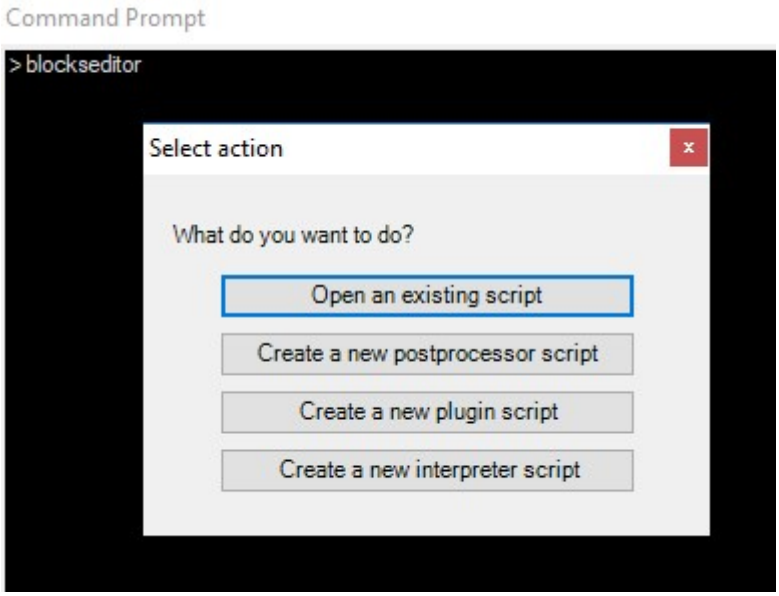
Blockscripts

Blockscript is a visual programming language (VPL) used to create plug-ins, postprocessors, and interpreters for the CNCSimulator Pro software. You will need a little bit of programming experience to be able to create these. Blockscript is inspired by the Scratch programming editor created by the MIT University.

Please note that the Blockscript editor, as well as this help information, are aimed at advanced users only. You will have to be experienced with the CNCSimulator Pro software, computers in general and have some level of programming experience. We will try our best to explain how Blockscripts works by using examples and pictures, but you will need to be prepared to experiment on your own, in able to learn the process of making virtual machines. We only offer basic and limited support on this part of the software.

To open the Blockscript editor, first go to the Tools menu and select “Cmd Prompt”.

At the Command Prompt line, type blockseditor and press Enter.



You will be presented with a menu where you can select if you want to open an existing script or create a new one. There are three different type of scripts you can create: postprocessors, plugins, and interpreters.

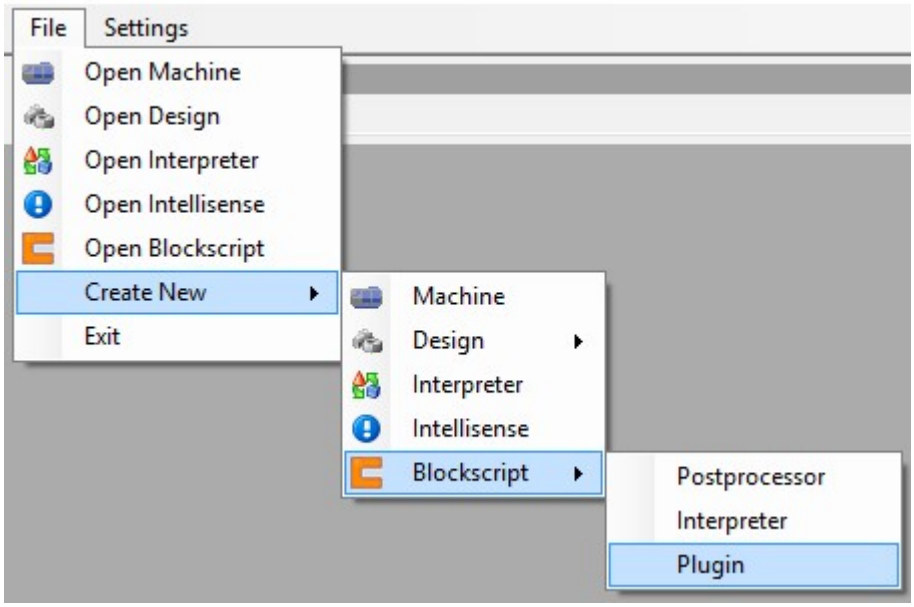
In the Blockscript editor, the user creates programs by dragging and connecting graphical pieces of logical blocks onto a "canvas".

© CNCSimulator.com 2018

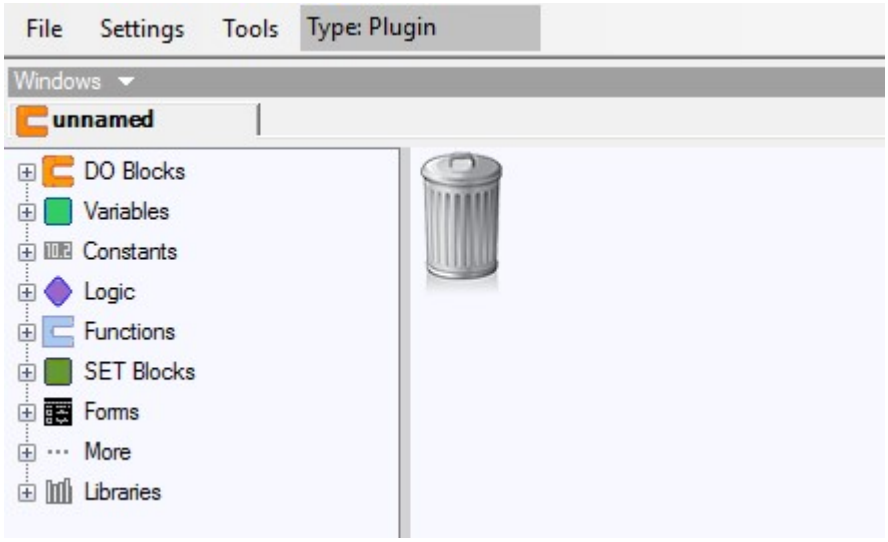
The "Hello World" example

All programming manuals should start with a Hello World example, so let us make one.

Start by creating a new Blockscript document for plug-ins.



You will see a blank canvas with a trash bin on it, as well as a tree view with blocks to the left.



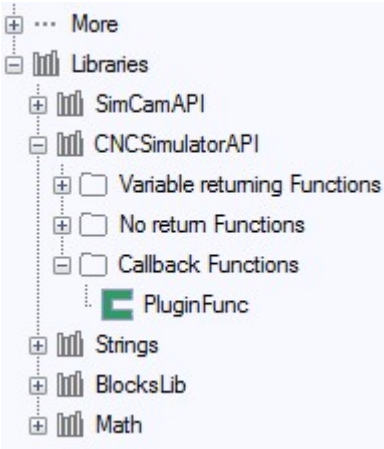
You can drag blocks from the tree on the left to the canvas. When you want to delete blocks, just drag them to the trash bin.

When the simulator executes a plug-in, it looks for a function called PluginFunc in the CNCSimulatorAPI library.

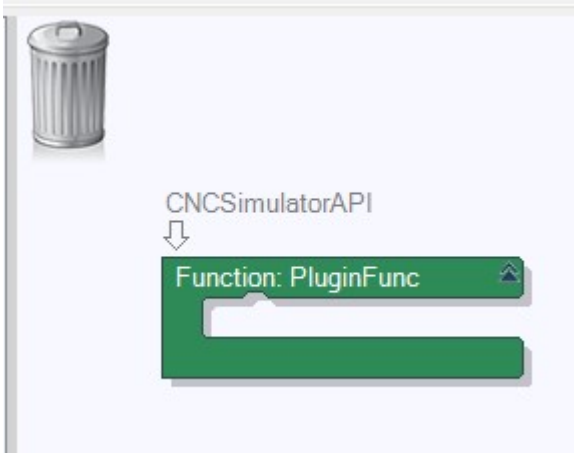
© CNCSimulator.com 2018

To see the function libraries, click on the plus symbol to the left of the Libraries tree item.

You can see that there are several libraries there, we will talk more about these later on. For now, just open (click on the plus) the CNCSimulatorAPI node and then the Callback Functions node.



There you will find the PluginFunc function block. Drag it to the canvas.

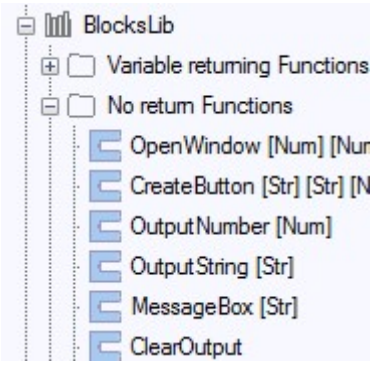


There are some labels and symbols on the block. Here is an explanation:

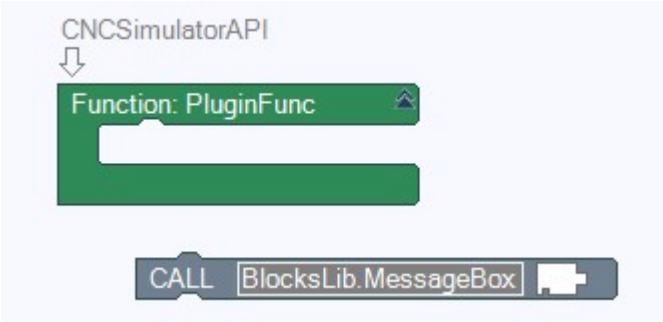
© CNCSimulator.com 2018

As for now, the function is empty, and nothing will happen if we call it from the simulator. Let us put a "Hello World!" message box inside it.

Open the BlocksLib node in the tree and then the No return Functions node.



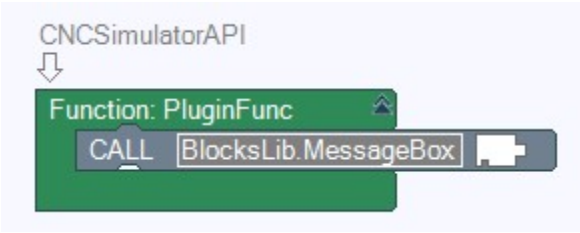
Drag the MessageBox item onto the canvas.



Note that this block looks different. It cannot engulf other blocks like the function block, and it has a strange looking hole to the right. Let us explain what this means.

First, at the top of the block is a small "bump". It is a male connector. It can only connect to female connectors with the same shape. As you can see, the function has a female one with the same shape.

Drag the MessageBox block inside the function block. When the connector lights up, you can release the block, and they will magnetically connect to each other.



You can also disconnect blocks by dragging them apart.

You always drag a male connector to a female one to connect. It does not work the other way around.

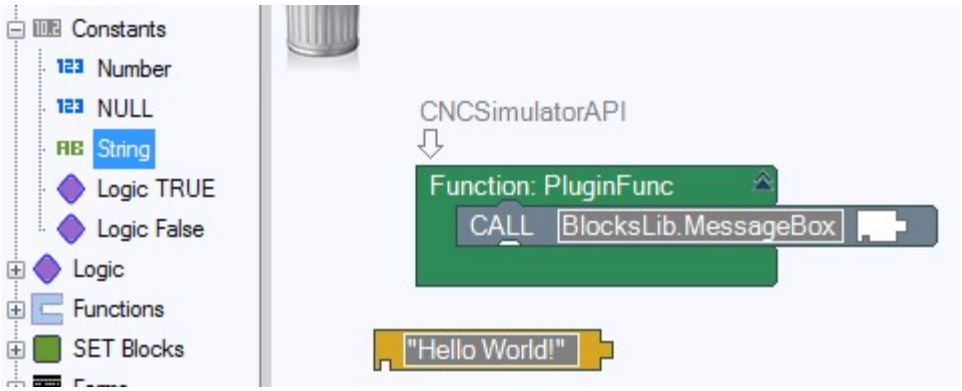
© CNCSimulator.com 2018

The hole in the MessageBox block is also a connector. It has the shape of text strings. This means that for example a number, that has another shape, will not fit into the hole. This is a way for Blockscript to ensure that each function gets the correct data.

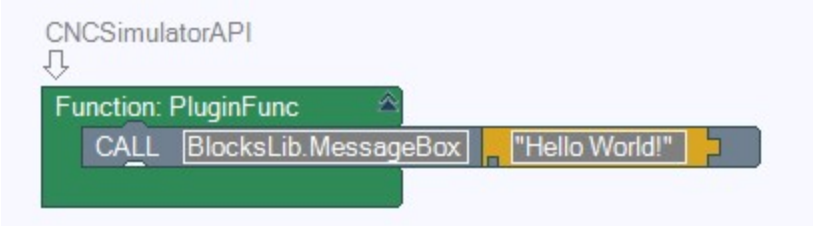
Let us create a text string with the text "Hello World!".

In this case, we want a "constant" text string, as we do not intend to change its value. Open the Constants node in the tree and drag a string onto the canvas.

Type in "Hello World!", when asked to define the constant.



If you look at the shape of the string, you can see that even though it is wider, the shape is similar to the hole in the MessageBox block. Drag the string to the hole so that it connects.



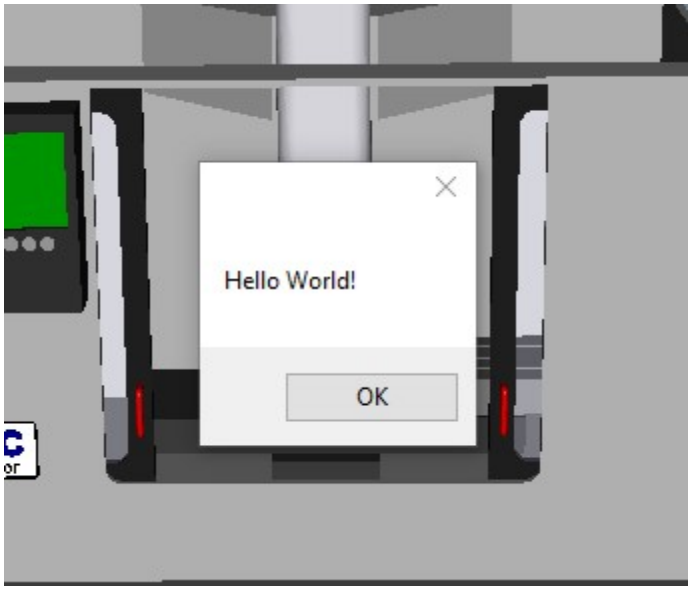
Our example plug-in is done!

