

**Mach3 Version 3.x  
Macro Programmers Reference Manual**

**Draft Revision 0.13  
10/8/2009 7:07:00 PM**

## Table of Contents

Introduction .....	1
Legacy Scripting Functions Grouped Alphabetically .....	2
ActivateSignal .....	2
AppendTeachFile .....	2
AskTextQuestion .....	3
CloseDigFile .....	4
CloseTeachFile .....	4
Code .....	5
CoupleSlave .....	6
DeactivateSignal .....	7
DoMenu .....	7
DoOEMButton .....	8
DoSpinCCW .....	9
DoSpinCW .....	9
DoSpinStop .....	10
EndTHC .....	10
FeedRate .....	10
FileName .....	11
GetABSPosition .....	11
GetACoor .....	12
GetCoord .....	13
GetCurrentTool .....	14
GetDRO .....	14
GetDROString .....	15
GetIJMode .....	15
GetLED .....	16
GetMainFolder .....	17
GetOEMDRO .....	17
GetOEMLed .....	18
GetPage .....	18
GetParam .....	19
GetPortByte .....	21
GetToolParam .....	21
GetRPM .....	22
GetSafeZ .....	22
GetScale .....	23
GetSelectedTool .....	24
GetTimer .....	24
GetToolChangeStart .....	25
GetToolDesc .....	25
GetTurretAng .....	26
GetUserDRO .....	26
GetUserLabel .....	27
GetUserLED .....	27
GetVar .....	28

GetXCoor .....	28
GetYCoor .....	29
GetZCoor .....	30
GotoSafeZ .....	30
HelpAbout .....	31
IsActive .....	31
IsDiameter .....	32
IsEStop .....	32
IsLoading .....	33
IsMoving .....	33
IsOutputActive .....	34
IsSafeZ .....	35
IsStopped .....	35
IsSuchSignal .....	36
JogOff .....	36
JogOn .....	37
LoadFile .....	38
LoadRun .....	38
LoadStandardLayout .....	38
LoadTeachFile .....	39
LoadWizard .....	40
MachMsg .....	40
MaxX .....	42
MaxY .....	42
Message .....	42
MinX .....	43
MinY .....	43
nFmt .....	44
NotifyPlugins .....	44
OpenDigFile .....	44
OpenTeachFile .....	45
Param1 .....	46
Param2 .....	46
Param3 .....	47
PlayWave .....	47
PutPortByte .....	48
Question .....	48
QueueDepth .....	49
Random .....	49
RefCombination .....	50
ResetAxisSwap .....	51
ResetTHC .....	51
RetractMode .....	52
roun .....	52
RunFile .....	53
SaveWizard .....	53

SetButtonText .....	54
SetCurrentTool .....	54
SetDRO .....	55
SetFeedRate .....	55
SetFormula .....	56
SetIJMode .....	57
SetMachZero .....	57
SetOEMDRO .....	58
SetPage .....	58
SetParam .....	58
SetPulley .....	60
SetSafeZ .....	61
SetScale .....	61
SetSpinSpeed .....	62
SetTicker .....	62
SetTimer .....	63
SetToolParam .....	63
SetToolX .....	64
SetToolZ .....	65
SetTriggerMacro .....	65
SetUserDRO .....	66
SetUserLabel .....	66
SetUserLED .....	67
SetVar .....	67
SingleVerify .....	68
SingleVerifyReport .....	68
Sleep .....	69
Speak .....	69
StartTHC .....	70
StraightFeed .....	70
StraightTraverse .....	71
SwapAxis .....	71
SystemWaitFor .....	72
THCOff .....	73
THCOn .....	73
ToggleScreens .....	73
ToolLengthOffset .....	74
VerifyAxis .....	74
ZeroTHC .....	75
Legacy Functions Grouped By Function .....	77
Digitizing .....	77
G-Code & G-code Files .....	77
Lathe-only Functions .....	77
Mach3 Configuration & Status .....	77
Referencing, Verifying & Zeroing Axes .....	78
SafeZ .....	78

Wizards& Plugins .....	78
Machine Status & Control.....	78
Motion Control.....	78
Spindle Control .....	79
Tool Parameters and Toolchanges .....	79
Torch Height Control .....	79
Screensets.....	79
User Dialogs.....	80
Signals and Port I/O .....	80
Teach Files .....	81
Miscellaneous.....	81
Modbus Functions Grouped Alphabetically .....	82
GetInBit.....	82
GetInput .....	82
ResetOutBit.....	82
SetHomannString .....	83
SetModIOString.....	83
SetModOutput.....	84
SetOutBit.....	84
WaitForPoll – Unreliable....	84
Serial Output Functions Grouped Alphabetically .....	86
SendSerial .....	86

## Introduction

This Programmers Reference Manual documents the commonly used VB function calls available to macro programmers using Mach3 version 3. This information is being provided primarily to help Mach3 users understand existing macro code. While this interface will continue to be supported by future Mach3 versions for some period of time, Mach3 version 4 will provide a completely new, much more regular interface, *much* higher functionality interface for VB macro programming. It is *strongly* recommended that all new VB code use the new interface, as support for this old one *will* be discontinued at some point in the, possibly not too distant, future. In addition, it is unlikely there will be any further updates or bug fixes to this now obsolete interface after the release of Mach3 version 4, so any existing bugs and anomalies (and there are quite a few) will remain.

No attempt has been made to make this an exhaustive document covering all of the VB functionality. There are many functions which were never previously documented, or which were documented incompletely or incorrectly. In many cases, these functions are not included in this document. There are a number of functions which were partially documented, but found to either not function as documented, to have significant restriction in their operation, or, in some cases, were felt to be either of no real value, or even risky to use. These functions are generally not included in this document. The functionality described herein has been tested against Mach3 version 3.042.020. Some functions may behave differently in other versions. Some functions will be missing entirely in some earlier versions.

# Legacy Scripting Functions Grouped Alphabetically

## ***ActivateSignal***

Sub ActivateSignal (SigNum As Integer)

This function causes the specified Mach output signal to be driven to its active state. If the signal is defined in Config->Ports&Pins as ActiveHigh, it will be driven to a logic High level, otherwise it will be driven to a logic Low level.

### ***Arguments:***

SignalID must be one of the pre-defined Mach3 VB output signal constants (see VB Constants), or other value or expression that evaluates to one of those values.

### ***Return Value:***

None

### ***Example:***

```
ActivateSignal(OUTPUT2)  ' Turn on Flux Capacitor
Sleep(1000)              ' Give it time to charge fully
DeactivateSignal(OUTPUT2) ' Turn it off
```

### ***See also:***

DeactivateSignal(), VB Constants

## ***AppendTeachFile***

Function AppendTeachFile(Filename As String) As Integer

This function re-opens an existing Teach file at Gcode\Filename in the Mach3 directory, and appends any commands subsequently executed via MDI or Code() to that file, until CloseTeachFile() is executed. The specified file must already exist.

### ***Arguments:***

Filename is the name of the Teach file to be re-opened. The file must reside in the Gcode subdirectory of the Mach3 install directory.

### ***Return Value:***

A non-zero value is returned if the operation was successful.

### ***Example:***

```
' Create a new Teach File in Mach3\Gcode
MyTeachFile = "TeachMe.nc"
Err = OpenTeachFile(MyTeachFile)
```

```

If Err <> 0 Then
    ' Teach file created successfully
    ' Write some G Code to it
    Code "G00 X0 Y0"
    Code "G02 X0 Y0 I-1 J0 F40"
    ' Close the Teach file
    CloseTeachFile()
    ' Now load the teach file for execution
    LoadTeachFile()
Else
    ' OpenTeachFile failed
    Message "Unable to open Teach File"
End If

```

**See also:**

OpenTeachFile(), CloseTeachFile()

## AskTextQuestion

Function AskTextQuestion(Prompt As String) As String

This function displays a dialog box containing the specified prompt string, and waits for the user to enter a text string in the dialogs text box. The user entered string is returned to the caller.

### Arguments:

Prompt string is the string that will be displayed above the text box when the dialog is displayed.

### Return Value:

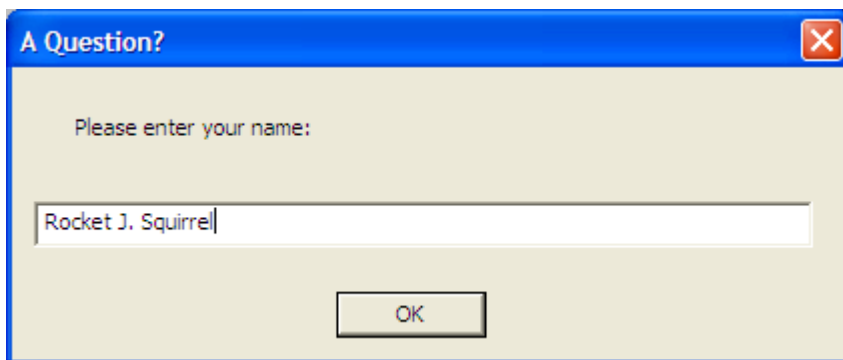
The text string entered by the user.

### Example:

```

Dim UserName As String
UserName = AskTextQuestion("Please enter your name:")
Message "Hello, " & UserName & "!"

```



**See also:**



Message(), Question(), AskTextQuestion(), MachMsg(), GetCoord()

## **CloseDigFile**

Sub CloseDigFile()

This function closes an open digitizing file. If there is no open digitizing file, it does nothing.

### **Arguments:**

None

### **Return Value:**

None

### **Example:**

```
OpenDigFile()      ` Open the digitizing file
ProbeOutline()     ` Call my custom probing function
CloseDigFile()     ` Close the digitizing file
```

### **See also:**

OpenDigFile(), SetProbeActive(), IsProbing()

## **CloseTeachFile**

Sub CloseTeachFile()

This function closes an open Teach file. The file must have previously been opened by either OpenTeachFile, or AppendTeachFile(). If not Teach file is currently open, it does nothing.

### **Arguments:**

None

### **Return Value:**

None

### **Example:**

```
TeachFile = "MyTeachFile.nc"
Dim Err As Integer
Err = OpenTeachFile(TeachFile)
If Err = 0 Then
    MsgBox("Unable To Open: " & Teachfile)
Else
    ' MDI commands entered here are written to TeachFile
    Code("G55")
    Code("G00 X0.000 Y0.000 Z0.500")
End If
' Now close TeachFile
CloseTeachFile()
```

```

' Do something else here
' Now re-open TeachFile for append
Err = AppendTeachFile(TeachFile)
If Err = 0 Then
    MsgBox("Unable To Open: " & TeachFile)
Else
    ' MDI commands entered here are written to TeachFile
    Code("G56")
    Code("G00 X0.000 Y0.000 Z0.500")
End If
' Now close TeachFile
CloseTeachFile()

```

***See also:***

OpenTeachFile(), AppendTeachFile()

## **Code**

Sub Code(Gcode As String)

This function executes the single line of G-code passed as the argument, exactly as if it had been entered on the MDI line, or encountered in a G-Code program. Note that the G-code line is placed in the execution queue, but, in general, the Code() function will return before the line has actually been executed. If it is important that your program know that the line has completed execution, a While loop must be used with the IsMoving() function, as shown in the example below.

Note also that if a Teach file is currently open, the G-code line will not actually be executed, but will simply be written out to the Teach file.

The following optional modes of the Code() function are documented here only for completeness. Their use is discouraged, and support for them may be discontinued at any time, without warning.

Sub Code("LOAD:" & FilePath)

This mode loads a G-Code file from the specified FilePath. The LoadFile() function should be used instead.

Sub Code("SAVE\_XML")

This method immediately saves the XML configuration file.

