CS LAB s.c.
ElectronicLaboratory

simCNC
Steuerungssoftware

Leitfaden zu den Python Scripts

Inhaltsverzeichnis

I. Python - Grundlagen

Die simCNC Steuerungssoftware wurde mit der Unterstützung von Scripts ausgestattet, mit denen Tätigkeiten wie Werkzeugwechsel, Messung der Werkzeuglänge automatisiert werden können. Der Einsatzbereich für Scripts ist nicht auf diese Tätigkeiten beschränkt. Manchmal werden viel kompliziertere Scripts erstellt. Um den Software-Benutzern die Erstellung eigener Scripts zu erleichtern, verwendet simCNC Python als Scriptsprache.

Python ist eine universelle, höhere Programmiersprache mit ausgebautem Paket von Standardbibliotheken, das sich vor allem durch die Lesbarkeit und Klarheit des Quelltextes auszeichnet. Die Syntax der Sprache ist transparent und kompakt.

Die Python Sprache wird sowohl von Einsteigern als auch Professionellen verwendet. Die große Popularität dieser Sprache trägt zur Erstellung zahlreicher Tutorials und Bibliotheken, die auf unzähligen Websites zu finden sind, bei. Damit sind nur unsere Lust und Vorstellungskraft das einzige Hindernis, das uns bei der Erstellung eines eigenen Scripts im Wege stehen kann.

II. SimCNC und Python - Wissenswertes

Bei der Installation der simCNC-Software auf der Festplatte wird Python als separate Software installiert. Die simCNC-Software kommuniziert mit Python über das Netzwerkprotokoll UDP mithilfe von speziell erstellter Bibliothek „___COMM.pyc”. Neben dieser Bibliothek haben die Softwareentwickler auch die Bibliothek „___DEVICE.pyc" erstellt, die einen Bestand an Funktionen zur Steuerung der simCNC-Software mithilfe von Makros beinhaltet.

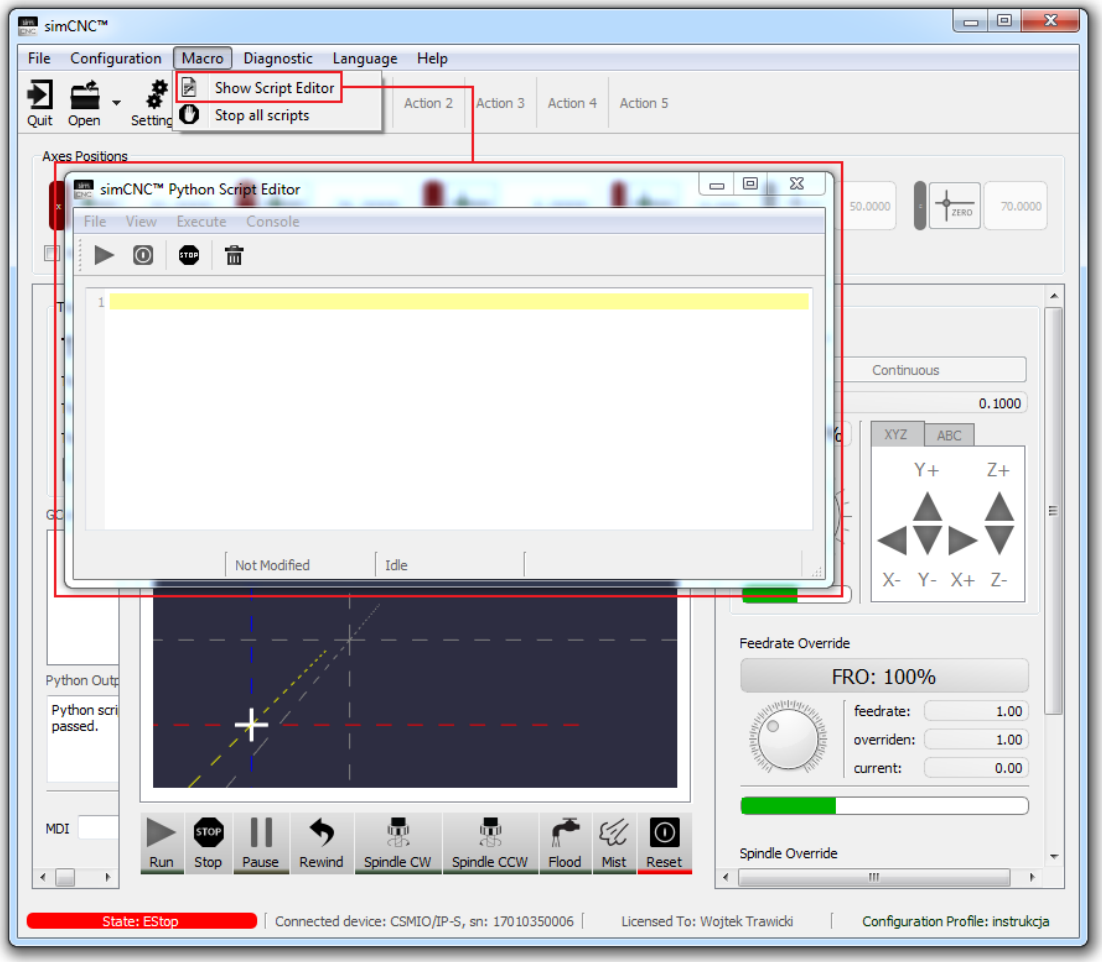
Diese Lösung ermöglicht:

- die Verwendung eines fortgeschrittenen externen Entwicklungseditors mit Debug-Option, z.B. Visual Studio Code,
- die Steuerung der simCNC-Software über UDP im lokalen Netzwerk mithilfe eines anderen PC.

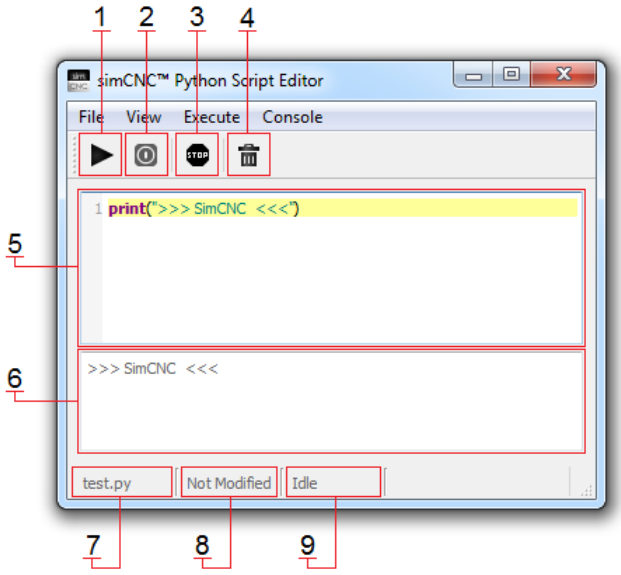
Die beiden obigen Lösungen werden in einer separaten Dokumentation näher besprochen. Diese Anleitung bezieht sich auf den Scripteditor, der in die simCNC-Software integriert wurde.

III. Start und Beschreibung des Python-Scripteditors

Um den Python-Scripteditor zu starten, klicken Sie auf der oberen Fensterleiste der simCNC-Software die Option „Macro“ (Makros), dann wählen Sie in der Dropdown-Liste die Option „Show Script Editor” (Scripteditor anzeigen).



1. „Run” (Start) - Script starten.
2. „Stop” (Stop) - Script stoppen.
3. „Stop Trajectory Planer” (Bewegungsplaner stoppen) - Pendant für die Taste „STOP” auf dem Hauptbildschirm von simCNC.
4. „Clear Console” (Konsole leeren) - entfernt den Fensterinhalt der Python Konsole (siehe Punkt 6).
5. Python Script bearbeiten - hier kann der Script erstellt bzw. bearbeitet werden.
6. Python Konsole - Ort, an dem Informationen zu den Scripts (Fehler und Ausnahmen) sowie eigene Mitteilungen (Beinspiel nebenan) angezeigt werden.
7. Scriptbezeichnung.
8. Information, ob der Script seit dem letzten Speichervorgang geändert wurde.
9. Information über den Scriptstand.



IV. Ctrl + Space - Tipps im Python-Scripteditor

Der Python-Scripteditor wurde mit einem Tippmechanismus ausgestattet. Um ihn zu starten, drücken Sie gleichzeitig die Tasten „Ctrl" + „Space". Diese Kombination kann jederzeit verwendet werden.

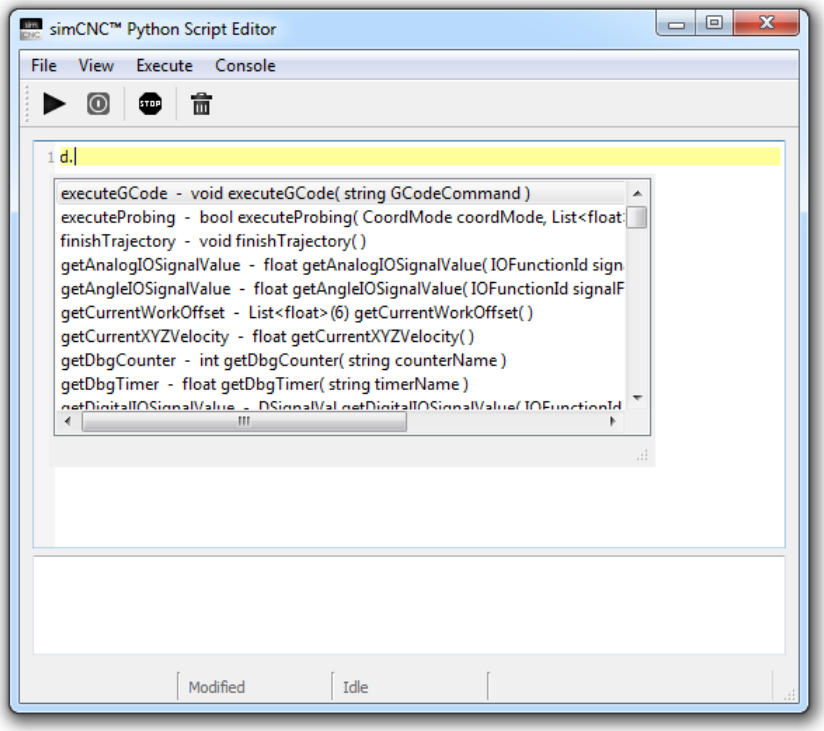
1) Wenn Sie sich die Liste der Funktionen in der Bibliothek „___DEVICE.pyc" ansehen möchten.

Geben Sie „d." ein und drücken Sie die Kombination der Tasten „Ctrl" + „Space". Danach erscheint das Fenster mit der Liste der Funktionen in der Bibliothek „___DEVICE.pyc"

Um eine Funktion zu wählen, beleuchten Sie die Funktion mithilfe der Pfeiltasten und bestätigen Sie mit „Enter" bzw. klicken Sie einfach auf die gewählte Funktion mithilfe der linken Maustaste.

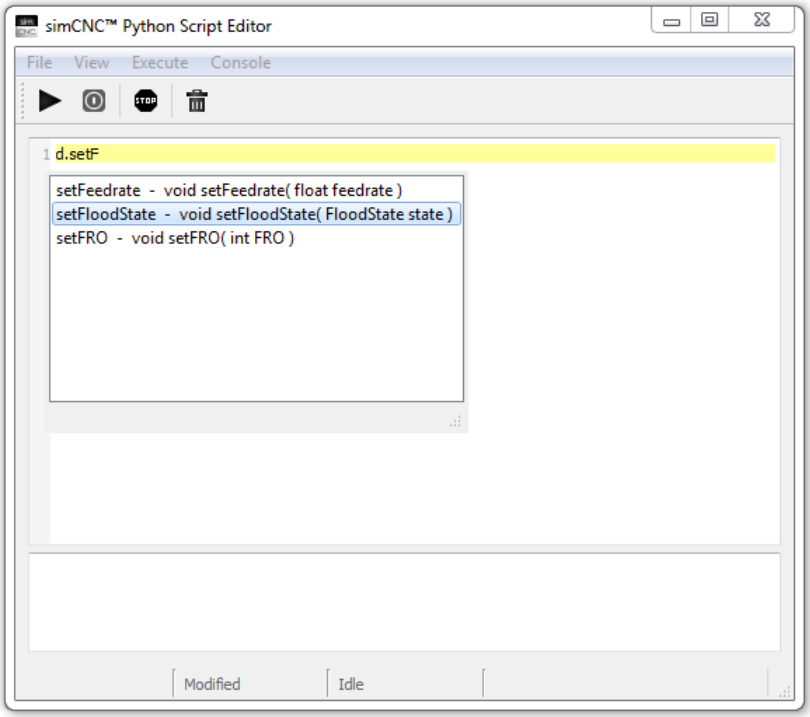
ACHTUNG!

Die erwähnte Liste umfasst nicht die Funktionen, die das Herunterladen des Moduls erfordern. Hier sind Funktionen gemeint, die direkten Zugriff auf digitale und analoge Schnittstellen bieten.



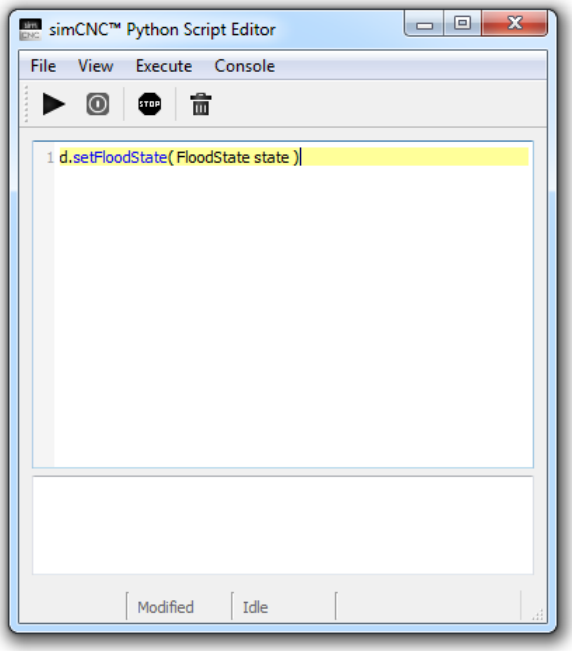
2) Wenn Sie eine unvollständige Funktionsbezeichnung kennen:

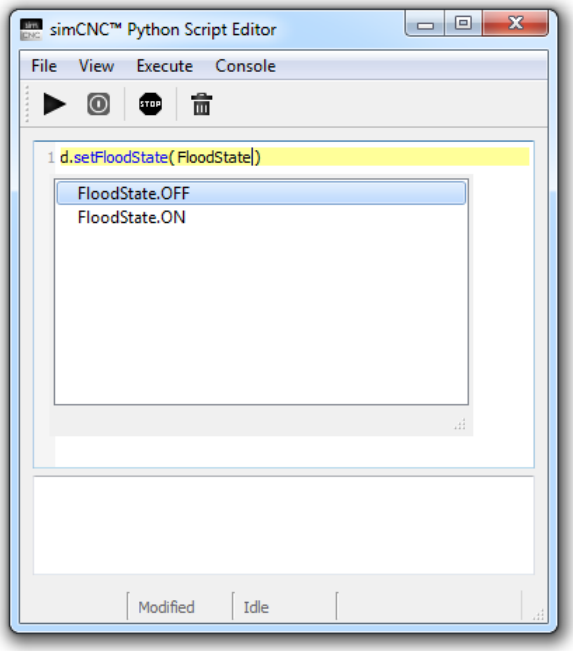
Tragen Sie auch „d.” ein und drücken Sie die Kombination der Tasten „Ctrl” + „Space", diesmal sehen Sie sich nicht die Liste der Funktionen an, sondern tragen Sie den Ihnen bekannten Anfang der Funktionsbezeichnung ein, das Fenster vervollständigt dann die passenden Endungen.



3) Wenn Sie die gewünschte Funktion gefunden haben, aber Sie kennen nicht den Wert des Arguments.