What this Blockscript basically says is: When the user clicks this plug-in from the simulator, execute the PluginFunc that calls the MessageBox function of the CNCSimulatorAPI library with the constant string "Hello World!".

Click Save As in the menu and save it to the simulators plug-in folder. Fill out the information about the document.

In the simulator, open the Tools menu, and in the Plugins sub menu, click on Reload Plugins. Now you should see a Hello World item (depends on how you named it) in the menu. Click on it.

© CNCSimulator.com 2018



OK, that was easy. Let us try a little bit more advanced example.

© CNCSimulator.com 2018

The Unit converter example

In this example, we are going to create a little converter that can convert numbers from inches to millimeters and vice versa.

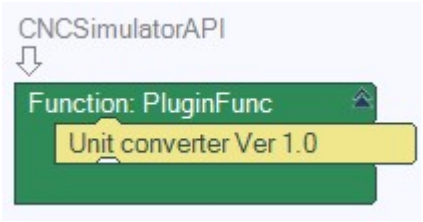
Just as in the Hello World example, create a new Blockscript document and drag the PluginFunc block to the canvas.

Under the More tree item, find the comment block and drag it to the canvas.

It is a good practice to add comments to any program. These will help you remember things when doing changes to the program later.

Enter "Unit converter Ver 1.0" into the textbox shown.

Drag and dock the comment block into the function.

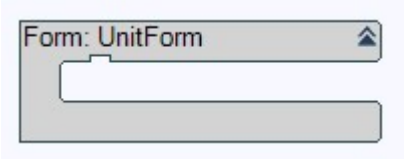


We are now going to create a Form. This is a popup window that you can use to get information from the user when the plug-in is running.

Find the Forms item in the tree view and open it. Drag the Create Form item onto the canvas.

© CNCSimulator.com 2018

Enter "UnitForm" in the text box.

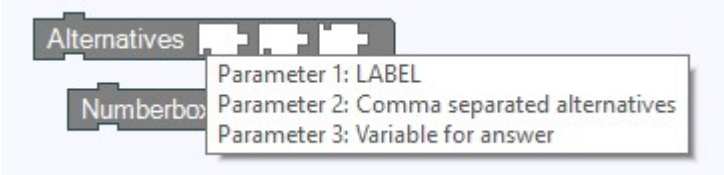


In the tree view, drag a Form Item Alternatives block to the canvas.

Also, drag a Form Item Numberbox.

We can see that the Alternatives block takes three arguments and that the Numberbox takes two.

If you hover the mouse over a block, you will see some help information about it.



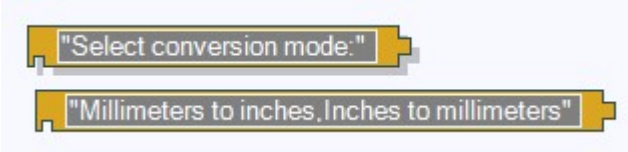
Here we can see that the Alternatives block expects a Label, some comma separated alternatives, and a variable for the answer.

The shape of the holes tells us that these are strings and that the two first ones have to be readable, and the last one has to be writeable.

We will talk more about these shapes later, but for now, just know that these shapes prevents us from putting a read-only variable where a writeable variable is expected, and also to put a number where a string is expected and so on.

© CNCSimulator.com 2018

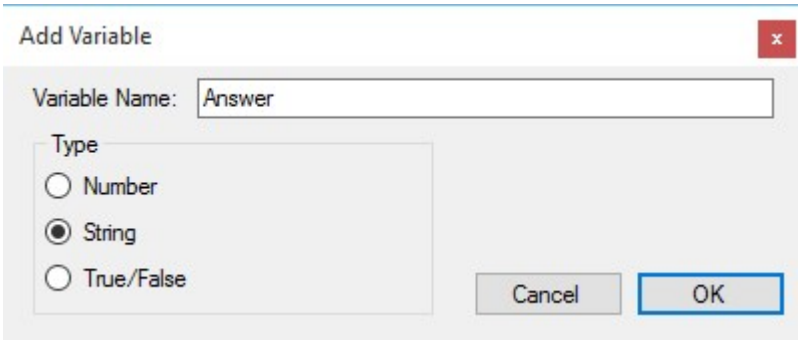
We now need two string constants, one for the label of the Alternatives Form Item (a combo box) and one for the alternatives. Under the Constants item in the tree, drag two strings onto the canvas. Put the text "Select conversion mode:" in the first one and " Millimeters to inches,Inches to millimeters" in the second one. This one is the alternatives to choose from.

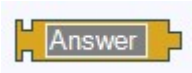


We can see that these are read-only (you cannot write to constants) by comparing the shape to the holes above.

Let us now create a variable for the answer.

Under the Variables item in the tree, drag Create New to the canvas and type in Answer in the name box. Also select String under Type and click OK.



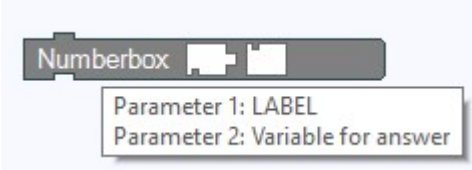


The variable has a shape that lets us know it is both readable and writeable.

Now, drag and dock these three blocks into the Alternatives block.

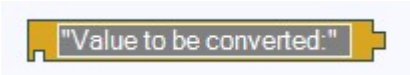
© CNCSimulator.com 2018

For the Numberbox that we added earlier, we also need to create a constant and a variable.



As you can see, the second hole is flat on the right side. It means it expects a number.

The first hole is a read-only string connector. Let us create a Constant string as we did earlier and give it the text " Value to be converted:".

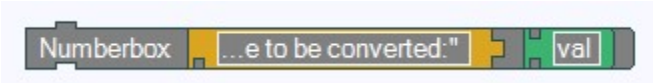


If you want to change the text of a constant, just double-click it and type in a new string.

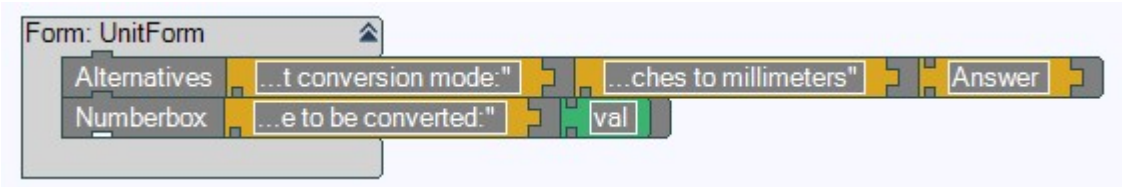
Now, create a number variable and name it "val". Look above on how to create new variables.

The colors of blocks, also tells us something about their type.

Drag and dock these two items into the Numberbox item.



Connect the Numberbox item to the Alternatives item and connect them both inside the form.



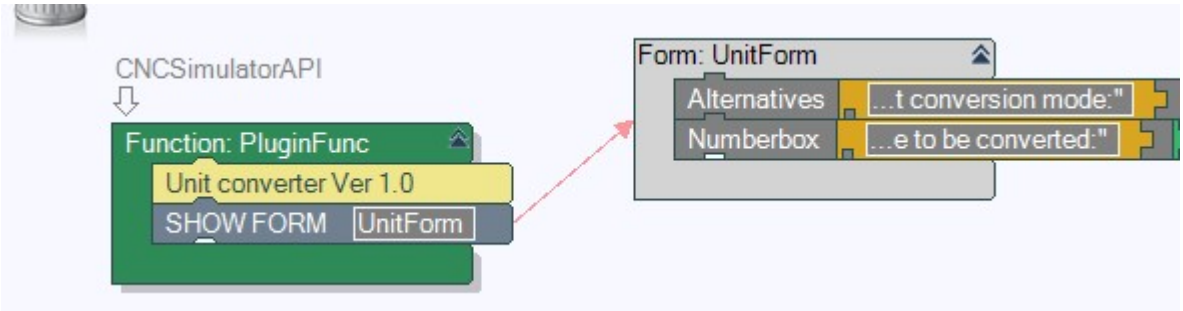
Great! Now we need to call the form from inside our plug-in function.


Click on the refresh button under the tree view.

Under the Forms item in the tree, open Show a form, and there you will find the form we just created. Drag it onto the canvas.

© CNCSimulator.com 2018

Drag and dock the Show Form block to the comment inside the function.



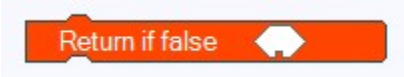
The arrow makes it easier to see what block is calling up what form.

We need to end the plug-in if the user clicks Cancel in the form.

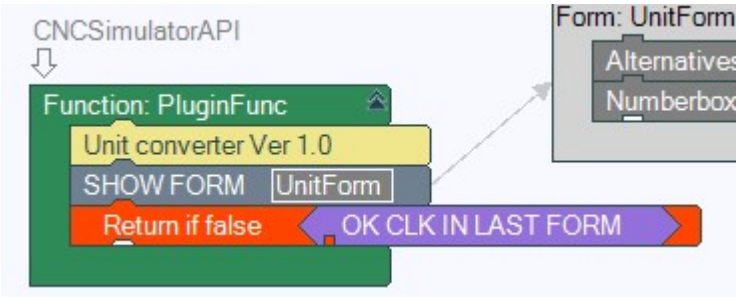
To add this functionality, drag a "User Clicked OK in the last Form" block onto the canvas. You find it under the Forms tree item.

From the shape and color of this variable, we get that it is a logic true/false block. It can only have two values, true or false.

Under the Functions tree item, drag a "Return if FALSE" block onto the canvas.



The shape of the hole tells us that it expects a readable logic variable. Now drag the "OK CLK IN LAST FORM" variable and dock it in this one. Then dock the result inside the function.

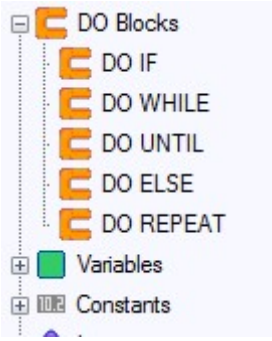


OK, now the function will return (end) if the user did not click OK on the form (meaning he clicked Cancel).

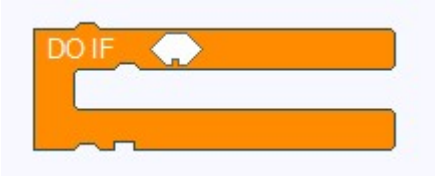
© CNCSimulator.com 2018

Now we have to split the execution path depending on the answer of the user. We should either convert from millimeters to inches or vice versa.

For this, we use the so-called conditional DO-blocks.



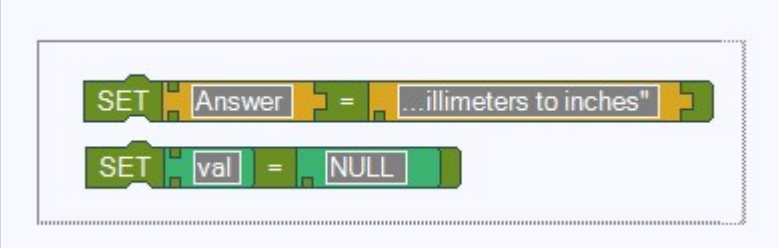
Drag a DO IF block onto the canvas.



As you can see, we have a logical true/false block connector at the top. The blocks inside the DO IF-block will only execute if this variable is TRUE.

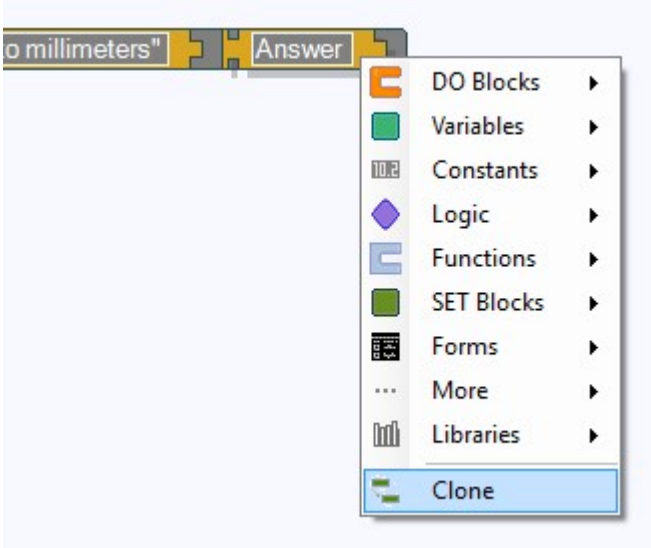
We are going to test the answer from the Alternatives block in the form.

Quick Tip! You can select more than one block by clicking the canvas and dragging a rectangle selector around them. This way you can move several blocks at once.

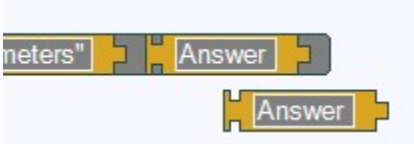


© CNCSimulator.com 2018

Right click on the Answer variable in the form and select Clone from the menu.



A clone of the block will be created. As the block is a clone, it will always have the same value as the original block.



Now, create a string constant with the text "Millimeters to inches". Make sure the spelling is exactly as the alternative in the Alternatives block. Otherwise it will not work.

We now need to compare Answer and the constant to see if they are the same. Under Logic, Compare strings, drag a [Str1] = [Str2] block onto the canvas.

