Function list PCM__Calypso_2023 - 7.6_

Calculation and comparison operators

+ Sum of two numbers or strings	**2+7** ÿ **9** **"C:\tmp"+ "\test.txt"** ÿ **"C:\tmp\test.txt"**
- Difference between two numbers	**9-7** ÿ **2**
*** Product of two numbers	**2*7** ÿ **14**
/ Quotient of two numbers	**12/3 ÿ4**
**** Power of two numbers	**10**2** ÿ **10^2** ÿ **100** **10**3** ÿ **10^3** ÿ **1000**
< Truth value of "a less than b"	**6<2** ÿ **false** **2<6** ÿ **true**
> Truth value of "a greater than b"	**6>2** ÿ **true** **2>6** ÿ **false**
<> Truth value of "a not equal to b"	**6<>2** ÿ **true** **6<>6** ÿ **false**
== Truth value of "a equals b"	**6==6** ÿ **true** **2==6** ÿ **false**
>= Truth value of "a greater than or equal to b"	**7>=6** ÿ **true** **7>=7** ÿ **true** **6>=7** ÿ **false**
<= Truth value of "a less than or equal to b"	**7<=6** ÿ **false** **7<=7** ÿ **true** **6<=7** ÿ **true**

Mathematical functions

abs(number)
Amount of a number

abs(12.3) ÿ **12.3**
abs(-12.3) ÿ **12.3**

angle(degree angle[,minutes][,seconds])
Conversion of angles in degrees-minutes-seconds to decimal degrees

angle(12,30,30) ÿ **12,50833333**

arccos(number)
Arcuscosinus in Grad

arccos(0.8660) ÿ **30**

arcsin(number)
Arcussinus in Grad

arcsin(0.866026) ÿ **60**

arctan(number)
Arctangent in Grad

arctan(0.57735) ÿ **30**

arctan2(Zahl1,Zahl2)
Arctangent of the quotient of number1 and number2 in degrees

arctan2(60,60) ÿ **45**

calculatePointOnHelix(Radius,index,Ganghöhe,Trapezwinkel,Kegelwinkel)
Generation of target data for 3D curve on cylinder or cone
Thread (curve) ÿ **Point generator: Start=1 End=1800 Step=1** ÿ **calculatePointOnHelix(50,index,10,60,0)**

convert(value, unit, input unit, output unit)
Converts a value from one unit of measurement to another unit of measurement.

Value:	The value to convert.
Unit of measurement:	angle, length, temperature
Input unit:	input, protocol, (degree, radiant), (micrometer, micro, mm, millimeter, inch), (celsius, fahrenheit)
Output unit:	input, protocol, (degree, radiant), (micrometer, micro, mm, millimeter, inch), (celsius, fahrenheit)

cos(Gradwinkel)
Cosine value of degree angle

cos(30) ÿ **0.8660**

cosRad(Radiantwinkel)
Cosine value of radian angle

cosRad(0.523583) ÿ **0.8660**

deg (Radiant shop)
Degree value of radian angle

you(3.141592653) ÿ **180**

exp(number)
e^number

exp(0) ÿ **1**
exp(1) ÿ **2.718**

int(number)
Integer part of number

int(12.3) ÿ **12**
int(-12.8) ÿ **-12**

ln(number)
Natural logarithm of number

ln(12) ÿ **2,4849**

log(number)
Decimal logarithm of number

log(10000) ÿ **4**

max(number1[,number2,...])
Maximum of the numbers passed

max(10,3,5,11,6) ÿ **11**

min(number1[,number2,...])
Minimum of numbers passed

min(10,3,5,11,6) ÿ **3**

mod(number1,number2)
Number1 modulo Number2
mod(10,3) ÿ **1 (10/3 = 3 Remainder 1)**

not(expression)
Logical negation of the truth value of the expression
false ÿ **true**
true ÿ **false**

ord(Boolean value) returns
1 (if true) or 0 (if false)
word(10>9) ÿ **1 word(10<9)** ÿ **0**

point(x,y,z[,nx,ny,nz])
Point with X, Y, Z coordinates and optional normal vector
point(10.2,7.4,6.3,0,0,1)

rad(Gradwinkel)
Radiant value of degree angle
rad(180) ÿ **3,1415926535898**

radArccos(number)
Arccosine in radians
radArcos(1) ÿ **0**
radArcos(0) ÿ **1.5708**

radArcsin(number)
Arcsine in radians
radFace(1) ÿ **1.5708**

radArctan(number)
Arctangent in radians
radArctan(1) ÿ **0.7854**

radArctan2(Zahl1,Zahl2)
Arctangent of the quotient of number1 and number2 in radians
radArctan2(60,60) ÿ **0.7854**

round(number[,number of decimal places])
Number rounded to number of decimal places
round(3.1415926535898,2) ÿ **3,14**
round(100.12345) ÿ **100**

sign(Zahl1[,Zahl2])
Sign of number1 or number1*sign(number2)
multiplies the sign of number2 with number1
sign(-10,-2) ÿ **10**

signWithZero(Zahl1[,Zahl2])
Sign of number1 or 0 or number1*signWithZero(number2)
signWithZero(200,-5) ÿ **-200**
first number is multiplied by the sign of the second number multiplied

sin(Gradwinkel)
Sinuswert von Gradwinkel
sin(60) ÿ **0.8660**

sinRad(Radiantwinkel)
Sine value of radian angle
sinRad(1.04716) ÿ **0.8660**

sqrt(number)
Square root of number
sqrt(225) ÿ **15**

3rd root of number
125(1/3)** ÿ **5**

squared(number)
Number squared
squared(3) ÿ **9**

tan(Gradwinkel)
Tangent value of degree angle
tan(30) ÿ **0.57735**

tanRad(Radiantwinkel)
Tangent value of radian angle
tanRad(0.5235987) ÿ **0.57735**

vector(x,y,z)
Point with X, Y and Z coordinates
PointA = vector(10,20,30)
PointA.x ÿ **10 PointA.y** ÿ **20 PointA.z** ÿ **30**

String functions

asc(character)
ASCII code (number) of characters

```
Hugo = inquireText("Enter characters, you will get the ASCII code!")
tt = asc(A)
message (tt) A ÿ 65
```

chr(number)
Characters with the specified ASCII code

```
Hugo = inquire("Enter ASCII code, you will get the character!")
tt = chr(65)
message (tt) 65 ÿ A
```

cr()
new line

```
cr()
```

format(number)
Converting number to string

```
format(3278,45) ÿ "3278,45"
```

formatL(number [,total number of digits [,number of decimal places]])
creates a left-aligned format of a floating point number

```
formatL(12345.6789,15,2) ÿ "12345.6700 "
```

formatR(number [,total number of digits [,number of decimal places]])
creates a right-aligned format of a floating point number.

```
formatR(12345.67,15,6) ÿ " 12345.670000"
```

inStr([n1,]String1,String2)
Index of string1 within string2 after the n1-th or first character

```
inStr("Hallo","o") ÿ 5
inStr(3,"Position","o") ÿ 7
```

len(string)
Length of the string

```
len("Hello") ÿ 5
```

mid(string,n1[,length])
Copy of string from n1-th character to the end or to the specified length

```
mid("Hello",2,1) ÿ a
```

qm()
Inserts double quotes into a string

```
"
```

strElement(n,character,string) n-th element of the string with character as separator

```
strElement(3,"/","aa/bb/cc/dd/ee") ÿ cc
strElement(1,"d",0.1234d) ÿ 0.1234
```

subStr(string,index1,index2)
Copy of string from index1 to index2

```
subStr("This is a test",4,9) = "s is a"
subStr("abcdefgh",4,7) ÿ defg
```

text(Wert1[,Wert2,Wert3,...])
String of all passed parameters

```
text(20,30,20) ÿ 203020
text("the","the","that") ÿ thethethat
"Diameter" + text(1) ÿ Diameter1
"Diameter" + "1" ÿ Diameter1
```

val(string)
Converting string to number

```
val(12.3) ÿ 12.3d
```

Input and output

confirm([condition])
Dialog for querying a Boolean value

confirm("Are you sure?")
Yes No
Return value, true or false

display(Wert)
String or number is displayed in a separate window **display(12345,cr(),"message")**
If multiple calls are made, everything is displayed in one window **display("Check hole_10")**
displayed, the CNC sequence is
not interrupted or stopped!

