CS LAB s.c.
ElectronicLaboratory

user * guide * manual *

simCNC
control software

Guide To Python Scripts

CONTENT

I. Python – General Information

simCNC control software supports scripts that automate procedures such as tool change or tool length measurement. The usability of the scripts does not end with such procedures, however, and it is sometimes necessary to write scripts that are much more complex. To help the software's users create their own scripts, simCNC uses Python as a script language.

Python is a general-purpose high-level programming language with an extensive set of standard libraries, which mainly features simple and clear source code. Its syntax is also clear and concise.

Python is used by both beginners and professionals. With the language being so popular, many tutorials and libraries are created that can be found on innumerable websites. With such things in place, the only obstacle on the way to write your own script is your willingness and imagination.

II. SimCNC And Python, What Should You Know?

When simCNC is installed on the hard disk, Python is installed as separate software, too. simCNC communicates with Python via the UDP Internet protocol using a special library called ___COMM.pyc. In addition to this library, the developers have also prepared a library called ___DEVICE.pyc, which contains a collection of functionalities to control simCNC using macros.

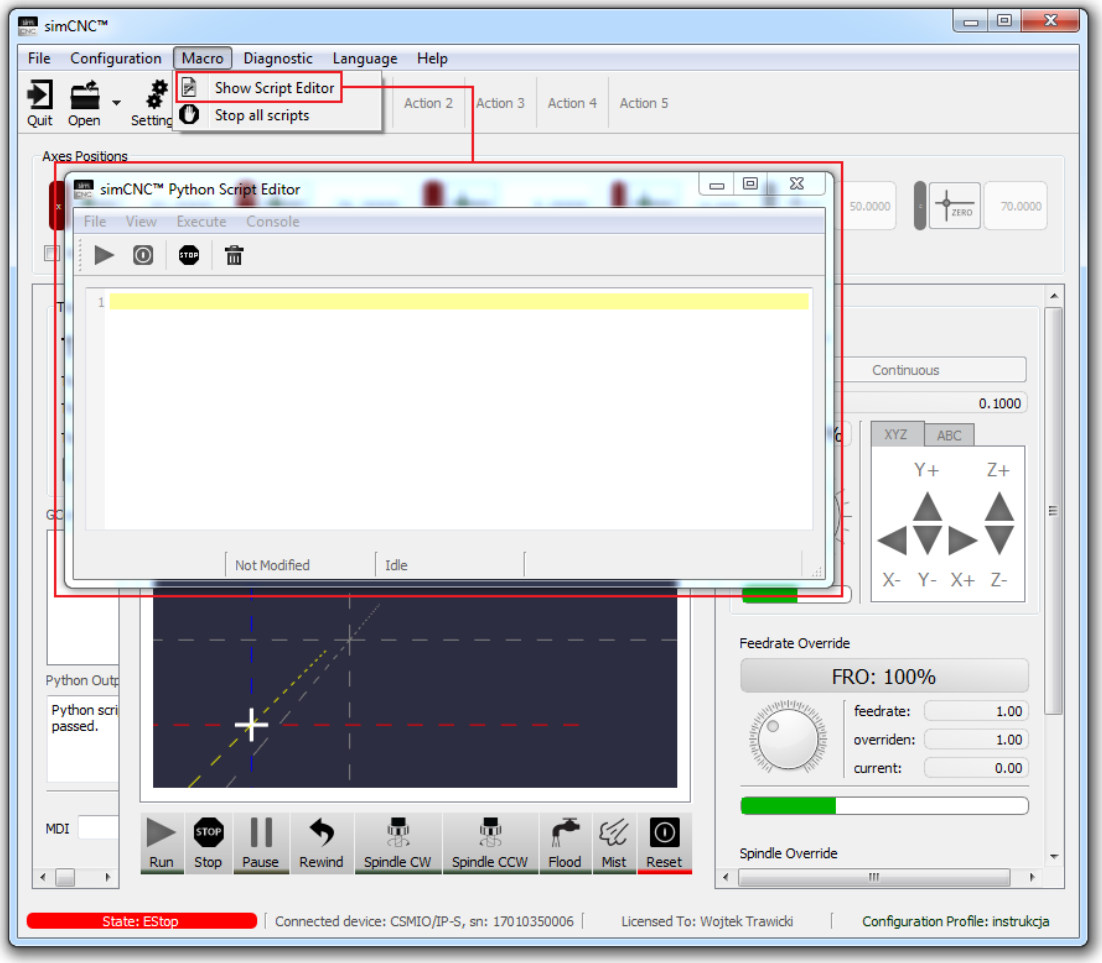
Such a solution makes it possible to:

- use an advanced external programming editor with debugging feature, such as Visual Studio Code.
- control simCNC from another PC across a local network via UDP.

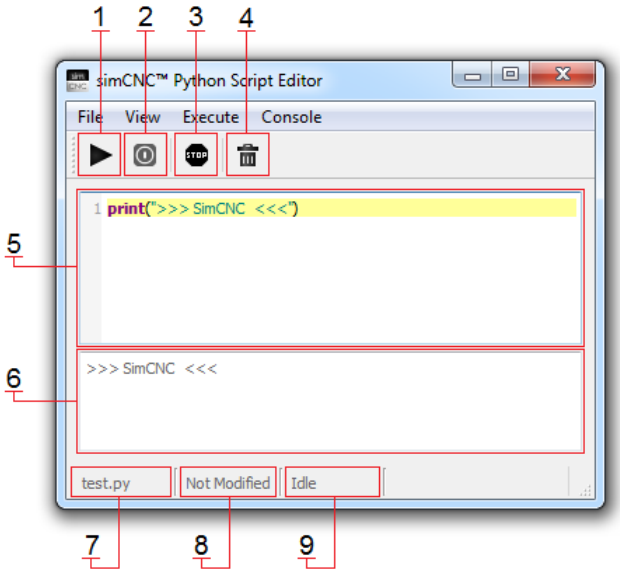
The two solutions will be addressed in more detail in supplementary documentation, whereas this manual describes the script editor incorporated into simCNC.

III. Python Script Editor - Start-Up And Description

To run the Python script editor, click on Macro in the upper bar of your simCNC window, and then select Show Script Editor from the drop-down list.



1. Run – run the script.
2. Stop – stop the script.
3. Stop Trajectory Planer – equivalent of the STOP button on the simCNC main screen.
4. Clear Console – clears the contents of the Python console's window (see item 6).
5. Python script edition – here you can create or edit scripts.
6. Python console – it is the place where information on scripts (errors and exceptions) and your own messages (see an example on the right) are shown.
7. Script name.
8. Information if the script has been modified since its last saving.
9. Script status.



IV. Ctrl + Space – Python Script Editor Prompts

The Python script editor is provided with a prompt mechanism. To launch it in the editor window, you need only to press Ctrl + space simultaneously. You can use this combination at any time.

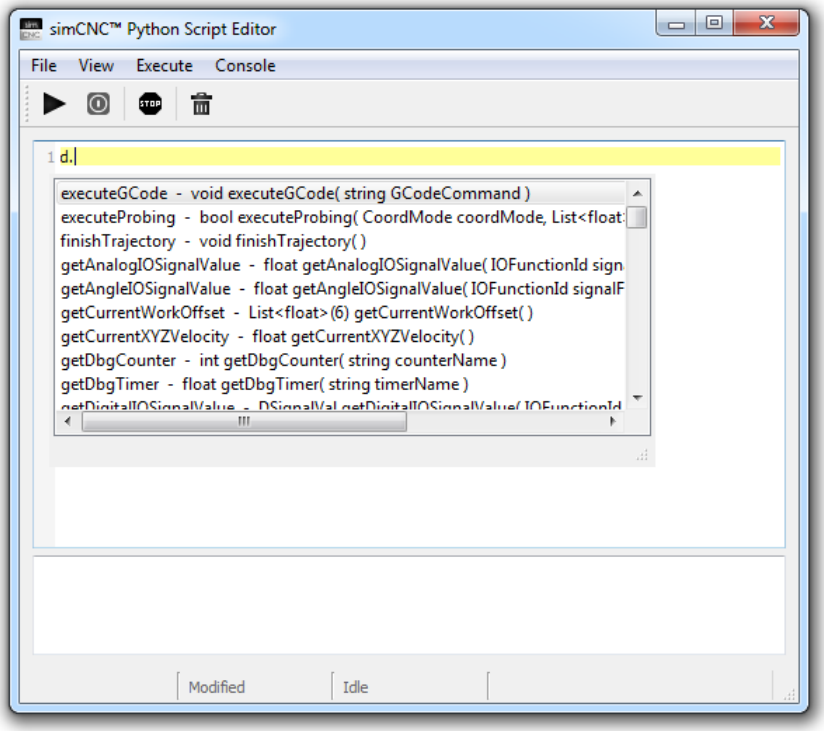
1) To view a list of the functions included in a ___DEVICE.pyc library

In such an event, enter „d." and press the key combination Ctrl + space. Then a window will appear containing a list of the functions included in a ___DEVICE.pyc library.

To select a function, highlight it using keyboard cursors and confirm the selection by pressing Enter, or simply click the selected function with the left mouse button.

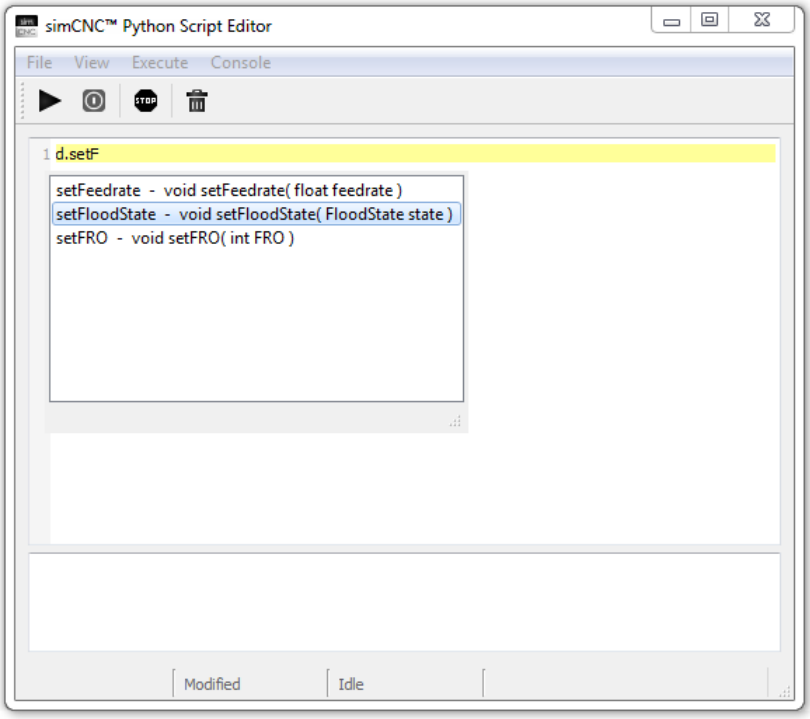
ATTENTION!

The list does not include functions that require getting a module and provide direct access to digital and analog ports.



