Built-in BobCAD-CAM Post Processor Lua Functions

Introduction

This topic contains the reference for built-in BobCAD-CAM Post Processor Lua Functions. These functions can be used through-out the post processor to add more customized modifications to the Post Variables or other Post Processor output.

How to use Lua Functions:

Below will describe how you can set up Lua Functions in the Post Processor. The built-in Lua Functions in this document are already defined. You will be able to call these functions two ways:

## Call Lua Function Method 1:

Use **lua\_func\_FunctionName(args)** in a standard Post Block in the Post Processor.

**Example:** All valid function calls

1002. Start of program with turning cycles

lua\_func\_UnitsComment

lua\_func\_ShowOperationData()

lua\_func\_ShowValueFromOperation("depth\_of\_cut\_or\_stepover")

## Call Lua Function Method 2:

You can also call these Lua functions directly in Lua Blocks 2701 – 2799. Just remove **“lua\_func\_”** and make sure you have parentheses **()**

* Example Function with No Arguments: **ShowOperationData()**
* Example Function with Arguments: **round(-0.23456, 3)**
* Example Function with Table as Argument: **formatNumber({num = "MILL\_GetXRapid", prefix = "X", numDecimalPlaces = 1 , includeDotAfterInt = false})**

For more details about BobCAD Lua APIs, navigate [**HERE**](https://bobcad.com/components/webhelp/BC_Lua/PostProcessing.html)

For more information about the Lua Programming Language, navigate [**HERE**](https://www.lua.org/manual/5.2/)

## Creating Lua Functions

If you want to create your own functions, we have defined two different methods for utilizing Lua scripting with the BobCAD post processing system:

## Method 1

Much like the VB scripting blocks that have existed for years, the same structure can be utilized starting with blocks **2701-2799** within a \*.bcpst post processor file.

By utilizing a variable named **lua\_block\_#**, in a standard Post Block, this will call the corresponding block within the post containing the Lua scripting, for example:

lua\_block\_1 calls the 2701. block

lua\_block\_2 calls the 2702. block

…

lua\_block\_99 calls the 2799. block

Method 2

Alternatively, we have created a second method for linking your Lua scripts to a post processor which involves just keeping all your Lua scripts in a separate \*.lua file with the same name as the \*.BcPst file.

Let's say you are working on the BC\_3x\_Mill.bcpst post processor.

## Method 2a

* You can simply create a BC\_3x\_Mill.lua file and put all of your Lua scripts in this file. These files must be located in the same directory, which is typically the Posts\Mill, Posts\Lathe, or Posts\MillTurn folder located in your products Data folder.

**OR**

## Method 2b

You can also create a sub directory inside of your Posts\Mill, Posts\Lathe, or Posts\MillTurn folder and add block 732. Lua sub folder? "MySubFolder" into the post processor itself, and then the posting engine will look for this sub folder and load any \*.lua files that are located inside of this sub folder.

* Inside of your Posts/Mill folder you can create a folder named whatever you want, let's use 3x\_Mill as our folder name.
* Inside of this 3x\_Mill folder you create the MyLua.lua file which contains all of your Lua scripting functions. The name can be whatever you like and you can also have multiple Lua files.
* Add block 732. Lua sub folder? "3x\_Mill" to the BC\_3x\_Mill.bcpst so that the system knows where to find your Lua scripts.

By utilizing a variable named **lua\_func\_FunctionName** inside the post processor this will call the corresponding Lua function from the loaded Lua file using the methods mentioned above.

# BC\_Lua\_Functions.lua

\*\* You can use any function with or without lua\_func\_

\*\* Use **lua\_func\_** if calling from a standard Post Block. If used in Lua Blocks 2701 – 2799, remove this.