Löschen Sie die Bezeichnung des Arguments und drücken Sie die Kombination der Tasten „Ctrl” + „Space” (falls zuvor nicht betätigt), das Fenster vervollständigt alle zu der jeweiligen Funktion passenden Varianten des Wertes des Arguments.





V. Digitale und analoge Schnittstellen - direkter Zugriff

1) Modul herunterladen.

Um den Zugriff auf digitale oder analoge Schnittstellen der Geräte von CS-LAB zu erhalten, laden Sie zunächst das Modul herunter.

Module d.getModule(ModuleType type, int ID) – Gibt das Modul auf das Gerät zurück.

FUNKTIONSARGUMENTE:

ModuleType type – Gerätetyp

WIR UNTERSCHEIDEN ZWISCHEN:

- ModuleType.IP – Treiber CSMIO/IP-S, CSMIO/IP-A und CSMIO/IP-M
- ModuleType.MPG – Modul für manuelle Handhabung von Achsen CSMIO-MPG
- ModuleType.ENC – Gewindemodul CSMIO-ENC
- ModuleType.IO – Modul von zusätzlichen Ein/Aus CSMIO-IO
- ModuleType.DRV – Antriebe simDrive

int ID – Gerätenummer.

WIR UNTERSCHEIDEN ZWISCHEN:

CSMIO-IO (ID von 0 bis 15)
SimDrive (ID von 0 bis 5)

Falls Sie mehrere Module von zusätzlichen Ein/Aus CSMIO-IO bzw. simDrive Servoantrieben verwenden, geben Sie zur Differenzierung die richtige Gerätenummer ein. Diese Regel gilt nicht für die Steuerungen CSMIO/IP, Modul CSMIO-MPG und CSMIO-ENC. In diesem Falle ist die Gerätenummer immer 0, weil sie im Steuersystem immer einzeln auftreten.

ERGEBNIS DER FUNKTION:

Module - Modul für die Geräte

BEISPIELE:

```
mod_IP = d.getModule( ModuleType.IP, 0 )                  # Modul für den Treiber CSMIO/IP laden
mod_MPG = d.getModule( ModuleType.MPG, 0 )                # Modul für CSMIO-MPG laden
mod_ENC = d.getModule( ModuleType.ENC, 0 )                # Modul für CSMIO-ENC laden
mod_IO_0 = d.getModule( ModuleType.IO, 0 )                # Modul für CSMIO-IO Nummer 0 laden
mod_IO_15 = d.getModule( ModuleType.IO, 15 )              # Modul für CSMIO-IO Nummer 15 laden
mod_simDrive_0 = d.getModule( ModuleType.DRV, 0 )         # Modul für simDrive Nummer 0 laden
mod_simDrive_5 = d.getModule( ModuleType.DRV, 5 )         # Modul für simDrive Nummer 5 laden
```

ACHTUNG!

„mod_IP", „mod_MPG", „mod_ENC", „mod_IO_0", „mod_IO_15", „mod_simDrive_0", und „mod_simDrive_5" stehen für Eigennamen von Modulen (Aufnahmen), die für die Zwecke der Anleitung erfunden wurden. Namen der Module können beliebig sein, aber klar bestimmen, welches Gerät sie betreffen.

2) Digitale Schnittstellen

a) An digitalen Schnittstellen ablesen.

DIOPinVal getDigitalIO(IOPortDir direction, int digitalPin) – Gibt den Zustand von Pin, digitaler Eingangs- bzw. Ausgangsschnittstelle zurück (gibt den Zustand des digitalen Ein- bzw. Ausgangs zurück).

FUNKTIONSARGUMENTE:

IOPortDir direction – Bestimmt die Richtung der Schnittstelle

WIR UNTERSCHEIDEN ZWISCHEN:

IOPortDir.InputPort - Eingangsschnittstelle
IOPortDir.OutputPort - Ausgangsschnittstelle

int digitalPin – Pin-Nummer (Nummer des digitalen Ein- bzw. Ausgangs)

ERGEBNIS DER FUNKTION:

DIOPinVal – Pin-Zustand.

WIR UNTERSCHEIDEN ZWISCHEN:

DIOPinVal.PinReset – Hoher Pin-Zustand
DIOPinVal.PinSet – Tiefer Pin-Zustand

BEISPIELE:

Steuerungen CSMIO/IP-S, CSMIO/IP-A und CSMIO/IP-M

```
mod_IP = d.getModule( ModuleType.IP, 0 )

# Den Zustand des digitalen Eingangs Nr. 8 ablesen
if mod_IP.getDigitalIO( IOPortDir.InputPort, 8 ) == DIOPinVal.PinReset :
    print("CSMIO/IP digital input 8 = 0")
if mod_IP.getDigitalIO( IOPortDir.InputPort, 8 ) == DIOPinVal.PinSet :
    print("CSMIO/IP digital input 8 = 1")
```

```
# Den Zustand des digitalen Ausgangs Nr. 4 ablesen
if mod_IP.getDigitalIO( IOPortDir.OutputPort, 4 ) == DIOPinVal.PinReset :
    print("CSMIO/IP digital output 4 = 0")
if mod_IP.getDigitalIO( IOPortDir.OutputPort, 4 ) == DIOPinVal.PinSet :
    print("CSMIO/IP digital output 4 = 1")
```

CSMIO-MPG - Modul für manuelle Handhabung von Achsen

```
mod_MPG = d.getModule( ModuleType.MPG, 0 )

# Den Zustand des digitalen Eingangs Nr. 3 ablesen
if mod_MPG.getDigitalIO( IOPortDir.InputPort, 3 ) == DIOPinVal.PinReset :
    print("CSMIO-MPG digital input 3 = 0")
if mod_MPG.getDigitalIO( IOPortDir.InputPort, 3 ) == DIOPinVal.PinSet :
    print("CSMIO-MPG digital input 3 = 1")

# Den Zustand des digitalen Ausgangs Nr. 0 ablesen
if mod_MPG.getDigitalIO( IOPortDir.OutputPort, 0 ) == DIOPinVal.PinReset :
    print("CSMIO-MPG digital output 0 = 0")
if mod_MPG.getDigitalIO( IOPortDir.OutputPort, 0 ) == DIOPinVal.PinSet :
    print("CSMIO-MPG digital output 0 = 1")
```

CSMIO-IO - Modul von zusätzlichen Ein/Aus – Nummer 0

```
mod_IO_0 = d.getModule( ModuleType.IO, 0 )

# Den Zustand des digitalen Eingangs Nr. 7 ablesen
if mod_IO _0.getDigitalIO( IOPortDir.InputPort, 7 ) == DIOPinVal.PinReset :
    print("CSMIO-IO 0, digital input 7 = 0")
if mod_IO_0.getDigitalIO( IOPortDir.InputPort, 7 ) == DIOPinVal.PinSet :
    print("CSMIO-IO 0, digital input 7 = 1")

# Den Zustand des digitalen Ausgangs Nr. 2 ablesen
if mod_IO_0.getDigitalIO( IOPortDir.OutputPort, 2 ) == DIOPinVal.PinReset :
    print("CSMIO-IO 0, digital output 2 = 0")
if mod_IO_0.getDigitalIO( IOPortDir.OutputPort, 2 ) == DIOPinVal.PinSet :
    print("CSMIO-IO 0, digital output 2 = 1")
```

CSMIO-IO - Modul von zusätzlichen Ein/Aus – Nummer 15

```
mod_IO_15 = d.getModule( ModuleType.IO, 15 )

# Den Zustand des digitalen Eingangs Nr. 11 ablesen
if mod_IO_15.getDigitalIO( IOPortDir.InputPort, 11 ) == DIOPinVal.PinReset :
    print("CSMIO-IO 15, digital input 11 = 0")
if mod_IO_15.getDigitalIO( IOPortDir.InputPort, 11 ) == DIOPinVal.PinSet :
    print("CSMIO-IO 15, digital input 11 = 1")
```

```
# Den Zustand des digitalen Ausgangs Nr. 5 ablesen
if mod_IO_15.getDigitalIO( IOPortDir.OutputPort, 5) == DIOPinVal.PinReset :
    print("CSMIO-IO 15, digital output 5 = 0")
if mod_IO_15.getDigitalIO( IOPortDir.OutputPort, 5) == DIOPinVal.PinSet :
    print("CSMIO-IO 15, digital output 5 = 1")
```

b) Auf digitale Schnittstelle speichern

void setDigitalIO(int digitalPin, DIOPinVal value) – Stellt den Zustand von Pin, digitaler Schnittstelle ein (stellt den Zustand des digitalen Ausgangs ein).

FUNKTIONSARGUMENTE:

int digitalPin – Pin-Nummer (Nummer des digitalen Ausgangs)
DIOPinVal value – Pin-Zustand

WIR UNTERSCHEIDEN ZWISCHEN:

DIOPinVal.PinReset – Tiefer Pin-Zustand
DIOPinVal.PinSet – Hoher Pin-Zustand

BEISPIEL:

Steuerungen CSMIO/IP-S, CSMIO/IP-A und CSMIO/IP-M

```
import time
mod_IP = d.getModule( ModuleType.IP, 0 )

# Den hohen Zustand am digitalen Ausgang Nr. 1 einstellen
mod_IP.setDigitalIO( 1, DIOPinVal.PinSet )
time.sleep(1)

# Den tiefen Zustand am digitalen Ausgang Nr. 1 einstellen
mod_IP.setDigitalIO( 1, DIOPinVal.PinReset )
```

CSMIO-MPG - Modul für manuelle Handhabung von Achsen

```
import time
mod_MPG = d.getModule( ModuleType.MPG, 0 )

# Den hohen Zustand am digitalen Ausgang Nr. 0 einstellen
mod_MPG.setDigitalIO( 0, DIOPinVal.PinSet )
time.sleep(1)

# Den tiefen Zustand am digitalen Ausgang Nr. 0 einstellen
mod_MPG.setDigitalIO( 0, DIOPinVal.PinReset )
```

CSMIO-IO - Modul von zusätzlichen Ein/Aus – Nummer 0

```
import time
mod_IO_0 = d.getModule( ModuleType.IO, 0 )

# Den hohen Zustand am digitalen Ausgang Nr. 5 einstellen
mod_IO_0.setDigitalIO( 5, DIOPinVal.PinSet )
time.sleep(1)

# Den tiefen Zustand am digitalen Ausgang Nr. 5 einstellen
mod_IO_0.setDigitalIO( 5, DIOPinVal.PinReset )
```

CSMIO-IO - Modul von zusätzlichen Ein/Aus – Nummer 15

```
import time
mod_IO_15 = d.getModule( ModuleType.IO, 15 )

# Den hohen Zustand am digitalen Ausgang Nr. 3 einstellen
mod_IO_15.setDigitalIO( 3, DIOPinVal.PinSet )
time.sleep(1)

# Den tiefen Zustand am digitalen Ausgang Nr. 3 einstellen
mod_IO_15.setDigitalIO( 3, DIOPinVal.PinReset )
```

3) Analoge Schnittstellen

a) An analogen Schnittstellen ablesen

float getAnalogIO(IOPortDir direction, int analogPin) – Gibt den Spannungswert von Pin, analoger Eingangs- bzw. Ausgangsschnittstelle zurück (gibt den Spannungswert des analogen Ein- bzw. Ausgangs zurück).

FUNKTIONSARGUMENTE:

IOPortDir direction – Bestimmt die Richtung der Schnittstelle

WIR UNTERSCHEIDEN ZWISCHEN:

IOPortDir.InputPort	- Eingangsschnittstelle
IOPortDir.OutputPort	- Ausgangsschnittstelle

int analogPin – Pin-Nummer (Nummer des analogen Ein- bzw. Ausgangs)

ERGEBNIS DER FUNKTION:

float – Spannungswert am Pin

BEISPIELE:

Steuerungen CSMIO/IP-S, CSMIO/IP-A und CSMIO/IP-M

```
mod_IP = d.getModule( ModuleType.IP, 0 )

# Den Spannungswert des analogen Eingangs Nr. 0 ablesen
val = mod_IP.getAnalogIO( IOPortDir.InputPort, 0 )
print("CSMIO/IP analog input 0 = " + str(val) + "V")

# Den Spannungswert des analogen Eingangs Nr. 1 ablesen
val = mod_IP.getAnalogIO( IOPortDir.InputPort, 1 )
print("CSMIO/IP analog input 1 = " + str(val) + "V")

# Den Spannungswert des analogen Ausgangs Nr. 0 ablesen
val = mod_IP.getAnalogIO( IOPortDir.OutputPort, 0 )
print("CSMIO/IP analog output 0 = " + str(val) + "V")

# Den Spannungswert des analogen Ausgangs Nr. 1 ablesen
val = mod_IP.getAnalogIO( IOPortDir.OutputPort, 1 )
print("CSMIO/IP analog output 1 = " + str(val) + "V")
```

CSMIO-MPG - Modul für manuelle Handhabung von Achsen

```
mod_MPG = d.getModule( ModuleType.MPG, 0 )

# Den Spannungswert des analogen Eingangs Nr. 0 ablesen
val = mod_MPG.getAnalogIO( IOPortDir.InputPort, 0 )
print("CSMIO-MPG analog input 0 = " + str(val) + "V")

# Den Spannungswert des analogen Eingangs Nr. 1 ablesen
val = mod_MPG.getAnalogIO( IOPortDir.InputPort, 1 )
print("CSMIO-MPG analog input 1 = " + str(val) + "V")
```

b) Auf digitale Schnittstellen speichern

void setAnalogIO(int analogPin, float value) – Stellt die Spannung am Pin der analogen Schnittstelle ein.

FUNKTIONSARGUMENTE:

int analogPin – Pin-Nummer (Nummer des analogen Ausgangs)
float value – Spannungswert am Pin

BEISPIELE:

Steuerungen CSMIO/IP-S, CSMIO/IP-A und CSMIO/IP-M

```
mod_IP = d.getModule( ModuleType.IP, 0 )

# Den Spannungswert 5V am analogen Ausgang Nummer 0 einstellen
val = 5
mod_IP.setAnalogIO( 0, val )

# Den Spannungswert 2V am analogen Ausgang Nummer 1 einstellen
val = 2
mod_IP.setAnalogIO( 1, val )
```

ACHTUNG!

Treiber CSMIO/IP-S, CSMIO/IP-A, CSMIO/IP-M

Die Spannung an den Pins der analogen Eingangs- und Ausgangsschnittstellen kann im Bereich von 0 bis 10 V mit einer Frequenz von 12 bit liegen.

Modul für manuelle Handhabung von Achsen CSMIO-MPG

Die Spannung an den Pins der analogen Eingangsschnittstelle, Modul für manuelle Handhabung von Achsen CSMIO-MPG, kann im Bereich von 0 bis 5V mit einer Frequenz von 12 bit liegen.

4) Encoder-Schnittstellen - CSMIO-ENC

a) Den Winkel des Encoders ablesen

float getEncoderIOAngle(int channel = 0) - Gibt den Winkel des Encoders, gemessen ab Signal index, zurück.

FUNKTIONSARGUMENTE:

int channel - Encoderkanal

ERGEBNIS DER FUNKTION:

float - Encoderwinkel

BEISPIEL:

```
mod_ENC = d.getModule( ModuleType.ENC, 0)

# Den Winkel des Encoders Nummer 0 ablesen
val = mod_ENC.getEncoderIOAngle( 0 )
print("CSMIO-ENC encoder chanel 0 angle = " + str(val))
```

b) Die Position des Encoders ablesen

int getEncoderIOPostion(int channel = 0) - Gibt die Position des Encoders, gemessen ab erstem Signal index, zurück.

FUNKTIONSARGUMENTE:

int channel - Encoderkanal

ERGEBNIS DER FUNKTION:

int - Position des Encoders

BEISPIEL:

```
mod_ENC = d.getModule( ModuleType.ENC, 0 )

# Die Position des Encoders Nummer 2 ablesen
val = mod_ENC.getEncoderIOPosition( 2 )
print("CSMIO-ENC encoder chanel 0 postion = " + str(val))
```

c) Die Geschwindigkeit des Encoders ablesen

float getEncoderIORPM(int channel = 0) – Gibt die Drehgeschwindigkeit des Encoders (RPM) zurück.