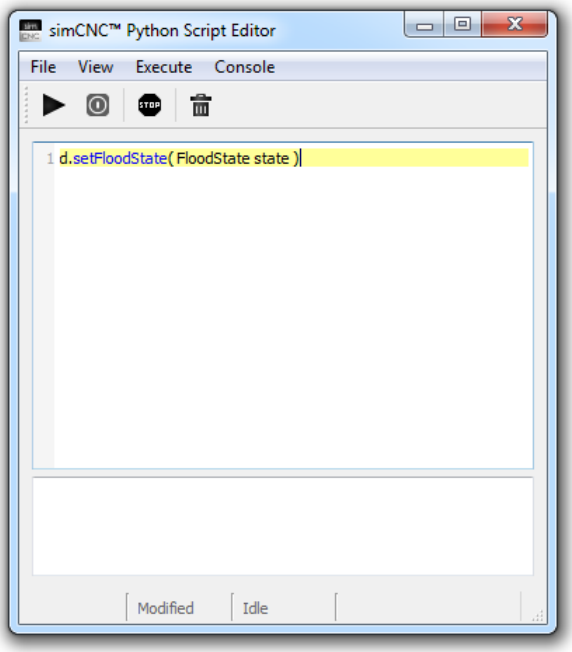
2) If you do not know the full name of a function:

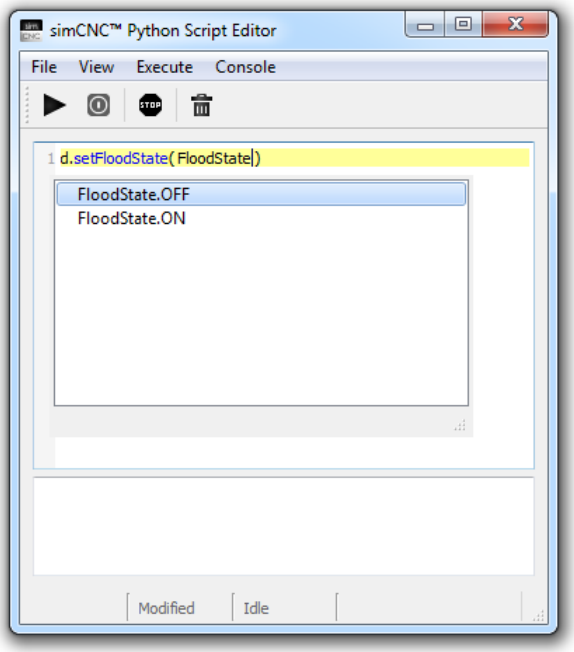
In such an event, enter „d.” and press the key combination Ctrl + space. This time you do not browse the list of available functions, but you enter the beginning of the function name, and the prompt window will show you matching endings.



3) If you have found the required function, but you do not know an argument value.

In such an event, all what you need to do is to delete the argument name and press the key combination Ctrl + space (if it was not pressed earlier). The prompt window will show you all the variants of the argument value that match a specific function.





V. Digital And Analog Port – Direct Access

1) Getting a Module

To access digital or analog ports of CS-LAB devices, you should first get a module (a module representing a device, a handle).

Module d.getModule(ModuleType type, int ID) – Returns a device module.

FUNCTION ARGUMENTS:

ModuleType type – Type of device

AVAILABLE OPTIONS:

ModuleType.IP	– CSMIO/IP-S, CSMIO/IP-A, and CSMIO/IP-M controllers
ModuleType.MPG	– CSMIO-MPG manual axis control module
ModuleType.ENC	– CSMIO-ENC threading module
ModuleType.IO	–CSMIO-IO extra I/O module
ModuleType.DRV	– simDrives

int ID – Device number.

AVAILABLE OPTIONS:

CSMIO-IO (ID 0 to 15)
SimDrive (ID 0 to 5)

If more than one CSMIO-IO extra I/O module or simDrive are used, you should enter the appropriate device number for their identification. This does not apply to the CSMIO/IP controller and the CSMIO-MPG and CSMIO-ENC modules. Their device number is 0 because they always appear in the control system one at a time.

FUNCTION RESULT:

Module - Device module

EXAMPLES:

```
mod_IP = d.getModule( ModuleType.IP, 0 )                # Getting a module to the CSMIO/IP
mod_MPG = d.getModule( ModuleType.MPG, 0 )              # Getting a module to CSMIO-MPG
mod_ENC = d.getModule( ModuleType.ENC, 0 )              # Getting a module to CSMIO-ENC
mod_IO_0 = d.getModule( ModuleType.IO, 0 )              # Getting a module to CSMIO-IO number 0
mod_IO_15 = d.getModule( ModuleType.IO, 15 )            # Getting a module to CSMIO-IO number 15
mod_simDrive_0 = d.getModule( ModuleType.DRV, 0 )       # Getting a module to simDrive number 0
mod_simDrive_5 = d.getModule( ModuleType.DRV, 5 )       # Getting a module to simDrive number 5
```

ATTENTION!

mod_IP, mod_MPG, mod_ENC, mod_IO_0, mod_IO_15, mod_simDrive_0, and mod_simDrive_5 are the proper names of the modules (handles) created for the purposes of the manual. A module can have any name, which should clearly state what devices it relates to.

2) Digital Ports

a) Reading digital ports

DIOPinVal getDigitalIO(IOPortDir direction, int digitalPin) – Returns the pin value of a digital input or output (returns the digital input or output value).

FUNCTION ARGUMENTS:

IOPortDir direction – Specifies the port direction

AVAILABLE OPTIONS:

IOPortDir.InputPort - Input port
IOPortDir.OutputPort - Output port

int digitalPin – Pin number (digital input or output number)

FUNCTION RESULT:

DIOPinVal – Pin value

AVAILABLE OPTIONS:

DIOPinVal.PinReset – Low pin value
DIOPinVal.PinSet – High pin value

EXAMPLES:

CSMIO/IP-S, CSMIO/IP-A, and CSMIO/IP-M controllers

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

# Reading digital input No. 8
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")
```

```
# Reading digital output No. 4
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 - JOG module

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

# Reading digital input No. 3
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")

# Reading digital output No. 0.
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 - I/O module - number 0

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

# Reading digital input No. 7
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")

# Reading digital output No. 2
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 - I/O module - number 15

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

# Reading digital input No. 11
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")
```

```
# Reading digital output No. 5
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) Saving to digital ports

void setDigitalIO(int digitalPin, DIOPinVal value) – Sets the pin value of a digital port (sets a digital output value).

FUNCTION ARGUMENTS:

int digitalPin – Pin number (digital output number)
DIOPinVal value – Pin value

AVAILABLE OPTIONS:

DIOPinVal.PinReset – Low pin value
DIOPinVal.PinSet – High pin value

EXAMPLE:

CSMIO/IP-S, CSMIO/IP-A, and CSMIO/IP-M controllers

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

# Setting the high value on digital output No. 1
mod_IP.setDigitalIO( 1, DIOPinVal.PinSet )
time.sleep(1)

# Setting the low value on digital output No. 1
mod_IP.setDigitalIO( 1, DIOPinVal.PinReset )
```

CSMIO-MPG - JOG module

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

# Setting the high value on digital output No. 0
mod_MPG.setDigitalIO( 0, DIOPinVal.PinSet )
time.sleep(1)

# Setting the low value on digital output No. 0
mod_MPG.setDigitalIO( 0, DIOPinVal.PinReset )
```

CSMIO-IO - I/O module - number 0

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

# Setting the high value on digital output No. 5
mod_IO_0.setDigitalIO( 5, DIOPinVal.PinSet )
time.sleep(1)

# Setting the low value on digital output No. 5
mod_IO_0.setDigitalIO( 5, DIOPinVal.PinReset )
```

CSMIO-IO - I/O module - number 15

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

# Setting the high value on digital output No. 3
mod_IO_15.setDigitalIO( 3, DIOPinVal.PinSet )
time.sleep(1)

# Setting the low value on digital output No. 3
mod_IO_15.setDigitalIO( 3, DIOPinVal.PinReset )
```

3) Analog Ports

a) Reading analog ports

float getAnalogIO(IOPortDir direction, int analogPin) – Returns the pin voltage of an analog input or output port (returns the analog input or output voltage).

FUNCTION ARGUMENTS:

IOPortDir direction – Specifies the port direction

AVAILABLE OPTIONS:

IOPortDir.InputPort - Input port
IOPortDir.OutputPort - Output port

int analogPin – Pin number (analog input or output number)

FUNCTION RESULT:

float – Pin voltage

EXAMPLES:

CSMIO/IP-S, CSMIO/IP-A, and CSMIO/IP-M controllers

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

# Encoder angle input No. 0
val = mod_IP.getAnalogIO( IOPortDir.InputPort, 0 )
print("CSMIO/IP analog input 0 = " + str(val) + "V")

# Reading the voltage of analog input No. 1
val = mod_IP.getAnalogIO( IOPortDir.InputPort, 1 )
print("CSMIO/IP analog input 1 = " + str(val) + "V")

# Reading the voltage of analog output No. 0
val = mod_IP.getAnalogIO( IOPortDir.OutputPort, 0 )
print("CSMIO/IP analog output 0 = " + str(val) + "V")

# Reading the voltage of analog output No. 1
val = mod_IP.getAnalogIO( IOPortDir.OutputPort, 1 )
print("CSMIO/IP analog output 1 = " + str(val) + "V")
```

CSMIO-MPG - JOG module

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

# Reading the voltage of analog input No. 0
val = mod_MPG.getAnalogIO( IOPortDir.InputPort, 0 )
print("CSMIO-MPG analog input 0 = " + str(val) + "V")

# Reading the voltage of analog input No. 1
val = mod_MPG.getAnalogIO( IOPortDir.InputPort, 1 )
print("CSMIO-MPG analog input 1 = " + str(val) + "V")
```

b) Saving to analog ports

void setAnalogIO(int analogPin, float value) – Sets the voltage of an analog port pin.

FUNCTION ARGUMENTS:

int analogPin – Pin number (analog output number)
float value – Pin voltage

EXAMPLES:

CSMIO/IP-S, CSMIO/IP-A, and CSMIO/IP-M controllers

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

# Setting a voltage of 5 V on analog output No. 0
val = 5
mod_IP.setAnalogIO( 0, val )

# Setting a voltage of 2 V on analog output No. 1
val = 2
mod_IP.setAnalogIO( 1, val )
```

ATTENTION!

CSMIO/IP-S, CSMIO/IP-A, and CSMIO/IP-M controllers

The pin voltage of analog input and output ports can range from 0 V to 10 V with a resolution of 12 bit.

CSMIO-MPG module for manual axis control

The pin voltage of the CSMIO-MPG manual axis control module's analog input port can range from 0 V to 5 V with a resolution of 12 bit.

4) Encoder Ports - CSMIO-ENC

a) Reading encoder angles

float getEncoderIOAngle(int channel = 0) - Returns the encoder angle as calculated from the index signal.

FUNCTION ARGUMENTS:

int channel – Encoder channel

FUNCTION RESULT:

float – Encoder angle

EXAMPLE:

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

# Reading the angle of encoder No. 0
val = mod_ENC.getEncoderIOAngle( 0 )
print("CSMIO-ENC encoder chanel 0 angle = " + str(val))
```

b) Reading encoder positions

int getEncoderIOPostion(int channel = 0) – Returns the encoder position as calculated from the first index signal.

FUNCTION ARGUMENTS:

int channel – Encoder channel

FUNCTION RESULT:

int – Encoder position

EXAMPLE:

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

# Reading the position of encoder No. 2
val = mod_ENC.getEncoderIOPosition( 2 )
print("CSMIO-ENC encoder chanel 0 postion = " + str(val))
```

c) Reading encoder velocities

float getEncoderIORPM(int channel = 0) – Returns the encoder's rotational speed (RPM).

FUNCTION ARGUMENTS:

int channel – Encoder channel

FUNCTION RESULT:

float – Encoder rotational speed

EXAMPLE:

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

# Reading the rotational speed of encoder No. 4
val = mod_ENC.getEncoderIORPM( 4 )
print("CSMIO-ENC encoder chanel 0 rpm = " + str(val))
```

ATTENTION!

Threading module channels 0, 2, and 4 are physical channels, whereas channels 1, 3, and 5 are virtual copies of the physical channels to be used in the future.

Until further notice, physical channels 0, 2, and 4 in simCNC configuration correspond to channels 0, 1, and 2.

VI. Reading And Saving Axial Coordinates

1) Reading Coordinates

List<float>(6) getPosition(CoordMode coordMode) - Returns the current machine or program coordinates for all the axes.