|  |  |
| --- | --- |
| **lua\_func\_ShowOperationData** | |
| Mill Job | This function retrieves the current operations ID value of the Operation in the CAM Tree, iterates through the returned table and displays the key-value pairs in a message box.  Use ShowValueFromOperation function to extract the key or subkey you need to use. |
| Lathe Job |
| Mill Turn Job |
| **Example:**  1003. Tool change for turning cycles  lua\_func\_ShowOperationData | |

|  |  |
| --- | --- |
| **lua\_func\_ShowValueFromOperation("operation\_value")** | |
| Mill Job | This function takes a operation\_value as an argument, retrieves the current operations ID value in the CAM Tree, iterates through the returned table and returns the value of the key or subkey that matches the operation\_value. If the key or subkey is not found, it returns nil.  Use ShowOperationData function to find out to key you need to use |
| Lathe Job |
| Mill Turn Job |
| **Example:** lua\_func\_ShowValueFromOperation("depth\_of\_cut\_or\_stepover") | |

# BC\_NC\_Output\_Lua\_Functions.lua

\*\* You can use any function with or without lua\_func\_

\*\* Use **lua\_func\_** if calling from a standard Post Block. If used in Lua Blocks 2701 – 2799, remove this.

|  |  |
| --- | --- |
| **round(num, numDecimalPlaces)** | |
| Mill Job | Round a number to a specified number of decimal places  \*\* Ideally used in lua blocks 2701 – 2799 |
| Lathe Job | **Args:**  num: The number to be rounded  numDecimalPlaces: The number of decimal places to round to |
| Mill Turn Job | **Returns:**  The rounded number |
| **Example: (In Lua Block 2701 - 2799)**  local threads\_per\_inch = "E" .. round(1 / pitch, 0) | |

|  |  |
| --- | --- |
| **lua\_func\_formatNumber(args)** | |
| Mill Job | Format a number to a specified number of decimal places, with optional leading zero, thousands separator, and dot after integer. |
| Lathe Job | **Args:** A table with the following keys:  **num**: (required) The number to be formatted  **numDecimalPlaces**: The number of decimal places to round to.  Default is rounded to 4 decimal places.  **multiply**: (Optional) A multiplier number to be applied to the number before formatting.  Default is 1.  **add**: (Optional) A number to be added to the number before formatting. Default is 0.  **subtract**: (Optional) A number to be subtracted from the number before formatting.  Default is 0.  **divide**: (Optional) A number to be divided by the number before formatting.  Default is 1.  **includeLeadingZero**: (Optional) (true or false) Whether to include a leading zero for numbers  less than 1. Default is true.  **useThousandsSeparator**: (Optional) (true or false) Whether to include a thousands separator.  Default is false.  **includeDotAfterInt:** (Optional) (true or false) Whether to include a dot after the integer part if  the number is a whole number. Default is true.  **prefix:** (Optional) A string of the prefix to be added to the formatted number. (Primarily used  for BobCAD API functions) |
| Mill Turn Job | **Returns:**  The formatted number as a string |
| formatNumber is a very versatile function that allows you to format just about any value you can think of in a BobCAD-CAM Post Processor.  Using a Table (eg. myTable = { arg1 = 1, arg2 = false }) as an input argument allows you to explicitly define the parameters you want to change. If not explicitly defined, the function will use the default state.    **Example:**  There are 3 ways you can input a value:  **Method 1:** input any number for the “num” argument  lua\_func\_formatNumber({num = 24, includeDotAfterInt = false}) // Ouput: 24  **Method 2:** input a string of a VBScript BobCAD API with no () for the “num” argument  (If Lathe Job X Rapid plane is 2.25 (radius))  lua\_func\_formatNumber({num = "MILL\_GetXRapid", prefix = "X", numDecimalPlaces = 1 , includeDotAfterInt = false}) // Outputs: X2.3  **Method 3:** Use the BcPost.RunVBApi(“VBScript\_BobCAD\_API”) function to grab any value and use for the “num” argument  lua\_func\_formatNumber({num = BcPost.RunVBApi("MILL\_GetXRapid") , prefix = "X", numDecimalPlaces = 1 , includeDotAfterInt = false}) // Outputs: X2.3  **Note:** You can use method 3 to obtain the value of any VBScript BobCAD API and use it as an input for any Lua Function if needed. | |