display("#displayCLOSE")
Closes all windows Display

display("#displayOFF")
Ignores further display commands until the next display("#displayON") command.
This disables all control outputs with one line

display("#displayON")
Turns the consideration of the display commands back on

inquireList(Title,Point1[,Point2,...])
Dialog for selecting from a list **inquireList("Please select","123","456","789")**

inquireNumber([Titel])
Dialog for querying a number **inquireNumber("Please enter a number")**

inquireParameterList([p1,c1,p2,c2,...])
Dialog for entering multiple parameters p1... with comment c1...
inquireParameterList("D1","diameter_large","D2","diameter_small")

inquirePasswordText([Titel]) **inquirePasswordText("Please enter password")**
Dialog for querying a character string. Input is encrypted

inquireText([Titel]) **inquireText("Please enter a name")**
Dialog for querying a character string

list([Wert1,Wert2,...]) **list(Para1,Para2,Para3)**
Generates a list from all passed parameters ÿ **List("Level1" "Level2" "Level3")**

message([Wert1,Wert2,...]) **message("CNC process is aborted!")**
Output dialog **message("The value of variable P1 is:", P1)**

print([Wert1,Wert2,...])
Output of character strings or numbers in the work log,
but must be activated! **print("Print this dialog text in the work log")**

redrawCAD()
Recalculates all elements and updates the CAD window

File commands

addToFile(Filename[,Value1,Value2,...])
Adds new line to file

```
wd = getWD()
row=getActual("Circle1").diameter
addToFile(wd+"\info.txt",zeile)
addToFile("info.txt",data1," ",data2," ",data3)
```

addToFileUtf8(Filename[,Value1,Value2,...])
Appends new line to UTF8 encoded file

zipDir()
Compresses a folder

```
test = getActualInspectionDir()
zipDir(test)
```

copyFile(FilenameOld,FilenameNew)
Copying a file

```
copyFile("test.txt","test2.txt")
```

deleteFile(filename)
Delete the specified file

```
deleteFile("C.\tmp\test.txt")
```

deleteSTLPoints()
Deleting the set of points read via readSTLFile

```
deleteSTLPoints()
```

directoryPath(string)
List of

"actual"	The directory for actual values
"actualInspection"	Das Verzeichnis des aktuellen Prüfplans
"applicationData"	The user directory
"defaultInspections"	The default directory of inspection plans
"defaultResults"	The standard directory of results
"formplot"	The list of CALYPSO form plots:
"inspections"	The user-dependent directory of test plans
"programFiles"	The directory of program data
"protform"	The directory of protocol templates
"results"	The user-dependent directory of results:
"workarea"	The workarea directory

exportCharacteristicsHierarchy()
Returns a list of all test characteristics of a test plan to an ASCII file named cfHierarchy.txt in the current test plan directory.

exportCharacteristicsHierarchy() ÿ **cfHierarchy.txt**

exportFrequencyParameters([feature group or "complete"])
Export of frequency parameters

exportPoints ("measurement_element_name","path_and_file_name")
Exports actual values and, if applicable, target values and deviations of the measuring element to the file

```
exportPoints("Circle1","c:\temp\points.txt")
```

fileExists(filename)
Checks whether a file or directory exists

```
fileExists("C.\tmp\test.txt")
```
ÿ **true or false**

generateTableColumn(column ID, function name)
Writing table columns

getActualInspectionDir()
List of current test plan

"C:\Users\Public\Documents\Zeiss\CALYPSO\workarea\inspections\Pruefplan"

getRESULTS()
Returns the directory where the test plan results are stored.

getWD()
getWD()
Current directory **"C:\Users\Public\Documents\Zeiss\CALYPSO\"**

importResultFile

readCTFile(filename)
Reading a CT dataset

readListFile(filename)
liste = readListFile("C:\tmp\test.txt")
Read list from ASCII file

List includes additional test characteristics to be measured **addCF(list) (see p. 9)**

readPCMFile(filename) **readPCMFile("C:\Listen\test.para")**
Reading a PCM file

readUserAttributes(filename)
Reading in test characteristic attributes

readSTLFile(Dateiname)
Reading STL files as point sets

renameFile(FilenameOld,FilenameNew) **renameFile("alt.txt","neu.txt")**
Renaming a file

runPCMFile(filename)
Processing a text file with PCM commands **runPCMFile("C:\tmp\zusatz.txt")**

saveMeasurementPlanAs() saveMeasurementPlanAs("c:\temp\MyPlan", true)
Parameter 1: File path. File path under which the current test plan is saved
If the file path does not exist, it will be created.
Parameter 2: Save CAD model (optional) Boolean. Default: false.
Specifies whether the CAD model should be saved

saveViewToFile Saves
a file with *"CAD ÿ View ÿ Save View"* **saveViewToFile(viewname,filename)**
saved CAD view as jpeg file **saveViewToFile("wuerfel","C:\tmp\wuerfel.jpeg")**

selectFileDialog([Path (or path symbol like directoryPath)[,Filter[,Dialog text[,Return mode]]]])
File selection dialog **Datei = selectFileDialog("C:\tmp\Archiv\PCM")**
ÿ **"C:\tmp\Archive\PCM\myFile.txt"**

waitForFile(filename[,message,flag1,timeout,logtext,flag2])
waitForFile("C:\tmp\file.txt","Please wait",1,5,"that's it",1)
Stops the process until one of the following events occurs:
– A file "filename" exists
– A set waiting time has been reached
– The system reports “Directory does not exist”

filename	File name with absolute or relative path	
message	This text will be displayed on the screen while waiting	
flag1	0: Do not display file name	1: Show file name
timeout	Waiting time in seconds	

log text Message in the log if the waiting time has been reached
flag2 0: do not check if directory exists 1: check if directory exists
Return value: 0 File exists
1 waiting time has been reached
2 Directory not available

writeActualsToFile([test characteristic,contact points true/false,coordinate system]file)
Export of actual values with point status

writeActualsToVDA([test characteristic,touchpoints true/false,]file)
Export of actual values in VDA format **File**
is written to the current directory. writeActualsToVDA("Test.vda")
File is written to the directory C:\temp File is writeActualsToVDA("C:\temp\Test.vda")
written to the directory C:\temp and provided with the part number.
writeActualsToVDA("C:\temp\Test"+getRecordHead("partnbinc")+".txt")
Exports surface points **writeActualsToVDA("Diameter1",true,"c:\test.vda")**
Exports probe center coordinates **writeActualsToVDA("Diameter1","c:\test.vda")**

writeBliskFinishFile(Bliskname)
Writes the file briskFinish.xml to communicate with BladePro

writeBliskStartFile(Bliskname)
Writes the file briskStartup.xml to communicate with BladePro

writeCurveDistanceAsPlaneToFile(inspection characteristic,plane axis 1 or 2 or 3,file)
Curve distances as coordinates for a plane. (Coordinates of the plane axis correspond to the curve distance)

writeDiffCoordSysToFile([System1[loop index][actual system true/false]]System2[loop index][actual system true/false]File)
Export a difference system of two coordinate systems as a 4x4 matrix

writePointsCloudToFile([Test Characteristic]File)
Export of point sets as CALYPSO session file

writeUserAttributes[filename[,1.attributename[,2.attributename...]]])
Export of inspection characteristic attributes

Measurement-specific functions

addCF(Prüfmerkmalsname[,PMName2,...])
Activates test characteristics to be measured

addCF("Y-value_2d-line1")

addME(MeasuringElementName1[,MEName2,...])
Activates measuring elements to be measured
(only works if “By measuring element list” is selected under Sequence flow.)

addME("Point_K1")

assignStrategy(StrategyName)
Measurement strategy assignment is loaded and applied immediately (without
Dialogue). A new strategy group is created with the name
of the loaded catalog. The measurement strategies are then placed in this strategy group
The command has a single parameter.

autoSubsequEval([0=Off][1=On with feature calculation][2=On without feature calculation])
Mode "Measurement only for automatic subsequent evaluation" on the CMM computer for Autorun/FACS
Mode "Measurement only for automatic subsequent evaluation" on the CMM computer for Autorun/FACS

baseSystem()
Access to properties of the base system

baseSystem().x	**x-value of the base system**
baseSystem().y	**y-value of the base system**
baseSystem().z	**z-value of the base system**
baseSystem().valueA	**valueA value from the base system**
baseSystem().euler1	**euler1 vom Basissystem in rad (Kippwinkel)**
baseSystem().euler2	**euler2 vom Basissystem in rad (Kippwinkel)**
baseSystem().euler3	**euler3 from the base system in rad (tilt angle)**

clearParameter([Parametername1, ..2, ...])
Delete all or individual parameter values

clearParameter()
clearParameter("Hugo")
clearParameter("Hugo","Susie")

compute(string)
Calculates the value of a term or a PCM parameter

Max_Moritz = 4711
Teil1 = "Max"
Teil2 = "Moritz"
or = Tile1+"_"+Tile2
Hugo = compute(part1+"_"+part2)
ÿ **4711**
Wolf = (Teil1+"_"+Teil2)
ÿ **Max_Moritz**

defineFunctionEnd(return value)
Marking end of function

defineFunctionStart(function name)
Marking function start

defineRecheckPartNumber
Setting the special part number for the repeat measurement

differenceSystem()
Access to properties of the difference system of an iterative alignment
Possible values for “property” are: “x”, “y”, “z”, “valueA”, “euler1”, “euler2” and “euler3”:

differenceSystem().Property

endInspection()
Test plan process is terminated

endInspection()
endInspection("NO_DATA_OUTPUT",1)
The second parameter '1' after the comma also terminates the elements for the basic system.