FUNCTION ARGUMENTS:

CoordMode coordMode - Specifies the type of coordinates

AVAILABLE OPTIONS:

CoordMode.Machine – Machine coordinates
CoordMode.Program – Program coordinates

FUNCTION RESULT:

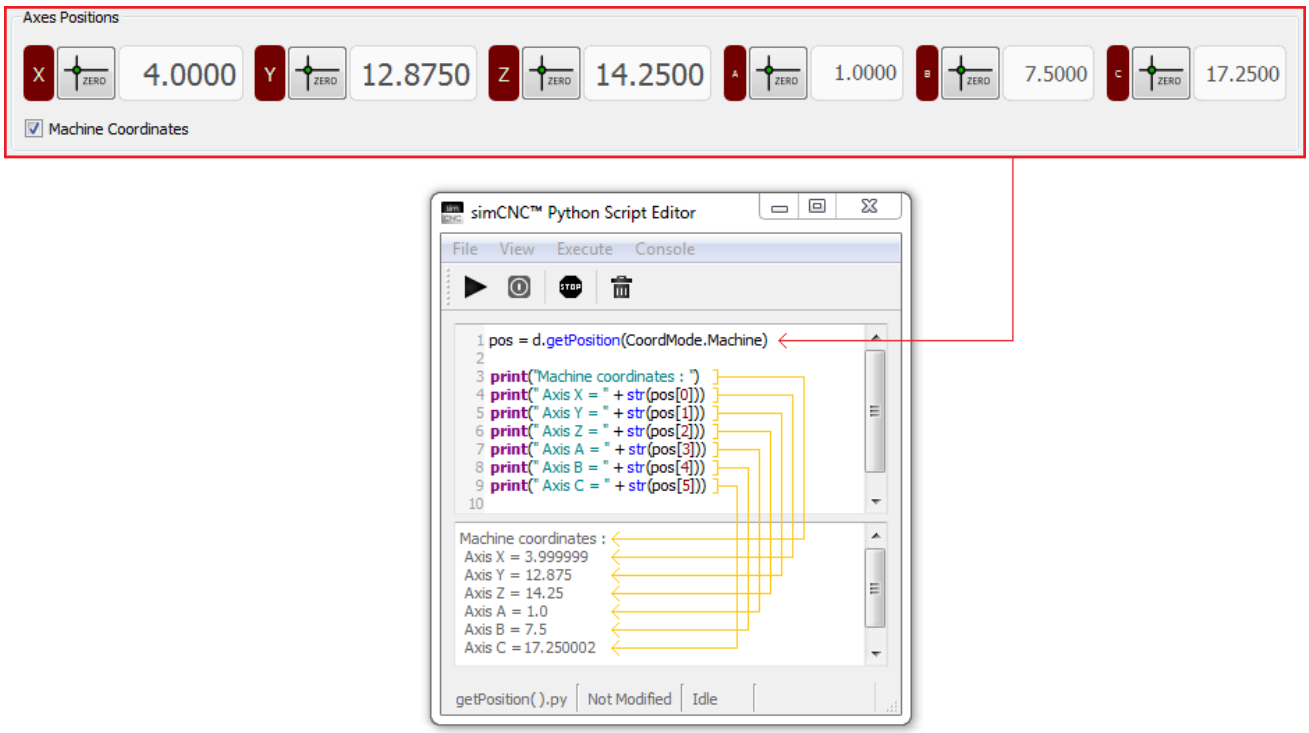
List<float>(6) – a list of the six coordinates (X, Y, Z, A, B, C) of the current position

EXAMPLES:

Reading the machine coordinates of axes X, Y, Z, A, B, and C

```
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]))
```

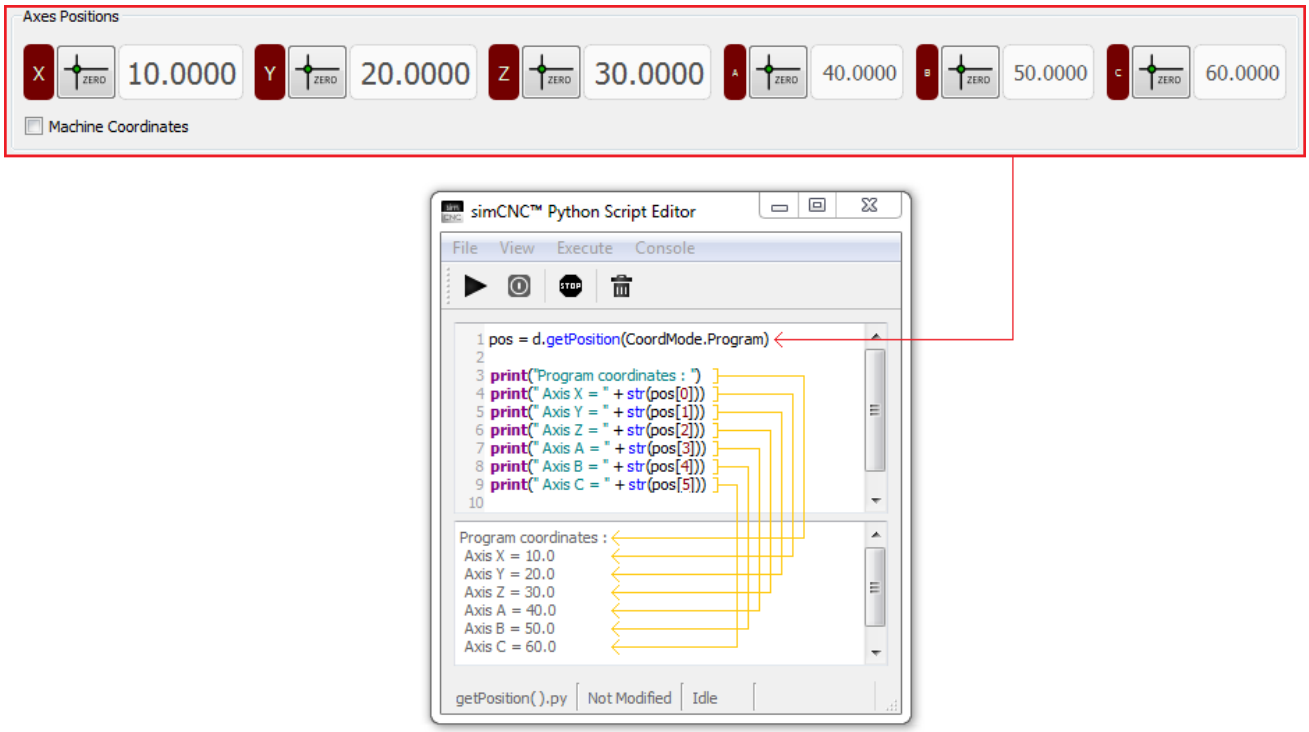
After running the example presented above, the d.getPosition(CoordMode.Machine) function will read the machine coordinates of all six axes and put them on a six-item list, which is indexed from 0 through 5. Then use function print( ) to display in the Python console the machine coordinates that are stored in the list.



Reading the program coordinates of axes X, Y, Z, A, B, and C

```
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]))
```

After running the example presented above, the d.getPosition(CoordMode.Machine) function will read the machine coordinates of all six axes and put them on a six-item list, which is indexed from 0 through 5. Then use function print( ) to display in the Python console the machine coordinates that are stored in the list.



2) Saving Coordinates

void setAxisProgPosition(Axis axis, float value) – Sets the program coordinates of a selected axis.

FUNCTION ARGUMENTS:

Axis axis – Axis for which the new program coordinates will be saved

AVAILABLE OPTIONS:

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

float value – New program coordinates

EXAMPLES:

Saving the program coordinates of axes X, Y, Z, A, B, and C.

```
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 )
```

The same but using a list.

```
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. Moving the Machine's Axes

void moveAxisIncremental(Axis axis, float distance, float velocity) – Moves a selected axis by a set distance at a preset velocity (incremental movement).

ARGUMENTS:

Axis axis – Moving axis

AVAILABLE OPTIONS:

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

float distance – Distance of the motion
float velocity – Velocity of the motion

EXAMPLES:

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

After running the example presented above, axes X and Y move from the current position, along the circumference of a 20 mm or 20" wide square with a velocity of 1000 mm/min or inch/min.

void moveToPosition(CoordMode coordMode, List<float>[6] position, float velocity) – Moves to the set position expressed in machine or program coordinates at a preset resultant velocity.

ARGUMENTS:

CoordMode coordMode – Specified the type of coordinates

AVAILABLE OPTIONS:

- CoordMode.Machine – Machine coordinates
- CoordMode.Program – Program coordinates

List<float>[6] position – A list of the six coordinates (X, Y, Z, A, B, C) of the final position
float velocity – Set resultant velocity

EXAMPLES:

```
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 )
```

After running the example presented above, axes X and Y will use the program coordinates to move from the current position, along the circumference of a 20 mm or 20” wide square with a velocity of 1000 mm/min or inch/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 )
```

After running the example presented above, axes X and Y will use the machine coordinates to move from the current position, along the circumference of a 20 mm or 20” wide square with a velocity of 1000 mm/min or inch/min.

The two examples show how lists can be used to modify coordinates.
You will see such a mode of calculation in scripts available from CS-LAB’s website. Therefore, it is worth taking a closer look at it already now.

void parallelMoveToPosition(CoordMode coordMode, Axis axis, float position, float velocity) – Moves a selected independent axis to the set position expressed in machine or program coordinates at a preset resultant velocity. The functionality is available if the option "Use external points" is left unselected.

ARGUMENTS:

CoordMode coordMode - Specifies the type of coordinates

AVAILABLE OPTIONS:

CoordMode.Machine – Machine coordinates
CoordMode.Program – Program coordinates

Axis axis – Independent moving axis

AVAILABLE OPTIONS:

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

float position – final position of the axis
float velocity – Preset velocity

EXAMPLES:

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

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

After running the example presented above, axis X will use the program coordinates to move to 25 mm or inches with a velocity of 100 mm/min or inch/min and then return to the starting point with a velocity of 20 mm/min or inches/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 )
```

After running the example presented above, axis Z will use the program coordinates to move to 78 mm or inches with a velocity of 350 mm/min or inches /min and then return to the starting point with a velocity of 500 mm/min or inches/min.

float getCurrentXYZVelocity( ) - Returns the current resultant velocities of axes X, Y, and Z.

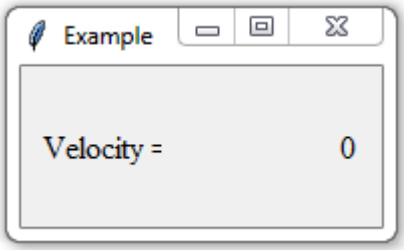
FUNCTION RESULT:

float – Resultant velocity

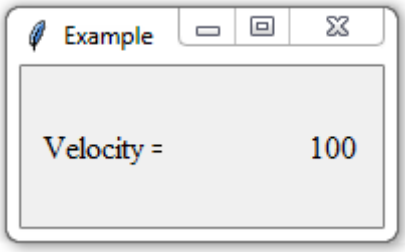
EXAMPLES:

```
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()
```

After running the example presented above, the script will cause a window to appear showing the current resultant velocities of axes X, Y, and Z. On the right, you can see a picture showing the situation where none of the axes is moving.



To make sure that the script is running, select the Keyboard control option or use the combination Alt + j and move an axis or axes with keyboard cursors.



VIII. Probing

bool executeProbing(CoordMode coordMode, List<float>[6] position, int probeIndex, float velocity) – Performs probing with a selected probe up to the set position expressed in machine or program coordinates with a preset resultant velocity.

ARGUMENTS:

CoordMode coordMode – Specifies the type of coordinates

AVAILABLE OPTIONS:
CoordMode.Machine – Machine coordinates
CoordMode.Program – Program coordinates

List<float>[6] position – A list of the six coordinates (X, Y, Z, A, B, C) of the final position
int probeIndex – Probe number (probe configurations can be found in Settings/Modules/Special IO)
float velocity – Resultant velocity

FUNCTION RESULT:

bool – Probing result

AVAILABLE OPTIONS:
True – If the probe was activated
False – If an axis or axes reached the final position and the probe was not activated

EXAMPLES can be found in the description of the next functionality

List<float>[6] getProbingPosition(CoordMode coordMode) - Returns the activation position of the probe or the final position expressed in machine or program coordinates. The functionality returns the final position only if the probe is not activated during the probing.

ARGUMENTS:

CoordMode coordMode - Specifies the type of coordinates

AVAILABLE OPTIONS:
CoordMode.Machine - Machine coordinates
CoordMode.Program - Program coordinates

FUNCTION RESULT:

List<float>[6] - a list of the six coordinates (X, Y, Z, A, B, C) of the probe activation position or the final position.

EXAMPLES of both options described above:

```
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" )
```

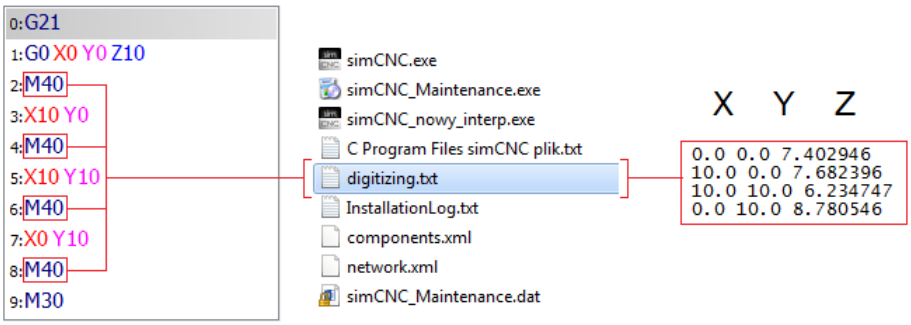
The script described above shows how to perform probing using axis Z, starting from the current position across a distance defined by the Distance parameter. Once probing, whatever its result, is completed, axis Z always returns to its starting position. Probing_vel is a parameter that relates to probing velocity and Return_vel to the velocity of the probe returning to the starting position. The probing result is shown in a pop-up window.

Digitization Made Easy!

Slightly modified, the script presented above can be used as a macro to record a grid of points across a probed area. A grid of points is most often recorded when copying a relief or when milling or engraving a rough surface, e.g. stone.

```
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 !!!")
```

Now you only need to save the script shown above e.g. under the name “M40” and execute it from gcode at some intervals. Upon execution, the script performs probing and save the probe’s activation position and those of X and Y in the file C:\Program Files\simCNC\digitizing.txt. Below you can see how the macro works using as an example some simple gcode that creates motion along the circumference of a 10 x 10 large square.



Should the probe fail to activate within a distance defined as Distance in the file digitizing.txt, a Probing Failed !!! warning will be inserted.

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

VIII. Tool Changer Magazine

1) Tool Number

int getSelectedToolNumber( ) - Returns the number of a tool selected from Gcode, the MDI line, or a Python script.

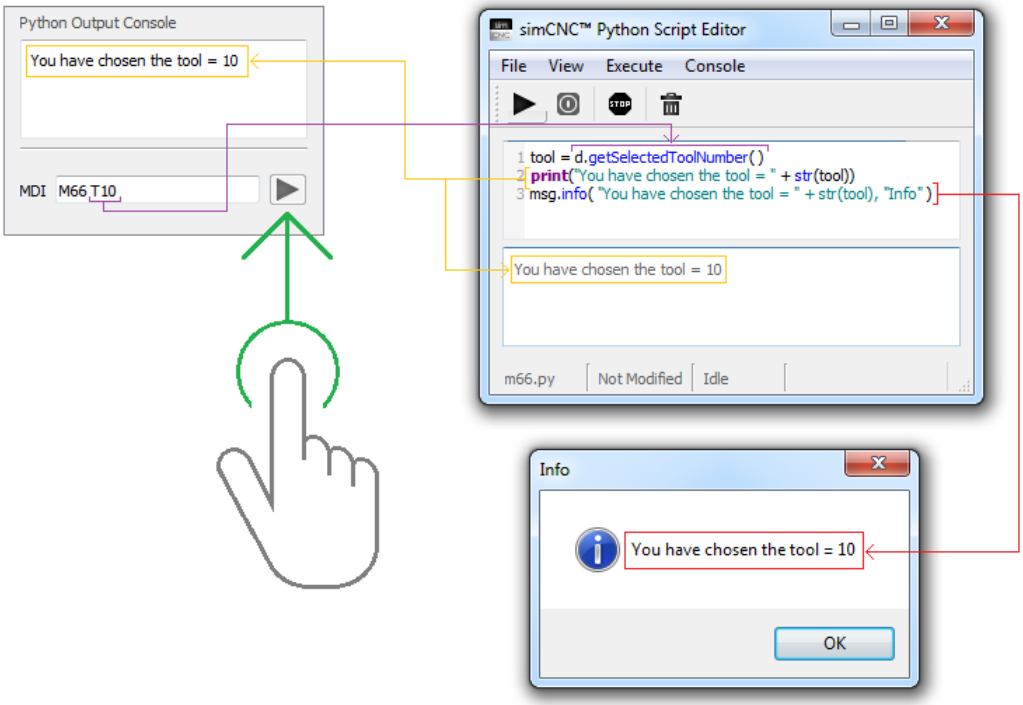
FUNCTION RESULT:

int – Tool number

EXAMPLE:

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

Save this script e.g. as M66, and then execute it from Gcode or the MDI line adding the command T + any tool number (e.g. M66 T10). Then a window should appear informing you which tool number is selected. For instance, we used the MDI line and tool No. 10. We recommend using the macro M66 on purpose to show you that the d.getSelectedToolNumber( ) function works with any macro, and not only with M6.



void setSelectedToolNumber(int toolNumber) – Sets the number of a selected tool – it is equivalent to the Txx command executed from Gcode or the MDI line.

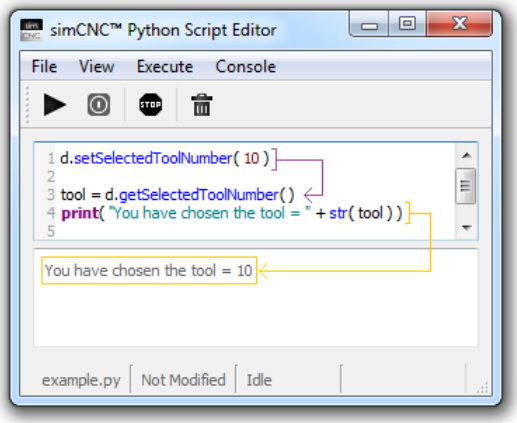
ARGUMENTS:

int toolNumber – Tool number

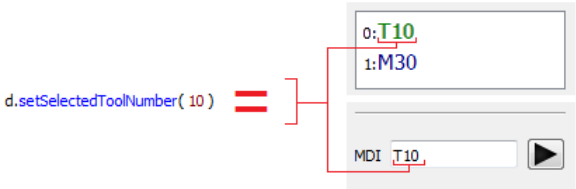
EXAMPLE:

```
d.setSelectedToolNumber( 10 )                    # Sets the number of a selected tool

tool = d.getSelectedToolNumber( )                # Returns the number of a selected tool
print( "You have chosen the tool = " + str( tool ) )
```

After running the example presented above, the d.setSelectedToolNumber( ) function will set the number of the selected tool to 10, and the d.getSelectedToolNumber( ) function, which is already known to you, will return the same tool number.



As you can easily see now, the d.setSelectedToolNumber(10) function makes exactly the same as the T10 command executed from Gcode or the MDI line.



void setSpindleToolNumber(int toolNumber) – Sets the number of a tool in the spindle.

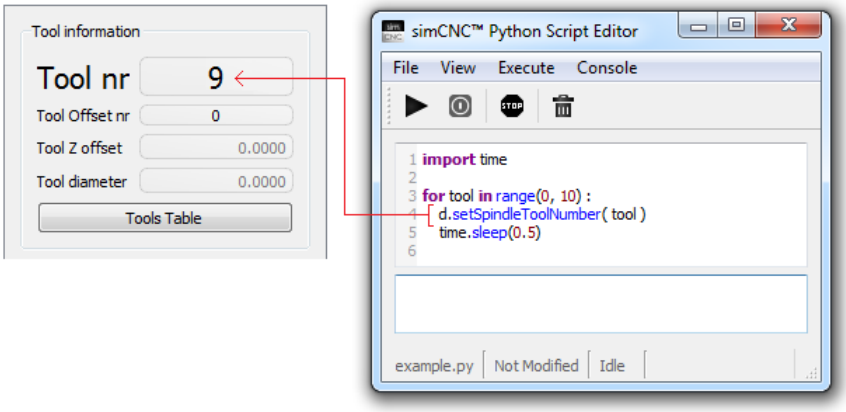
ARGUMENTS:

int toolNumber – Tool number

EXAMPLE:

```
import time

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

After running the example presented above, every 0.5 seconds the d.setSpindleToolNumber(tool) function in a for loop will increase the tool number on the simCNC screen until it reaches 9.



int getSpindleToolNumber( ) - Returns the number of a tool in the spindle.

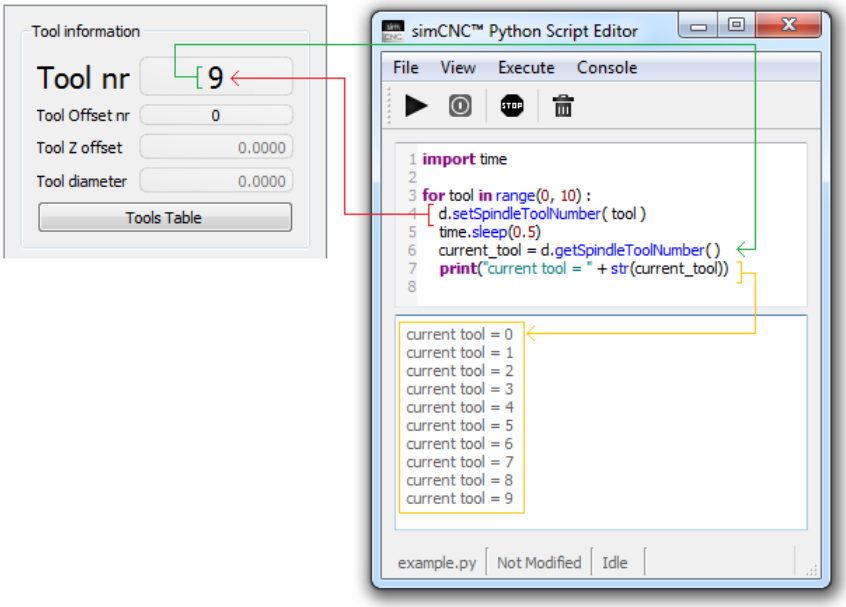
FUNCTION RESULT:

int – Tool number

EXAMPLE:

```
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))
```

This script is a continuation of the previous example. This time the for loop comes with the current_tool = d.getSpindleToolNumber( ) function, which with each run of the for loop returns the current number of the tool in the spindle. Then the tool number is presented in the Python console using the "print" function.



2) Tool Length

void setToolLength(int toolNumber, float toolLength) – Sets a tool length offset.