|  |  |
| --- | --- |
| **convertAngle(angle, mode, numDecimalPlaces)** | |
| Mill Job | Convert an angle between degrees and radians. |
| Lathe Job | **Args:**  angle: A number of the angle to be converted  mode: The conversion mode. Can be "degreesToRadians" or "radiansToDegrees".  numDecimalPlaces: (Optional) The number of decimal places to round to.  Default is no rounding. |
| Mill Turn Job | **Returns:**  The converted angle |
| **Example:**  local angleInRad = convertAngle(180, "degreesToRadians") // Outputs: 3.14159265358979  local angleInRad = convertAngle(180, "degreesToRadians", 4) // Outputs: 3.1416  local angleInDeg = convertAngle(3.1415, "radiansToDegrees", 0) // Outputs: 180  **Note:** Input args sequentially if args is not a table. (eg. lua\_func\_convertAngle(3.1415, 0) is invalid) | |

|  |  |
| --- | --- |
| **includeDotAfterNum(num, includeDotAfterInt)** | |
| Mill Job | Include a dot after the integer numbers. Ideally used in lua blocks 2701 -2799. |
| Lathe Job | **Args:**  num: The number to be formatted  includeDotAfterInt: (true or false) Whether to include a dot after the integer part  of the number |
| Mill Turn Job | **Returns:**  The number with a dot after if it is an integer and if includeDotAfterInt is true |
| **Example: (In Lua Block 2701 - 2799)**  local dwell = includeDotAfterNum(dwell, true) // if input dwell is 3, outputs: 3. | |

|  |  |
| --- | --- |
| **GetValueFromOperation(“operation\_value”)** | |
| Mill Job | Get a value from the current operation based on a search key. Use lua\_func\_ShowOperationData to figure out the “operation\_value”. Ideally used in lua blocks 2701 -2799. |
| Lathe Job | **Args:**  “operation\_value”: A string of the key to search for in the operation's parameters. |
| Mill Turn Job | **Returns:**  The value associated with the search key, or nil if the key is not found. |
| **Example:**  -- Get the thread pitch from the current operation  local threadPitch = GetValueFromOperation("thread\_pitch")  -- Outputs: The value of thread\_pitch, or nil if not found | |

## General Functions

|  |  |
| --- | --- |
| **lua\_func\_UnitsComment** | |
| Mill Job | Outputs a comment in the NC file with the units of the job. |
| Lathe Job |  |
| Mill Turn Job | **Returns:**  Outputs a comment in the NC file with the units of the job. |
| **Used for Post Blocks:**  Used for start of file blocks, but could also be used in tool change blocks as well  630 and 631: Adjust these post questions to set the comment syntax | |
| **Example: (Posted out in the NC File)**  ( Units: inch ) | |

|  |  |
| --- | --- |
| **lua\_func\_IfDwellOutput(prefix, includeDotAfterInt)** | |
| Mill Job | Check if a dwell exists and output with a prefix if it does. This function is used for Peck Drilling cycles since they do not have a separate dwell post block. |
| Lathe Job | **Args:**  prefix: A string of the prefix to be used in the dwell value  includeDotAfterInt: (true or false) Whether to include a dot after the integer part  of the value |
| Mill Turn Job | **Returns:**  The dwell value with a prefix if dwell exists, otherwise nil. |
| **Used for Post Blocks:**  Mill:  73. High speed peck drill canned cycle - Fast peck  83. Peck drill canned cycle  Any other post block that uses a 'dwell' post variable  Lathe:  1126. Peck drill canned cycle  1121. High speed peck drill canned cycle  Any other post block that uses a 'dwell' post variable | |
| **Example:**  1126. Peck drill canned cycle  n,g\_canned\_cycle,x\_f,drill\_depth,reference\_plane,peck\_drill\_increment,lua\_func\_IfDwellOutput("P", true),canned\_feed\_rate  // Output: P3. If 3 set for dwell on Tool Page | |