executeFunctionNamed(function name)
Execute Function

getActual([element name,loop index]).attribute
Access to actual values of the test characteristic/measuring element

getActual("Ebene2").z
getActual("Kreis1",6).y
getActual(„Kegel1",LOOP3).x

The following information can be used for **"attribute" :**

Attribute	Return value
a1	**Shop 1**
a2	**Shop 2**
angleForDisplay	**Store**
angleSegment	**Angle range**
apexAngle	**Cone shop**
apexAngleHalf	**Halber Kegelwinkel**
area	**Area of a curve**
boreIsMissing	**Information whether hole is missing, if yes true, if no false**
container	**Id:MeasuringElementName; idTestCharacteristicName**
coordPolAngle	**Polar coordinate angle**
coordPolHeight	**Polar coordinate height**
coordPolRadius	**Polarkoordinate Radius**
deviation	**Deviation in the set tolerance mode**
diameter	**diameter**
diameter2	**Diameter 2 of the ellipse**
distance	**Distance of a symmetry point**
elementIsMissing	**"true" if element could be measured "false" if element could not be measured**
endAngleUAxis	**Angle of the second contact point (circle-in-contour fitting)**
endPointA	**Angle of the endpoint in polar coordinates**
endPointH	**Height in z-direction of the end point in polar coordinates**
endPointR	**Radius of the end point in polar coordinates**
endPointX	**x-value of the end point of a 2d line**
endPointY	**y-value of the end point of a 2d line**
endPointZ	**z-value of the end point of a 2d line**
expansionA	**Length (rectangle or groove)**
expansionB	**Width (rectangle or groove)**
focus1A	**Angle of the first focal point in polar coordinates**
focus1H	**Height in z-direction of the first focal point in polar coordinates**
focus1R	**Radius of the first focal point in polar coordinates**
focus1X	**x-value of the first focus of an ellipse**
focus1Y	**y-value of the first focus of an ellipse**
focus1Z	**z-value of the first focus of an ellipse**

focus2A	Angle of the second focal point in polar coordinates
focus2H	Height in z-direction of the second focal point in polar coordinates
focus2R	Radius of the second focal point in polar coordinates
focus2X	x-value of the second focus of an ellipse
focus2Y	y-value of the second focus of an ellipse
focus2Z	z-value of the second focus of an ellipse
form	Formal errors
gap	Gap thickness between contour and fitted circle
getCamLift	Maximum cam lift or cam lift
getCamLiftAngle	Angle (polar position) of the maximum cam lift or cam lift in the polar coordinate system of the cam
getCamLiftAngleAccPnt	Angle (polar position) of the maximum cam lift. The first setpoint in the ascending direction serves as the reference (angle = 0). The angle queried corresponds to the angle that is written to the file when the results are output as a text file.
getCamLiftIndex	Number of the setpoint with the maximum curve stroke
inclinationAngle	Kipp shop
indexOFmaxDev	Index of Max Deviation
indexOFminDev	Index of MIN deviation
only	Long
length	Length of a curve
materialDetected	“true” if material is present; “false” if no material is present, “9999” if error (not defined)
maxDev	maximum deviation “+”
mean	Mean value of the special element point set
midPoint	Center of a circle / a 2d line
myDev	maximum deviation "-"
planeNumber	Spatial axis
plcn	Spatial axis as number
radius	Radius
radiusD2	Radius 2
rotationAngle	Drehwinkel
rotX	Angle of rotation around the X-axis
company	Angle of rotation around the Y-axis
snot	Angle of rotation around the Z-axis
sigma	scattering
toleranceState	Returns the tolerance status of the inspection characteristic
transX	Displacement in X direction
transY	Shift in Y direction
transZ	Displacement in Z direction
x	x-value of the reference point x-value of the zero point for coordinate systems and hole patterns
xMaxDev	maximum deviation in x-direction
xMinDev	minimum deviation in x-direction
and	y-value of the reference point y-value of the zero point for coordinate systems and hole patterns

theMaxDev	**maximum deviation in y-direction**
yMinDev	**minimum deviation in y-direction**
with	**z-value of the reference point** **z-value of the zero point for coordinate systems and hole patterns**
zMaxDev	**maximum deviation in z-direction**
zMinDev	**minimal deviation in z-direction**
getMaxCurveJump	**Curve jump**
getOffsetKind	**Tolerance offset of the curve shape**
getOffsetValue	**Tolerance shape of the curve shape**
lowTolAtMaxDev	**Curve**
uppTolAtMaxDev	**Curve**
lowTolAtMinDev	**Curve**
uppTolAtMinDev	**Curve**

The tolerance form for “DIN Position” RectangleXY provides 2 results, the deviation in x and in y.
The **individual** deviations can be accessed as follows :

getActual("Position_Kreis1^X").deviation
getActual("Position_Kreis1^Y").deviation

getActualCurvePointCoord("Kurvenname","X"[,Schleifenindex],Punktnummer)

Access to the coordinates (X,Y,Z and angle, radius, height) of each individual curve point

getActualCurvePointDev("Curvename"[,Schleifenindex],Point number)

Access to the deviation of each curve point

getActualCurvePointDevX("Curvename"[, index], Point number)

Access to the X deviation of each curve point

getActualCurvePointDevY("Curvename"[, index], Point number)

Access to the Y deviation of each curve point

getActualCurvePointDevZ("Curvename"[, index], Point number)

Access to the Z deviation of each curve point

getActualCurvePointLowTol("Kurvenname"[,Schleifenindex],Punktnummer)

Access to the lower tolerance of each curve point

getActualCurvePointUppTol("Curvename"[,Schleifenindex],Point number)

Access to the upper tolerance of each curve point

getActualProfilePointCoord("FreiformFlächenForm","X",Punktnummer)

Access to the coordinate (X,Y,Z,Nx,Ny,Nz and Angle, Radius, Height) of each freeform surface point

getActualProfilePointDev("FreiformFlächenForm",Punktnummer)

Access to the deviation of each freeform surface point

getCFAttribute

Returns the current value of an attribute. Can only be used within functions without specifying the characteristic name.

getCFAttribute(attributname)
getCFAttribute(prüfmermalsname,attributname)

getCFGroupname Returns
the current highest group name for the test characteristic. Can only be used within functions without specifying the test characteristic name.
getCFGroupname()

getCFGroupname("test characteristic name")

getCFGroupname2 Returns
the current deepest group name for the test characteristic. Can only be used within functions without specifying the test characteristic name.
getCFGroupname2()

getCFGroupname2("test characteristic name")

getCFNames
Returns the test characteristics of the group!

getCFNames("Group") ÿ
OrderedCollection ('Dm_Circle' 'Roundness1' 'Flatness1')

getCornerPointFromCAD("min" or "max")
Key points of the CAD model

```
eck_klein_z = getCornerPointFromCAD("max").z
eck = getCornerPointFromCAD("min")
small_X = corner.x
small_Y = corner.y
small_Z = corner.z
```

getFunctionObjectName()
Returns the name of the current test characteristic
(Only usable between defineFunctionStart and defineFunctionEnd!)

getGroupNames
Displays all group names

getMaxActual([hole pattern name]) **getMaxActual("Name of the fit").deviation**
Access to the actual value of a hole pattern or 2-point diameter with the largest actual value
2-point diameter, maximum: **getMaxActual("Elementname[,loopindex]").actual**
2-point diameter, maximum angle: **getMaxActual("Elementname[,loopindex]").angleOfRadiusPoint**

getMinActual(Elementname,Schleifenindex) **see above…**
Access to the actual value of a hole pattern or 2-point diameter with the smallest actual value

getNominal(element name[,loop index]) **getNominal("Circle1").Property**
Access to target values of the test characteristic/measuring element

getNominal().identifier	Access to the name of the test characteristic
getNominal().testPlanGeometry.identifier	Access to name of the measuring element

getNominalCurvePointCoord("CurveName", "X" ,Punktnummer)
Access to the coordinates (X,Y,Z,Nx,Ny,Nz and angle, radius, height) of each individual curve point

getParameterNamed(parameter name[,loop index]) getParameterName(parameter name)
Returns the value of the parameter whose name is the value of the string variable **parametername .**

