Funktionsliste PCM Calypso 2023 - 7.6

Rechen- und Vergleichsoperatoren

+
Summe zweier Zahlen oder Zeichenketten

2+7 ➔ 9
"C:\tmp"+ "\test.txt" ➔ "C:\tmp\test.txt"

-
Differenz zweier Zahlen

9-7 ➔ 2

*****
Produkt zweier Zahlen

2*7 ➔ 14

/
Quotient zweier Zahlen

12/3 ➔4

******
Potenz zweier Zahlen

102 ➔ 10^2 ➔ 100**
103 ➔ 10^3 ➔ 1000**

<
Wahrheitswert von "a kleiner b"

6<2 ➔ false
2<6 ➔ true

>
Wahrheitswert von "a größer b"

6>2 ➔ true
2>6 ➔ false

<>
Wahrheitswert von "a ungleich b"

6<>2 ➔ true
6<>6 ➔ false

==
Wahrheitswert von "a gleich b"

6==6 ➔ true
2==6 ➔ false

>=
Wahrheitswert von "a größer oder gleich b"

7>=6 ➔ true
7>=7 ➔ true
6>=7 ➔ false

<=
Wahrheitswert von "a kleiner oder gleich b"

7<=6 ➔ false
7<=7 ➔ true
6<=7 ➔ true

Mathematische Funktionen

abs(Zahl)
Betrag einer Zahl

abs(12.3) ➔ 12.3
abs(-12.3) ➔ 12.3

angle(Gradwinkel[,Minuten][,Sekunden])
Umrechnung von Winkel in Grad-Minuten-Sekunden in Dezimalgrad

angle(12,30,30) ➔ 12,50833333

arccos(Zahl)
Arcuscosinus in Grad

arccos(0.8660) ➔ 30

arcsin(Zahl)
Arcussinus in Grad

arcsin(0.866026) ➔ 60

arctan(Zahl)
Arcustangens in Grad

arctan(0.57735) ➔ 30

arctan2(Zahl1,Zahl2)
Arcustangens vom Quotienten von Zahl1 und Zahl2 in Grad

arctan2(60,60) ➔ 45

calculatePointOnHelix(Radius,index,Ganghöhe,Trapezwinkel,Kegelwinkel)
Erzeugung von Solldaten für 3d-Kurve auf Zylinder oder Kegel

Gewinde (Kurve) ➔ Punktegenerator: Start=1 End=1800 Schritt=1 ➔ calculatePointOnHelix(50,index,10,60,0)

convert(Wert, Maßeinheit, Eingangseinheit, Ausgangseinheit)
Wandelt einen Wert von einer Maßeinheit in eine andere Maßeinheit um.

Wert:	Der zu konvertierende Wert.
Maßeinheit:	angle, length, temperature
Eingangseinheit:	input, protocol, (degree, radiant), (micrometer, micro, mm, millimeter, inch), (celsius, fahrenheit)
Ausgangseinheit:	input, protocol, (degree, radiant), (micrometer, micro, mm, millimeter, inch), (celsius, fahrenheit)

cos(Gradwinkel)
Cosinuswert von Gradwinkel

cos(30) ➔ 0.8660

cosRad(Radiantwinkel)
Cosinuswert von Radiantwinkel

cosRad(0.523583) ➔ 0.8660

deg(Radiantwinkel)
Gradwert von Radiantwinkel

deg(3.141592653) ➔ 180

exp(Zahl)
e^Zahl

exp(0) ➔ 1
exp(1) ➔ 2.718

int(Zahl)
Ganzzahliger Anteil von Zahl

int(12.3) ➔ 12
int(-12.8) ➔ -12

ln(Zahl)
Natürlicher Logarithmus von Zahl

ln(12) ➔ 2,4849

log(Zahl)
Dekadischer Logarithmus von Zahl

log(10000) ➔ 4

max(Zahl1[,Zahl2,...])
Maximum der übergebenen Zahlen

max(10,3,5,11,6) ➔ 11

min(Zahl1[,Zahl2,...])
Minimum der übergebenen Zahlen

min(10,3,5,11,6) ➔ 3

mod(Zahl1,Zahl2) **mod(10,3) ➔ 1 (10/3 = 3 Rest 1)**
Zahl1 modulo Zahl2

not(Ausdruck) **false ➔ true**
Logische Negation des Wahrheitswerts des Ausdrucks **true ➔ false**

ord(Boolescher Wert) **ord(10>9) ➔ 1 ord(10<9) ➔ 0**
liefert 1 (bei true) oder 0 (bei false)

point(x,y,z[,nx,ny,nz]) **point(10.2,7.4,6.3,0,0,1)**
Punkt mit X-,Y-,Z-Koordinaten und optional Normalenvektor

rad(Gradwinkel) **rad(180) ➔ 3,1415926535898**
Radiantwert von Gradwinkel

radArccos(Zahl) **radArcos(1) ➔ 0**
Arcuscosinus im Bogenmaß **radArcos(0) ➔ 1.5708**

radArcsin(Zahl) **radArcsin(1) ➔ 1.5708**
Arcussinus im Bogenmaß

radArctan(Zahl)
Arcustangens im Bogenmaß **radArctan(1) ➔ 0.7854**

radArctan2(Zahl1,Zahl2) **radArctan2(60,60) ➔ 0.7854**
Arcustangens vom Quotienten von Zahl1 und Zahl2 im Bogenmaß

round(Zahl[,Anzahl Nachkommastellen]) **round(3.1415926535898,2) ➔ 3,14**
Zahl gerundet auf Anzahl Nachkommastellen **round(100.12345) ➔ 100**

sign(Zahl1[,Zahl2]) **sign(-10,-2) ➔ 10**
Vorzeichen von Zahl1 oder Zahl1*sign(Zahl2)
multpliziert das Vorzeichen der Zahl2 mit Zahl1

signWithZero(Zahl1[,Zahl2]) **signWithZero(200,-5) ➔ -200**
Vorzeichen von Zahl1 oder 0 oder Zahl1*signWithZero(Zahl2) erste Zahl wird mit dem Vorzeichen der zweiten Zahl multipliziert

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

sinRad(Radiantwinkel) **sinRad(1.04716) ➔ 0.8660**
Sinuswert von Radiantwinkel

sqrt(Zahl) **sqrt(225) ➔ 15**
Quadratwurzel von Zahl

3te Wurzel aus Zahl **125**(1/3) ➔ 5**

squared(Zahl) **squared(3) ➔ 9**
Zahl zum Quadrat

tan(Gradwinkel) **tan(30) ➔ 0.57735**
Tangenswert von Gradwinkel

tanRad(Radiantwinkel) **tanRad(0.5235987) ➔ 0.57735**
Tangenswert von Radiantwinkel

vector(x,y,z) **PunktA = vector(10,20,30)**
Punkt mit X-,Y- und Z-Koordinaten **PunktA.x ➔ 10 PunktA.y ➔ 20 PunktA.z ➔ 30**

Zeichenkettenfunktionen

asc(Zeichen)
ASCII-Code (Zahl) von Zeichen

```
Hugo = inquireText("Zeichen eingeben, Sie bekommen dann den ASCII Code!")
tt = asc(A)
message (tt)        A ➔ 65
```

chr(Zahl)
Zeichen mit dem angegebenen ASCII-Code

```
Hugo = inquire("ASCII Code eingeben, Sie bekommen dann das Zeichen!")
tt = chr(65)
message (tt)      65 ➔ A
```

cr()
Zeilenumbruch

cr()

format(Zahl)
Umwandlung von Zahl zu Zeichenkette

format(3278,45) ➔ "3278,45"

formatL(Zahl [,Anzahl Stellen gesamt [,Anzahl Nachkommastellen]])
erstellt ein linksbündiges Format einer Gleitkommazahl

formatL(12345.6789,15,2) ➔ "12345.6700 "

formatR(Zahl [,Anzahl Stellen gesamt [,Anzahl Nachkommastellen]])
erstellt ein rechtsbündiges Format einer Gleitkommazahl.

formatR(12345.67,15,6) ➔ " 12345.670000"

inStr([n1,]Zeichenkette1,Zeichenkette2)
Index von Zeichenkette1 innerhalb Zeichenkette2
nach dem n1-ten oder dem ersten Zeichen

inStr("Hallo","o") ➔ 5
inStr(3,"Position","o") ➔ 7

len(Zeichenkette)
Länge der Zeichenkette

len("Hallo") ➔ 5

mid(Zeichenkette,n1[,Länge])
Kopie von Zeichenkette vom n1-ten Zeichen bis zum Ende bzw. zur angegebenen Länge

mid("Hallo",2,1) ➔ a

qm()
Fügt Doppelhochkomma in eine Zeichenkette ein

"

strElement(n,Zeichen,Zeichenkette)
n-te Element der Zeichenkette mit Zeichen als Trennzeichen

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

subStr(Zeichenkette,Index1,Index2)
Kopie von Zeichenkette von Index1 bis Index2

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

text(Wert1[,Wert2,Wert3,...])
Zeichenkette aus allen übergebenen Parametern

text(20,30,20) ➔ 203020
text("der","die","das") ➔ derdiedas
"Durchmesser" + text(1) ➔ Durchmesser1
"Durchmesser" + "1" ➔ Durchmesser1

val(Zeichenkette)
Umwandlung von Zeichenkette zu Zahl

val(12.3) ➔ 12.3d

Ein- und Ausgabe

confirm([Bedingung])
Dialog zur Abfrage eines Booleschen Werts

confirm("Sind Sie sicher?")
Ja / Nein
Rückgabewert, true or false

display(Wert)
Zeichenkette oder Zahl wird in separatem Fenster ausgegeben
Bei mehreren Aufrufen wird alles in einem Fenster
angezeigt, der CNC-Ablauf wird dabei
nicht unterbrochen oder angehalten!

display(12345,cr(),"meldung")
display("Bohrung_10 überprüfen")

display("#displayCLOSE")
Schließt alle Fenster Anzeige

display("#displayOFF")
Ignoriert weitere display-Befehle bis zum nächsten Befehl display("#displayON").
Damit werden alle Kontrollausgaben mit einer Zeile deaktiviert

display("#displayON")
Schaltet die Berücksichtigung der display-Befehle wieder ein

inquireList(Titel,Punkt1[,Punkt2,...])
Dialog zur Auswahl aus einer Liste

inquireList("Bitte wählen","123","456","789")

inquireNumber([Titel])
Dialog zur Abfrage einer Zahl

inquireNumber("Bitte geben Sie eine Zahl ein")

inquireParameterList([p1,c1,p2,c2,...])
Dialog zur Eingabe von mehreren Parametern p1... mit Kommentar c1...