## Lathe Functions

|  |  |
| --- | --- |
| **lua\_func\_ThreadsPerInch(prefix, numDecimalPlaces, includeDotAfterInt)** | |
| Lathe Job | Convert a pitch value to threads per inch for the Lathe Thread Operation. |
| Mill Turn Job | **Args:**  prefix: A string of the prefix to be used in the threads per inch value  numDecimalPlaces: (Optional) The number of decimal places to round the threads per  inch value to. Default is 4.  includeDotAfterInt: (Optional) (true or false) Whether to include a dot after the integer  part of the value |
|  | **Returns:**  The threads per inch value with a prefix rounded to the nearest whole number. |
| **Used for Post Blocks:**  1087 (Start of thread (G76) cycle) | |
| **Example:**  1087. Start of thread (G76) cycle  n,'G76',thread\_x2,thread\_z2,taper\_height,thread\_first\_cut, lua\_func\_ThreadsPerInch("U"),thread\_angle\_in  **Note:** Input args sequentially if args is not a table. (eg. lua\_func\_ThreadsPerInch("U", true) is invalid) | |

|  |  |
| --- | --- |
| **lua\_func\_RadiusIArcMoveBlock1025** | |
| Lathe Job | Outputs the Lathe arc move post block with I values converted from diameter to radius. I and K values rounded to the nearest 4 decimals. |
| Mill Turn Job | **Args:**  none |
|  | **Returns:**  The whole post block for the Lathe Arc Move (Post Block: 1025) with I values  converted from diameter to radius.  Outputs: n,g\_arc\_move,x\_f,z\_f,'"..arc\_i\_value.."','"..arc\_k\_value.."',feed\_rate |
| **Used for Post Blocks:**  1025 (Arc move (Lathe)) | |
| **Example:**  1025. Arc move  lua\_func\_RadiusIArcMoveBlock1025  // Output: The whole post block rounded to 4 decimal places | |

|  |  |
| --- | --- |
| **lua\_func\_ArcCenterXToRadius(prefixI, prefixK, numDecimalPlaces)** | |
| Lathe Job | Outputs the arc center I and K (Or other specified prefix) values for the Lathe Arc Move post block. |
| Mill Turn Job | **Args:**  prefixI: A string of the prefix to be used in the I value  prefixK: A string of the prefix to be used in the K value  numDecimalPlaces: (Optional) The number of decimal places to round the I and K  values to. Default is 4. |
|  | **Returns:**  The I and K (Or other specified prefix) values for the Lathe Arc Move post block. |
| **Used for Post Blocks:**  1025 (Arc move (Lathe)) | |
| **Example:**  Replace arc\_center with lua\_func\_ArcCenterXToRadius("I", "K", 4)  Example Post Block Line: n,g\_arc\_move,x\_f,z\_f,lua\_func\_ArcCenterXToRadius("I", "K", 4),feed\_rate  **Note:** Input args sequentially if args is not a table. (eg. lua\_func\_ArcCenterXToRadius("I", 4) is invalid) | |

# BC\_Adv\_Posting\_Page.lua

\*\* You can use any function with or without lua\_func\_

\*\* Use **lua\_func\_** if calling from a standard Post Block. If used in Lua Blocks 2701 – 2799, remove this.

The following functions allow you to define an Adv Posting Page directly in the Post Processor without having to create the adv posting custom file manually.

It is REQUIRED to have the **lua\_func\_FinalizeAdvPostingPage** function at the end of the Create functions to create the Adv Posting Custom File.

View the Advanced Posting with Custom Files link [**HERE**](https://bobcad.com/components/webhelp/PostProcessorHelpSystemFiles/Topics/advancedpostingwithcustomfiles1.html) for more info about the Adv Posting Page

### Example Initialization of the Advanced Posting Page in the “Current Settings” page of a Job:

0. File Header

// Initialize the Adv Posting Page

lua\_func\_CreateCheckBox({setPosition = 1, assignCheckBoxLabel = "Use Tool Changer", setDefaultToOnOff = 1})

lua\_func\_CreateComboBox({setPosition = 1, assignComboBoxLabel = "ComboBox Label 1", setDefaultSelection = 1, assignChoiceLabels = {"Choice 1", "Choice 2", "Choice 3"}})

lua\_func\_CreateIntegerEditBox({setPosition = 5, assignEditBoxLabel = "Integer Edit Box", setDefaultIntegerNumber = 10})

lua\_func\_CreateRealEditBox({setPosition = 2, assignEditBoxLabel = "Real Edit Box", setDefaultDecimalNumber = 1.23})

lua\_func\_CreateStringEditBox({setPosition = 3, assignEditBoxLabel = "String Edit Box", setDefaultStringText = "Default Text"})

lua\_func\_FinalizeAdvPostingPage({postProcessorName = "BC\_Single\_Line\_TESTING\_LUA\_FUNCS", extension = "CustomSettings", jobType = "Lathe"})