This block compares two strings and becomes TRUE if they are exactly the same.

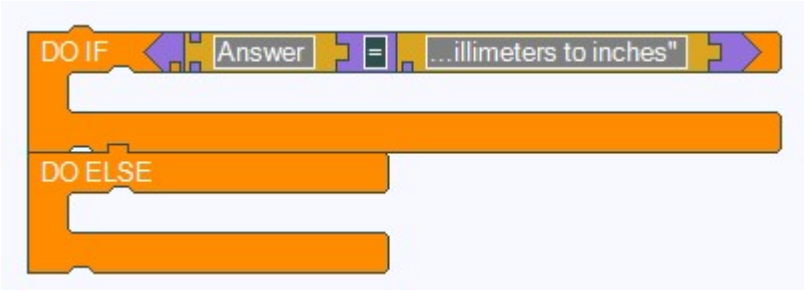
Connect Answer and the string constant we created with this logic block. Then drag it to the DO IF block.

In plain English, this block says, "DO the blocks inside my belly if Answer is equal to "Millimeters to inches".

© CNCSimulator.com 2018

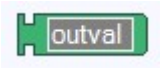
We also need to do something else if the Answer is NOT same as the constant (meaning the user has selected the other option).

From the DO blocks, drag a DO ELSE block onto the canvas and connect it under the DO IF block.



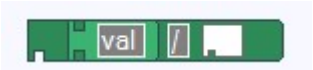
Now it is time to add the blocks that should be executed in the DO IF and DO ELSE blocks.

First we create a number variable called, "outval".

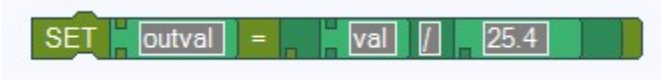


Then under SET Blocks, drag a Set Numeric Variable block onto the canvas and dock the variable.

From Variables, drag a Divide numbers block onto the canvas and dock a cloned "val" variable at the first connector.



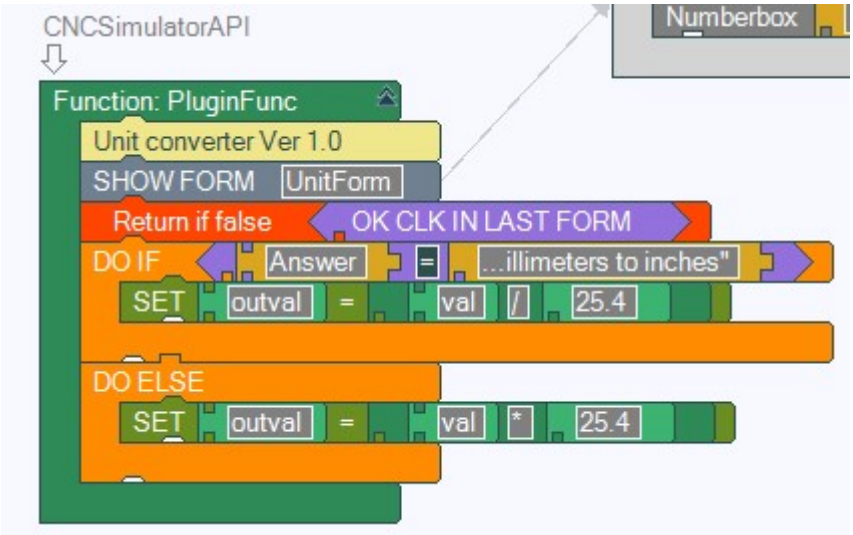
Create a number constant with the value 25.4 and dock it at the last connector. Dock the Divide numbers block to the set block.



This is our calculation to convert val (the number typed in by the user) to inches. Drag it to the DO IF block.

Repeat and do almost the same thing for the DO ELSE block. This time use a Multiply numbers block.

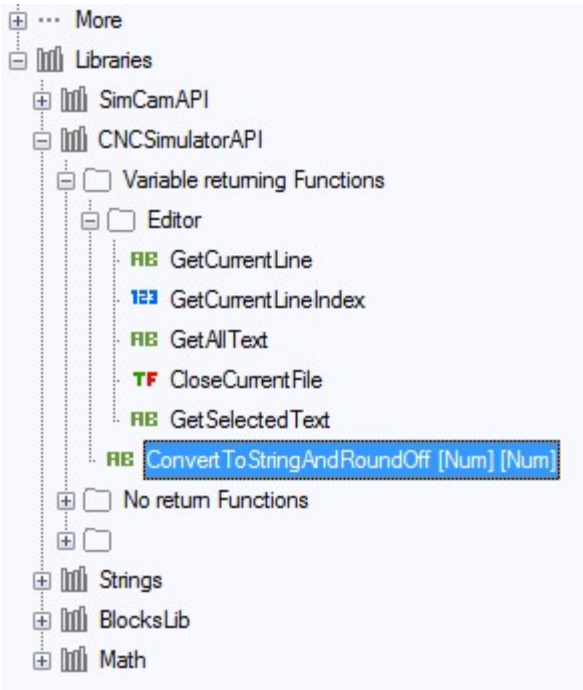
© CNCSimulator.com 2018



Our function is getting quite colorful, don't you think?

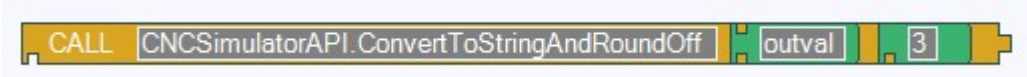
As a final step, we are going to round off the resulting value to three decimals, and send it to the CNCSimulator Pro editor.

Drag the round-off block in the CNCSimulatorAPI library to the canvas.

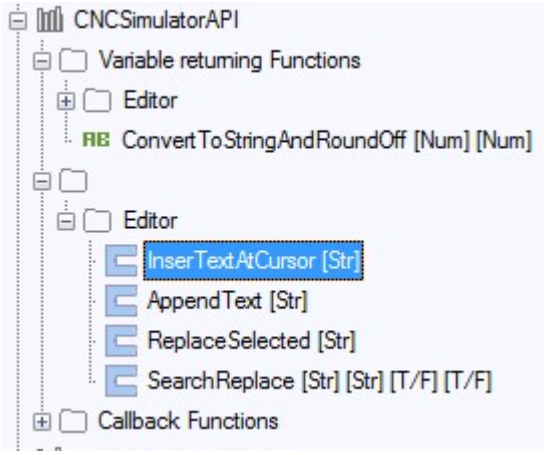


© CNCSimulator.com 2018

Dock a clone of outval and a numeric constant of value 3 into it.



From the CNCSimulatorAPI library, also drag a InsertTextAtCursor block to the canvas.

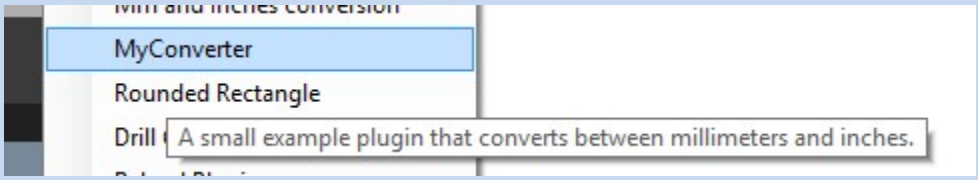


Dock the other call into it.

Dock it at the end of the function.

Time to test the plug-in! Save it and give it a unique name. There is probably already a converter plug-in, so let us name it "MyConverter". Save it with this name.

Quick Tip! When you save a plug-in, you will see a dialog window. The Name field will be the text you see in the simulator Plugin menu. If you type in text in the Info field, it will show up as a tooltip when you hover the mouse over the plug-in menu item in the simulator.

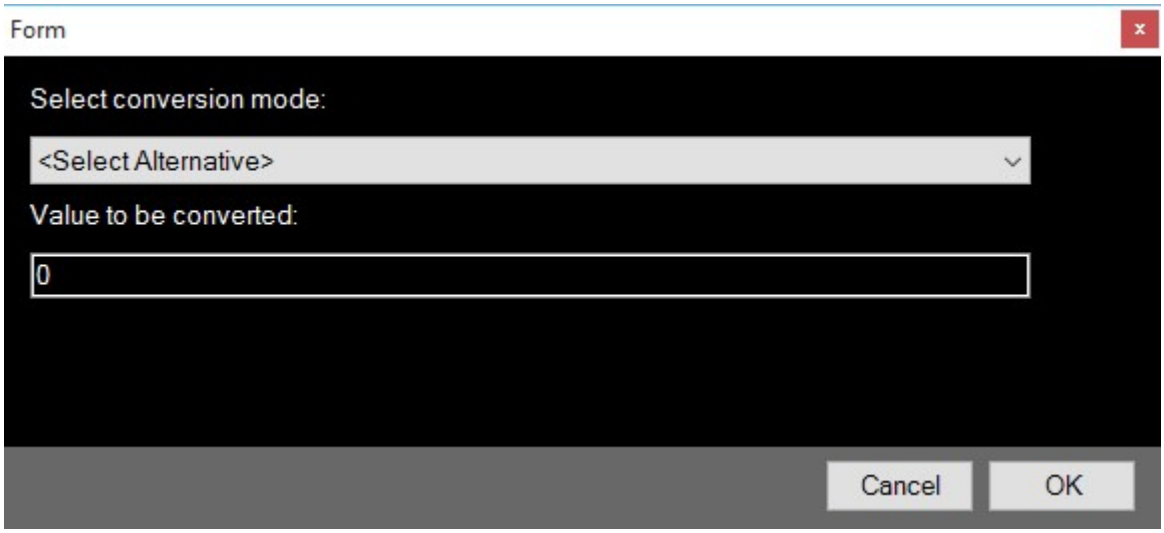


Switch to the simulator and click "Reload plugins" in the Plugins menu.

Click on MyConverter.

© CNCSimulator.com 2018

You should see a form looking like this.

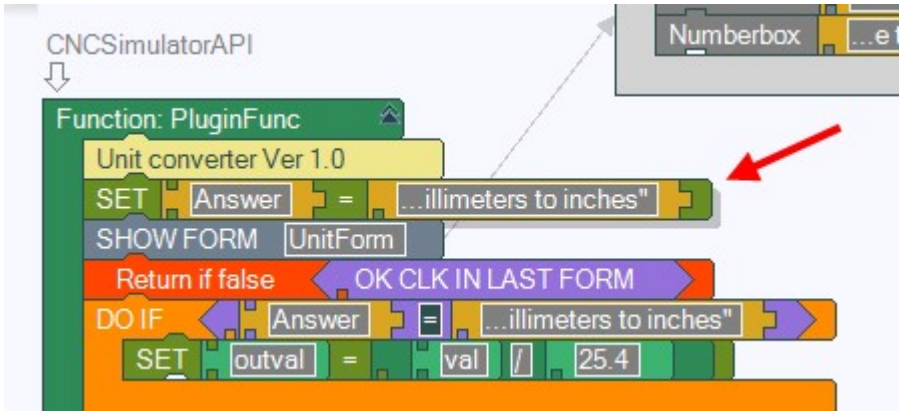


Select a conversion mode and type in a value.

At the cursor in the editor, you should see:

```
3.937
```

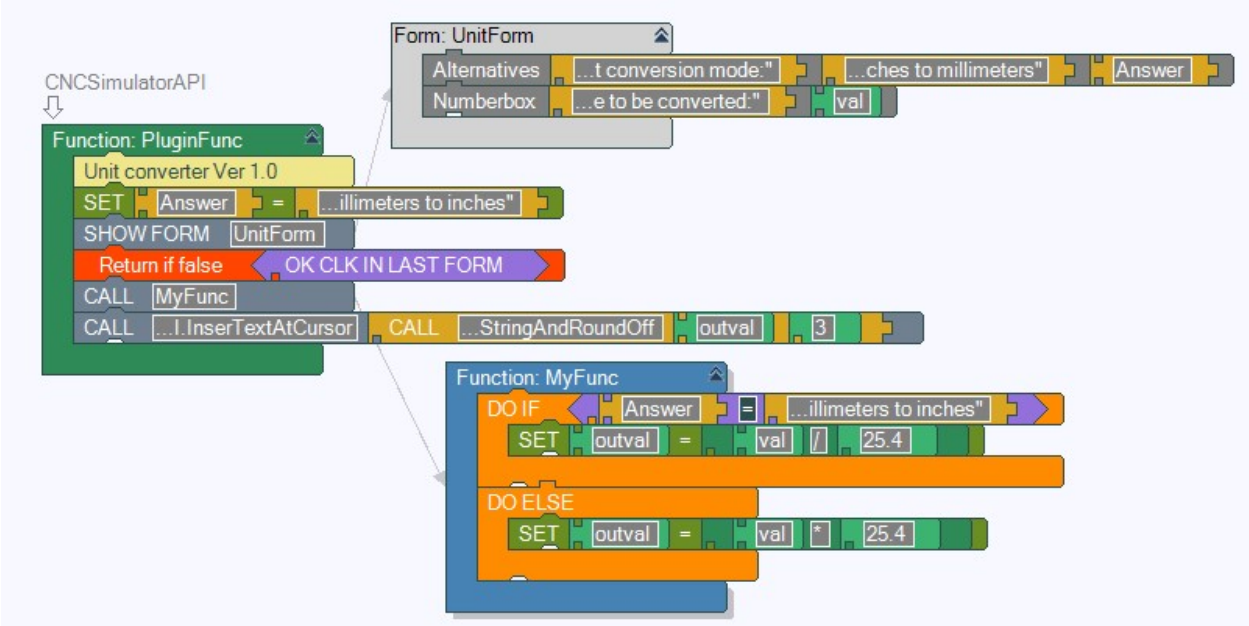
If you want one of the alternatives to be preselected in the Alternative box, just set the value to the alternative you want before opening the form.



