Z TH.Z.N 10.000 10.000  
DF TH.DM.N 5.850 TH.DM.U -TH.DM.L 0.800 -6.570 6.170 <----  
TP LMC TH.TP.U 0.800 13.932 -90.000 12.932 ---->  
END OF DIMENSION TP6  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[103]  
ASSIGN/E.WM1.TP = TP6.TP.DEV  
ASSIGN/E.WM1.BONUS = TP6.TP.BTOL  
ASSIGN/E.WM1.GTOL = 1  
ASSIGN/E.MEMORY[203] = E.WM1
```

SELPL X

GEO_TOLL (TP, MEMORY[101], INNER, MEMORY[103], INNER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (TP, MEMORY[101], INNER, MEMORY[103], OUTER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (TP, MEMORY[101], OUTER, MEMORY[103], INNER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (TP, MEMORY[101], OUTER, MEMORY[103], OUTER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01) translates to →

If True Position Error Mode is radius (IsTPRadius = True)

```
DIM TP7= TRUE POSITION OF CIRCLE E.MEMORY[101].ID UNITS=MM ,$
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF
DISPLAY=RADIUS
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL
Y TH.Y.N 10.000
Z TH.Z.N 10.000
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L 0.200 0.000 0.000 ---#---
D1 5.850 5.850 0.010 0.010 CIRCLE E.MEMORY[103].ID AT RFS
TP MMC TH.TP.U 0.200 0.000 0.000 0.000 #-----
END OF DIMENSION TP7
```

```
ASSIGN/E.MEMORY[101].TP = TP7.TP.DEV/2
ASSIGN/E.MEMORY[101].BONUS = TP7.TP.BTOL/2
ASSIGN/E.MEMORY[101].GTOL = 1
```

If True Position Error Mode is diameter (IsTPRadius = False)

```
DIM TP7= TRUE POSITION OF CIRCLE E.MEMORY[101].ID UNITS=MM ,$
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF
DISPLAY=DIAMETER
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL
Y TH.Y.N 10.000
Z TH.Z.N 10.000
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L 0.200 0.000 0.000 ---#---
D1 5.850 5.850 0.010 0.010 CIRCLE E.MEMORY[103].ID AT RFS
TP MMC TH.TP.U 0.200 0.000 0.000 0.000 #-----
END OF DIMENSION TP7
```

```
ASSIGN/E.MEMORY[101].TP = TP7.TP.DEV
ASSIGN/E.MEMORY[101].BONUS = TP7.TP.BTOL
ASSIGN/E.MEMORY[101].GTOL = 1
```

Tutor Translator

SELPL X

GEO_TOLL (>MEMORY[204], TP, MEMORY[101], INNER, MEMORY[103], INNER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (>MEMORY[204], TP, MEMORY[101], INNER, MEMORY[103], OUTER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (>MEMORY[204], TP, MEMORY[101], OUTER, MEMORY[103], INNER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (>MEMORY[204], TP, MEMORY[101], OUTER, MEMORY[103], OUTER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01) translates to →

If True Position Error Mode is radius (IsTPRadius = True)

```
DIM TP8= TRUE POSITION OF CIRCLE E.MEMORY[101].ID UNITS=MM ,$  
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF  
DISPLAY=RADIUS  
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL  
Y TH.Y.N 10.000 0.000  
Z TH.Z.N 10.000 0.000  
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L 0.200 0.000 0.000 ---#---  
D1 5.850 5.850 0.010 0.010 CIRCLE E.MEMORY[103].ID AT RFS  
TP MMC TH.TP.U 0.200 0.000 0.000 0.000 #-----  
END OF DIMENSION TP8  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[101]  
ASSIGN/E.WM1.TP = TP8.TP.DEV/2  
ASSIGN/E.WM1.BONUS = TP8.TP.BTOL/2  
ASSIGN/E.WM1.GTOL = 1  
ASSIGN/E.MEMORY[204] = E.WM1
```

If True Position Error Mode is diameter (IsTPRadius = False)

```
DIM TP8= TRUE POSITION OF CIRCLE E.MEMORY[101].ID UNITS=MM ,$  
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF  
DISPLAY=DIAMETER  
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL  
Y TH.Y.N 10.000 0.000  
Z TH.Z.N 10.000 0.000  
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L 0.200 0.000 0.000 ---#---  
D1 5.850 5.850 0.010 0.010 CIRCLE E.MEMORY[103].ID AT RFS  
TP MMC TH.TP.U 0.200 0.000 0.000 0.000 #-----  
END OF DIMENSION TP8  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[101]  
ASSIGN/E.WM1.TP = TP8.TP.DEV/2
```

```

ASSIGN/E.WM1.BONUS = TP8.TP.BTOL/2
ASSIGN/E.WM1.GTOL = 1
ASSIGN/E.MEMORY[204] = E.WM1

```

SELPL X

GEO_TOLL (TP, MEMORY[101], INNER, MEMORY[103], INNER, LMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (TP, MEMORY[101], INNER, MEMORY[103], OUTER, LMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (TP, MEMORY[101], OUTER, MEMORY[103], INNER, LMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (TP, MEMORY[101], OUTER, MEMORY[103], OUTER, LMMC2, DM=11.7, UDM=0.01, LDM=-0.01)
translates to →

If True Position Error Mode is radius (IsTPRadius = True)

```

DIM TP9= TRUE POSITION OF CIRCLE E.MEMORY[101].ID UNITS=MM ,$
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF
DISPLAY=RADIUS
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL
Y TH.Y.N 10.000
Z TH.Z.N 10.000
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L 0.200 0.000 0.000 ---#---
D1 5.850 5.850 0.010 0.010 CIRCLE E.MEMORY[103].ID AT RFS
TP MMC TH.TP.U 0.200 0.000 0.000 0.000 #-----
END OF DIMENSION TP9

```

```

ASSIGN/E.MEMORY[101].TP = TP9.TP.DEV/2
ASSIGN/E.MEMORY[101].BONUS = TP9.TP.BTOL/2
ASSIGN/E.MEMORY[101].GTOL = 1

```

If True Position Error Mode is diameter (IsTPRadius = False)

```

DIM TP9= TRUE POSITION OF CIRCLE E.MEMORY[101].ID UNITS=MM ,$
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF
DISPLAY=DIAMETER
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL
Y TH.Y.N 10.000
Z TH.Z.N 10.000
DF TH.DM.N 6.000 TH.DM.U -TH.DM.L 0.200 0.000 0.000 ---#---
D1 5.850 5.850 0.010 0.010 CIRCLE E.MEMORY[103].ID AT RFS
TP MMC TH.TP.U 0.200 0.000 0.000 0.000 #-----
END OF DIMENSION TP9

```

```

ASSIGN/E.MEMORY[101].TP = TP9.TP.DEV
ASSIGN/E.MEMORY[101].BONUS = TP9.TP.BTOL
ASSIGN/E.MEMORY[101].GTOL = 1

```

Tutor Translator

SELPL X

GEO_TOLL (>MEMORY[205], TP, MEMORY[101], INNER, MEMORY[103], INNER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (>MEMORY[205], TP, MEMORY[101], INNER, MEMORY[103], OUTER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (>MEMORY[205], TP, MEMORY[101], OUTER, MEMORY[103], INNER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01)

GEO_TOLL (>MEMORY[205], TP, MEMORY[101], OUTER, MEMORY[103], OUTER, MMC2, DM=11.7, UDM=0.01, LDM=-0.01) translates to →

If True Position Error Mode is radius (IsTPRadius = True)

```
DIM TP10= TRUE POSITION OF CIRCLE E.MEMORY[101].ID UNITS=MM ,$  
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF  
DISPLAY=RADIUS  
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL  
Y TH.Y.N 10.000 -9.700  
Z TH.Z.N 10.000 10.000  
DF TH.DM.N 5.850 TH.DM.U -TH.DM.L 0.800 -6.570 6.170 <----  
TP LMC TH.TP.U 0.800 13.932 -90.000 12.932 ----->  
END OF DIMENSION TP10  
  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[101]  
ASSIGN/E.WM1.TP = TP10.TP.DEV/2  
ASSIGN/E.WM1.BONUS = TP10.TP.BTOL/2  
ASSIGN/E.WM1.GTOL = 1  
ASSIGN/E.MEMORY[205] = E.WM1
```

If True Position Error Mode is diameter (IsTPRadius = False)

```
DIM TP10= TRUE POSITION OF CIRCLE E.MEMORY[101].ID UNITS=MM ,$  
GRAPH=OFF TEXT=OFF MULT=1.00 OUTPUT=NONE DEV PERPEN CENTERLINE=OFF  
DISPLAY=DIAMETER  
AX NOMINAL MEAS +TOL -TOL BONUS DEV DEVANG OUTTOL  
Y TH.Y.N 10.000 -9.700  
Z TH.Z.N 10.000 10.000  
DF TH.DM.N 5.850 TH.DM.U -TH.DM.L 0.800 -6.570 6.170 <----  
-  
TP LMC TH.TP.U 0.800 13.932 -90.000 12.932 ----->  
END OF DIMENSION TP10  
ASSIGN/E.WM2 = E.WM1  
ASSIGN/E.WM1 = E.MEMORY[101]  
ASSIGN/E.WM1.TP = TP10.TP.DEV  
ASSIGN/E.WM1.BONUS = TP10.TP.BTOL  
ASSIGN/E.WM1.GTOL = 1  
ASSIGN/E.MEMORY[205] = E.WM1
```

APPROACH / NO_APPROACH Commands

APPROACH (0.2016411, 0.3528719, 0.9136861) translates to →

```
ASSIGN/E.APPR_VECT.ON = 1
ASSIGN/E.APPR_VECT.IJK = -UNIT(MPOINT(0.2016411, 0.3528719, 0.9136861))
```

APPROACH (VECT) translates to →

```
ASSIGN/E.APPR_VECT.ON = 1
ASSIGN/E.APPR_VECT.IJK = -VECT
```

Approach (-cos (ANG1)*cos (ANGLO),-sin (ANG1)*cos (ANGLO),-sin (ANGLO)) translates to →

```
ASSIGN/E.APPR_VECT.ON = 1
ASSIGN/E.APPR_VECT.IJK = -UNIT(MPOINT(-COS(ANG1)*COS(ANGLO),-
SIN(ANG1)*COS(ANGLO),-SIN(ANGLO)))
```

NO_APPROACH translates to →

```
ASSIGN/E.APPR_VECT.ON = 0
```

PL3L_DISTANCE Command

PL3L_DISTANCE (-4.8887, 2.4481, 2.6532) translates to →

```
ASSIGN/E.PL3L.D1 = -4.8887
ASSIGN/E.PL3L.D2 = 2.4481
ASSIGN/E.PL3L.D3 = 2.6532
```

Element Comment

This command is normally placed before measure, construction, relation, tolerance and define element commands. With PC-DMIS this comment is stored in the TutorElement using the new COMMENT field. It is necessary to store the comment after the TutorElement creation.

MEMORY[2] = "This is the second point" MSH(MEMORY[2],1) path PT2 translates to →

```
ASSIGN/E.BADMEAS=0
PNT1 = FEAT/POINT,RECT
...
...
ENDMEAS/
ASSIGN/E.WM2 = E.WM1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT1" )
```

Tutor Translator

```
ASSIGN/E.WM1.COMMENT = "This is the second point"  
ASSIGN/E.MEMORY[2] = E.WM1  
...  
... Evaluation commands.  
...
```

In case a string variable is in the place of an immediate string the command is:

WM1=STRVAR requires translation to →

```
ASSIGN/E.WM1=E.STRVAR
```

Element Evaluation

Tutor evaluates element when it executes a direct or indirect measurement (Mxxx or Ixxx) command or when it executes an OUTPUT command. Evaluation is translated with a TUTOR_OUTPUT subroutine call.

```
SUBROUTINE/TUTOR_OUTPUT,TutorSub.prg  
TE = : TUTOR ELEMENT STRUCTURE,  
OFT = OFMT : TUTOR OUTPUT FORMAT TABLE,  
TT = : THEORETICAL VALUES TABLE,  
ELEMNAME = : NAME OF TUTOR ELEMENT STRUCTURE,
```

The evaluation of the element

MEMORY[10] translates to →

```
=CALLSUB/TUTOR_OUTPUT,TutorSub.PRG:E.MEMORY[10], E.OFMT, E.TH, "MEMORY[10]",,
```

 The output layout is fixed. It is not possible to program the Headers, Footers, Titles, Vertical layouts or the Field order. Output labels are always given in English.

DELAY Command

The DELAY command is not translated because PC-DMIS does not support this capability.

EMERGENCY Command

The EMERGENCY command is not translated because PC-DMIS does not support this capability.

ASSIGNMENT Commands

The following assignment commands are supported for translation into PC-DMIS:

[INTVAR as an integer variable](#)

[REALVAR as an integer variable](#)

[BOOLVAR as an integer variable](#)

[STRINGVAR as an integer variable](#)

INTVAR as an integer variable

READ (INTVAR) translates to →

```
RC1 =COMMENT/INPUT,YES,Enter a valid integer value!
ASSIGN/E.INTVAR = INT(RC1.INPUT)
COMMENT/READOUT,NO,"Read: "+ RC1.INPUT
```

READ PROMPT “This is a prompt. Enter an integer value!” (INTVAR) translates to →

```
RC1 =COMMENT/INPUT,YES, This is a prompt. Enter a valid integer value!
ASSIGN/E.INTVAR = INT(RC1.INPUT)
COMMENT/READOUT,NO," This is a prompt. Enter a valid integer value!: "+ RC1.INPUT
```

READ PROMPT PROMVAR (INTVAR) translates to →

PROMVAR is a string variable that contains the prompt.

```
RC1 =COMMENT/INPUT,YES, E.PROMVAR
ASSIGN/E.INTVAR = INT(RC1.INPUT)
COMMENT/READOUT,NO, E.PROMVAR+": "+ RC1.INPUT
```

REALVAR as an integer variable

READ (REALVAR) translates to →

```
RC2 =COMMENT/INPUT,NO,Enter a valid real value!
ASSIGN/E.REALVAR = DOUBLE(RC2.INPUT)
COMMENT/READOUT,NO,"Read: "+RC2.INPUT
```

Tutor Translator

READ PROMPT “This is a prompt. Enter a Real value!” (REALVAR) translates to →

```
RC2 =COMMENT/INPUT,NO, This is a prompt. Enter a valid  
ASSIGN/E.REALVAR = DOUBLE(RC2.INPUT)  
COMMENT/READOUT,NO," This is a prompt. Enter a valid real value! "+RC2.INPUT
```

READ PROMPT PROMVAR (REALVAR) translates to →

PROMVAR is a string variable that contains the prompt.

```
RC2 =COMMENT/INPUT,NO, E.PROMVAR  
ASSIGN/E.REALVAR = DOUBLE(RC2.INPUT)  
COMMENT/READOUT,NO, E.PROMVAR+": "+RC2.INPUT
```

BOOLVAR as an integer variable

READ (BOOLVAR) translates to →

```
RC3 =COMMENT/INPUT,NO,Enter a boolean value (0=False,1=True)!  
ASSIGN/E.BOOLVAR = INT(RC3.INPUT)  
COMMENT/READOUT,NO,"Read: "+BOOLVAL(RC3.INPUT)
```

READ PROMPT “This is a prompt. Enter a boolean value (0=False,1=True)!” (BOOLVAR) translates to →

```
RC3 =COMMENT/INPUT,NO, This is a prompt. Enter a boolean value  
(0=False,1=True)!  
ASSIGN/E.BOOLVAR = INT(RC3.INPUT)  
COMMENT/READOUT,NO," This is a prompt. Enter a boolean value (0=False,1=True)!:  
"+E.BOOLVAL(RC3.INPUT)
```

READ PROMPT PROMVAR (BOOLVAR) translates to →

PROMVAR is a string variable that contains the prompt.

```
RC3 =COMMENT/INPUT,NO, E.PROMVAR  
ASSIGN/E.BOOLVAR = INT(RC3.INPUT)  
COMMENT/READOUT,NO, E.PROMVAR+": "+E.BOOLVAL(RC3.INPUT)
```

STRINGVAR as an integer variable

READ (STRINGVAR) translates to →

```
RC4 =COMMENT/INPUT,NO,Enter a valid string!  
ASSIGN/E.STRVAR = RC4.INPUT  
COMMENT/READOUT,NO,"Read: "+RC4.INPUT
```

READ PROMPT “This is a prompt. Enter a valid string!” (STRVAR)

```
RC4 =COMMENT/INPUT,NO, This is a prompt. Enter a valid string!
ASSIGN/E.STRVAR = RC4.INPUT
COMMENT/READOUT,NO,"This is a prompt. Enter a valid string!": " + RC4.INPUT
```

READ PROMPT PROMVAR (STRINGVAR) translates to →

PROMVAR is a string variable that contains the prompt.