FUNKTIONSARGUMENTE:

int channel - Encoderkanal

ERGEBNIS DER FUNKTION:

float - Drehgeschwindigkeit des Encoders

BEISPIEL:

```
mod_ENC = d.getModule( ModuleType.ENC, 0)

# Die Drehgeschwindigkeit des Encoders Nummer 4 ablesen
val = mod_ENC.getEncoderIORPM( 4 )
print("CSMIO-ENC encoder chanel 0 rpm = " + str(val))
```

ACHTUNG!

Die Kanäle des Gewindemoduls mit den Nummern 0, 2 und 4 sind physikalische Kanäle, die Kanäle 1, 3 und 5 sind virtuelle Kopien physikalischer Kanäle, die zukünftig verwendet werden.

Bis zum Widerruf entsprechen die erwähnten physikalischen Kanäle 0, 2 und 4 den Kanälen mit den Nummern 0, 1 und 2 in der simCNC Konfiguration.

VI. Koordinaten der Achsen ablesen und speichern

1) Koordinaten ablesen

List<float>(6) getPosition(CoordMode coordMode) - Gibt den aktuellen Wert der Maschinen- bzw. Programmkoordinaten aller Achsen zurück.

FUNKTIONSARGUMENTE:

CoordMode coordMode - Bestimmt die Art der Koordinaten

WIR UNTERSCHEIDEN ZWISCHEN:

CoordMode.Machine - Maschinenkoordinaten
CoordMode.Program - Programmkoordinaten

ERGEBNIS DER FUNKTION:

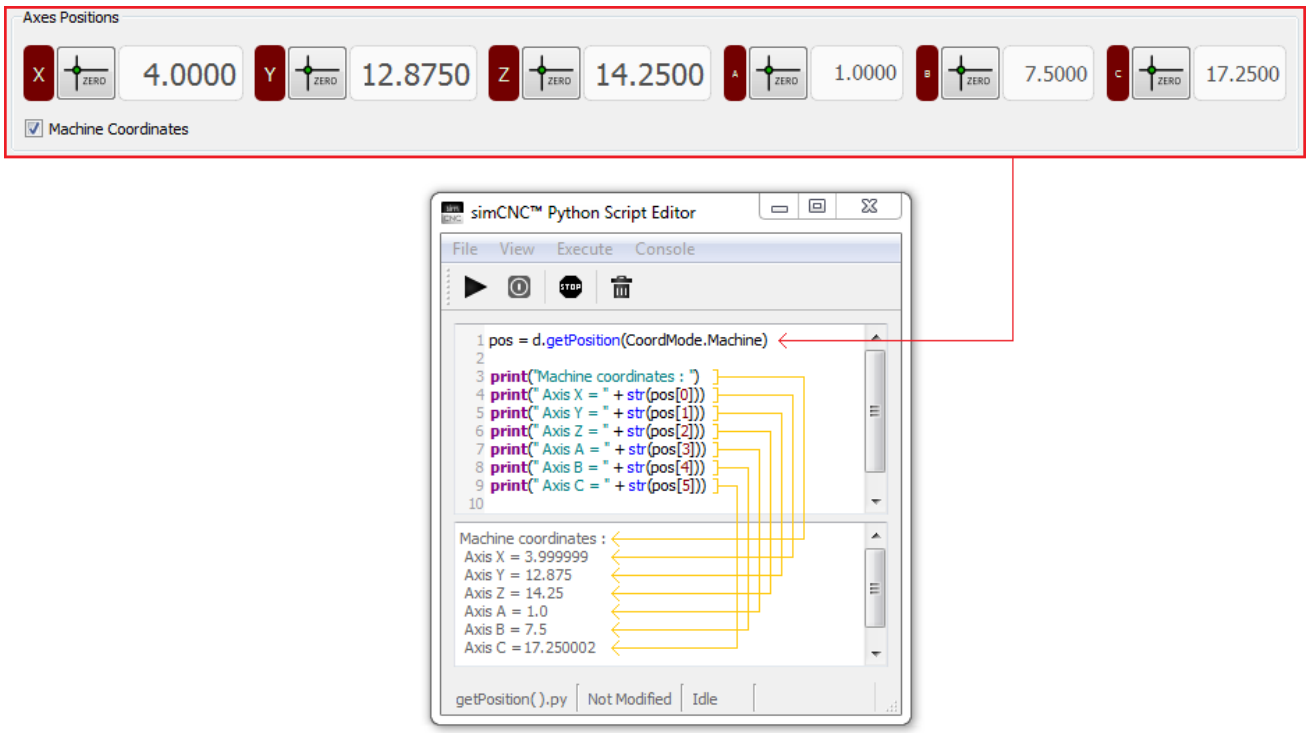
List<float>(6) - Liste von 6 Koordinaten (X, Y, Z, A, B, C) der aktuellen Position

BEISPIELE:

Maschinenkoordinaten der Achsen X, Y, Z, A, B und C ablesen

```
pos = d.getPosition( CoordMode.Machine )

print("Machine coordinates : ")
print(" Axis X = " + str(pos[0]))
print(" Axis Y = " + str(pos[1]))
print(" Axis Z = " + str(pos[2]))
print(" Axis A = " + str(pos[3]))
print(" Axis B = " + str(pos[4]))
print(" Axis C = " + str(pos[5]))
```

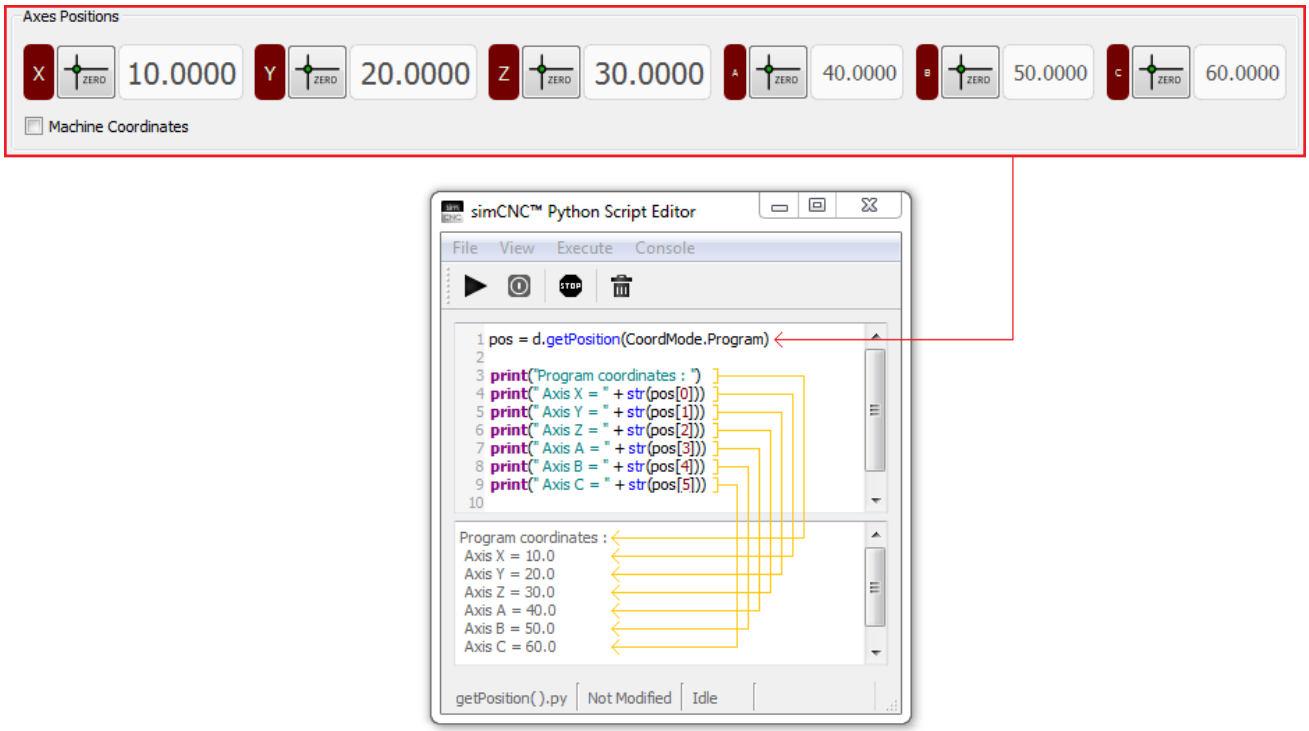
Wenn Sie das obige Beispiel aktivieren, liest die Funktion „d.getPosition(CoordMode.Machine)" die Maschinenkoordinaten aller 6 Achsen ab und ordnet sie in einer aus 6 Elementen bestehenden, von 0 bis 5 indexierten Liste. Die in der Liste hinterlegten Werte der Maschinenkoordinaten werden dann mithilfe der Funktion „print( )" in der Python Konsole angezeigt.



Programmkoordinaten der Achsen X, Y, Z, A, B und C ablesen

```
pos = d.getPosition( CoordMode.Program )

print("Program coordinates : ")
print(" Axis X = " + str(pos[0]))
print(" Axis Y = " + str(pos[1]))
print(" Axis Z = " + str(pos[2]))
print(" Axis A = " + str(pos[3]))
print(" Axis B = " + str(pos[4]))
print(" Axis C = " + str(pos[5]))
```

Wenn Sie das obige Beispiel aktivieren, liest die Funktion „d.getPosition(CoordMode.Machine)" die Maschinenkoordinaten aller 6 Achsen ab und ordnet sie in einer aus 6 Elementen bestehenden, von 0 bis 5 indexierten Liste. Die in der Liste hinterlegten Werte der Maschinenkoordinaten werden dann mithilfe der Funktion „print( )" in der Python Konsole angezeigt.



2) Koordinaten speichern

void setAxisProgPosition(Axis axis, float value) - Stellt den Wert der Programmkoordinaten der gewählten Achse ein.

FUNKTIONSARGUMENTE:

Axis axis - Achse, für die der neue Wert der Programmkoordinaten gespeichert wird

WIR UNTERSCHEIDEN ZWISCHEN:

- Axis.X – Oś X
- Axis.Y – Oś Y
- Axis.Z – Oś Z
- Axis.A – Oś A
- Axis.B – Oś B
- Axis.C – Oś C

float value - Neuer Wert der Programmkoordinaten

BEISPIELE:

Programmkoordinaten der Achsen X, Y, Z, A, B und C speichern

```
x = 10
y = 20
z = 30
a = 40
b = 50
c = 60

d.setAxisProgPosition( Axis.X, x )
d.setAxisProgPosition( Axis.Y, y )
d.setAxisProgPosition( Axis.Z, z )
d.setAxisProgPosition( Axis.A, a )
d.setAxisProgPosition( Axis.B, b )
d.setAxisProgPosition( Axis.C, c )
```

Das Gleiche mit Verwendung der Liste

```
pos = (10, 20, 30, 40, 50, 60)

d.setAxisProgPosition(Axis.X, pos[0])
d.setAxisProgPosition(Axis.Y, pos[1])
d.setAxisProgPosition(Axis.Z, pos[2])
d.setAxisProgPosition(Axis.A, pos[3])
d.setAxisProgPosition(Axis.B, pos[4])
d.setAxisProgPosition(Axis.C, pos[5])
```

VII. Maschinenachsen bewegen

void moveAxisIncremental(Axis axis, float distance, float velocity) - Bewegt die gewählte Achse um die vorgegebene Strecke (inkrementelle Bewegung) mit der vorgegebenen Geschwindigkeit.

ARGUMENTE:

Axis axis - Achse, die sich bewegt

WIR UNTERSCHEIDEN ZWISCHEN:

- Axis.X - X-Achse
- Axis.Y - Y-Achse
- Axis.Z - Z-Achse
- Axis.A - A-Achse
- Axis.B - B-Achse
- Axis.C - C-Achse

float distance - Bewegungsstrecke
float velocity - Bewegungsgeschwindigkeit

BEISPIELE:

```
d.moveAxisIncremental( Axis.X, 20, 1000 )
d.moveAxisIncremental( Axis.Y, 20, 1000 )
d.moveAxisIncremental( Axis.X, -20, 1000 )
d.moveAxisIncremental( Axis.Y, -20, 1000 )
```

Wenn Sie das obige Beispiel aktivieren, bewegen sich die Achsen X und Y von der aktuellen Position aus am Umfang eines Quadrats mit einer Seite von 20 mm bzw. Zoll mit einer Geschwindigkeit von 1.000 mm/Min. bzw. Zoll/Min.

void moveToPosition(CoordMode coordMode, List<float>[6] position, float velocity) - Bewegt sich an die vorgegebene Position, die in den Maschinen- bzw. Programmkoordinaten mit der vorgegebenen resultierenden Geschwindigkeit angegeben ist.

ARGUMENTE:

CoordMode coordMode - Bestimmt die Art der Koordinaten

WIR UNTERSCHEIDEN ZWISCHEN:

- CoordMode.Machine - Maschinenkoordinaten
- CoordMode.Program - Programmkoordinaten

List<float>[6] position - Liste von 6 Koordinaten (X, Y, Z, A, B, C) der Zielposition
float velocity - Die vorgegebene resultierende Geschwindigkeit

BEISPIELE:

```
X = 0
Y = 1
pos = d.getPosition(CoordMode.Program)

pos[X] = pos[X] + 20
d.moveToPosition( CoordMode.Program, pos, 1000 )
pos[Y] = pos[Y] + 20
d.moveToPosition( CoordMode.Program, pos, 1000 )
pos[X] = pos[X] - 20
d.moveToPosition( CoordMode.Program, pos, 1000 )
pos[Y] = pos[Y] - 20
d.moveToPosition( CoordMode.Program, pos, 1000 )
```

Wenn Sie das obige Beispiel aktivieren, bewegen sich die Achsen X und Y mithilfe der Programmkoordinaten von der aktuellen Position aus am Umfang eines Quadrats mit einer Seite von 20 mm bzw. Zoll mit einer Geschwindigkeit von 1.000 mm/Min. bzw. Zoll/Min.

```
X = 0
Y = 1
pos = d.getPosition(CoordMode.Machine)

pos[X] = pos[X] + 20
d.moveToPosition( CoordMode.Machine, pos, 1000 )
pos[Y] = pos[Y] + 20
d.moveToPosition( CoordMode.Machine, pos, 1000 )
pos[X] = pos[X] - 20
d.moveToPosition( CoordMode.Machine, pos, 1000 )
pos[Y] = pos[Y] - 20
d.moveToPosition( CoordMode.Machine, pos, 1000 )
```

Wenn Sie das obige Beispiel aktivieren, bewegen sich die Achsen X und Y mithilfe der Programmkoordinaten von der aktuellen Position aus am Umfang eines Quadrats mit einer Seite von 20 mm bzw. Zoll mit einer Geschwindigkeit von 1.000 mm/Min. bzw. Zoll/Min.

Die beiden obigen Beispiele zeigen, wie die Listen zur Veränderung der Koordinaten genutzt werden können. Da Sie diese Berechnung sehr oft in Scripts sehen können, die auf der Seite CS-LAB veröffentlicht werden, ist es jetzt schon beachtenswert.

void parallelMoveToPosition(CoordMode coordMode, Axis axis, float position, float velocity) - Bewegt die gewählte unabhängige Achse an die vorgegebene Position, die in den Maschinen- bzw. Programmkoordinaten angegeben ist, mit der vorgegebenen Geschwindigkeit. Die Funktion funktioniert, wenn die Option „Use external points" („Steuerung mithilfe des Bewegungsplaners") ausgewählt wurde.