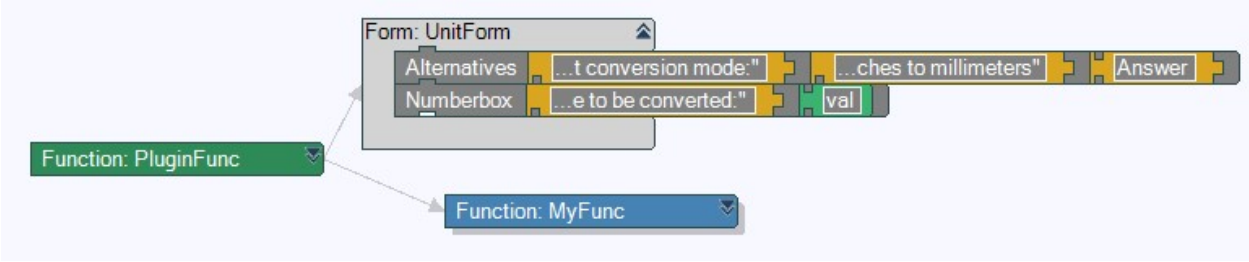
© CNCSimulator.com 2018

To avoid that a function gets too big, you can always split it up into more functions. This is also handy to avoid repeating the same blocks in the program.

Here we have moved out the conditional blocks to a function called MyFunc.

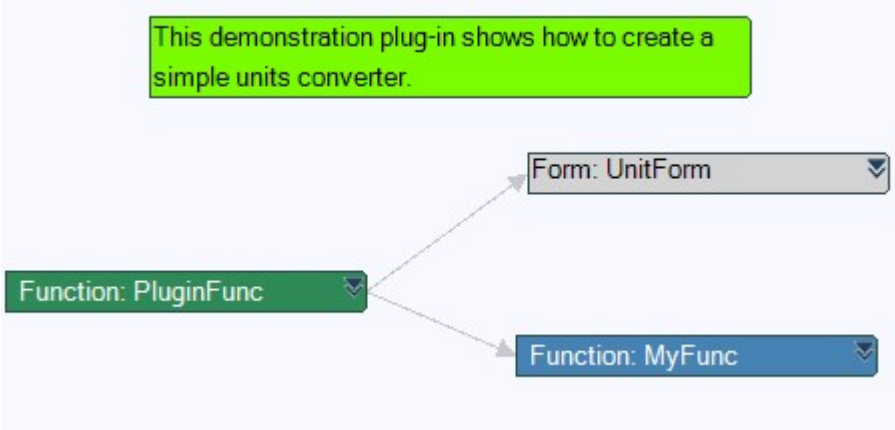


To avoid getting overwhelmed by clutter on the canvas, close all functions and forms that you are not currently working on. Use the small up-pointing arrows at the top right corner of the blocks to do that.

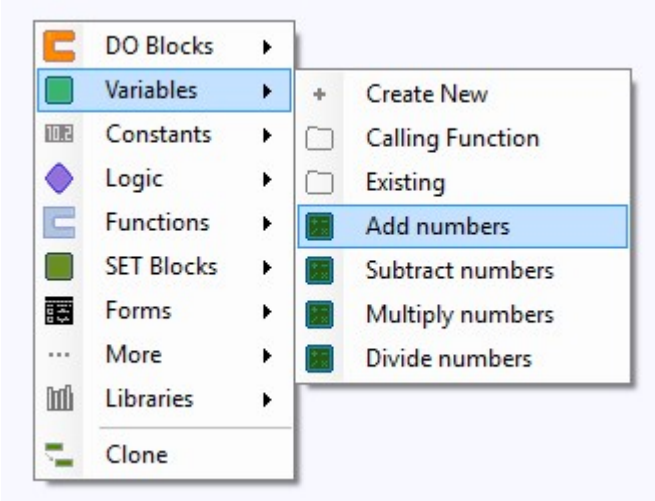


© CNCSimulator.com 2018

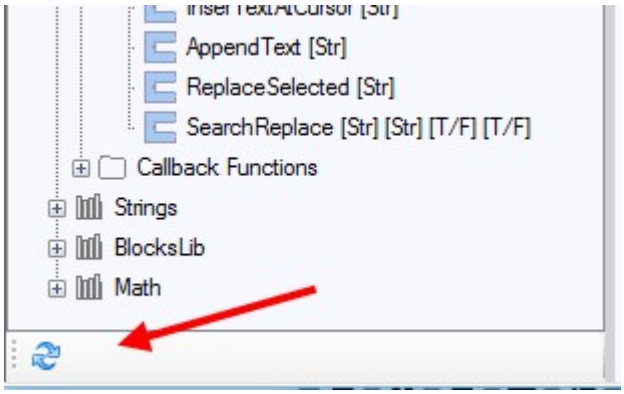
Add info-blocks as post-it-like reminders in your blockscripts.



There is an alternative way to adding blocks. By right-clicking the canvas, you get a popup-menu.



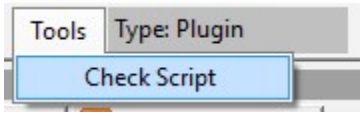
When you create forms, variables, and functions, they show up in the right-click menu automatically. To make them show up in the tree view, click on the refresh button at the bottom of the panel.



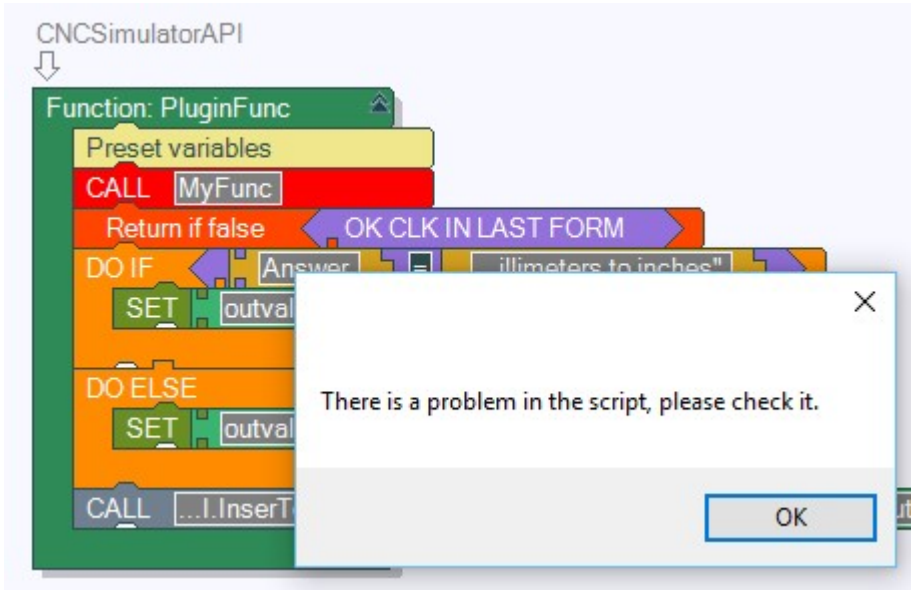
© CNCSimulator.com 2018

Testing scripts

You can test your scripts for errors by clicking Check Script in the Tools menu.



If there is a problem with the script, a message box will show and the block where the problem was found will show in red.



In this script, we are trying to call a function named "MyFunc" that does no longer exist.

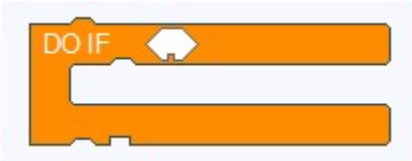
© CNCSimulator.com 2018

Different types of blocks

Here is a list of different block types and their function.

DO Blocks

DO IF



This block will execute the internal blocks if the Logic variable is True.

DO WHILE

This block will execute the internal blocks repeatedly, as long as the Logic variable is True.

DO UNTIL

This bloc will execute the internal blocks repeatedly, until the Logic variable becomes True.

DO REPEAT n TIMES

This bloc will repeat the internal blocks as many times as the number parameter says.

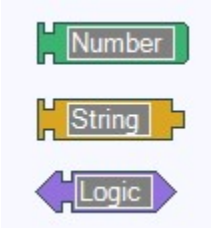
DO ELSE

This bloc can be connected to a DO IF block, and will be executed if the DO IF logic variable is False.

© CNCSimulator.com 2018

Variables

There are three types of variables in Blockscript. Numbers, Strings, and Logic variables.



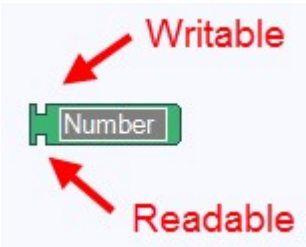
A number can be any number, decimal or integer, positive or negative.

A string is a piece of text.

A logic variable can be either True or False.

They have different shapes on the short sides, so they will fit only in the correct type connectors. This prevents you from putting a string in a connector where a number is expected, for example.

A variable can also be readable, writable, or both. This is visible by the small slots to the left.

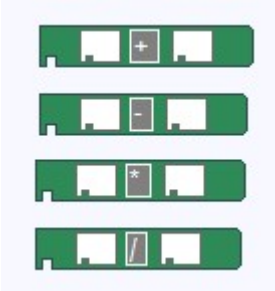


A connector has a corresponding pin to let you know what it expects.

As you can see, a DO REPEAT block expects a readable number. It does not matter if it is writable or not.

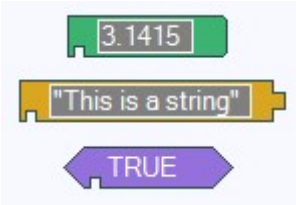
© CNCSimulator.com 2018

Under variables, you also find blocks that do basic math operations on numbers, like adding two of them together.



Constants

Constants are like variables, but they are read-only and will never change (during execution) once initialized.

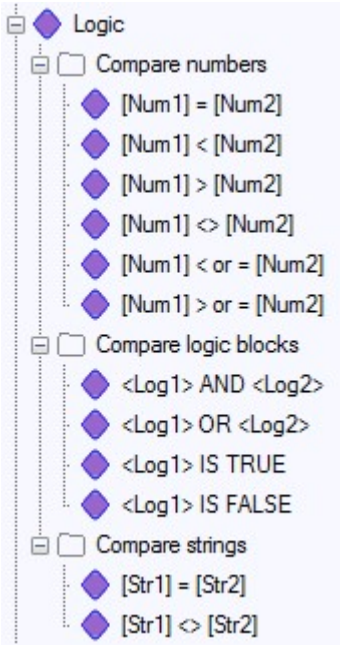


As you can see, they miss the upper right slot.

© CNCSimulator.com 2018

Logic blocks

Under Logic in the tree view, you find a different kind of logic operations.



Using these blocks, we can compare numbers or strings. Let's take a couple of examples.

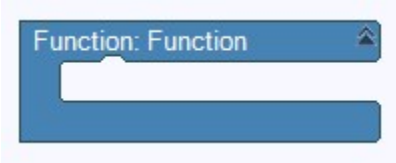
This block will compare two numbers and will be True if the first number is less than the second one.

This block will compare two strings (look at the shape of the holes) and will be True if they are exactly the same.

© CNCSimulator.com 2018

Functions

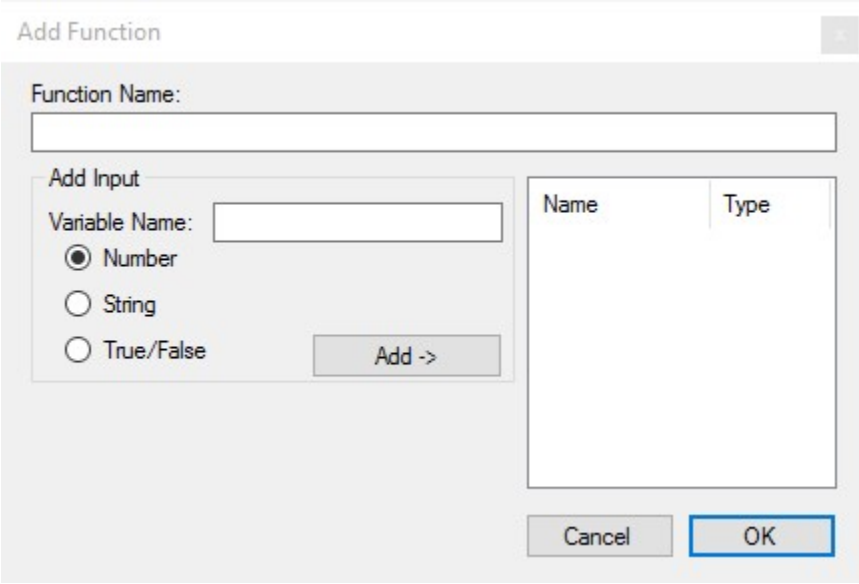
Functions are groups of blocks that can be called as one command.



They are normally used to create a good structure and avoid repeating blocks.

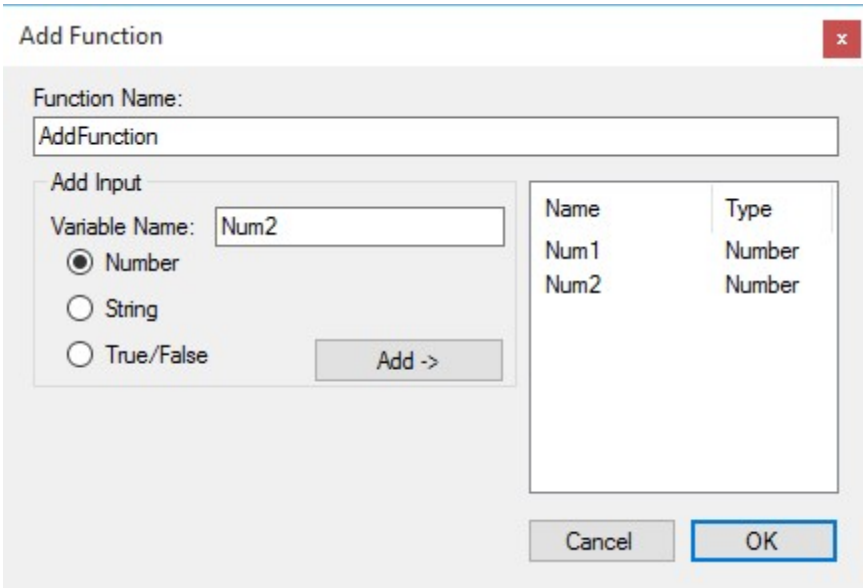
The most basic function, like the one above, takes no parameters.