A screenshot of a computer

Description automatically generated

Below is an image of all the possible combinations of the Adv Posting page:

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **lua\_func\_FinalizeAdvPostingPage(args)** | |
| Mill Job | Takes the string text of the Create functions defined and creates an Advanced Posting Custom File in the specified C:\BobCAD-CAM Data\BobCAD-CAM V36\Posts folder.  **IMPORTANT:** This function is required for all post processors that utilize the Adv Posting Lua Functions. Place it at the bottom of all the Create Adv Posting lua functions. |
| Lathe Job | **Args:** A table with the following keys:  postProcessorName: A string of the exact name of the post processor minus the  extension.  extension: A string of the extension for the Advanced Posting Custom file. Click this  [**LINK**](https://bobcad.com/components/webhelp/PostProcessorHelpSystemFiles/Topics/advancedpostingwithcustomfiles1.html) to view the different extensions used.  jobType: A string of the post processor’s job type.  Use: “mill”, “lathe”, or “millturn” |
| Mill Turn Job | **Returns:**  An Advanced Posting Custom File placed in the BobCAD-CAM Data folder. View the  “Adv Posting” page in the software once you “Post” out the current job at least  once. |
| **Used for Post Blocks:**  Ideally, use in the 0. File Header | |
| **Example FinalizeAdvPostingPage:**  lua\_func\_FinalizeAdvPostingPage({postProcessorName = "BC\_Single\_Line\_TESTING\_LUA\_FUNCS", extension = "CustomSettings", jobType = "Lathe"}) | |

|  |  |
| --- | --- |
| **lua\_func\_CreateCheckBox(args)** | |
| Mill Job | Creates a Check Box on the Adv Posting Page set to a default value. |
| Lathe Job | **Args:** A table with the following keys:  setPosition: An integer number that sets the location of the check box on the Adv  Posting page. Starting at Position 1 through 9  assignCheckBoxLabel: A string of the name given to the check box to distinguish  what the check box is used for.  setDefaultToOnOff: An integer number that sets the default value of the check  box. (0 = Off, 1 = On) |
| Mill Turn Job |  |
| **Used for Post Blocks:**  Ideally, use in the 0. File Header | |
| **Example CreateCheckBox:**  lua\_func\_CreateCheckBox({setPosition = 1, assignCheckBoxLabel = "Use Tool Changer", setDefaultToOnOff = 1}) | |

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **lua\_func\_CreateComboBox(args)** | |
| Mill Job | Creates a Combo Box on the Adv Posting Page set to a default value. |
| Lathe Job | **Args:** A table with the following keys:  setPosition: An integer number that sets the location of the check box on the Adv  Posting page. Starting at Position 1 through 25. Positions 21 – 25 are  wider.  assignComboBoxLabel: A string of the name given to the combo box to distinguish  what the combo box is used for.  setDefaultSelection: An integer number that sets the default choice of the combo  box. Starting at an index of 0 through the number of choice  labels setup. (eg. Choice 1 = 0, Choice 3 = 2)  assignChoiceLabels: a table of choices for selection in the combo box. eg.  assignChoiceLabels = {"Choice 1", "Choice 2", "Choice 3"} |
| Mill Turn Job |  |
| **Used for Post Blocks:**  Ideally, use in the 0. File Header | |
| **Example CreateComboBox:**  lua\_func\_CreateComboBox({setPosition = 1, assignComboBoxLabel = "ComboBox Label 1", setDefaultSelection = 1, assignChoiceLabels = {"Choice 1", "Choice 2", "Choice 3"}}) | |