```
RC4 =COMMENT/INPUT,NO, E.PROMVAR
ASSIGN/E.STRVAR = RC4.INPUT
COMMENT/READOUT,NO, E.PROMVAR+": "RC4.INPUT
```

THEO Command

The THEO command is translated as an ASSIGN command.

THEO (field=NomVal) translates to →

```
ASSIGN/TH.field.N = NomVal
```

THEO (Ufield =UpToVal) translates to →

```
ASSIGN/TH.field.UT = UpToVal
```

THEO (Lfield = LowToVal) translates to →

```
ASSIGN/TH.field.LT = LowToVal
```

The value of **field** identifies the field of the element and the associated theoretical or a tolerance values of the following: X, Y, Z, PR, PA, CX, CY, CZ, DM, DM2, DS, A, AXY, AYZ, AZX, F, TP, SDEV.

The ISODM field is ignored.

Output Control

Tutor output control is translated to PC-DMIS to determine how part program results are output for review.

Tutor Translator

Output Basics

HEADER("C:\WTutor\header.txt") translates to →

```
=CALLSUB/TUTOR_HEADER,TutorSub.PRG:E.OFMT,"C:\WTutor\header.txt",,
```

HEADER("This is an Header String") translates to →

```
=CALLSUB/TUTOR_HEADER,TutorSub.PRG: E.OFMT,"This is an Header String",,  
HEADERFILE = "C:\WTutor\header.txt"
```

HEADER(HEADERFILE) translates to →

```
ASSIGN/HEADERFILE = "C:\WTutor\header.txt"  
=CALLSUB/TUTOR_HEADER,TutorSub.PRG: E.OFMT, HEADERFILE,,  
HEADERSTR = "This is an Header String"
```

HEADER(HEADERSTR) translates to →

```
ASSIGN/HEADERSTR = "This is an header string"  
=CALLSUB/TUTOR_HEADER,TutorSub.PRG: E.OFMT, HEADERSTR,,
```

OUTPUT ON translates to →

```
ASSIGN/ E.OFMT.ON=1
```

OUTPUT OFF translates to →

```
ASSIGN/ E.OFMT.ON=0
```

OUTPUT MEMORY[10] translates to →

```
=CALLSUB/TUTOR_OUTPUT,TutorSub.PRG:E.MEMORY[10], E.OFMT, E.TH,"MEMORY[10]",,,
```

OUTPUT DY MEMORY[10] translates to →

```
=CALLSUB/TUTOR_OUTPUT,TutorSub.PRG:E.MEMORY[10], E.OFMT, E.TH,"MEMORY[10]",2,,
```

OUTPUT PRN MEMORY[10] translates to →

```
=CALLSUB/TUTOR_OUTPUT,TutorSub.PRG:E.MEMORY[10], E.OFMT, E.TH,"MEMORY[10]",3,,
```

OUTPUT FILE1 MEMORY[10] translates to →

```
=CALLSUB/TUTOR_OUTPUT,TutorSub.PRG:E.MEMORY[10], E.OFMT, E.TH,"MEMORY[10]",4,,
```

OUTPUT FILE2 MEMORY[10] translates to →

```
=CALLSUB/TUTOR_OUTPUT,TutorSub.PRG:E.MEMORY[10], E.OFMT, E.TH,"MEMORY[10]",5,,
```

OUTPUT FILE3 MEMORY[10] translates to →

```
=CALLSUB/TUTOR_OUTPUT,TutorSub.PRG:E.MEMORY[10], E.OFMT, E.TH,"MEMORY[10]",6,,
```

OUTPUT FILE4 MEMORY[10] translates to →

```
=CALLSUB/TUTOR_OUTPUT,TutorSub.PRG:E.MEMORY[10], E.OFMT, E.TH,"MEMORY[10]",7,,
```

The following two commands use a string variable for the STR_VAR value:

OUTPUT STR_VAR translates to →

```
=CALLSUB/TUTOR_OUTSTRING,TutorSub.PRG: E.OFMT,1, E.STR_VAR,,
```

OUTPUT ("THIS IS AN OUTPUT STING") translates to →

```
=CALLSUB/TUTOR_OUTSTRING,TutorSub.PRG: E.OFMT,1, "THIS IS AN OUTPUT STING",,
```

 The following Tutor output commands are not translated:

```
OUTPUT SLINE(1, 2, 3 of 4) MEMORY[10]
COMPARE
OUT_GRAPH
SYNCHRODEV or NO_SYNCHRODEV (PD-DMIS is always synchronized)
```

Output to Screen

The DY (Varlist) command is translated as an assignment command in which all the variables in the varlist are converted to string and concatenated to the other variable types (if any). The resulting string (DY_STRING) is passed to the Tutor_dy subroutine that emulates the DY output.

DY translates to →

```
=CALLSUB/TUTOR_DEV_ENDIS,TutorSub.PRG: E.OFMT,2,1,,
```

NODY translates to →

```
=CALLSUB/TUTOR_DEV_ENDIS,TutorSub.PRG: E.OFMT,2,0,,
```

The conversion rules are:

- **String** : No conversion.
ASSIGN/DY_STRING = E.STRING_VAR
or
ASSIGN/DY_STRING = "STRING"
- **Integer** : Use the FORMAT function:
ASSIGN/DY_STRING = FORMAT("%d", E.INT_VAR);
or
ASSIGN/DY_STRING = "10"
- **Real**: Use the FORMAT function:
ASSIGN/DY_STRING = FORMAT("%f", E.REAL_VAR);

Tutor Translator

- or
ASSIGN/DY_STRING = "10.04"
- **Vector** and **Coord**: Use the FORMAT function:
ASSIGN/DY_STRING = FORMAT("%f %f %f", E.VECT_VAR)
 - **Boolean**: Use the BOOLVAL function:
ASSIGN/DY_STRING = E.BOOLVAL (E.BOOL_VAR)

The BOOLVAL function is defined as:

```
ASSIGN/ E.BOOLVAL = FUNCTION( (BVAL) , IF(BVAL<>0 , "TRUE" , "FALSE" ) )
```

DY ("THIS IS A STRING") translates to →

```
ASSIGN/DY_STRING = " THIS IS A STRING"  
=CALLSUB/TUTOR_DY,TutorSub.PRG: E.OFMT, DY_STRING,,
```

DY (INT_VAR) translates to →

```
ASSIGN/DY_STRING = FORMAT("%d", E.INT_VAR)  
=CALLSUB/TUTOR_DY,TutorSub.PRG: E.OFMT, DY_STRING,,
```

DY (REAL_VAR) translates to →

```
ASSIGN/DY_STRING = FORMAT("%f", E.REAL_VAR)  
=CALLSUB/TUTOR_DY,TutorSub.PRG: E.OFMT, DY_STRING,,
```

DY (VECT_VAR) translates to →

```
ASSIGN/DY_STRING = FORMAT("%f %f %f", E.VECT_VAR)  
=CALLSUB/TUTOR_DY,TutorSub.PRG: E.OFMT, DY_STRING,,
```

The same for Coordinate variables

DY (BOOL_VAR) translates to →

```
ASSIGN/DY_STRING = E.BOOLVAL(E.BOOL_VAR)  
=CALLSUB/TUTOR_DY,TutorSub.PRG: E.OFMT, DY_STRING,,
```

DY ("INT VAR = ", INT_VAR, "; REAL VAR = ", REAL_VAR, "; VECT VAR = ", VECT_VAR) translates to →

```
ASSIGN/DY_STRING = "INT_VAR = " + E.INT_VAR + "; REAL_VAR = " + FORMAT("%f", E.REAL_VAR) +  
"; VECT_VAR = " + FORMAT("%f %f %f", E.VECT_VAR) =CALLSUB/TUTOR_DY,TutorSub.PRG:  
E.OFMT, DY_STRING,,
```

Where the values are defined as:

- INT_VAR is a variable of type integer.
- REAL_VAR is a variable of type Real.
- VECT_VAR is a variable of type Vector.

Output to Printer

The PRN (Varlist) command is translated as an assignment command in which all the variables in the varlist are converted to string and concatenated to the other variable types (if any). The resulting string (PRN_STRING) is passed to the Tutor_prn subroutine that emulates the PRN output.

PRN translates to →

```
=CALLSUB/TUTOR_DEV_ENDIS,TutorSub.PRG: E.OFMT,2,1,,
```

NOPRN translates to →

```
=CALLSUB/TUTOR_DEV_ENDIS,TutorSub.PRG: E.OFMT,2,0,,
```

PRN ("THIS IS A STRING") translates to →

```
ASSIGN/PRN_STRING = " THIS IS A STRING"
=CALLSUB/TUTOR_PRN,TutorSub.PRG: E.OFMT, PRN_STRING,,
```

PRN (INT_VAR) translates to →

```
ASSIGN/PRN_STRING = FORMAT("%d", E.INT_VAR)
=CALLSUB/TUTOR_PRN,TutorSub.PRG: E.OFMT, PRN_STRING,,
```

PRN (REAL_VAR) translates to →

```
ASSIGN/PRN_STRING = FORMAT("%f", E.REAL_VAR)
=CALLSUB/TUTOR_PRN,TutorSub.PRG: E.OFMT, PRN_STRING,,
```

PRN (VECT_VAR) translates to →

```
ASSIGN/PRN_STRING = FORMAT("%f %f %f", E.VECT_VAR)
=CALLSUB/TUTOR_PRN,TutorSub.PRG: E.OFMT, PRN_STRING,,
```

The same for coordinate variables.

PRN (BOOL_VAR) translates to →

```
ASSIGN/PRN_STRING = E.BOOLVAL(E.BOOL_VAR)
=CALLSUB/TUTOR_PRN,TutorSub.PRG: E.OFMT, PRN_STRING,,
```

PRN ("INT_VAR = ", INT_VAR, "; REAL_VAR = ", REAL_VAR, "; VECT_VAR = ", VECT_VAR)
translates to →

```
ASSIGN/PRN_STRING = "INT_VAR = " + E.INT_VAR + "; REAL_VAR = " + FORMAT("%f",
E.REAL_VAR) + "; VECT_VAR = " + FORMAT("%f %f %f", E.VECT_VAR)
=CALLSUB/TUTOR_PRN,TutorSub.PRG: E.OFMT, PRN_STRING,,
```

Tutor Translator

Where the values are defined as:

- INT_VAR is a variable of type integer.
- REAL_VAR is a variable of type Real.
- VECT_VAR is a variable of type Vector.

PRN_FF translates to →

FORMFEED/

Input / Output on Serial Line

Serial Lines are not supported. The following commands are not translated:

OPEN_SL, CLOSE_SL, SL, NOSL, & SL_INPUT

Output to File

OPEN (“FILENAME.MEA”)

OPEN 1 (“FILENAME.MEA”) translates to →

```
=CALLSUB/TUTOR_OPEN,TUTORSUB.PRG: E.OFMT,1,"FILENAME.MEA",  
OPEN 2 ("FILENAME.MEA")  
OPEN 3 ("FILENAME.MEA")  
OPEN 4 ("FILENAME.MEA")
```

The above commands are translated like the “OPEN 1” example with the corresponding file number changed.

FILE

FILE 1 translates to →

```
=CALLSUB/TUTOR_DEV_ENDIS,TutorSub.PRG: E.OFMT,4,1,,
```

FILE 2, FILE 3 & FILE 4 are translated like the “FILE 1” example with the corresponding file number changed.

NOFILE

NOFILE 1 translates to →

```
=CALLSUB/TUTOR_DEV_ENDIS,TutorSub.PRG: E.OFMT,4,0,,
```

NOFILE 2, NOFILE 3, & NOFILE 4 are translated like the “NOFILE 1” example with the corresponding file number changed.

The FILE (Varlist) command is translated as an assignment command in which all the variables in the varlist are converted to string and concatenated to the other variable types (if any). The

resulting string (FILE_STRING) is passed to the Tutor_FILE subroutine that emulates the FILE output.

The conversion rules are the same for DY commands:

FILE (“THIS IS A STRING”)

FILE 1 (“THIS IS A STRING”) translates to →

```
ASSIGN/FILE_STRING = " THIS IS A STRING"
=CALLSUB/TUTOR_FILE,TutorSub.PRG: E.OFMT, FILE_STRING,1,,
```

FILE 2(“THIS IS A STRING”)

FILE 3(“THIS IS A STRING”)

FILE 4(“THIS IS A STRING”)

The above commands are translated like the “FILE 1” example with the corresponding file number changed.

FILE (INT_VAR)

FILE 1 (INT_VAR) translates to →

```
ASSIGN/FILE_STRING = FORMAT("%d", E.INT_VAR)
=CALLSUB/TUTOR_FILE,TutorSub.PRG: E.OFMT, FILE_STRING,1,,
```

FILE 2(INT_VAR)

FILE 3(INT_VAR)

FILE 4(INT_VAR)

The above commands are translated like the “FILE 1” example with the corresponding file number changed.

FILE (REAL_VAR)

FILE 1 (REAL_VAR) translates to →

```
ASSIGN/FILE_STRING = FORMAT("%f", E.REAL_VAR)
=CALLSUB/TUTOR_FILE,TutorSub.PRG: E.OFMT, FILE_STRING,1,,
```

FILE 2(REAL_VAR)

FILE 3(REAL_VAR)

FILE 4(REAL_VAR)

Tutor Translator

The above commands are translated like the “FILE 1” example with the corresponding file number changed.

FILE (VECT_VAR)

FILE 1 (VECT_VAR) translates to →

```
ASSIGN/FILE_STRING = FORMAT("%f %f %f", E.VECT_VAR)
=CALLSUB/TUTOR_FILE,TutorSub.PRG: E.OFMT, FILE_STRING,1,,
```

FILE 2(VECT_VAR)

FILE 3(VECT_VAR)

FILE 4(VECT_VAR)

The above commands are translated like the “FILE 1” example with the corresponding file number changed.

The commands with COORDINATE variables are translated in the same way as Vector variables.

FILE (BOOL_VAR)

FILE 1 (BOOL_VAR) translates to →

```
ASSIGN/FILE_STRING = E.BOOLVAL(E.BOOL_VAR)=CALLSUB/TUTOR_FILE,TutorSub.PRG:
E.OFMT, FILE_STRING,,
```

FILE 2(BOOL_VAR)

FILE 3(BOOL_VAR)

FILE 4(BOOL_VAR)

The above commands are translated like the “FILE 1” example with the corresponding file number changed.

FILE (“INT VAR = ” , INT_VAR,”; REAL VAR = ”,REAL_VAR,”; VECT VAR = ”,VECT_VAR)

**FILE 1 (“INT VAR = ” , INT_VAR,”; REAL VAR = ”,REAL_VAR,”; VECT VAR =
”,VECT_VAR)**
translates to →

```
ASSIGN/FILE_STRING = "INT_VAR = " + E.INT_VAR + "; REAL_VAR = " + FORMAT("%f",
E.REAL_VAR) + "; VECT_VAR = " + FORMAT("%f %f %f", E.VECT_VAR)
=CALLSUB/TUTOR_FILE,TutorSub.PRG: E.OFMT, FILE_STRING,1,,
```

Where the values are defined as:

- INT_VAR is a variable of type integer.
- REAL_VAR is a variable of type Real.

- VECT_VAR is a variable of type Vector.

FILE 2(BOOL_VAR)

FILE 3(BOOL_VAR)

FILE 4(BOOL_VAR)

The above commands are translated like the “FILE 1” example with the corresponding file number changed.

CLOSE

CLOSE 1 translates to →

```
=CALLSUB/TUTOR_CLOSE,TUTORSUB.PRG: E.OFMT,1,,
```

CLOSE 2, CLOSE 3 & CLOSE 4 are translated like the “CLOSE 1” example with the corresponding file number changed.

Text File Management

Tutor allows I/O on 4 text files identified by F0 through F4.

OPENF (F0,"FileToOpen.txt") translates to →

```
IF/E.F0.NAME <> "NONE"
COMMENT/OPER, "FILE E.F0 ALREADY OPEN"
END_IF/

ELSE/
ASSIGN/E.F0.NAME = "FileToOpen.txt"
ASSIGN/E.F0.OPEN = "NO"
END_ELSE/
```

OPENF (F0,"FileToOpen.txt", APPEND) translates to →