***Arguments:***

A single G-code line to be executed as a String

***Return Value:***

None

**Example:**

```
` Select our fixture
Code("G55")
` Move away from the vise
Code("G00 X-4.000 Y1.000 Z1.000")
` Wait for movement to complete
While (IsMoving())
    ` Sleep, so other threads can run while we're waiting
    Sleep(100)
Wend
` Done
```

**See also:**

## **CoupleSlave**

Sub Function CoupleSlave(State As Integer)

This function causes any slaved axis to be coupled to, or de-coupled from, its master axis during homing.

**Arguments:**

State is an Integer value of 0 or 1 that defines whether the slave axis should be coupled to its master axis during homing. If State is 0, the axes will be un-coupled. If State is 1, the axes will be coupled.

**Return Value:**

None

**Example:**

```
` Define the axes
Dim Xaxis As Integer
Xaxis = 0
Dim Yaxis As Integer
Yaxis = 0

` Couple the slaved A axis to its master X axis
CoupleSlave(1)
` Home the master and slave
SingleVerify(Xaxis)
` Un-couple the X and A axes
CoupleSlave(0)
```

**See also:**

None

## ***DeactivateSignal***

Sub DeactivateSignal (SigNum As Integer)

This function causes the specified Mach output signal to be driven to its inactive state. If the signal is defined in Config->Ports&Pins as ActiveHigh, it will be driven to a logic Low level, otherwise it will be driven to a logic High level.

### ***Arguments:***

SignalID must be one of the pre-defined Mach3 VB output signal constants (see VB Constants), or other value or expression that evaluates to one of those values.

### ***Return Value:***

None

### ***Example:***

```
ActivateSignal (OUTPUT2)      ` Turn on Flux Capacitor
Sleep(1000)                   ` Give it time to charge fully
DeactivateSignal (OUTPUT2)    ` Turn it off
```

### ***See also:***

ActivateSignal (), VB Constants

## ***DoMenu***

Sub DoMenu (MenuIndex As Integer, MenuItem As Integer)

This function allows a macro script to invoke any function available through any of the Mach3 menus, exactly as if the user had clicked on the menu with the mouse. The specific menu item to be invoked is specified by the two arguments. The first indicates which menu the item to be invoked resides in. The second indicates which item within that menu is to be invoked.

### ***Arguments:***

MenuIndex is the 0-based index of the menu to activate. The File menu is index 0, the Config menu is index 1, etc.

MenuItem is the 0-based index of the menu item to activate. The first item on a given menu is index 0, the second is index 1, etc.

### ***Return Value:***

None

### ***Example:***

```
` Define menu indices, left-to-right, starting with 0
Dim FileMenu As Integer
FileMenu = 0
Dim ConfigMenu As Integer
ConfigMenu = 1
...
` Define File menu items, top-to-bottom, starting with 0
```

```

Dim FileMenuLoadGCodeMenuItem
FileMenuLoadGCodeMenuItem = 0
Dim FileMenuLazyCAMMenuItem As Integer
FileMenuLazyCAMMenuItem = 1
Dim FileMenuCloseFilesMenuItem As Integer
FileMenuCloseFilesMenuItem = 2
Dim FileMenuExitMenuItem As Integer
FileMenuExitMenuItem = 3
` Define Config menu items, top-to-bottom, starting with 0
Dim ConfigMenuDefineNativeUnitsMenuItem As Integer
ConfigMenuDefineNativeUnitsMenuItem = 0
Dim ConfigMenuPortsAndPinsMenuItem As Integer
ConfigMenuPortsAndPinsMenuItem = 1
...
` Pop-up the Config->Ports & Pins dialog
DoMenu(ConfigMenu, ConfigMenuPortsAndPinsMenuItem)

```

**See also:**

DoButton(), DoOEMButton()

## **DoOEMButton**

Sub DoOEMButton(OEMButtonCode As Integer)

This function allows a macro to execute any on-screen button function which has an assigned Button OEM code. The specified function is invoked exactly as if the user had clicked the corresponding on-screen button with the mouse. Note that no actual on-screen button need exist for this function to work. This is simply an easy means to execute any of the “OEM Button” functions through VB.

**Arguments:**

OEMButtonCode must be one of the pre-defined Mach3 OEM Button Code constants (see VB Constants), or other value or expression that evaluates to one of those values.

**Return Value:**

None

**Example:**

```

` Define OEM codes for Mist On and Mist Off
Dim OEMButtonMistOn As Integer
OEMButtonMistOn = 226
Dim OEMButtonMistOff As Integer
OEMButtonMistOff = 227
` Turn Mist coolant on for 3 seconds
DoOEMButton(OEMButtonMistOn)
Sleep(3000)
DoOEMButton(OEMButtonMistOff)

```

**See also:**

DoButton(), DoOEMButton(), DoMenu()

## ***DoSpinCCW***

Sub DoSpinCCW()

This function turns the spindle on, rotating counter-clockwise.

***Arguments:***

None

***Return Value:***

None

***Example:***

```
` Turn on the spindle, turning CCW
DoSpinCCW()
` Let it run 5 seconds
Sleep(5000)
` Now turn it off
DoSpinStop()
```

***See also:***

DoSpinCW(), DoSpinStop()

## ***DoSpinCW***

Sub DoSpinCW()

This function turns the spindle on, rotating clockwise.

***Arguments:***

None

***Return Value:***

None

***Example:***

```
` Turn on the spindle, turning CW
DoSpinCW()
` Let it run 5 seconds
Sleep(5000)
` Now turn it off
DoSpinStop()
```

***See also:***

DoSpinCCW(), DoSpinStop()

## ***DoSpinStop***

Sub DoSpinStop()

This function turns off the spindle.

### ***Arguments:***

None

### ***Return Value:***

None

### ***Example:***

```
` Turn on the spindle, turning CW
DoSpinCW()
` Let it run 5 seconds
Sleep(5000)
` Now turn it off
DoSpinStop()
```

### ***See also:***

DoSpinCW(), DoSpindCCW()

## ***EndTHC***

Sub EndTHC()

This function turns off torch height control. It is functionally identical to THCOff().

### ***Arguments:***

None

### ***Return Value:***

None

### ***Example:***

```
StartTHC() ` Turn on torch height control
...          ` Do some cutting here
EndTHC()    ` Turn off torch height control
```

### ***See also:***

StartTHC(), THCon(), EndTHC(), THCOff(), ZeroTHC(), ResetTHC()

## ***FeedRate***

Sub FeedRate() As Double

This function gets the current feedrate. Note that Feedrate is specified in units per minute.

**Arguments:**

Feedrate specified in units/minute, as a Double

**Return Value:**

None

**Example:**

```
` Set the feedrate to 123.456 inches/minute
SetFeedRate(123.456 / 60)
` Get the current feedrate, in inches/minute, and display it
CurrentFeedrate = FeedRate()
` Display it on the status line
Message "Current feedrate = " & CurrentFeedrate
```

**See also:**

SetFeedrate()

**FileName**

Function FileName() As String

This function returns the filename and path of the currently loaded G-Code file, if any. If no file is currently loaded, the string "No File Loaded." is returned instead.

**Arguments:**

None

**Return Value:**

Current G-Code file name and path, or "No File Loaded."

**Example:**

```
` Show user current G-Code file name and path
Message "Current file is: " & FileName()
```

**See also:**

LoadFile(), LoadRun()

**GetABSPosition**

Function GetABSPosition(Axis As Integer) As Double

This function returns the machine position of the specified axis.

**Arguments:**

Axis is the axis whose machine position is being requested as follows:

0 = X Axis

1 = Y Axis



2 = Z Axis  
3 = A Axis  
4 = B Axis  
5 = C Axis

***Return Value:***

Machine position as a Double

***Example:***

```
` Define the axes
Dim Xaxis As Integer
Xaxis = 0
Dim Yaxis As Integer
Yaxis = 1
Dim Zaxis As Integer
Zaxis = 2

` Get Y Axis Machine Position
Dim AxisPos As Double
AxisPos = GetABSPosition(Yaxis)
` Put it on the status line
Message "Y Axis Machine Pos = " & AxisPos
```

***See also:***

SetMachZero(), MinX(), MaxX(), MinY(), MaxY()

## **GetACoor**

Function GetACoor() As Double

This function is used in conjunction with the GetCoord() function to get X, Y, Z and A axis coordinate values from the user. The GetACoor() function will return the A value entered by the user in the last GetCoord() function call.

***Arguments:***

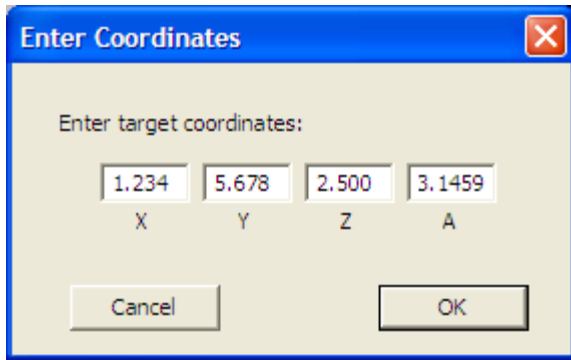
None

***Return Value:***

A Axis coordinate value from last GetCoord() call

***Example:***

```
GetCoord("Enter target coordinates:")
Message "Coordinates are: " & GetXCoor() & " " & GetYCoor() & " "
& GetZCoor() & " " & GetACoor()
```



***See also:***

GetCoord(), GetXCoord(), GetYCoord(), GetZCoord()

## **GetCoord**

Sub GetCoord(Prompt As String)

This function displays a dialog box containing the Prompt string, along with four textboxes, labelled X, Y, Z and A, into which the user can enter four coordinate values. The values are stored in variables within Mach3 which can be retrieved using the GetXCoord(), GetYCoord(), GetZCoord() and GetACoord() functions.

***Arguments:***

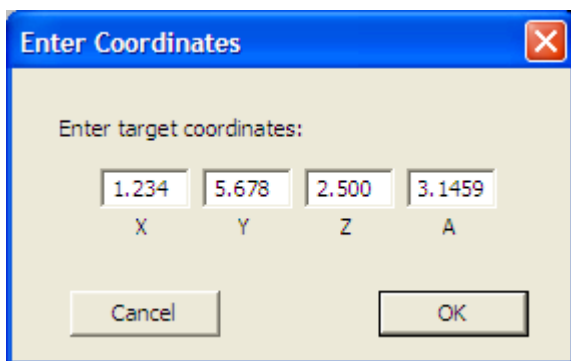
Prompt is a String that is displayed in the dialog box.

***Return Value:***

None

***Example:***

```
GetCoord("Enter target coordinates:")
Message "Coordinates are: " & GetXCoord() & " " & GetYCoord() & " "
& GetZCoord() & " " & GetACoord()
```



***See also:***

GetXCoord(), GetYCoord(), GetZCoord(), GetACoord()

## **GetCurrentTool**

Function GetCurrentTool() As Integer

This function returns the currently active tool number.

### **Arguments:**

None

### **Return Value:**

Current tool number, 1-253.

### **Example:**

```
` Tell the user which tool is active
Message "Current Tool is " & GetCurrentTool() & " = > " &
GetToolDesc()
```

### **See also:**

SetCurrentTool(), GetSelectedTool(), ToolLengthOffset(), GetToolParam(), SetToolParam(),  
GetToolChangeStart(), GetToolDesc(), SetToolX(), SetToolZ()

## **GetDRO**

Function GetDRO(DRONum As Integer) As Double

This legacy function takes the DRO number passed as its argument, adds 800 to it, and returns the value of the Mach OEM DRO of that number. GetDRO(DRONum) is exactly equivalent to GetOEMDRO(DRONum + 800). The only difference between this function and GetOEMDRO() is that this function adds 800 to DRONum to calculate the actual DRO number to be accessed. This function is deprecated, and its use is *strongly* discouraged. Use of the GetOEMDRO/SetOEMDRO and GetUserDRO/SetUserDRO functions is strongly recommended.

### **Arguments:**

DRONum is the OEM DRO number to read, minus 800. Valid DRONums are from 0 to 200, which correspond to OEM DRO numbers 800 to 1000.