A close-up of a computer screen

Description automatically generated

|  |  |
| --- | --- |
| **lua\_func\_CreateIntegerEditBox(args)** | |
| Mill Job | Creates an Integer Edit Box on the Adv Posting Page set to a default value. |
| Lathe Job | **Args:** A table with the following keys:  setPosition: An integer number that sets the location of the check box on the Adv  Posting page. Starting at Position 1 through 25. Positions 21 – 25 are  wider.  assignEditBoxLabel: A string of the name given to the integer edit box to  distinguish what the edit box is used for.  setDefaultIntegerNumber: An integer number that sets the default edit box value. |
| Mill Turn Job |  |
| **Used for Post Blocks:**  Ideally use in the 0. File Header | |
| **Example CreateIntegerEditBox:**  lua\_func\_CreateIntegerEditBox({setPosition = 5, assignEditBoxLabel = "Integer Edit Box", setDefaultIntegerNumber = 10}) | |

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **lua\_func\_CreateRealEditBox(args)** | |
| Mill Job | Creates a Real Edit Box on the Adv Posting Page set to a default value. |
| Lathe Job | **Args:** A table with the following keys:  setPosition: An integer number that sets the location of the check box on the Adv  Posting page. Starting at Position 1 through 25. Positions 21 – 25 are  wider.  assignEditBoxLabel: A string of the name given to the real edit box to  distinguish what the edit box is used for.  setDefaultDecimalNumber: A decimal number that sets the default edit box value. |
| Mill Turn Job |  |
| **Used for Post Blocks:**  Ideally, use in the 0. File Header | |
| **Example CreateRealEditBox:**  lua\_func\_CreateRealEditBox({setPosition = 2, assignEditBoxLabel = "Real Edit Box", setDefaultDecimalNumber = 1.23}) | |

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **lua\_func\_CreateStringEditBox(args)** | |
| Mill Job | Creates a Real Edit Box on the Adv Posting Page set to a default value. |
| Lathe Job | **Args:** A table with the following keys:  setPosition: An integer number that sets the location of the check box on the Adv  Posting page. Starting at Position 1 through 25. Positions 21 – 25 are  wider.  assignEditBoxLabel: A string of the name given to the string edit box to  distinguish what the edit box is used for.  setDefaultDecimalNumber: A string that sets the default edit box value. |
| Mill Turn Job |  |
| **Used for Post Blocks:**  Ideally, use in the 0. File Header | |
| **Example CreateStringEditBox:**  lua\_func\_CreateStringEditBox({setPosition = 3, assignEditBoxLabel = "String Edit Box", setDefaultStringText = "Default Text"}) | |

A screenshot of a computer

Description automatically generated

# BobCAD Lua APIs – Post Processing

\*\* These lua APIs are used in Post Blocks 2701 - 2799

|  |  |
| --- | --- |
| **BcPost.RunVBApi("VBScriptAPIName")** | |
| Mill Job | This function can be utilized for calling an existing VB API call from the posting engine.  Please refer to the Posting Variable and API Reference in the help system of the BobCAD-CAM product you are working with for a complete list of the VB APIs.  **IMPORTANT**: This function exists to fully support all of our VB API functions and will be the primary always working method. With that being said it is also important to note that any VB API that we created prior to BobCAD-CAM V32 you can also simply just use BcPost.vbAPIname(). |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **vbApiName** – any existing VB function name as a string that has been implemented to work in the BobCAD posting engine. * **vbInputParas** - the input value to the VB function. For almost all BobCAD posting API's they all only have a single input parameter. For the few instances where the VB API requires multiple input parameters (eg. MILL\_SetIntMemoryLocation(index, integer)) this parameter should be a table containing the various inputs. | |
| **Example:**  -- Get the string from memory location 10  BcPost.RunVBApi("MILL\_GetStringMemoryLoc", 10)  -- Using a table for the second input to pass both parameters to a memory location  BcPost.RunVBApi("MILL\_SetStringMemoryLoc", {10,"Save Me"}) | |

|  |  |
| --- | --- |
| **BcPost.OutputText(“text”)** | |
| Mill Job | This function is utilized to output a string of text inside the NC program. |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **text** – string value of the text to output in the NC program | |
| **Example:**  -- Output the following text  BcPost.OutputText("HERE IS SOME TEXT IN YOUR CODE") | |