ARGUMENTE:

CoordMode coordMode - Bestimmt die Art der Koordinaten

WIR UNTERSCHEIDEN ZWISCHEN:

- CoordMode.Machine – Maschinenkoordinaten
- CoordMode.Program – Programmkoordinaten

Axis axis - Die unabhängige Achse, die sich bewegt

WIR UNTERSCHEIDEN ZWISCHEN:

- Axis.X - X-Achse
- Axis.Y - Y-Achse
- Axis.Z - Z-Achse
- Axis.A - A-Achse
- Axis.B - B-Achse
- Axis.C - C-Achse

float position - Zielposition der Achse
float velocity - Die vorgegebene Geschwindigkeit

BEISPIELE:

```
X = 0
pos = d.getPosition(CoordMode.Program)

d.parallelMoveToPosition( CoordMode.Program, Axis.X, 25, 100 )
d.parallelMoveToPosition( CoordMode.Program, Axis.X, pos[X], 20 )
```

Wenn Sie das obige Beispiel aktivieren, bewegt sich die X-Achse mithilfe der Programmkoordinaten an die Position 25 mm bzw. Zoll mit einer Geschwindigkeit von 100 mm/Min. bzw. Zoll/Min., dann kehrt sie zum Startpunkt mit einer Geschwindigkeit von 20 mm/Min. bzw. Zoll/Min.

```
Z = 2
pos = d.getPosition(CoordMode.Program)

d.parallelMoveToPosition( CoordMode. Machine, Axis.Z, 78, 350 )
d.parallelMoveToPosition( CoordMode. Machine, Axis.Z, pos[Z], 500 )
```

Wenn Sie das obige Beispiel aktivieren, bewegt sich die Z-Achse mithilfe der Maschinenkoordinaten an die Position 78 mm bzw. Zoll mit einer Geschwindigkeit von 350 mm/Min. bzw. Zoll/Min., dann kehrt sie zum Startpunkt mit einer Geschwindigkeit von 500 mm/Min. bzw. Zoll/Min.

float getCurrentXYZVelocity( ) - Gibt die aktuelle resultierende Geschwindigkeit der Achsen X, Y und Z.

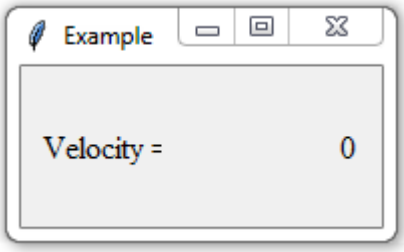
ERGEBNIS DER FUNKTION:

float - Resultierende Geschwindigkeit

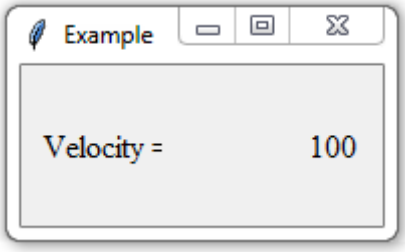
BEISPIELE:

```
from tkinter import *

def show():
    velocity = d.getCurrentXYZVelocity( )
    L2["text"] = str(int(velocity))
    window.after(5, show)

window = Tk()
window.geometry("180x80")
window.title( "Example" )

L1 = Label(window, text="Velocity =", font= ("Times New Roman",12))
L1.place(x=10, y=30, height=20, width=65)

L2 = Label(window, font= ("Times New Roman",12), anchor = "e")
L2.place(x=70, y=30 ,height=20, width=100)

window.after(5, show)
mainloop()
```

Wenn Sie das obige Beispiel aktivieren, zeigt das Script das Fenster mit der aktuellen resultierenden Geschwindigkeit der Achsen X, Y und Z an. Nebenan Bild mit der Situation, in der sich keine der Achsen bewegt.



Um zu überprüfen, ob. das Script funktioniert, wählen Sie die Option „Steuerung mit der Tastatur" aus bzw. verwenden Sie die Tastenkombination „Alt + j" und bewegen Sie die Achse bzw. Achsen mithilfe der Pfeiltasten.



VIII. Sondierung

bool executeProbing(CoordMode coordMode, List<float>[6] position, int probeIndex, float velocity) - Führt die Sondierung mithilfe der gewählten Sonde an die vorgegebene Position, die in den Maschinen- bzw. Programmkoordinaten angegeben ist, mit der vorgegebenen resultierenden Geschwindigkeit durch.

ARGUMENTE:

CoordMode coordMode - Bestimmt die Art der Koordinaten

WIR UNTERSCHEIDEN ZWISCHEN:

CoordMode.Machine - Maschinenkoordinaten
CoordMode.Program - Programmkoordinaten

List<float>[6] position - Liste von 6 Koordinaten (X, Y, Z, A, B, C) der Zielposition
int probeIndex - Nummer der Sonde (Konfigurationen der Sonden finden Sie in Settings/Modules/Special IO)
float velocity - Resultierende Geschwindigkeit

ERGEBNIS DER FUNKTION:

bool - Ergebnis der Sondierung

WIR UNTERSCHEIDEN ZWISCHEN:

True - Falls die Sonde angesprochen hat
False - Falls die Achse bzw. Achsen ihre Zielposition erreicht haben und die Sonde nicht angesprochen hat

BEISPIELE finden Sie in der Beschreibung der nächsten Funktion

List<float>[6] getProbingPosition(CoordMode coordMode) - Gibt die Position, in der die Sonde angesprochen hat, bzw. die Zielposition, die in den Maschinen- bzw. Programmkoordinaten angegeben ist, zurück. Die Funktion gibt die Zielposition nur dann zurück, wenn die Sonde bei der Sondierung nicht angesprochen hat.

ARGUMENTE:

CoordMode coordMode - Bestimmt die Art der Koordinaten

WIR UNTERSCHEIDEN ZWISCHEN:

CoordMode.Machine - Maschinenkoordinaten
CoordMode.Program - Programmkoordinaten

ERGEBNIS DER FUNKTION:

List<float>[6] - Liste von 6 Koordinaten (X, Y, Z, A, B, C) der Position, in der die Sonde anspricht, bzw. der Zielposition.

BEISPIELE für die beiden obigen:

```
Probing_vel = 10
Return_vel = 100
Distance = 20

Starting_position = d.getPosition( CoordMode.Program )
Maximum_position = Starting_position.copy()
Maximum_position[2] -= Distance

if ( d.executeProbing( CoordMode.Program, Maximum_position, 0, Probing_vel ) == False ):
    d.moveToPosition( CoordMode.Program, Starting_position, Return_vel )
    msg.err( "Probing Failed !!!" , "Inactive probe" )
else:
    d.moveToPosition( CoordMode.Program, Starting_position, Return_vel )
    Probe_position = d.getProbingPosition( CoordMode.Program )
    msg.info( "Position = " + str(Probe_position[2]),"Probing - Z axis" )
```

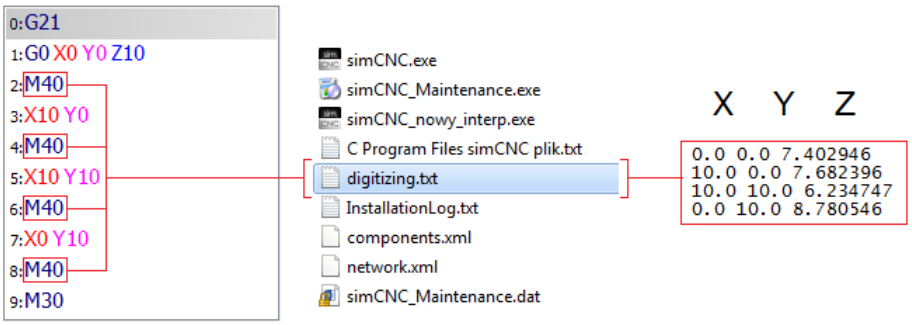
Das obige Script zeigt, wie die Sondierung mithilfe der Z-Achse von der aktuellen Position aus auf der durch den Parameter „Distance" durchgeführt werden kann. Nach Beendigung der Sondierung kehrt die Z-Achse unabhängig vom Ergebnis der Sondierung immer zur Startposition zurück. Der Parameter „Probing_vel" ist für die Geschwindigkeit der Sondierung und der Parameter „Return_vel" für die Geschwindigkeit der Rückkehr der Sonde zur Startposition verantwortlich. Das Ergebnis der Sondierung wird mithilfe des erscheinenden Fensters dargestellt.

Einfache Methode zur Digitalisierung!

Das vorher präsentierte Script kann nach kleiner Änderung als Makro fungieren, das ein Punktraster der sondierten Fläche speichert. Das Speichern des Punktrasters wird meistens beim Kopieren des Flachreliefs, Fräsen oder Gravieren auf unebener Oberfläche, z.B. Steinoberfläche, verwendet.

```
import sys

File_path = "digitizing.txt"                    #C:\Program Files\simCNC\digitizing.txt

try:
    File = open(File_path, "a+")
except IOError:
    sys.exit("\n No access to file !!!")

Probing_vel = 100
Return_vel = 1000
Distance = 20
Starting_position = d.getPosition( CoordMode.Program )
Maximum_position = Starting_position.copy()
Maximum_position[2] -= Distance

if ( d.executeProbing( CoordMode.Program, Maximum_position, 0, Probing_vel ) == False ):
    d.moveToPosition( CoordMode.Program, Starting_position, Return_vel )
    Saved_text ="Probing Failed !!! \n"
else:
    d.moveToPosition( CoordMode.Program, Starting_position, Return_vel )
    Probe_position = d.getProbingPosition( CoordMode.Program )
    Saved_text = str(Probe_position[0]) + " " + str(Probe_position[1]) + " " + str(Probe_position[2]) + "\n"

try:
    File.write(Saved_text)
except IOError:
    File.close()
    sys.exit("\n File write error !!!")
```

Speichern Sie nur das obige Script z.B. unter M40 und rufen Sie es mithilfe des gcod in Abständen ab. Wenn das Script abgerufen wird, führt es die Sondierung durch, dann speichert es die Position, in der die Sonde angesprochen hat, sowie die Position der Achsen X und Y in der Datei C:\Program Files\simCNC\digitizing.txt. Nachstehend können Sie das Ergebnis der Makrofunktion am Beispiel eines einfachen gcod sehen, der eine Bewegung am Umfang eines Quadrats mit den Massen 10 x 10 bildet.



Falls die Sonde auf der durch den Parameter festgelegten „Distance" nicht anspricht, wird in der Datei „digitizing.txt" der Eintrag „Probing Failed !!!" gespeichert.

```
0.0 0.0 6.716997
10.0 0.0 7.318946
Probing Failed !!!
0.0 10.0 8.143046
```

VIII. Werkzeuglager

1) Werkzeugnummer.

int getSelectedToolNumber( ) - Gibt die Nummer des gewählten Werkzeugs mithilfe des gcod, MDI-Zeile bzw. Script Python zurück.

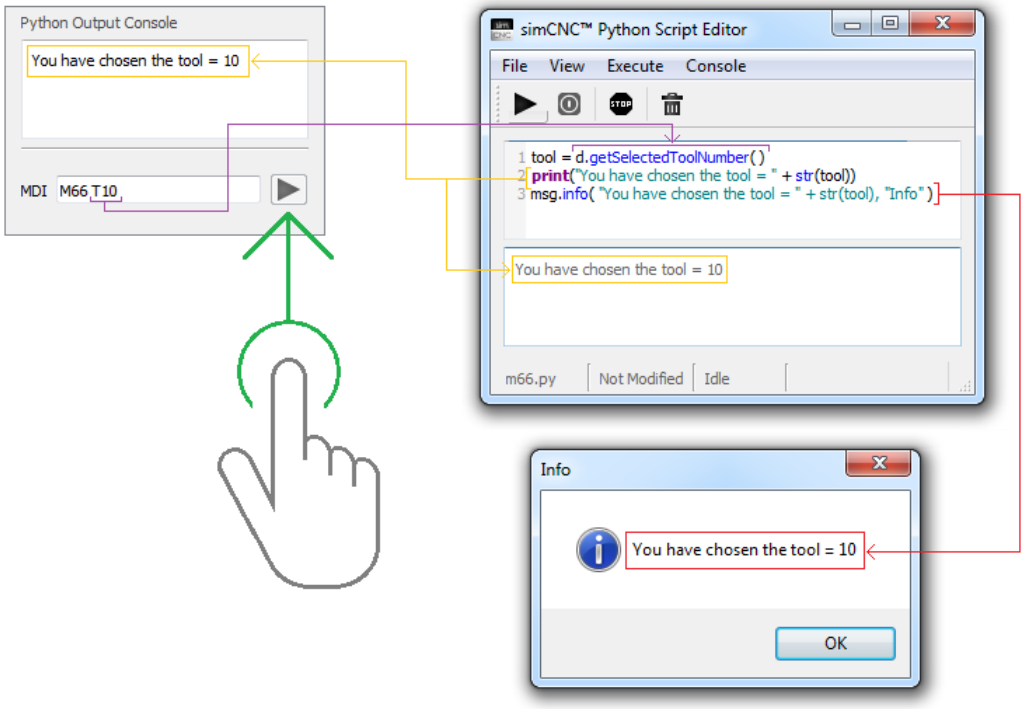
ERGEBNIS DER FUNKTION:

int - Werkzeugnummer

BEISPIEL:

```
tool = d.getSelectedToolNumber( )
print("You have chosen the tool = " + str(tool))
msg.info( "You have chosen the tool = " + str(tool), "Info" )
```

Speichern Sie das obige Script z.B. unter M66, dann rufen Sie es mithilfe des gcod oder MDI-Zeile ab, indem Sie den Befehl „T" + beliebige Werkzeugnummer (z.B. M66 T10) hinzufügen. Nach diesem Vorgang erscheint das Fenster mit der gewählten Werkzeugnummer. Beispielsweise wurde von uns die MDI-Zeile und die Werkzeugnummer 10 verwendet. Wir haben das Makro M66 absichtlich empfohlen, um zu zeigen, dass die Funktion d.getSelectedToolNumber( ) mit einem beliebigen Makro, nicht nur M6 funktioniert.



void setSelectedToolNumber(int toolNumber) - Stellt die Nummer des gewählten Werkzeugs ein – Pendant des Befehls „Txx", der mithilfe des gcod bzw. MDI-Zeile abgerufen wird.

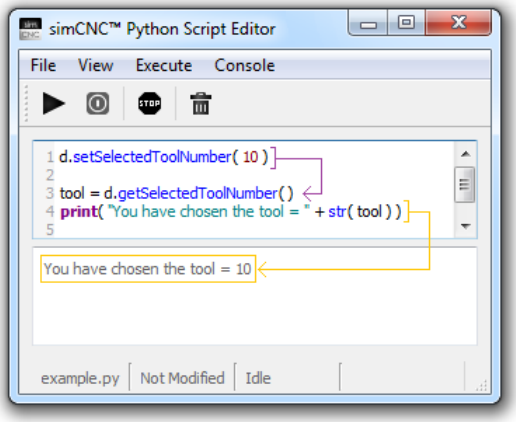
ARGUMENTE:

int toolNumber - Werkzeugnummer

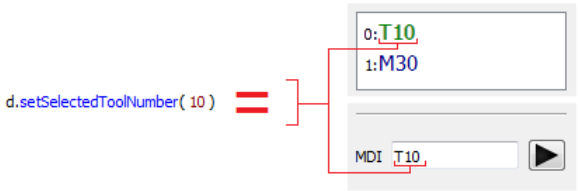
BEISPIEL:

```
d.setSelectedToolNumber( 10 )                    # Stellt die Nummer des gewählten Werkzeugs ein

tool = d.getSelectedToolNumber( )                # Gibt die Nummer des gewählten Werkzeugs zurück
print( "You have chosen the tool = " + str( tool ) )
```

Wenn die Funktion „d.setSelectedToolNumber( )" nach der Aktivierung des obigen Beispiels die Nummer des gewählten Werkzeugs auf 10 einstellt, gibt die uns bereits bekannte Funktion „d.getSelectedToolNumber( )" dieselbe Werkzeugnummer zurück.



Es ist einfach zu sehen, dass die Funktion d.setSelectedToolNumber(10) genau dieselbe Aufgabe erfüllt wie der Befehl T10, der mithilfe des gcod bzw. MDI-Zeile abgerufen wird.



void setSpindleToolNumber(int toolNumber) - Stellt die Werkzeugnummer in der Spindel ein.