When you drag the "Create Function" item to the canvas, a dialog box will show to help you create the type of function you want.



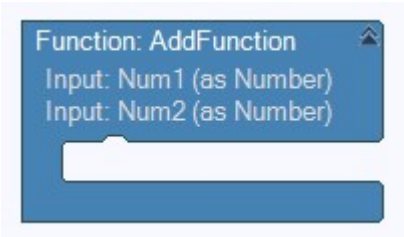
You give it a name (without spaces), and you can then add input variables. Let us create a function that adds two numbers together.

© CNCSimulator.com 2018



Here we have added two input numbers.

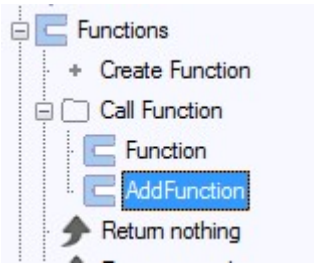
This is how the function will look.



Now, in the tree view, we click on the refresh button to make the new function show up.

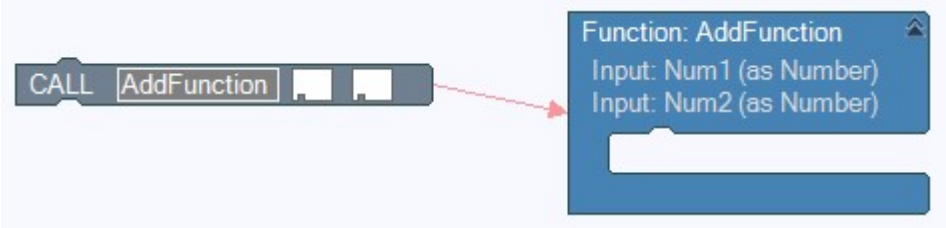
We need to add a block that calls our new function from our main function.

Under "Call Function", drag the new function to the canvas.

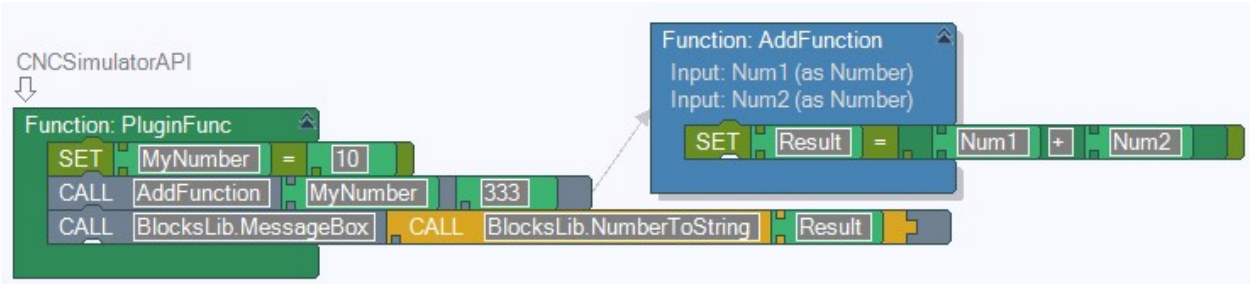


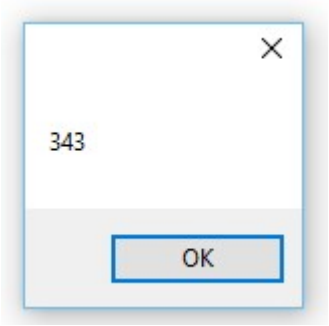
As you can see from the picture, the caller has our two input numbers.

© CNCSimulator.com 2018



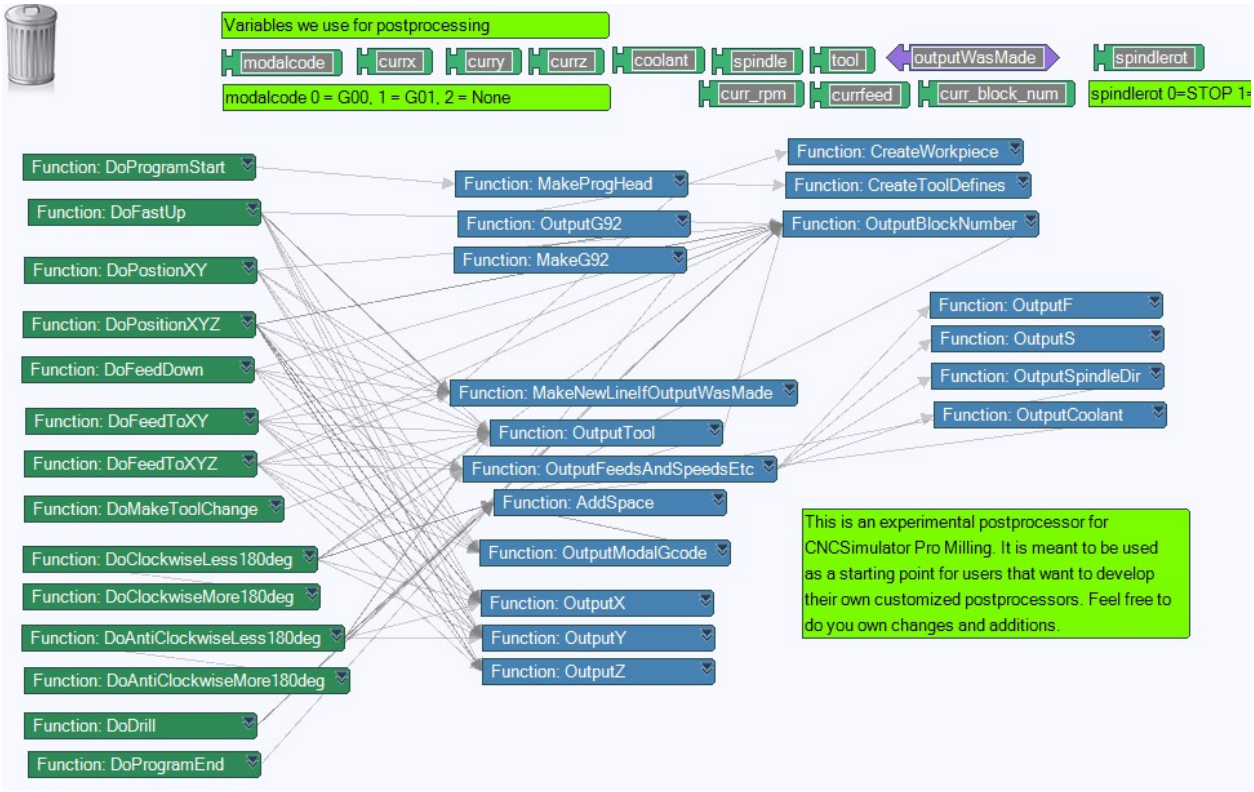
Now we add some blocks to add the numbers together, and we call the function from the main function. Finally, we display the result in a message box.





© CNCSimulator.com 2018

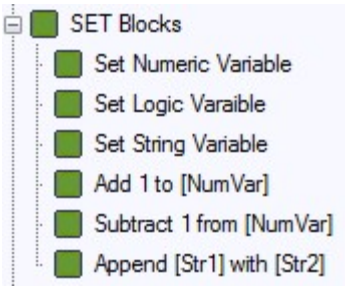
These are all very basic examples. As you can probably guess, Blockscripts can grow pretty big for complex things like postprocessors and interpreters. Fortunately, we can open and close functions and move them around as we like to keep things neat.



© CNCSimulator.com 2018

Set blocks

Set blocks are used to give or change values of variables.



Here are a couple of examples:

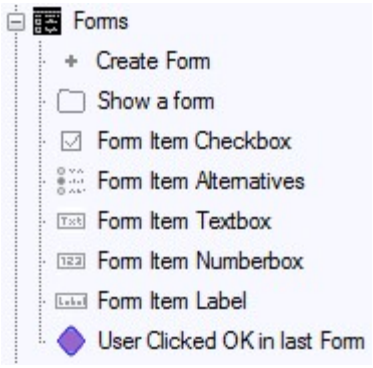
This Set block will append one string with another. If the first string is "Hello" and the other one is "World!", the first string will be "Hello World!" after the block has executed.

This Set block will increment the number variable by one.

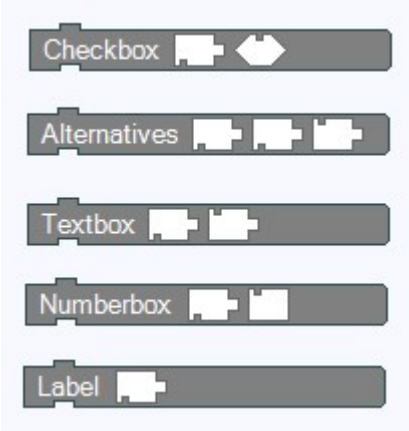
© CNCSimulator.com 2018

Forms

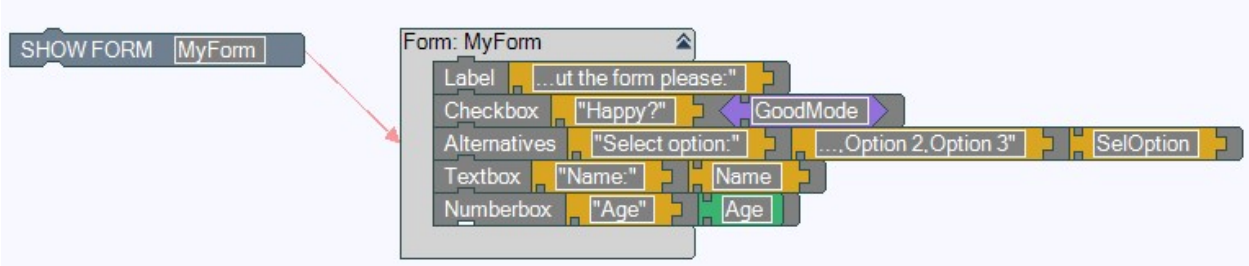
By using forms, you can get input data from the user of the script.



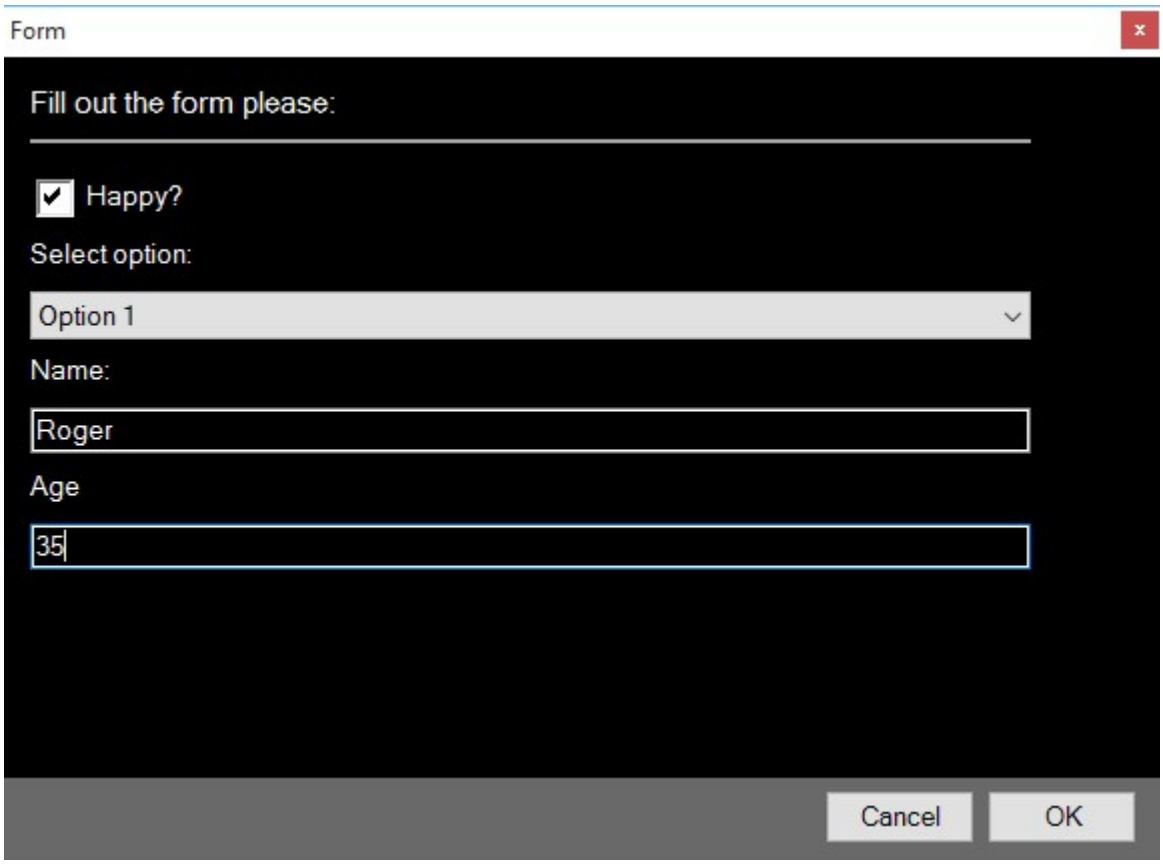
A form can contain different kind of items to collect data from the user. These are check boxes, Alternatives, Text boxes, Number boxes, and Labels.



Here we have created a demonstration form that shows each form item type.