inquireParameterList("D1","Durchmesser_groß","D2","Durchmesser_klein")

inquirePasswordText([Titel])
Dialog zur Abfrage einer Zeichenkette. Eingabe wird verschlüsselt

inquirePasswordText("Bitte Passwort eingeben")

inquireText([Titel])
Dialog zur Abfrage einer Zeichenkette

inquireText("Bitte geben Sie einen Namen ein")

list([Wert1,Wert2,...])
Generiert eine Liste aus allen übergebenen Parametern

list(Para1,Para2,Para3)
➔ List("Ebene1" "Ebene2" "Ebene3")

message([Wert1,Wert2,...])
Ausgabedialog

message("CNC-Ablauf wird abgebrochen!")
message("Der Wert der Variablen P1 ist:", P1)

print([Wert1,Wert2,...])
Ausgabe von Zeichenketten oder Zahlen im Arbeitsprotokoll,
muss aber aktiviert sein!

print("Diesen Dialogtext in das Arbeitsprotokoll drucken")

redrawCAD()
Berechnet alle Elemente neu und aktualisiert das CAD-Fenster

Dateibefehle

addToFile(Dateiname[,Wert1,Wert2,...])
Fügt neue Zeile an Datei an

```
wd = getWD()
zeile=getActual("Kreis1").diameter
addToFile(wd+"\info.txt",zeile)
addToFile("info.txt",data1,"  ",data2,"  ",data3)
```

addToFileUtf8(Dateiname[,Wert1,Wert2,...])
Fügt neue Zeile an UTF8-codierte Datei an

zipDir()
Komprimiert einen Ordner

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

copyFile(DateinameAlt,DateinameNeu)
Kopieren einer Datei

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

deleteFile(Dateiname)
Löschen der angegebene Datei

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

deleteSTLPoints()
Löschen der über readSTLFile eingelesenen Punktemenge

```
deleteSTLPoints()
```

directoryPath(Zeichenkette)
Verzeichnis der

"actuals"	Das Verzeichnis für Istwerte
"actualInspection"	Das Verzeichnis des aktuellen Prüfplans
"applicationData"	Das Anwenderverzeichnis
"defaultInspections"	Das Standard-Verzeichnis der Prüfpläne
"defaultResults"	Das Standard-Verzeichnis der Ergebnisse
"formplot"	Das Verzeichnis der CALYPSO-Formplotts:
"inspections"	Das benutzerabhängige Verzeichnis der Prüfpläne
"programFiles"	Das Verzeichnis der Programmdaten
"protform"	Das Verzeichnis der Protokoll-Vorlagen
"results"	Das benutzerabhängiges Verzeichnis der Ergebnisse:
"workarea"	Das Verzeichnis workarea

exportCharacteristicsHierarchy()
Gibt eine Liste aller Prüfmerkmale eines Prüfplans in eine ASCII-Datei namens cfHierarchy.txt im aktuellen Prüfplanverzeichnis aus.

```
exportCharacteristicsHierarchy() ➔ cfHierarchy.txt
```

exportFrequencyParameters([Merkmalsgruppe oder "complete"])
Export der Häufigkeits-Parameter

exportPoints ("messelementname","pfad_und_dateiname")
Exportiert Istwerte und ggf. Sollwerte und Abweichungen des Messelements in die Datei

```
exportPoints("Kreis1","c:\temp\punkte.txt")
```

fileExists(Dateiname)
Prüft, ob eine Datei oder ein Verzeichnis existiert

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

generateTableColumn(Spalten-ID, Funktionsname)
Schreiben von Tabellen-Spalten

getActualInspectionDir()
Verzeichnis des aktuellen Prüfplans

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

getRESULTS()
Gibt das Verzeichnis zurück, in dem die Ergebnisse des Prüfplans gespeichert sind.

getWD() **getWD()**
Aktuelles Verzeichnis **"C:\Users\Public\Documents\Zeiss\CALYPSO\"**

importResultFile

readCTFile(Dateiname)
Einlesen eines CT-Datensatzes

readListFile(Dateiname) **liste = readListFile("C:\tmp\test.txt")**
Liste aus ASCII-File einlesen

Liste beinhaltet zusätzlich zu messende Prüfmerkmale **addCF(liste)** **(siehe S. 9)**

readPCMFile(Dateiname) **readPCMFile("C:\Listen\test.para")**
Einlesen einer PCM-Datei

readUserAttributes(Dateiname)
Einlesen von Prüfmerkmalsattributen

readSTLFile(Dateiname)
Einlesen von STL-Dateien als Punktmenge

renameFile(DateinameAlt,DateinameNeu) **renameFile("alt.txt","neu.txt")**
Umbenennen einer Datei

runPCMFile(Dateiname)
Abarbeiten einer Textdatei mit PCM-Befehlen **runPCMFile("C:\tmp\zusatz.txt")**

saveMeasurementPlanAs() **saveMeasurementPlanAs("c:\temp\MyPlan", true)**
Parameter 1: Dateipfad. Dateipfad unter dem der aktuelle Prüfplan gespeichert wird. Existiert der Dateipfad nicht, dann wird dieser erzeugt.
Parameter 2: CAD Modell speichern (optional) Boolean. Standard: false.
Gibt an, ob das CAD Modell gespeichert werden soll

saveViewToFile
Speichert eine mit *„CAD → Ansicht → Ansicht speichern“* abgelegte CAD-Ansicht als jpeg-Datei **saveViewToFile(viewname,dateiname)** **saveViewToFile("wuerfel","C:\tmp\wuerfel.jpeg")**

selectFileDialog([Pfad (oder Pfad-Symbol wie directoryPath)[,Filter[,Dialogtext[,Rückgabemodus]]]])
Dialog zur Datei-Auswahl **Datei = selectFileDialog("C:\tmp\Archiv\PCM")**
➔ **"C:\tmp\Archiv\PCM\meineDatei.txt"**

waitForFile(dateiname[,meldung,flag1,timeout,protokolltext,flag2])
waitForFile("C:\tmp\datei.txt","Bitte Warten",1,5,"das wars dann",1)
Stoppt den Ablauf, bis eines der folgenden Ereignisse eintritt:
– Eine Datei "Dateiname" ist vorhanden
– Eine eingestellte Wartezeit ist erreicht
– Das System meldet „Verzeichnis existiert nicht“

dateiname	Dateiname mit absolutem oder relativem Pfad
meldung	Dieser Text wird während des Wartens am Bildschirm angezeigt
flag1	0: Dateiname nicht anzeigen 1: Dateiname anzeigen
timeout	Wartezeit in Sekunden
protokolltext	Meldung im Protokoll, falls die Wartezeit erreicht wurde
flag2	0: nicht prüfen, ob Verzeichnis existiert 1: prüfen, ob Verzeichnis existiert
	Rückgabewert: 0 Datei ist vorhanden
	1 Wartezeit wurde erreicht
	2 Verzeichnis nicht vorhanden

writeActualsToFile([Prüfmerkmal,Berührpunkte true/false,Koordinatensystem]Datei)
Export von Istwerten mit Punktstatus

writeActualsToVDA([Prüfmerkmal,Berührpunkte true/false,]Datei)
Export von Istwerten im VDA-Format

Datei wird ins aktuelle Verzeichnis geschrieben. **writeActualsToVDA("Test.vda")**
Datei wird ins Verzeichnis C:\temp geschrieben **writeActualsToVDA("C:\temp\Test.vda")**
Datei wird ins Verzeichnis C:\temp geschrieben und mit der Teilenummer versehen.
writeActualsToVDA("C:\temp\Test"+getRecordHead("partnbinc")+".txt")

Exportiert Oberflächenpunkte **writeActualsToVDA("Durchmesser1",true,"c:\test.vda")**
Exportiert Tastermittelpunktskoordinaten **writeActualsToVDA("Durchmesser1","c:\test.vda")**

writeBliskFinishFile(Bliskname)
Schreibt die Datei bliskFinish.xml zur Kommunikation mit BladePro

writeBliskStartFile(Bliskname)
Schreibt die Datei bliskStartup.xml zur Kommunikation mit BladePro

writeCurveDistanceAsPlaneToFile(Prüfmerkmal,Ebenenachse 1 oder 2 oder 3,Datei)
Kurvenabstände als Koordinaten für eine Ebene. (Koordinaten der Ebenenachse entspricht der Kurvendistanz)

writeDiffCoordSysToFile([System1[Schleifenindex][Ist-System true/false]]System2[Schleifenindex][Ist-System true/false]Datei)
Export eines Differenzsystems zweier Koordinatensysteme als 4x4 Matrix

writePointsCloudToFile([Prüfmerkmal]Datei)
Export von Punktemengen als CALYPSO-Session-Datei

writeUserAttributes[Dateiname[,1.Attributsname[,2.Attributsname...]]])
Export von Prüfmerkmalsattributen

Messspezifische Funktionen

addCF(Prüfmerkmalsname[,PMName2,...])
Aktiviert zu messende Prüfmerkmale **addCF("Y-Wert_2d-Gerade1")**

addME(Messelementname1[,MEName2,...])
Aktiviert zu messende Messelemente **addME("Punkt_K1")**
(Wirkt nur, wenn unter Reihenfolge Ablauf „Nach Messelement-Liste“ ausgewählt ist.)

assignStrategy(StrategyName**)**
Messstrategie-Zuweisung wird geladen und sofort angewendet (ohne Dialog). Es wird eine neue Strategiegruppe angelegt, mit dem Namen des geladen Katalogs. Die Messstrategien werden dann in diese Strategiegruppe erzeugt. Der Befehl hat einen einzigen Übergabeparameter.