To access parameter values, you can only use fixed parameter names in expressions. Using this function, you can make the parameter name variable.

```
List = readListFile(file)
for i=1 to 10
test[i] = getParameterNamed(Liste,i)
next i
```

getRecheckMode()
Access to repeat measurement mode

Repeat measurement on ÿ **true**
Repeat measurement off ÿ **0**

getRunID ()
Query of the selected feature group (Run)

getRunID()

getStartSetting

Querying the CNC start parameters
When creating via PCM selection, a
Selection window

selection	Selection
clearOldRes	Reset old results
meMode	Order of procedure
naviMode	Driving between measuring elements
runMode	Expiration mode
baseSystemType	Type coordinate system
baseSystemRealName printer	Name coordinate system
speed	Drucker
	speed
manAligment	Manual alignment
activeTechnology	Execution strategy

inspectionToleranceState()
returns the tolerance status of the inspection plan

Return value	**meaning**
notDefined	**Status is not yet determined.**
inTolerance	**Inspection plan within tolerance: All elements are within the specified tolerances.**
outOfTolerance	**Test plan out of tolerance: At least one element of the test plan is outside the specified tolerances.**
outOfLimit	**Test plan out of limits: At least one element of the test plan is outside the specified warning limits.**

inspectionToleranceState("toleranzstatus") **Total = inspectionToleranceState("total")**

total	**Total number of test characteristics**
intolerance	**Number of test characteristics in tolerance**
outOfLimit	**Number of test characteristics above tolerance limit**
outOfTolerance	**Number of test characteristics outside tolerance**
noResult	**Number of non-calculated test characteristics**
wpSystem	**Number of coordinate systems**
wpSystemNoResult	**Number of uncalculated coordinate systems**

isParameterDefined(parameter name[,loop index]) **test = isParameterDefined(Hugo)** ÿ **"true" or "false"**
Query whether parameters are assigned (Boolean return)

measure(Elementname[,Schleifenindex]) **measure("Kreis1")**
Measuring an element

measuringForce([element name or force,element name])
Setting or reading the measuring force **measuringForce(measuring force in mN,"measuring element")**
Reading the measuring force **measuringForce("Zylinder1")**
Setting the measuring force **measuringForce(100,"Zylinder1")**

numberOfPointDistanceExceed(curve shape name, comparison value)
Number of curve points with a deviation greater than

numberOfPointDistanceLessThan(curve shape name, comparison value)
Number of curve points with a deviation less than

outputMultiQdas outputMultiQdas(attributname)
Generates a separate QDAS evaluation according to a specified test characteristic attribute.

setBaseSystem("Name of the base system") **setBaseSystem("Test1")**
Integrate the base system dynamically, start with existing alignment

setCF(Test characteristic name[,PMName2,...]) or (List name)
Resets test characteristics to be measured **setCF(readListFile("C:\tmp\xxx.txt"))**
Ascii file with test characteristic names and / or group names
Here, one PM or group is entered per line, without quotation marks and commas

setCFAttribute(AttributName,Wert)
Set attribute for current test characteristic

setCurveJump
setCurveJump("Test Characteristic", "CurveJumpType", KSArea, KSTolerance, [Overlap])

Defining the curve jump tolerance (CST): **setCurveJump("KurvenSprung1","point", 15, 0.05, 0.5)**
Check element = featureElement

CurveJumpType = point, angle, length KSRange

= value as number of points, angle range or length of the curve range Tolerance = tolerance value
Overlap = value: 0, 0.1, 0.2 ...

0.9, 1.0

setElementStrategy([Messelementname],Strategiename)
Setting the multiple strategy of a measuring element

setFilter(["Test characteristic"],"Test element","Filter type","Filter type","Filter criterion",Value1,[Value2], [Connect segments])

setFilter("Diameter_Circle","spline","low","undulation", 50)
setFilter("Flatness","gauss","low","wavelength", 1.8)

Prüfelement = featureElement, featureElement1, featureElement2, primaryDatum, secondaryDatum...
Filtertyp = gauss, spline, 2 RC, off, fromFeature Filterart =
low, band, high Filterkriterium =
undulation, wavelength Seg.verbinden = true,false

If no test characteristic is used, a measuring element must be used instead of the test element.
If the filter is switched off with "off" or the settings are adopted from the test element with "fromFeature", all other parameters can be omitted.

setGeometryInnerOuter("INNER" or "OUTER")
Setting the inside/outside identifier of a measuring element

setGeometryInnerOuter("INNER")
InOut = "INNER"
setGeometryInnerOuter(InOut)

setInspectionStrategy(Strategiename)
Setting the multiple strategy of the test plan (empty strategy name for fallback strategy)

setLineScanGlobalSetting(setting name,setting directory)
Global setting of LineScan settings

setMissingBore([measuring element name],search distance)
Set missing hole for measuring element Hugo = getActual("Kreis1").boreIsMissing

setMissingBore("Circle1",10) true

- (missing bore), false (existing bore)

setOutlier(["Test characteristic"],"Test element",Factor iW,Factor aW,"Calc.Data reduction.",[Value],[Until compensation.Element],Number of iterations,"Pre-filter",Value1,Value2)

setOutlier("Diameter_Circle",3,3,"adjacentPoints",0,false, 1, "undulation", 10, 5000)

Prüfelement = featureElement, featureElement1, featureElement2, primaryDatum, secondaryDatum...
Factor iW (inside workpiece) = number, off, fromFeature Factor aW
(outside workpiece) = number Range data red. =
onlyOutliers, "adjacentPoints" UntilCompensationElement
= true,false Number of iterations = number
Pre-filter = undulation,
wavelength

If no test characteristic is used, a measuring element must be used instead of the test element.
If the outlier is switched off with "off" or the settings are adopted from the test element with "fromFeature", all other parameters can be omitted.

setParameterNamed(Parametername,Wert[,Index])
Assignment using parameter name (if index specified, assignment in list)

settingPut(„useAddCFsForTableFile“.true)
Statistics

setRecheckMode (Recheck[,Reset][,Maximum])

setRecheckMode (1,0,3)
setRecheckMode (0)

Sets the repeat measurement mode for the test plan Recheck = 0 ÿ Repeat measurement disabled Recheck = 1 ÿ Repeat measurement enabled Reset = 0 ÿ No resetting of part numbers Reset = 1 ÿ All part numbers are reset

Maximum ÿ Maximum number of repeat measurements.

Maximum 0 ÿ no limit

setRoughnessValue("Test characteristic", "Roughness parameter", target value, upper tolerance [,lower tolerance])
Setting the target value and tolerance[s] of a roughness parameter ("Roughness1","Rz",13.0, 2.0):

setRunID(feature group[,name2,...])

setRunID("Feature group name1")

Resetting the selected feature group (or more.)
Any selection that has already been made will be overwritten. If several feature groups are specified in one call, the total selection forms the scope of the test.