© CNCSimulator.com 2018

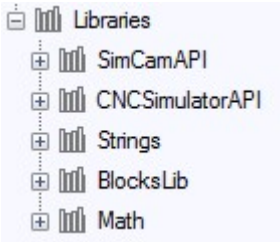


See our previous examples, for more on how to use forms.

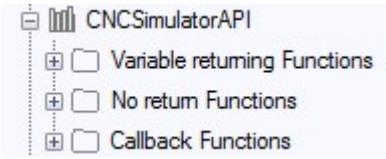
© CNCSimulator.com 2018

Libraries

Blockscript comes with various libraries that can be used to perform functions in SimCam or in the simulator. There is also a separate library with math functions.



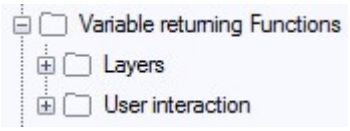
Each library is divided into groups, depending on the function type.



A function can return a variable, be no-return, and be a "callback function".

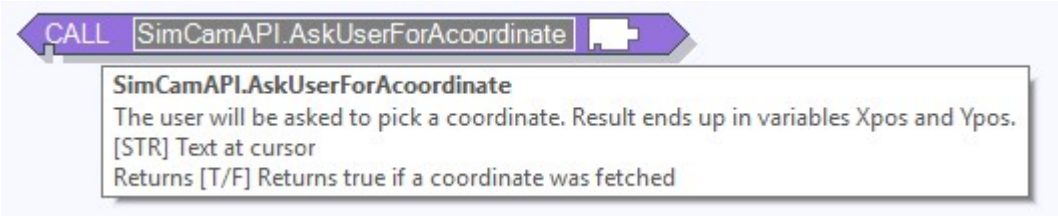
A callback function is a function that is called from the simulator, or from SimCam. Other functions are called from the blockscript to the simulator or SimCam.

These groups can be further divided into sub-groups, to help you navigate through the many function blocks in the libraries.



Let us have a look at a typical function from the SimCamAPI.

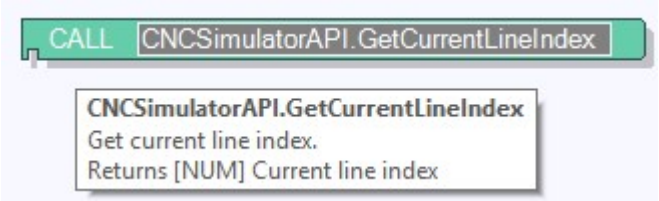
By holding the mouse over the function, a help tooltip will show, explaining the basics.



This function call will ask the user to pick a coordinate in SimCam showing the text string at the cursor. If a coordinate was picked, it will be True.

© CNCSimulator.com 2018

Here is another example:



This function in the CNCSimulatorAPI gets the index of the current line in the editor.

For a complete list of functions in a library, right-click it and select "Show Documentation".

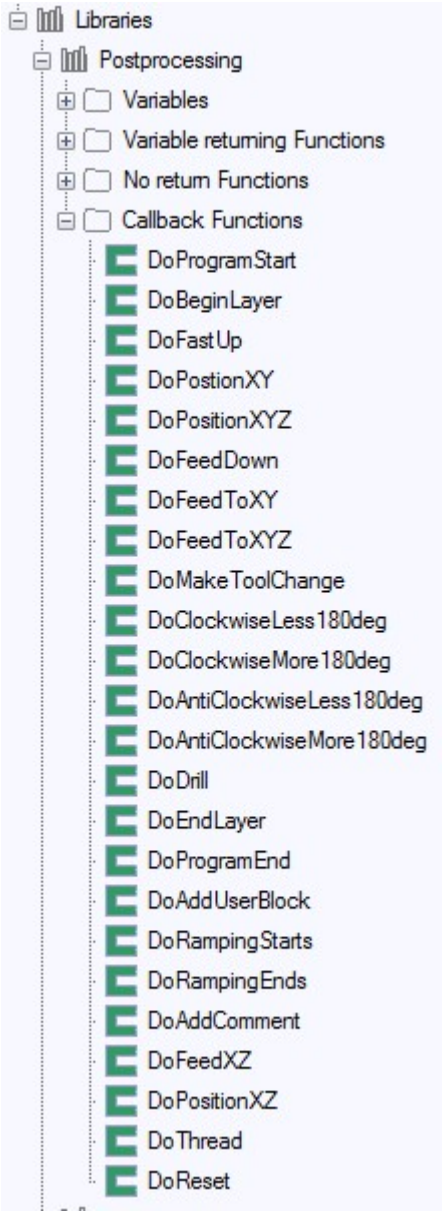
© CNCSimulator.com 2018

Creating Postprocessor scripts

Postprocessors create CNC-machine compatible code from SimCam documents. The geometrical (and tool data, machining data, etc.) information in the document, needs to be translated into a format that can be understood by a particular CNC machine controller. The default postprocessors built-in to SimCam creates a dialect of CNC code that can be interpreted correctly by the simulator.

To create a new postprocessor script, click Create New - Blockscript - Postprocessor, in the menu.

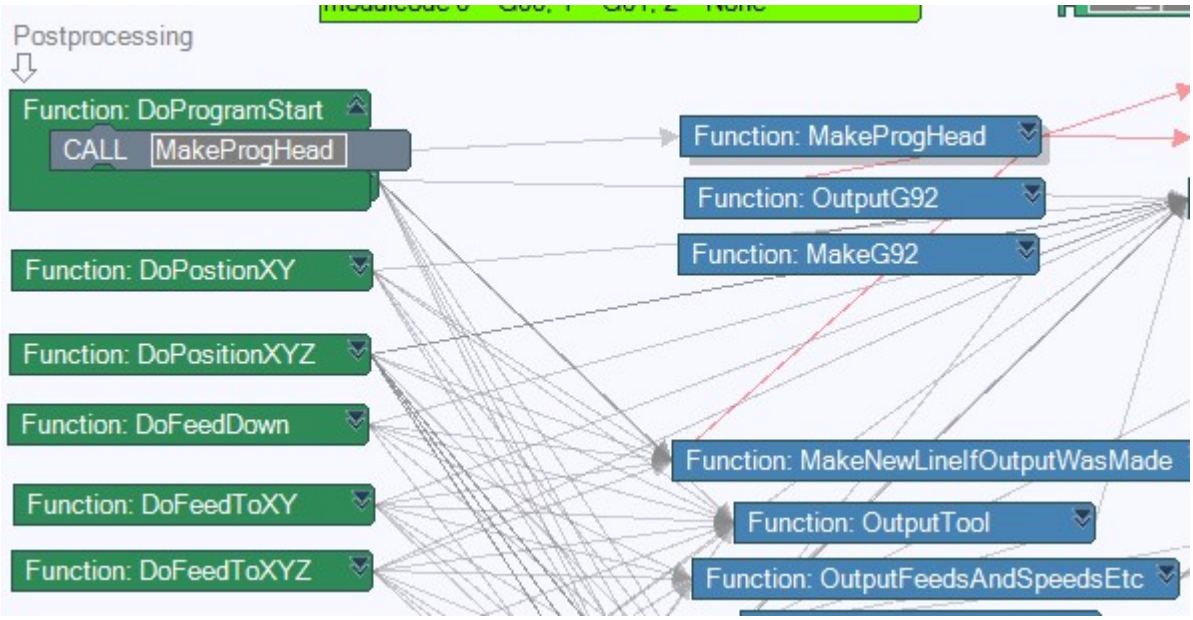
During the post process, SimCam will call various callback functions in the Postprocessing library.



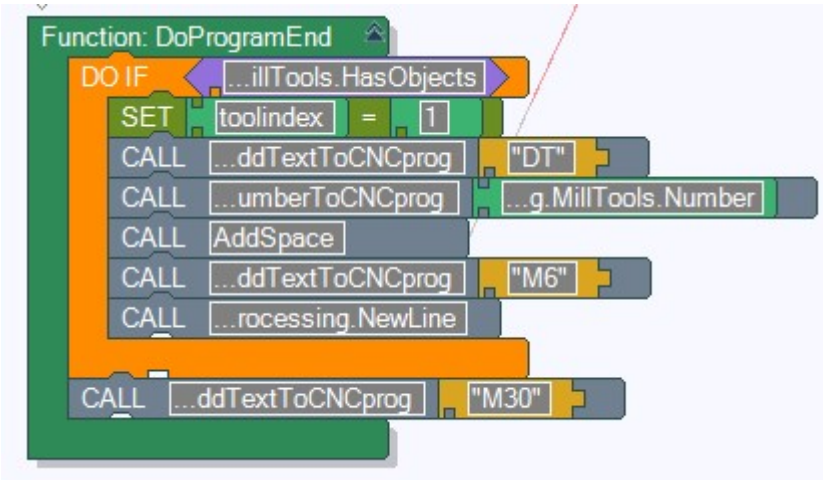
© CNCSimulator.com 2018

You will need to drag all functions that you want your postprocessor to handle into the canvas, and put your blocks into them.

If we look at our demonstration postprocessor, we can see that it handles many of the callback functions. The picture shows a few of them.



Here is a picture of the function that handles the DoProgramEnd callback.



We cannot go through the process of creating a whole postprocessor here, as it would make the instruction far too long. Instead, we have provided a demonstration postprocessor, that you can look at and use as your starting point for creating your own.

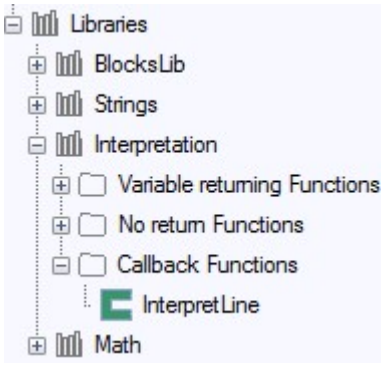
© CNCSimulator.com 2018

Creating Interpreter Scripts

Earlier in this documentation, we have looked at how to create interpreters by using the Interpreter Editor. Here we are going to have a look on how to make them using Blockscript.

When you create a new Interpreter blockscript, you will see a library called "Interpretation".

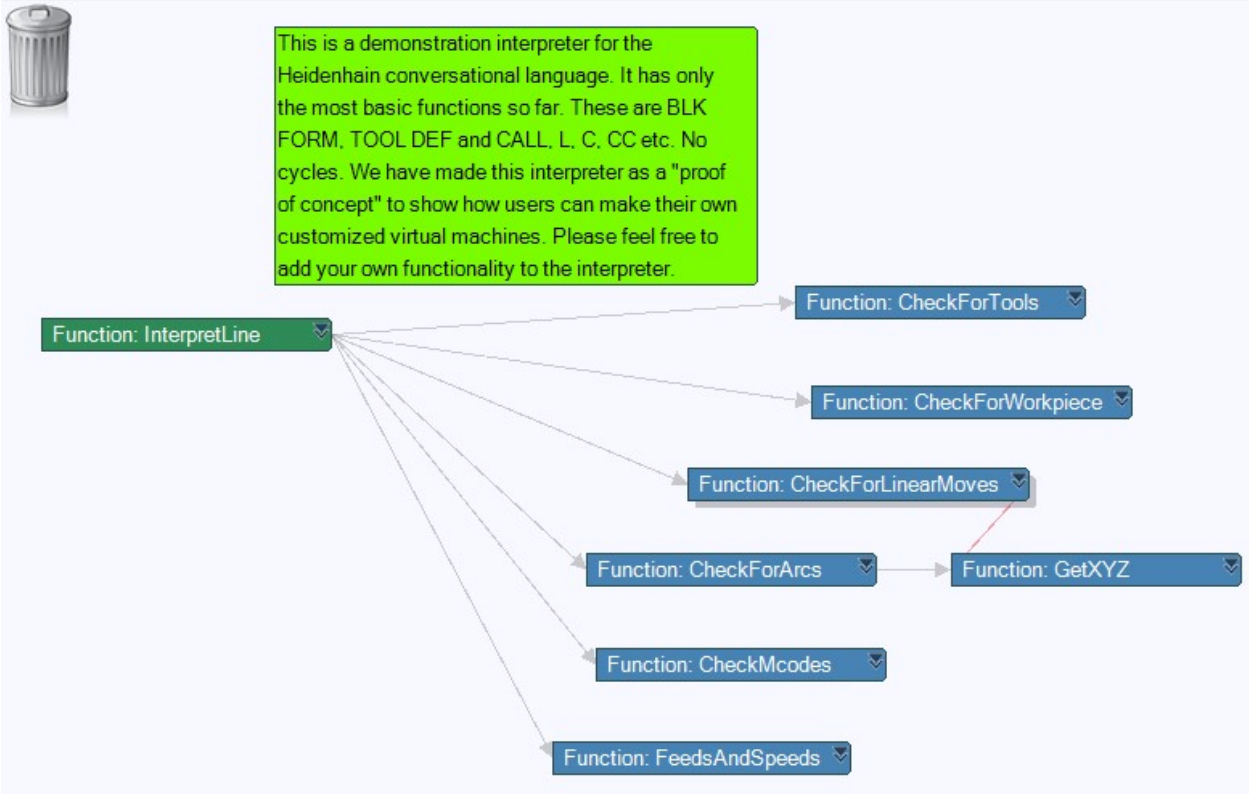
In this library, there is a callback function called, "InterpretLine".



It is the only function you need to handle. It will be called from the simulator for each block in the editor. It is up to you to tell the simulator what to do for every type of CNC instruction in the blocks.

We have made a very basic demonstration interpreter for the Heidenhain conversational language, that you can use as your starting point for writing your own interpreters.

© CNCSimulator.com 2018



© CNCSimulator.com 2018

Actions Reference

Nothing
Does nothing. Use it for labeled dividers in the list.

CoolantOn and CoolantOff
Turns on or off the coolant.

SpindleOnClockwise and SpindleOnAntiClockwise
Starts rotating the spindle clockwise or anti-clockwise.