|  |  |
| --- | --- |
| **BcPost.ProcessPostLine(“postString”)** | |
| Mill Job | This function is utilized to process a line of standard BobCAD posting variables. Input must be a string formatted exactly as a posting line using system posting variables. (Example: “n, rapid\_move, xr, yr, ‘M08’”).  The system post processes these variables as it would by using the posting engine and outputs the posted string to the posted NC file. |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **postString** – string value of the text to output in the NC program | |
| **Example:**  -- Output the NC code for the following post variables  BcPost.ProcessPostLine("n, rapid\_move, xr, yr, 'M08'")  -- Show an example using a Lua variable to show proper concatenation  -- Remember that concatenation is done with .. operator in Lua  local xVal = 1.0  local yVal = 2.0  BcPost.ProcessPostLine("n,rapid\_move\_forced, 'X"..xVal.."','Y"..yVal.."'") | |

|  |  |
| --- | --- |
| **BcPost.RunPostLine(“postString”)** | |
| Mill Job | This function is utilized to process a line of standard BobCAD posting variables. Input must be a string formatted exactly as a posting line using system posting variables.    The system post processes these variables as it would by using the posting engine and outputs the posted string to the posted NC file.  A newline character will be added to the end of the output code. |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **postString** – string value of the text to output in the NC program | |
| **Example:**  -- Output the NC code for the following post variables  BcPost.RunPostLine("n, rapid\_move, xr, yr, 'M08'") | |

|  |  |
| --- | --- |
| **BcPost.RunBlock(blockNum)** | |
| Mill Job | This function is utilized to process a line of standard BobCAD posting variables. Input must be a string formatted exactly as a posting line using system posting variables. (Example: “n, rapid\_move, xr, yr, ‘M08’”).  The system post processes these variables as it would by using the posting engine and outputs the posted string to the posted NC file. |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **blockNum** – integer value of the posting block that should be called | |
| **Example:**  -- Call block 12. Cutter compensation left to output cutter comp on  BcPost.RunBlock(12) | |

|  |  |
| --- | --- |
| **BcPost.RunPostVariables(“postString”)** | |
| Mill Job | This function is utilized to process a line of standard BobCAD posting variables. Input must be a string formatted exactly as a posting line using system posting variables.    The system post processes these variables as it would by using the posting engine and outputs the posted string to the posted NC file. |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **postString** – string value of the text to output in the NC program | |
| **Example:**  -- Output the NC code for the following post variables  BcPost.RunPostLine("xr, yr") | |

|  |  |
| --- | --- |
| **BcPost.RunVBScript(code)** | |
| Mill Job | This function allows you to pass VB scripting code via the Lua engine. Because our posting engine has always had the VB Scripting for applications engine in it for many years, we implemented this call just in case any existing code would want to be reused.  **IMPORTANT:** The syntax handling for this is important, as it is actually Lua parsing by passing a string and then VB all from our C++ code base, you must be a little stringent on your characters specifically when dealing with quotes and new line characters. Please review the examples below to understand more! |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **code** – string containing the VBScript code that should be run | |
| **Return:**  Will return the output of the VBScript code in a string format when using the MILL\_SetReturnString() function that exists in our VB scripting language. | |
| **Example:**  -- This example shows how to format the quote characters as the input to the function is a string itself  -- The message box being displayed is the VB message box  BcPost.RunVBScript("MsgBox(\"Hi from VB Lua\")")  -- This small example shows the need for the \n newline characters and how to return a value  test = BcPost.RunVBScript("a=17 \n b=12 \n c=a+b \n CALL MILL\_SetReturnString(c)")  Bcc.ShowMessageBox(test, {Title = "Lua calling VB"}) | |

|  |  |
| --- | --- |
| **BcPost.GetValueOfDataBlock(blockNumber)** | |
| Mill Job | This is function is utilized to get the value defined for a Post Question from the post processor.  At the time of this writing we currently know that post questions in blocks 750-999 do not return any value. This is planned to be addressed. |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **blockNumber** – integer value of the block number you wish to retrieve the data from | |
| **Return:**  The value of the requested posting variable which can be various types depending on the block number requested. | |