### **Return Value:**

Contents of OEM DRO DRONum + 800

### **Example:**

```
` Define the axes
Dim Xaxis As Integer
Xaxis = 800
Dim Yaxis As Integer
Yaxis = 801
Dim Zaxis As Integer
Zaxis = 802
```

```

` Write 1.2345 to Z axis DRO
SetOEMDRO(Zaxis, 1.2345)
` Show the user the Z Axis DRO value, using GetOEMDRO()
MsgBox "Using GetOEMDRO() Z Axis DRO reads: " & GetOEMDRO(Zaxis )
` Now show him the same thing, this time using GetDRO()
MsgBox "Using GetDRO() Z Axis DRO reads: " & GetDRO(Zaxis-800)

```

**See also:**

SetOEMDRO(), GetOEMDRO(), SetUserDRO(), GetUserDRO()

## GetDROString

Function GetDROString(DRONum As Integer) As String

This legacy function takes the DRO number passed as its argument, adds 800 to it, and returns the value of the Mach OEM DRO of that number, rounded to four decimal places, and formatted as a String. This function is deprecated, and its use is **strongly** discouraged. Use of the GetOEMDRO/SetOEMDRO and GetUserDRO/SetUserDRO functions is strongly recommended.

**Arguments:**

DRONum is the OEM DRO number to read, minus 800. Valid DRONums are from 0 to 200, which correspond to OEM DRO numbers 800 to 1000.

**Return Value:**

Contents of OEM DRO DRONum + 800

**Example:**

```

` Define the axes
Dim Xaxis As Integer
Xaxis = 800
Dim Yaxis As Integer
Yaxis = 801
Dim Zaxis As Integer
Zaxis = 802

` Write 1.2345 to Z axis DRO
SetOEMDRO(Zaxis, 1.2345)
` Show the user the Z Axis DRO value, using GetOEMDRO()
MsgBox "Using GetOEMDRO() Z Axis DRO reads: " & GetOEMDRO(Zaxis )
` Now show him the same thing, this time using GetDRO()
MsgBox "Using GetDRO() Z Axis DRO reads: " & GetDRO(Zaxis-800)

```

**See also:**

SetOEMDRO(), GetOEMDRO(), SetUserDRO(), GetUserDRO()

## GetIJMode

Function GetIJMode() As Integer

This function returns the current IJ mode (absolute/incremental), as set in Config->GeneralConfig.

**Arguments:**

None

**Return Value:**

0 indicates absolute IJ mode is enabled

1 indicates incremental IJ mode is enabled

**Example:**

```
` Show user the current IJ mode
If GetIJMode() Then
    Message "IJ Mode is incremental"
Else
    Message "IJ Mode is absolute"
End If
```

**See also:**

SetIJMode()

**GetLED**

Function GetLED(LEDNum As Integer) As Integer

This legacy function takes the LED number passed as its argument, adds 800 to it, and returns the state of the Mach OEM LED of that number. GetLED(LEDNum) is exactly equivalent to GetLED(LEDNum + 800). This function is deprecated, and its use is *strongly* discouraged. Use of the GetOEMLED and GetUserLED/SetUserLED functions is strongly recommended.

**Arguments:**

LEDNum is the OEM LED number to read, minus 800. Valid DRONums are from 0 to 55, which correspond to OEM LED numbers 800 to 855.

**Return Value:**

0 indicates LED (LEDNum+800) is currently turned off

1 indicates LED (LEDNum+800) is currently turned on

**Example:**

```
` Define the LEDs
Dim InchModeLED As Integer
Dim mmModeLED As Integer

` Are we in inch or metric mode?
` First, doing it the "wrong" way...
` Force Inch mode
Code "G20"
InchModeLED = 1
mmModeLED = 2
If GetLED(InchModeLED) Then
```

```

        ` We are in Inch Mode
        Message "Inch Mode"
    ElseIf GetLED(mmModeLED) Then
        ` We are in mm Mode
        Message "mm Mode"
    End If

    ` Now doing it the "right" way...
    ` Force mm mode
    Code "G21"
    InchModeLED = 801
    mmModeLED = 802
    If GetOEMLED(InchModeLED) Then
        ` We are in Inch Mode
        Message "Inch Mode"
    ElseIf GetOEMLED(mmModeLED) Then
        ` We are in mm Mode
        Message "mm Mode"
    End If

```

*See also:*

GetOEMLED(), SetUserLED(), GetUserLED()

## **GetMainFolder**

Function GetMainFolder() As String

This function returns the full path to the Mach3 installation folder.

*Arguments:*

None

*Return Value:*

String full filesystem path to Mach3 installation folder

*Example:*

```

    ` Show the user where Mach3 is installed
    Message "Mach3 is installed at: " & GetMainFolder()

```

*See also:*

## **GetOEMDRO**

Function GetOEMDRO(DRONum As Integer) As Double

This function returns the value of OEM DRO DRONum.

*Arguments:*

DRONum must be a valid OEM DRO number.

***Return Value:***

Returns a Double value of the specified DRO

***Example:***

***See also:***

SetOEMDRO(), SetUserDRO(), GetUserDRO()

## **GetOEMLed**

Function GetOEMLED(LEDNum As Integer) As Integer

This function returns the value of OEM LED LEDNum.

***Arguments:***

LEDNum must be a valid OEM LED number.

***Return Value:***

Returns an Integer value representing the current state of the specified LED. 0 indicates the LED of off (unlit), 1 indicated the LED is on (lit).

***Example:***

***See also:***

SetOEMLED(), SetUserLED(), GetUserLED()

## **GetPage**

Function GetPage() As Integer

This function returns the number of the currently active screenset page.

***Arguments:***

None

***Return Value:***

Current screenset page number, as Integer

***Example:***

```
` Make sure user is on Diagnostics page
If Not GetPage() = 5 Then
    MsgBox "Please switch to Diagnostics page..."
End If
```

***See also:***

SetPage()

## GetParam

Function GetParam(ParamName As String) As Double

This function allows a number of Mach3 internal parameters (not to be confused with G-code parameters) to be read. Each Mach3 parameter is identified by name. The current value of the parameter whose name is given by ParamName is returned as a Double. Valid parameters are:

<b><i>Parameter</i></b>	<b><i>Description</i></b>
XMachine	X axis machine position
YMachine	Y axis machine position
ZMachine	Z axis machine position
Encoder1	Encoder1 Count
Encoder2	Encoder2 Count
Encoder3	Encoder3 Count
Encoder4	Encoder4 Count
MPG1	MPG1 Count
MPG2	MPG2 Count
MPG3	MPG3 Count
XScale	X axis scale factor
YScale	Y axis scale factor
ZScale	Z axis scale factor
AScale	A axis scale factor
BScale	B axis scale factor
CScale	C axis scale factor
FeedRate	Feedrate
Units	Current units (inch/mm). 0 = mm, 1 = inch
StepsPerAxisX	X axis steps per unit
StepsPerAxisY	Y axis steps per unit
StepsPerAxisZ	Z axis steps per unit
StepsPerAxisA	A axis steps per unit
StepsPerAxisB	B axis steps per unit
StepsPerAxisC	C axis steps per unit
VelocitiesX	X axis maximum velocity, from motor tuning, in units/second
VelocitiesY	Y axis maximum velocity, from motor tuning, in units/second
VelocitiesZ	Z axis maximum velocity, from motor tuning, in units/second
VelocitiesA	A axis maximum velocity, from motor tuning, in units/second
VelocitiesB	B axis maximum velocity, from motor tuning, in units/second
VelocitiesC	C axis maximum velocity, from motor tuning, in units/second
AccelerationX	X axis maximum acceleration, from motor tuning
AccelerationY	Y axis maximum acceleration, from motor tuning
AccelerationZ	Z axis maximum acceleration, from motor tuning
AccelerationA	A axis maximum acceleration, from motor tuning
AccelerationB	B axis maximum acceleration, from motor tuning



AccelerationC	C axis maximum acceleration, from motor tuning
SpindleSpeed	Should modify Spindle Speed, but does not work in all versions. User SetSpinSpeed() instead.
ZInhibitOn	Z Inhibit Enable. 0=Z inhibit disabled, 1=Z inhibit enabled. When Z inhibit is enabled, the Z axis will not be allowed to move below the depth specified by the ZinhibitDepth parameter.
ZInhibitDepth	Z Inhibit Depth. When Z inhibit is enabled, the Z axis will not be allowed to move below the depth specified by the ZinhibitDepth parameter.
SafeZ	SafeZ height
XDRO	X axis DRO
YDRO	Y axis DRO
ZDRO	Z axis DRO
ADRO	A axis DRO
BDRO	B axis DRO
CDRO	C axis DRO
Boundry	Toolpath Boundaries display enable. 0=>disable boundaries display, 1=>enable boundaries display
XRefPer	X axis homing speed, as % of rapid speed
YRefPer	Y axis homing speed, as % of rapid speed
ZRefPer	Z axis homing speed, as % of rapid speed
ARefPer	A axis homing speed, as % of rapid speed
BRefPer	B axis homing speed, as % of rapid speed
CRefPer	C axis homing speed, as % of rapid speed
TotalHours	Running count of total Mach3 up-time

**Arguments:**

ParamName is the String name of the parameter to be returned. This must be one of the above names.

**Return Value:**

Current value of the requested parameter as a Double

**Example:**

```
` Get the new scale factor from the user
ScaleFactor = Question "Enter new scale factor:"
` Set the new scale factor for X/Y/Z
SetParam("Xscale", ScaleFactor)
SetParam("Yscale", ScaleFactor)
SetParam("Zscale", ScaleFactor)
```

**See also:**

SetParam()

## **GetPortByte**

Function GetPortByte(PortAddr As Integer) As Integer

This function reads the 8-bit PC I/O port whose address is given by PortAddr, and returns the 8-bit data value read from the port as an unsigned integer value. This function can be used for reading hardware devices not directly supported by Mach3.

### **Arguments:**

PortAddr is the Integer address of the port to be read

### **Return Value:**

Unsigned 8-bit integer value read from the port.

### **Example:**

```
` Our port address
PortAddr = 1016 ` 0x3f8
` Read data register of parallel port at 0x3f8
PortData = GetPortByte(PortAddr)
```

### **See also:**

PutPortByte()

## **GetToolParam**

Function GetToolParam(ToolNum As Integer, ParamNum As Integer)

This function allows any tool parameter, except the description text, for any tool to be read. ToolNum is the number of the tool whose parameters are being set, and can be from 1 to 255. ParamNum is a parameter number, defined as follows:

For Mach3Mill:

- 1 = Diameter
- 2 = Z Offset
- 3 = X Wear
- 4 = Z Wear

For Mach3Turn:

- 1 = Tip Type
- 2 = Tool Radius
- 3 = X Offset
- 4 = Z Offset
- 5 = X Wear
- 6 = Z Wear
- 7 = Turret Angle

### **Arguments:**

ToolNum is an Integer tool number, and must be between 1 and 255.

### **Return Value:**

Requested parameter value, as a Double

**Example:**

```
` Define some constants
DiameterParam = 1
ZoffsetParam = 2
Xwear Param = 3
ZwearParam = 4
` Display tool #23 parameters
Diam = GetToolParam(23, DiameterParam)
Length = GetToolParam(23, ZoffsetParam)
Xwear = GetToolParam(23, XwearParam)
Zwear = GetToolParam(23, ZwearParam)
Desc = GetToolDesc(23)
Message "Tool 23: Diam=" & Diam & " Length=" & Length _
& " Xwear=" & Xwear & " Zwear=" & Zwear & " Desc=" & Desc
```

**See also:**

SetToolParam(), GetToolDesc()

## **GetRPM**

Function GetRPM() As Double

This function returns the currently commanded spindle speed (S-word) as a Double. Note that this returns the most recent S-word value, and not the actual spindle RPM.

**Arguments:**

None

**Return Value:**

Currently commanded spindle speed as a Double.