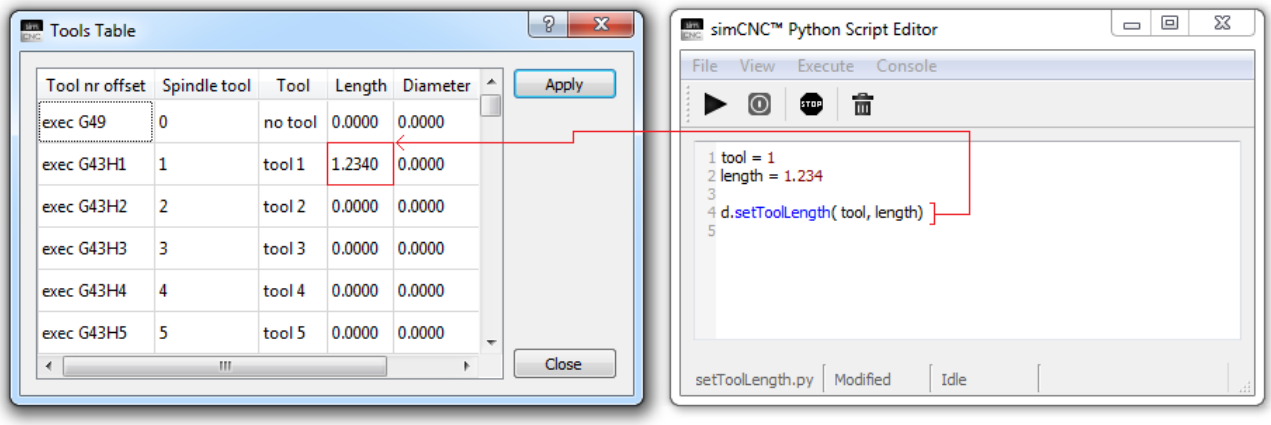
ARGUMENTS:

int toolNumber – Tool number
float toolLength – Tool length offset

EXAMPLE:

```
tool = 1
length = 1.234

d.setToolLength( tool, length )
```

After running the example presented above, the length offset of tool No. 1 will be set to 1.234.



float getToolLength(int toolNumber) - Returns the tool length offset.

ARGUMENTS:

int toolNumber – Tool number

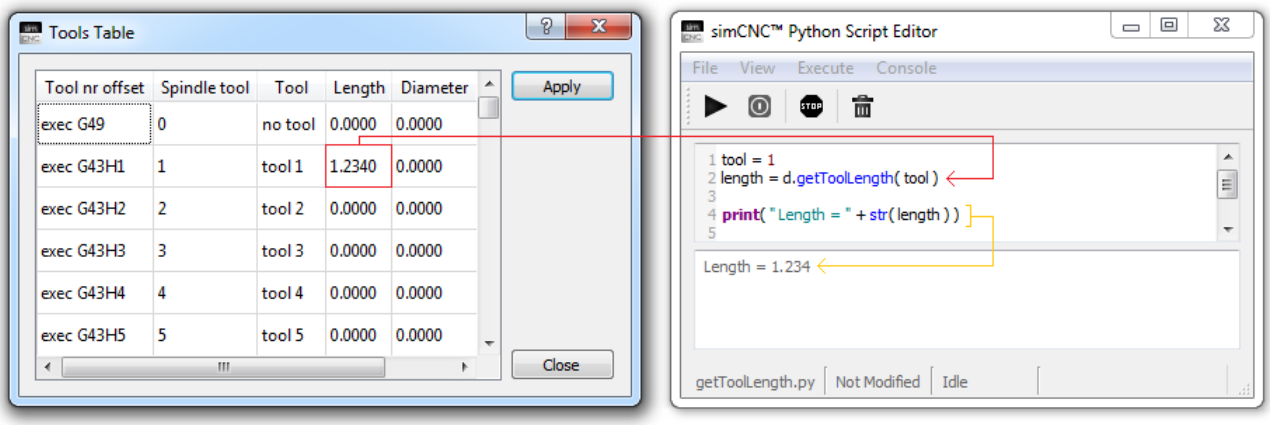
FUNCTION RESULT:

float - Tool length offset value

EXAMPLE:

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

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

After running the example presented above, the length offset of tool No. 1 will be shown in the Python console.



void setToolOffsetNumber(int toolNumber) – Sets the number of a tool length offset.

ARGUMENTS:

int toolNumber – Tool number

EXAMPLE:

```
tool = 1

d.setToolOffsetNumber( tool )
```

After running the example presented above, the tool length offset number is set to 1.

int getToolOffsetNumber( ) - Returns the number of the current tool length offset.

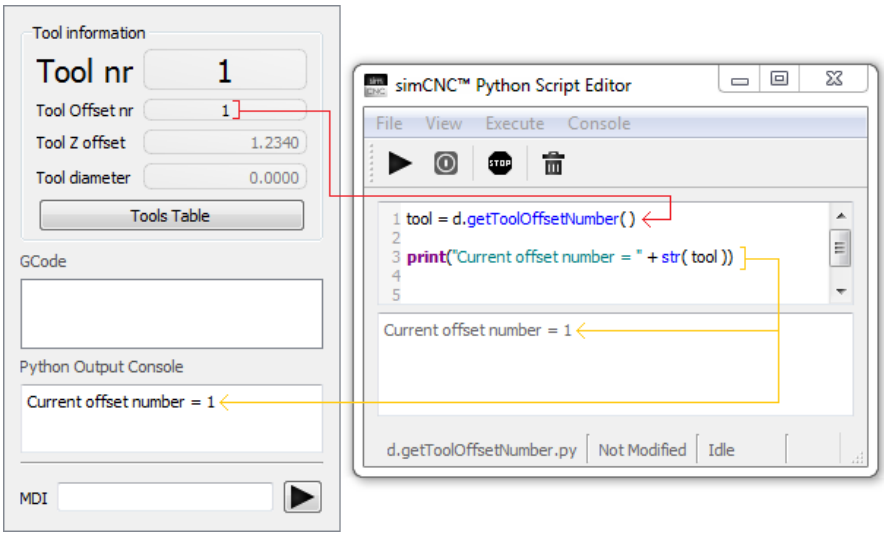
ARGUMENTS:

int – Tool number

EXAMPLE:

```
tool = d.getToolOffsetNumber( )

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

After running the example presented above, the number of the current length offset tool will be shown in the Python console.



3) Tool Diameter

ATTENTION!

At present, SimCNC does not use tool diameters, which means that no tool diameter compensation is currently supported.

void setToolDiameter(int toolNumber, float toolDiameter) – Sets a tool diameter.

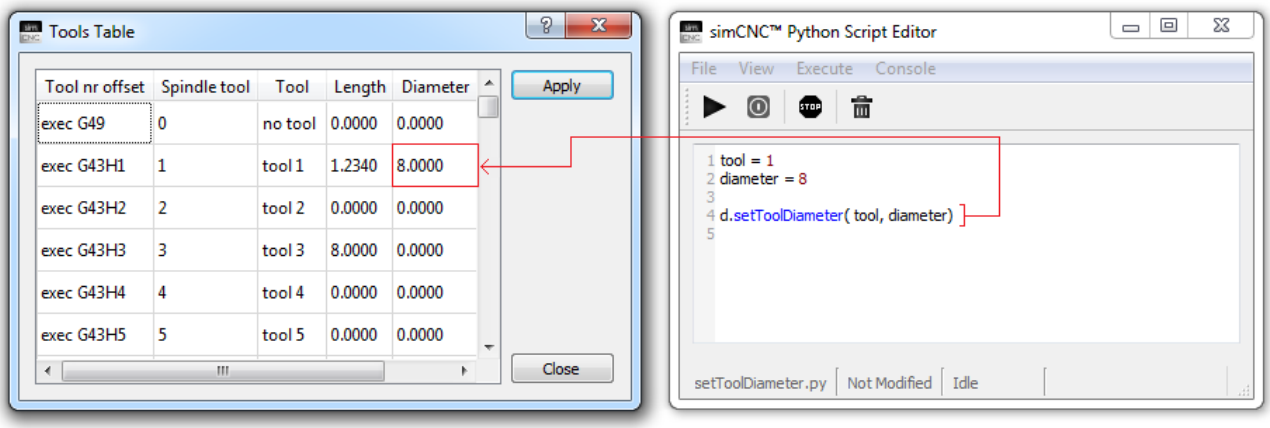
ARGUMENTS:

int toolNumber - Tool number
float toolDiameter - Tool diameter

EXAMPLE:

```
tool = 1
diameter = 8

d.setToolDiameter( tool, diameter )
```

After running the example presented above, the diameter of tool No. 1 will be set to 8.



float getToolDiameter(int toolNumber) - Returns the tool diameter.

ARGUMENTS:

int toolNumber – Tool number

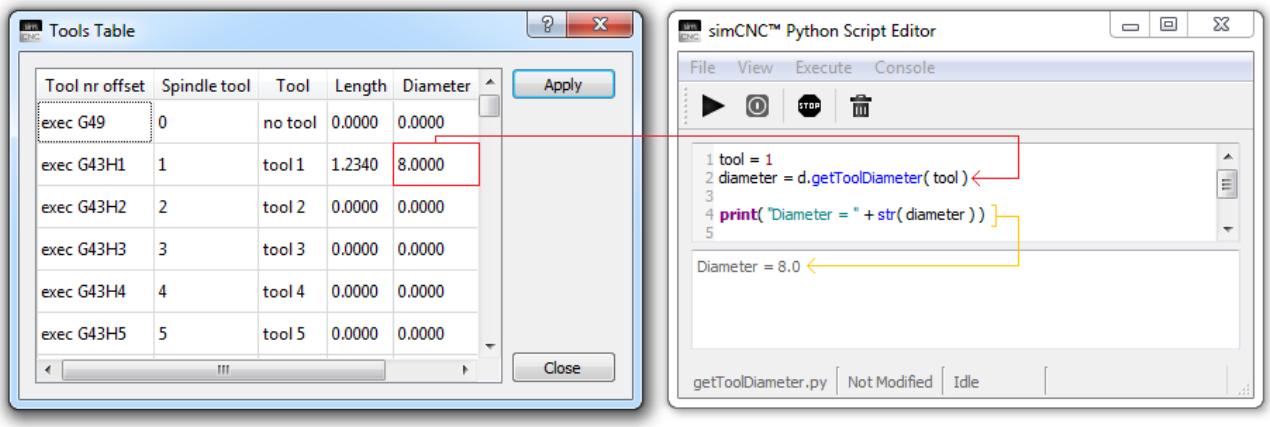
FUNCTION RESULT:

float – Tool diameter

EXAMPLE:

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

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

After running the example presented above, the diameter of tool No. 1 will be shown in the Python console.



IX. Spindle, Coolant, and Mist

1) Spindle

void setSpindleSpeed(float spindleSpeed) – Sets a preset rotational speed (RPM) of the spindle.

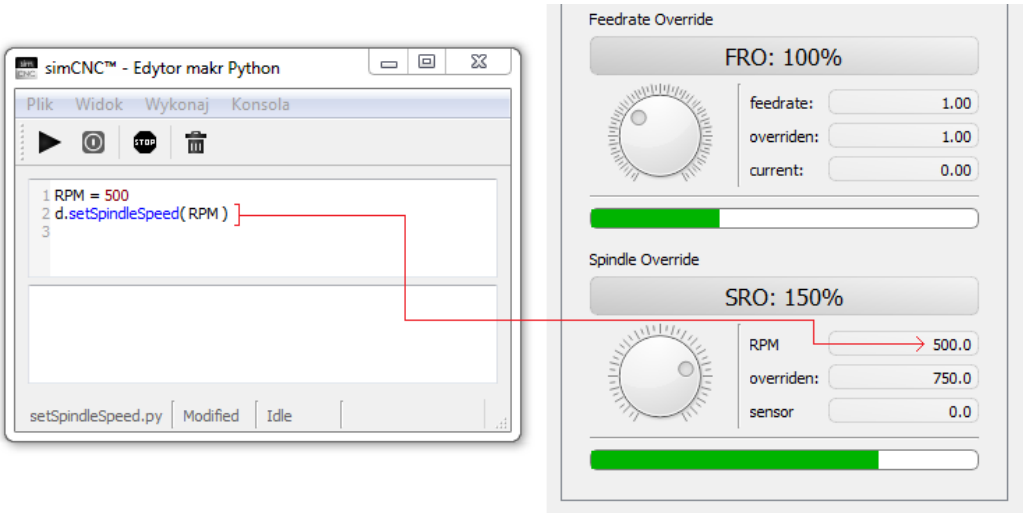
ARGUMENTS:

float spindleSpeed – preset rotational speed (RPM) of the spindle

EXAMPLE:

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

After running the example presented above, the preset rotational speed of the spindle will be set to 500 RPM.



float getSpindleSpeed( ) - Returns the preset rotational speed (RPM) of the spindle.

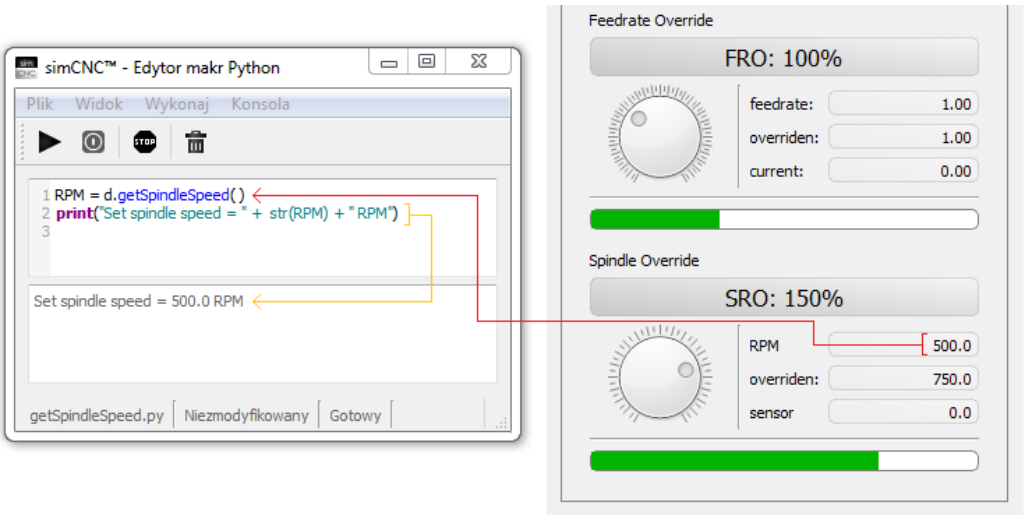
FUNCTION RESULT:

float – preset rotational speed (RPM)

EXAMPLE:

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

After running the example presented above, the preset rotational speed of the spindle will be shown in the Python console.



void setSpindleState(SpindleState spindleState) – Sets the spindle state

ARGUMENTS:

SpindleState spindleState – Spindle state

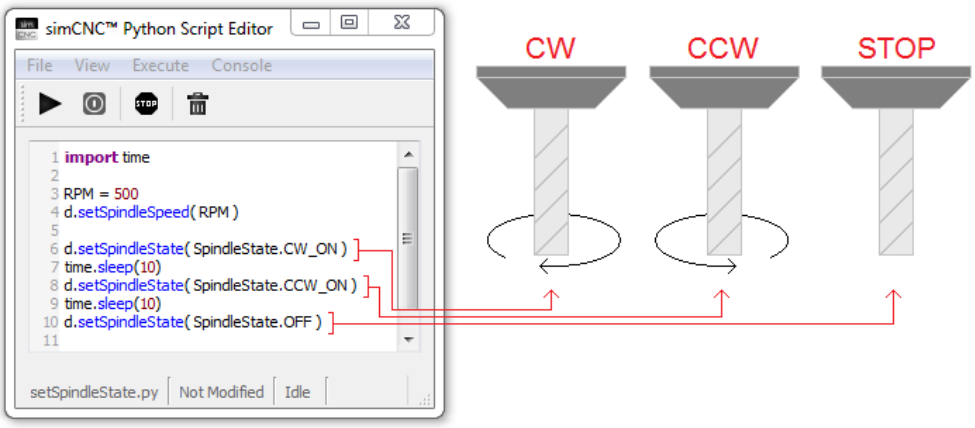
AVAILABLE OPTIONS:

SpindleState.CW_ON	- Clockwise rotation of the spindle
SpindleState.CCW_ON	- Counterclockwise rotation of the spindle
SpindleState.OFF	- No rotation of the spindle

EXAMPLE:

```
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 )
```

After running the example presented above, the spindle rotates clockwise for 10 seconds, then it switches to counterclockwise rotation, and switches off after 10 seconds.
Please note that the setSpindleState functions suspends the script for the duration of the spindle acceleration and deceleration ramps.



SpindleState getSpindleState( ) - Returns the current spindle state.

FUNCTION RESULT:

SpindleState – Spindle state

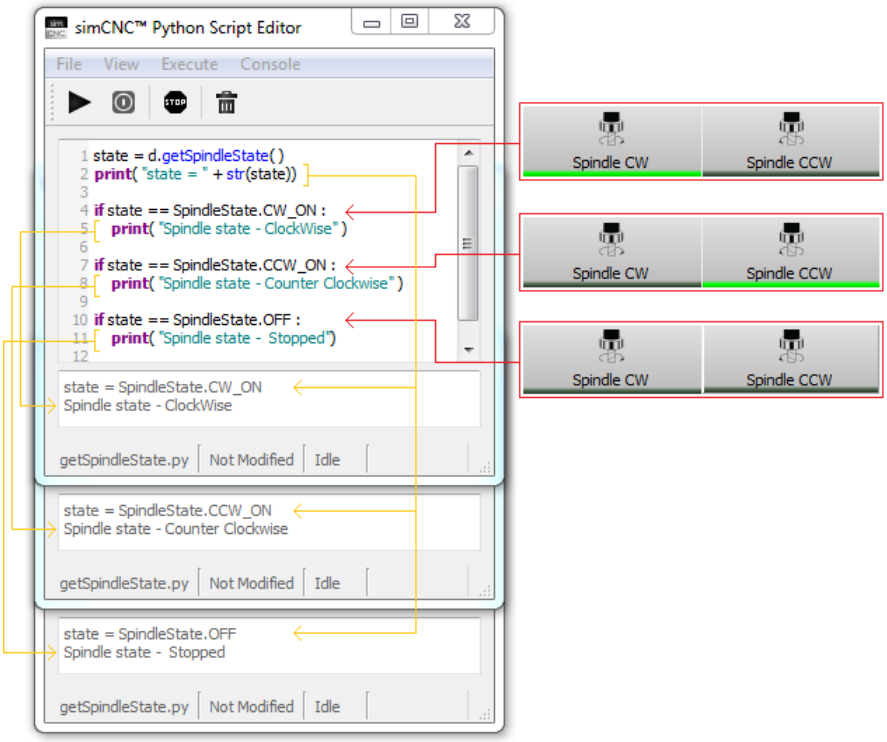
AVAILABLE OPTIONS:

SpindleState.CW_ON	- Clockwise rotation of the spindle
SpindleState.CCW_ON	- Counterclockwise rotation of the spindle
SpindleState.OFF	- No rotation of the spindle

EXAMPLE:

```
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")
```

After running the example presented above, the current state of the spindle resulting from the d.getSpindleState( ) function will be shown in the first line of the Python console. In the second line, your own message will be shown to provide a more detailed description of the current spindle state.
The figure below shows how the script works like for all spindle states.



void waitForSpindleSetSpeed(int timeout_sec) – Waits until the spindle exceeds a specific rotational speed for a certain duration. The expected rotational speed of the spindle is defined by a preset rotational speed and the Spindle Ready Level parameter.

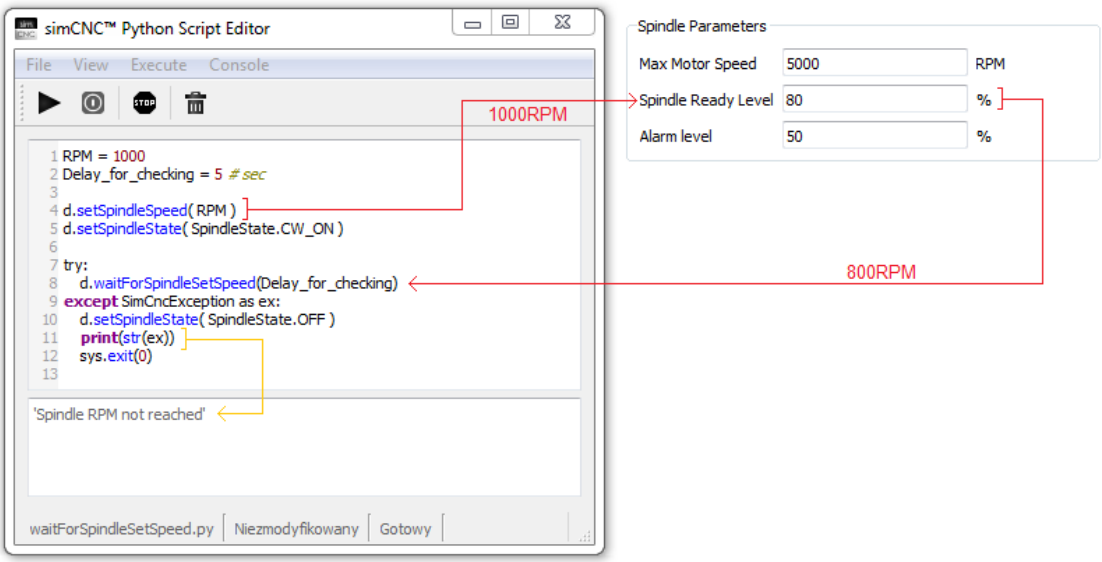
ARGUMENTS:

int timeout_sec – Timeout (in seconds)

EXAMPLE:

```
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)
```

After running the example presented above, the spindle starts to rotate clockwise (line 5 – see the figure below) with a preset rotational speed of 1000 RPM (line 4). Then the waitForSpindleSetSpeed function suspends the execution of the script until the spindle reaches 80% of its preset rotational speed, i.e. 800 RPM (80% of 1000 RPM = 800 RPM). If the spindle does not reach 800 RPM within 5 seconds, the waitForSpindleSetSpeed function will report a SimCncException (line 9). This will stop the spindle (line 10), show the contents of the exception in the Python console (line 11) and terminate the script upon request (line 12) so that it is not executed anymore. The example described above requires CSMIO-ENC for its execution because it provides information on the current rotational speed of the spindle.



Slightly modified, the previous example can be used as a M3 macro to suspend the execution of gcode until the spindle accelerates up to the expected rotational speed or stop the machine should there be any problem with reaching such a rotational speed.

```
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) Coolant

void setFloodState(FloodState state) – Sets the state of liquid coolant.