Curve

getActualCurveMaxSegDev("curve name",segment number)
Maximum deviation of a tolerated curve segment

getActualCurveMinSegDev("curve name",segment number)
Minimum deviation of a tolerated curve segment

getActualCurvePointCoord("Kurvenname","X"[,Schleifenindex],Punktnummer)
Access to the coordinates (X,Y,Z and angle, radius, height) of each individual curve point

getActualCurvePointDev("Curvename"[,Schleifenindex],Point number)
Access to the deviation of each curve point

getActualCurvePointDevX("curve name"[,loop index],point number)
Access to the X deviation of each curve point

getActualCurvePointDevY("curve name"[,loop index],point number)
Access to the Y deviation of each curve point

getActualCurvePointDevZ("Curvename"[,Schleifenindex],Point number)
Access to the Z deviation of each curve point

getActualCurvePointLowTol("Kurvenname"[,Schleifenindex],Punktnummer)
Access to the lower tolerance of each curve point

getActualCurvePointUppTol("Curvename"[,Schleifenindex],Point number)
Access to the upper tolerance of each curve point

getNominalCurvePointCoord("Curvename","X",Punktnummer)
Access to the coordinates (X,Y,Z,Nx,Ny,Nz and angle, radius, height) of each individual curve point

numberOfPointDistanceExceed("curve shape name", comparison value)
Number of curve points with a deviation greater than

numberOfPointDistanceLessThan("curve shape name", comparison value)
Number of curve points with a deviation less than

setCurveJump("Test Characteristic", "CurveJumpType", KSArea, KSTolerance, [Overlap])
Defining the curve jump tolerance (KST):

Example: setCurveJump("KurvenSprung1", "point", 15, 0.05, 0.5)

Check element = featureElement

Curve jump type = point, angle, length KS range

= value as number of points, angle range or length of the curve range Tolerance = tolerance value

Overlap = Value: 0, 0.1, 0.2 ... 0.9, 1.0

writeBliskFinishFile("Bliskname"["EXCEL1" ["EXCEL2"["ASCII"]]])
Writes the file briskFinish.xml [possibly with additional output parameters] for communication with BladePro.

writeBliskStartFile("Bliskname")
Writes the file briskStartup.xml to communicate with BladePro

writeCurveDistanceAsPlaneToFile("Test characteristic" ,plane axis 1 or 2 or 3, "File name")
Curve distances as coordinates for a plane. (Coordinates of the plane axis correspond to the curve distance)

writeNominalCurve("Curve name" ["Coordinate system"])
Export of curve setpoints in the ..\workarea\results folder in any coordinate system.

CMM-specific functions and travel commands

calculateRotosAngles

Calculates the ideal angular position for a surface measurement taking into account the real geometry.
The prerequisite is that the surface measurement only includes one measurement strategy.
0 (no collision-free angular position), 3 (1 collision-free angular position --> a, b, c) or 6 (2 possible angular positions --> a1, b1, c1, a2, b2, c2) values in RAD are returned.

Parameter 1: Name of the surface measurement

cmmBorder(Axis)

Returns the CMM limit of each axis in millimeters in the machine coordinate system.
Use e.g. cmmBorder("-X").
Possible axes: -X,+X,-Y,+Y,-Z,+Z

changeStylusNo()

changeStylusNo("1")

Changing the active button

changeStylusSystem()

changeStylusSystem("NameTastersystem")

CNC change of the probe system

cmmCanUseNavigator()

Query whether KMG NAVIGATOR-capable **cmmCanUseNavigator()** ÿ **true / false**

cncBreak()

cncBreak()

Termination of the CNC process

displayPositionCMM()

displayPositionCMM()

Messuhr-Dialog

Coordinates can be set to ”0” Retrieval via start **StartPoint=displayPositionCMM()**
point.x, .y, .z **Hugo = Startpunk.y**

getActualSide()

Current security level (for flexible path selection when changing buttons)

getCNCMode()

Current driving mode of the measuring device

getNextStylusSystemName()

Next button system when changing buttons

getPositionCMM()

Current position in device coordinates

getProbe([Taster,Tastersystem])

Access to button properties

getProbe().radius
getProbe("1_bottom").radius
getProbe("1_bottom","Stern_Tasterbau").radius
getProbe().stdProbeDev

Store
.anglePosA1 .anglePosA2 Store
Mode of measurement
Calibrated yes / no
Name of the configuration
Name des Taststifts
diameter
Radius
Date of last measurement
Calibration power of the calibration
Temperature of measurement
Shaft direction **getProbe() shaftDirection.x**
Shaft length
Shaft radius
Scattering of the measurement
Tasterkoordinate in X **getProbe(“1“,“Starprobe“).probeVector.x**
Tasterkoordinate in Y
Tasterkoordinate in Z
.calibMode .calibration .confName .probeName .diameter .radius .probeDate .probeForce .probeTemp .shaftDirection .shaftLen .shaftRadius .stdProbeDev .probeVector.x .probeVector.y .p Probe radius empty

baseSystem().machine.machineControl.sphereNormalObject.sphereRadius
Access to radius sphere normal

baseSystem().machine.machineControl.sphereNormalObject.spherePosition.x
Access to X coordinate sphere normal in machine coordinates

baseSystem().machine.machineControl.sphereNormalObject.spherePosition.y
Access to Y coordinate sphere normal in machine coordinates

baseSystem().machine.machineControl.sphereNormalObject.spherePosition.z
Access to Z coordinate sphere standard in machine coordinates

getProbeRadiusFromActual(Messelementname[,Punktindex[,Elementindex]])
Probe radius from the contact point of a measured element

getRackAssignment()
Active magazine assignment (for flexible path selection when changing probes)

getRotosAngles
Returns the angular position of a surface measurement (3 values in RAD).
Parameter 1: Name of the surface measurement

getRTData()
Querying various RT data **getRTData("rtId")**
see: PCM commands for the rotary table

getRTOffset()
Access to the rotary table offset

getRTPosition()
Access to the actual value of the last rotary table position

getRTPositionCMM()
Access to the actual value of the last rotary table position in device coordinates

getRunMode()
Querying various process states

getTemperatureCorrection()
Accessing properties of the temperature correction object

function	Return value
temperatureCorrection	**Boolean value: the temperature correction is active or not active**
.coefficientPart	**Expansion coefficient of the component**
.temperaturePart1	**Temperature of workpiece sensor 1**
.temperaturePart2	**Temperature of workpiece sensor 2**
.temperatureProbe	**Temperature of the temperature button**
.temperatureTableFrontBottom	**Temperature of the sensor at the front bottom of the table**
.temperatureTableFrontTop	**Temperature of the sensor at the top of the plate**
.temperatureTableRearBottom	**Temperature of the sensor at the rear bottom of the table**
.temperatureTableRearTop	**Temperature of the sensor at the top of the back of the plate**

.temperatureX1	**Temperature of sensor 1 on X-scale / Y/ Z**
.temperatureX2	**Temperature of sensor 2 on X-scale / Y/ Z**
.temperatureX	**Temperature of the scale in X-direction**
.temperatureY	**Temperature of the scale in Y direction**
.temperatureZ	**Temperature of the scale in Z direction**
.factorX	**Correction factor of the scale in X-direction / Y/ Z**
factorPart	**Correction factor of the component**
tempWk	**Temperature value used in the CNC process of the Workpiece**

Only for PRISMO ultra and ACCURA 2 with option TFK

isManProbeChange()
Query for manual button change (for flexible path selection when changing buttons)

lowerableRT()
Lowers or raises the lowerable turntable.

lowerableRT("UP") ÿ **The LRT is raised.**
lowerableRT("DOWN") ÿ **The LRT is lowered.**
lowerableRT("STATUS") ÿ **The state of the LRT is queried.**
The return value is "UP" or "DOWN".

measureMTMValue()
Measure the moment of inertia (workpiece including RT) and then set

measVolumeIlluminationOff()
Deactivates the measuring room lighting (O-Inspect)

myVolumeIlluminationOn()
Activates the measuring room lighting (O-Inspect)

positionCMM(x,y,z[,Achse1,Achse2,Achse3]) positionCMM(200,-200,-10,"Z","Y","X")
Travel to specified CMM position in machine coordinate system in specified axis order

positionRS(x,y,z[,coordinate system,probe]) **positionRS(50,60,70, "KS","1_-Z")**
Travel to specified position in specified **positionRS(50,60,70, "","1_-Z")** (base system)
Coordinate system with specified probe

readActualWPTemperature(empty = selected sensor or temp. sensor as string) e.g. ("1") , ("1+2")

Reading the workpiece sensor temperature (only for control output, without consideration for the temp. comp.)

readWPTemperature()
Early reading of the sensor temperature

rtAxisDefinition(rtmodus)
Values for rtmode: "load" or "measure".

searchDistance(number)
Sets the detection path to new value Number **searchDistance(5)** ÿ **5mm**

setCNCMode(Modus)
Setting the driving mode of the measuring device **setCNCMode(manual/cnc)**

setInspectionDrivingSpeed(speed in mm/s)
Change general travel speed

setMTMValue(MTM-Wert in kg m²)

Setting the mass moment of inertia (workpiece including RT)

setNavigationPath("toProbeChange" "fromProbeChange" , pathname)

Selection of the path before/after button change

setNewGeoStatToActualProbe(x,y,z,i,j,k) **setNewGeoStatToActualProbe(0,0,-47,0,0,-1)**

Sets current button to new values (x,y,z,nx,ny,nz)

setNewRadiusToActualProbe(r) **setNewRadiusToActualProbe(1.5)**

Sets current probe to new radius (r)

setRotosAngles(parameter1, parameter2, parameter3, parameter4)

Sets the angle position of a surface measurement.