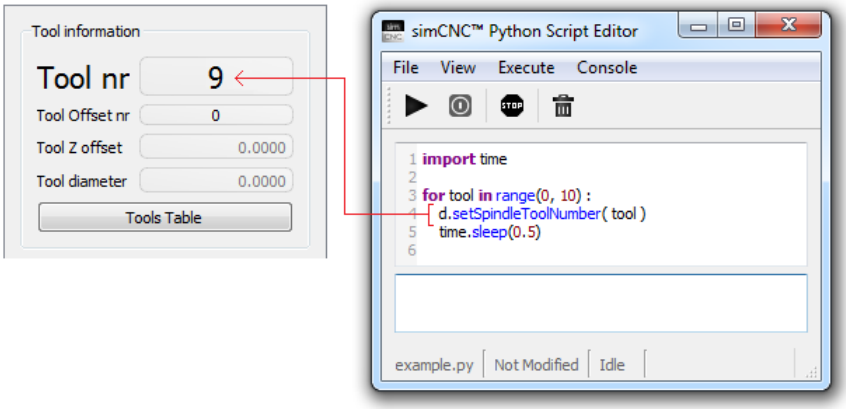
ARGUMENTE:

int toolNumber - Werkzeugnummer

BEISPIEL:

```
import time

for tool in range(0, 10) :
  d.setSpindleToolNumber( tool )
  time.sleep(0.5)
```

Wenn Sie das obige Beispiel aktivieren, erhöht die Funktion „d.setSpindleToolNumber(tool)" in der Schleife „for" alle 0.5 Sekunden die Werkzeugnummer auf dem Bildschirm simCNC, bis sie den Wert 9 erreicht.



int getSpindleToolNumber( ) - Gibt die Werkzeugnummer in der Spindel zurück.

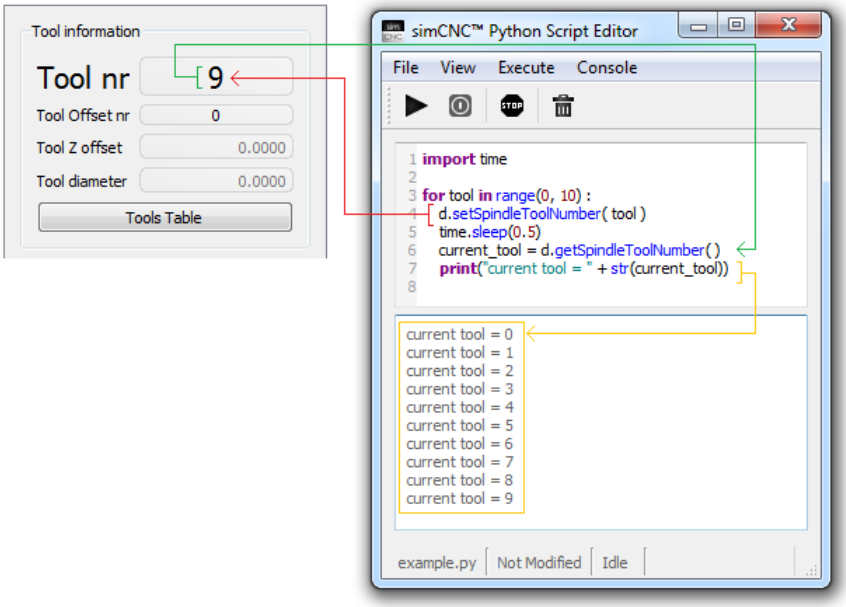
ERGEBNIS DER FUNKTION:

int - Werkzeugnummer

BEISPIEL:

```
import time

for tool in range(0, 10) :
  d.setSpindleToolNumber( tool )
  time.sleep(0.5)
  current_tool = d.getSpindleToolNumber( )
  print("current tool = " + str(current_tool))
```

Das obige Script ist Fortsetzung des vorigen Beispiels. Diesmal wurde der Schleife „for" die Funktion „current_tool = d.getSpindleToolNumber( )" hinzugefügt, die mit jedem Umlauf der Schleife „for" die aktuelle Werkzeugnummer in der Spindel zurückgibt, dann wird die Werkzeugnummer mithilfe der Funktion „print" in der Python Konsole dargestellt.



2) Werkzeuglänge

void setToolLength(int toolNumber, float toolLength) - Stellt den Offset-Wert der Werkzeuglänge ein.

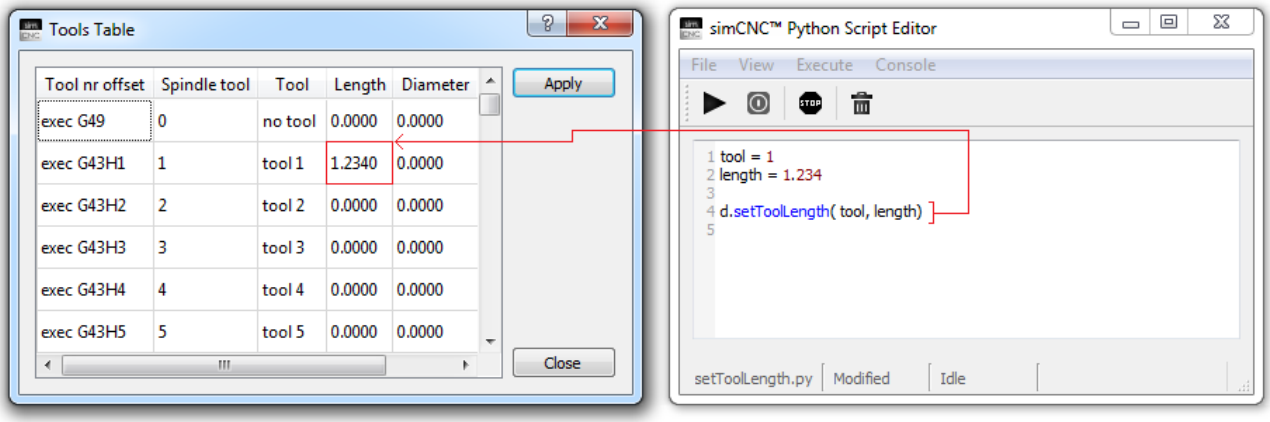
ARGUMENTE:

int toolNumber - Werkzeugnummer
float toolLength - Offset-Wert der Werkzeuglänge

BEISPIEL:

```
tool = 1
length = 1.234

d.setToolLength( tool, length )
```

Wenn Sie das obige Beispiel aktivieren, wird Offset der Werkzeuglänge Nummer 1 auf 1.234 eingestellt.



float getToolLength(int toolNumber) - Gibt den Offset-Wert der Werkzeuglänge zurück.

ARGUMENTE:

int toolNumber - Werkzeugnummer

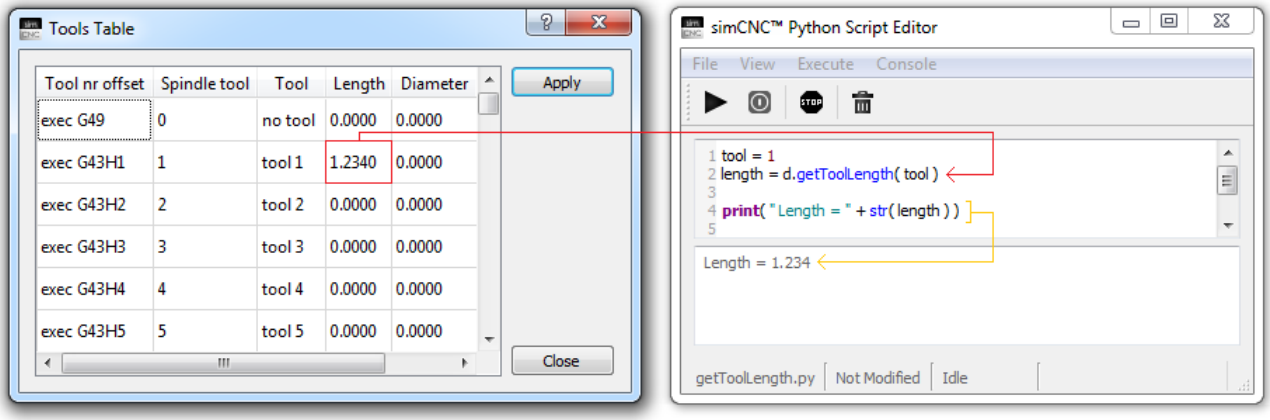
ERGEBNIS DER FUNKTION:

float - Offset der Werkzeuglänge

BEISPIEL:

```
tool = 1
length = d.getToolLength( tool )

print( " Length = " + str( length ) )
```

Wenn Sie das obige Beispiel aktivieren, wird in der Python Konsole der Offset der Werkzeuglange Nummer 1 angezeigt.



void setToolOffsetNumber(int toolNumber) - Stellt die Nummer Offset der Werkzeuglange ein.

ARGUMENTE:

int toolNumber - Werkzeugnummer

BEISPIEL:

```
tool = 1

d.setToolOffsetNumber( tool )
```

Wenn Sie das obige Beispiel aktivieren, wird Offset der Werkzeuglange auf 1 eingestellt.

int getToolOffsetNumber( ) - Gibt die aktuell verwendete Nummer des Werkzeuglangen-Offset zurück.

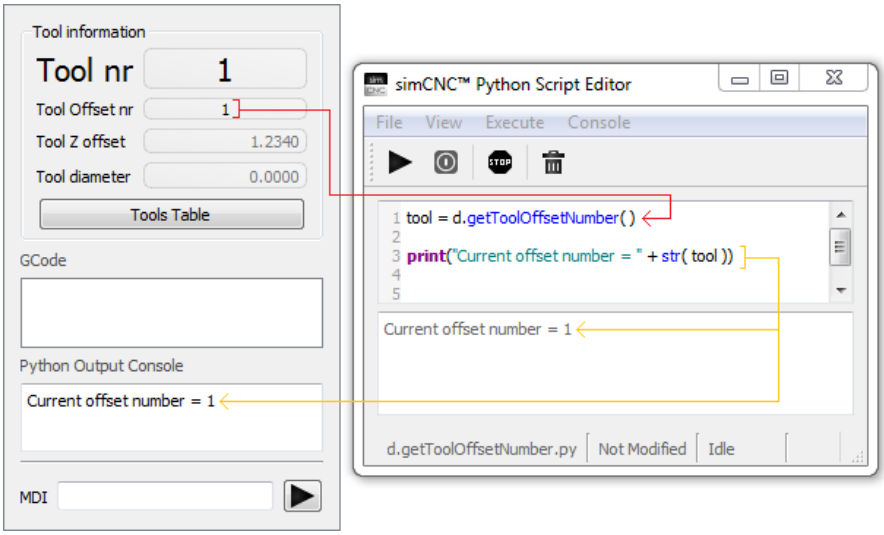
ARGUMENTE:

int - Werkzeugnummer

BEISPIEL:

```
tool = d.getToolOffsetNumber( )

print("Current offset number = " + str( tool ))
```

Wenn Sie das obige Beispiel aktivieren, wird in der Python Konsole die Nummer des aktuell verwendeten Werkzeuglangen-Offsets angezeigt.



3) Werkzeugdurchmesser

ACHTUNG!

SimCNC verwendet zurzeit keine Werkzeugdurchmesserwerte, d.h. simCNC unterstuetzt zurzeit nicht die Funktion der Kompensation des Werkzeugdurchmessers.

void setToolDiameter(int toolNumber, float toolDiameter) - Stellt den Wert des Werkzeugdurchmessers ein.

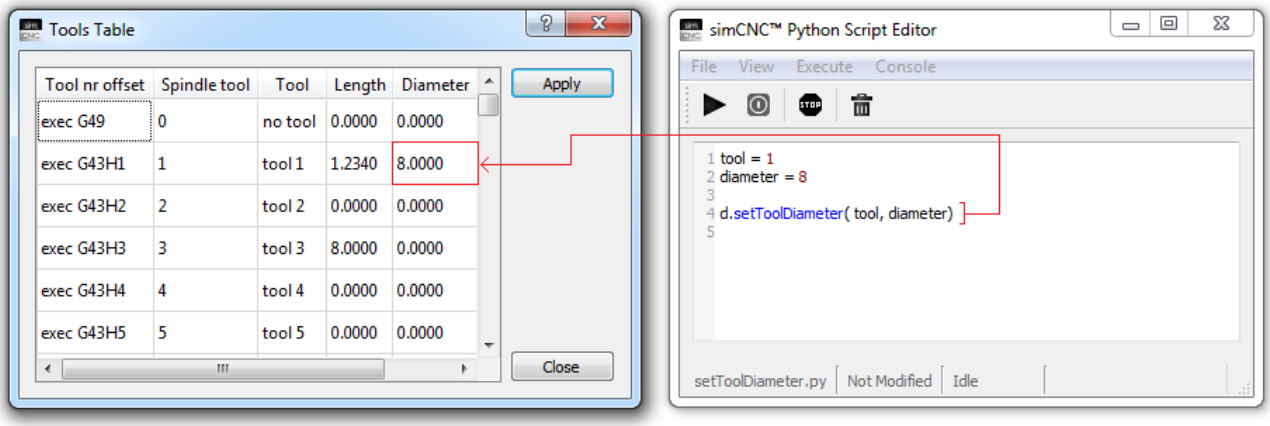
ARGUMENTE:

int toolNumber - Werkzeugnummer
float toolDiameter - Werkzeugdurchmesser

BEISPIEL:

```
tool = 1
diameter = 8

d.setToolDiameter( tool, diameter )
```

Wenn Sie das obige Beispiel aktivieren, wird der Werkzeugdurchmesser Nummer 1 auf 8 eingestellt.



float getToolDiameter(int toolNumber) - Gibt den Wert des Werkzeugdurchmessers zurück.

ARGUMENTE:

int toolNumber - Werkzeugnummer

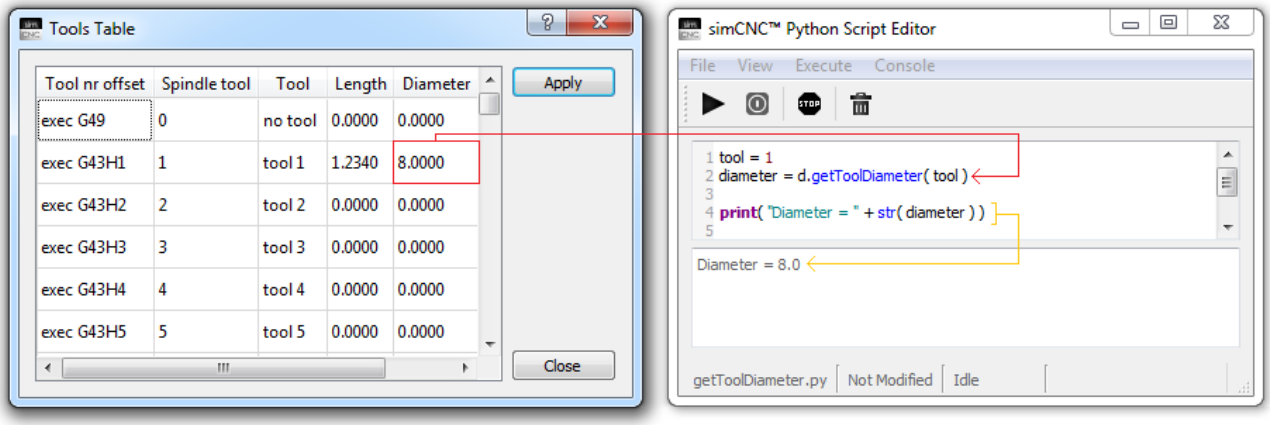
ERGEBNIS DER FUNKTION:

float - Werkzeugmurchmesser

BEISPIEL:

```
tool = 1
diameter = d.getToolDiameter( tool )

print( "Diameter = " + str( diameter ) )
```

Wenn Sie das obige Beispiel aktivieren, wird in der Python Konsole der Wert des Werkzeugdurchmessers Nummer 1 angezeigt.



IX. Spindel, Kältemittel und Nebel

1) Spindel

void setSpindleSpeed(float spindleSpeed) - Stellt die vorgegebene Drehgeschwindigkeit der Spindel (RPM) ein.