autoSubsequEval([0=Aus][1=Ein mit Merkmalsberechnung][2=Ein ohne Merkmalsberechnung])
Modus "Nur Messung für autom. nachtr. Auswerten" am KMG-Rechner für Autorun/FACS
Modus "Nur Messung für autom. nachtr. Auswerten" am KMG-Rechner für Autorun/FACS

baseSystem()
Zugriff auf Eigenschaften des Basissystems

baseSystem().x	**x-Wert vom Basissystem**
baseSystem().y	**y-Wert vom Basissystem**
baseSystem().z	**z-Wert vom Basissystem**
baseSystem().valueA	**valueA-Wert vom Basissystem**
baseSystem().euler1	**euler1 vom Basissystem in rad (Kippwinkel)**
baseSystem().euler2	**euler2 vom Basissystem in rad (Kippwinkel)**
baseSystem().euler3	**euler3 vom Basissystem in rad (Kippwinkel)**

clearParameter([Parametername1, ..2, ...])
Löschen aller oder einzelner Parameterwerte

clearParameter()
clearParameter("Hugo")
clearParameter("Hugo","Susi")

compute(Zeichenkette)
Berechnet den Wert eines Terms oder eines PCM-Parameters

Max_Moritz = 4711
Teil1 = "Max"
Teil2 = "Moritz"
neu = Teil1+"_"+Teil2
Hugo = compute(Teil1+"_"+Teil2)
➔ 4711
Susi = (Teil1+"_"+Teil2)
➔ Max_Moritz

defineFunctionEnd(Rückgabewert)
Kennzeichnung Funktionsende

defineFunctionStart(Funktionsname)
Kennzeichnung Funktionsanfang

defineRecheckPartNumber
Setzen der speziellen Teilnummer für die Wiederholmessung

differenceSystem() **differenceSystem().Eigenschaft**
Zugriff auf Eigenschaften des Differenz-System einer iterativen Ausrichtung
Mögliche Werte für „Eigenschaft“ sind: „x“, „y“, „z“, „valueA“, „euler1“, „euler2“ und „euler3“:

endInspection() **endInspection()**

Prüfplanablauf wird beendet **endInspection("NO_DATA_OUTPUT",1)**

Der 2. Parameter '1' nach dem Komma bewirkt auch ein Beenden bei den Elementen fürs Basis-System.

executeFunctionNamed(Funktionsname)

Ausführen Funktion

getActual([Elementname,Schleifenindex]).attribut

Zugriff auf Istwerte des Prüfmerkmals/Messelements **getActual("Ebene2").z**
getActual("Kreis1",6).y
getActual(„Kegel1",LOOP3).x

Für **"attribut"** können folgende Angaben stehen:

Attribut	Rückgabewert
a1	**Winkel 1**
a2	**Winkel 2**
angleForDisplay	**Winkel**
angleSegment	**Winkelbereich**
apexAngle	**Kegelwinkel**
apexAngleHalf	**Halber Kegelwinkel**
area	**Flächeninhalt einer Kurve**
boreIsMissing	**Information ob Bohrung fehlt, bei ja true, bei nein false**
container	**Id:MesselementName; idPrüfmerkmalName**
coordPolAngle	**Polarkoordinate Winkel**
coordPolHeight	**Polarkoordinate Höhe**
coordPolRadius	**Polarkoordinate Radius**
deviation	**Abweichung im jeweils eingestellten Toleranz-Modus**
diameter	**Durchmesser**
diameter2	**Durchmesser 2 der Ellipse**
distance	**Abstand eines Symmetriepunktes**
elementIsMissing	**„true", wenn Element gemessen werden konnte „false", wenn Element nicht gemessen werden konnte**
endAngleUAxis	**Winkel des zweiten Berührpunkts (Kreis-in-Kontur-Einpassung)**
endPointA	**Winkel des Endpunktes in Polarkoordinaten**
endPointH	**Höhe in z-Richtung des Endpunktes in Polarkoordinaten**
endPointR	**Radius des Endpunktes in Polarkoordinaten**
endPointX	**x-Wert des Endpunktes einer 2d-Geraden**
endPointY	**y-Wert des Endpunktes einer 2d-Geraden**
endPointZ	**z-Wert des Endpunktes einer 2d-Geraden**
expansionA	**Länge (Rechteck oder Nut)**
expansionB	**Breite (Rechteck oder Nut)**
focus1A	**Winkel des ersten Brennpunkts in Polarkoordinaten**
focus1H	**Höhe in z-Richtung des ersten Brennpunkts in Polarkoordinaten**
focus1R	**Radius des ersten Brennpunkts in Polarkoordinaten**
focus1X	**x-Wert des ersten Brennpunkts einer Ellipse**
focus1Y	**y-Wert des ersten Brennpunkts einer Ellipse**
focus1Z	**z-Wert des ersten Brennpunkts einer Ellipse**

focus2A	**Winkel des zweiten Brennpunkts in Polarkoordinaten**
focus2H	**Höhe in z-Richtung des zweiten Brennpunkts in Polarkoordinaten**
focus2R	**Radius des zweiten Brennpunkts in Polarkoordinaten**
focus2X	**x-Wert des zweiten Brennpunkts einer Ellipse**
focus2Y	**y-Wert des zweiten Brennpunkts einer Ellipse**
focus2Z	**z-Wert des zweiten Brennpunkts einer Ellipse**
form	**Formfehler**
gap	**Spaltdicke zwischen Kontur und eingepasstem Kreis**
getCamLift	**Maximaler Kurvenhub oder Nockenhub**
getCamLiftAngle	**Winkel (polare Lage) des maximalen Kurvenhubs oder Nockenhubs im Polarkoordinatensystem der Kurve**
getCamLiftAngleAccPnt	**Winkel (polare Lage) des maximalen Nockenhubs. Als Bezug (Winkel = 0) dient der erste Sollpunkt in aufsteigender Richtung. Der abgefragte Winkel entspricht dem Winkel, der bei Ausgabe der Ergebnisse als Textdatei in die Datei geschrieben wird.**
getCamLiftIndex	**Nummer des Sollpunkts mit dem maximalen Kurvenhub**
inclinationAngle	**Kippwinkel**
indexOFmaxDev	**Index von Max-Abweichung**
indexOFminDev	**Index von MIN-Abweichung**
len	**Länge**
length	**Länge einer Kurve**
materialDetected	**“true“, wen Material vorhanden; "false", wenn kein Material vorhanden, “9999“ bei Fehler (nicht definiert)**
maxDev	**maximale Abweichung „+“**
mean	**Mittelwert des Sonderelements Punktemenge**
midPoint	**Mittelpunkt eines Kreises / einer 2d-Geraden**
minDev	**maximale Abweichung „-“**
planeNumber	**Raumachse**
plcn	**Raumachse als Nummer**
radius	**Radius**
radiusD2	**Radius 2**
rotationAngle	**Drehwinkel**
rotX	**Drehwinkel um die X-Achse**
rotY	**Drehwinkel um die Y-Achse**
rotZ	**Drehwinkel um die Z-Achse**
sigma	**Streuung**
toleranceState	**Gibt den Toleranzstatus des Prüfmerkmals zurück**
transX	**Verschiebung in X-Richtung**
transY	**Verschiebung in Y-Richtung**
transZ	**Verschiebung in Z-Richtung**
x	**x-Wert des Referenzpunkts** **x-Wert des Nullpunkts bei Koordinatensystemen und Lochbildern**
xMaxDev	**maximale Abweichung in x-Richtung**
xMinDev	**minimale Abweichung in x-Richtung**
y	**y-Wert des Referenzpunkts** **y-Wert des Nullpunkts bei Koordinatensystemen und Lochbildern**

yMaxDev	**maximale Abweichung in y-Richtung**
yMinDev	**minimale Abweichung in y-Richtung**
z	**z-Wert des Referenzpunkts** **z-Wert des Nullpunkts bei Koordinatensystemen und Lochbildern**
zMaxDev	**maximale Abweichung in z-Richtung**
zMinDev	**minimale Abweichung in z-Richtung**
getMaxCurveJump	**Kurvensprunges**
getOffsetKind	**Toleranzoffset der Kurvenform**
getOffsetValue	**Toleranzform der Kurvenform**
lowTolAtMaxDev	**Kurve**
uppTolAtMaxDev	**Kurve**
lowTolAtMinDev	**Kurve**
uppTolAtMinDev	**Kurve**

Die Toleranzform bei „DIN Position“ RechteckXY liefert 2 Ergebnisse, die Abweichung in x und in y.
Es kann folgendermaßen auf die **einzelnen** Abweichungen zugegriffen werden:

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

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

Zugriff auf die Koordinaten (X,Y,Z und Angle, Radius, Height) jedes einzelnen Kurvenpunktes

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

Zugriff auf die Abweichung jedes einzelnen Kurvenpunktes

getActualCurvePointDevX("Kurvenname"[, index],Punktnummer)

Zugriff auf die X Abweichung jedes einzelnen Kurvenpunktes

getActualCurvePointDevY("Kurvenname"[, index],Punktnummer)

Zugriff auf die Y Abweichung jedes einzelnen Kurvenpunktes

getActualCurvePointDevZ("Kurvenname"[, index],Punktnummer)

Zugriff auf die Z Abweichung jedes einzelnen Kurvenpunktes

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

Zugriff auf die untere Toleranz jedes einzelnen Kurvenpunktes

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

Zugriff auf die obere Toleranz jedes einzelnen Kurvenpunktes

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

Zugriff auf die Koordinate (X,Y,Z,Nx,Ny,Nz und Angle, Radius, Height) jedes einzelnen Freiformflächen Punktes

getActualProfilePointDev("FreiformFlächenForm",Punktnummer)

Zugriff auf die Abweichung jedes einzelnen Freiformflächen Punktes

getCFAttribute

Gibt den aktuellen Wert eines Attributs zurück. Ohne Vorgabe des Prüfmerkmalnamens nur innerhalb von Funktionen verwendbar.

getCFAttribute(attributname)
getCFAttribute(prüfmermalsname,attributname)

getCFGroupname

Gibt den aktuellen höchsten Gruppennamen zum Prüfmerkmal zurück. Ohne Angabe des Prüfmerkmalnamens nur innerhalb von Funktionen verwendbar.

getCFGroupname()
getCFGroupname(“prüfmerkmalsname“)

getCFGroupname2

Gibt den aktuellen tiefsten Gruppennamen zum Prüfmerkmal zurück. Ohne Angabe des Prüfmerkmalnamens nur innerhalb von Funktionen verwendbar.

getCFGroupname2()
getCFGroupname2("prüfmerkmalsname")

getCFNames

Gibt die Prüfmerkmale der Gruppe zurück!

getCFNames("Gruppe") ➔
OrderedCollection ('Dm_Kreis' 'Rundheit1' 'Ebenheit1')

getCornerPointFromCAD("min" oder "max")

Eckpunkte des CAD-Modells

```
eck_klein_z = getCornerPointFromCAD("max").z
eck = getCornerPointFromCAD("min")
klein_X = eck.x
klein_Y = eck.y
klein_Z = eck.z
```

getFunctionObjectName()

Gibt den Namen des aktuellen Prüfmerkmals zurück
(Nur verwendbar zwischen defineFunctionStart und defineFunctionEnd!)

getGroupNames

Gibt alle Gruppennamen aus

getMaxActual([Lochbildname])

getMaxActual("Name der Einpassung").deviation

Zugriff auf Istwert eines Lochbilds oder 2-Punkt-Durchmessers mit dem größten Istwert

2-Punkt-Durchmesser, Maximum: **getMaxActual("Elementname[,Schleifenindex]").actual**

2-Punkt-Durchmesser, Maximumswinkel: **getMaxActual("Elementname[,Schleifenindex]").angleOfRadiusPoint**

getMinActual(Elementname,Schleifenindex)

siehe oben…

Zugriff auf Istwert eines Lochbilds oder 2-Punkt-Durchmessers mit dem kleinsten Istwert

getNominal(Elementname[,Schleifenindex])

getNominal("Kreis1").Eigenschaft

Zugriff auf Sollwerte des Prüfmerkmals/Messelements

getNominal().identifier	Zugriff auf Name des Prüfmerkmals
getNominal().testPlanGeometry.identifier	Zugriff auf Name des Messelements

getNominalCurvePointCoord("Kurvenname", "X" ,Punktnummer)

Zugriff auf die Koordinaten (X,Y,Z,Nx,Ny,Nz und Angle, Radius, Height) jedes einzelnen Kurvenpunktes

getParameterNamed(Parametername[,Schleifenindex])

getParameterName(parametername)

Gibt den Wert desjenigen Parameters zurück, dessen Name der Wert der Zeichenketten-Variablen **parametername** ist.