ARGUMENTS:

FloodState state - State of liquid coolant

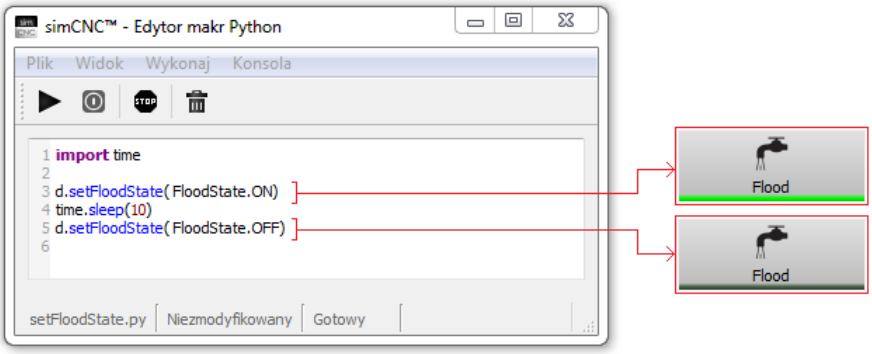
AVAILABLE OPTIONS:

FloodState.ON	- Coolant activation
FloodState.OFF	- Coolant deactivation

EXAMPLE:

```
import time

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

After running the example presented above, the coolant will be enabled for 10 seconds.



FloodState getFloodState( ) - Returns the state of liquid coolant.

FUNCTION RESULT:

FloodState - State of liquid coolant

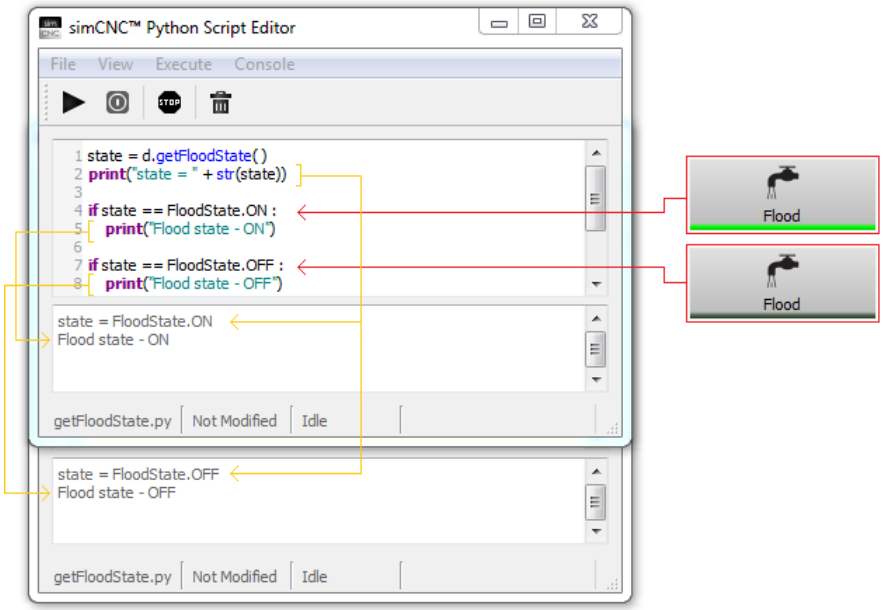
AVAILABLE OPTIONS:

FloodState.ON - Coolant enabled
FloodState.OFF - Coolant disabled

EXAMPLE:

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

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

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

After running the example presented above, the coolant state resulting from the d.getFloodState( ) function will be shown in the first line of the Python console. In the second line, your own message will be shown to provide a more detailed description of the current coolant state.
The figure below shows how the script works like for all coolant states.



3) Mist

void setMistState(MistState state) – Sets the state of cooling mist.

ARGUMENTS:

MistState state – State of cooling mist

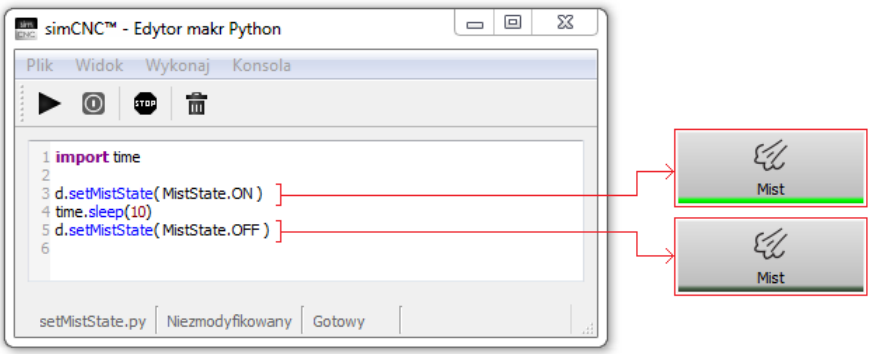
AVAILABLE OPTIONS:

MistState.ON - Mist activation
MistState.OFF - Mist deactivation

EXAMPLE:

```
import time

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

After running the example presented above, the mist will be enabled for 10 seconds.



MistState getMistState( ) - Returns the state of cooling mist.

FUNCTION RESULT:

MistState – State of cooling mist

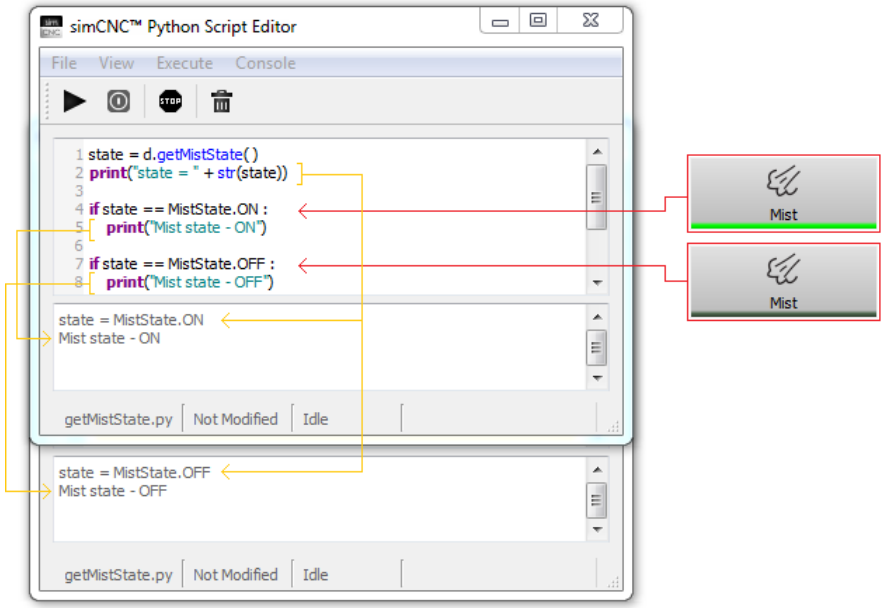
AVAILABLE OPTIONS:

MistState.ON - Mist enabled
MistState.OFF - Mist disabled

EXAMPLE:

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

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

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

After running the example presented above, the mist state resulting from the getMistState( ) function will be shown in the first line of the Python console. In the second line, your own message will be shown to provide a more detailed description of the current mist state.
The figure below shows how the script works like for all mist states.



X. Work Offset – Material Databases

List<float>[6] getCurrentWorkOffset( ) – Returns the coordinates of the current work offset.

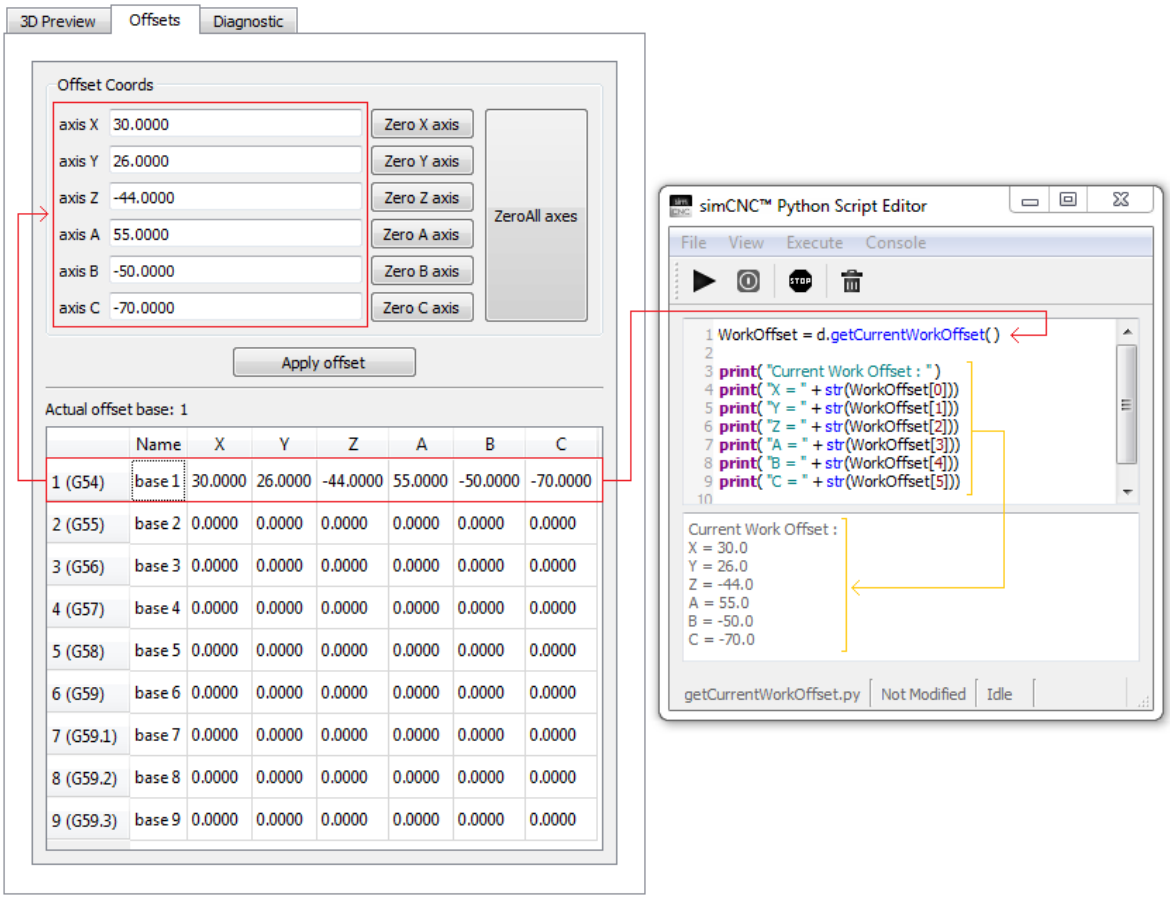
FUNCTION RESULT:

List<float>[6] - a list of the six coordinates (X, Y, Z, A, B, C) of the current work offset

EXAMPLE:

```
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]))
```

After running the example presented above, the coordinates of the current work offset will be read and shown in the Python console.



List<float>[6] getWorkOffset(int workOffsetIndex) - Returns the coordinates of the work offset with a predefined index.

ARGUMENTS:

int workOffsetIndex – work offset index (number)

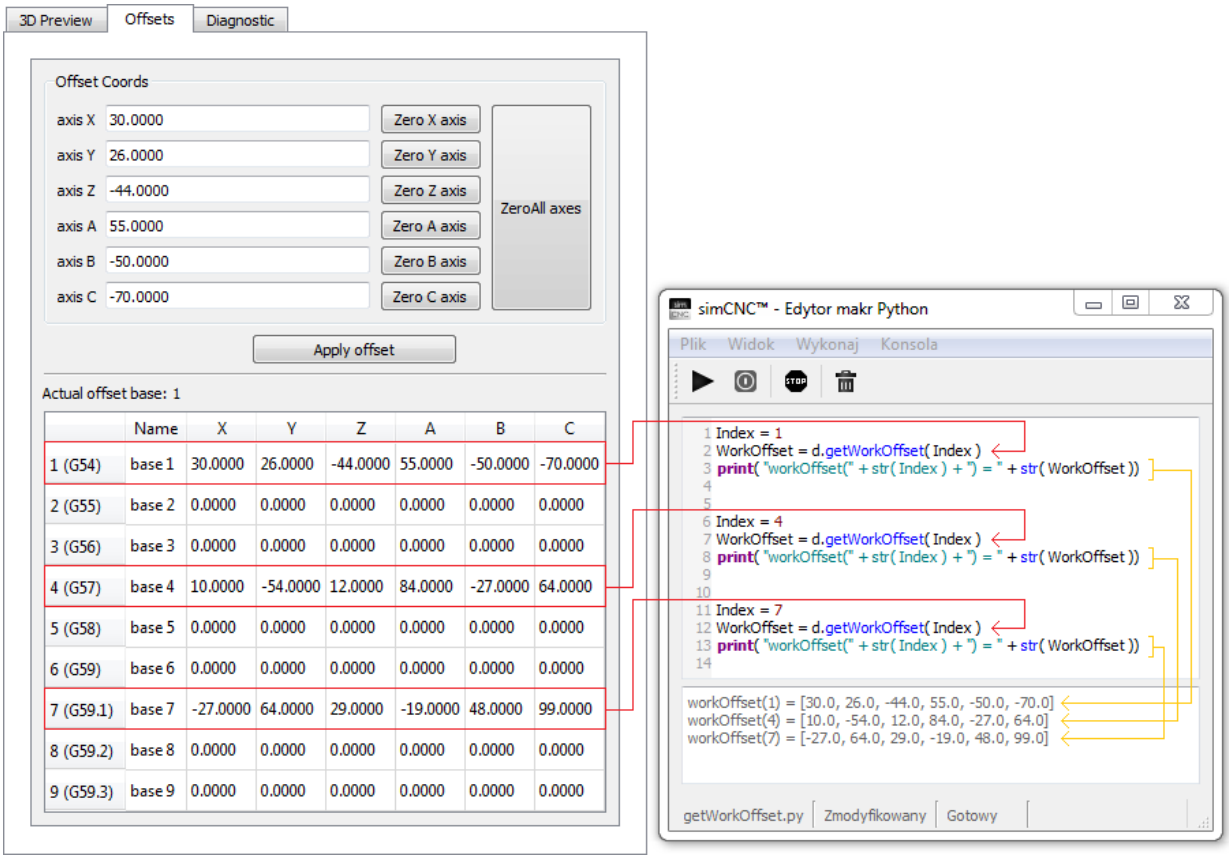
FUNCTION RESULT:

List<float>[6] - a list of the six coordinates (X, Y, Z, A, B, C) of the work offset

EXAMPLE:

```
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 ))
```

After running the example presented above, the coordinates of work offsets No. 1 (G54), 4 (G57), and 7 (G59.1) will be read and shown in the Python console.