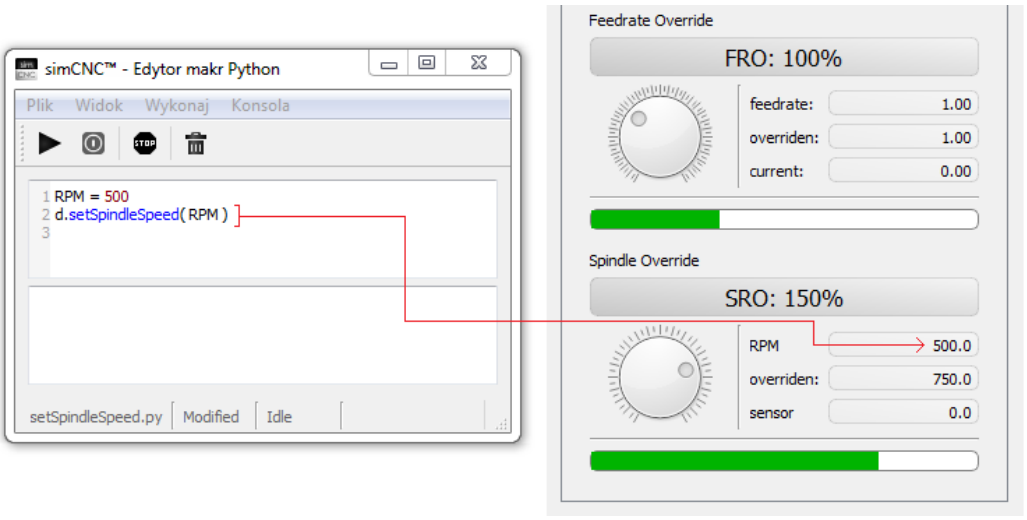
ARGUMENTE:

float spindleSpeed - die vorgegebene Drehgeschwindigkeit der Spindel (RPM)

BEISPIEL:

```
RPM = 500
d.setSpindleSpeed( RPM )
```

Wenn Sie das obige Beispiel aktivieren, wird die vorgegebene Drehgeschwindigkeit der Spindel auf 500 RPM eingestellt.



float getSpindleSpeed( ) - Gibt die vorgegebene Drehgeschwindigkeit der Spindel (RPM) zurück.

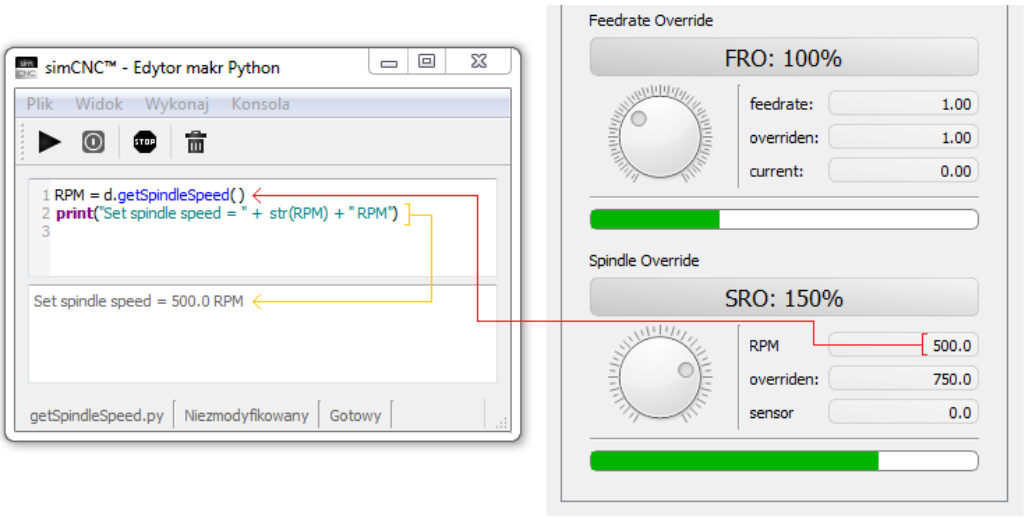
ERGEBNIS DER FUNKTION:

float – die vorgegebene Drehgeschwindigkeit der Spindel (RPM)

BEISPIEL:

```
RPM = d.getSpindleSpeed( )
print("Set spindle speed = " + str(RPM))
```

Wenn Sie das obige Beispiel aktivieren, wird in der Python Konsole die vorgegebene Drehgeschwindigkeit der Spindel angezeigt.



void setSpindleState(SpindleState spindleState) - Stellt den Spindel-Zustand ein

ARGUMENTE:

SpindleState spindleState - Spindel-Zustand

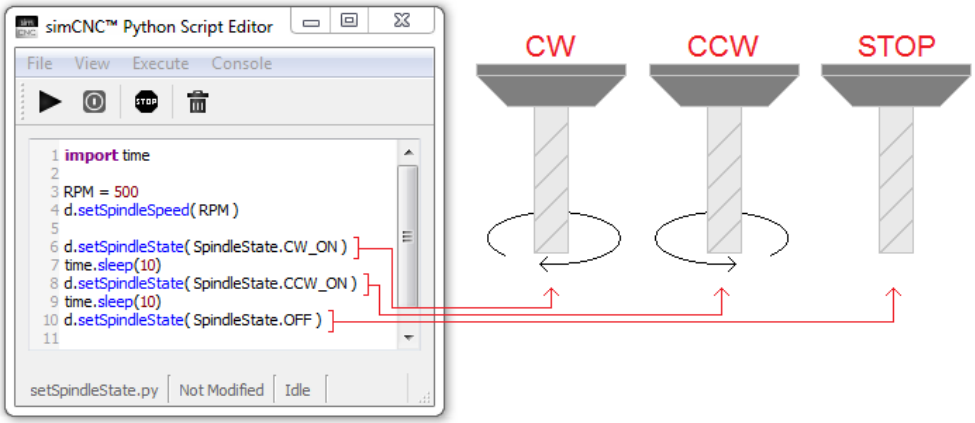
WIR UNTERSCHEIDEN ZWISCHEN:

SpindleState.CW_ON	- Umdrehungen der Spindel im Uhrzeigersinn
SpindleState.CCW_ON	- Umdrehungen der Spindel gegen den Uhrzeigersinn
SpindleState.OFF	- Umdrehungen der Spindel ausgeschaltet

BEISPIEL:

```
import time

RPM = 500
d.setSpindleSpeed( RPM )

d.setSpindleState( SpindleState.CW_ON )
time.sleep(10)
d.setSpindleState( SpindleState.CCW_ON )
time.sleep(10)
d.setSpindleState( SpindleState.OFF )
```

Wenn Sie das obige Beispiel aktivieren, wird die Spindel für 10 Sekunden eingeschaltet und dreht sich im Uhrzeigersinn, dann wechselt sie die Richtung und dreht sich gegen den Uhrzeigersinn, nach 10 Sekunden wird sie ausgeschaltet.
Beachten Sie, dass die Funktion setSpindleState die Ausführung des Scripts für die Dauer der Beschleunigungs- und Bremsrampe der Spindel stoppt.



SpindleState getSpindleState( ) - Gibt den aktuellen Spindel-Zustand zurück.

ERGEBNIS DER FUNKTION:

SpindleState - Spindel-Zustand

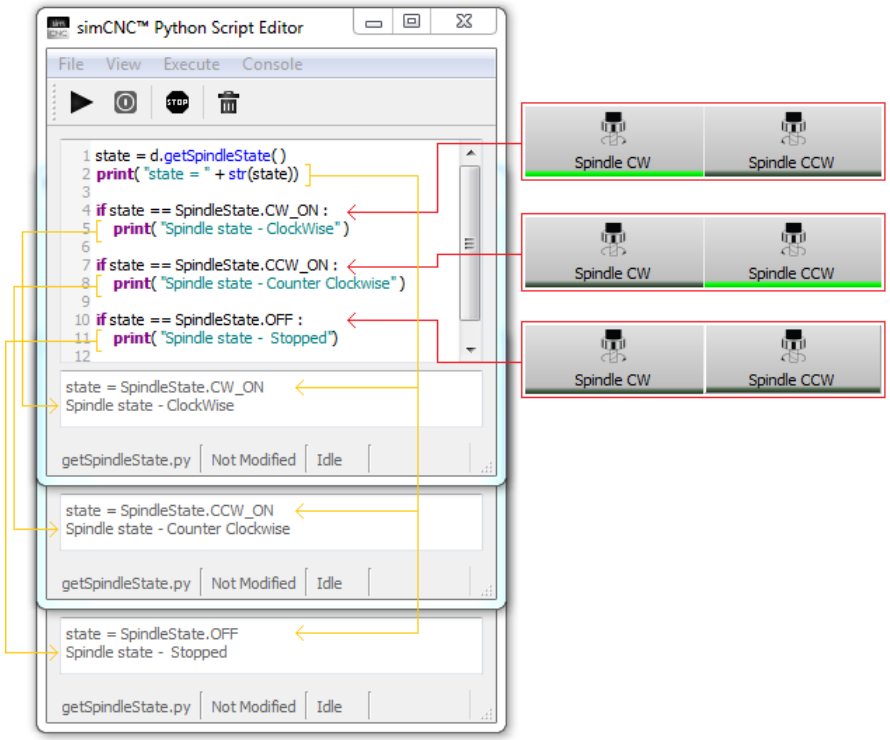
WIR UNTERSCHEIDEN ZWISCHEN:

SpindleState.CW_ON	- Umdrehungen der Spindel im Uhrzeigersinn
SpindleState.CCW_ON	- Umdrehungen der Spindel gegen den Uhrzeigersinn
SpindleState.OFF	- Umdrehungen der Spindel ausgeschaltet

BEISPIEL:

```
state = d.getSpindleState( )
print( "state = " + str(state))

if state == SpindleState.CW_ON :
  print( "Spindle state - ClockWise" )

if state == SpindleState.CCW_ON :
  print( "Spindle state - Counter Clockwise" )

if state == SpindleState.OFF :
  print( "Spindle state - Stopped")
```

Wenn Sie das obige Beispiel aktivieren, wird in der ersten Zeile der Python Konsole der aktuelle Spindel-Zustand angezeigt, der Ergebnis der Funktion d.getSpindleState( ) ist. In der zweiten Zeile wird die eigene Mitteilung angezeigt, die klarer den aktuellen Spindel-Zustand beschreibt.
Im nachstehenden Bild wurde die Funktion des Scripts in allen Spindel-Zuständen dargestellt.



void waitForSpindleSetSpeed(int timeout_sec) - Wartet auf eine Überschreitung der definierten Drehgeschwindigkeit der Spindel über eine bestimmte Zeit. Die erwartete Drehgeschwindigkeit der Spindel wird durch die vorgegebene Drehgeschwindigkeit und den Parameter „ Spindle Ready Level" („Bereitschaftslevel") bestimmt.

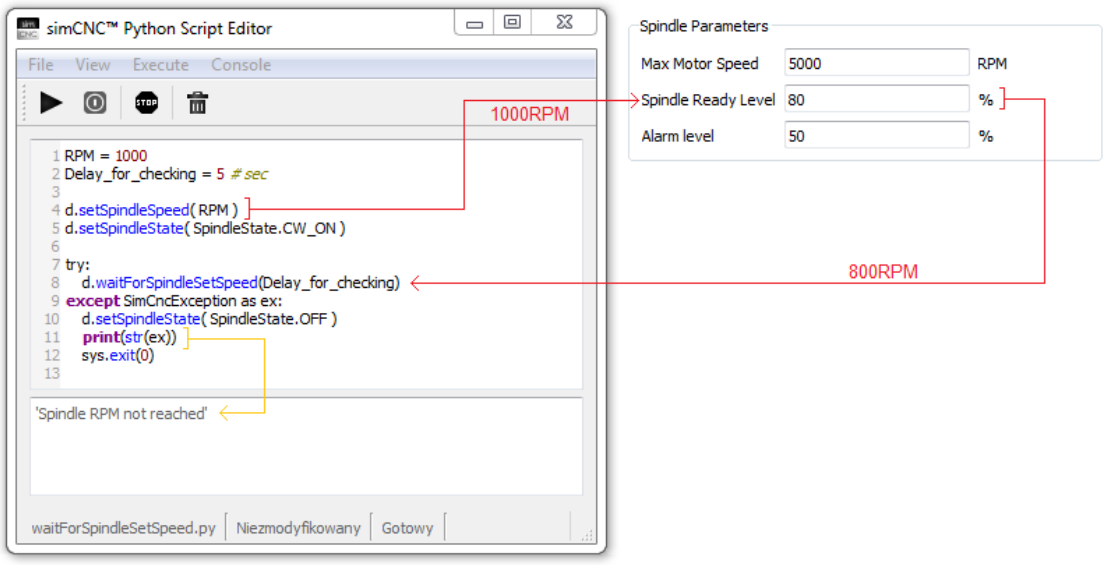
ARGUMENTE:

int timeout_sec - Wartezeit (in Sekunden)

BEISPIEL:

```
RPM = 500
Delay_for_checking = 5 # sec

d.setSpindleSpeed( RPM )
d.setSpindleState( SpindleState.CW_ON )

try:
  d.waitForSpindleSetSpeed(Delay_for_checking)
except SimCncException as ex:
  print(str(ex))
  sys.exit(0)
```

Wenn Sie das obige Beispiel aktivieren, wird die Spindel eingeschaltet und dreht sich im Uhrzeigersinn (Zeile 5 – siehe das Bild unten) mit der vorgegebener Drehgeschwindigkeit von 1.000 RPM (Zeile 4). Dann stoppt die Funktion „waitForSpindleSetSpeed" die Ausführung des Scripts, bis die Spindel 80% der vorgegebenen Drehgeschwindigkeit von 800 RPM (80% von 1000 RPM = 800 RPM) erreicht. Falls die Spindel die 800 RPM innerhalb von 5 Sekunden nicht erreicht, meldet die Funktion „waitForSpindleSetSpeed" die Ausnahme „SimCncException" (Zeile 9). Dies führt zum Stopp der Spindel (Zeile 10), zur Anzeige der Ausnahmetextes in der Python Konsole (Zeile 11) und Beendigung des Scripts auf Befehl (Zeile 12), um die weitere Ausführung des Scripts zu verhindern. Für das obige Beispiel ist das Modul CSMIO-ENC notwendig, weil es Informationen über die aktuelle Drehgeschwindigkeit der Spindel liefert.



Das vorige Beispiel kann nach kleiner Änderung als Makro M3 verwenden, das die Ausführung des gcod für die Zeit der Spindelbeschleunigung auf die erwartete Drehgeschwindigkeit oder die Arbeit der Maschine bei Problemen mit der Erreichung der vorgegebenen Drehgeschwindigkeit stoppt.

```
Delay_for_checking = 5 # sec

d.setSpindleState( SpindleState.CW_ON )

try:
  d.waitForSpindleSetSpeed( Delay_for_checking )
except SimCncException as ex:
  d.setSpindleState( SpindleState.OFF )
  print(str(ex))
  d.stopTrajectory( )
  sys.exit(0)
```

2) Kältemittel

void setFloodState(FloodState state) - Stellt den Zustand des flüssigen Kältemittels ein.

ARGUMENTE:

FloodState state - Zustand des flüssigen Kältemittels

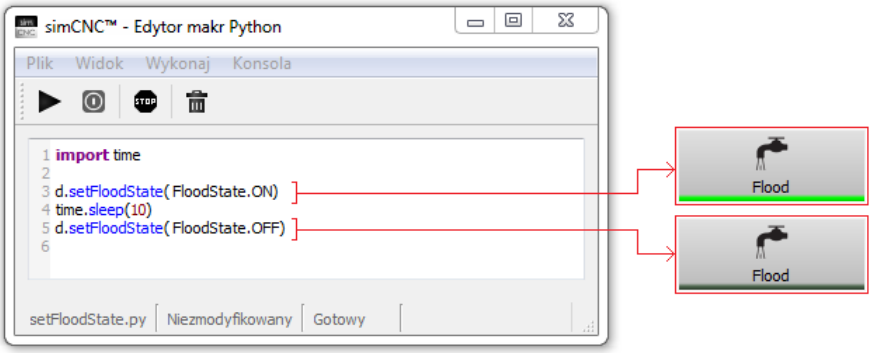
WIR UNTERSCHEIDEN ZWISCHEN:

FloodState.ON	- Kältemittel einschalten
FloodState.OFF	- Kältemittel ausschalten

BEISPIEL:

```
import time

d.setFloodState( FloodState.ON )
time.sleep(10)
d.setFloodState( FloodState.OFF )
```

Wenn Sie das obige Beispiel aktivieren, wird das Kältemittel für 10 Sekunden eingeschaltet.