Zum Zugriff auf Parameterwerte können Sie in Ausdrücken nur fixe Parameternamen verwenden. Mit Hilfe dieser Funktion können Sie den Parameternamen variabel gestalten.

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

getRecheckMode()

Zugriff auf den Modus für Wiederholmessungen

Wiederholmessung ein ➔ true
Wiederholmessung aus ➔ 0

getRunID ()

Abfrage der gewählten Merkmalsgruppe (Run)

getRunID()

getStartSetting
Abfrage der CNC-Startparameter
Beim Anlegen über PCM-Auswahl öffnet sich ein Auswahlfenster

selection	Auswahl
clearOldRes	Alte Resultate rücksetzen
meMode	Reihenfolge Ablauf
naviMode	Fahren zw. Messelemente
runMode	Ablauf-Modus
baseSystemType	Typ Koordinatensystem
baseSystemRealName	Name Koordinatensystem
printer	Drucker
speed	Geschwindigkeit
manAligment	Manuelle Ausrichtung
activeTechnology	Ausführende Strategie

inspectionToleranceState()
liefert den Toleranzstatus des Prüfplans

Rückgabewert	Bedeutung
notDefined	**Status steht noch nicht fest.**
inTolerance	**Prüfplan in Toleranz: Alle Elemente sind innerhalb der angegebenen Toleranzen.**
outOfTolerance	**Prüfplan außer Toleranz: Mindestens ein Element des Prüfplans ist außerhalb der angegebenen Toleranzen.**
outOfLimit	**Prüfplan außerhalb der Grenzen: Mindestens ein Element des Prüfplans ist außerhalb der angegebenen Warngrenzen.**

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

total	**Anzahl der Prüfmerkmale insgesamt**
intolerance	**Anzahl der Prüfmerkmale in Toleranz**
outOfLimit	**Anzahl der Prüfmerkmale über Toleranzgrenze**
outOfTolerance	**Anzahl der Prüfmerkmale außer Toleranz**
noResult	**Anzahl der nicht berechneten Prüfmerkmale**
wpSystem	**Anzahl der Koordinatensysteme**
wpSystemNoResult	**Anzahl der nicht berechneten Koordinatensysteme**

isParameterDefined(Parametername[,Schleifenindex]) **test = isParameterDefined(Hugo) ➔ "true" oder "false"**
Abfrage, ob Parameter zugewiesen (Boolesche Rückgabe)

measure(Elementname[,Schleifenindex]) **measure("Kreis1")**
Messen eines Elements

measuringForce([Elementname oder Kraft,Elementname])
Setzen oder Auslesen der Messkraft **measuringForce(Messkraft in mN,"Messelement")**
Auslesen der Messkraft **measuringForce("Zylinder1")**
Setzen der Messkraft **measuringForce(100,"Zylinder1")**

numberOfPointDistanceExceed(Kurvenformname, Vergleichswert)
Anzahl der Kurvenpunkte mit einer Abweichung größer als

numberOfPointDistanceLessThan(Kurvenformname, Vergleichswert)
Anzahl der Kurvenpunkte mit einer Abweichung kleiner als

outputMultiQdas **outputMultiQdas(attributname)**
Erzeugt eine getrennte QDAS-Auswertung nach vorgebbarem Prüfmerkmals-Attribut.

setBaseSystem("Name des Basissystems") **setBaseSystem("Test1")**
Basissystem dynamisch einbinden, Start mit vorhandener Ausrichtung

setCF(Prüfmerkmalsname[,PMName2,...]) oder (Listenname)
Setzt zu messende Prüfmerkmale neu **setCF(readListFile("C:\tmp\xxx.txt"))**
Ascii-Datei mit Prüfmerkmalsnamen und / oder Gruppennamen
hierbei steht pro Zeile ein PM oder Gruppe, ohne Anführungszeichen und Komma

setCFAttribute(AttributName,Wert)
Attribut für akt. Prüfmerkmal setzen

setCurveJump
setCurveJump("Prüfmerkmal", "KurvenSprungArt", KSBereich, KSToleranz, [Überlappung])

Definieren der Kurvensprungtoleranz (KST): s**etCurveJump("KurvenSprung1","point", 15, 0.05, 0.5)**

Prüfelement = featureElement
KurvenSprungArt = point, angle, length
KSBereich = Wert als Punkteanzahl, Winkelbereich oder Länge des Kurvenbereichs
Toleranz = Toleranzwert
Überlappung = Wert: 0, 0.1, 0.2 ... 0.9, 1.0

setElementStrategy([Messelementname],Strategiename)
Setzen der Mehrfach-Strategie eines Messelements

setFilter(["Prüfmerkmal"],"Prüfelement","Filtertyp","Filterart","Filterkriterium",Wert1,[Wert2], [Seg.verbinden])

setFilter("Durchmesser_Kreis","spline","low","undulation", 50)
setFilter("Ebenheit","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

Wird kein Prüfmerkmal verwendet, so ist statt des Prüfelements ein Messelement zu verwenden.
Bei Ausschalten des Filters mit "off" oder Übernahme der Einstellungen vom Prüfelement mit "fromFeature" können alle weiteren Parameter auch weggelassen werden.

setGeometryInnerOuter("INNER" oder "OUTER")
Setzen der Innen-/Aussen-Kennung eines Messelements

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

setInspectionStrategy(Strategiename)
Setzen der Mehrfach-Strategie des Prüfplans
(leerer Strategiename für Ausweichstrategie)

setLineScanGlobalSetting(Name der Einstellung,Verzeichnis der Einstellungen)
Globales Setzen der LineScan Einstellungen

setMissingBore([Messelementname],Suchdistanz)
Fehlende Bohrung für Messelement setzen

setMissingBore("Kreis1",10)

Hugo = getActual("Kreis1").boreIsMissing ・ true (fehlende Bohrung) , false (vorhandene Bohrung)

setOutlier(["Prüfmerkmal"],"Prüfelement",Faktor i.W,Faktor a.W,"Ber.Datenred.",[Wert],[BisAusgleich.Element],Anz.Iterationen,"Vorfilter",Wert1,Wert2)

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

Prüfelement = featureElement, featureElement1, featureElement2, primaryDatum, secondaryDatum...
Faktor i.W (innerhalb Werkstück) = Zahl, off, fromFeature
Faktor a.W (außerhalb Werkstück) = Zahl
Ber.Datenred. = onlyOutliers, „adjacentPoints“
BisAusgleich.Element = true,false
Anz.Iterationen = Zahl
Vorfilter = undulation, wavelength

Wird kein Prüfmerkmal verwendet, so ist statt des Prüfelements ein Messelement zu verwenden.
Bei Ausschalten des Ausreißers mit "off" oder Übernahme der Einstellungen vom Prüfelement mit "fromFeature" können alle weiteren Parameter auch weggelassen werden.

setParameterNamed(Parametername,Wert[,Index])
Zuweisung mittels Parametername (bei Indexvorgabe Zuweisung in Liste)

settingPut(„useAddCFsForTableFile“.true)
Statistik

setRecheckMode (Recheck[,Reset][,Maximum])
Setzt den Modus für Wiederholmessungen für den Prüfplan

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

Recheck = 0 ➔ Wiederholmessung deaktiviert
Recheck = 1 ➔ Wiederholmessung aktiviert
Reset = 0 ➔ Kein Rücksetzen von Teilnummern
Reset = 1 ➔ Alle Teilnummern werden rückgesetzt
Maximum ➔ Maximale Anzahl der Wiederholmessungen.
Maximum 0 ➔ keine Begrenzung

setRoughnessValue("Prüfmerkmal", "Rauheitsparameter", Sollwert, obere Toleranz [,untere Toleranz])
Setzen des Sollwerts und der Toleranz[en] eines Rauheitsparameters("Rauheit1","Rz",13.0, 2.0):

setRunID(Merkmalsgruppe[,Name2,...])
Neusetzen der gewählten Merkmalsgruppe (od. mehr.)
Eine bisher gesetzte Auswahl wird damit überschrieben. Bei Vorgabe von mehreren Merkmalsgruppen in einem Aufruf bildet die Gesamtmenge der Auswahl den Prüfumfang.

setRunID("Merkmalsgruppenname1")