SpindleOff
Turn off the spindle.

RadiusCompOnLeft and RadiusCompOnRight
Turns on radius compensation to the left or right.

RadiusCompOff
Disables the radius compensation.

© CNCSimulator.com 2018

MoveZeroPoint

Use to move the programmed origin.

If Parameter is 0, it expects the location of the zero point in the X, Y, and Z parameter.

If Parameter is 1, then the index of the zero point (in the zero point registry) is expected as the first numerical parameter.

If Parameter is 2 to 7, then the G54-G59 zero points will be fetched from the zero point registry.

G54.!P	MoveZeroPoint	2
G55	MoveZeroPoint	3
G56	MoveZeroPoint	4
G57	MoveZeroPoint	5
G58	MoveZeroPoint	6
G59	MoveZeroPoint	7

SetIncrementalMode

Set the positioning mode to incremental coordinates.

SetAbsoluteMode

Set the positioning mode to absolute coordinates.

© CNCSimulator.com 2018

SetNewFeedXY

Set the XY-feed.

The feed value is expected as the first numerical value fetched.

Value pos	Expected
0	Feed rate XY

SetNewFeedZ

Set the Z-feed.

The feed value is expected as the first numerical value fetched.

Value pos	Expected
0	Feed rate Z

SetNewRpm

Set the spindle rpm.

The rpm value is expected as the first numerical value fetched.

Value pos	Expected
0	RPM

© CNCSimulator.com 2018

SetMaxRpm

Set the maximum allowed rpm. This will not create an alarm if the rpm is programmed higher, but it will limit the rpm, affecting for example, the machine time calculations.

The max rpm value is expected as the first numerical value fetched.

Value pos	Expected
0	Max RPM

SetConstantSurfaceSpeed

Put the machine in constant surface speed mode. If there is a numerical value, it will be used as the SFM or SMM value (surface feet per minute/ surface meters per minute).

Value pos	Expected
0	SFM or SMM

SetRpmMode

Set the machine to rpm mode. Cancels Constant Surface Speed mode.

SetPerRevolutionFeed

Set the machine in feed per revolution mode.

SetPerMinuteFeed

Set the machine in feed per minute mode. Cancels feed per revolution mode.

SelectTool

Select a tool from the user-defined tool registry. Exception: if **ReadTasDefinedTool** has been executed, the tool will be read from the program defined tool registry.

SelectDefinedTool

Select a mill tool previously defined by **DefineMillTool.**

SelectEmbeddedTool

Select a tool from the embedded tool registry.

ExecuteToolChange

Execute an automatic tool change.

© CNCSimulator.com 2018

GoFast

Move with full speed to another location.

Expects the new position as the X, Y, and Z parameter.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos

GoSlow

Move with programmed feed rate to another location.

Expects the new position as the X, Y, and Z parameter.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos

GoClockwiseArcCenterAbs

Move along a clockwise arc where the center is programmed with absolute coordinates.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos
3	X center pos
4	Y center pos
5	Z center pos

© CNCSimulator.com 2018

GoClockwiseArcCenterInc

Move along a clockwise arc where the center is programmed with incremental coordinates.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos
3	X center pos
4	Y center pos
5	Z center pos

GoClockwiseArcCenterRadius

Move along a clockwise arc with a specific radius.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos
3	Radius

GoAntiClockwiseArcCenterAbs

Move along an anti-clockwise arc where the center is programmed with absolute coordinates.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos
3	X center pos
4	Y center pos
5	Z center pos

© CNCSimulator.com 2018

GoAntiClockwiseArcCenterInc

Move along an anti-clockwise arc where the center is programmed with incremental coordinates.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos
3	X center pos
4	Y center pos
5	Z center pos

GoAntiClockwiseArcCenterRadius

Move along an anti-clockwise arc with a specific radius.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos
3	Radius

OptionalStop

Perform an optional stop. Will pause the simulation if the Op.stop button on the virtual CNC controller is activated.

ProgramStop

Perform an unconditional stop. Will pause the simulation.

ProgramEnd

Ends the simulation.

Info

Information actions. Not normally used in interpreters.

© CNCSimulator.com 2018

AddMachineTime

Add a programmed amount of time to the machine time counter.

Value pos	Expected
0	Time to add in milliseconds

AddWorkpiece

Add a workpiece to the simulated project.

Value pos	Expected
0	Workpiece index
1	X pos
2	Y pos
3	Z pos

AddDefinedWorkpiece

Add a defined workpiece to the simulated project.

Value pos	Expected
0	Workpiece index
1	X pos
2	Y pos
3	Z pos

AddEmbeddedWorkpiece

Add an embedded workpiece to the simulated project.

Value pos	Expected
0	Workpiece index
1	X pos
2	Y pos
3	Z pos

© CNCSimulator.com 2018

EnforceInches
Enforce the use of inches. If the machine is in millimeter mode, an alarm will show.

EnforceMillimeters
Enforce the use of millimeters. If the machine is in inch mode, an alarm will show.

SetView
Set the viewport.

Expects eighteen numbers that defines the 3D viewport.

Example Trigger Line: $SetView{#}{#}{#}{#}{#}{#}{#}{#}{#}{#}{#}{#}{#}{#}{#}{#}{#}{#}

These values can be created by the user via the Tools menu.

EnforceMill
Enforce the use of a milling machine. If another type of machine is loaded, an alarm will show.

EnforceLathe
Enforce the use of a lathe machine. If another type of machine is loaded, an alarm will show.

GoToEditorLine
Jump to a specified line in the editor.

Value pos	Expected
0	Line index

GotoMachineZeroG28style
Go to the reference position.

Set3DprinterExtruderSpeed
Set the extruder feed for a 3D printer.

Value pos	Expected
0	Extruder speed

© CNCSimulator.com 2018

Set3DprinterExtruderSpeedToZero

Set the extruder feed to 0 for a 3D printer.

CuttingOn and CuttingOff

Turn the cutting on or off in cutting machines.

Dwell

Pause simulation for a specified time.

Value pos	Expected
0	Milliseconds

Reserved

Does nothing.

AbsoluteCenters

Tell the simulator that all circle centers are given in absolute coordinates, in place of the default incremental ones.

SetG28pos

Set the reference position.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos

GoAbsoluteNoOffsetTo

Move with programmed feed with no offset to an absolute position.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos

© CNCSimulator.com 2018

SetToolChangePos

Set the tool change position.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos

ToolLengthCompensationOn

Turn on tool length compensation.

ToolLengthCompensationOff

Turn off tool length compensation.

MoveToolTo

Teleport the tool to a new location.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos

ShowMessage

Show a message on the simulation screen.

Expects one string for the text and two numerical values.

Value pos	Expected
0	Timeout in seconds
1	Pause (0/1)

If value 1 is set to 1, the simulation will pause while the message is visible.

© CNCSimulator.com 2018

SetCutting

Set the workpiece cutting parameters.

Value pos	Expected
0	Cut from left
1	Cut from right
2	Cut from front
3	Cut from back
4	Cut from top

Pause

Pause the simulation for the set amount of milliseconds.

Value pos	Expected
0	milliseconds

StartStopWatch

Start the internal stop watch.

ShowStopWatch

Show the internal stop watch.

OpenDoor and CloseDoor

On machines with doors, these actions open and close the doors.

© CNCSimulator.com 2018

AddMillPart

Creates and adds a milling workpiece to the machine table.

Value pos	Expected
0	X size
1	Y size
2	Z size
3	X position
4	Y position
5	Z position
6	Part color Red
7	Part color Green
8	Part color Blue
9	Top color Red
10	Top color Green
11	Top color Blue
12	Material style

For value number 12, the different material styles are:

0 = Not Solid
1 = Metal
2 = Plastic
3 = Wood
4 = Transparent

Colors are given as RGB values with individual color values in the range of 0-255.

© CNCSimulator.com 2018

SetUpTool

Setup a milling tool and put it in the holder.

Value pos	Expected
0	Tip angle
1	Shaft diameter
2	Shaft length
3	Tool diameter
4	Tool length
5	Tool tip type

For value 5, the different tool tip types are:

0 = Flat
1 = Ball
2 = Pointed

SetBufferResolution

Set the resolution of the 3D solid buffer.

Value pos	Expected
0	Voxels per mm

This special command is used to override the resolution setting for the solid buffer. The value given is the number of 3D pixels (voxels) per millimeter. A value of 1 will make each 3D pixel 1 cubic millimeter big. A value of 10 will make each 3D pixel 0.1 mm^3 big. A value of 1 will create ugly results but will run fast. On the other hand, a value of 10 will create high resolution workpieces, but will run slow. We do not recommend using values higher than 10, as you could easily run out of memory due to huge material buffer allocation. Important: this command must come before a workpiece is added.

DefineMillWorkpiece

Define a milling workpiece.

Expects one string that defines the name and material of the workpiece followed by four numbers:

Value pos	Expected
0	Workpiece index
1	X size
2	Y size
3	Z size

The string has the following format:

"N:name_of_workpiece:M:material_to_use"

© CNCSimulator.com 2018

DefineLatheWorkpiece

Define a lathe workpiece.

Expects one string that defines the name and material of the workpiece followed by four numbers:

Value pos	Expected
0	Workpiece index
1	Length
2	Diameter
3	Inner diameter

The string has the following format:

"N:name_of_workpiece:M:material_to_use"

DefineMaterial

Define a workpiece material.

Expects a string followed by eight numbers:

Value pos	Expected
0	Material index
1	Part color Red
2	Part color Green
3	Part color Blue
4	Top color Red
5	Top color Green
6	Top color Blue
7	Material style

The string has the following format:

"N:name_of_material"

For value number 7, the different material styles are:

0 = Not Solid
1 = Metal
2 = Plastic
3 = Wood
4 = Transparent

Colors are given as RGB values with individual color values in the range of 0-255.

© CNCSimulator.com 2018

DefineMillTool

Define a milling tool.

Expects a string followed by seven numbers:

Value pos	Expected
0	Tool index
1	Tip angle
2	Shaft diameter
3	Shaft length
4	Tool diameter
5	Tool length
6	Tool tip type

The string has the following format:

"N:name_of_tool"

For value 6, the different tool tip types are:

0 = Flat
1 = Ball
2 = Pointed

DefineZeroPoint

Define a zero point.

Expects a string that defines the name of the zero point followed by four numbers.

Value pos	Expected
0	Zeropoint index
1	X pos
2	Y pos
3	Zpos

The string has the following format:
"N:name_of_zeropoint"

© CNCSimulator.com 2018

UseMaterial
Use a material.

Expects one string with the name of the material to use. This action looks for the material in the user defined materials registry.

UseEmbeddedMaterial
Use an embedded material.

Expects one string with the name of the material to use. This action looks for the material in the embedded materials registry.

ShowToolpaths
Turn on or off visible tool paths.

SetCuttingWidth
Set the cut width for cutting machines.

Value pos	Expected
0	Cutting width

InvertLatheWorkpiece
Invert the workpiece in a lathe chuck.

DefineDoffsetDiameter
Define the diameter for D offsets.

Value pos	Expected
0	D offset index
1	D offset value

© CNCSimulator.com 2018

SetWorkingPlaneXY
Set the ARC working plane to X-Y.

SetWorkingPlaneXZ
Set the ARC working plane to X-Z.

SetWorkingPlaneYZ
Set the ARC working plane to Y-Z.

GoFastNoCrashCheck
Move with full speed without using the crash check.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos

LoadMachine
Not yet in use.

BlockSpindleRot
Block the spindle rotation.

ReadTasDefinedTool
Read all tool select values as defined tools.

Restart
Restart the simulation.

© CNCSimulator.com 2018

GoFastMultiAxis

Move with full speed to another location. Special for machines with more than 3 axes.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos
3	Not used
4	C pos

GoSlowMultiAxis

Move with programmed feed rate to another location. Special for machines with more than 3 axes.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos
3	Not used
4	C pos

RotateWorkpiece

Rotate a milling workpiece 90 degrees around the Z axis.

ExecuteToolChangeNoMove

Execute a tool change without moving the tool to the tool change position.

ForceSurfaceColor

Override the default surface color set by the material.

Value pos	Expected
0	Surf color Red
1	Surf color Green
2	Surf color Blue

© CNCSimulator.com 2018

CreateLatheTool

Simulator specific command. Use embedded tools or tools from the user defined lathe tools registry.

UpdatePosition

Used only internally in the simulator.

SetToolPosition

Set the current tool position.

Value pos	Expected
0	X pos
1	Y pos
2	Z pos

RotateCoordSyst

Rotate the coordinate system.

Value pos	Expected
0	X center
1	Y center
2	Angle

CancelCoordRot

Cancel the rotation of the coordinate system.

© CNCSimulator.com 2018

Functions Reference

Functions work a bit different internally than the Actions. Here is a description of each available function.

Comment_betw_these

Put one start character and one end character separated by a comma on the trigger line, to tell the interpreter engine not to read between these characters.

Example Trigger Line: (,)

Comment_from_this_char_to_end_of_line

Put a character here that is used to indicate that the rest of the block is a comment.

Example Trigger Line: ;

© CNCSimulator.com 2018

Call_sub_routine

Function to call a sub routine.

Set the parameter to the type of call you want to make.

0 = Normal subroutine in the same file.
1 - 5 = Not in use.
6 = Call subroutine in other file.

Different calls expect different parameters. Here is a list:

Normal call (0)

Value pos	Expected
0	Block number
1	Repetitions

Prg number in program has to start with either an "O" or a ":".

Examples:

O100
:25

Example Trigger Line: M97,M98,!$[P#][L#]

Call to other file (6)

Value pos	Expected
0	Repetitions

A string defines the filename.