```
IF/E.F0.NAME <> "NONE"
COMMENT/OPER, "FILE F0 ALREADY OPEN"
END_IF/

ELSE/
ASSIGN/E.F0.NAME = "FileToOpen.txt"
E.F0.PTR =FILE/OPEN,E.F0.NAME,APPEND
ASSIGN/E.F0.OPEN = "A"
END_ELSE/
```

Tutor Translator

RESET(F0,Status) translates to →

```
IF/E.F0.OPEN <> "A"
STS =FILE/EXISTS,E.F0.NAME
IF/STS <> 0
ASSIGN/STATUS = 0
END_IF/

ELSE/
ASSIGN/STATUS = 1
END_ELSE/

IF/E.F0.OPEN <> "NO"
FILE/CLOSE,E.F0.PTR
ASSIGN/E.F0.OPEN = "NO"
END_IF/

IF/STS<>0
E.F0.PTR =FILE/OPEN,E.F0.NAME,READ
ASSIGN/E.F0.OPEN = "R"
END_IF/

ELSE/
ASSIGN/E.F0.NAME = "NONE"
END_ELSE/
END_IF/
```

RESET(F0) translates to →

```
IF/E.F0.OPEN <> "A"
STS =FILE/EXISTS,E.F0.NAME
IF/E.F0.OPEN <> "NO"
FILE/CLOSE,E.F0.PTR
ASSIGN/E.F0.OPEN = "NO"
END_IF/
IF/STS<>0
E.F0.PTR =FILE/OPEN,E.F0.NAME,READ
ASSIGN/E.F0.OPEN = "R"
END_IF/

ELSE/
ASSIGN/E.F0.NAME = "NONE"
END_ELSE/
END_IF/
```

READ (F0,IntVar, RealVar, RealVar1, BoolVar, StringVar) translates to →

```
E.INTVAR      =FILE/READ_UPTO,E.F0.PTR," "
E.REALVAR     =FILE/READ_UPTO,E.F0.PTR," "
E.REALVAR1    =FILE/READ_UPTO,E.F0.PTR," "
E.STRINGVAR   =FILE/READ_UPTO,E.F0.PTR," "
E.BOOLSTR     =FILE/READ_UPTO,E.F0.PTR," "
ASSIGN/ E.BOOLVAR = EQUAL(E.BOOLSTR,"True")
```

READLN (F0,IntVar, RealVar) translates to →

```
E.INTVAR      =FILE/READ_UPTO,E.F0.PTR," "
E.REALVAR     =FILE/READ_UPTO,E.F0.PTR," "
VRES         =FILE/READLINE,E.F0.PTR,{DUMMY}
```

REWRITE (F0) translates to →

```
IF/E.F0.OPEN <> "A"
IF/E.F0.OPEN <> "NO"
FILE/CLOSE,E.F0.PTR
ASSIGN/E.F0.OPEN = "NO"
END_IF/

ELSE/
  E.F0.PTR  =FILE/OPEN,E.F0.NAME,WRITE
  ASSIGN/E.F0.OPEN  = "W"
  END_ELSE/
  END_IF/
```

WRITE (F0, " ",INTVAR, " ",REALVAR, " ",VECTVAR, " ",BOOLVAR, " ",STRINGVAR) translates to →

```
ASSIGN/STRTOWRITE = FORMAT("%d", E.INTVAR)+" "+FORMAT("%f", E.REALVAR)+" "+
FORMAT("%f %f %f", E.VECT_VAR)+" "+E.BOOLVAL(E.BOOLVAR)+" "+E.STRINGVAR
FILE/WRITE_BLOCK,E.F0.PTR,STRTOWRITE
```

Where the values are defined as:

- INT_VAR is a variable of type integer.
- REAL_VAR is a variable of type Real.
- BOOLVAR is a variable of type Boolean.
- VECT_VAR is a variable of type Vector.

Tutor Translator

WRITELN (F0," ",INTVAR," ",REALVAR) translates to →

```
ASSIGN/STRTOWRITE = FORMAT( " +"%d", E.INTVAR)+" "+FORMAT( "%f",
E.REALVAR)                                FILE/WRITELINE,E.F0.PTR,STRTOWRITE
```

Where the values are defined as:

- INT_VAR is a variable of type integer.
- REAL_VAR is a variable of type Real.

CLOSEF(F0) translates to →

```
IF/E.F0.OPEN <> "NO"
FILE/CLOSE,E.F0.PTR
ASSIGN/E.F0.OPEN = "NO"
ASSIGN/E.F0.NAME = "NONE"
END_IF/
FILE/CLOSE,E.F0.PTR
```

 The OPENF, READ, READLN, RESET, REWRITE, WRITE, WRITELN and CLOSEF text file commands listed before refer to E.F0 file ID. The same commands for file F1, F2 and F3 IDs E.F0 ID is replaced with the proper ID.

Output Protocol and Output Format

Format Command

The **FORMAT** command is translated as one or more calls to the TUTOR_FORMAT subroutine supplied in the TutorSub.prg library.

The parameters are:

1. Output format table. Is initialized at the beginning of the part program. The definitions for each of the values that follow E.OFMT are defined below.
2. First E.OFMT value: Device for which the format will be applied:
1=ALL
2=DY
3=PRN
4=FILE1
5=FILE2
6=FILE3
7=FILE4
3. Second E.OFMT value: Element type - Identifies the feature for which format will be applied:
1=Should
2=Circle
3=Sphere
4=Line
5=Cone

- 6=Cylinder
 7=Plane
 8=Curve
 9=Slot
 10=Set.
 11=Ellipse
 12=Surface .
 113=Paraboloid .
 114=Torus.
 115=Pick
 116=PointByRel
 117=CircleByRel
 118=LineByRel
 119=PlaneByRel
4. Third E.OFMT value: Field Identifier:
 1=X 2=Y 3=Z 4=PR 5=PA 6=CX 7=CY 8=CZ
 9=DM 10=DM2 11=DS 12=A
 13=AXY 14=AYZ 15=AZX 16=F
 17=SDEV 18=TD
5. Fourth E.OFMT value:Field format value:
 0=NONE No output
 1=MEAS Only measured value
 2=ERR Error
 3=OOT Out of tolerance

FORMAT DY(SHOULD, X=OOT, Y=ERR Z=MEAS) translates to →

```
=CALLSUB/TUTOR_FORMAT, TutorSub.PRG: E.OFMT,2,1,1,3,,  

=CALLSUB/TUTOR_FORMAT, TutorSub.PRG: E.OFMT,2,1,2,2,,  

=CALLSUB/TUTOR_FORMAT, TutorSub.PRG: E.OFMT,2,1,3,1,,
```

FORMAT DY(SHOULD, X=OOT, Y=OOT Z=OOT,CX=OOT,CY=OOT,CZ=OOT) translates to →

```
=CALLSUB/TUTORFORMAT, TutorSub.PRG: E.OFMT,2,1,1,3,,  

=CALLSUB/TUTORFORMAT, TutorSub.PRG: E.OFMT,2,1,2,3,,  

=CALLSUB/TUTORFORMAT, TutorSub.PRG: E.OFMT,2,1,3,3,,  

=CALLSUB/TUTORFORMAT, TutorSub.PRG: E.OFMT,2,1,6,3,,  

=CALLSUB/TUTORFORMAT, TutorSub.PRG: E.OFMT,2,1,7,3,,  

=CALLSUB/TUTORFORMAT, TutorSub.PRG: E.OFMT,2,1,8,3,,
```

Block Commands

BLKNB 10 translates to →

```
ASSIGN/ E.OFMT.BLKN = 10
```

BLKNB ON translates to →

```
ASSIGN/ E.OFMT.BLKNON = "ON"
```

BLKNB OFF translates to →

```
ASSIGN/ E.OFMT.BLKNON = "ON"
```

Tutor Translator

INCRBLK translates to →

```
ASSIGN/ E.OFMT.BLKN = E.OFMT.BLKN+1
```

DECRBLK translates to →

```
ASSIGN/ E.OFMT.BLKN = E.OFMT.BLKN-1
```

Dimension Mode Commands

SET_DIM (A=FULL) translates to →

```
ASSIGN/ E.OFMT.AFULL = 1
```

SET_DIM (A=HALF) translates to →

```
ASSIGN/ E.OFMT.AFULL = 0
```

SET_DIM (AREL=ACUTE) translates to →

```
ASSIGN/ E.OFMT.AREL = 1
```

SET_DIM (AREL=OBTUSE) translates to →

```
ASSIGN/ E.OFT.AREL = 0
```

SET_DIM (DM=WHOLE) translates to →

```
ASSIGN/ E.OFMT.DM = 1
```

SET_DIM (DM=HALF) translates to →

```
ASSIGN/ E.OFMT.DM = 0
```

SET_DIM (AMODE=A360) translates to →

```
ASSIGN/ E.OFMT.AMODE = 0
```

SET_DIM (AMODE=A180) translates to →

```
ASSIGN/ E.OFMT.AMODE = 1
```

 The SET_DIM command does not have effect on the first version of the translator even if it is already translated.

OUT_FORMAT (11,4) translates to →

```
ASSIGN/ E.OFMT.DF = "%11.4f"
```

OOT_COND translates to →

```
ASSIGN/ E.OFMT.OOTCOND = "ON"
```

NO_OOT_COND translates to →

```
ASSIGN/ E.OFMT.OOTCOND = "OFF"
```



The OOT_COND and NO_OOT_COND commands do not have effect on the first version of the translator even if it is already translated. .

NOTE: The following Tutor DOS commands are not translated:

- COMPARE.
- NO_COMPARE.
- PARTIAL_OUTPUT.
- COMPLETE_OUTPUT

THEO Commands

The THEO command is translated as an assignment of a TH table field.

THEO (X=10.0,UX=0.002,LX=0.002) translates to →

```
ASSIGN/E.TH.X.N = 10.0
ASSIGN/E.TH.X.UT = 0.002
ASSIGN/E.TH.X.LT = 0.002
```



The OOT_COND and NO_OOT_COND commands do not have effect

Tutor allows the assignment of DMS decimal format values to angle fields (A,AXY,AYZ and AZY). PC-DMIS only allows the assignment of decimal format values to angle fields. Therefore, the values is converted using the DMS2DEG function:

THEO (A=10.4534,UA=0.0015,LA=0.0015) translates to →

```
ASSIGN/TH.A.N = DMS2DEG(10.4534,OFMT.DMS)
ASSIGN/TH.A.UT = DMS2DEG(0.0015,OFMT.DMS)
ASSIGN/TH.A.LT = DMS2DEG(0.0015,OFMT.DMS)
```

NONCRITICITY(X=40,Y=50,Z=40) translates to →

```
ASSIGN/TH.X.NC = 40.0
ASSIGN/TH.Y.NC = 50.0
ASSIGN/TH.Z.NC = 40.0
```



The NONCRITICITY command does not have effect

Input/Output Parameters

The command INCH_MODE and METRIC_MODE that change the measure unit at runtime are translated only as a variable assignment because PC-DMIS does not allow run time measure unit change.

Translator notifies the operator every time these instructions are encountered.

METRIC_MODE translates to →

ASSIGN/ E.OFMT.METRIC=1

INCH_MODE translates to →

ASSIGN/ E.OFMT.METRIC=0

 In PC-DMIS only angle output can displayed in the DMS format. This setting is a part program attribute and can not be changed at execution time. The DEG_MODE and DMS_MODE commands change the measurement unit format at runtime. They are translated as a variable assignment. This variable is used by DMS2DEG function.

DEG_MODE translates to →

ASSIGN/ E.OFMT.DMS=0

DMS_MODE translates to →

ASSIGN/ E.OFMT.METRIC=0

 The SCALE Command is not translated.

Loading/Saving Output Formats

The SAVE and LOAD commands do not have effect on the first version of translator..

SAVE (FORMAT, “C:\FIL\POLI.DAT”) translates to →

=CALLSUB/SAVE_FORMAT,TutorSub.PRG: E.OFMT, "C:\FIL\POLI.DAT"

LOAD (FORMAT, “C:\FIL\POLI.DAT”) translates to →

=CALLSUB/LOAD_FORMAT,TutorSub.PRG: E.OFMT, "C:\FIL\POLI.DAT"

Element Evaluation

Tutor evaluates element when it executes a direct or indirect measurement (Mxxx or Ixxx) command or when it executes an OUTPUT command. Evaluation is translated with a TUTOR_OUTPUT subroutine call.

```
SUBROUTINE/TUTOR_OUTPUT,TutorSub.prg
TE = : TUTOR ELEMENT STRUCTURE,
OFT = OFMT : TUTOR OUTPUT FORMAT TABLE,
TT = : THEORETICAL VALUES TABLE,
ELEMNAME = : NAME OF TUTOR ELEMENT STRUCTURE,
```

The evaluation of the element

MEMORY[10] translates to →

```
=CALLSUB/TUTOR_OUTPUT,TutorSub.PRG:E.MEMORY[10], E.OFMT, E.TH,"MEMORY[10]","",
```



The output layout is fixed. It is not possible to program the Headers, Footers, Titles, Vertical layouts or the Field order. Output labels are always given in English.

Element Buffer Management

The define_element commands are translate as a generic feature:

DEFINE_ELEMENT (MEMORY[20],PICK,x=10.,Y=20.,z=30.) translates to →

```
PNT1 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,10,20,30,$
MEAS/XYZ,10,20,30,$
NOM/IJK,0,0,1,$
MEAS/IJK,0,0,1
ASSIGN/E.MEMORY[20] = TUTORELEMENT( "PNT1" )
```

DEFINE_ELEMENT (E.WM1,SHOULD,x=10.,Y=20.,z=30.) translates to →

```
PNT2 =GENERIC/POINT,DEPENDENT,RECT,$
NOM/XYZ,10,20,30,$
MEAS/XYZ,10,20,30,$
NOM/IJK,0,0,1,$
MEAS/IJK,0,0,1
ASSIGN/E.WM1 = TUTORELEMENT( "PNT2" )
```

Tutor Translator

DEFINE_ELEMENT (MEMORY[123], CIRCLE, x=10., Y=20., z=30., dm=40.) translates to →

```
CIR1 =GENERIC/CIRCLE,DEPENDENT,RECT,OUT,$  
NOM/XYZ,10,20,30,$  
MEAS/XYZ,10,20,30,$  
NOM/IJK,0,0,1,$  
MEAS/IJK,0,0,1,$  
DIAMETER/40,40  
ASSIGN/E.MEMORY[123] = TUTORELEMENT( "CIR1" )
```

DEFINE_ELEMENT (MEMORY[124], SPHERE, x=10., Y=20., z=30., dm=40.) translates to →

```
SPH1 =GENERIC/SPHERE,DEPENDENT,RECT,OUT,$  
NOM/XYZ,10,20,30,$  
MEAS/XYZ,10,20,30,$  
NOM/IJK,0,0,1,$  
MEAS/IJK,0,0,1,$  
DIAMETER/40,40  
ASSIGN/E.MEMORY[124] = TUTORELEMENT( "SPH1" )
```

Single diameter cylinder.

DEFINE_ELEMENT (MEMORY[4], cylind, x=1., y=2., z=3., cx=1., cy=.0, cz=.0, dm=4., dm2=4.) translates to →

```
CYL1 =GENERIC/CYLINDER,DEPENDENT,RECT,OUT,$  
NOM/XYZ,1,2,3,$  
MEAS/XYZ,1,2,3,$  
NOM/IJK,1,0,0,$  
MEAS/IJK,1,0,0,$  
DIAMETER/4,4,$  
DISTANCE/0,0  
ASSIGN/E.MEMORY[4] = TUTORELEMENT( "CYL1" )
```



The define_element for 2 diameter cylinders is not translated because PC-DMIS does not have this kind of feature.

DEFINE_ELEMENT (MEMORY[6], cone, x=1., y=2., z=3., cx=1., cy=.0, cz=.0, a=4.) translates to →

```
CON1 =GENERIC/CONE,DEPENDENT,RECT,OUT,$  
NOM/XYZ,1,2,3,$  
MEAS/XYZ,1,2,3,$  
NOM/IJK,1,0,0,$  
MEAS/IJK,1,0,0,$  
DISTANCE/0,0,$  
ANGLE/DMS2DEG(4.2756,OFMT.DMS),DMS2DEG(4.2756,OFMT.DMS)  
ASSIGN/E.MEMORY[6] = TUTORELEMENT( "CON1" )
```

DEFINE_ELEMENT (MEMORY[7],line,x=1.,y=2.,z=3.,cx=1.,cy=.0,cz=.0) translates to →