Kurve

getActualCurveMaxSegDev("Kurvenname",Segmentnummer)
Maximale Abweichung eines tolerierten Kurvensegments

getActualCurveMinSegDev("Kurvenname",Segmentnummer)
Minimale Abweichung eines tolerierten Kurvensegments

getActualCurvePointCoord("Kurvenname","X"[,Schleifenindex],Punktnummer)
Zugriff auf die Koordinaten (X,Y,Z und Angle, Radius, Height) jedes einzelnen Kurvenpunktes

getActualCurvePointDev("Kurvenname"[,Schleifenindex],Punktnummer)
Zugriff auf die Abweichung jedes einzelnen Kurvenpunktes

getActualCurvePointDevX("Kurvenname"[,Schleifenindex],Punktnummer)
Zugriff auf die X Abweichung jedes einzelnen Kurvenpunktes

getActualCurvePointDevY("Kurvenname"[,Schleifenindex],Punktnummer)
Zugriff auf die Y Abweichung jedes einzelnen Kurvenpunktes

getActualCurvePointDevZ("Kurvenname"[,Schleifenindex],Punktnummer)
Zugriff auf die Z Abweichung jedes einzelnen Kurvenpunktes

getActualCurvePointLowTol("Kurvenname"[,Schleifenindex],Punktnummer)
Zugriff auf die untere Toleranz jedes einzelnen Kurvenpunktes

getActualCurvePointUppTol("Kurvenname"[,Schleifenindex],Punktnummer)
Zugriff auf die obere Toleranz jedes einzelnen Kurvenpunktes

getNominalCurvePointCoord("Kurvenname","X",Punktnummer)
Zugriff auf die Koordinaten (X,Y,Z,Nx,Ny,Nz und Angle, Radius, Height) jedes einzelnen Kurvenpunktes

numberOfPointDistanceExceed("Kurvenformname",Vergleichswert)
Anzahl der Kurvenpunkte mit einer Abweichung größer als

numberOfPointDistanceLessThan("Kurvenformname", Vergleichswert)
Anzahl der Kurvenpunkte mit einer Abweichung kleiner als

setCurveJump("Prüfmerkmal", "KurvenSprungArt", KSBereich, KSToleranz, [Überlappung])
Definieren der Kurvensprungtoleranz (KST):

Bsp: setCurveJump("KurvenSprung1", "point", 15, 0.05, 0.5)
Prüfelement = featureElement
KurvenSprungArt = point, angle, length
KSBereich = Wert als Punkteanzahl, Winkelbereich oder Länge des Kurvenbereichs
Toleranz = Toleranzwert
Überlappung = Wert: 0, 0.1, 0.2 ... 0.9, 1.0

writeBliskFinishFile("Bliskname"[,"EXCEL1" [,"EXCEL2"[,"ASCII"]]])
Schreibt die Datei bliskFinish.xml [evtl. mit zusätzlichen Ausgabeparametern] zur Kommunikation mit BladePro.

writeBliskStartFile("Bliskname")
Schreibt die Datei bliskStartup.xml zur Kommunikation mit BladePro

writeCurveDistanceAsPlaneToFile("Prüfmerkmal" ,Ebenenachse 1 oder 2 oder 3, "Dateiname")
Kurvenabstände als Koordinaten für eine Ebene. (Koordinaten der Ebenenachse entspricht der Kurvendistanz)

writeNominalCurve("Kurvenname" [,"Koordinatensystem"])
Export von Kurvensollwerten im ..\workarea\results Ordner in einem beliebigen Koordinatensystem.

KMG-spezifische Funktionen und Fahrbefehle

calculateRotosAngles

Berechnet die ideale Winkelstellung für eine Oberflächenmessung unter Berücksichtigung der Realgeometrie.
Vorraussetzung ist, dass die Oberflächenmessung nur eine Messstrategie beinhaltet.
Es werden 0 (keine Winkelstellung kollisionsfrei), 3 (1 Winkelstellung kollisionsfrei --> a, b, c) oder 6 (2 mögliche Winkelstellungen --> a1, b1, c1, a2, b2, c2) Werte in RAD zurückgegeben.

Parameter 1: Name der Oberflächenmessung

cmmBorder(Achse)

Gibt die KMG-Grenze jeweils einer Achse in Millimeter im Maschinenkoordinatensystem zurück.
Verwendung z. B. cmmBorder("-X").
Mögliche Achsen: -X,+X,-Y,+Y,-Z,+Z

changeStylusNo()

changeStylusNo("1")

Wechsel des aktiven Tasters

changeStylusSystem()

changeStylusSystem("NameTastersystem")

CNC-Wechsel des Tastersystems

cmmCanUseNavigator()

Abfrage, ob KMG NAVIGATOR-fähig

cmmCanUseNavigator() ➔ true / false

cncBreak()

cncBreak()

Abbruch des CNC-Ablaufs

displayPositionCMM()

displayPositionCMM()

Messuhr-Dialog
Koordinaten können auf ”0” gesetzt werden
Abruf über Startpunkt.x, .y, .z

Startpunkt=displayPositionCMM()
Hugo = Startpunk.y

getActualSide()

Aktuelle Sicherheitsebene (zur flexiblen Pfadauswahl bei Tasterwechsel)

getCNCMode()

Aktueller Fahrmodus des Messgeräts

getNextStylusSystemName()

Nächstes Tastersystem bei Tasterwechsel

getPositionCMM()

Aktuelle Position in Gerätekoordinaten

getProbe([Taster,Tastersystem])

Zugriff auf Eigenschaften des Tasters

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

.anglePosA1	Winkel	
.anglePosA2	Winkel	
.calibMode	Modus der Einmessung	
.calibration	Kalibriert ja / nein	
.confName	Name der Konfiguration	
.probeName	Name des Taststifts	
.diameter	Durchmesser	
.radius	Radius	
.probeDate	Datum der letzten Einmessung	
.probeForce	Einmesskraft der Einmessung	
.probeTemp	Temperatur der Einmessung	
.shaftDirection	Schaftrichtung	**getProbe()shaftDirection.x**
.shaftLen	Schaftlänge	
.shaftRadius	Schaftradius	
.stdProbeDev	Streuung der Einmessung	
.probeVector.x	Tasterkoordinate in X	**getProbe(“1“,“Sterntaster“).probeVector.x**
.probeVector.y	Tasterkoordinate in Y	
.probeVector.z	Tasterkoordinate in Z	
.probeType	Tastergeometrie	

baseSystem().machine.machineControl.sphereNormalObject.sphereRadius
Zugriff auf Radius Kugelnormal

baseSystem().machine.machineControl.sphereNormalObject.spherePosition.x
Zugriff auf X-Koordinate Kugelnormal in Maschinenkoordinaten

baseSystem().machine.machineControl.sphereNormalObject.spherePosition.y
Zugriff auf Y-Koordinate Kugelnormal in Maschinenkoordinaten

baseSystem().machine.machineControl.sphereNormalObject.spherePosition.z
Zugriff auf Z-Koordinate Kugelnormal in Maschinenkoordinaten

getProbeRadiusFromActual(Messelementname[,Punktindex[,Elementindex]])
Tasterradius vom Antastpunkt eines gemessenen Elementes

getRackAssignment()
Aktive Magazinbelegung (zur flexiblen Pfadauswahl bei Tasterwechsel)

getRotosAngles
Gibt die Winkelstellung einer Oberflächenmessung zurück (3 Werte in RAD).
Parameter 1: Name der Oberflächenmessung

getRTData()
Abfrage diverser RT-Daten **getRTData("rtId")**
siehe: PCM Befehle für den Drehtisch

getRTOffset()
Zugriff auf den Drehtisch-Offset

getRTPosition()
Zugriff auf den Istwert der letzten Drehtisch-Position

getRTPositionCMM()
Zugriff auf den Istwert der letzten Drehtisch-Position in Gerätekoordinaten

getRunMode()
Abfrage diverser Ablauf-Zustände

getTemperatureCorrection()
Zugriff auf Eigenschaften des Temperaturkorrekturobjekts

Funktion	Rückgabewert
temperatureCorrection	**Boolescher Wert: die Temperaturkorrektur ist aktiv oder nicht aktiv**
.coefficientPart	**Ausdehnungskoeffizient des Bauteils**
.temperaturePart1	**Temperatur des Werkstückfühlers 1**
.temperaturePart2	**Temperatur des Werkstückfühlers 2**
.temperatureProbe	**Temperatur des Temperaturtasters**
.temperatureTableFrontBottom	**Temperatur des Fühlers vorne unten an der Platte**
.temperatureTableFrontTop	**Temperatur des Fühlers vorne oben an der Platte**
.temperatureTableRearBottom	**Temperatur des Fühlers hinten unten an der Platte**
.temperatureTableRearTop	**Temperatur des Fühlers hinten oben an der Platte**

.temperatureX1	**Temperatur des Fühlers 1 am X-Maßstab / Y/ Z**
.temperatureX2	**Temperatur des Fühlers 2 am X-Maßstab / Y/ Z**
.temperatureX	**Temperatur des Maßstabs in X-Richtung**
.temperatureY	**Temperatur des Maßstabs in Y-Richtung**
.temperatureZ	**Temperatur des Maßstabs in Z-Richtung**
.factorX	**Korrekturfaktor des Maßstabs in X-Richtung / Y/ Z**
factorPart	**Korrekturfaktor des Bauteils**
tempWk	**Im CNC-Ablauf verwendeter Temperaturwert des Werkstücks**

Nur noch für PRISMO ultra und ACCURA 2 mit Option TFK

isManProbeChange()
Abfrage auf manuellen Tasterwechsel (zur flexiblen Pfadauswahl bei Tasterwechsel)

lowerableRT()
Senkt den absenkbaren Drehtisch ab oder hebt ihn an.

lowerableRT("UP") ➔ Der LRT wird angehoben.
lowerableRT("DOWN") ➔ Der LRT wird abgesenkt.
lowerableRT("STATUS") ➔ Der Zustand des LRT wird abgefragt.
Der Rückgabewert ist "UP" oder "DOWN".

measureMTMValue()
Messen des Massenträgheitsmoments (Werkstück inkl. RT) und anschl. setzen

measVolumeIlluminationOff()
Deaktiviert die Messraumbeleuchtung(O-Inspect)

measVolumeIlluminationOn()
Aktiviert die Messraumbeleuchtung(O-Inspect)

positionCMM(x,y,z[,Achse1,Achse2,Achse3]) **positionCMM(200,-200,-10,"Z","Y","X")**
Fahrt zu angegebener KMG-Position im Maschinenkoordinatensystem in angegebener Achsenreihenfolge

positionRS(x,y,z[,Koordinatensystem,Taster]) **positionRS(50,60,70, "KS","1_-Z")**
Fahrt zu angegebener Position im angegebenen **positionRS(50,60,70, "","1_-Z")** (Basissystem)
Koordinatensystem mit angegebenem Taster

readActualWPTemperature(leer = selektierte Fühler oder Temp.fühler als Zeichenkette)
z.B. ("1") , ("1+2")
Lesen der Werkstückfühler-Temperatur (nur zur Kontrollausgabe, ohne Berücksichtigung für die Temp.komp.)

readWPTemperature()
Vorzeitiges Lesen der Fühler-Temperatur

rtAxisDefinition(rtmodus)
Werte für rtmodus: "load" oder "measure".