Parameter 1: Name of the surface measurement
Parameter 2-4: Angle AC in RAD

setWPTemperature(workpiece temperature in degrees Celsius)

Specification of the workpiece temperature for the temp.comp.

stepRS(x,y,z[,coordinate system]) **stepRS(10,0,0, "Coordinate System1")**

Step by specified coordinates in the specified coordinate system

System commands

actCalypsoWindowName() **CalName = actCalypsoWindowName()**
Current window name of CALYPSO

closeSocket()
Closes communication channel between Calypso and external application

date([1][2]) **date()** ÿ **July 12, 2023**
Access to current date **date(1)** ÿ **12.07.2023**
date(2) ÿ **2023-07-12**

dateAndTime() **dateAndTime()** ÿ **Jul 12 2023 09:34:44**
Access to current date and time **dateAndTime(1)ÿ 12.07.2023 09:35:43**
dateAndTime(2)ÿ 2023-07-12__09-36-45

dateInNumbers() **dateInNumbers()** ÿ **12.7.2023**
Access to current date with numeric values

expandCalypso()
Maximizes the CALYPSO program window again or brings it to the front of the screen.

facsIsActive()
Returns whether FACS is active. **facsIsActive()** ÿ **true / false**

language() **language()** ÿ **de**
outputs the set language in Calypso

millisecondClockValue() **millisecondClockValue()**
Access to millisecond counter (1/100sec)

minimumLengthOfNumber
Sets the number of leading zeros in the inspection characteristic name in extractCADPoints to the specified value

openSocket()
Opens communication channel between Calypso and external application

systemCall()
Calling system commands **systemCall("C:\Zeiss \loeschen.bat")**
systemCall("C:\Program Files (x86)\Microsoft Office\Office15\Excel.exe C:\Users\idut16\Desktop\xxx.xlsx")

systemCallForResultAccess(command)
Calling system commands and waiting for them to be processed. External application can access Calypso data

systemCallIcon **systemCallIcon("D:\philips\main\help.bat")**
Calls system commands and waits for them to be processed. Program is started as an icon

systemCallWithWait(command)
Calling system commands and waiting for them to be processed

time() **time()** ÿ **09:44:47**
Access to current time **time(1)** ÿ **09:44:47**
time(2) ÿ **09-44-47**

timeInSeconds() **timeInSeconds()** ÿ **35292**
Access to current time in seconds

userDefinedMessage((value1, ..2, ..3, ...) or (list name))
Custom message to the observer. List with free content of strings or numbers

wait(nSekunden) **wait(10)**
CNC process is stopped for n seconds ÿ **Test plan is paused for 10 seconds**

Presentation protocol

defineProtocol(keyword,function value(s)) **defineProtocol (parameter, value)** ÿ
Flexible protocol output with PCM **see list of protocol output**

getRecHdForFil(protocol header variable)
Returns the current value of a log header variable in file name format
Timestamp = getRecHdForFil("time") ÿ
15_16_32

getRecordHead(log header variable)
Access to protocol header variables **Timestamp = getRecordHead("time")** ÿ
15:16:32

getRecordHeadM(log header variable)
For subsequent evaluation: Access to protocol header variables of the measurement

getRecordHeadUnchanged(log header variable)
Access to protocol header variables without character replacement in file definitions

presentationOff()
Presentation protocol is not displayed

presentationOn()
Presentation protocol is displayed

SetElementsForProtocol

Determines which test characteristics are printed with the following command
“displayGraphicsWhileSelection”.

setProtocolOutput()
Controls the protocol output (lists and printing) independently of the CNC start settings.
setProtocolOutput(protocol type,output function)

protocol type: "compactProtocol", "presentationProtocol", "workingProtocol". **output function:** "list" or "print"

setProtocolSetting()
Access to presentation protocol settings
Ausgabeformat ÿ default
setProtocolSetting("outputFormat","default")
Attach freely definable pages
setProtocolSetting("userDefinedPages","userProtocol1.gra")

setProtocolSetting("databaseSyncAfterRunFinished", true)
setProtocolSetting("databaseSyncAfterRunFinished", false) This allows you to activate or deactivate data synchronization via PCM.

setRecordHead(log header variable,value)
Setting protocol header variables

SetViewsForProtocol()
Determines which CAD views are output in the protocol.
SetViewsForProtocol(viewname1,viewname2,…..)

closeAllPlotts()
closes all protocols at the end of a CNC run, except the current ones

List log output

Parameter	Possible values		
"#name"	freely definable character string		
"#type"	"compactProtocol"	"presentationProtocol"	"workingProtocol"
"#format"	"<format name for protocol header>"	"<formatname>"	
"#elements"	"allME" "allMENom" "allCF" "allCFLimit" "allCFOutTol"	"allCF" "allCFLimit" "allCFOutTol"	
"#order"	"alphabetic" "itemType" "rangeOfTolAndDev" "likeRun"	"likeList" "alphabetic" "itemType" "rangeOfTolAndDev" "likeRun"	
"#medium"	"screen" "print" "pdf" "ps"	"screen" "print" "pdf" "ps"	"screen" "print" "pdf" "ps"
"#warningLimitCF"	percentage		

```
// //1. Output action
        defineProtocol("#name","Test PP1")
        defineProtocol("#type","presentationProtocol")
        defineProtocol("#format","default_compressed_tolicon")
        defineProtocol("#medium","printer")
        defineProtocol("#elements","allCF")
        defineProtocol("#order","likeList")
// 2. Output action
        defineProtocol("#name","Test PP2")
        defineProtocol("#type","presentationProtocol")
        defineProtocol("#format","default")
        defineProtocol("#order","rangeOfTolAndDev")
        defineProtocol("#medium","printer")
        defineProtocol("#elements","allCFOutTol")
```

For the output parameter "#medium" it is possible to specify several values:

defineProtocol ("#medium",value1, ... ,valuen)

Function value	Meaning
compactProtocol	Compact protocol
presentationProtocol	Presentation Protocol
workingProtocol	Work log
allME	All measuring elements and all test characteristics
allMENom	Only measuring elements with test characteristics
allCF	All test characteristics
allCFLimit	Test characteristics from warning limit
allCFOutTol	Test characteristics out of tolerance
likeList	How to list
Alphabetic	Alphabetical
elementType	By element type
rangeOfTolAndDev	By degree of deviation
likeRun	After expiration
Screen	Screen
Print	Drucker
Pdf	PDF edition
Ps	PostScript

Parameterize security groups

	Plain text	PCM-Syntax
Security level	ÿ SE +Z ÿ SG	SP +Z
Security group	+Z ÿ RTSE +Z	GRP +Z
RT security level	ÿ RTG +Z	RTSP +Z
RT Security Group		RTG +Z
Own security group ÿ SG new +Z		GRP neutral +Z
Own security group ÿ Brand new +Z		Brand New +Z

PCM syntax is programmed, plain text is output country-specifically!

Grind

for-next

Loop without condition:

for i=1 to 4 [step 2] \... \next i
specific loop

---

for i=1 to 4
Everything that is written here is repeated, in this case, 4 times
next i

---

for i=1 to 4
message("This is run number: ",i)
next i

---

for i=1 to 8 step 2
message("This is run number: ",i) //(Every 2nd step is carried out)
next i //(1 – 3 – 5 – 7)

Loop with condition at the end:

repeat \... \until i>10
conditional loop

repeat
Everything between repeat and until is repeated until the condition is met at the end.

until

Example:
i=0
repeat task
i=i+1 (If the addition of the index is missing, there is a risk of an endless loop) **until**
i==4

Linking multiple conditions

Bsp:
(Hugo > 11.9) and (Hugo < 12.1)
(Hugo < 11.9) or (Franz < 55) or (Peter > 12.8) ((xyz
> 10.9) and (xyz < 11.1)) or ((abc > 10.9) and (abc < 11.1))

Condition:

if i<10 then \... \endif
simple condition

if
Everything between if and endif is executed if the **condition at the beginning** is met.
endif

if i<10 then \... \else \... \endif
Condition with alternative

if
Everything between if and else will be executed if the condition at the beginning is met.
else
otherwise what is between else and endif is executed
endif

Condition with multiple alternatives:

selectCase i \case 10 \... \[case 1 to 5 \...] \[caseElse \...] \endSelect condition with multiple alternatives

```
Number =
7 selectCase
        Number case is <= 0
                Message = "Number is too small"
        case 1, 2, 3
                Message = "Number is less than 4"
        case 4 to 7
                Message = "Number is between 4 and 7" case is
        >= 8 and (Number <= 11)
                Message = "Number is 8, 9, 10 or 11" caseElse

                Message = "Number is out of range"
endSelect
display(message)
```