```
LIN1 =GENERIC/LINE,DEPENDENT,RECT,$
NOM/XYZ,1,2,3,$
MEAS/XYZ,1,2,3,$
NOM/IJK,1,0,0,$
MEAS/IJK,1,0,0,$
DISTANCE/0,0
ASSIGN/E.MEMORY[ 7 ] = TUTORELEMENT( "LIN1" )
```

DEFINE_ELEMENT (MEMORY[8], plane, x=1., y=2., z=3., cx=1., cy=.0, cz=.0) translates to →

```
PLN1 =GENERIC/PLANE,DEPENDENT,RECT,$
NOM/XYZ,1,2,3,$
MEAS/XYZ,1,2,3,$
NOM/IJK,1,0,0,$
MEAS/IJK,1,0,0
ASSIGN/E.MEMORY[ 8 ] = TUTORELEMENT( "PLN1" )
```

DEFINE_ELEMENT (MEMORY[10], parab, x=1., y=2., z=3., cx=1., cy=.0, cz=.0, dm=4.) translates to →

```
F1 =GENERIC/NONE,DEPENDENT,RECT,OUT,$
NOM/XYZ,1,2,3,$
MEAS/XYZ,1,2,3,$
NOM/IJK,1,0,0,$
MEAS/IJK,1,0,0,$
DIAMETER/4,4,$
ANGLE/0,0,$
DISTANCE/0,0
ASSIGN/E.MEMORY[ 10 ] = TUTORELEMENT( "F1" )
```

DEFINE_ELEMENT (MEMORY[11], SLOT, X=1., Y=2., Z=3., CX=1., CY=.0, CZ=.0, DM=21., DM2=140.) translates to →

```
SLT1 =GENERIC/ROUND_SLOT,DEPENDENT,RECT,OUT,$
NOM/XYZ,1,2,3,$
MEAS/XYZ,1,2,0,$
NOM/IJK,1,0,0,$
MEAS/IJK,1,0,0,$
NOM/IJK2,1,0,0,$
MEAS/IJK2,1,0,0,$
DIAMETER/21,21,$
DISTANCE/140,140
ASSIGN/E.MEMORY[ 11 ] = TUTORELEMENT( "SLT1" )
```

Tutor Translator

DEFINE_ELEMENT (MEMORY[12], ellips, x=1., y=2., z=3., cx=1., cy=.0, cz=.0, dm=15., dm2=109.)
translates to →

```
ELL1 =GENERIC/NONE,DEPENDENT,RECT,OUT,$  
NOM/XYZ,1,2,3,$  
MEAS/XYZ,1,2,3,$  
NOM/IJK,1,0,0,$  
MEAS/IJK,1,0,0,$  
DIAMETER/15,15,$  
ANGLE/0,0,$  
DISTANCE/109,109  
ASSIGN/E.MEMORY[12] = TUTORELEMENT( "ELL1" )
```

DEFINE_ELEMENT (MEMORY[13], torus, x=1., y=1., z=1., cx=1., cy=.0, cz=.0, dm=12., dm2=3.)
translates to →

```
F2 =GENERIC/NONE,DEPENDENT,RECT,OUT,$  
NOM/XYZ,1,2,3,$  
MEAS/XYZ,1,2,3,$  
NOM/IJK,1,0,0,$  
MEAS/IJK,1,0,0,$  
DIAMETER/12,12,$  
ANGLE/0,0,$  
DISTANCE/3,3  
ASSIGN/E.MEMORY[13] = TUTORELEMENT( "F2" )
```

Movement Commands

MOVE (x=10, y=20,z=30) translates to →

```
MOVE/POINT,NORMAL,10,20,30
```

MOVE (valx, valy,valz) translates to →

```
MOVE/POINT,NORMAL,10,20,30
```

MOVE (CoordVar) translates to →

```
MOVE/POINT,NORMAL, E.COORDVAR.X, E.COORDVAR.Y, E.COORDVAR.Z
```

MOVE (VectVar) translates to →

```
MOVE/POINT,NORMAL, E.VECTVAR.X, E.VECTVAR.Y, E.VECTVAR.Z
```

MOVE (COORDVAR, INCRVECT) translates to →

```
MOVE/POINT,NORMAL, E.COORDVAR.X+ E.INCRVECT.X, E.COORDVAR.Y+ E.INCRVECT.Y,  
E.COORDVAR.Z+ E.INCRVECT.Z
```

MOVE (VECTVAR, INCRVECT) translates to →

```
MOVE/POINT,NORMAL, E.VECTVAR.X+ E.INCRVECT.X, E.VECTVAR.Y+ E.INCRVECT.Y,
E.VECTVAR.Z+ E.INCRVECT.Z
```

Motion Control

MANMOVE translates to →

```
MODE/MANUAL
```

NCMOVE translates to →

```
IF/GETSETTING("Dcc mode") == 1
C1 =COMMENT/YESNO,NO, "Do you want continue in CNC mode?"
IF/C1.INPUT=="YES"
IF/GETSETTING("Current Alignment")=="STARTUP"
COMMENT/OPER,NO,TUTOR TRANSLATOR WARNING!
,Move and hits commands executed in STARTUP alignment could be ,affected by a
probe offset error due to the difference in probe ,management between Tutor and
PC-DMIS!
,Continue in safety mode or change moves and hits mode to RELEARN in order to
relearn them!
END_IF/
MODE/DCC
END_IF/
END_IF/
```

NCMOVE NOCONF translates to →

```
MODE/DCC
```

The command FLY(ON) uses the length of the last value defined with FLY(ON,LENGTH=val). The default value is 30 mm. PC-DMIS does not allow FLY without a length parameter so a variable must be defined at the beginning of the part program and initialized with:

- 30 mm if the PC-DMIS part program is in metric mode.
- 1.1811 inch if the PC-DMIS part program is in inch mode.

FLY(ON) translates to →

```
FLY/ON,E.FLYLEN
```

FLY(ON,LENGTH=23) translates to →

```
ASSIGN/E.FLYLEN = 23
FLY/ON,E.FLYLEN
```

Tutor Translator

FLY(OFF) translates to →

```
FLY/OFF, E.FLYLEN
```

Movement Parameters

In PC-DMIS the measure speed is limited to 20% of the max CMM speed. In contrast Tutor allows the measure speed to be set higher than 20% (up to 100%). If the measurement speed is higher than 20 then it is limited to 20 for PC-DMIS.

MSPEED 10 translates to →

```
TOUCHSPEED/10
```

MSPEED 80 translates to →

```
TOUCHSPEED/20
```

```
MSPEEDVAL = 50
```

```
...
```

MSPEED MSPEEDVAL translates to →

```
ASSIGN/ E.MSPEEDVAL = 50
```

```
...
```

```
TOUCHSPEED/ E.MSPEEDVAL
```

PSPEED 80 translates to →

```
MOVE SPEED/80
```

```
PSPEEDVAL = 50
```

```
...
```

PSPEED PSPEEDVAL translates to →

```
ASSIGN/ E.PSPEEDVAL = 50
```

```
...
```

```
MOVE SPEED/ E.PSPEEDVAL
```

The **ACC** command is translated as COMMENT/DOC command with the suggestion to use the PC-DMIS ACCELERATION command.

ACC 50 translates to →

```
COMMENT/DOC,NO,=====
,= ACC 50
,= Not translated command: Modify manually
,= Suggested command:
,= ACCELERATION/MAXACCELX=Xval, MAXACCELY=Yval,
,= MAXXACELZ=Zval
=====
```

DIST_APPROACH 12. translates to →

```
PREHIT / 12.
RETRACT / 12.
```

DIST_APPROACH , 12. translates to →

```
CHECK/ 12.
```

DIST_APPROACH 12. , 24. translates to →

```
PREHIT / 12.
RETRACT / 12.
CHECK/ 24.
```

 The FEED and OPT_SPEED commands are not translated because they are not supported by PC-DMIS

System Functions

Date Function

RVAR = DATE translates to →

where *RVAR* is a real var.

```
ASSIGN/ E.RVAR = SYSTEMDATE( "yyMM" . 'dd' )
```

Time Function

TVAR = TIME translates to →

where *TVAR* is a real var.

```
ASSIGN/ E.TVAR = SYSTEMTIME( "HHmm'.'ss'00'" )
```

Type Function

IVAR = MEMORY[1] |**TYPE** translates to →

where *IVAR* is an integer variable

```
=CALLSUB/TUTOR_TYPE,TutorSub.PRG:E.MEMORY[1], E.IVAR,,
```

Info Function

IVAR = INFO_FUNC(0) translates to →

```
ASSIGN/ E.IVAR = E.OFMT.INFO_FUNC_0
```

IVAR = INFO_FUNC(1) translates to →

```
ASSIGN/ E.IVAR = E.OFMT.METRIC == 0
```

The following INFO_FUNC are not translated:

- **INFO_FUNC(2)** : probe head number.
- **INFO_FUNC(3)** : probe tip number.
- **INFO_FUNC(9)** : CMM acceleration.
- **INFO_FUNC(12)** : Feed percentage.
- **INFO_FUNC(13)** : Optimal speed.
- **INFO_FUNC(14)** : Qualification gauge.
- **INFO_FUNC(19)** : Calibration probe X offset
- **INFO_FUNC(20)** : Calibration probe Y offset
- **INFO_FUNC(21)** : Calibration probe Z offset
- **INFO_FUNC(22)** : Scale factor X.
- **INFO_FUNC(23)** : Scale factor Y.
- **INFO_FUNC(24)** : Scale factor Z.
- **INFO_FUNC(25)** : Scale factor.
- **INFO_FUNC(30)** through **INFO_FUNC(61)** Thermal sensor read.

Current tip radius commands

RVAR = INFO_FUNC(4) translates to →

Where *RVAR* is a real var.

```
ASSIGN/ E.RVAR = PROBEDATA( "DIAMETER" ) / 2
```

IVAR = INFO_FUNC(4) translates to →

Where *IVAR* is an integer var.

```
ASSIGN/ E.RVAR = INT(PROBedata( "DIAMETER" ) / 2)
```

Current reference system number command

CUR_REF = INFO_FUNC(5) translates to →

```
ASSIGN/ E.CUR_REF = E.CUR_ALIGN
```

Current Approach command

CurAppro = INFO_FUNC(6) translates to →

```
ASSIGN/ E.CURAPPRO = GETSETTING( "PREHIT" )
```

Current Block Number command

CurBlkNm = INFO_FUNC(7) translates to →

```
ASSIGN/ E.CURBLKNM = E.OFMT.BLKN
```

Current Current Selpl command

CurSelPl = INFO_FUNC(8) translates to →

```
ASSIGN/E.CURSELPL = GETSETTING( "Workplane Value" ) % 3
IF/E.CURSELPL == 0
ASSIGN/E.CURSELPL = 3
END_IF/
```

Current Mspeed command

CurMSpeed = INFO_FUNC(10) translates to →

```
ASSIGN/ E.CURMSPEED = GETSETTING( "TOUCH SPEED" )
```

Current Pspeed command

CurPSpeed = INFO_FUNC(11) translates to →

```
ASSIGN/ E.CURPSPEED = GETSETTING( "MOVE SPEED" )
```

Current Fly Mode command

CurFlyMode = INFO_FUNC(15) translates to →

```
ASSIGN/ E.CURFLYMODE = GETSETTING("FLY MODE")
```

Ph9 present command

Ph9Present = INFO_FUNC(16) translates to →

```
ASSIGN/ E.PH9PRESENT = GETSETTING("PH9 PRESENT")
```

CMM Type command

CMMType = INFO_FUNC(17) translates to →

```
IF/GETSETTING( "MANUAL CMM" )
```

```
ASSIGN/ E.CMMTYPE = 0
```

```
END_IF/
```

```
ELSE/
```

```
IF/GETSETTING( "Dcc Mode" )== 1
```

```
ASSIGN/ E.CMMTYPE = 2
```

```
END_IF/
```

```
ELSE/
```

```
ASSIGN/ E.CMMTYPE = 1
```

```
END_ELSE/
```

```
END_ELSE/
```

Depth command

CurDepth = INFO_FUNC(18) translates to →

```
ASSIGN/ E.CURDEPTH = GETSETTING("CHECK")
```

Theoretical value commands

The following theoretical command is a representation of all of the theoretical commands. The value of X is replaced with the respective THEO value. The assigned FUNC() value is given below for each of the THEO values.

TheoX = INFO_FUNC(71) translates to →

```
ASSIGN/ E.THEO_X = E.TH.X.N  
X = 71      Y = 72      Z = 73      PR = 74      PA = 75  
CX = 76     CY = 77     CZ = 78      DM = 79      DM2 = 80  
DS = 81     A = 82     AXY = 83     AYZ = 84     AZX = 85
```

Theoretical value management:

While PC-DMIS features contain both theoretical and actual values (X,Y,Z,I,J,K,. . .) Tutor features don't have associated theoretical values.

When Tutor needs to compare an evaluated (measure or constructed) feature, it searches in the Output format table (once for each feature type) for feature fields that are required (X,Y,Z, CX, CY,CZ , . . .). If the fields are required, then they are compared with theoretical values. If comparison is required, Tutor uses values stored in the theoretical table (a single table for all types of feature). The fields involved in the evaluation are normally filled with the appropriate values before the feature evaluation. Feature evaluation can be made immediately after the measure or deferred to a second time.

During translation there is no way to be sure to have all theoretical values of a feature. This is true for several reasons:

- Field(s) values are not set yet.
- The Field value is not meaningful for the evaluation.
- Field(s) value could be defined inside a called procedure.

The adopted solution provides that when a measurement command is encountered Tutor translator looks for a measurement Path or MoveTf (hit) commands and adds them in the PC-DMIS measurement command as HIT/BASIC.... THEO=NO.

When the Measurement command is completed the **DmisFeatCmd.CalculateNominals** function is called.

Nevertheless this method does not guarantee the accuracy of nominal values because of differences in hits and probes.

Measurement hits and move commands

The main differences in measurement hits and move commands are:

1. Tutor moveTf (hit) coordinates are the probe center ball while PC-DMIS HIT/BASIC coordinates are the measure point (on the part).
2. Tutor moveTf don't have the measurement direction.
3. Tutor measured points can be supplied in two ways:
 - a. Using a path associated to the measure. This is very close to a PC-DMIS measured command.
 - b. Executing a proper number of moveTf commands after the measured command. In this case Translator tries to find these commands and add them in the PC-DMIS measurement command. Nevertheless it easy to find Tutor part programs that use FOR or LOOP commands to measure a feature. Because PC-DMIS does not have this capability, this kind of measurement is not translated.

Play Function

The PLAY command is not translated

Exec Function

The EXEC command modes are partially supported by PC-DMIS. The Tutor mode value is composed of the application execution mode and window mode.

The **Application Execution Modes** are:

- **0 (EXEC_NORMAL)** starts the application and the part program continues with the execution
- **1 (EXEC_WAIT)** starts the application and the part program waits until the application terminates execution before continuing.
- **16 (EXEC_NORMAL_ONCE)** checks that the application is not active and then starts the application in EXEC_NORMAL mode (if the application is active, the command is ignored).
- **17 (EXEC_WAIT_ONCE)** checks that the application is not active and then starts the application in EXEC_WAIT mode (if the application is active, the command is ignored)

The **Application Window Modes** are:

- **0 (DEFAULT)** opens the window with the default format and position for the application.
- **256 (EXEC_SHOW_NORMAL)** opens the window with the format and position the application had the last time it was started
- **512 (EXEC_SHOW_NORMAL_NA)** opens the window with the format and position the application had the last time it was started, but the window that was open before the application was started remains active.
- **768 (EXEC_SHOW_ICONIZED)** starts the application with the window reduced to an icon.
- **1024 (EXEC_SHOW_ICON_NA)** starts the application with the window reduced to an icon, while the window that was active before the application was started remains active.
- **1280 (EXEC_SHOW_MAXIMIZED)** starts the application with the window enlarged to fill the entire screen.

PC-DMIS supports only the following execution modes:

- EXEC_NORMAL → NO_DISPLAY , NO_WAIT
- EXEC_WAIT → NO_DISPLAY , WAIT

MODE = 0

EXEC("NOTEPAD.EXE", MODE, "FILETOEDIT.TXT"); translates to →