FloodState getFloodState( ) - Gibt den Zustand des flüssigen Kältemittels zurück.

ERGEBNIS DER FUNKTION:

FloodState - Zustand des flüssigen Kältemittels

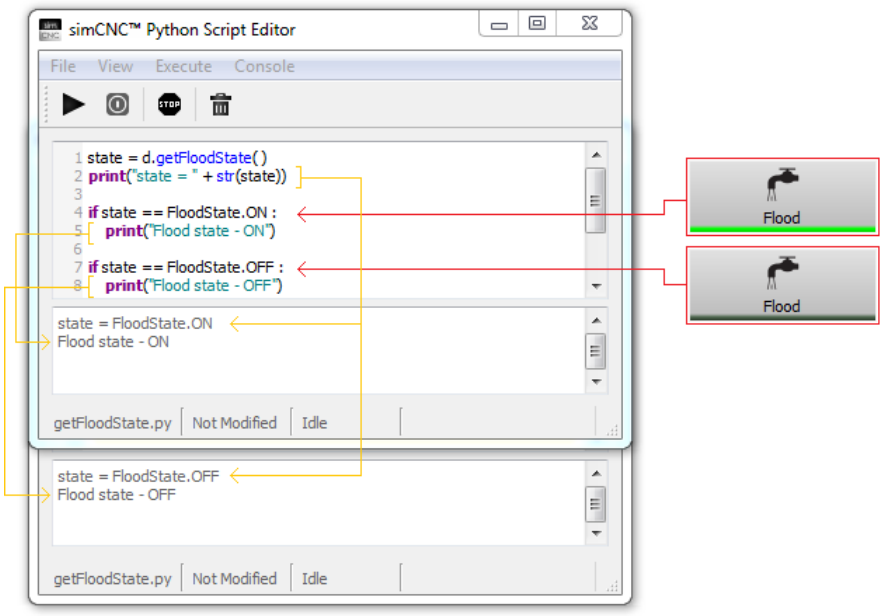
WIR UNTERSCHEIDEN ZWISCHEN:

FloodState.ON - Kältemittel eingeschaltet
FloodState.OFF - Kältemittel ausgeschaltet

BEISPIEL:

```
state = d.getFloodState( )
print("state = " + str(state))

if state == FloodState.ON :
    print("Flood state - ON")

if state == FloodState.OFF :
    print("Flood state - OFF")
```

Wenn Sie das obige Beispiel aktivieren, wird in der ersten Zeile der Python Konsole der aktuelle Kältemittel-Zustand angezeigt, der Ergebnis der Funktion d.getFloodState( ) ist. In der zweiten Zeile wird die eigene Mitteilung angezeigt, die klarer den aktuellen Kältemittel-Zustand beschreibt.
Im nachstehenden Bild wurde die Funktion des Scripts in beiden Kältemittel-Zuständen dargestellt.



3) Nebel

void setMistState(MistState state) - Stellt den Kältenebel-Zustand ein.

ARGUMENTE:

MistState state - Kältenebel-Zustand

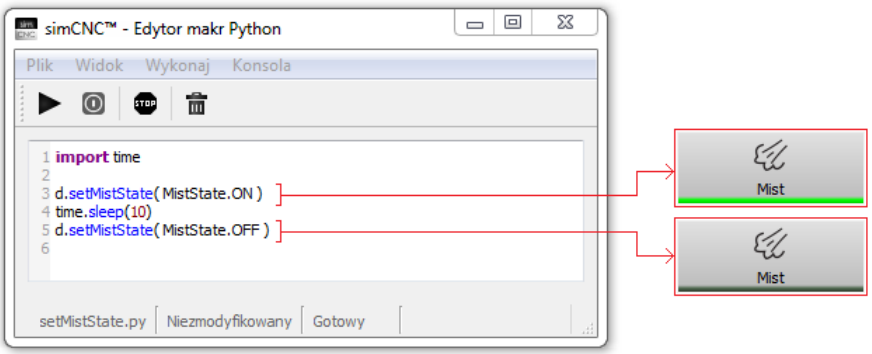
WIR UNTERSCHEIDEN ZWISCHEN:

MistState.ON - Nebel einschalten
MistState.OFF - Nebel ausschalten

BEISPIEL:

```
import time

d. ( MistState.ON )
time.sleep(10)
d.setMistState( MistState.OFF )
```

Wenn Sie das obige Beispiel aktivieren, wird das Kältemittel für 10 Sekunden eingeschaltet.



MistState getMistState( ) - Gibt den Kältenebel-Zustand zurück.

ERGEBNIS DER FUNKTION:

MistState - Kältenebel-Zustand

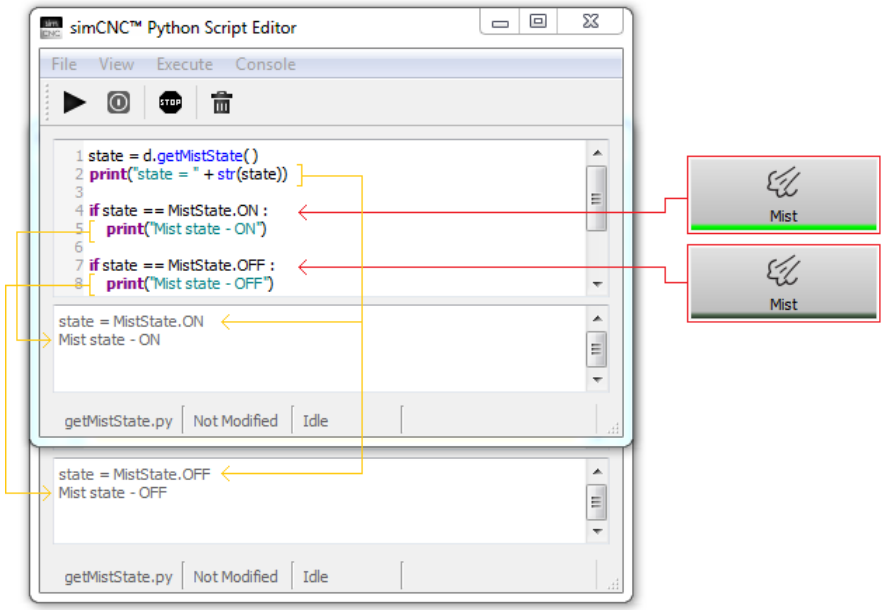
WIR UNTERSCHEIDEN ZWISCHEN:

MistState.ON - Kältenebel eingeschaltet
MistState.OFF - Kältenebel ausgeschaltet

BEISPIEL:

```
state = d.getMistState( )
print("state = " + str(state))

if state == MistState.ON :
    print("Mist state - ON")

if state == MistState.OFF :
    print("Mist state - OFF")
```

Wenn Sie das obige Beispiel aktivieren, wird in der ersten Zeile der Python Konsole der aktuelle Nebel-Zustand dargestellt, der Ergebnis der Funktion getMistState( ) ist. In der zweiten Zeile wird die eigene Mitteilung angezeigt, die klarer den aktuellen Nebel-Zustand beschreibt.
Im nachstehenden Bild wurde die Funktion des Scripts in beiden Nebel-Zuständen dargestellt.



X. Arbeitsoffset - Materialbasen.

List<float>[6] getCurrentWorkOffset( ) - Gibt die Koordinaten des aktuellen Arbeitsoffsets zurück.

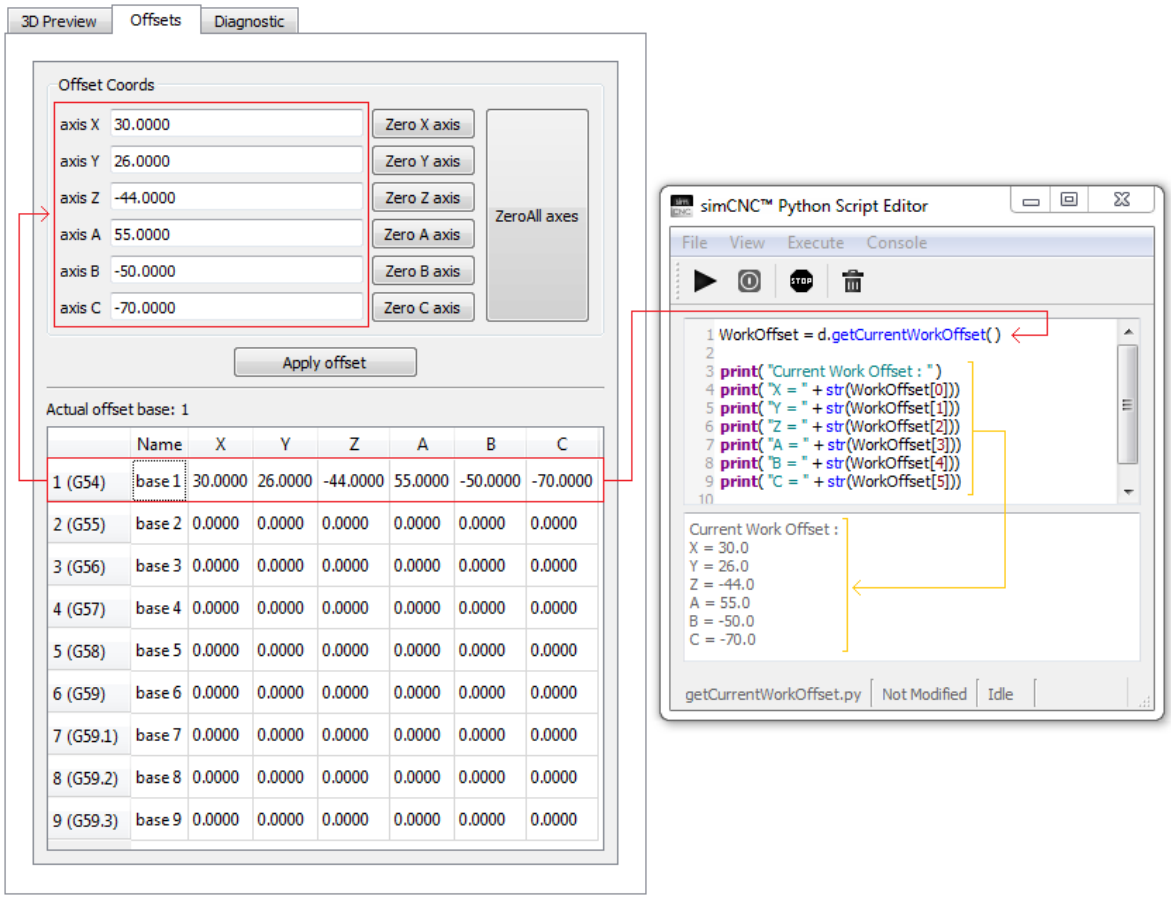
ERGEBNIS DER FUNKTION:

List<float>[6] - Liste von 6 Koordinaten (X, Y, Z, A, B, C), die Arbeitsoffset bestimmen.

BEISPIEL:

```
CurrentWorkOffset = d.getCurrentWorkOffset( )

print( "Current Work Offset : " )
print( "X = " + str(CurrentWorkOffset[0]))
print( "Y = " + str(CurrentWorkOffset[1]))
print( "Z = " + str(CurrentWorkOffset[2]))
print( "A = " + str(CurrentWorkOffset[3]))
print( "B = " + str(CurrentWorkOffset[4]))
print( "C = " + str(CurrentWorkOffset[5]))
```

Wenn Sie das obige Beispiel aktivieren, werden in der Python Konsole die Koordinaten des aktuellen Arbeitsoffsets abgelesen und angezeigt.



List<float>[6] getWorkOffset(int workOffsetIndex) - Gibt die Koordinaten des Arbeitsoffsets mit dem vorgegebenen Index zurück.

ARGUMENTE:

int workOffsetIndex - Index (Nummer) des Arbeitsoffsets

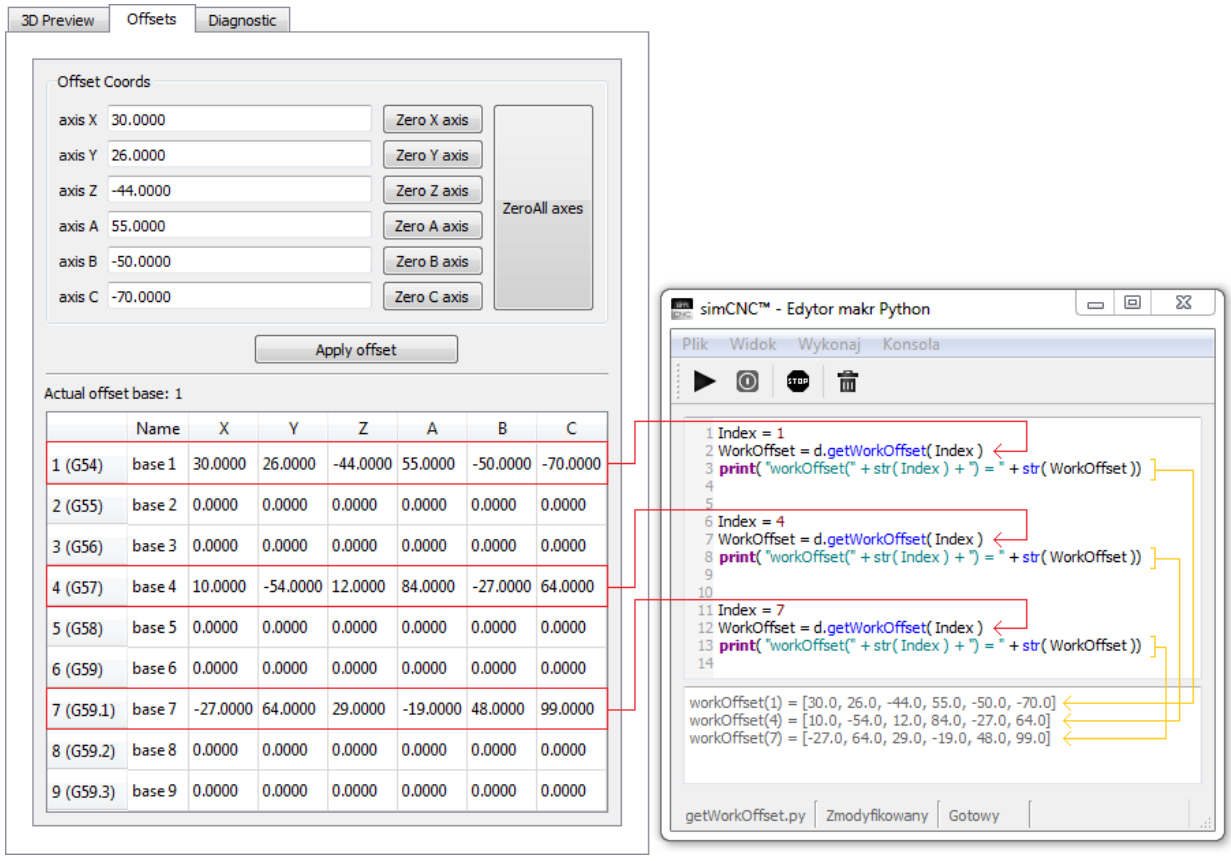
ERGEBNIS DER FUNKTION:

List<float>[6] - Liste von 6 Koordinaten (X, Y, Z, A, B, C) des Arbeitsoffsets

BEISPIEL:

```
Index = 1
WorkOffset = d.getWorkOffset( Index )
print( "workOffset(" + str( Index ) + ") = " + str( WorkOffset ))

Index = 4
WorkOffset = d.getWorkOffset( Index )
print( "workOffset(" + str( Index ) + ") = " + str( WorkOffset ))

Index = 7
WorkOffset = d.getWorkOffset( Index )
print( "workOffset(" + str( Index ) + ") = " + str( WorkOffset ))
```

Wenn Sie das obige Beispiel aktivieren, werden in der Python Konsole die Koordinaten des Arbeitsoffsets Nummer 1 (G54), 4 (G57) und 7 (G59.1) abgelesen und angezeigt.