**Example:**

```
` Show current S-word
Message "S-Word = " & GetRPM()
```

**See also:**

SetSpinSpeed(), DoSpinCW(), DoSpinCCW(), DoSpinStop()

## **GetSafeZ**

Function GetSafeZ() As Double

This function returns the current SafeZ height.

**Arguments:**

None

**Return Value:**

Current SafeZ height as a Double.

**Example:**

```
` Get current SafeZ height
OldSafeZ = GetSafeZ()
` Set new SafeZ height
NewSafeZ = 1.5750
SetSafeZ(NewSafeZ)
...
` Restore old SafeZ height
SetSafeZ(OldSafeZ)
```

**See also:**

SetSafeZ()

## GetScale

Function GetScale(Axis As Integer) As Double

This function returns the current scale factor for axis Axis.

**Arguments:**

Axis is the Integer Axis. 0=X, 1=Y, 2=Z, 3=A, etc.

**Return Value:**

Current scale factor for specified axis, as a Double

**Example:**

```
` Define some constants
Xaxis = 0
Yaxis = 1
Zaxis = 2
` Get the current axis scale factors
OldXScaleFactor = GetScale(Xaxis)
OldYScaleFactor = GetScale(Yaxis)
OldZScaleFactor = GetScale(Zaxis)
` Get the new scale factor from the user
ScaleFactor = Question("Enter new scale factor:")
` Set new scale factors for X/Y/Z
SetScale(Xaxis, ScaleFactor)
SetScale(Yaxis, ScaleFactor)
SetScale(Zaxis, ScaleFactor)
```

**See also:**

SetScale()

## **GetSelectedTool**

Function GetSelectedTool() As Integer

This function returns the tool specified by the most recent toolchange (M6) command. This function is typically used in the M6Start macro to make the selected tool the current tool.

### **Arguments:**

None

### **Return Value:**

Selected tool as an Integer

### **Example:**

```
` Sample M6Start macro
` Get selected tool
NewTool = GetSelectedTool()
` Make it the current tool
SetCurrentTool(NewTool)
```

### **See also:**

GetCurrentTool(), SetCurrentTool()

## **GetTimer**

Function GetTimer(TimerNum As Integer) As Double

This function returns the current count for the specified timer. Mach3 provides 25 timers, numbered 0 to 24, which can be used for timing in VB scripts. To time an event, first clear the timer using SetTimer(), then use GetTimer() to read the timer. Note that this function works only with the parallel port driver, and support for this function may be removed without notice in a future release.

### **Arguments:**

TimerNum is an Integer timer number, which must be between 0 and 24.

### **Return Value:**

Double value of timer TimerNum.

### **Example:**

```
` Clear timer 15
SetTimer(15)
` Wait for OEM Trigger 10 to go active
While IsActive(OEMTRIG10) = False Then
    Sleep 10
Wend
` See how long it took
Message "OEMTRIG10 active after " & GetTimer(15) & " seconds"
```

### **See also:**

SetTimer()

## **GetToolChangeStart**

Function GetToolChangeStart(Axis As Integer) As Double

This function returns the position of the specified axis at the time a toolchange started. This is typically used in an M6End macro to restore the axis positions to the positions they were in before the toolchange.

### **Arguments:**

Axis is the Integer Axis. 0=X, 1=Y, 2=Z, 3=A, etc.

### **Return Value:**

Double position of the specified axis at the start of the last toolchange.

### **Example:**

```
` Example M6End macro
` Define some constants
Xaxis = 0
Yaxis = 1
Zaxis = 2
` Move all axes back to where they were before the toolchange
Xpos = GetToolChangeStart(Xaxis)
Ypos = GetToolChangeStart(Yaxis)
Zpos = GetToolChangeStart(Zaxis)
Code "G00 X" & Xpos & " Y" & Ypos & " Z" & Zpos
` Wait for move to complete
While IsMoving()
    Sleep 100
Wend
```

### **See also:**

GetSelectedTool(), GetCurrentTool(), SetCurrentTool()

## **GetToolDesc**

Function GetToolDesc(ToolNum As Integer) As String

This function returns the tool descriptor text for the tool specified by ToolNum.

### **Arguments:**

ToolNum is an Integer tool number, from 1 to 255

### **Return Value:**

Tool table description text for specified tool, as a String

### **Example:**

```
` Show user the current tool description
Message "Tool " & GetCurrentTool() & ": " & _
GetToolDesc(GetCurrentTool())
```

*See also:*

GetToolParam(), SetToolParam()

## **GetTurretAng**

Function GetTurretAng() As Double

This function returns the current lathe tool turret angle.

*Arguments:*

None

*Return Value:*

Current lathe tool turret angle, as a Double

*Example:*

```
` Display the current tool turret angle
Message "Tool turret position = " & GetTurretAng()
```

*See also:*

## **GetUserDRO**

Function GetUserDRO(DRONum As Integer) As Double

This function returns the value of User DRO DRONum as a Double.

*Arguments:*

DRONum is the Integer User DRO number to be set. Valid User DRO numbers range from 1000-2254.

*Return Value:*

Current value of UserDRO DROVal as a Double

*Example:*

```
` Define some constants
MyWidgetDRO = 1125
` Set MyWidgetDRO to 1.234
SetUserDRO(MyWidgetDRO, 1.234)
...
` Get current value of MyWidgetDRO
MyDROVal = GetUserDRO(MyWidgetDRO)
```

*See also:*

GetUserDRO(), SetOEMDRO(), GetOEMDRO()

## ***GetUserLabel***

Function GetUserLabel(LabelNum As Integer) As String

This function allows the user to retrieve the value of an on-screen User label. “User” labels are those that are created in the screen designer with the default text containing the String “UserLabel” followed by one or more digits.

### ***Arguments:***

LabelNum is the numeric portion of the user label default text. LabelNum must be between 0 and 255.

### ***Return Value:***

None

### ***Example:***

```
` Change the text in UserLabel25
SetUserLabel(25, "This is Label 25")
...
` Retrieve the text from UserLabel25
LabelText = GetUserLabel(25)
```

### ***See also:***

SetUserLabel()

## ***GetUserLED***

Function GetUserLED(LEDNum As Integer) As Integer

This function allows the current state of a User LED to be retrieved.

### ***Arguments:***

LEDNum is the User LED whose state is to be retrieved, which must be in the range of 1000 to 2254

### ***Return Value:***

Current state of the specified User LED. 0 indicates the LED is off (unlit), 1 indicates the LED is on (lit).

### ***Example:***

```
` Define some constants
FluxCapacitorControl = OUTPUT1 ` Output that controls the flux
capacitor
FluxCapacitorLED = 1234 ` LED that indicates flux capacitor is
active
` Turn on the Flux capacitor
ActivateSignal(FluxCapacitorControl)
` Turn on the Flux Capacitor LED for the operator
SetUserLED(FluxCapacitorLED, 1)
```



```
...  
' Is the Flux Capacitor on?  
FluxCapacitorOn = GetUserLED(FluxCapacitorLED)
```

**See also:**

GetUserLED(), SetOEMLED(), GetOEMLED()

## **GetVar**

Function GetVar(VarNum As Integer) As Double

This function returns the current value of the Mach variable specified by VarNum as a Double. Mach variables are accessible both to VB scripts, using the SetVar() and GetVar() functions, as well as G-code programs, using the #nnnn syntax.

**Arguments:**

VarNum is Integer the number of the Mach variable to be retrieved.

**Return Value:**

Current value of the specified variable, as a Double.

**Example:**

```
' Set a variable 1234 to our target position of 2.3456  
SetVar(1234, 2.3456)  
' Now move X to our target position  
Code "G0 X #1234"  
' Or, another way...  
Code "G0 X " & GetVar(1234)
```

**See also:**

SetVar()

## **GetXCoor**

Function GetXCoor() As Double

This function is used in conjunction with the GetCoord() function to get X, Y, Z and A axis coordinate values from the user. The GetXCoor() function will return the X value entered by the user in the last GetCoord() function call.

**Arguments:**

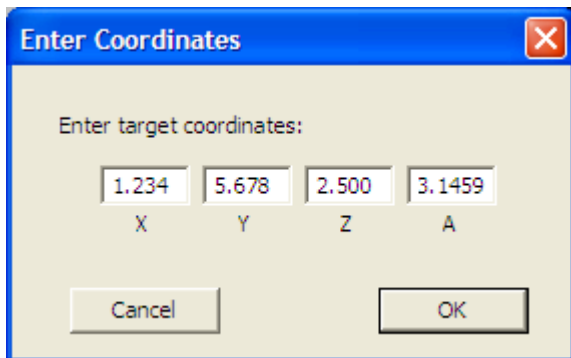
None

**Return Value:**

X Axis coordinate value from last GetCoord() call

**Example:**

```
GetCoord("Enter target coordinates:")  
Message "Coordinates are: " & GetXCoord() & " " & GetYCoord() _  
& " " & GetZCoord() & " " & GetACoord()
```



*See also:*

GetCoord(), GetYCoord(), GetZCoord(), GetACoord()

## **GetYCoord**

Function GetYCoord() As Double

This function is used in conjunction with the GetCoord() function to get X, Y, Z and A axis coordinate values from the user. The GetYCoord() function will return the Y value entered by the user in the last GetCoord() function call.

*Arguments:*

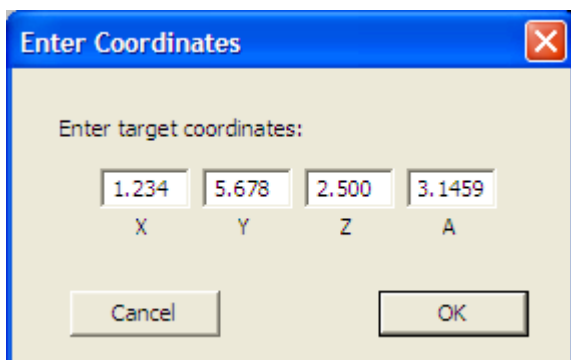
None

*Return Value:*

Y Axis coordinate value from last GetCoord() call

*Example:*

```
GetCoord("Enter target coordinates:")  
Message "Coordinates are: " & GetXCoord() & " " & GetYCoord() _  
& " " & GetZCoord() & " " & GetACoord()
```



**See also:**

GetCoord(), GetXCoord(), GetZCoord(), GetACoord()

## GetZCoord

Function GetZCoord() As Double

This function is used in conjunction with the GetCoord() function to get X, Y, Z and A axis coordinate values from the user. The GetZCoord() function will return the Z value entered by the user in the last GetCoord() function call.

**Arguments:**

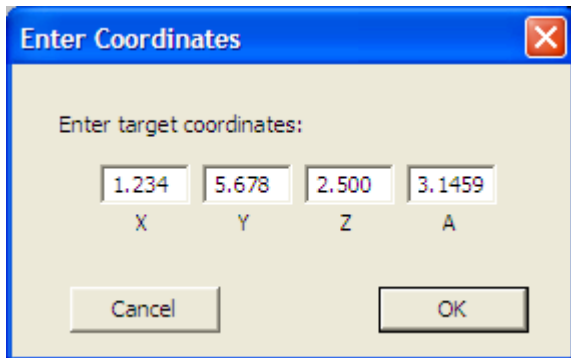
None

**Return Value:**

Z Axis coordinate value from last GetCoord() call

**Example:**

```
GetCoord("Enter target coordinates:")  
Message "Coordinates are: " & GetXCoord() & " " & GetYCoord() _  
& " " & GetZCoord() & " " & GetACoord()
```



**See also:**

GetCoord(), GetXCoord(), GetYCoord(), GetACoord()

## GotoSafeZ

Sub GotoSafeZ()

This function will move the Z axis to the Safe\_Z position, if Safe\_Z is enabled in Config->Safe\_Z Setup. If Safe\_Z is not enabled, an error message will be displayed on the status line, and no move takes place.