|  |  |
| --- | --- |
| **BcPost.SetValueOfDataBlock(blockNumber, value)** | |
| Mill Job | This function is utilized to set the value defined for a Post Question from the post processor. |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **blockNumber** – string value of the BobCAD posting variable name * **value** - the value to assign to the post question   **Note:** For the boolean post blocks which use y / n as the item in the post processor, you must use true or false in this function. | |

|  |  |
| --- | --- |
| **BcPost.GetValueOfPostVariable(postVariable)** | |
| Mill Job | This function is utilized get the current value of existing native posting engine variable at the current time this function is called. This API does not currently support all of the BobCAD posting engine variables, and a list of the supported variables can be found in the [Post Variables Supported topic](https://bobcad.com/components/webhelp/BC_Lua/PostVariablesSupported.html).  Please refer to the Posting Variable and API Reference in the help system of the BobCAD-CAM product you are working with for a complete description of the posting variables and VB APIs. |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **postVariable** – string value of the BobCAD posting variable name | |
| **Return:**  The value of the requested posting variable | |
| **Example:**  -- Get the current and previous X Y Z positions  x = BcPost.GetValueOfPostVariable("x\_f")  y = BcPost.GetValueOfPostVariable("y\_f")  z = BcPost.GetValueOfPostVariable("z\_f")  xPrev = BcPost.GetValueOfPostVariable("prev\_x")  yPrev = BcPost.GetValueOfPostVariable("prev\_y")  zPrev = BcPost.GetValueOfPostVariable("prev\_z") | |

|  |  |
| --- | --- |
| **BcPost.GetValueOfOperation(paramName)** | |
| Mill Job | This function is utilized get operation data such as the names, types, and custom posting information. |
| Lathe Job |
| Mill Turn Job |
| **Parameters:**   * **paramName** – this function has a fixed set of parameter strings that you can pass to get different information related to the machining operation that is currently posting. | |
| **Return:**  The value of the requested piece of operation data. | |
| **Example:**  -- Get the current operations type value in the CAM Tree  value = BcPost.GetValueOfOperation("Type")  Bcc.ShowMessageBox("Operation Type: "..value, {Title="Operation Type"})  -- Get the current operations job name in the CAM Tree  value = BcPost.GetValueOfOperation("JobName")  Bcc.ShowMessageBox("Job Name: "..value, {Title="Job Name"})  -- Get the current operations feature name in the CAM Tree  value = BcPost.GetValueOfOperation("FeatureName")  Bcc.ShowMessageBox("Feature Name: "..value, {Title="Feature Name"})  -- Get the current operations name in the CAM Tree  value = BcPost.GetValueOfOperation("OperationName")  Bcc.ShowMessageBox("Operation Name: "..value, {Title="Operation Name"}) | |

The list of possible inputs for***paramName***are shown below:

|  |  |
| --- | --- |
| **paramName** | **Description** |
| **ID** | Get the unique ID of the operation within the document.  It is important to note that the ID value can even change after closing and reopening the document. |
| **Type** | Get a distinct operation type integer which can be utilized for specifically handling a single type of operation.  The definitions for the type table are located in the [Operation Type Reference](https://bobcad.com/components/webhelp/BC_Lua/OperationTypeReference.html) topic |
| **JobName** | Get the name of the machining job that this operation is contained in the CAM Tree |
| **FeatureName** | Get the name of the feature which this operation is contained in the CAM Tree |
| **OperationName** | Get the name of the operation contained in the CAM Tree |
| **UserCheckBoxVariables** | Get a table returned containing the values of all the Advanced Posting pages checkbox variables |
| **UserEditIntegerVariables** | Get a table returned containing the values of all the Advanced Posting pages integer text fields |
| **UserEditRealVariables** | Get a table returned containing the values of all the Advanced Posting pages double / real text fields |
| **UserSelectComboVariables** | Get a table containing the indexes of all the Advanced Posting pages comboboxes |
| **UserEditStringVariables** | Get a table containing all the string values of all the Advanced Posting pages string text fields |