ATTENTION!

simCNC can currently save 9 work offsets (9 material databases).
So the work offsets can be indexed from 1 (G54) to 9 (G59.3).
For clarity: 1 = G54 | 2 = G55 | 3 = G56 | 4 = G57 | 5 = G58 | 6 = G59 | 7 = G59.1 | 8 = G59.2 | 9 = G59.3

int getWorkOffsetNumber( ) - Returns the index of the current work offset.

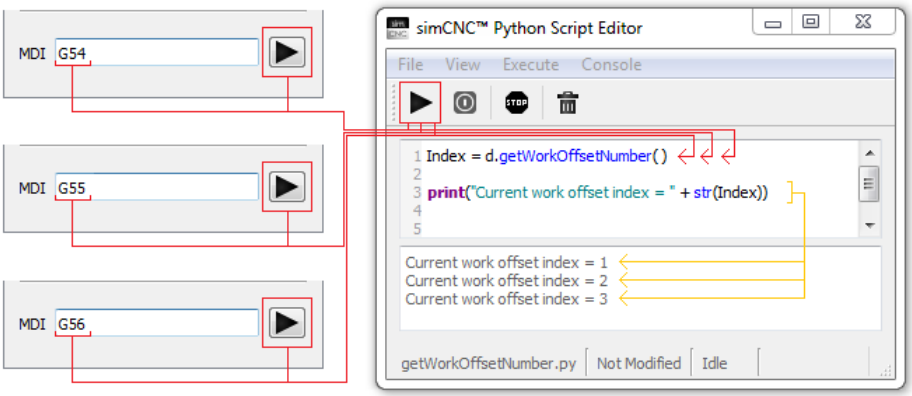
FUNCTION RESULT:

int – work offset index (number)

EXAMPLE:

```
Index = d.getWorkOffsetNumber( )

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

Before running the example described above, enter the G54 command in the MDI line and execute it by pressing the button on the right of the MDI line. Then run an example script. Once you do it, information on the index of the current work offset will appear in the Python console. Repeat the same procedures using the G55 and G56 commands to achieve the same result as in the figure below.



setWorkOffset(int workOffsetIndex, List<float>[6] workOffsetValues) – Modifies the coordinates of the work offset with a predefined index.

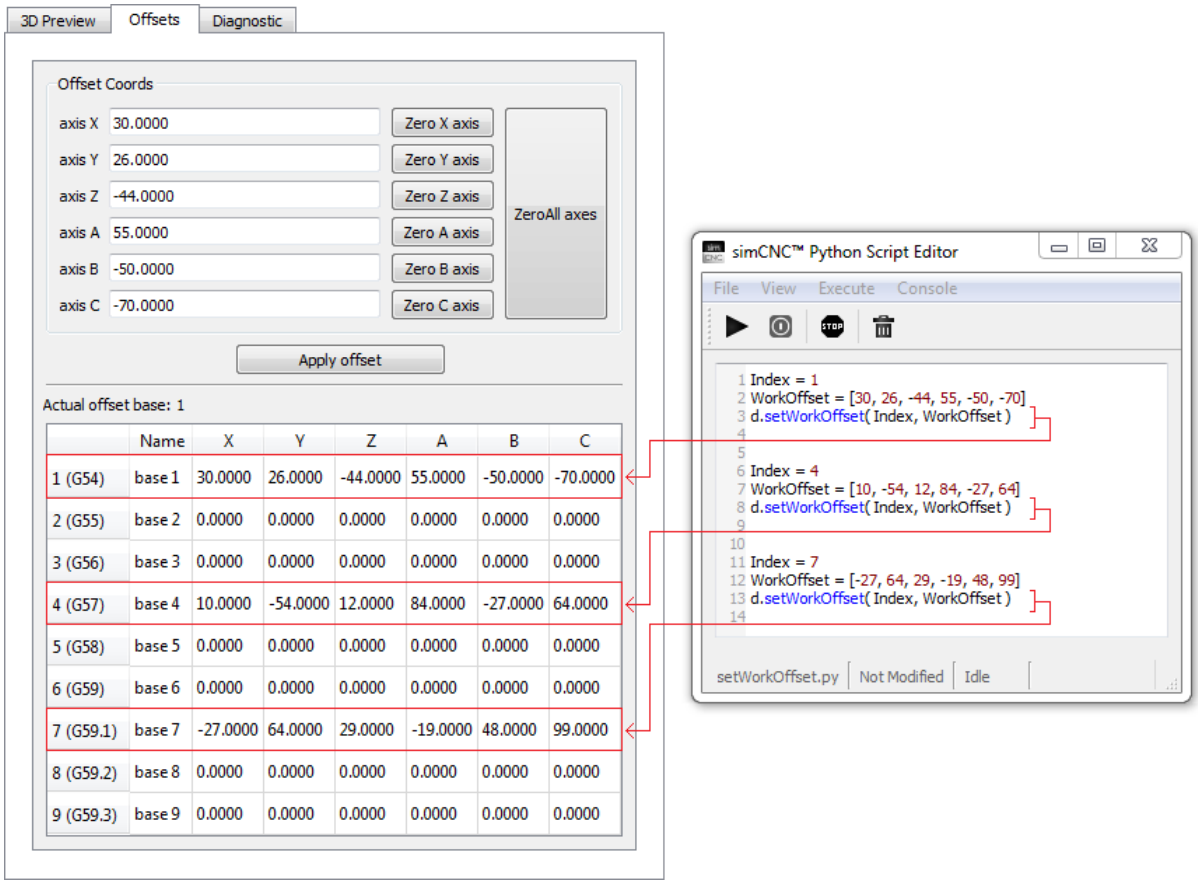
ARGUMENTS:

int workOffsetIndex - work offset index (number)
List<float>[6] workOffsetValues - six new coordinates (X, Y, Z, A, B, C) of the work offset

EXAMPLE:

```
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 )
```

After running the example presented above, the work offsets indexed as 1 (G54), 4 (57), and 7 (G59.1) will be set to new coordinates [30, 26, -44, 55, -50, -70], [10, -54, 12, 84, -27, 64] i [-27, 64, 29, -19, 48, 99].



setCurrentWorkOffset(List<float>[6] workOffsetValues) - Modifies the coordinates of the current work offset.

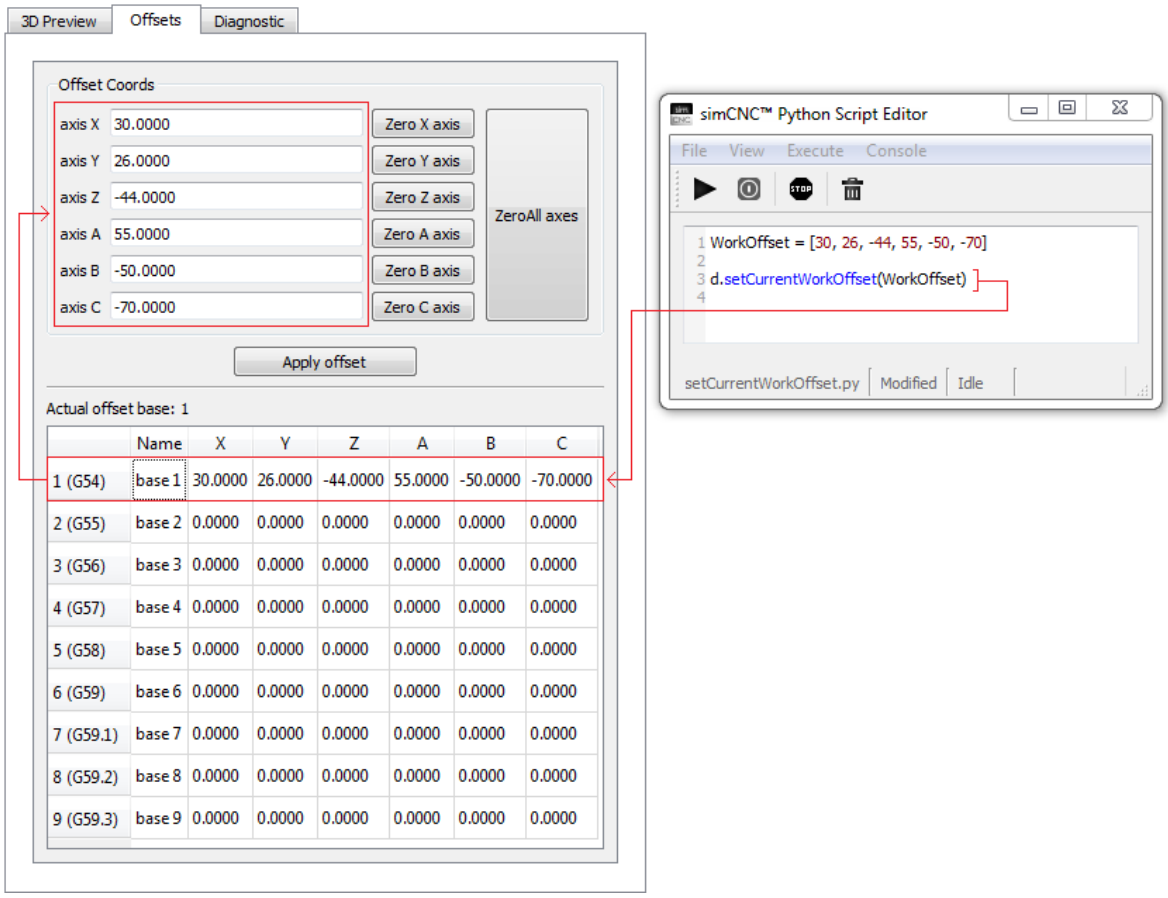
ARGUMENTS:

List<float>[6] workOffsetValues - six new coordinates values (X, Y, Z, A, B, C) of the current work offset

EXAMPLE:

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

d.setCurrentWorkOffset( WorkOffset )
```

After running the example presented above, the work offset indexed as 1 (G54) will be set to new coordinates [30, 26, -44, 55, -50, -70].



setWorkOffsetNumber(int workOffsetIndex) - Modifies the coordinates of the current work offset.

ARGUMENTS:

int workOffsetIndex – work offset index (number)

EXAMPLE:

```
Index = 4

d.setWorkOffsetNumber( Index )
```

After running the example presented above, the current work office will be switched to the work offset indexed as 4 (G57).