**Arguments:**

None

**Return Value:**

None

**Example:**

```
` Move Z axis to Safe_Z position  
GotoSafeZ()
```

**See also:**

GetSafeZ(), SetSafeZ()

## **HelpAbout**

Sub HelpAbout()

This function displays a dialog box showing the current version of the scripting engine.

**Arguments:**

None

**Return Value:**

None

**Example:**

```
` Show the user the scripting engine version  
HelpAbout()
```

**See also:**

## **IsActive**

Function IsActive(Signal As Integer) As Boolean

This function returns a Boolean True if the current state of the specified input signal is its active state. Note that in terms of actual signal level, the term “active” depends on how the signal is defined. If the signal is defined in Config->Ports&Pins->InputSignals as ActiveLow, then IsActive() will return True when that signal is being driven to a logic low. If the signal is defined in Config->Ports&Pins->InputSignals as ActiveHigh, then IsActive() will return True when that signal is being driven to a logic high.

**Arguments:**

Signal is an integer value corresponding to one of pre-defined input signal constants.

**Return Value:**

False indicates the signal is currently in its inactive state

True indicates the signal is currently in its active state

**Example:**

```
` Show the user the state of the INPUT #1 input  
If IsActive(INPUT1) Then
```

```

        Message "INPUT #1 input is active"
Else
    Message "INPUT #1 input is inactive"
End If

```

*See also:*

## ***IsDiameter***

Function IsDiameter() As Integer

This lathe-only function returns 1 if Mach3 is currently operating in Diameter mode.

***Arguments:***

None

***Return Value:***

0 indicates Mach3 is currently operating in Radius mode

1 indicates Mach3 is currently operating in Diameter mode

***Example:***

```

` Tell the user what mode we're in
If IsDiameter() Then
    Message "Currently in Diameter mode"
Else
    Message "Currently in Radius mode"
End If

```

*See also:*

## ***IsEStop***

Function IsEStop() As Integer

This function returns 1 if Mach3 is currently in E-Stop mode.

***Arguments:***

None

***Return Value:***

0 indicates Mach3 is currently not in E-Stop mode

1 indicates Mach3 is currently in E-Stop mode

***Example:***

```

Dim FluxCapacitorControl As Integer
FluxCapacitorControl = OUTPUT3

` Are we in E-Stop?

```

```

If IsEStop() Then
    ' Yes, so turn off the flux capacitor
    DeactivateOutput(FluxCapacitorControl)
Else
    ' No, so turn on the flux capacitor
    ActivateOutput(FluxCapacitorControl)
End If

```

*See also:*

## ***IsLoading***

Function IsLoading() As Integer

This function returns 1 if a G-code file is currently being loaded.

### ***Arguments:***

None

### ***Return Value:***

0 indicates a G-code file is not currently being loaded

1 indicates a G-code file is currently being loaded

### ***Example:***

```

Dim FluxCapacitorControl As Integer
FluxCapacitorControl = OUTPUT3

' Are we in loading a new G-code file?
If IsLoading() Then
    ' Yes, so turn off the flux capacitor
    DeactivateOutput(FluxCapacitorControl)
Else
    ' No, so turn on the flux capacitor
    ActivateOutput(FluxCapacitorControl)
End If

```

*See also:*

## ***IsMoving***

Function IsMoving() As Integer

This function returns 1 if any axis is currently moving. This is most often used when commanding motion within a macro, to pause macro execution until the motion is complete. . This is a complement to IsStopped().

**Arguments:**

None

**Return Value:**

0 if all axes currently stopped  
1 if any axis is currently moving

**Example:**

```
` Move Z axis to Safe_Z position
GotoSafeZ()
` Wait for SafeZ move to complete
Sleep(100)
While IsMoving()
    Sleep(100)
Wend
```

**See also:**

IsStopped()

## **IsOutputActive**

Function IsOutputActive(Signal As Integer) As Boolean

This function returns Boolean True if the current state of the specified output signal is its active state. Note that in terms of actual signal level, the term “active” depends on how the signal is defined. If the signal is defined in Config->Ports&Pins->OutputSignals as ActiveLow, then IsOutputActive() will return a True value when that signal is being driven to a logic low. If the signal is defined in Config->Ports&Pins->InputSignals as ActiveHigh, then IsOutputActive() will return a True value when that signal is being driven to a logic high.

**Arguments:**

Signal is an integer value corresponding to one of pre-defined output signal constants.

**Return Value:**

A Boolean value indicating the state of the specified output signal.

**Example:**

```
` Set OUTPUT1 to its active state
ActivateSignal(OUTPUT1)
` Show the user the state of the OUTPUT #1 input
If IsActive(OUTPUT1) Then
    MsgBox "OUTPUT #1 output is active"
Else
    MsgBox "OUTPUT #1 output is inactive"
End If
` Set OUTPUT1 to its inactive state
DeactivateSignal(OUTPUT1)
` Show the user the state of the OUTPUT #1 input
```

```

If IsActive(OUTPUT1) Then
    MsgBox "OUTPUT #1 output is active"
Else
    MsgBox "OUTPUT #1 output is inactive"
End If

```

*See also:*

## ***IsSafeZ***

Function IsSafeZ() As Integer

This function returns 1 if Safe\_Z is enabled in Config->Safe\_Z Setup.

### ***Arguments:***

None

### ***Return Value:***

0 indicates Safe\_Z is not enabled in Config->Safe\_Z Setup

1 indicates Safe\_Z is enabled in Config->Safe\_Z Setup

### ***Example:***

```

` Show user IsSafeZ()
If IsSafeZ() Then
    Message "SafeZ is enabled"
Else
    Message "SafeZ is disabled"
End If

```

*See also:*

## ***IsStopped***

Function IsStopped() As Integer

This function returns 1 if all axes are currently stopped. This is most often used when commanding motion within a macro, to pause macro execution until the motion is complete. This is a complement to IsMoving().

### ***Arguments:***

None

### ***Return Value:***

0 if any axis is currently moving

1 if all axes are currently stopped

### ***Example:***

```

` Move Z axis to Safe_Z position

```



```

GotoSafeZ()
` Wait for SafeZ move to complete
Sleep(100)
While Not IsStopped()
    Sleep(100)
Wend

```

*See also:*

IsMoving()

## ***IsSuchSignal***

Function IsSuchSignal(SignalID As Integer) As Integer

This function returns an Integer value indicating whether the specified signal is defined in Config->Ports&Pins. A 0 return value indicates the signal is not defined, while a non-zero return value indicates the signal is defined. This can be used, for example, to ensure a PROBE input is properly defined before trying to do probing.

### ***Arguments:***

SignalID must be one of the pre-defined Mach3 VB output signal constants (see VB Constants), or other value or expression that evaluates to one of those values.

### ***Return Value:***

Signal definition state, as an Integer. 0 => signal is not defined, 1 => signal is defined

### ***Example:***

```

` Is a YHOME input signal properly defined?
If IsSuchSignal(YHOME) = 0 Then
    Message "Error! No YHOME input is defined"
End If

```

*See also:*

## ***JogOff***

Function JogOff(Axis As Integer)

This function is used to stop jogging of the specified axis. It is typically used in conjunction with JogOn() to jog an axis under control of a script.

### ***Arguments:***

Axis is an Integer, specifying the axis for which to disable jogging. 0=X, 1=Y, 2=Z, 3=A, etc.

### ***Return Value:***

None

**Example:**

```
` Define some constants
Xaxis = 0
Yaxis = 1
Zaxis = 2
Plus = 0
Minus = 1
` Jog the Y axis in the minus direction for one second
JogOn(Yaxis, Minus)
Sleep(1000)
` Now stop it
JogOff(Yaxis)
```

**See also:**

JogOn()

**JogOn**

Function JogOn(Axis As Integer, Dir as Integer)

This function is used to start an axis jogging in a specified direction at the current default jog speed. Once the axis starts jogging, it will continue until stopped by means if the JogOff() function, execution of a Stop button command, E-stop, or hitting a limit.

**Arguments:**

Axis is an Integer, specifying the axis for which to disable jogging. 0=X, 1=Y, 2=Z, 3=A, etc.

Dir is the direction in which to jog. 0=+ direction, 1=- direction

**Return Value:**

None

**Example:**

```
` Define some constants
Xaxis = 0
Yaxis = 1
Zaxis = 2
Plus = 0
Minus = 1
` Jog the Y axis in the minus direction for one second
JogOn(Yaxis, Minus)
Sleep(1000)
` Now stop it
JogOff(Yaxis)
```

**See also:**

JogOff()

## **LoadFile**

Sub LoadFile(FilePath As String)

This function loads the G-code file specified by FilePath.

### **Arguments:**

FilePath is the full filesystem path to the G-code file to be loaded

### **Return Value:**

None

### **Example:**

```
` Load the roadrunner demo file
LoadFile( "C:\Mach3\Gcode\roadrunner.tap" )
` Now run it
RunFile( )
```

### **See also:**

RunFile()

## **LoadRun**

Sub LoadRun(Filepath As String)

This function loads the G-code file specified by Filepath, then immediately begins execution. This is exactly equivalent to a LoadFile() followed by a RunFile().

### **Arguments:**

Filepath is the String path to the G-code file to be run.

### **Return Value:**

None

### **Example:**

```
` Load and run the roadrunner demo file
LoadRun( "C:\Mach3\Gcode\roadrunner.tap" )
```

### **See also:**

LoadFile(), RunFile(), Filename(), IsLoading()

## **LoadStandardLayout**

Sub LoadStandardLayout()

This function re-loads the current default screenset. In most cases, this will be the currently loaded screenset. This is used primarily to re-load the default screenset when exiting a Wizard.

**Arguments:**

None

**Return Value:**

None

**Example:**

```
` Load default screenset  
LoadStandardLayout()
```

**See also:**

ToggleScreens()

## **LoadTeachFile**

Sub LoadTeachFile()

This function loads the most recent Teach file into Mach3 for execution. The file must first have been opened or created using OpenTeachFile() or AppendTeachFile() during the current Mach3 session.

**Arguments:**

None

**Return Value:**

None

**Example:**

```
` Create a new Teach File in Mach3\Gcode  
MyTeachFile = "TeachMe.nc"  
Err = OpenTeachFile(MyTeachFile)  
If Err <> 0 Then  
    ` Teach file created successfully  
    ` Write some G Code to it  
    Code "G00 X0 Y0"  
    Code "G02 X0 Y0 I-1 J0 F40"  
    ` Close the Teach file  
    CloseTeachFile()  
    ` Now load the teach file for execution  
    LoadTeachFile()  
Else  
    ` OpenTeachFile failed  
    Message "Unable to open Teach File"  
End If
```

**See also:**

OpenTeachFile(), AppendTeachFile(), CloseTeachFile()

## ***LoadWizard***

Sub LoadWizard(WizardName As String)

This function loads and runs the specified wizard. Wizards reside in the Mach3\Addons directory, and each wizard consists of a number of files with a subdirectory of the Addons directory. The WizardName String passed as the argument to LoadWizard must be just the name of the top-level subdirectory for the wizard.

### ***Arguments:***

WizardName is the name of the Wizard to be run, as a String

### ***Return Value:***

None

### ***Example:***

```
` Load and run the "Circular Pocket" Wizard  
LoadWizard("Circular Pocket")
```

### ***See also:***

## ***MachMsg***

Function MachMsg(Prompt As String, Title As String, DialogType As Integer) As Integer

This function display a dialog box with one of several different combinations of buttons, and waits for the user to click one of the buttons. The Title argument String is displayed in the title bar of the dialog. The Message argument String is displayed in the client portion of the dialog, above the buttons. Clicking on any button closes the dialog, and the return value of the function indicates which button the user clicked.