Example Trigger Line: M97,M98,!P{$}[L#]

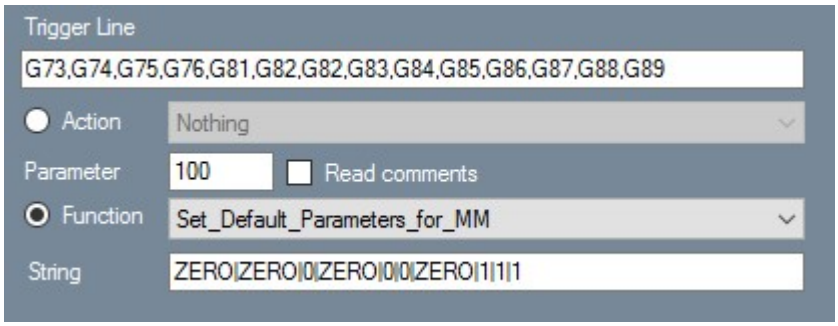
Return_from_sub_routine

Return to block after the calling block, either in the same file or in the calling file.

© CNCSimulator.com 2018

Set_Default_Parameters_for_MM

This function is used to set default parameters for different cycles when the simulator is set to millimeters. Set the parameter field to the ID of the cycle.



In the Parameter field, you set the number of the cycle you want to send the default parameters to. The string lines up the default values separated by vertical bar character. Put the word ZERO where the value should not be initiated to anything. For ZERO values, default values will be used when applicable.

See the Canned Cycles Reference for an explanation of the built-in cycles.

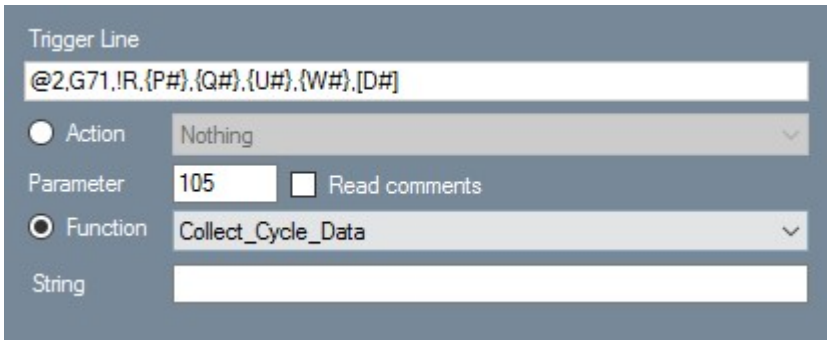
Set_Default_Parameters_for_INCH

This is the same as the previous function, except it is called only when the simulator is set to inches.

Collect_Cycle_Data

Collect data for the build-in canned cycles.

Set the parameter field to the ID of the cycle.



This definition will collect the P, Q, U, W, and D parameters and set them to cycle 105. The P value will have index 2, the Q will have index 3, and so forth.

See the Canned Cycles Reference, for an explanation of the built-in cycles.

© CNCSimulator.com 2018

Set_Cycle_Start_Block

Will set the start block for cycles that has a contour definition spanning several blocks.

Set the parameter field to the ID of the cycle.

Value pos	Expected
0	Start block number

See the Canned Cycles Reference, for an explanation of the built-in cycles.

Set_Cycle_End_Block

Will set the end block for cycles that has a contour definition spanning several blocks.

Set the parameter field to the ID of the cycle.

Value pos	Expected
0	End block number

See the Canned Cycles Reference, for an explanation of the built-in cycles.

Start_Cycle

Start a built-in canned cycle.

Set the parameter field to the ID of the cycle.

See the Canned Cycles Reference, for an explanation of the built-in cycles.

Run_Blocks_betw_Start_and_End_Blocks

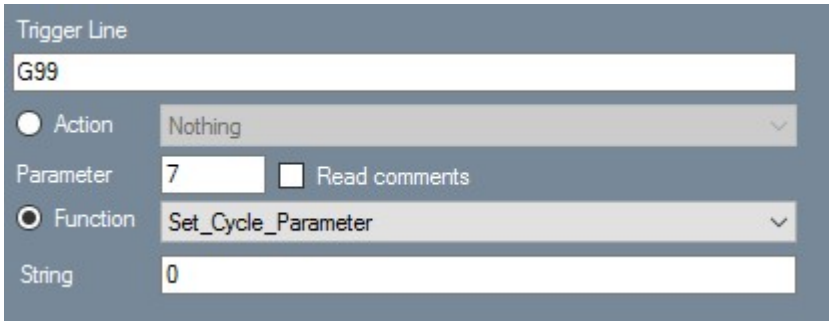
Execute the blocks set by Set_Cycle_Start_Block and Set_Cycle_End_Block.

© CNCSimulator.com 2018

Set_Cycle_Parameter

Set an individual parameter for the last used cycle to the specified value.

For example, this definition will set parameter 7 to 0 when G99 is found. It will set it to the last used cycle. This means that if a Set_Default_Parameters or Collect_Cycle_Data has been called for cycle 100, this is the cycle that will be used here.



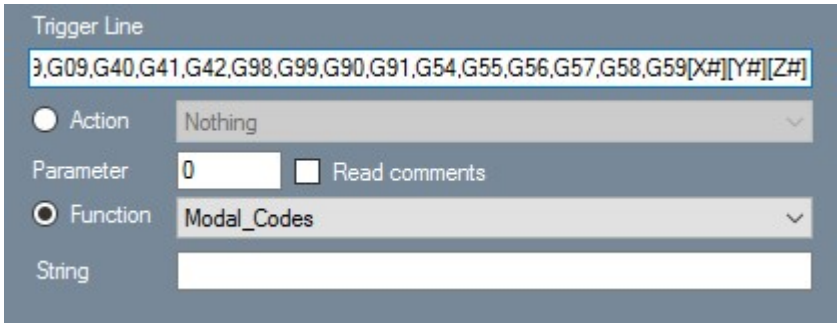
See the Canned Cycles Reference, for an explanation of the built-in cycles.

End_Cycle

Call to end any started cycle.

Modal_Codes

Defines what codes should trigger a modal function.



For example, on the picture we see a definition that tells the interpreter that when G09, G40, G41, G42, etc. is found, collect the X, Y, and Z values and execute the last used modal function (fast transport, feed movement, clockwise, or counter-clockwise arc).

© CNCSimulator.com 2018

Codes_that_cancels_Modal_Codes

Call once for each code that should cancel the current modal function. These definitions should come after definition of modal codes in the list.

Modal Codes	
!G,X,Y,Z,I,J,K,U,W,!G73[X#][Y#][Z#][I#][J#][K...	Modal_Codes
G9,G09,G40,G41,G42,G98,G99,G90,G91,...	Modal_Codes
Codes that breaks active modal code	
G0	Codes_that_cancels_M...
G00	Codes_that_cancels_M...
G1	Codes_that_cancels_M...
G01	Codes_that_cancels_M...
G2	Codes_that_cancels_M...
G02	Codes_that_cancels_M...

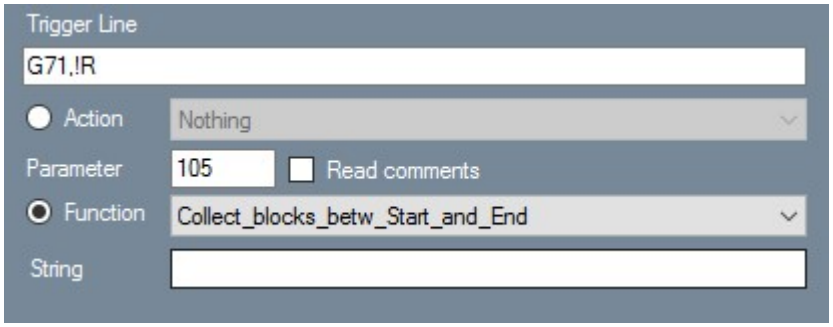
End_of_line_word

Tells the interpreter what characters or words should end a block. The interpreter will not read behind these, except for definitions that have the Read comments checkbox ticked.

Codes that ends a block	
(	End_of_line_word
:	End_of_line_word
/	End_of_line_word

Collect_blocks_betw_Start_and_End

Use this function to tell the interpreter to collect the blocks defined by Set_Cycle_Start_Block and Set_Cycle_End_Block, and send them to the cycle set by given ID.



This example collects the blocks between start and end and sends them to cycle 105.

© CNCSimulator.com 2018

Canned Cycles reference

There are the built-in cycles and their parameters:

100 Drilling Cycle

Parameter position	Function
0	X position (default = current X pos)
1	Y position (default = current Y pos)
2	Z drill depth
3	Drill start depth (default = current Z pos)
4	Peck size
5	Dwell ms
6	Initial depth (default = current Z pos)
7	Retract to initial when value > 0
8	Drill at start position when value > 0
9	Chip break value (default = 0)

© CNCSimulator.com 2018

102 Circular Drilling Cycle

Parameter position	Function
0	X position (default = current X pos)
1	Y position (default = current Y pos)
2	Z drill depth
3	Drill start depth (default = current Z pos)
4	Peck depth
5	Start angle
6	Angle step
7	Radius
8	Number of holes
9	Dwell ms (default = 0)
10	Chip break value (default = 0)

104 Lathe Threading Cycle

Parameter position	Function
0	Not used
1	Depth per cut
2	Not used
3	End X
4	End Z
5	Not used
6	Not used
7	Pitch

© CNCSimulator.com 2018

105 Lathe Roughing and Finishing Cycle

Parameter position	Function
0	Depth per cut
1	Retraction X,Z
2	Clearance X when number of parameter = 4
3	Clearance Z when number of parameter = 4
4	Clearance X when number of parameters > 4
5	Clearance Z when number of parameters > 4
6	Special depth per cut when number of parameters > 6 and parameter 0 = ZERO.

If parameter 6 is used for depth per cut, it will be interpreted as thousands if the value is greater than the start diameter (X).

If the function is going to be used as a finishing cycle, put the word "FINISH" at the string field.

107 Lathe Grooving Cycle

Parameter position	Function
0	Retract value
1	End X
2	End Z
3	Peck value
4	Stepping in Z

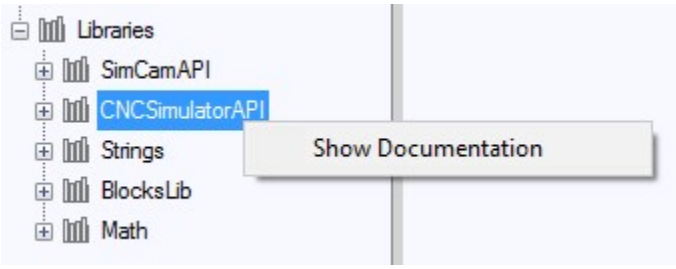
© CNCSimulator.com 2018

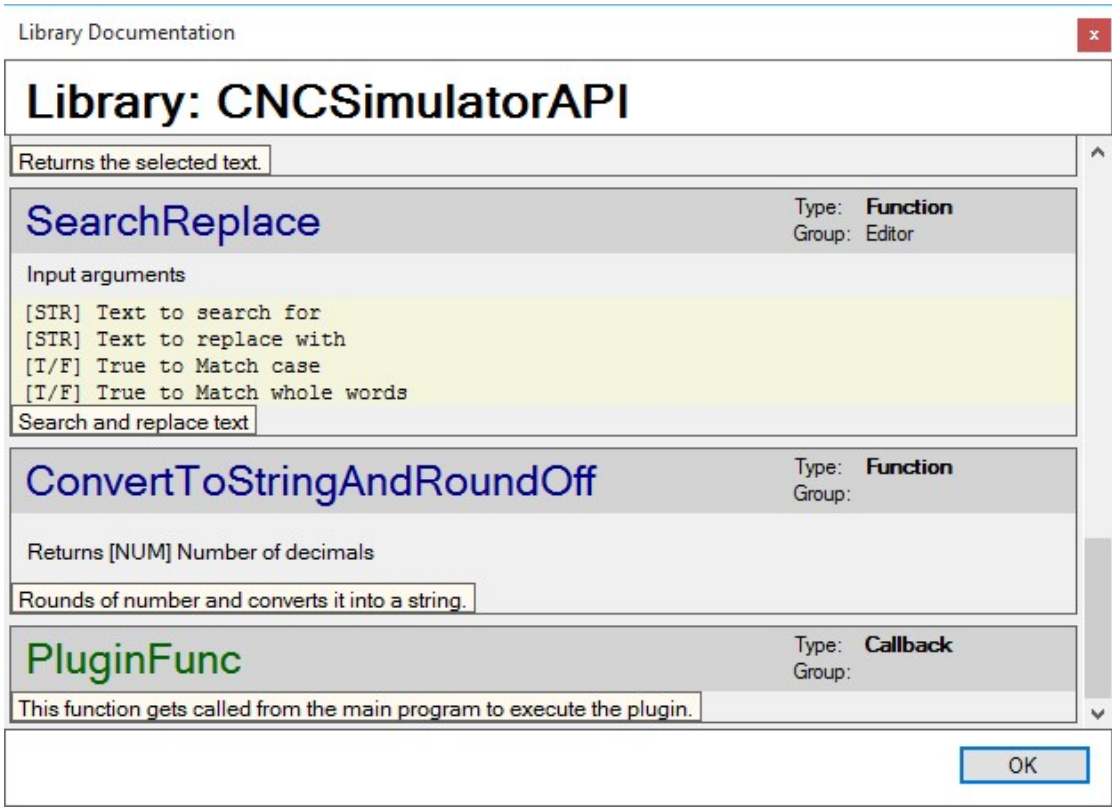
API Libraries

The Blockscript editor comes with a few standard libraries like BlocksLib, Math, and Strings.

When a user creates a blockscript, the editor adds libraries specific to that script type. For example, if a postprocessor script is created, a Postprocessing library is added.

This documentation does not contain a reference to all the library functions, as such a reference would become obsolete quite quickly. Instead, the user can show documentation for a library that will be created on-the-fly, just by right-clicking the library and selecting "Show Documentation".





© CNCSimulator.com 2018

© CNCSimulator.com 2018