```
ASSIGN/MODE=0
IF/MODE%265 == 1
EXTERNALCOMMAND/NO_DISPLAY, WAIT ; C:\WINNT\SYSTEM32\NOTEPAD.EXE
C:\WINNT\WTUTOR.INI
END_IF/

ELSE/
EXTERNALCOMMAND/DISPLAY,NO_WAIT ; C:\WINNT\SYSTEM32\NOTEPAD.EXE
C:\WINNT\WTUTOR.INI
END_ELSE/
```

SYSTEM_INFO Function

SYSTEM_INFO(PROBE,HEAD,TIP,RAD,OFFSET,APOS,BPOS,QUAL) translates to →

```

IF/PROBEDATA( "ID" ,TIPARRAY[HEAD,TIP]) <> 0
ASSIGN/ E.RAD = PROBEDATA( "DIAM" ,TIPARRAY[HEAD,TIP])/2
ASSIGN/ E.OFFSET = PROBEDATA( "OFFSET" ,TIPARRAY[HEAD,TIP])
ASSIGN/ E.APOS = PROBEDATA( "A" ,TIPARRAY[HEAD,TIP])
ASSIGN/ E.BPOS = PROBEDATA( "B" ,TIPARRAY[HEAD,TIP])
ASSIGN/ E.QUAL = 1
END_IF/

ELSE/
ASSIGN/ E.QUAL = -3
END_ELSE/

```

SYSTEM_INFO (REFSYS,SYS,MROT,MINV,TRANS,I) translates to →

```

ASSIGN/ E.I = 0
IF/E.ALIGNMENTS[SYS]<>"NONE"
ASSIGN/ SYS_INFO_SAVE_REF = GETSETTING("Current Alignment")
RECALL/ALIGNMENT,INTERNAL,STARTUP
ASSIGN/SYS_INFO_REF = E.ALIGNMENTS[SYS]
ASSIGN/ E.MROT[0,0] = SYS_INFO_REF.XAXIS.I
ASSIGN/ E.MROT[1,0] = SYS_INFO_REF.XAXIS.J
ASSIGN/ E.MROT[2,0] = SYS_INFO_REF.XAXIS.K
ASSIGN/ E.MROT[0,1] = SYS_INFO_REF.YAXIS.I
ASSIGN/ E.MROT[1,1] = SYS_INFO_REF.YAXIS.J
ASSIGN/ E.MROT[2,1] = SYS_INFO_REF.YAXIS.K
ASSIGN/ E.MROT[0,2] = SYS_INFO_REF.ZAXIS.I
ASSIGN/ E.MROT[1,2] = SYS_INFO_REF.ZAXIS.J
ASSIGN/ E.MROT[2,2] = SYS_INFO_REF.ZAXIS.K
ASSIGN/ E.TRANS = SYS_INFO_REF.ORIGIN

RECALL/ALIGNMENT,INTERNAL, SYS_INFO_SAVE_REF
ASSIGN/ E.MINV[0,0] = E.MROT[0,0]
ASSIGN/ E.MINV[0,1] = E.MROT[1,0]
ASSIGN/ E.MINV[0,2] = E.MROT[2,0]
ASSIGN/ E.MINV[1,0] = E.MROT[0,1]
ASSIGN/ E.MINV[1,1] = E.MROT[1,1]
ASSIGN/ E.MINV[1,2] = E.MROT[2,1]
ASSIGN/ E.MINV[2,0] = E.MROT[0,2]
ASSIGN/ E.MINV[2,1] = E.MROT[1,2]
ASSIGN/ E.MINV[2,2] = E.MROT[2,2]
ASSIGN/ E.I = 1
END_IF/

```

Tutor Translator

Read I/O channels command

SYSTEM_INFO(IO,WORD1,WORD2) translates to →

```
CS1 =CALLSUB/TUTOR_IO,TUTORSUB.PRG:"READ", E.WORD1, E.WORD2,,
```

Write I/O channels commands

SYSTEM_INFO(IO,43690,21845)WRITE translates to →

```
CS1 =CALLSUB/TUTOR_IO,TUTORSUB.PRG:"WRITE", 43690,21845,,
```

SYSTEM_INFO(IO,WORD1,WORD2)WRITE translates to →

```
CS1 =CALLSUB/TUTOR_IO,TUTORSUB.PRG:"WRITE", E.WORD1, E.WORD2,,
```

The following SYSTEM_INFO commands are not translated:

- SYSTEM_INFO(PROBE,HEAD,TIP,RAD,OFFSET,APOS,BPOS,QUAL) WRITE
- SYSTEM_INFO (REFSYS,SYS,MROT,MINV,TRANS,I) WRITE
- SYSTEM_INFO (TP200, ON)
- SYSTEM_INFO (TP200, OFF)

GET_POSITION Command

GET_POSITION(C) translates to →

```
PNT1 =FEAT/POINT,RECT  
THEO/0,0,0,0,0,1  
ACTL/10,20,30,0,0,1  
READPOINT/  
ASSIGN/ E.C.XYZ = PNT1.XYZ
```

GETDIR Command

GETDIR(WM1, VV) translates to →

Where *VV* is a vector variable

```
ASSIGN/E.VV.IJK = E.WM1.ID.IJK
```

GETDIR(MEMORY[22], VV) translates to →

Where *VV* is a vector variable

```
ASSIGN/VV.IJK = E.MEMORY[22].ID.IJK
```

INCR Command

INCR MYVAR translates to →

Where *MYVAR* is an Integer or Real variable

```
ASSIGN/E.MYVAR = E.MYVAR+1
```

DECR Command

DECR MYVAR translates to →

Where *MYVAR* is an Integer or Real variable

```
ASSIGN/E.MYVAR = E.MYVAR-1
```

Scanning Control

The scanning command in Tutor is:

```
TFSCAN ("filename", pt1, pt2, dir, mins, maxs, comp, thresh)
```

Where:

- **filename** - The name of the file containing the picked points. By specifying the suffix .MEA, the output format is set to Tutor for point elements of the stored points. By specifying any other suffix or omitting the suffix, only the coordinates of the points taken are stored in the file (typical format of scanning files .SCN).
- **pt1** - The element with the starting point for the scan.
- **pt2** - The element with the end scan point
- **dir** - The vector that defines the scanning direction. This can be a variable vector.
- **mins** - The value of the minimum step. This can be a real variable.
- **maxs** - The value of the maximum step. This can be a real variable.
- **comp** - The compensation flag (TRUE or FALSE) for the scanning points. This can be a boolean variable.
- **thresh** - The threshold value for the final point of the scanning. This it can be a real variable.

TFSCAN("SCAN1.SCN,MEMORY[1],MEMORY[2],{0.707,0.707,0},1.0,2.0,TRUE,5.0) translates to →

```
ASSIGN/MIN_STEP = 1.0
ASSIGN/MAX_STEP = 2.0
ASSIGN/DIR_VECT = UNIT(MPOINT(0.707,0.707,0))
ASSIGN/TRSH = 5.0
ASSIGN/COMP_ONOFF = "YES"
PNT3 = FEAT/POINT,RECT
THEO/0,0,0,0,0,1
ACTL/0,0,0,0,0,1
```

Tutor Translator

```
READPOINT/
ASSIGN/INI_VECT = UNIT(PNT3.XYZ-E.MEMORY[1].XYZ)
SCN2    =FEAT/SCAN, LINEAROPEN, SHOWHITS=YES, SHOWALLPARAMS=YES
EXEC MODE=NORMAL, NOMS MODE=MASTER, CLEARPLANE=NO, SINGLEPOINT=NO, THICKNESS=0
FINDNOMS=2, SELECTEDONLY=NO, USEBESTFIT=NO, PROBECOMP=NO, AUTO MOVE=NO, DISTANCE=0
DIR1=LINE, INCR=2.54
HITTYPE=VECTOR
INITVEC=0,0,0
DIRVEC=0,0,0
CUTVEC=0,0,1
POINT1=0,0,0
MEAS/SCAN
BASICSCAN/LINE, SHOWHITS=YES, SHOWALLPARAMS=YES
E.MEMORY[1].X,E.MEMORY[1].Y,E.MEMORY[1].Z,E.MEMORY[2].X,E.MEMORY[2].Y,E.MEMORY[2].Z,CutVec=0,0,1,DirVec=DIR_VECT.I,DIR_VECT.J,DIR_VECT.K
InitVec=INI_VECT.I,INI_VECT.J,INI_VECT.K,EndVec=0,0,1,THICKNESS=0
FILTER/DISTANCE, MAX_STEP
EXEC MODE=RELEARN
BOUNDARY/SPHERE,E.MEMORY[2].X,E.MEMORY[2].Y,E.MEMORY[2].Z,EndVec=0,0,1,TRSH,1
HITTYPE/VECTOR
NOMS MODE=MASTER
ENDSCAN
ENDMEAS/
CS1   =CALLSUB/TFSCAN_WRITE, TUTORSUB:"SCAN1.SCN", {SCN2}, "REWRITE", ,
```

The command **TFSCAN ("filename", pt1, pt2, dir, mins, maxs, comp) APPEND n** executes the scanning of a polygon from the number of the starting scanned line n. If the n values is omitted, the default value is used; n=1. To effect the scanning of a closed line the value of pt2 is omitted.

TFSCAN("SCAN1.SCN,MEMORY[1],,VECT_VAR,1.0,2.0,TRUE,5.0)APPEND 200 translates to →

```
ASSIGN/MIN_STEP = 1.0
ASSIGN/DIR_VECT = UNIT(VECT_VAR)
ASSIGN/TRSH = 5.0
ASSIGN/COMP_ONOFF = "NO"
PNT4    =FEAT/POINT,RECT
THEO/0,0,0,0,0,1
ACTL/10,10,10,0,0,1
READPOINT/
ASSIGN/INI_VECT = UNIT(PNT4.XYZ-E.MEMORY[1].XYZ)
SCN3    =FEAT/SCAN, LINEARCLOSE, SHOWHITS=YES, SHOWALLPARAMS=YES
EXEC MODE=NORMAL, NOMS MODE=MASTER, CLEARPLANE=NO, SINGLEPOINT=NO, THICKNESS=0
FINDNOMS=2, SELECTEDONLY=NO, USEBESTFIT=NO, PROBECOMP=NO, AUTO
MOVE=NO,DISTANCE=0
DIR1=LINE, INCR=2.54
HITTYPE=VECTOR
INITVEC=0,0,0
DIRVEC=0,0,0
CUTVEC=0,0,1
PLANEVEC=0,0,0
```

```

POINT1=0,0,0
MEAS/SCAN
BASICSCAN/LINE, SHOWHITS=YES, SHOWALLPARAMS=YES
E.MEMORY[1].X,E.MEMORY[1].Y,E.MEMORY[1].Z,E.MEMORY[1].X,E.MEMORY[1].Y,E.MEMORY[1].Z,CutVec=0,0,1,DirVec=DIR_VECT.I,DIR_VECT.J,DIR_VECT.K
InitVec=INI_VECT.I,INI_VECT.J,INI_VECT.K,EndVec=0,0,1,THICKNESS=0
FILTER/DISTANCE,MAX_STEP
EXEC MODE=NORMAL
BOUNDARY/SPHERE,E.MEMORY[1].X,E.MEMORY[1].Y,E.MEMORY[1].Z,EndVec=0,0,1,TRSH,1
HITTYPE/VECTOR
NOMS MODE=MASTER
ENDSCAN
ENDMEAS/
CS2 =CALLSUB/TFSCAN_WRITE, TUTORSUB:"SCAN1.SCN", {SCN3}, "APPEND", 200,

```



If **comp** is a variable the COMP_ONOFF assignment becomes:

```

IF/ E.COMP == 0
ASSIGN/ E.COMP_ONOFF = "NO"
END_IF/
ELSE/
ASSIGN/ E.COMP_ONOFF = "YES"
END_ELSE/

```

In the TFSCAN translation the minimum step is ignored and only .SCN file format is supported.

Flow Control

Unconditional

The label definition:

L_NAME: translates to →

L_NAME= /LABEL

The command

JUMP L_NAME translates to →

GOTO/L_NAME

Conditional

The following conditional expressions that are supported by Tutor for Windows are:

Tutor Translator

```

1. If log_exprs THEN
    . . . . (Commands) . . .
    END_IF

2. If log_exprs THEN
    . . . . (Commands) . . .
    ELSE
    . . . . (Commands) . . .
    END_IF

3. If log_exprs THEN
    . . . . (Commands) . . .
    ELSIF log_exprs THEN
    . . . . (Commands) . . .
    END_IF

4. If log_exprs THEN
    . . . . (Commands) . . .
    ELSIF log_exprs THEN
    . . . . (Commands) . . .
    ELSE
    . . . . (Commands) . . .
    END_IF

```

Where **log_exprs** is a logical expression that can be True or False (see the “[Relational and Logical Operators](#)” chapter).

The Following table shows the translation from Tutor to PC-DMIS

TUTOR Command	PC-DMIS Equivalent
IF V1 EQ 1 THEN (Commands 1)\ END_IF	IF/E.V1 == 1 (Commands 1) END_ELSE/
IF V1 EQ 1 THEN (Commands 1) ELSE (Commands 2) END_IF	IF/E.V1 == 1 (Commands 1) END_IF/ ELSE/ (Commands 2) END_ELSE/
IF V1 EQ 1 THEN (Commands 1) ELSIF V2 EQ (V1-1.29) THEN	IF/E.V1 == 1 (Commands 1) END_IF/ ELSE_IF/E.V2==E.V1-1.29 (Commands 2)

(Commands 2)	
. . . .	END_ELSE_IF/	
END_IF		
IF V1 EQ 1	IF/E.V1 == 1	
THEN	
. . . .	(Commands 1)	
(Commands 1)	
. . . .	END_IF/	
ELSIF V2 EQ (V1-1.29)	ELSE_IF/E.V2==E.V1-1.29	
THEN	
. . . .	(Commands 2)	
(Commands 2)	
. . . .	END_ELSE_IF/	
ELSE	ELSE/	
.	
(Commands 3)	(Commands 3)	
.	
END_IF	END_ELSE/	

Logical Functions

Since Tutor logical functions don't have corresponding functions in PC-DMIS they are translated as a boolean variable evaluation. This variable is used in place of the logical function in expression evaluations. This means:

1. If a logical function is encountered in a logical expression its translation is inserted before the command that evaluates the expression.
2. If the same logical function is used more than one time in an expression the boolean variable name (starting from the second one) is enumerated adding an underscore plus the next number in the sequence.

EOF Functions

The EOF intrinsic function is not available in PC-DMIS.

The logical function

EOF(F0) translates to →

```
FILE/SAVE_POSITION,E.F0.PTR
EOFTEST = FILE/READ_CHARACTER,E.F0.PTR
ASSIGN/E.EOF_F0 = INT(ORD(EOFTEST) == 65535)
FILE/RECALL_POSITION,E.F0.PTR
IF EOF(E.F0) THEN
. . .
. . .
```

Tutor Translator

END_IF translates to →

```
FILE/SAVE_POSITION,E.F0.PTR  
EOFTEST = FILE/READ_CHARACTER,E.F0.PTR  
ASSIGN/EOF_F0 = INT(ORD(EOFTEST) == 65535)  
FILE/RECALL_POSITION,E.F0.PTR  
IF/EOF_F0  
. . .  
. . .  
END_IF/
```

BOOL_VAR =EOF(E.F0) translates to →

```
FILE/SAVE_POSITION,E.F0.PTR  
EOFTEST = FILE/READ_CHARACTER,E.F0.PTR  
ASSIGN/EOF_F0 = INT(ORD(EOFTEST) == 65535)  
FILE/RECALL_POSITION,E.F0.PTR  
ASSIGN/E.BOOL_VAR = EOF_F0
```



For file F1, F2 and F3, use respectively the EOF_F1, EOF_F2 and EOF_F3 variables and E.F1PTR, E.F2PTR and E.F3PTR.

EOL Functions

The EOL intrinsic function is not available in PC-DMIS.

The logical function

EOL(F0) translates to →

```
FILE/SAVE_POSITION,E.F0.PTR  
EOLTEST = FILE/READ_CHARACTER,E.F0.PTR  
ASSIGN/EOL_F0 = INT(ORD(EOLTEST) == 10)  
FILE/RECALL_POSITION,E.F0.PTR  
IF EOL(E.F0) THEN  
. . .  
. . .
```

END_IF translates to →