### ***Arguments:***

Message is the String to be displayed in the client area of the dialog, above the button(s)

Title is the String to be displayed in the title bar of the dialog

DialogType is an Integer value which defines which buttons will be displayed on the dialog as follows:

- 0 = OK button
- 1 = OK, Cancel buttons
- 2 = Abort, Retry, Ignore buttons
- 3 = Yes, No, Cancel buttons
- 4 = Yes, No buttons
- 5 = Retry, Cancel buttons
- 6 = Cancel, Try Again, Continue buttons

### ***Return Value:***

Integer value indicating which button the user clicked, as follows:

OK	= 1
Cancel	= 2

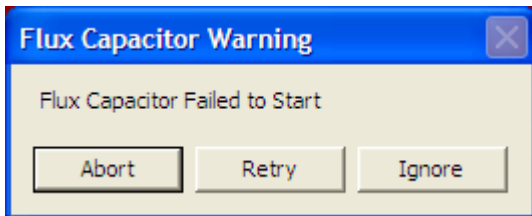
Abort	= 3
Rety	= 4
Ignore	= 5
Yes	= 6
No	= 7
Try Again	= 10
Continue	= 11

**Example:**

```

` Define some constants for MachMsg dialog types
MachMsgTypeOK = 0
MachMsgTypeOKCancel = 1
MachMsgTypeAbortRetryIgnore = 2
MachMsgTypeYesNoCancel = 3
MachMsgTypeYesNo = 4
MachMsgTypeRetryCancel = 5
MachMsgTypeCancelTryAgainContinue = 6
` Define some constants for MachMsg return codes
MachMsgReturnOK = 1
MachMsgReturnCancel = 2
MachMsgReturnAbort = 3
MachMsgReturnRetry = 4
MachMsgReturnIgnore = 5
MachMsgReturnYes = 6
MachMsgReturnNo = 7
MachMsgReturnTryAgain = 10
MachMsgReturnContinue = 11
` Display an Abort/Retry/Ignore dialog
Ret = MachMsg("Flux Capacitor Failed to Start", _
    "Flux Capacitor Warning", MachMsgTypeAbortRetryIgnore)
If Ret = MachMsgReturnAbort Then
    ` Handle Abort here
ElseIf Ret = MachMsgReturnRetry Then
    ` Handle Retry here
ElseIf Ret = MachMsgReturnIgnore Then
    ` Handle Ignore here
End If

```



**See also:**

Message(), Question(), AskTextQuestion(), GetCoord(),

## **MaxX**

Function MaxX() As Double

This function returns the maximum X extent of the currently open G-code file as a Double

### ***Arguments:***

None

### ***Return Value:***

Maximum X extent of currently loaded G-code file as a Double

### ***Example:***

```
` Show user the current program extents  
Message "MinX=" & MinX() & " MaxX=" & MaxX() & _  
" MinY=" & MinY() & " MaxY=" & MaxY()
```

### ***See also:***

MinX(), MinY, MaxY()

## **MaxY**

Function MaxY() As Double

This function returns the maximum Y extent of the currently open G-code file as a Double

### ***Arguments:***

None

### ***Return Value:***

Maximum Y extent of currently loaded G-code file as a Double

### ***Example:***

```
` Show user the current program extents  
Message "MinX=" & MinX() & " MaxX=" & MaxX() & _  
" MinY=" & MinY() & " MaxY=" & MaxY()
```

### ***See also:***

MaxX(), MinY, MaxY()

## **Message**

Sub Message(MessageText As String)

This function displays MessageText on the status line.

**Arguments:**

MessageText is the String text to display on the status line.

**Return Value:**

None

**Example:**

```
` Display the current tool number on the status line  
Message "Current tool is " & GetCurrentTool()
```

**See also:**

Message(), Question(), Ask TextQuestion(), MachMsg(), GetCoord()

## **MinX**

Function MinX() As Double

This function returns the minimum X extent of the currently open G-code file as a Double

**Arguments:**

None

**Return Value:**

Minimum X extent of currently loaded G-code file as a Double

**Example:**

```
` Show user the current program extents  
Message "MinX=" & MinX() & " MaxX=" & MaxX() & _  
" MinY=" & MinY() & " MaxY=" & MaxY()
```

**See also:**

MinX(), MinY, MaxY()

## **MinY**

Function MinY() As Double

This function returns the minimum Y extent of the currently open G-code file as a Double

**Arguments:**

None

**Return Value:**

Minimum Y extent of currently loaded G-code file as a Double

**Example:**

```
` Show user the current program extents  
Message "MinX=" & MinX() & " MaxX=" & MaxX() & _  
" MinY=" & MinY() & " MaxY=" & MaxY()
```



*See also:*

MinX(), MinY, MaxY()

## ***nFmt***

Function nFmt(Val As Double, Digits As Integer) As Double

This function rounds a Double value to the specified number of decimal places

***Arguments:***

Val is the Double value to be rounded

Digits is the number of digits right of the decimal place to round to

***Return Value:***

Rounded value, as a Double

***Example:***

```
` Round 1.23456789 to 4 decimal places  
` Display it - will display as 1.2345  
Message nFmt(1.23456789, 4)
```

*See also:*

## ***NotifyPlugins***

Sub NotifyPlugins(Event As Integer)

This function invokes the MyNotify method of all currently loaded Plug-ins, passing Event as the argument.

***Arguments:***

Event is an Integer value that will be passed as the argument to the MyNotify methods of all loaded plug-ins.

***Return Value:***

None

***Example:***

*See also:*

## ***OpenDigFile***

Sub OpenDigFile()

This function opens a digitization log file. When OpenDigFile is executed a FileOpen dialog will be displayed, and the user can select an existing file, or enter a new file name, to which digitization (probing) data points will be written. Once opened, G31 commands will cause the probe trigger position to be written to the digitization file. When digitization is complete, CloseDigFile() can be used to close the file.

**Arguments:**

None

**Return Value:**

None

**Example:**

**See also:**

## **OpenTeachFile**

Function OpenTeachFile (Filename As String) As Integer

This function re-opens an existing Teach file at Gcode\Filename in the Mach3 directory, and appends any commands subsequently executed via MDI or Code() to that file, until CloseTeachFile() is executed.

**Arguments:**

F

**Return Value:**

A

**Example:**

```
` Create a new Teach File in Mach3\Gcode
MyTeachFile = "TeachMe.nc"
Err = OpenTeachFile(MyTeachFile)
If Err <> 0 Then
    ` Teach file created successfully
    ` Write some G Code to it
    Code "G00 X0 Y0"
    Code "G02 X0 Y0 I-1 J0 F40"
    ` Close the Teach file
    CloseTeachFile()
    ` Now load the teach file for execution
    LoadTeachFile()
Else
    ` OpenTeachFile failed
    Message "Unable to open Teach File"
End If
```

**See also:**

AppendTeachFile(), CloseTeachFile()

## **Param1**

Function Param1() As Double

This function returns the value of the P parameter passed to an M-macro.

### **Arguments:**

None

### **Return Value:**

Double value passed a P parameter to M-macro

### **Example:**

```
` This macro expects three arguments: P, Q, & R
` If put into an M-macro, and invoked via MDI, it will
` display the argument values on the status line
` For example, if using M1200:
`   M1200 P1.234 Q2.345 R3.456
` Executing the above line to MDI will display:
`   P=1.234 Q=2.345 R=3.456
Parg = Param1()
Qarg = Param2()
Rarg = Param3()
Message "P=" & Parg & "Q=" & Qarg & "R=" & Rarg
```

### **See also:**

Param2(), Param3()

## **Param2**

Function Param2() As Double

This function returns the value of the Q parameter passed to an M-macro.

### **Arguments:**

None

### **Return Value:**

Double value passed a Q parameter to M-macro

### **Example:**

```
` This macro expects three arguments: P, Q, & R
` If put into an M-macro, and invoked via MDI, it will
` display the argument values on the status line
` For example, if using M1200:
`   M1200 P1.234 Q2.345 R3.456
```

```

` Executing the above line to MDI will display:
`      P=1.234 Q=2.345 R=3.456
Parg = Param1()
Qarg = Param2()
Rarg = Param3()
Message "P=" & Parg & "Q=" & Qarg & "R=" & Rarg

```

**See also:**

Param1(), Param3()

## **Param3**

Function Param3() As Double

This function returns the value of the R parameter passed to an M-macro.

**Arguments:**

None

**Return Value:**

Double value passed a R parameter to M-macro

**Example:**

```

` This macro expects three arguments: P, Q, & R
` If put into an M-macro, and invoked via MDI, it will
` display the argument values on the status line
` For example, if using M1200:
`      M1200 P1.234 Q2.345 R3.456
` Executing the above line to MDI will display:
`      P=1.234 Q=2.345 R=3.456
Parg = Param1()
Qarg = Param2()
Rarg = Param3()
Message "P=" & Parg & "Q=" & Qarg & "R=" & Rarg

```

**See also:**

Param1(), Param2()

## **PlayWave**

Sub PlayWave(Filename As String)

This function plays a .WAV file through the PC's audio system. Filename gives the name of the WAV file to be played. All WAV files must reside in the Mach3 directory.

**Arguments:**

Filepath is the String filepath to the WAV file to be played

***Return Value:***

None

***Example:***

```
PlayWave("C:\WINDOWS\Media\Windows XP Startup.wav")
```

***See also:***

## ***PutPortByte***

Sub PutPortByte(PortAddr As Integer, Value As Integer)

This function writes the 8-bit data given by Value to the 8-bit PC I/O port whose address is given by PortAddr. This function can be used for writing hardware devices not directly supported by Mach3.

***Arguments:***

PortAddr is the Integer address of the port to be written

Value is the Integer value to be written. Only the 8 LSBs are written.

***Return Value:***

None

***Example:***

```
` Our port address
PortAddr = 1016 ` 0x3f8
PortData = 154 ` 0xa5
` Write 0xa5 to data register of parallel port at 0x3f8
PortData = PutPortByte(PortAddr, PortData)
```

***See also:***

GetPortByte()

## ***Question***

Function Question(Prompt As String) As Double

This function displays a dialog box with an OK button, and a text box into which the user can enter a numeric value. The Prompt string is displayed above the test box. This value is returned as a Double. This can be used for getting a single numeric value from the user.

***Arguments:***

Prompt is the prompt string displayed in the Question dialog.

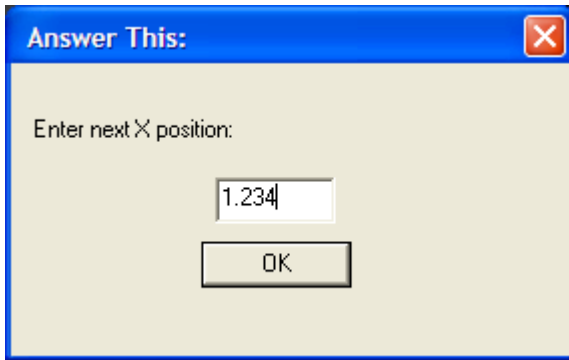
***Return Value:***

Double value entered into text box by user

***Example:***

```
` Get next X position from user
NextPos = Question("Enter next X position:")
```

```
` Go there  
Code "G00 X" & NextPos
```



*See also:*

Message(), Question(), AskTextQuestion(), MachMsg(), GetCoord()

## **QueueDepth**

Function QueueDepth() As Integer

This function returns the current number of entries in the trajectory planning queue.

*Arguments:*

None

*Return Value:*

Number of trajectory planner queue entries as an Integer

*Example:*

```
` Show current queue depth  
Message "Queue depth = " & QueueDepth()
```

*See also:*

## **Random**

Function Random() As Double

This function returns a pseudo-random number between 0 and 1. This can be used in place of the VB rnd() function.

*Arguments:*

None

*Return Value:*

Next pseudo-random number in sequence, as a Double

**Example:**

```
` Get next random number
NextRand = Random( )
```

**See also:**

## **RefCombination**

Sub RefCombination(Axes As Integer)

This function allows any combination of axes to be simultaneously referenced (homed). Which axes will be referenced is determined by the Axes argument, which is a bit-mapped variable, with the bits mapped as defined below.