searchDistance(Zahl)
Setzt den Antastsuchweg auf neuen Wert Zahl **searchDistance(5) ➔ 5mm**

setCNCMode(Modus)
Setzen des Fahrmodus des Messgeräts **setCNCMode(manual/cnc)**

setInspectionDrivingSpeed(Geschw. in mm/s)
Allg. Verfahrgeschwindigkeit ändern

setMTMValue(MTM-Wert in kg m²)
Setzen des Massenträgheitsmoments (Werkstück inkl. RT)

setNavigationPath("toProbeChange" "fromProbeChange" , Pfadname)
Auswahl des Pfades vor/nach Tasterwechsel

setNewGeoStatToActualProbe(x,y,z,i,j,k) **setNewGeoStatToActualProbe(0,0,-47,0,0,-1)**
Setzt aktuellen Taster auf neue Werte (x,y,z,nx,ny,nz)

setNewRadiusToActualProbe(r) **setNewRadiusToActualProbe(1.5)**
Setzt aktuellen Taster auf neuen Radius (r)

setRotosAngles(parameter1, parameter2, parameter3, parameter4**)**
Setzt die Winkelstellung einer Oberflächenmessung.
Parameter 1: Name der Oberflächenmessung
Parameter 2-4: Winkel A-C in RAD

setWPTemperature(Werkstücktemp. in Grad Celsius)
Vorgabe der Werkstücktemp. für die Temp.Komp.

stepRS(x,y,z[,Koordinatensystem]) **stepRS(10,0,0, "Koordinatensystem1")**
Schritt um angegebene Koordinaten im angegebenen Koordinatensystem

Systembefehle

actCalypsoWindowName() **CalName = actCalypsoWindowName()**
Akt. Fenstername von CALYPSO

closeSocket()
Schließt Kommunikationskanal zwischen Calypso und externer Anwendung

date([1][2]) **date() ➔ 12. Juli 2023**
Zugriff auf aktuelles Datum **date(1) ➔ 12.07.2023**
date(2) ➔ 2023-07-12

dateAndTime() **dateAndTime() ➔ 12. Juli 2023 09:34:44**
Zugriff auf aktuelles Datum und Uhrzeit **dateAndTime(1)➔ 12.07.2023 09:35:43**
dateAndTime(2)➔ 2023-07-12__09-36-45

dateInNumbers() **dateInNumbers() ➔ 12.7.2023**
Zugriff auf aktuelles Datum mit Zahlenwerten

expandCalypso()
Maximiert das Programmfenster von CALYPSO wieder oder bringt es auf dem Bildschirm nach vorn.

facsIsActive()
Gibt zurück, ob FACS aktiv ist. **facsIsActive() ➔ true / false**

language() **language() ➔ de**
gibt die eingestellte Sprache in Calypso aus

millisecondClockValue() **millisecondClockValue()**
Zugriff auf Millisekunden-Zähler (1/100sek)

minimumLengthOfNumber
Legt die Anzahl der führenden Nullen im Prüfmerkmalnamen bei extractCADPoints auf den angebenen Wert fest

openSocket()
Öffnet Kommunikationskanal zwischen Calypso und externer Anwendung

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

systemCallForResultAccess(Befehl)
Aufruf von Systembefehlen und Warten auf deren Abarbeitung. Externe Anwendung kann auf Calypso-Daten zugreifen

systemCallIcon **systemCallIcon("D:\philips\main\help.bat")**
Ruft System-Befehle auf und wartet auf deren Abarbeitung. Programm wird als Ikone gestartet

systemCallWithWait(Befehl)
Aufruf von Systembefehlen und Warten auf deren Abarbeitung

time() **time() ➔ 09:44:47**
Zugriff auf aktuelle Uhrzeit **time(1) ➔ 09:44:47**
time(2) ➔ 09-44-47

timeInSeconds() **timeInSeconds() ➔ 35292**
Zugriff auf aktuelle Uhrzeit in Sekunden

userDefinedMessage((Wert1, ..2, ..3, ...) oder (Listenname))
Benutzerdefinierte Nachricht an den Observer. Liste mit freiem Inhalt aus Zeichenketten oder Zahlen

wait(nSekunden) **wait(10)**
CNC-Ablauf wird für nSekunden angehalten **➔ Prüfplan wird für 10 Sekunden angehalten**

Präsentationsprotokoll

defineProtocol(Schlüsselwort,Funktionswert(e)) **defineProtocol (Parameter,Wert)**
Flexible Protokoll-Ausgabe mit PCM ➔ **siehe Liste Protokollausgabe**

getRecHdForFil(Protokollkopfvariable)
Gibt den aktuellen Wert einer Protokollkopfvariablen dateinamentauglich zurück
Zeitstempel = getRecHdForFil("time")
➔ **15_16_32**

getRecordHead(Protokollkopfvariable)
Zugriff auf Protokollkopfvariablen **Zeitstempel = getRecordHead("time")**
➔ **15:16:32**

getRecordHeadM(Protokollkopfvariable)
Bei nachträglichem Auswerten: Zugriff auf Protokollkopfvariablen der Messung

getRecordHeadUnchanged(Protokollkopfvariable)
Zugriff auf Protokollkopfvariablen ohne Zeichenersetzung bei Dateidefinitionen

presentationOff()
Präsentationsprotokoll wird nicht angezeigt

presentationOn()
Präsentationsprotokoll wird angezeigt

SetElementsForProtocol
Bestimmt, welche Prüfmerkmale mit dem nachfolgenden Kommando „displayGraphicsWhileSelection“ gedruckt werden.

setProtocolOutput()
Steuert die Protokollausgabe (Listen und Drucken) unabhängig von den CNC-Starteinstellungen. **setProtocolOutput(protokollart,ausgabefunktion)**

protokollart: "compactProtocol", "presentationProtocol", "workingProtocol".
ausgabefunktion: "list" oder "print"

setProtocolSetting()
Zugriff auf Einstellungen des Präsentationsprotokolls
Ausgabeformat ➔ default
setProtocolSetting("outputFormat","default")
Frei definierbare Seiten anhängen
setProtocolSetting("userDefinedPages","userProtocol1.gra")

setProtocolSetting("databaseSyncAfterRunFinished", true)
setProtocolSetting("databaseSyncAfterRunFinished", false)
damit kann man die Datensynchronisation via PCM aktivieren bzw. deaktivieren.

setRecordHead(Protokollkopfvariable,Wert)
Setzen von Protokollkopfvariablen

SetViewsForProtocol()
Bestimmt welche CAD-Ansichten im Protokoll mit ausgegeben werden.
SetViewsForProtocol(viewname1,viewname2,…..)

closeAllPlotts()
schließt alle Protokolle am Schluß eines CNC-Ablaufs, außer die Aktuellen

Liste Protokollausgabe

Parameter	Mögliche Werte		
"#name"	frei definierbare Zeichenkette		
"#type"	"compactProtocol"	"presentationProtocol"	"workingProtocol"
"#format"	"<formatname für protokollkopf>"	"<formatname>"	–
"#elements"	"allME" "allMENom" "allCF" "allCFLimit" "allCFOutTol"	"allCF" "allCFLimit" "allCFOutTol"	–
"#order"	"alphabetic" "elementType" "rangeOfTolAndDev" "likeRun"	"likeList" "alphabetic" "elementType" "rangeOfTolAndDev" "likeRun"	–
"#medium"	"screen" "print" "pdf" "ps"	"screen" "print" "pdf" "ps"	"screen" "print" "pdf" "ps"
"#warningLimitCF"	Prozentzahl		

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

Für den Ausgabeparameter "#medium" ist die Angabe mehrerer Werte möglich:

defineProtocol ("#medium",wert1, ... ,wertn)

Funktionswert	Bedeutung
compactProtocol	Kompaktprotokoll
presentationProtocol	Präsentationsprotokoll
workingProtocol	Arbeitsprotokoll
allME	Alle Messelemente und alle Prüfmerkmale
allMENom	Nur Messelemente mit Prüfmerkmalen
allCF	Alle Prüfmerkmale
allCFLimit	Prüfmerkmale ab Warngrenze
allCFOutTol	Prüfmerkmale außer Toleranz
likeList	Wie Liste
Alphabetic	alphabetisch
elementType	Nach Elementtyp
rangeOfTolAndDev	Nach Abweichnungsgrad
likeRun	Nach Ablauf
Screen	Bildschirm
Print	Drucker
Pdf	PDF-Ausgabe
Ps	PostScript

Sicherheitsgruppen parametrisieren

		Klartext	PCM-Syntax
Sicherheitsebene	➔	SE +Z	SP +Z
Sicherheitsgruppe	➔	SG +Z	GRP +Z
RT-Sicherheitsebene	➔	RTSE +Z	RTSP +Z
RT-Sicherheitsgruppe	➔	RTG +Z	RTG +Z
Eigene Sicherheitsgruppe	➔	SG neu +Z	GRP neu +Z
Eigene Sicherheitsgruppe	➔	Ganz Neu +Z	Ganz Neu +Z

PCM-Syntax wird programmiert, Klartext wird länderspezifisch ausgegeben!