```
FILE/SAVE_POSITION,E.F0.PTR  
EOFTEST = FILE/READ_CHARACTER,E.F0.PTR  
ASSIGN/EOL_F0 = INT(ORD(EOFTEST) == 65535)  
FILE/RECALL_POSITION,E.F0.PTR  
IF/EOL_F0  
. . .  
. . .  
END_IF/
```



For file F1, F2 and F3, use respectively the EOF_F1, EOF_F2 and EOF_F3 variables and E.F1PTR, E.F2PTR and E.F3PTR.

If EOF (or EOL) intrinsic function is used in an expression the EOF_E.F0 (or EOL_E.F0) variable is read before expression translation.

IF EOF(F0) OR EOL(F1) THEN

....

....

END_IF translates to →

```
FILE/SAVE_POSITION,E.F0.PTR
EOFTEST = FILE/READ_CHARACTER,E.F0.PTR
ASSIGN/EOF_F0 = INT(ORD(EOFTEST) == 65535)
FILE/RECALL_POSITION,E.F0.PTR
FILE/SAVE_POSITION,E.F1PTR
EOLTEST = FILE/READ_CHARACTER,E.F0.PTR
ASSIGN/EOL_F1 = INT(ORD(EOLTEST) == 10)
FILE/RECALL_POSITION,E.F1PTR
IF/EOF_F0 OR EOL_F1
.....
.....
END_IF/
```

FOUND Functions

The intrinsic function FOUND is not available on PC-DMIS.

The following translations are logical functions:

FOUND(F0,"GOOFY") translates to →

```
ASSIGN/SEARCH_STR = "GOOFY"
ASSIGN/SIZE_F0 = LEN(SEARCH_STR)
ASSIGN/FOUND_F0 = 0
DO/
FILE/SAVE_POSITION,E.F0.PTR
STS_F0 = FILE/READLINE,E.F0.PTR,{LINE_F0}
IF/STS_F0<>"EOF"
ASSIGN/FOUND_F0 = INDEX(LINE_F0,SEARCH_STR)
IF/FOUND_F0 > 0
FILE/RECALL_POSITION,E.F0.PTR
LINE_F0 = FILE/READ_BLOCK,E.F0.PTR,FOUND_F0+SIZE_F0-1
END_IF/
END_IF/
UNTIL/(STS_F0=="EOF") OR (FOUND_F0 <> 0)
```

Tutor Translator

IF FOUND(F0,"GOOFY") THEN

.....

.....

END_IF translates to →

.....Same as *FOUND(F0,"GOOFY")* function plus the following lines are added:

```
IF/FOUND_F0 <> 0
.
.
.
.
END_IF/
```

BOOL_VAR = FOUND(F0,"GOOFY") translates to →

.....Same as *FOUND(F0,"GOOFY")* function plus the following line is added:

```
ASSIGN/BOOL_VAR=FOUND_E.F0
```

FOUND(F0,STR_VAL) translates to →

```
ASSIGN/SEARCH_STR = STR_VAL
ASSIGN/SIZE_F0 = LEN(SEARCH_STR)
ASSIGN/FOUND_F0 = 0
DO/
FILE/SAVE_POSITION,E.F0.PTR
STS_F0 =FILE/READLINE,E.F0.PTR,{LINE_F0}
IF/STS_F0<>"EOF"
ASSIGN/FOUND_F0 = INDEX(LINE_F0,SEARCH_STR)
IF/FOUND_F0 > 0
FILE/RECALL_POSITION,E.F0.PTR
LINE_F0 =FILE/READ_BLOCK,E.F0.PTR,FOUND_F0+SIZE_F0-1
END_IF/
END_IF/
UNTIL/(STS_F0=="EOF") OR (FOUND_F0 <> 0)
```

IF FOUND(F0,STR_VAR) THEN

....
....

END_IF translates to →

.....Same as *FOUND(F0,STR_VAL)* function plus the following lines are added:

```
IF/FOUND_F0 <> 0
.
.
.
END_IF/
```

BOOL_VAR = FOUND(F0,STR_VAL) translates to →

.....Same as *FOUND(F0,STR_VAL)* function plus the following line is added:

```
ASSIGN/BOOL_VAR=FOUND_E.F0
```

FOUND(F0,"GOOFY":3) translates to →

```
ASSIGN/SEARCH_STR = "GOOFY"
ASSIGN/SIZE_F0 = 3
ASSIGN/FOUND_F0 = 0
DO/
FILE/SAVE_POSITION,E.F0.PTR
STS_F0      =FILE/READLINE,E.F0.PTR,{LINE_F0}
IF/STS_F0<>"EOF"
ASSIGN/FOUND_F0 = INDEX(LINE_F0,SEARCH_STR)
IF/FOUND_F0 > 0
FILE/RECALL_POSITION,E.F0.PTR
LINE_F0      =FILE/READ_BLOCK,E.F0.PTR,FOUND_F0+SIZE_F0-1
END_IF/
END_IF/
UNTIL/(STS_F0=="EOF") OR (FOUND_F0 <> 0)
```

Tutor Translator

IF FOUND(F0,"GOOFY":3) THEN

....
....

END_IF translates to →

.....Same as *FOUND(F0,"GOOFY":3)* function plus the following lines are added:

```
IF/FOUND_F0 <> 0  
.  
. . . .  
. . . .  
END_IF /
```

BOOL_VAR = FOUND(F0,"GOOFY") translates to →

.....Same as *FOUND(F0,"GOOFY":3)* function plus the following line is added:

```
ASSIGN/BOOL_VAR=FOUND_E.F0
```



For file F1 use the SIZE_F1, FOUND_F1,STS_F1, LINE_F1 and E.F1PTR. If FOUND intrinsic function is used in an expression the FOUND_Fx variable is evaluated before expression translation (x value is 0,1, 2, or 3)

ENCODE Functions

The variables that are used as part of the ENCODE functions are defined as follows:

- INT_VAR is an Integer variable.
- IND is an integer variable.
- REAL_VAR is a real variable with format info (total and decimal digit)
- BOOL_VAR is a Boolean variable.



The translation of the ENCODE command never fails.

The logical function

ENCODE(STR_VAR,INT_VAR, IND) translates to →

```
ASSIGN/ENCODE_STR = STR(E.INT_VAR)  
ASSIGN/ENCODE_LEN = LEN(ENCODE_STR)  
ASSIGN/LENCODE_STR = LEFT(E.STR_VAR, E.IND-1)  
ASSIGN/ E.IND = E.IND+ENCODE_LEN  
ASSIGN/RENCODE_STR = MID(E.STR_VAR, E.IND-1)  
ASSIGN/ E.STR_VAR = STR(LENCODE_STR)+ STR(ENCODE_STR) + STR(RENCODE_STR)  
ASSIGN/ENCODE = 1
```

IF ENCODE(STR_VAR,INT_VAR, IND) THEN

.....
.....

END_IF translates to →

.....Same as *ENCODE(STR_VAR, INT_VAR,IND)* function plus the following lines are added:

```
IF/ENCODE == 1
.
.
.
END_IF/
```

BOOL_VAR = ENCODE(STR_VAR,INT_VAR,IND) translates to →

.....Same as *ENCODE(STR_VAR, INT_VAR,IND)* function plus the following line is added:

```
ASSIGN/ E.BOOL_VAR =ENCODE
```

The logical function

ENCODE(STR_VAR,REAL_VAR:11:4,IND) translates to →

```
ASSIGN/ENCODE_STR = FORMAT( "%11.4f" , E.REAL_VAR)
ASSIGN/ENCODE_LEN = LEN(ENCODE_STR)
ASSIGN/LENCODE_STR = LEFT(E.STR_VAR, E.IND-1)
ASSIGN/ E.IND = E.IND+ENCODE_LEN
ASSIGN/RENCODE_STR = MID(E.STR_VAR, E.IND-1)
ASSIGN/ E.STR_VAR = LENCODE_STR + ENCODE_STR + RENCODE_STR
ASSIGN/ENCODE = 1
```

IF ENCODE(STR_VAR,REAL_VAR:11:4,IND) THEN

.....
.....

END_IF translates to →

.....Same as *ENCODE(STR_VAR,REAL_VAR:11:4,IND)* function plus the following lines are added:

```
IF/ENCODE == 1
.
.
.
END_IF/
```

Tutor Translator

BOOL_VAR= ENCODE(STR_VAR,REAL_VAR:11:4,IND) translates to →

.....Same as *ENCODE(STR_VAR,REAL_VAR:11:4,IND)* function plus the following line is added:

ASSIGN/ E.BOOL_VAR=ENCODE

The logical function

ENCODE(STR_VAR,BOOL_VAR,IND) translates to →

```
ASSIGN/ENCODE_STR = E.BOOLVAL(E.BOOL_VAR)
ASSIGN/ENCODE_LEN = LEN(ENCODE_STR)
ASSIGN/LENCODE_STR = LEFT(E.STR_VAR, E.IND-1)
ASSIGN/ E.IND = E.IND+ENCODE_LEN
ASSIGN/RENCODE_STR = MID(E.STR_VAR, E.IND -1)
ASSIGN/ E.STR_VAR = STR(LENCODE_STR)+ STR(ENCODE_STR)+ STR(RENCODE_STR)
ASSIGN/ENCODE = 1
```

IF ENCODE(STR_VAR,BOOL_VAR,IND) THEN

· · ·
· · ·

END_IF translates to →

.....Same as *ENCODE(STR_VAR,BOOL_VAR,IND)* function plus the following lines are added:

```
IF/ENCODE == 1
· · · ·
· · · ·
END_IF/
```

BOOL_VAR1=ENCODE(STR_VAR,BOOL_VAR,INDEX) translates to →

.....Same as *ENCODE(STR_VAR,BOOL_VAR,IND)* function plus the following line is added:

ASSIGN/ E.BOOL_VAR=ENCODE

The logical function

ENCODE(STR_VAR,INT_VAR) translates to →

```
ASSIGN/ENCODE_STR = STR(E.INT_VAR)
ASSIGN/ENCODE_LEN = LEN(ENCODE_STR)
ASSIGN/LENCODE_STR = LEFT(E.STR_VAR,0)
ASSIGN/RENCODE_STR = MID(E.STR_VAR,ENCODE_LEN)
ASSIGN/ E.STR_VAR = LENCODE_STR + ENCODE_STR + RENCODE_STR
ASSIGN/ENCODE = 1
```

IF ENCODE(STR_VAR,INT_VAR) THEN

....
....

END_IF translates to →

.....Same as *ENCODE(STR_VAR,INT_VAR)* function plus the following lines are added:

```
IF/ENCODE == 1
.
.
.
END_IF/
```

BOOLVAR=ENCODE(STR_VAR,INT_VAR) translates to →

.....Same as *ENCODE(STR_VAR,INT_VAR)* function plus the following line is added:

```
ASSIGN/ E.BOOL_VAR=ENCODE
```

The logical function

ENCODE(STR_VAR, REAL_VAR:11:4) translates to →

```
ASSIGN/ENCODE_STR = FORMAT("%11.4f", E.REAL_VAR)
ASSIGN/ENCODE_LEN = LEN(ENCODE_STR)
ASSIGN/LENCODE_STR = LEFT(E.STR_VAR,0)
ASSIGN/RENCODE_STR = MID(E.STR_VAR,ENCODE_LEN)
ASSIGN/ E.STR_VAR = LENCODE_STR + ENCODE_STR + RENCODE_STR
ASSIGN/ENCODE = 1
```

Tutor Translator

IF ENCODE(STR_VAR, REAL_VAR:11:4) THEN

....
....

END_IF translates to →

.....Same as *ENCODE(STR_VAR,REAL_VAR:11:4)* function plus the following lines are added:

```
IF/ENCODE == 1  
....  
....  
END_IF/
```

BOOL_VAR=ENCODE(STR_VAR, REAL_VAR:11:4) translates to →

.....Same as *ENCODE(STR_VAR,REAL_VAR:11:4)* function plus the following line is added:

```
ASSIGN/ E.REAL_VAR=ENCODE
```

The logical function

ENCODE(STR_VAR, BOOL_VAR) translates to →

```
ASSIGN/ENCODE_STR = E.BOOLVAL(E.BOOL_VAR)  
ASSIGN/ENCODE_LEN = LEN(ENCODE_STR)  
ASSIGN/LENCODE_STR = LEFT(E.STR_VAR,0)  
ASSIGN/RENCODE_STR = MID(E.STR_VAR,ENCODE_LEN)  
ASSIGN/ E.STR_VAR = LENCODE_STR + ENCODE_STR + RENCODE_STR  
ASSIGN/ENCODE = 1\
```

IF ENCODE(STR_VAR, BOOL_VAR) THEN

....
....

END_IF translates to →

.....Same as *ENCODE(STR_VAR,BOOL_VAR)* function plus the following lines are added:

```
IF/ENCODE == 1  
....  
....  
END_IF/
```

ASSIGN/BOOL_VAR1=ENCODE(STR_VAR,BOOL_VAR) translates to →

.....Same as ENCODE(STR_VAR,BOOL_VAR) function plus the following line is added:

ASSIGN/ E.BOOL VAR1=ENCODE

DECODE Functions

The variables that are used as part of the ENCODE functions are defined as follows:

- INT_VAR is an Integer variable.
 - IND is an integer variable.
 - REAL_VAR is a real variable with format info (total and decimal digit)
 - BOOL_VAR is a Boolean variable.

The logical function

DECODE(STR_VAR, INT_VAR,IND) translates to →

```
ASSIGN/DECODE_STR = MID(E.STR_VAR, E.IND-1)
ASSIGN/ E.INT_VAR = INT(DECODE_STR)
DO/
ASSIGN/INDX = STRPBRK(DECODE_STR, " ,;.")
ASSIGN/ E.IND = E.IND + INDX
ASSIGN/DECODE_STR = MID(E.STR_VAR, E.IND-1)
UNTIL/INDX <> 1
ASSIGN/DECODE = 1
```

IF DECODE(STR_VAR, INT_VAR,IND) THEN

1

END IF translates to →

.....Same as DECODE(STR VAR,INT VAR,IND) function plus the following lines are added:

```
IF/DECODE == 1  
.  
.  
.  
.  
.  
.  
END IF/
```

BOOL VAR=DECODE(STR VAR, INT VAR,IND) translates to →

.....Same as DECODE(STR, VAR, INT, VAR, IND) function plus the following line is added:

ASSIGN/ E-BOOT: VAR=DECODE

Tutor Translator

The logical function

DECODE(STR_VAR, REAL_VAR,IND) translates to →

```
ASSIGN/DECODE_STR = MID(E.STR_VAR, E.IND-1)
ASSIGN/ E.REAL_VAR = DOUBLE(DECODE_STR)
DO/
ASSIGN/INDX = STRPBRK(DECODE_STR, " ,;:")
ASSIGN/ E.IND = E.IND + INDX
ASSIGN/DECODE_STR = MID(E.STR_VAR, E.IND-1)
UNTIL/INDX <> 1
ASSIGN/DECODE = 1
```

IF DECODE(STR_VAR, REAL_VAR,IND) THEN

....

....

END_IF translates to →

.....Same as *DECODE(STR_VAR,REAL_VAR,IND)* function plus the following lines are added:

```
IF/DECODE == 1
. . .
. . .
END_IF/
```

BOOL_VAR=DECODE(STR_VAR, REAL_VAR,IND) translates to →

.....Same as *DECODE(STR_VAR,REAL_VAR,IND)* function plus the following line is added:

```
ASSIGN/ E.BOOL_VAR=DECODE
```

The logical function

DECODE(STR_VAR, BOOL_VAR,IND) translates to →

```
ASSIGN/DECODE_STR = MID(E.STR_VAR, E.IND-1)
ASSIGN/ E.BOOL_VAR = DECODE_BOOL(DECODE_STR)
IF/ E.BOOL_VAR < 0
ASSIGN/DECODE = 0
END_IF/
ELSE/
ASSIGN/DECODE = 1
IF/ E.BOOL_VAR == 0
ASSIGN/ E.IND = E.IND+ INDEX(DECODE_STR, "FALSE") +4
```

```

END_IF/
ELSE/
ASSIGN/ E.IND = E.IND+ INDEX(DECODE_STR, "TRUE") +3
END_ELSE/
END_ELSE/

```

IF DECODE(STR_VAR, BOOL_VAR,IND) THEN

....

....

END_IF translates to →

.....Same as *DECODE(STR_VAR,BOOL_VAR,IND)* function plus the following lines are added:

```

IF/DECODE == 1
. . .
. . .
END_IF/

```

BOOL_VAR=DECODE(STR_VAR, BOOL_VAR,IND) translates to →

.....Same as *DECODE(STR_VAR,BOOL_VAR,IND)* function plus the following line is added:

```
ASSIGN/ E.BOOL_VAR=DECODE
```

The logical function

DECODE(STR_VAR,INT_VAR) translates to →

```

ASSIGN/ E.INT_VAR = INT(E.STR_VAR)
ASSIGN/DECODE = 1

```

Tutor Translator

IF DECODE(STR_VAR, INT_VAR) THEN

....
....

END_IF translates to →

.....Same as *DECODE(STR_VAR,INT_VAR)* function plus the following lines are added:

```
IF/DECODE == 1  
....  
....  
END_IF/
```

BOOL_VAR=DECODE(STR_VAR,INT_VAR) translates to →

.....Same as *DECODE(STR_VAR,INT_VAR)* function plus the following line is added:

```
ASSIGN/ E.BOOL_VAR=DECODE
```

The logical function

DECODE(STR_VAR, REAL_VAR) translates to →

```
ASSIGN/ E.REAL_VAR = DOUBLE(E.STR_VAR)  
ASSIGN/DECODE = 1
```

IF DECODE(STR_VAR, REAL_VAR) THEN

....
....

END_IF translates to →

.....Same as *DECODE(STR_VAR,REAL_VAR)* function plus the following lines are added:

```
IF/DECODE == 1  
....  
....  
END_IF/
```

ASSIGN/BOOL_VAR=DECODE(STR_VAR, REAL_VAR) translates to →

.....Same as DECODE(STR_VAR,REAL_VAR) function plus the following line is added:

ASSIGN/ E.BOOL_VAR=DECODE

The logical function

DECODE(STR_VAR, BOOL_VAR) translates to →

```
ASSIGN/ E.BOOL_VAR = DECODE_BOOL(E.STR_VAR)
IF/ E.BOOL_VAR < 0
ASSIGN/ E.BOOL_VAR = 0
ASSIGN/DECODE = 0
END_IF/
ELSE/
ASSIGN/DECODE = 1
END_ELSE/
```

IF DECODE(STR_VAR, BOOL_VAR) THEN

— 1 —

FND_IF translates to →

.....Same as DECODE(STR VAR,BOOL VAR) function plus the following lines are added:

```
IF/DECODE == 1  
.  
.  
.  
.  
.  
.  
.  
.  
END_IF/
```

BOOL_VAR1=DECODE(STR_VAR, BOOL_VAR) translates to →

.....Same as DECODE(STR_VAR,BOOL_VAR) function plus the following line is added:

ASSIGN/ E.BOOL_VAR1=DECODE

Tutor Translator

IF DECODE(STR_VAR,INT_VAR) OR DECODE(STR_VAR, BOOL_VAR) THEN

....

....

END_IF translates to →

```
ASSIGN/ E.INT_VAR = INT(E.STR_VAR)
ASSIGN/DECODE = 1
ASSIGN/ E.BOOL_VAR = E.DECODE_BOOL(E.STR_VAR)
IF/ E.BOOL_VAR < 0
ASSIGN/ E.BOOL_VAR = 0
ASSIGN/DECODE_2 = 0
END_IF/
ELSE/
ASSIGN/DECODE_2 = 1
END_ELSE/

IF /DECODE==1 OR DECODE_2==1
. . .
. . .
END_IF/
```

OOT Function

IF OOT THEN

....

....

END_IF translates to →

```
IF / E.OFMT.OOT== 1
. . .
. . .
END_IF/
```

CRITICAL Function

The CRITICAL function does not have effect

IF CRITICAL THEN

....

....

END_IF translates to →

```
IF / E.OFMT.CRIT== 1
. . .
. . .
END_IF/
```

BADMEAS Function

The Tutor BADMEAS function is simulated with a variable that is set by the ONERROR command. The BADMEAS variable is initialized to at the beginning of a part program and before each measure (Mxx) command.

IF BADMEAS THEN

....

....

END_IF translates to →

```
IF / E.BADMEAS <> 0
. . .
. . .
END_IF/
```

Emergency Control

NO_EME_BDMS

NO_EME_BDMS translates to →

```
ASSIGN/ E.BADMEAS = 0
ONERROR/PROBE_MISS,SET E.BADMEAS
```

EME_BDMS

EME_BDMS translates to →

```
ASSIGN/ E.BADMEAS = 0  
ONERROR/PROBE_MISS,OFF
```

Repetition of Commands

FOR Commands

The Tutor FOR syntax is:

```
FOR CONTROL_VAR=VAL1 TO VAL2 [By VAL3]  
. . . .  
. . . .  
END_FOR
```

Where:

- VAL1 is the start integer value. It could be a variable.
- VAL2 is the stop integer value. It could be a variable.
- VAL3 is the optional integer step value. It could be a variable.

Translation depends on type of VAL3:

- Val3 is an immediate value or missing.
- Val3 is an integer variable

For each of FOR repetition commands the following is true:

- The name of the control variable becomes FOR_name
- The label END_FORx is used for EXIT and EXIF command translation. x is the progressive FOR command ID.

Commands where Val3 is an immediate value or missing.**FOR I=0 TO 10**

....

....

END_FOR translates to →*where I is the control variable and is an integer.*

```

ASSIGN/FOR_I = 0
WHILE/FOR_I <= 10
. . .
. . .
ASSIGN/FOR_I = FOR_I+1
END WHILE/
END_FOR1 =LABEL/

```

FOR I=100 TO 10

....

....

END_FOR translates to →*where I is the control variable and is an integer.*

```

ASSIGN/FOR_I = 100
WHILE/FOR_I >= 10
. . .
. . .
ASSIGN/FOR_I = FOR_I-1
END WHILE/
END_FOR2 =LABEL/

```

Tutor Translator

FOR I=0 TO 10 BY 2

....

....

END_FOR translates to →

where *I* is the control variable and is an integer.

```
ASSIGN/FOR_I = 0
WHILE/FOR_I <= 10
. . .
. . .
ASSIGN/FOR_I = FOR_I+2
END WHILE/
END FOR3 =LABEL/
```

FOR I=100 TO 10 BY 10

....

....

END_FOR translates to →

where *I* is the control variable and is an integer.

```
ASSIGN/FOR_I = 100
WHILE/FOR_I >= 10
. . .
. . .
ASSIGN/FOR_I = FOR_I-10
END WHILE/
END FOR4 =LABEL/
```

FOR J=VAL1 TO VAL2

....

....

END_FOR translates to →

```
ASSIGN/FOR_J = VAL1
WHILE/FOR_J <= VAL2
. . .
. . .
ASSIGN/FOR_J = FOR_J+1
END WHILE/
END FOR5 =LABEL/
```

Command where Val3 is an integer variable.

FOR J=VAL1 TO VAL2 BY VAL3

....

....

END_FOR translates to →

```
ASSIGN/FOR_J = VAL1
ASSIGN/CONTINUE_I = 1
WHILE/(( E.VAR3>0)AND(FOR_J<= E.VAR2))OR((E.VAR3<0)AND(FOR_J>= E.VAR2))
. . .
. . .
ASSIGN/FOR_J = FOR_J+ E.VAR3
END FOR6 =LABEL/
END WHILE/
```

LOOP Command

The label END_LOOPx is used for EXIT and EXIF command translation. x is the progressive LOOP command. ID

LOOP

....

....

END_LOOP translates to →

```
DO/  
. . . .  
. . . .  
UNTIL/0  
END_LOOP1 =LABEL/
```

EXIT Command

The exit command terminates the [FOR](#) and [LOOP](#) cycles.

Terminate a FOR cycle.

EXIT translates to →

GOTO/END_FORx

where the x is the ID of current FOR command.

Terminate a LOOP cycle.

EXIT translates to →

GOTO/END_LOOPx

where the x is the ID of current LOOP command.

EXIF Command

The exit command conditionally terminates the [FOR](#) and [LOOP](#) cycles.

Terminate a FOR cycle.

EXIF VALUE EQ 3 translates to →

```
IF/ E.VALUE==3
GOTO/END_FORx
END_IF/
```

where the x is the ID of current FOR command.

Terminate a LOOP cycle.

EXIF VALUE EQ 3 translates to →

```
IF/ E.VALUE==3
GOTO/END_LOOPx
END_IF/
```

where the x is the ID of current LOOP command.

Internal Procedure Call

PICKPT (12,{ 1.32,3.42,564.746},res) translates to →

Where *PICKPT* is a procedure (procedure *PICKPT* (integer A;vector B,C)) and *res* is a vector variable.

```
CS1 =CALLSUB/PICKPT,:E,12,MPOINT(1.32,3.42,564.746),E.RES,,
```

External Procedure Call

The Tutor for windows external procedure call command allows two formats:

1. With explicit external module name:
PROC_NAME ext "MODULE_NAME.TEC"
2. With implicit external module name:
PROC_NAME ext
In this case the name of external module is the last used in explicit procedure call or the used with LOAD (EXT PROC "MODULENAME.TEC") command.

LOAD (EXT PROC "DEA_PROC.TEC") translates to →

```
ASSIGN/E.EXTPROC = "DEA_PROC.PRG"
```

Tutor Translator

SP_5(33,{12.,23.,34.},{12.864345,10.,20.}, "x") ext "c:\wTutor\procedur\dea_proc.tec" translates to →

```
ASSIGN/E.EXTPROC = "DEA_PROC.PRG"
CS2 =CALLSUB/SP_5, E.EXTPROC:E, 33, MPOINT(120.,23.0,34.0)
,MPOINT(12.864345,10.0,20.0), "X",,
```

SP_5(IntVar,VectorVar1,VectorVar2,StringVar) ext translates to →

```
=CALLSUB/SP_5,E.EXTPROC:E, E.INTVAR, E.VECTORVAR1, E.VECTORVAR2, E.STRINGVAR,,
```

Procedures

Tutor procedures are translated as subroutines in PC-DMIS. The main problem is due to the fact that Tutor procedure can directly address the global variables, while PC-DMIS does not. To let PC-DMIS to access global variables is necessary that they are part of a global structured variable (environment) passed as parameter each procedure call.

Global variables for Procedures are defined as:

1. All variables defined by Tutor part program including memory arrays, E.WM1, E.WM2.
2. All auxiliary PC-DMIS variables used to emulate Tutor functionality. For variables used in the ENCODE, DECODE, FOR and LOOP translation does not need to be modified
3. All FUNCTION variables:
E.OFF2ROT, E.BIGGEST, E.CALC_AXY, E.CALC_AYZ, E.CALC_AZY, E.BOOLVAL,
E.PDMS2DEG, E.DMS2DEG

⚠ Warning: Tutor procedures that changes current PCDMIS settings like current alignment or current probe doesn't work on current PCDMIS version

Procedure Variables

Variables name used by procedure are translated in the following ways:

1. **Global variable:** the “E.” prefix is added: “GLOBVAR” → “E.GLOBVAR”
2. **Formal parameters:** a “_” tail is added to the name: “GOOFY” → “GOOFY_”
3. **Local variables:** a “_” tail is added to the name: “LOCVAR” → “LOCVAR_”

Local variables declaration follows the same rule of global variables. See the section on [“Variable Declaration”](#) for more information.

Procedure Declaration

The procedure declaration

procedure PICKPT (integer A;vector B,C)" translates to →

```
SUBROUTINE/PICKPT,
E = : ENVIRONMENT,
SID = CALLED_ID
A_ = : INTEGER,
B_ = : VECTOR,
C_ = : VECTOR,
```



In the description field of each formal parameter there is the variable type.

END Procedure Declaration

END_PROCEDURE translates to →

ENDSUB/

Core Functions

These basic functions are part of the Tutor translator and help to accomplish some of the tasks of translating Tutor commands.

```
ASSIGN/ E.OFF2ROT = FUNCTION( (X,Y) , IF(X==0,IF(Y>0,90,90),RAD2DEG(ATAN(Y/X))) )
```

BIGGEST

```
ASSIGN/ E.BIGGEST =
FUNCTION((F),IF(ABS(F.I)>ABS(F.J),IF(ABS(F.K)>ABS(F.I),IF(F.K>0,E.ZPLUS,E.ZMINUS),
IF(F.I>0,E.XPLUS,E.XMINUS)),IF(ABS(F.K)>ABS(F.J),IF(F.K>0,E.ZPLUS,E.ZMINUS),
IF(F.J>0,E.YPLUS,E.YMINUS))))CALC_AXY
ASSIGN/ E.CALC_AXY = FUNCTION((V1,V2),ANGLEBETWEEN(UNIT(MPOINT(V1.I,V1.J,0)),
UNIT(MPOINT(V2.I,V2.J,0))))
```

CALC_AYZ

```
ASSIGN/ E.CALC_AYZ = FUNCTION((V1,V2),ANGLEBETWEEN(UNIT(MPOINT(0,V1.J,V1.K)),
UNIT(MPOINT(0,V2.J,V2.K))))
```

CALC_AZY

```
ASSIGN/ E.CALC_AZX = FUNCTION((V1,V2),ANGLEBETWEEN(UNIT(MPOINT(V1.I,0,V1.K)),
UNIT(MPOINT(V2.I,0,V2.K))))
```

BOOLVAL

```
ASSIGN/ E.BOOLVAL = FUNCTION((BVAL),IF(BVAL<>0,"TRUE","FALSE"))
```

PDMS2DEG

```
ASSIGN/ E.PDMS2DEG = FUNCTION((VAL),INT(VAL)+INT((VAL-INT(VAL))*100)/60+((VAL-INT(VAL))*100)-INT((VAL-INT(VAL))*100)*100/3600)
```

PDMS2DEG