**Arguments:**

Axes is a bit-mapped value that defines which axes are to be referenced. The value of Axes can be calculated by adding the values corresponding to the individual axes to be referenced. The axis values are:

X = 1  
Y = 2  
Z = 4  
A = 8  
B = 16  
C = 32

So, for example, to reference the X, Z and B axes,  $\text{Axes} = 1 + 4 + 16 = 21$ .

**Return Value:**

None

**Example:**

```
` Define some constants
RefX = 1
RefY = 2
RefZ = 4
RefA = 8
RefB = 16
RefC = 32
` Reference Y, Z and C axes
RefCombination(RefY + RefZ + RefC)
```

**See also:**

VerifyAxis(), SingleVerify(), SingleVerifyReport()

## **ResetAxisSwap**

Sub ResetAxisSwap()

This function un-does the effect of a preceding SwapAxis(). Note that this is the only way to undo a Swap, and it is illegal to perform two SwapAxis() calls without an intervening ResetAxisSwap().

### **Arguments:**

None

### **Return Value:**

None

### **Example:**

```
` Define some constants
Xaxis = 0
Yaxis = 1
Zaxis = 2
` Swap X and Y axes
SwapAxis(Xaxis, Yaxis)
` Do something with swapped axes here...
` Undo the SwapAxis
ResetAxisSwap()
` Now Swap Y and Z axes
SwapAxis(Yaxis, Zaxis)
```

### **See also:**

SwapAxis()

## **ResetTHC**

Sub ResetTHC()

This function resets the Torch Height. This function is identical to ZeroTHC(), except if a program is running, it does nothing.

### **Arguments:**

None

### **Return Value:**

None

### **Example:**

```
` Reset THC
ResetTHC()
```

### **See also:**

THCOn(), THCOff()



## ***RetractMode***

Function RetractMode() As Integer

This function returns the current Peck Drilling retract mode. If a G98 was last run, it will return a 0. If a G99 was most recently run, it will return a 1.

### ***Arguments:***

None

### ***Return Value:***

Integer value indicating which peck cycle was last run. 0=G98, 1=G99.

### ***Example:***

```
' Run a G98
Code "G98 Z-0.5 R0.1 F10"
' Wait for it to complete
While IsMoving()
    Sleep 100
Wend
' Now show retract mode - should show 0
MsgBox("RetractMode = " & RetractMode())
Code "Hit CycleStart to continue..."
' Run a G99
Code "G99 Z-0.5 R0.1 F10"
' Wait for it to complete
While IsMoving()
    Sleep 100
Wend
' Now show retract mode - should show 1
MsgBox("RetractMode = " & RetractMode())
Code "Hit CycleStart to continue..."
```

### ***See also:***

## ***roun***

Function roun(Val As Double) As Double

This function rounds the Double value specified by Val to four decimal places.

### ***Arguments:***

Val is the Double value to be rounded

### ***Return Value:***

Double value of Val rounded to four decimal places

**Example:**

```
` Round 1.23456789
Message "1.23456789 rounds to " & roun(1.23456789)
```

**See also:**

Round()

## **RunFile**

Sub RunFile()

This function begins execution of the currently loaded G-code program, if any. This is functionally identical to DoOEMButton(CYCLESTART)

**Arguments:**

None

**Return Value:**

None

**Example:**

```
` Load our G-code file
LoadFile("MyGCode.tap")
` Run it
RunFile()
```

**See also:**

FileName(), LoadFile(), LoadRun(), IsLoading(), DoOEMButton()

## **SaveWizard**

Sub SaveWizard()

This function can be used within a wizard to save the user-entered parameter value settings for the wizard, so the same values will be loaded the next time the wizard is invoked.

**Arguments:**

None

**Return Value:**

None

**Example:**

```
` Save current Wizard parameter values
SaveWizard()
```

**See also:**

## **SetButtonText**

Sub SetButtonText(Text As String)

This function allows the text of an on-screen button to be changed by that buttons associated button script. This can be used to change the text on a button based on the state of a variable or mode.

### **Arguments:**

Text is a String that specifies the new text to be displayed on the button.

### **Return Value:**

None

### **Example:**

```
` Example Spindle On/Off Toggle Button
SpindleCWLED = 11
If GetOEMLLED(SpindleCWLED) Then
    ` Spindle is on, so turn it off
    DoSpinStop()
    SetButtonText("Spindle On")
Else
    ` Spindle is off, so turn it on
    DoSpinCW()
    SetButtonText("Spindle Off")
End If
```

### **See also:**

## **SetCurrentTool**

Sub SetCurrentTool(ToolNum As Integer)

This function sets the current tool number to ToolNum. This is typically used in the M6Start script to make the selected tool the current tool.

### **Arguments:**

ToolNum is an Integer tool number, from 1 to 255.

### **Return Value:**

None

### **Example:**

```
` Typical M6Start script
` Get selected tool
NewTool = GetSelectedTool()
` Make it the current tool
SetCurrentTool(NewTool)
```

### **See also:**

GetCurrentTool(), GetSelectedTool()

## **SetDRO**

Function SetDRO(DRONum As Integer, DROVal As Double)

This legacy function sets the DRO number specified by DRONum plus 800 to the value specified by DROVal. SetDRO(DRONum, DROVal) is exactly equivalent to SetOEMDRO(DRONum + 800, DROVal). The only difference between this function and SetOEMDRO() is that this function adds 800 to DRONum to calculate the actual DRO number to be set. This function is deprecated, and its use is *strongly* discouraged. Use of the GetOEMDRO/SetOEMDRO and GetUserDRO/SetUserDRO functions is strongly recommended.

### **Arguments:**

DRONum is the Integer OEM DRO number to read, minus 800. Valid DRONums are from 0 to 200, which correspond to OEM DRO numbers 800 to 1000.

DROVal is the Double value to which the specified DRO will be set.

### **Return Value:**

None

### **Example:**

```
` Define the axes
Dim Xaxis As Integer
Xaxis = 800
Dim Yaxis As Integer
Yaxis = 801
Dim Zaxis As Integer
Zaxis = 802

` Write 1.2345 to Z axis DRO using SetDRO
SetOEMDRO(Zaxis - 800, 1.2345)
` Show the user the Z Axis DRO value, using GetOEMDRO()
MsgBox "Using SetDRO() Z Axis DRO reads: " & GetOEMDRO(Zaxis )
` Write 2.3456 to Z axis DRO using SetOEMDRO
SetOEMDRO(Zaxis, 2.3456)
` Show the user the Z Axis DRO value, using GetOEMDRO()
MsgBox "Using SetOEMDRO() Z Axis DRO reads: " & GetOEMDRO(Zaxis )
```

### **See also:**

SetOEMDRO(), GetOEMDRO(), SetUserDRO(), GetUserDRO(), GetDRO()

## **SetFeedRate**

Sub SetFeedRate(FeedRate As Double)

This function sets the feedrate. Note that FeedRate is specified in units per second, rather than units per minute.

**Arguments:**

FeedRate specified in units/second, as a Double

**Return Value:**

None

**Example:**

```
' Set the feedrate to 123.456 inches/minute
SetFeedRate(123.456 / 60)
' Get the current feedrate, in inches/minute, and display it
CurrentFeedrate = FeedRate()
' Display it on the status line
Message "Current feedrate = " & CurrentFeedrate
```

**See also:**

FeedRate()

## SetFormula

Sub SetFormula(Formula As String, Axis As Integer)

This function sets one of the axis formulas, accessible by FunctionCfg's->Formulas. These allow motion for axes to be defined by formulas involving the positions of other axes, using algebraic and trigonometric functions. Note that for Formulas to take effect, you must check the "Formulas enabled" checkbox in FunctionCfg's->Formulas.

**Arguments:**

Formula is a String that defines the algebraic/trigonometric function to be used to calculate axis position.

Axis is an Integer value that defines which axis the Formula is to be applied to

**Return Value:**

None

**Example:**

```
' Define some constants
Xaxis = 0
Yaxis = 1
Zaxis = 2
' Define the axis formulas such that a programmed linear move
' Y causes a sinusoidal move in X
' Z axis should move normally
SetFormula("Z", Zaxis)
SetFormula("Y", Yaxis)
SetFormula("Sin(y)", Xaxis)
```

**See also:**

## **SetIJMode**

Sub SetIJMode(IJMode As Integer)

This function sets the IJ mode. IJMode = 0 sets absolute mode, while IJMode = 1 sets incremental mode.

### **Arguments:**

IJMode is the Integer mode to select. 0=Absolute, 1=Incremental

### **Return Value:**

None

### **Example:**

```
` Define some constants
IJAbsolute = 0
IJIncremental = 1
` Set IJMode to incremental
SetIJMode(IJIncremental)
```

### **See also:**

GetIJMode()

## **SetMachZero**

Sub SetMachZero(Axis As Integer)

This function zeroes the machine position of the specified axis to the current position.

### **Arguments:**

Axis is an Integer value identifying the axis to be zeroed. 0=X, 1=Y, 2=Z, 3=W, etc.

### **Return Value:**

None

### **Example:**

```
` Define some constants
Xaxis = 0
Yaxis = 1
Zaxis = 2
` Set machine zero for Y axis to current position
SetMachZero(Yaxis)
```

### **See also:**

## **SetOEMDRO**

Sub SetOEMDRO(DRNum As Integer, DROVal as Double)

This function sets the OEM DRO specified by DRNum to the value specified by DROVal.

### ***Arguments:***

DRNum is an Integer value or expression that evaluates to one of the OEM DRO numbers.

### ***Return Value:***

None

### ***Example:***

```
` Define some constants
XaxisDRO = 0
YaxisDRO = 1
ZaxisDRO = 2
` Set Z Axis DRO to 1.234
SetOEMDRO(ZaxisDRO, 1.234)
```

### ***See also:***

GetOEMDRO(), SetUserDRO(), GetUserDRO()

## **SetPage**

Sub SetPage(PageNum As Integer)

This function switches the current display page to the one specified by PageNum.

### ***Arguments:***

PageNum is the Integer number of the display page to switch to.

### ***Return Value:***

None

### ***Example:***

```
` For 1024.set, change to Diagnostics page
SetPage(5)
```

### ***See also:***

GetPage()

## **SetParam**

Sub SetParam(ParamName As String, ParamVal As Double)

This function allows a number of Mach3 internal parameters (not to be confused with G-code parameters) to be set. Each Mach3 parameter is identified by name. The parameter whose name is given by ParamName is set to the value given by ParamVal. Valid parameters are:

<b>Parameter Name</b>	<b>Description</b>
RPMOverride	When this parameter is set to 1, the True Spindle Speed DRO (OEM DRO 39) is made writeable, over-riding Mach3s normal calculation of True Spindle Speed.
Xscale	X axis scale factor
YScale	Y axis scale factor
ZScale	Z axis scale factor
AScale	A axis scale factor
BScale	B axis scale factor
CScale	C axis scale factor
FeedRate	Feedrate
Units	Current units (inch/mm). 0 = mm, 1 = inch
StepsPerAxisX	X axis steps per unit
StepsPerAxisY	Y axis steps per unit
StepsPerAxisZ	Z axis steps per unit
StepsPerAxisA	A axis steps per unit
StepsPerAxisB	B axis steps per unit
StepsPerAxisC	C axis steps per unit
VelocitiesX	X axis maximum velocity, from motor tuning, in units/second
VelocitiesY	Y axis maximum velocity, from motor tuning, in units/second
VelocitiesZ	Z axis maximum velocity, from motor tuning, in units/second
VelocitiesA	A axis maximum velocity, from motor tuning, in units/second
VelocitiesB	B axis maximum velocity, from motor tuning, in units/second
VelocitiesC	C axis maximum velocity, from motor tuning, in units/second
AccelerationX	X axis maximum acceleration, from motor tuning
AccelerationY	Y axis maximum acceleration, from motor tuning
AccelerationZ	Z axis maximum acceleration, from motor tuning
AccelerationA	A axis maximum acceleration, from motor tuning
AccelerationB	B axis maximum acceleration, from motor tuning
AccelerationC	C axis maximum acceleration, from motor tuning
SpindleSpeed	Should modify Spindle Speed, but does not work on all versions. Use SetSpinSpeed() instead.
ZInhibitOn	Z Inhibit Enable. 0=Z inhibit disabled, 1=Z inhibit enabled. When Z inhibit is enabled, the Z axis will not be allowed to move below the depth specified by the ZInhibitDepth parameter.
ZInhibitDepth	Z Inhibit Depth. When Z inhibit is enabled, the Z axis will not be allowed to move below the depth specified by the ZInhibitDepth parameter.
SafeZ	SafeZ height
Boundry	Toolpath Boundaries display enable. 0=>disable boundaries display, 1=>enable boundaries display
XRefPer	X axis homing speed, as % of rapid speed
YRefPer	Y axis homing speed, as % of rapid speed



ZRefPer	Z axis homing speed, as % of rapid speed
ARefPer	A axis homing speed, as % of rapid speed
BRefPer	B axis homing speed, as % of rapid speed
CRefPer	C axis homing speed, as % of rapid speed
ToolPathLock	Toolpath Lock enable. 0=>Toolpath unlocked, 1=>Toolpath locked. When the toolpath is locked, it cannot be moved, scaled or rotated.
PrepMove	Preparatory Move Dialog Inhibit. 0=>Allow showing of Preparatory Move dialogs, 1=>Inhibit showing of Preparatory Move dialogs
AutoToolChange	??? – Indicates auto toolchanger in use?
ADirActive	??? – Reverse direction of A axis?