Schleifen

for-next

Schleife ohne Bedingung:

for i=1 to 4 [step 2] \... \next i
bestimmte Schleife

---

for i=1 to 4
Alles was hier steht wird, in diesem Fall, 4 mal wiederholt
next i

```
for i=1 to 4
    message("Das ist Durchlauf Nr: ",i)
next i
```

```
for i=1 to 8 step 2
    message("Das ist Durchlauf Nr: ",i)      //(Es wird jeder 2te Schritt durchgeführt)
next i                                       //( 1 – 3 – 5 – 7 )
```

Schleife mit Bedingung am Ende:

repeat \... \until i>10
bedingte Schleife

repeat
Alles was zwischen repeat und until steht wird solange wiederholt,
bis Bedingung am Ende erfüllt ist.
until

Bsp.: **i=0**
repeat
Aufgabe
i=i+1 (Falls die Aufaddierung des Index fehlt, Gefahr einer Endlos-Schleife)
until i==4

Verknüpfung von mehreren Bedingungen

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))

Bedingung:

if i<10 then \... \endif
einfache Bedingung

---
if
Alles was zwischen if und endif steht wird ausgeführt,
falls die **Bedingung am Anfang** erfüllt ist.
endif

---

if i<10 then \... \else \... \endif
Bedingung mit Alternative

---
if
Alles was zwischen if und else steht wird ausgeführt,
falls die Bedingung am Anfang erfüllt ist.
else
ansonsten wird ausgeführt was zwischen else und endif steht
endif

Bedingung mit mehreren Alternativen:

selectCase i \case 10 \... \[case 1 to 5 \...] \[caseElse \...] \endSelect
Bedingung mit mehreren Alternativen

```
Zahl = 7
selectCase Zahl
        case is <= 0
                Meldung = "Zahl ist zu klein"
        case 1, 2, 3
                Meldung = "Zahl ist kleiner 4"
        case 4 to 7
                Meldung = "Zahl ist zwischen 4 und 7"
        case is >= 8 and (Zahl <= 11)
                Meldung = "Zahl ist 8, 9, 10 oder 11"
        caseElse
                Meldung = "Zahl ist außerhalb des Bereichs"
endSelect
display(Meldung)
```

CAD

clearCAD()
Inhalt des CAD-Fensters löschen

deleteFreeCadEntities()
CAD-Entities werden gelöscht. Nur das CAD-Modell, Koordinatensysteme und Punkte bleiben

deleteTechnologyPoints

extractCADPoints
Extrahiert CAD Punkte

loadCADFile(Dateiname) **loadCADFile("C:\Users\Public\Documents\Zeiss\CALYPSO\cad\cadcube.sat")**
CAD-Datei laden vom Typ *.sab oder *.sat

loadCadFileWithPMI(CadFileName)
Lädt eine CAD-Datei mit PMI

createInspectionFromPMI()
Erstellt Prüfmerkmale und Messelemente aus PMI

moveCADModel(x,y,z) **moveCADModel(0,20,0)**
Verschieben des CAD-Modells um x,y,z Koordinaten

reflectCADModel
Spiegelt das CAD Modell an einer Ebene senkrecht zur angegebenen Richtung (1=X, 2=Y, 3=Z)

reorderBanner() **reorderBanner()**
Neusortieren der Fähnchen

rotateCADModel **rotateCADModel(10,1)**

rotateCADModel(Winkel in Grad[Achse als Zahl 1-3])
Rotieren des CAD-Modells um einen Winkel um eine Achse, die durch die Zahl 1,2,3 vorgegeben wird. Ohne Achsangabe wird um Z gedreht
Achse: 1➔ X; 2➔ Y; 3➔ Z

saveView
Speichern einer CAD-Ansicht

setArrowLength
Setzt die Pfeillänge in Relation zu der Höhe des CAD-Fensters (0-1)

setCoordSysVisible
Darstellen des Koordinatensystems

setFit()
Darstellen aller Elemente in der CAD Ansicht

setShaded() **setShaded()**
Rendern des CAD-Modells

setTarget(x,y,z)
Setzen des Rotationspunktes

setViewDirection(nx,ny,nz)
Setzen der Blickrichtung

setWireframe() **setWireframe()**
Anzeigen des CAD-Modells als Drahtmodell

showBanner() **showBanner()**
Zeigt die Fähnchen für alle Prüfmerkmale an

stitchingCAD(Wert)
StitchingTolerance in mm

Bearbeiten

addCircularPath(Messelement, Messhöhe, Anzahl Messpunkte, Schrittweite, Geschwindigkeit[,Winkelsegment])
Fügt zu einem Messelement eine Strategie Kreissegment hinzu

computeId(Texterweiterung)
Setzt aus Prüfplanname + . + Texterweiterung einen neuen Namen zusammen

deleteCharacteristic(Name des Prüfmerkmals)
Löscht ein Prüfmerkmal

deleteFeature(Name des Messelements)
Löscht ein Messelement

deleteTechnology(Messelement,Index des zu löschenden Technologiesegments)
Löscht ein Technologiesegment eines Messelements

getFeatureElement
Greift auf das Messelement des Prüfmerkmals zu

getTechnology(Messelement)
Liest die Technologie eines Messelements

getTechnologySegment(Technologie, Segmentnummer)
Liest das Segment einer Technologie

meineListe = list(11,22,33,44,55,66,77,88,99)
getTechnologySegment(meineListe,3) ➔ 33

renameFeature(Alter Name, Neuer Name)
Umbenennen Messelement

renameFeature("Kreis7","Test")
([Mit fortlaufender Nummerierung]) ➔ Test1

setGeometryToCircle(Messelement)
Wandelt das Messelement in einen Kreis

setGeometryToCone(Messelement)
Wandelt das Messelement in einen Kegel

setGeometryToCylinder(Messelement)
Wandelt das Messelement in einen Zylinder

setNumberOfPoints(Technologiesegment, Anzahl Punkte)
Setzt die Anzahl Punkte in einem Technologiesegment

setStep(Technologiesegment, Schrittweite)
Setzt die Schrittweite in einem Technologiesegment

Service

serviceCoeffizient(Dateiname,CoeffizientX,CoeffizientY,CoeffizientZ,CoeffizientA,GageId,ThermometerId,StylusId,K1,K2,K3,DifferntMpeNeeded)
Trägt die Angaben in die XML Datei ein

serviceEnd(Dateiname)
Beendet das Schreiben der XML Datei

serviceEndTemperatureProtElement(Dateiname)
Beendet das Schreiben eines TemperaturProtElements

serviceEndTemperatureProtocol(Dateiname)
Beendet das schreiben des Temperaturprotokolls

serviceHeader(Dateiname,Direction,Mode,Position,PositionEnd)
Beginn der XML Datei Position werden als Punkte (3 Koordinaten) übergeben

serviceSingleMeasurement(Dateiname,name,Target,Actual)
Schreiben einer Einzelmessung

serviceStartTemperatureProtElement(Dateiname,TempSensorType)
Start eines TemperatureProtElements bestehend aus mehreren Temperaturen

serviceStartTemperatureProtocol(Dateiname)
Beginn des Temperaturprotokolls

serviceTemperaturelement(Dateiname,Place,TempValue)
Schreiben eines Temperaturelements

PCM Befehle für den Drehtisch

rotateAbsolute(RT-Pos in RAD)
RT drehen auf: Bsp: RT auf 90 Grad drehen**:** **rotateAbsolute(rad(90))**

rotateWithDriveFree(rad(0))
drehen des RT nach vorherigem Freifahren

setRotation(RT-Pos in RAD)
aktuelle RT-Pos setzen auf

alignBSWithRt()
Mechanische Ausrichtung des Drehtisches nach Verdrehwinkel des Basissystem

rotateReference()
Referenzfahrt wird durchgeführt:

rtPositionOffset(Offsetwinkel in RAD) **rtPositionOffset(offsetwinkel in rad)**
Drehtisch zur Positionierung der Werkstücke

getRTOffset()
Ausgabe des RT-Offsets (Winkel nachdrehen Werkstück)

getRTPosition()
auslesen der aktuellen Drehtisch Position

getRTData()
Abfrage diverser RT-Daten **getRTData("rtId")**

rtId	**RT-Kennung**
rtAvailable	**RT vorhanden**
rtActivated	**RT für Prüfplan aktiviert**
rtActivatedOrPassive	**RT für Prüfplan aktiviert oder passiver RT**
passiveRT	**Passiver RT**
mobileRT	**Transportabler RT**
lowerableRT	**Absenkbarer RT**
mtmValue	**Aktueller MTM-Wert**
axisDate	**Datum RT-Achse**
location	**Lage**
orientation	**Orientierung**
wobble	**Taumelwinkel**
eccentricity	**Excenter**
angleToMainAxis	**Winkel zur Hauptachse**

getRTPositionCMM()
auslesen der aktuellen RT-Pos absolut

measureMTMValue()
Messen des Massenträgheitsmoments (Werkstück inkl. RT) und anschl. setzen

setMTMValue(MTM-Wert in kg m²)
Setzen des Massenträgheitsmoments (Werkstück inkl. RT)

lowerableRT()
Senkt den absenkbaren Drehtisch ab oder hebt ihn an.

lowerableRT("UP") ➔ Der LRT wird angehoben.
lowerableRT("DOWN") ➔ Der LRT wird abgesenkt.
lowerableRT() ➔ Der Zustand des LRT wird abgefragt.
Der Rückgabewert ist "UP" oder "DOWN".

rtAxisDefinition(rtmodus)
Werte für rtmodus: "load" oder "measure".

Übersicht: Namen der externen PCM-Dateien

Externe PCM-Dateien werden an bestimmten Stellen des Prüfplanablaufs automatisch ausgeführt, wenn sie einen bestimmten Namen haben und in einem bestimmten Verzeichnis liegen.

Dateien im ...		**werden ausgeführt ...**
Verzeichnis eines Prüfplans	➔	beim Ablauf des zugehörigen Prüfplans
globalen Prüfplanverzeichnis	➔	beim Ablauf jedes Prüfplans

PCM zum Nachschlagen

Der Name der externen Datei ist entscheidend dafür, an welcher Stelle im Prüfplanablauf die externe Datei ausgeführt wird:

Ausführung	**Batch-Datei**	**PCM-Dateiname**
Vor dem Start-Dialog	Inspection_pre_start_dialog.bat	inspection_pre_start_dialog_pcm.txt
Nach dem Laden des Prüfplans	inspection_post_load.bat	inspection_post_load_pcm.txt
Am Beginn des Ablaufs	inspection_start.bat	inspection_start_pcm.txt
Nach dem Ende der Messung; beim Ablauf nach Messelementen vor der Berechnung der Prüfmerkmale	measurement_end.bat	measurement_end_pcm.txt
Vor Ausführung der Eingangsparameter jedes Messelements		plugin_preFeature_pcm.txt
Vor Ausführung der Ausgangsparameter jedes Messelements		plugin_postFeature_pcm.txt
Vor Ausführung der Eingangsparameter jedes Prüfmerkmals		plugin_preCharacteristic_pcm.txt
Vor Ausführung der Ausgangsparameter jedes Prüfmerkmals		plugin_postCharacteristic_pcm.txt
Nach dem Ende der Berechnung, jedoch vor Protokoll-/ Dateiausgabe (nicht bei Ablauf mit einer Auswahl von Messelementen)	calculation_end.bat	calculation_end_pcm.txt
Nach Ende des Ablaufs und der Protokollausgabe	inspection_end.bat	inspection_end_pcm.txt
Nach Fertigstellung aller (auch extern erzeugter) Protokolle und Dateien, soweit sie von CALYPSO gestartet sind (z. B. PDF, DFD/DFX) (nicht bei Ablauf mit einer Auswahl von Messelementen)	report_end.bat	report_end_pcm.txt
Vor dem Speichern des Prüfplans	inspection_pre_save.bat	inspection_pre_save_pcm.txt

Protokollkopfdaten (Referenz)

In der folgenden Tabelle sind alle Protokollkopfdaten aufgeführt. Die "Kennung" wird zur Identifizierung der Protokollkopfdaten in der Tabellenergebnisdatei (...hdr.txt) verwendet.