```
ASSIGN/ E.DMS2DEG =
FUNCTION((VAL,ISDMS),IF((ISDMS),VAL,IF((VAL>=0),E.PDMS2DEG(VAL),-E.PDMS2DEG(-VAL))))
```

Tutor Simulation Subroutines

The following subroutines simulate Tutor functionalities for which PC-DMIS does not have direct compatibility:

- **ERASE_PROBE_SUB:** Probe 0 emulation
- **SAVE_REFSYS_SUB:** Save Refsys file emulation
- **LOAD_REFSYS_SUB:** Load Refsys file emulation.
This subroutine doesn't work on current PCDMIS version
- **THEO_INIT:** Initialize Tutor theoretical value table emulation
- **INIT_FORMAT_TBL:** Tutor output format table initialization
- **TUTOR_DEV_ENDIS:** Enable/Disable output device emulation
(Dy,NoDy,Prn/NoPrn,File/Nofile)
- **TUTOR_FORMAT:** Emulates Tutor Format command
- **TUTOR_OUTPUT:** Emulates Tutor evaluation output
- **TUTOR_HEADER:** Emulated Tutor Header command.
- **TUTOR_DY:** Emulates Tutor DY(text) command.
- **TUTOR_PRN:** Emulates Tutor DY(text) command.
- **TUTOR_FILE:** Emulates Tutor FILE n (text) command.
- **TUTOR_OPEN:** Emulates Tutor OPEN n (file) command.
- **TUTOR_CLOSE:** Emulates Tutor CLOSE n command.
- **TUTOR_TYPE:** Emulates the Tutor TYPE command
- **TFSCAN_WRITE:** Writes scanned points into a file. Only .SCN is supported
- **TUTOR_IO:** Emulates Tutor IO (SYSTEM_INFO(IO,.....))

These subroutines are available in the TutorSub.prg program supplied with the translator.

Look Ahead Procedure

The Look Ahead procedure searches for all commands that define the measured path of the element, starting from the measured command without a path. These commands are:

- Move
- Movetf
- Probe
- Dist_Approach

Look Ahead stops when:

- A measured command has been detected.
- An indirect measured command (construction) has been detected.
- A tolerance command has been detected.
- An alignment command has been detected.
- A procedure call has been detected.
- An exec command has been detected. (external application execution)

The remaining Tutor commands are divided into two groups:

The commands of the first group finish the Look Ahead procedure.

This group consists of measured commands, indirect measured tolerances, alignments, procedures, execution of external programs (exec), chain, loop, end_loop, for, end_for, if, end_if and jump.

The commands of the second group are ignored by the look-head procedure.

Problems in Developing the Tutor Translator

The main problems faced when developing the Tutor translator are:

Theoretical value management:

While PC-DMIS features contain both theoretical and actual values (X,Y,Z,I,J,K,. . .) Tutor features don't have associated theoretical values.

When Tutor needs to compare an evaluated (measure or constructed) feature, it searches in the Output format table (once for each feature type) for feature fields that are required (X,Y,Z, CX, CY,CZ , . . .). If the fields are required, then they are compared with theoretical values. If comparison is required, Tutor uses values stored in the theoretical table (a single table for all types of feature). The fields involved in the evaluation are normally filled with the appropriate values before the feature evaluation. Feature evaluation can be made immediately after the measure or deferred to a second time.

During translation there is no way to be sure to have all theoretical values of a feature. This is true for several reasons:

- Field(s) values are not set yet.
- The Field value is not meaningful for the evaluation.
- Field(s) value could be defined inside a called procedure.

The adopted solution provides that when a measurement command is encountered Tutor translator looks for a measurement Path or MoveTf (hit) commands and adds them in the PC-DMIS measurement command as HIT/BASIC.... THEO=NO.

When the Measurement command is completed the **DmisFeatCmd.CalculateNominals** function is called.

Nevertheless this method does not guarantee the accuracy of nominal values because of differences in hits and probes.

Tutor Translator

Measurement hits and move commands

The main differences in measurement hits and move commands are:

1. Tutor moveTf (hit) coordinates are the probe center ball while PC-DMIS HIT/BASIC coordinates are the measure point (on the part).
2. Tutor moveTf don't have the measurement direction.
3. Tutor measured points can be supplied in two ways:
 - a. Using a path associated to the measure. This is very close to a PC-DMIS measured command.
 - b. Executing a proper number of moveTf commands after the measured command. In this case Translator tries to find these commands and add them in the PC-DMIS measurement command. Nevertheless it's easy to find Tutor part programs that use FOR or LOOP commands to measure a feature. Because PC-DMIS does not have this capability, this kind of measurement is not translated.

Feature evaluation and output

The main differences are:

1. Theoretical management.
2. Fields that can be evaluated and output. For example, Tutor allows evaluation (compared with theoretical value) of a single component of a feature vector (Cx, Cy, Cz) while PC-DMIS location simply shows the vector with all its components (ijk).
3. Output Format.
4. Different meaning of some fields.

The adopted solution is to emulate the Tutor evaluation and output.

At begin of the part program the E.OFMT structured variable is initialized with the Tutor default output format table. The format table structure is:

1. For each Tutor supported device there are 15 feature tables, one for each Tutor feature. Serial lines that are not supported.
2. Each feature table is composed of 18 fields, one for each Tutor field (X, Y, Z, . . .). Each field can assume the following values: 0 = No output, 1= Only actual, 2= actual + error, 3=actual + error + tolerance.
3. For each Tutor supported device there is a table with the label of each field to be used for output.

The theoretical table (E.TT) is initialized at the beginning. This table is composed of 18 fields, one for each Tutor field (X, Y, Z, . . .), each field could contain the nominal value, the upper tolerance and the lower tolerance.

When a feature needs to be evaluated and sent to output the Tutor_Output subroutine is recalled. This subroutine determines which Tutor devices are enabled, checks fields that need to be evaluated and which format has to be used. The output string is prepared to send to the device, using the format function. The COMMENT/REPT command is used to emulate "Dy" and "Prn" Tutor device and the order of output columns cannot be changed.

Due to the fact that feature evaluation and output is translated as a single subroutine call it could be easy for the operator to replace it with one of PC-DMIS dimension commands and use the powerful PC-DMIS output.

Probe management

Tutor and PC-DMIS probe management are completely different.

Tutor

1. There is no theoretical probe definition. If PH9 is present the Roll and Pitch values are required.
2. When calibrating the gauge, the tool offset (distance from RAM and the tip) must be supplied. Calibration also computes the dynamic tip radius of calibration probe.
3. The qualification procedure computes the probe offset (in respect to the Calibration probe) and the tip radius.
4. Movement and measured coordinates are computed without the tool offset. This could be a problem executing DCC movement using the machine reference system (STARTUP). The Tool offset is only used to properly compensate/decompensate the coordinates and hits.
5. Probes are identified by a couple of numbers (Head and Tip).
6. Probes file management command (Load/Save).
7. Calibration/Qualification commands.
8. No automatic Calibration/Qualification procedure.

PC-DMIS

1. Theoretical Probe definition.
2. Movement and measured coordinates are computed taking care of all probe parameters.
3. Meaningful Probe Name.
4. Encapsulated Calibration/Qualification.
5. Probes file management command allows only Loading.

The adopted solution is to avoid Calibration and Qualification translation. The user is invited to use PC-DMIS probe management to create a probe file for all required probes. To relate Tutor with PC-DMIS probes the user must create a map file (.T2P = Tutor to PC-DMIS). Each line of this file contains Tutor Head and Tip number and the PC-DMIS probe name.

Because Tutor allows variables to be used for probe selection the translator creates a 2D array loaded with PC-DMIS probe names.

Limitations

The following limitations of Tutor Translator have been identified:

- Only English language is fully supported (First release).
- **Tutor procedures that changes current PCDMIS settings like current alignment or current probe doesn't work on current PCDMIS version.**
- **LOAS_REF SYS_SUB emulation subroutine doesn't work on current PCDMIS version.**

Tutor Translator

- Qualification part programs can not be translated. The operator must use PC-DMIS to define the necessary tools. Some utility application could be written in order to display the qualification data contained in the Tutor qualification file (.tip)
- The SAVE Probe File command can not be translated.
- These eight Tutor commands are not translated:
PH9 GAUGE CALIB QUALI
TDIAM TOOL_OFFSET CALIB_DIAM TIP_COMPENS
- The PROBE0 command can not be translated if the Head -Tip <=> TIP-PC-DMIS association is not made at translation time.
- All move and measurement CNC commands executed in reference system 0 (STARTUP) are effected by the calibration tool offset error and MUST be relearned. A COMMENT operator is added in the NCMOVE (MODE/DCC) translation that warns the operator when the CMM will move in CNC mode within the STARTUP alignment. This COMMENT must be removed by the operator when the translated part program was definitively tuned.
- The measured commands are not translatable if:
 1. The number of hits is unknown.
 2. There is no path associated or it is not possible to immediately individualize the move and movetf commands in a defined sequence.
- The TEACH variable is not translated because it asks for operator intervention to teach a path.
- The INCR_MODE / NO_INCR_MODE commands are not translated. Likewise, the Tutor part programs that use this convention are not translated.
- All commands that use ISO algorithm (FORM ISO keyword) and are **approximately** translated in the same way of a command that uses a Least Square algorithm.
- The ICONE, ICYL ,MCONE and MCYL commands with forced element axis options are simply translated as MCONE and MCYL.
- The MCYL 2 diameters measure command is not translated.
- The MPARAB and IPARAB are not translated.
- The MTORUS and ITORUS are not translated.
- In THEO command the ISODM field is ignored.
- Construction commands that use a scan files are not translated.
- The command to measure SLOT and SQSLOT operate in quite a different manner from Tutor so the operator must tune them.
- The command to construct SLOT is translated as an empty slot construction command.
- The command to construct a SQUARE SLOT is NOT translated.
- The angle field of a Relation Element is always the angle between the original element vector. The Min./Max. angle in Tutor is not translated.
- Intersection command. If the two elements don't intersect the intersection results are not computed and PC-DMIS displays an error message.
- Relation Distance. For Line-Point 2D, Line-Point 3D, Plane-Point 3D relation distances the PC-DMIS application point of the resulting line differs from the Tutor point.
- Geometric Tolerances with boundary elements are not translated.
- Geometric Orientation Tolerances with commands in which the checked element is a plane and the DIST is selected are not translated.
- The following Tutor Geometric Tolerance commands are not translated (these are in DOS format):
 - PARAL and PERPEN (eldat)
 - ANGOL(eldat, angle)
 - GEO_TOL (CONC, eldat)
 - GEO_TOL (COAX, eldat)
 - GEO_TOL (SYMM, eldat)

- GEO_TOL(SYMM_C, elres, eldat, elcheck, R1, R2)
 - MMC1 and LMC1 (eltyp)
 - MMC2 (eltyp, elmem, DM=value, UDM=value, PIN_CHECKED)
 - LMC2 (eltyp, elmem, DM=value, UDM=value, PIN_CHECKED).
 - The following Tutor commands for serial lines are not translated:
 - OUTPUT SLINE1 elmem
 - OUTPUT SLINE2 elmem
 - OUTPUT SLINE3 elmem
 - OUTPUT SLINE4 elmem
 - SL, OPEN_SL, CLOSE_SL, NOSL, & SL_INPUT
 - The Tutor DOS command, COMPARE is not translated.
 - The Tutor DOS command, NO_COMPARE is not translated.
 - The Tutor DOS command, PARTIAL_OUTPUT is not translated.
 - The Tutor DOS command, COMPLETE_OUTPUT is not translated.
 - The OUT_GRAPH command is not translated.
 - The output layout is fixed. Therefore, it is not possible to program Headers, Footers, Titles, Vertical layouts or Field order.
 - The output labels are always in English.
 - The SET_DIM command does not have effect on the first version of the translator even if it is already translated .
 - The OOT_COND and NO_OOT_COND commands do not have effect on the first version of the translator even if they are already translated.
 - The command INCH_MODE and METRIC_MODE that change the measurement units at runtime are not translated because PC-DMIS does not allow runtime measurement unit change.
 - The command DEG_MODE and DMS_MODE that change the angle measurement unit format at runtime are translated only as variable assignments because in PC-DMIS the angle DMS format is a Part Program attribute and cannot change at execution time. In any case this attribute applies only on output.
 - The SCALE command is not translated.
 - The LOAD(FORMAT,"file") and SAVE(FORMAT,"file" do not have effect on the first version of the translator even if they are already translated. The SAVE_FORMAT and LOAD_FORMAT subroutines will be modified in order to supply them.
 - The DEFINE_ELEMENT of 2 cylinder sections is not translated.
 - The LOAD(ELEMENT,"file") and SAVE(ELEMENT,"file" do not have effect on the first version of the translator even if they are already translated.
 - The FEED command is not translated because is not supported by PC-DMIS.
 - In PC-DMIS the measure speed is limited to 20% of the max CMM speed. In contrast Tutor allows the measure speed to be set higher than 20% (up to 100%). If the measurement speed is higher than 20 then it will be limited to 20 for PC-DMIS.
 - These four commands are not translated because PC-DMIS does not support this capability:

CHAIN	DELAY	OPT-SPEED	EMERGENCY
-------	-------	-----------	-----------
 - The command ACC is translated only as a COMMNET/DOC that explains that the PC-DMIS ACCELERATION command operate differently.
 - The Tutor COORD variable also stores the reference system active at the moment values are assigned to it. This functionality is not available in PC-DMIS.
 - The EXEC command modes are partially supported by PC-DMIS. The Tutor mode value is composed of the application execution mode and window mode.
 - The command EXEC modes are partially supported by PC-DMIS. The Tutor mode value is composed by application execution mode and window mode. PC-DMIS does not support the window mode.
- PC-DMIS supports only the following execution modes:
1. EXEC_NORMAL → NO_DISPLAY , NO_WAIT

2. EXEC_WAIT → NO_DISPLAY , WAIT

- In the TFSCAN translation the minimum step is ignored.
- In the TFSCAN translation only .SCN file format is supported.
- The translation of the ENCODE function never returns FALSE.
- The CRITICAL function does not have effect on the first version of the translator even if it is already translated.
- The following INFO_FUNC are not translated:
- INFO_FUNC(2) : probe head number.
- INFO_FUNC(3) : probe tip number.
- INFO_FUNC(9) : CMM acceleration.
- INFO_FUNC(12) : Feed percentage.
- INFO_FUNC(13) : Optimal speed.
- INFO_FUNC(14) : Qualification gauge.
- INFO_FUNC(19) : Calibration probe X offset
- INFO_FUNC(20) : Calibration probe Y offset
- INFO_FUNC(21) : Calibration probe Z offset
- INFO_FUNC(22) : Scale factor X.
- INFO_FUNC(23) : Scale factor Y.
- INFO_FUNC(24) : Scale factor Z.
- INFO_FUNC(25) : Scale factor.
- INFO_FUNC(30) through INFO_FUNC(61) Thermal sensor read.
- The PLAY command is not translated
- The following SYSTEM_INFO command are not translated:
- SYSTEM_INFO(PROBE,HEAD,TIP,RAD,OFFSET,APOS,BPOS,QUAL) WRITE
- SYSTEM_INFO (REFSYS,SYS,MROT,MINV,TRANS,I) WRITE
- SYSTEM_INFO (TP200, ON)
- SYSTEM_INFO (TP200, OFF)

Index

A

Alignment Commands	
Iskew	18
Preset.....	19
Skew1	16
Skew2	17
Use of Reference Planes	21
Use of Reference Systems.....	20
Alignment Commands.....	16
APPROACH / NO_APPROACH Commands.....	107
Architecture.....	1
ASSIGNMENT Commands	
BOOLVAR integer.....	110
INTVAR integer	109
REALVAR integer.....	109
STRINGVAR integer	110
ASSIGNMENT Commands.....	109
B	
BASIC script generator	1
BASIC script utility subroutines collection	2

C

Command Repetition	
EXIF.....	166
EXIT.....	166
FOR	162
LOOP.....	166
Command Repetition.....	162
Construction Commands	
Circle	51
Cone	57
Cylinder	59
Line.....	61
Middle Point.....	63
No PC-DMIS Equivalent	51
Paraboloid	64
Plane	64
RIDPT.....	71
Slot	68
Thickness	70
Three Level Plane.....	65
Torus	70
Construction Commands.....	50

Tutor Translator

Core Functions.....	169	Geometric Tolerances	86
D			
DELAY Command.....	108	I	
E			
Element Buffer Management	127	Internal Procedure Call.....	167
Element Comment	107	Introduction	1
Element Evaluation	108, 126	L	
EMERGENCY Command	108	Limitations	173
Emergency Control	161	Logical Functions	
Emulation subroutines	3	BADMEAS	161
External Procedure Call.....	167	CRITICAL	160
F			
Flow Control.....	143	DECODE	155
Functionality.....	1	ENCODE	150
G			
Geometric Tolerances		EOF	145
Angularity	90	EOL	146
Coaxiality.....	93	FOUND.....	147
Concentricity.....	92	OOT	160
Parallelism.....	88	Logical Functions	145
Perpendicularity.....	89	Look Ahead Procedure.....	170
Symmetry	94	M	
True Position	96	Measurement Commands	
		Canned Circle	45
		Circle	25
		Conditions.....	23
		Cone	27

Cylinder	28	Output to Screen.....	113
Ellipse.....	30	Serial Line.....	116
Line	31	Output Control.....	111
May have a PC-DMIS Equivalent	23	Output Protocol and Output Format	
Middle Point.....	33	Block Command	123
No PC-DMIS Equivalent.....	23	Dimension Mode Command	124
Paraboloid	34	Format Command.....	122
Pick	35	Input/Output Parameters	126
Plane	35	Loading and Saving	126
Slot.....	41	THEO Command	125
Sphere.....	43	Output Protocol and Output Format	122
Thickness	43	P	
Three Level Plane	36	Part Program Settings	7
Torus	44	Part Program Structure	4
Measurement Commands.....	22	PC-DMIS Settings	4
Motion Control.....	131	Probe Management Commands ...	14
Movement Commands.....	130	Problems in Developing	171
Movement Parameters.....	132	Procedures	
O		Declaration	169
Output Control		Variables.....	168
Basics.....	112	Procedures.....	168
Output to File.....	116	Program Translations	
Output to Printer.....	115	Arithmetical Operators	13

Tutor Translator

Functions.....	13	Current Current Selpl command	135
Program Declaration	10	Current Fly Mode command	136
Relational and Logical Operators	14	Current Mspeed command	135
Variable Declaration.....	12	Current Pspeed command.....	135
Vector Operators.....	14	Current reference system number command	135
Program Translations.....	10	Current tip radius commands...	134
R		Date	133
Relationship Between Elements		DECR Command.....	141
CIRCLE_BY_CONE	85	Depth command	136
Distance	83	Exec.....	138
Intersection.....	74	GET_POSITION Command....	140
Middle.....	81	GETDIR Command.....	140
Projection	72	INCR Command	141
Relationship Between Elements ...	71	Info.....	134
S		Ph9 present command.....	136
Scanning Control	141	Play.....	137
Simulation Subroutines	170	Read I/O channels command ..	140
Sphere Construction	68	SYSTEM_INFO	139
System Functions		Time.....	134
CMM Type command	136	Type.....	134
Current Approach command ...	135	Write I/O channels commands.	140
Current Block Number command	135	System Functions.....	133

Index

T	Probe Mapping	8
THEO Command	111	
U	Translation.....	8
User Interface	Translator options	7
	User Interface.....	6