CAD

clearCAD()
Clear the contents of the CAD window

deleteFreeCadEntities()
CAD entities are deleted. Only the CAD model, coordinate systems and points remain

deleteTechnologyPoints

extractCADPoints
Extracts CAD points

loadCADFile(filename)
loadCADFile("C:\Users\Public\Documents\Zeiss\CALYPSO\cad\cadcube.sat")
Load CAD file of type *.sab or *.sat

loadCadFileWithPMI(CadFileName)
Loads a CAD file with PMI

createInspectionFromPMI()
Creates test characteristics and measurement elements from PMI

moveCADModel(x,y,z)
moveCADModel(0,20,0)
Move the CAD model by x,y,z coordinates

reflectCADModel
Mirrors the CAD model on a plane perpendicular to the specified direction (1=X, 2=Y, 3=Z)

reorderBanner()
reorderBanner()
Reordering the flags

rotateCADModel **rotateCADModel(10,1)**

rotateCADModel(angle in degrees[axis as number 1-3])
Rotate the CAD model by an angle around an axis specified by the number 1,2,3. If no axis is specified, it is rotated around Z Axis: 1ÿ X; 2ÿ Y; 3ÿ Z

saveView

Saving a CAD view

setArrowLength

Sets the arrow length in relation to the height of the CAD window (0-1)

setCoordSysVisible

Displaying the coordinate system

setFit()

Display all elements in the CAD view

setShaded() **setShaded()**

Rendering the CAD model

setTarget(x,y,z)

Setting the rotation point

setViewDirection(nx,ny,nz)

Setting the viewing direction

setWireframe() **setWireframe()**

View the CAD model as a wireframe

showBanner() **showBanner()**

Displays the flags for all test characteristics

stitchingCAD(Wert)

StitchingTolerance in mm

Edit

addCircularPath(measuring element, measuring height, number of measuring points, step size, speed[, angle segment])

Adds a circle segment strategy to a measurement element

computeId(text extension)

Sets from test plan name + + Text expansion a new name together

deleteCharacteristic(Name des Prüfmerkmals)

Deletes a test characteristic

deleteFeature(Name des Messelements)

Deletes a measurement element

deleteTechnology(measurement element, index of the technology segment to be deleted)

Deletes a technology segment of a measuring element

getFeatureElement

Accesses the measurement element of the test characteristic

getTechnology(Messelement)

Reads the technology of a measuring element

getTechnologySegment(technology, segment number) **myList = list(11,22,33,44,55,66,77,88,99)**

Reads the segment of a technology **getTechnologySegment(myList,3)** ÿ **33**

renameFeature(Alter Name, Neuer Name) **renameFeature("Kreis7","Test")**

Rename measuring element ([With consecutive numbering]) ÿ Test1

setGeometryToCircle(Messelement)

Converts the measuring element into a circle

setGeometryToCone(Messelement)

Converts the measuring element into a cone

setGeometryToCylinder(Messelement)

Converts the measuring element into a cylinder

setNumberOfPoints(Technology Segment, Number of Points)

Sets the number of points in a technology segment

setStep(technology segment, step size)

Sets the step size in a technology segment

Service

serviceCoefficient(Filename,CoefficientX,CoefficientY,CoefficientZ,CoefficientA,GageId,ThermometerId,StylusId,K1,K2,K3,DifferntMpeNeeded)

Enters the information into the XML file

serviceEnd(filename)

Stops writing the XML file

serviceEndTemperatureProtElement(filename)

Stops writing a TemperatureProtElement

serviceEndTemperatureProtocol(filename)

Stops writing the temperature log

serviceHeader(Filename,Direction,Mode,Position,PositionEnd)

Beginning of the XML file position is passed as points (3 coordinates)

serviceSingleMeasurement(Dateiname,name,Target,Actual)

Writing a single measurement

serviceStartTemperatureProtElement(Filename,TempSensorType)

Start of a TemperatureProtElement consisting of several temperatures

serviceStartTemperatureProtocol(filename)

Start of temperature protocol

serviceTemperatureElement(FileName,Place,TempValue)

Writing a temperature element

PCM commands for the rotary table

rotateAbsolute(RT-Pos in RAD)
RT turn on:

Example: Rotate RT to 90 degrees: **rotateAbsolute(rad(90))**

rotateWithDriveFree(rad(0))
Turning the RT after clearing it

setRotation(RT-Pos in RAD)
current RT-Pos rely on

alignBSWithRt()
Mechanical alignment of the rotary table according to the angle of rotation of the base system

rotateReference()
Reference run is carried out:

rtPositionOffset(Offsetwinkel in RAD)
Rotating table for positioning the workpieces

rtPositionOffset(offsetwinkel in rad)

getRTOffset()
Output of the RT offset (angle of workpiece)

getRTPosition()
reading the current turntable position

getRTData()
Querying various RT data

getRTData("rtId")

rtId	**RT identifier**
rtAvailable	**RT available**
rtActivated	**RT activated for test plan**
rtActivatedOrPassive	**RT for test plan activated or passive RT**
passiveRT	**Passive RT**
mobileRT	**Transportabler RT**
lowerableRT	**Lowerable RT**
mtmValue	**Current MTM value**
axisDate	**Date RT axis**
location	**Make**
orientation	**orientation**
wobble	**Taumelwinkel**
eccentricity	**Excenter**
angleToMainAxis	**Angle to the main axis**

getRTPositionCMM()
reading the current RT-Pos absolute

measureMTMValue()
Measure the moment of inertia (workpiece including RT) and then set

setMTMValue(MTM-Wert in kg m²)
Setting the mass moment of inertia (workpiece including RT)

lowerableRT()
Lowers or raises the lowerable turntable.

lowerableRT("UP") ÿ **The LRT is raised.**
lowerableRT("DOWN") ÿ **The LRT is lowered.**
lowerableRT() ÿ **The state of the LRT is queried.**
The return value is "UP" or "DOWN".

rtAxisDefinition(rtmodus)
Values for rtmode: "load" or "measure".

Overview: Names of external PCM files

External PCM files are automatically executed at certain points in the test plan process,
if they have a specific name and are located in a specific directory.

Files in are being executed...
List of a test plan global ÿ when the associated test plan is executed
test plan list ÿ at the end of each test plan

PCM for reference
The name of the external file determines where
the external file is executed in the test plan process:

execution	Batch file	PCM file name
Before the start dialog	Inspection_pre_start_dialog.bat	inspection_pre_start_dialog_pcm.txt
After loading the test plan	inspection_post_load.bat	inspection_post_load_pcm.txt
At the beginning of the process	inspection_start.bat	inspection_start_pcm.txt
After the end of the measurement; during the process of measuring elements before calculating the test characteristics	measurement_end.bat	measurement_end_pcm.txt
Before executing the input parameters of each Measuring element		plugin_preFeature_pcm.txt
Before executing the output parameters of each Measuring element		plugin_postFeature_pcm.txt
Before executing the input parameters of each Test characteristic		plugin_preCharacteristic_pcm.txt
Before executing the output parameters of each Test characteristic		plugin_postCharacteristic_pcm.txt
After the end of the calculation, but before log/file output (not when running with a Selection of measuring elements)	calculation_end.bat	calculation_end_pcm.txt
After the end of the process and the protocol output	inspection_end.bat	inspection_end_pcm.txt
After completion of all (including externally generated) Protocols and files, as far as they are provided by CALYPSO started (e.g. PDF, DFD/DFX) (not at expiry with a selection of measuring elements)	report_end.bat	report_end_pcm.txt
Before saving the test plan	inspection_pre_save.bat	inspection_pre_save_pcm.txt

Protocol header data (reference)

The following table lists all log headers. The "identifier" is used to identify the log header in the table result file (...hdr.txt).