Datengruppe	Name des Feldes	Kennung
SYS	Prüfer	operid
	Datum	date
	Datum kurz	dateshort
	Uhrzeit	time
	KMG-Typ	dmeid
	Geräte-Gruppe	devicegroup
	KMG-Nr.	dmesn
	Steuerungstyp	controllertyp
	Software	dmeswi
	Software Revision	dmeswv
	Prüfplan-Name	planid
	CT-Datensatz	pointCloudFile
	Ablauf	measRun
	Blatt	actpgnr
	von	nrpgs
	Längenmaßeinheit	lengthunit
	Palettenplatznummer	palletlocationnumber
	Name	vda_name
	Abteilung:	vda_departm
	Telefon/Fax:	vda_phone
	Nr.	vda_number
	Temp. Werkst.	temperatureworkpiece
	Temp. Maßstab X	temperaturescalex
	Temp. Maßstab Y	temperaturescaley
	Temp. Maßstab Z	temperaturescalez
	Temp. Basissyst.	temperaturebasesystem
	Messdauer	durationofrun
	Toleranz-Modus	Deviationmode
	Toleranz-Modus-Key	deviationmodekey
	Anzahl Nachkommastellen	outdecimalunits
	Sprachkennung	languageid
	MCC-Prüfplan-Version	mccplanversion
	MMC/LMC-Modus	mmcmode
Autorun	erster Palettenplatz	firstpalletlocationnumber
	aktueller Palettenplatz	palletlocationnumber
	letzter Palettenplatz	lastpalletlocationnumber
	Palettenplatz	palletlocation
	Palettenkennung	palletid
	Palettenname	

EDIT	**Werkstückname**	**partid**
	Zeichnungsnummer	**drawingno**
	Rev. Werkstück	**partrv**
	Ser.Nr. Werkstück	**partsn**
	Kommentar Prüfplan	**partcomment**
	Spannmittel	**clmpid**
	Spannmittel-Nr	**clmpsn**
	Vorrichtung	**fixtid**
	Vorrichtungs-Nr.	**fixtsn**
	Vorige Operation	**prevop**
	Prüfberichts-Nr.	**vda_auditno**
	Version:	**vda_version**
	Sachnummer:	**vda_subjno**
	Name:	**vda_name**
	Bemerkung:	**vda_remark**
	Name	**partid**
START	**Teilnummer inkremental**	**partnbinc**
	Teil-Nummer	**partnbLong**
	Kommentar bei Start	**startcomment**
	Auftrag	**order**
	Werkzeug	**tooldf**
	Prozessplan	**mfgdev**
	Loskennung	**lotid**
	Prüfungs-Kennung	**procid**
	Palettenplatz	**palletlocation**
	Palettenkennung	**palletid**
	Prüfberichts-Nr.:	**vda_auditno**
	Version:	**vda_version**
	Sachnummer	**vda_subjno**
	Bemerkung:	**vda_remark**
	Name:	**aname**
	Unterschrift:	**vda_signature**
OTHER	**Erstellungsdatum**	**creationdate**
	Änderungsdatum	**changedate**
	Ersteller	**produceoper**
	Änderer	**changeoper**
	Software Rev Erstellung	**creationswi**
	KMG-Typ Erstellung	**creationdme**

Protokollparameter PiWeb		
"operation"	Arbeitsgang	${Qdb.Measurement(1086)}
"order"	Auftrag	${Qdb.Measurement(53)}
"contractingentity"	Auftraggeber	${Qdb.Measurement(1052)}
"vda_remark"	Bemerkung	${Qdb.Measurement(30)}
"manufacturer"	Hersteller	${Qdb.Measurement(1022)}
"cavity"	Kavität	${Qdb.Measurement(41)}
"measRun"	Ablauf	${Qdb.Measurement(10020, -1)}
"startcomment"	Kommentar bei Start	${Qdb.Measurement(9)}
"customer"	Kunde	${Qdb.Measurement(1062)}
"supplier"	Lieferant	${Qdb.Measurement(801)}
"line"	Linie	${Qdb.Measurement(22)}
"lotid"	Los-Kennung	${Qdb.Proberty("ProtocolHeadChargeIdentification")}
"manufacturingmachinename"	Maschinenbezeichnung	${Qdb.Measurement(10)}
"manufacturingmachinenumber"	Maschinennummer	${Qdb.Measurement(10)}
"nestname"	Nestbezeichnung	${Qdb.Measurement(7)}
"nestnumber"	Nestnummer	${Qdb.Measurement(7)}
"palletid"	Palettenkennung	${Qdb.Measurement(40)}
"palletlocation"	Palettenplatz	${Qdb.Measurement(38)}
"vda_auditno"	Prüfberichtsnummer	${Qdb.Proberty("ProtocolHeadAuditNumber")}
"checkreason"	Prüfgrund	${Qdb.Measurement(15)}
"procid"	Prüfungs-Kennung	${Qdb.Measurement(37)}
"partrv"	Rev. Werkstück	${Qdb.Proberty("ProtocolHeadPartVersion")}
"vda_subjno"	Sachnummer	${Qdb.Proberty("ProtocolHeadSubjectNumber")}
"shift"	Schicht	${Qdb.Measurement(850)}
"partsn"	Ser.Nr. Werkstück	${Qdb.Measurement(1001)}
"clmpid"	Spannmittel	${Qdb.Measurement(35)}
"clmpsn"	Spannmittel-Nr	${Qdb.Measurement(36)}
"samplesize"	Stichprobenumfang	${Qdb.Measurement(20)}
"partnbLong"	Teil-Nummer	Teil-Nummer
"partnbinc"	Teilnummer inkremental	${Qdb.Measurement(14)}
"vda_signature"	Unterschrift	${Qdb.Proberty("ProtocolHeadSignature")}
"variant"	Variante	${Qdb.Part(1011)}
"vda_version"	Version	${Qdb.Part(1004)}
"fixtid"	Vorrichtung	${Qdb.Measurement(1114)}
"fixtsn"	Vorrichtungs-Nr	${Qdb.Measurement(17)}
"material"	Werkstoff	${Qdb.Measurement(1032)}
"partid"	Werkstückname	${Qdb.Part(1342)}
"tooldf"	Werkzeug	${Qdb.Measurement(3107)}
"drawingindex"	Zeichnungsindex	${Qdb.Measurement(1043)}
"drawingno"	Zeichnungsnummer	${Qdb.Part(1041)}
	Prüfplanrevision	${Qdb.Measurement(1232,-1)}
"vda_departm"	Abteilung	
"changeoper"	Änderer	
"changedate"	Änderungsdatum	
"outdecimalunits"	Anzahl Nachkommastellen	
"actpgnr"	Blatt	
"pointCloudFile"	CT-Datensatz	
"date"	Datum	
"dateshort"	Datum kurz	
"produceoper"	Ersteller	
"creationdate"	Erstellungsdatum	

"devicegroup"	Geräte-Gruppe	
"dmesn"	KMG-Nr	
"dmeid"	KMG-Typ	
"creationdme"	KMG-Typ Erstellung	
"costcenter"	Kostenstelle	
"partcomment"	Kommentar Prüfplan	
"lengthunit"	Längenmaßeinheit	
"mccplanversion"	MCC-Prüfplan-Version	
"mmcmode"	MCC/LMC-Modus	
"durationofrun"	Messdauer	
"aname"	Name	
"vda_name"	Name:	
"vda_number"	Nr.	
"palletlocationnumber"	Palettenplatznummer	
"mfgdev"	Prozessplan	
"operid"	Prüfer	
"planid"	Prüfplanname	
"dmeswi"	Software	
"creationswi"	Software-Rev Erstellung	
"dmeswv"	Software-Version	
"languageid"	Sprachkennung	
"controllertyp"	Steuerungstyp	
"vda_phone"	Telefon/Fax	
"temperaturebasesystem"	Temp. Basissystem	
"temperaturescalex"	Temp. Maßstab X	
"temperaturescaley"	Temp. Maßstab Y	
"temperaturescalez"	Temp. Maßstab Z	
"temperatureworkpiece"	Temp. Werkstück	
"deviationmode"	Toleranz-Modus	
"deviationmodekey"	Toleranz-Modus-Key	
"time"	Uhrzeit	
"nrpgs"	von	
"prevop"	Vorige Operation	
"company"	Werk	

Qdp.Measurement ➔ messungsbezogene Variable *
Qdp.Property ➔ freie Variable
Qpd.Part ➔ bauteilbezogene Variable *

* zum Suchen & Filtern im PiWeb Reporting
Aus Text ➔ (Variable – Datenquelle – Messung) Auswahl treffen!!

Beschreibung des Attributschlüssels: ${Localization.AttributeKey.Description(xxxx)}
Beschreibung des Textelements: ${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 Wert
Textelement mit Variable ${Qdb.Property("MeasurementDuration")} // Messdauer
${Qdb.Property("NoResultErrorText")} // Fehlermeldung
${Qdb.Property("ISOevalType")} // Symbol für Berechnungsmethode

setProtocolSetting() **setProtocolSetting("databaseSyncAfterRunFinished",true)**
Datensysnchronisation aktivieren bzw. Deaktivieren **setProtocolSetting("databaseSyncAfterRunFinished",false)**

Sonderzeichen

µ	➔	alt 0181
Ø	➔	alt 0216 / alt 157
ø	➔	alt 0248
±	➔	alt 0177 / alt 241
≥	➔	alt 0179 (Symbol)
≤	➔	alt 0163 (Symbol)
•	➔	alt 0149
®	➔	alt 0174
©	➔	alt 0169
™	➔	alt 0153
‰	➔	alt 0137
¼	➔	alt 0188
½	➔	alt 0189
¾	➔	alt 0190
⅓	➔	2153, alt-c
⅛	➔	215b, alt-c
@	➔	alt 0064
√	➔	221a, alt-c
∑	➔	2211, alt-c
℅	➔	2105, alt-c
π	➔	03c0, alt-c
↲	➔	alt 8626
-	➔	alt 0045
„	➔	alt 0034 / alt 34

Strg + Z ➔ Rückgängig
STRG + Y ➔ User Interrupt