ACHTUNG!

simCNC kann zurzeit 9 Arbeitsoffsets (9 Materialbasen) speichern.
Die Arbeitsoffsets können damit Index von 1 (G54) bis 9 (G59.3) akzeptieren.
Im Klartext : 1 = G54 | 2 = G55 | 3 = G56 | 4 = G57 | 5 = G58 | 6 = G59 | 7 = G59.1 | 8 = G59.2 | 9 = G59.3

int getWorkOffsetNumber( ) - Gibt den Index des aktuellen Arbeitsoffsets zurück.

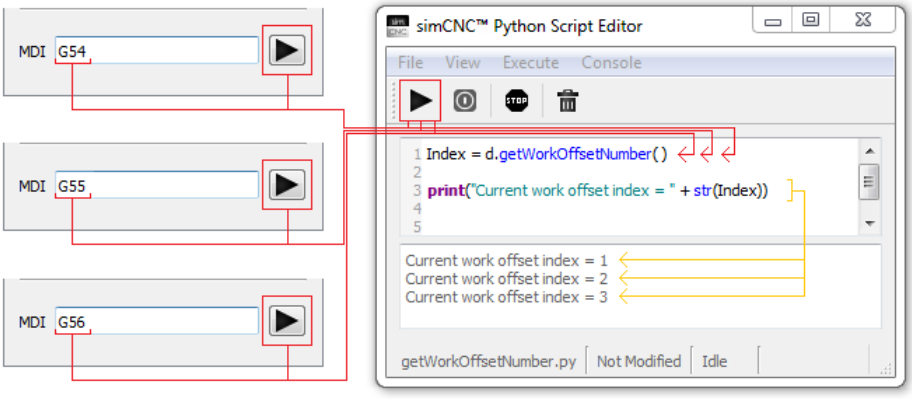
ERGEBNIS DER FUNKTION:

int - Index (Nummer) des Arbeitsoffsets

BEISPIEL:

```
Index = d.getWorkOffsetNumber( )

print("Current work offset index = " + str(Index))
```

Bevor Sie das obige Beispiel aktivieren, geben Sie den Befehl „G54" in die MDI-Zeile ein und führen Sie ihn aus, indem Sie die Schaltfläche an der rechten Seite der MDI-Zeile drücken. Starten Sie dann das Beispiel-Script. In der Python Konsole erscheint eine Information zum Index des aktuell verwendeten Arbeitsoffsets. Wiederholen Sie das mit den Befehlen „G55" und „G56", bis Sie das gleiche Ergebnis wie im Bild unten dargestellt erzielt haben.



setWorkOffset(int workOffsetIndex, List<float>[6] workOffsetValues) - Ändert die Koordinaten des Arbeitsoffsets mit dem vorgegebenem Index.

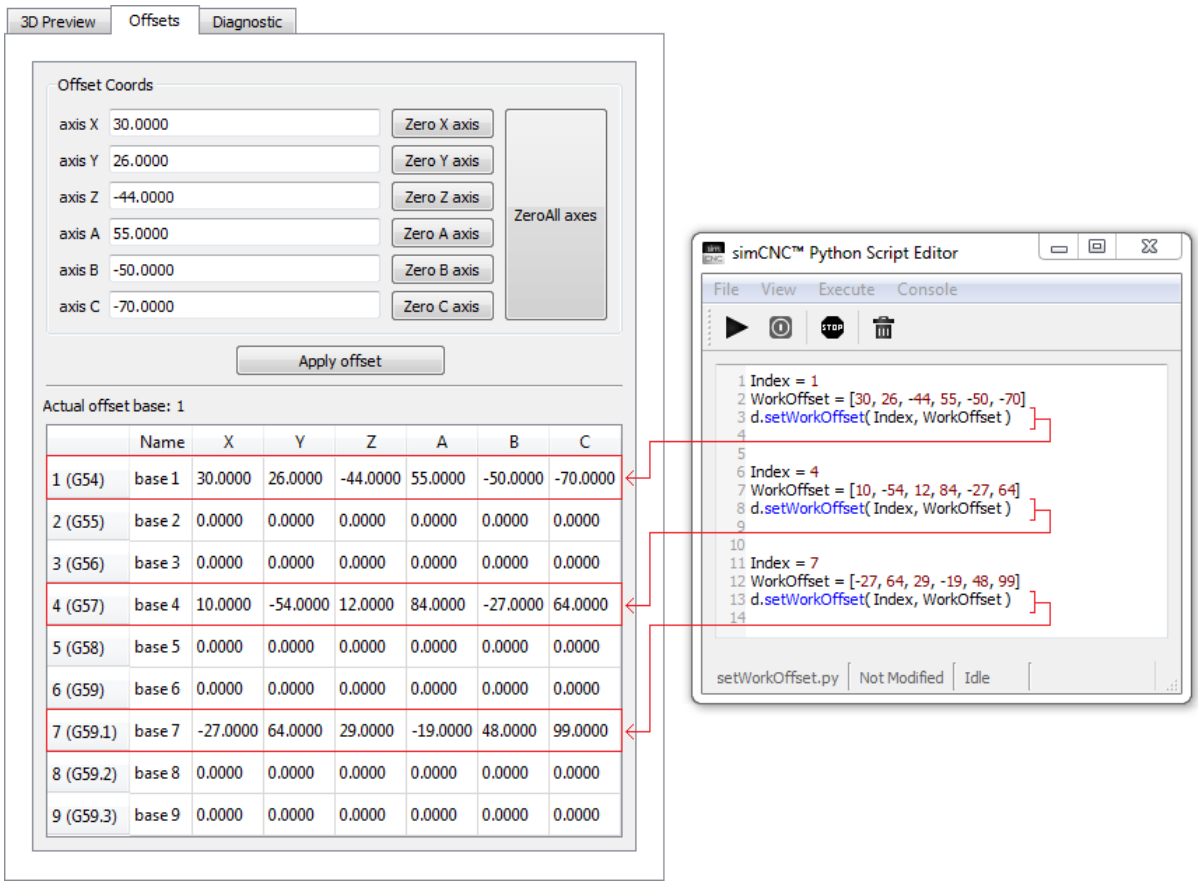
ARGUMENTE:

int workOffsetIndex - Index (Nummer) des Arbeitsoffsets
List<float>[6] workOffsetValues - neue Werte von 6 Koordinaten (X, Y, Z, A, B, C) des Arbeitsoffsets

BEISPIEL:

```
Index = 1
WorkOffset = [30, 26, -44, 55, -50, -70]
d.setWorkOffset( Index, WorkOffset )

Index = 4
WorkOffset = [10, -54, 12, 84, -27, 64]
d.setWorkOffset( Index, WorkOffset )

Index = 7
WorkOffset = [-27, 64, 29, -19, 48, 99]
d.setWorkOffset( Index, WorkOffset )
```

Wenn Sie das obige Beispiel aktivieren, werden die Arbeitsoffsets mit Index 1 (G54), 4 (57) und 7 (G59.1) auf neue Koordinaten [30, 26, -44, 55, -50, -70], [10, -54, 12, 84, -27, 64] und [-27, 64, 29, -19, 48, 99] eingestellt.



setCurrentWorkOffset(List<float>[6] workOffsetValues) - Ändert die Koordinaten des aktuellen Arbeitsoffsets.

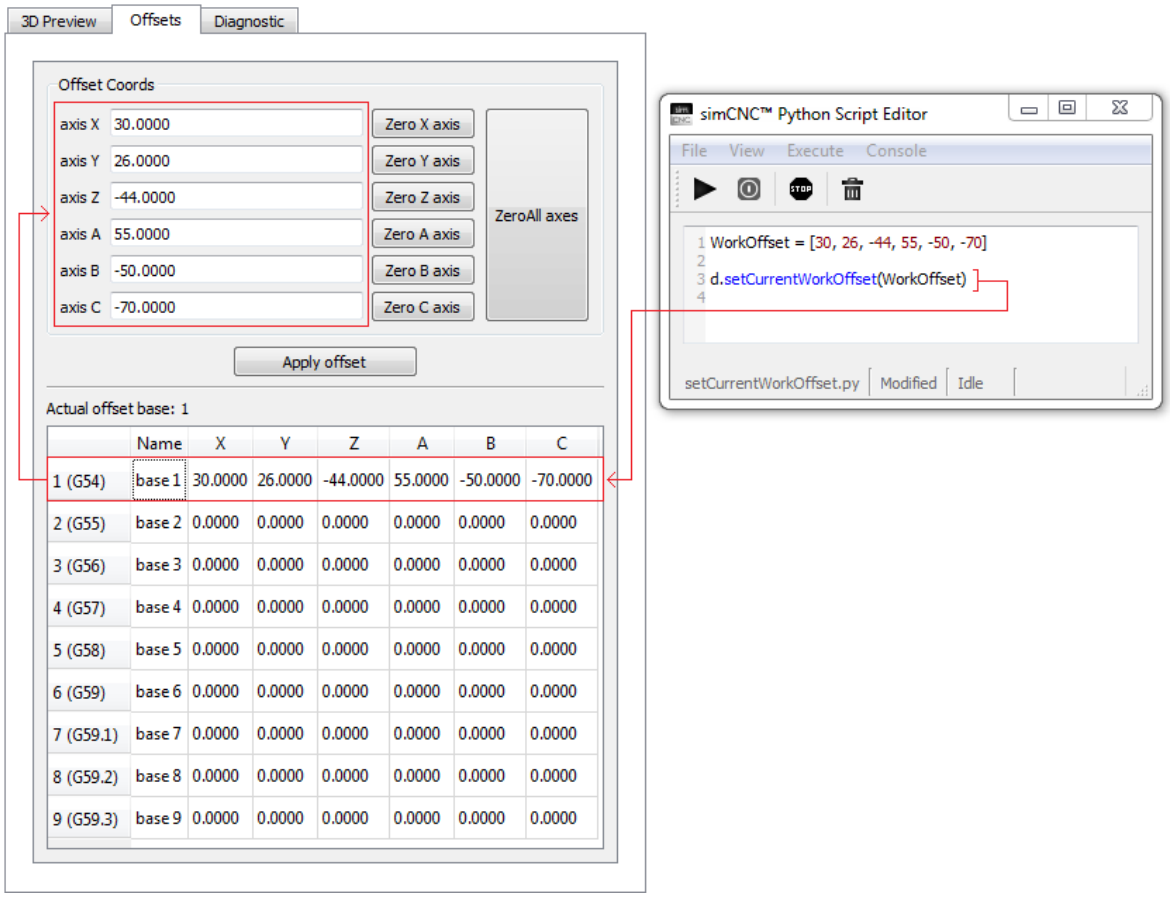
ARGUMENTE:

List<float>[6] workOffsetValues - neue Werte von 6 Koordinaten (X, Y, Z, A, B, C) des aktuellen Arbeitsoffsets

BEISPIEL:

```
WorkOffset = [30, 26, -44, 55, -50, -70]

d.setCurrentWorkOffset( WorkOffset )
```

Wenn Sie das obige Beispiel aktivieren, wird der Arbeitsoffset mit Index 1 (G54) auf den neuen Wert der Koordinaten [30, 26, -44, 55, -50, -70] eingestellt.



	Name	X	Y	Z	A	B	C
1 (G54)	base 1	30.0000	26.0000	-44.0000	55.0000	-50.0000	-70.0000
2 (G55)	base 2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3 (G56)	base 3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4 (G57)	base 4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5 (G58)	base 5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6 (G59)	base 6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7 (G59.1)	base 7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
8 (G59.2)	base 8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
9 (G59.3)	base 9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

setWorkOffsetNumber(int workOffsetIndex) - Ändert den Index des aktuellen Arbeitsoffsets.

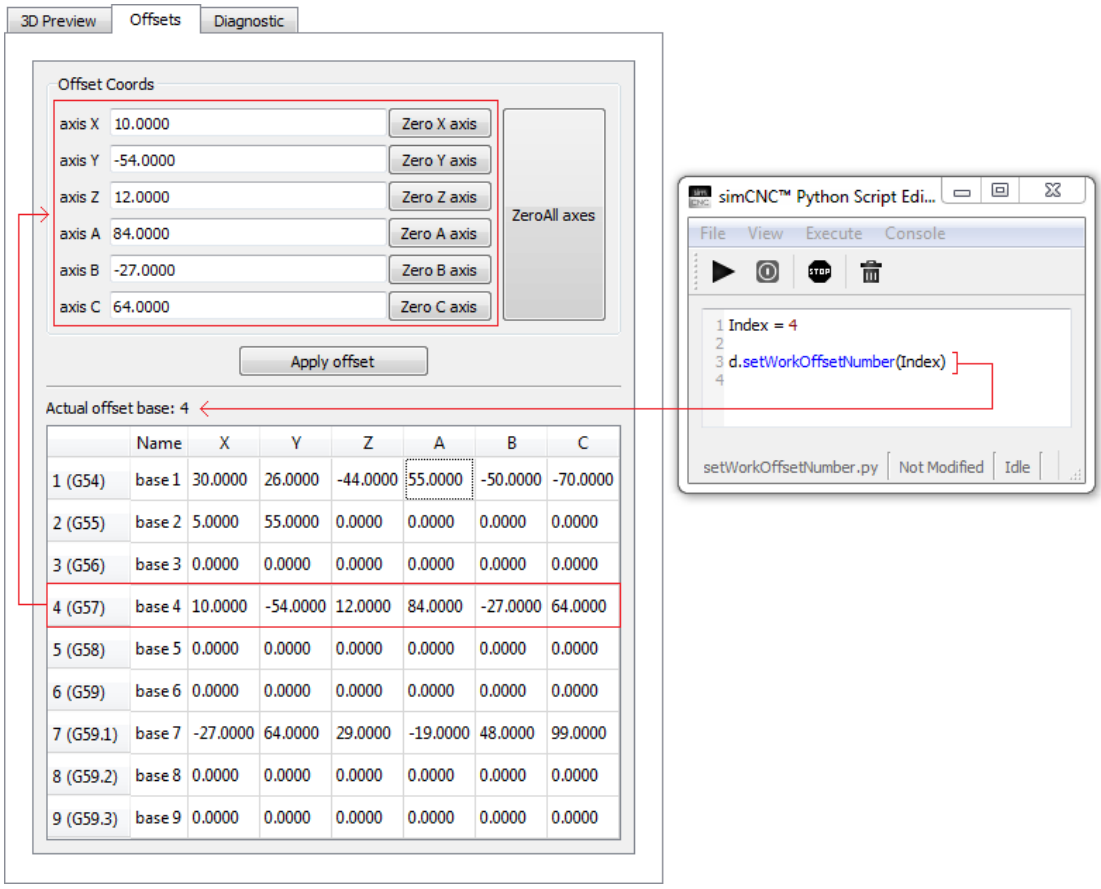
ARGUMENTE:

int workOffsetIndex - Index (Nummer) des Arbeitsoffsets

BEISPIEL:

```
Index = 4

d.setWorkOffsetNumber( Index )
```

Wenn Sie das obige Beispiel aktivieren, wird der aktuelle Arbeitsoffset auf Arbeitsoffset mit Index 4 (G57) eingestellt.



	Name	X	Y	Z	A	B	C
1 (G54)	base 1	30.0000	26.0000	-44.0000	55.0000	-50.0000	-70.0000
2 (G55)	base 2	5.0000	55.0000	0.0000	0.0000	0.0000	0.0000
3 (G56)	base 3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4 (G57)	base 4	10.0000	-54.0000	12.0000	84.0000	-27.0000	64.0000
5 (G58)	base 5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6 (G59)	base 6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7 (G59.1)	base 7	-27.0000	64.0000	29.0000	-19.0000	48.0000	99.0000
8 (G59.2)	base 8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
9 (G59.3)	base 9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000