Data group Name	of the field	Identifier
SYS	tester	operas
	Datum	date
	Date short	dateshort
	time	time
	KMG-Typ	dme
	Device group	devicegroup
	KMG-Nr.	dmesn
	Control type	controllertyp
	Software	dmeswi
	Software Revision	dmeswv
	Prüfplan-Name	planid
	CT dataset	pointCloudFile
	Sequence	measRun
	Blatt	actpgnr
	from	nrpgs
	Unit of length	lengthunit
	Pallet location number	palletlocationnumber
	Name	vda_name
	Department:	died_department
	Phone/Fax:	vda_phone
	No.	vda_number
	Temp. Workpiece	temperatureworkpiece
	Temp. Scale X	temperaturescalex
	Temp. Scale Y	temperaturescaley
	Temp. Scale Z	temperaturescalez
	Temp. Basic system	temperaturebasesystem
	Measurement duration	durationofrun
	Tolerance mode	Deviationmode
	Tolerance mode key	deviationmodekey
	Number of decimal places	outdecimalunits
	Language identification	languageid
	MCC-Prüfplan-Version	mccplanversion
	MMC/LMC-Modus	mmcmode
Autorun	first pallet location	firstpalletlocationnumber
	current pallet location	palletlocationnumber
	last pallet space	lastpalletlocationnumber
	Pallet space	palletlocation
	Pallet identification	pallets
	Palettenname	

EDIT	Workpiece name	party
	Drawing number	drawingno
	Rev. Workpiece	partrv
	Serial No. Workpiece	partsn
	Comment test plan	partcomment
	Spannmittel	clmpid
	Spannmittel-No	clmpsn
	contraption	fixtid
	Fixture No.	fixtsn
	Previous Operation	too much
	Test report no.	vda_audit
	Version:	vda_version
	Part number:	vda_subjno
	Name:	vda_name
	Remark:	vda_remark
	Name	party
START	Incremental teilnummer	partnbinc
	Part number	partnbLong
	Comment at start	startcomment
	Order	order
	Tool	tooldf
	Process plan	mfgdev
	Lot identification	lots
	Exam ID	fall
	Pallet space	palletlocation
	Pallet identification	pallets
	Test report no.:	vda_audit
	Version:	vda_version
	Part number	vda_subjno
	Remark:	vda_remark
	Name:	in there
	Signature:	vda_signature
OTHER	Creation Date	creationdate
	Date modified	changedate
	Creator	produceoper
	Changer	changeoper
	Software Rev Creation	creationswi
	CMM type creation	creationdme

Protocol parameters PiWeb		
"operation"	Operation	${Qdb.Measurement(1086)}
"order"	Order	${Qdb.Measurement(53)}
"contractingentity"	Client	${Qdb.Measurement(1052)}
"vda_remark"	remark	${Qdb.Measurement(30)}
"manufacturer"	Manufacturer	${Qdb.Measurement(1022)}
"cavity"	cavity	${Qdb.Measurement(41)}
"measRun"	Sequence	${Qdb.Measurement(10020, -1)}
"startcomment"	Comment at start	${Qdb.Measurement(9)}
"customer"	Could	${Qdb.Measurement(1062)}
"supplier"	supplier	${Qdb.Measurement(801)}
"line"	line	${Qdb.Measurement(22)}
"lots"	Lot identifier	${Qdb.Proberty("ProtocolHeadChargeIdentification")}
"manufacturingmachinename"	Machine designation	${Qdb.Measurement(10)}
"manufacturingmachinenumber"	Machine number	${Qdb.Measurement(10)}
"nestname"	Nest designation	${Qdb.Measurement(7)}
"nestnumber"	Nest number	${Qdb.Measurement(7)}
"palletid"	Pallet identification	${Qdb.Measurement(40)}
"palletlocation"	Pallet space	${Qdb.Measurement(38)}
"vda_auditno"	Test report number	${Qdb.Proberty("ProtocolHeadAuditNumber")}
"checkreason"	Reason for review	${Qdb.Measurement(15)}
"procid"	Exam ID	${Qdb.Measurement(37)}
"partrv"	Rev. Workpiece	${Qdb.Proberty("ProtocolHeadPartVersion")}
"vda_subjno"	Part number	${Qdb.Proberty("ProtocolHeadSubjectNumber")}
"shift"	layer	${Qdb.Measurement(850)}
"partsn"	Serial No. Workpiece	${Qdb.Measurement(1001)}
"clmpid"	Spannmittel	${Qdb.Measurement(35)}
"clmpsn"	Spannmittel-No	${Qdb.Measurement(36)}
"samplesize"	Sample size	${Qdb.Measurement(20)}
"partnbLong"	Part number	Part number
"partnbinc"	Incremental teilnummer	${Qdb.Measurement(14)}
"vda_signature"	Signature	${Qdb.Proberty("ProtocolHeadSignature")}
"variant"	variant	${Qdb.Part(1011)}
"vda_version"	Version	${Qdb.Part(1004)}
"fixtid"	contraption	${Qdb.Measurement(1114)}
"fixtsn"	Device No.	${Qdb.Measurement(17)}
"material"	material	${Qdb.Measurement(1032)}
"party"	Workpiece name	${Qdb.Part(1342)}
"tooldf"	Tool	${Qdb.Measurement(3107)}
"drawingindex"	Drawing index	${Qdb.Measurement(1043)}
"drawingno"	Drawing number	${Qdb.Part(1041)}
	Prüfplanrevision	${Qdb.Measurement(1232,-1)}
"vda_departm"	Department	
"changeoper"	Changer	
"changedate"	Date modified	
"outdecimalunits"	Number of decimal places	
"actpgnr"	Blatt	
"pointCloudFile"	CT dataset	
"date"	Datum	
"dateshort"	Date short	
"produceoper"	Creator	
"creationdate"	Creation Date	

"devicegroup"	Device group	
"dmesn"	KMG-Nr	
"thank you"	KMG-Typ	
"creationdme"	CMM type creation	
"costcenter"	Cost centre	
"partcomment"	Comment test plan	
"lengthunit"	Unit of length	
"mccplanversion"	MCC-Prüfplan-Version	
"mmcmode"	MCC/LMC-Modus	
"durationofrun"	Measurement duration	
in my name	Name	
"vda_name"	Name:	
"vda_number"	No.	
"palletlocationnumber"	Pallet location number	
"mfgdev"	Process plan	
"operid"	tester	
"planid"	Prüfplanname	
"dmeswi"	Software	
"creationswi"	Software Rev Creation	
"dmswv"	Software-Version	
"languageid"	Language identification	
"controllertyp"	Control type	
"vda_phone"	Telephone/Fax	
"temperaturebasesystem"	Temp. Basic system	
"temperaturescalex"	Temp. Scale X	
"temperaturescaley"	Temp. Scale Y	
"temperaturescalez"	Temp. Scale Z	
"temperatureworkpiece"	Temp. Workpiece	
"deviationmode"	Tolerance mode	
"deviationmodekey"	Tolerance mode key	
"time"	time	
"nrpgs"	from	
"prevop"	Previous Operation	
"company"	Work	

Qdp.Measurement ÿ messungsbezogene Variable *
Qdp.Property ÿ free variable
Qpd.Part ÿ component-related variable *

* for searching & filtering in PiWeb Reporting
Make a selection from text ÿ (variable – data source – measurement)!!

Description of the attribute key: ${Localization.AttributeKey.Description(xxxx)}
Description of the text element: ${TR("MeasurementDuration")}

Textelement mit Variable	${Qdb.Property("FeatureElementName")}	// Messelement $
Textelement mit Variable	{Qdb.Property("FormDev")}	// Form $
Textelement mit Variable	{Qdb.Property("Sigma")}	// Sigma $
Textelement mit Variable	{Qdb.Property("MinDev")}	// Min Wert $
Textelement mit Variable	{Qdb.Property("MaxDev")}	// Max value
Textelement mit Variable	${Qdb.Property("MeasurementDuration")}	// Measurement duration
	${Qdb.Property("NoResultErrorText")}	// Error message
	${Qdb.Property("ISOevalType")}	// Symbol for calculation method

setProtocolSetting()
Enable or disable data synchronization

setProtocolSetting("databaseSyncAfterRunFinished",true)
setProtocolSetting("databaseSyncAfterRunFinished",false)

special character

µ ÿ all 0181

Ø ÿ total 0216 / total 157

ø ÿ all 0248

± ÿ total 0177 / total 241

ÿ ÿ old 0179 (symbol) ÿ ÿ old 0163 (symbol)

• ÿ all 0149

® ÿ all 0174

© ÿ all 0169

™ ÿ all 0153

‰ ÿ all 0137

¼ ÿ total 0188

½ ÿ total 0189

¾ ÿ total 0190

ÿ ÿ 2153, all-c

ÿ ÿ 215b, alt-c

@ ÿ alt 0064 ÿ ÿ 221a, alt-c

ÿ ÿ 2211, alt-c

ÿ ÿ 2105, all-c

ÿ ÿ 03c0, alt-c

ÿ ÿ total 8626

- ÿ all 0045

„ ÿ all 0034 / all 34

Ctrl + Z ÿ Undo

CTRL + Y ÿ User Interrupt