**Arguments:**

ParamName is the String name of the parameter to be set. This must be one of the above names.  
ParamVal is the Double value to which the specified parameter will be set.

**Return Value:**

None

**Example:**

```
` Get the new scale factor from the user
ScaleFactor = Question "Enter new scale factor:"
` Set the new scale factor for X/Y/Z
SetParam("Xscale", ScaleFactor)
SetParam("Yscale", ScaleFactor)
SetParam("Zscale", ScaleFactor)
```

**See also:**

GetParam()

## SetPulley

Sub SetPulley(Pulley As Integer)

This function sets the current spindle pulley number. This allows Mach3 to properly scale the spindle speed output based on the current pulley and commanded spindle speed. Pulley ratios and allowable speed ranges are configured in Config->SpindlePulleys

**Arguments:**

Pulley is an Integer pulley number, from 1 to 15

**Return Value:**

None

**Example:**

```
` Prompt the user for new pulley setting
NewPulley = Question "Enter new pulley number:"
` Tell Mach3
SetPulley(NewPulley)
```

*See also:*

## **SetSafeZ**

Sub SetSafeZ(SafeZ As Double)

This function sets a new value for SafeZ. Note that SafeZ must be enabled in Config->SafeZConfig

### **Arguments:**

SafeZ is the Double value to set SafeZ to

### **Return Value:**

None

### **Example:**

```
` Change SafeZ to +2.000
SetSafeZ(2.000)
```

*See also:*

## **SetScale**

Sub SetScale(Axis As Integer, Scale As Double)

This function sets the scale factor for axis Axis to the value given by Scale.

### **Arguments:**

Axis is the Integer Axis. 0=X, 1=Y, 2=Z, 3=A, etc.

Scale is the Double Scale factor

### **Return Value:**

None

### **Example:**

```
` Define some constants
Xaxis = 0
Yaxis = 1
Zaxis = 2
` Get the curent axis scale factors
OldXScaleFactor = GetScale(Xaxis)
OldYScaleFactor = GetScale(Yaxis)
OldZScaleFactor = GetScale(Zaxis)
` Get the new scale factor from the user
ScaleFactor = Question("Enter new scale factor:")
` Set new scale factors for X/Y/Z
SetScale(Xaxis, ScaleFactor)
SetScale(Yaxis, ScaleFactor)
```

```
SetScale(Zaxis, ScaleFactor)
```

**See also:**

```
GetScale()
```

## **SetSpinSpeed**

Sub SetSpinSpeed(RPM As integer)

This function sets the spindle speed, in RPM, exactly as the “S” word in G-code does.

**Arguments:**

RPM is the Integer spindle speed desired.

**Return Value:**

None

**Example:**

```
` Turn spindle on at 2500 RPM
SetSpinSpeed(2500)
DoSpinCW()
```

**See also:**

```
DoSpinCW(), DoSpinCCW(), DoSpinStop()
```

## **SetTicker**

Sub SetTicker(TickerNum As Integer, TickerText As String)

This function loads the text given by TickerText into the ticker specified by TickerNum. A Ticker is a scrolling text label, with its default text set to “Tickernn”, where “nn” is a number between 0 and 255. A long message can be put in a Ticker, and it will scroll continuously so the entire message is visible, even if the label is shorter than the text.

**Arguments:**

TickerNum is the Integer number of the ticker to be written

TickerText is the String text to be written to ticker TickerNum

**Return Value:**

None

**Example:**

```
` Write the current file path to Ticker 25
SetTicker(25, FileName())
```

**See also:**

```
SetUserLabel(), Message()
```

## **SetTimer**

Function SetTimer(TimerNum As Integer)

This function clears the specified timer. Mach3 provides 25 timers, numbered 0 to 24, which can be used for timing in VB scripts. To time an event, first clear the timer using SetTimer(), then use GetTimer() to read the timer. Note that this function works only with the parallel port driver, and support for this function may be removed without notice in a future release.

### **Arguments:**

TimerNum is an Integer timer number, which must be between 0 and 24.

### **Return Value:**

None

### **Example:**

```
` Clear timer 15
SetTimer(15)
` Wait for OEM Trigger 10 to go active
While IsActive(OEMTRIG10) = False Then
    Sleep 10
Wend
` See how long it took
Message "OEMTRIG10 active after " & GetTimer(15) & " seconds"
```

### **See also:**

GetTimer()

## **SetToolParam**

Sub SetToolParam(ToolNum As Integer, ParamNum As Integer, ParamVal As Double)

This function allows any tool parameter for any tool to be set. ToolNum is the number of the tool whose parameters are being set, and can be from 1 to 255. ParamNum is a parameter number, defined as follows:

For Mach3Mill:

- 1 = Diameter
- 2 = Z Offset
- 3 = X Wear
- 4 = Z Wear

For Mach3Turn:

- 1 = Tip Type
- 2 = Tool Radius
- 3 = X Offset
- 4 = Z Offset
- 5 = X Wear
- 6 = Z Wear

7 = Turret Angle

**Arguments:**

ToolNum is an Integer tool number, and must be between 1 and 255.

ParamNum is an Integer parameter number, defined as described above.

**Return Value:**

None

**Example:**

```
` Define some constants
DiameterParam = 1
ZoffsetParam = 2
Xwear Param = 3
ZwearParam = 4
` Define tool #23
SetToolParam(23, DiameterParam, 0.2500)      ` Set tool diameter
= 1/4"
SetToolParam(23, ZoffsetParam, 1.2500)  ` Set length offset =
1.25"
SetToolParam(23, XwearParam, 0.0005)      ` Set X wear = 0.0005"
SetToolParam(23, ZwearParam, 0.0013)      ` Set Z wear = 0.0013"
```

**See also:**

GetToolParam(), GetToolDesc()

## SetToolX

Function SetToolX(Pos As Double)

This function sets the X axis DRO to the value given by Pos, then displays the message "Work Offset Shifted" on the status line. Note that with the exception of this message, SetToolX() is functionally identical to using SetOEMDRO to modify the X axis DRO value directly.

**Arguments:**

Pos is the Double position to which the X axis DRO will be set.

**Return Value:**

None

**Example:**

```
` Prompt the user to zero the X axis
Message "Move X axis to zero position.  Press OK when ready..."
` Zero the X axis DRO
SetToolX(0.0000)
```

**See also:**

SetToolZ()

## **SetToolZ**

Function SetToolZ(Pos As Double)

This function sets the Z axis DRO to the value given by Pos, then displays the message “Work Offset Shifted” on the status line. Note that with the exception of this message, SetToolZ() is functionally identical to using SetOEMDRO to modify the Z axis DRO value directly.

### ***Arguments:***

Pos is the Double position to which the Z axis DRO will be set.

### ***Return Value:***

None

### ***Example:***

```
` Prompt the user to zero the Z axis
Message "Move Z axis to zero position.  Press OK when ready..."
` Zero the Z axis DRO
SetToolZ(0.0000)
```

### ***See also:***

SetToolX()

## **SetTriggerMacro**

Sub SetTriggerMacro(MacroNum As Integer)

This function allows an M-macro to be associated with OEM code 301. This can be used to cause one of the OEM Trigger signals to automatically run a macro when asserted. To do this, the OEM Trigger signal must first be configured in Config->Ports&Pins->InputSignals. Then the OEM Trigger must be associated with OEM code 301 in Config->SystemHotKeys. Finally, SetTriggerMacro must be used to define which M-macro will be executed when OEM button code 301 is executed.

### ***Arguments:***

MacroNum is the number of the M-macro to be run when OEM button code 301 is executed.

### ***Return Value:***

None

### ***Example:***

```
` Assume OEMTRIGGER5 is assigned to OEM code 301
` Assign the M1025 macro to OEM Code 301
SetTriggerMacro(1025)
` Now, when OEMTRIGGER5 is driven to its active level, M1025.mls
will be executed
```

### ***See also:***

## ***SetUserDRO***

Sub SetUserDRO(DRONum As Integer, DROVal As Double)

This function sets the value of User DRO DRONum to DROVal.

### ***Arguments:***

DRONum is the Integer User DRO number to be set. Valide User DRO numbers range from 1000-2254.

DROVal is the Double value to which the User DRO will be set

### ***Return Value:***

None

### ***Example:***

```
` Define some constants
MyWidgetDRO = 1125
` Set MyWidgetDRO to 1.234
SetUserDRO(MyWidgetDRO, 1.234)
...
` Get current value of MyWidgetDRO
MyDROVal = GetUserDRO(MyWidgetDRO)
```

### ***See also:***

GetUserDRO(), SetOEMDRO(), GetOEMDRO()

## ***SetUserLabel***

Sub SetUserLabel(LabelNum As Integer, LabelText As String)

This function allows the user to change an on-screen “User” labels from VB, rather than by using a screenset editor. “User” lables are those that are created in the screen designer with the default text containing the String “UserLabel” followed by one or more digits.

### ***Arguments:***

LabelNum is the numeric portion of the user label default text. LabelNum must be between 0 and 255.

LabelText is the text to be placed into the label.

### ***Return Value:***

None

### ***Example:***

```
` Change the text in UserLabel25
SetUserLabel(25, "This is Label 25")
...
` Retrieve the text from UserLabel25
LabelText = GetUserLabel(25)
```

*See also:*

## **SetUserLED**

Sub SetUserLED(LEDNum As Integer, State As Integer)

This function allows the state of a User LED to be set or cleared. User LEDs are numbered from

### **Arguments:**

LEDNum is the User LED to be set, and must be in the range of 1000 to 2254

State is the new state of the User LED. 0 indicates the LED is off (unlit), 1 indicates the LED is on (lit).

### **Return Value:**

None

### **Example:**

```
` Define some constants
FluxCapacitorControl = OUTPUT1  ` Output that controls the flux
capacitor
FluxCapacitorLED = 1234  ` LED that indicates flux capacitor is
active
` Turn on the Flux capacitor
ActivateSignal(FluxCapacitorControl)
` Turn on the Flux Capacitor LED for the operator
SetUserLED(FluxCapacitorLED, 1)
...
` Is the Flux Capacitor on?
FluxCapacitorOn = GetUserLED(FluxCapacitorLED)
```

### **See also:**

GetUserLED(), SetOEMLED(), GetOEMLED()

## **SetVar**

Sub SetVar(VarNum As Integer, Val As Double)

This function sets the Mach variable specified by VarNum to the value given by Val. Mach variables are accessible both to VB scripts, using the SetVar() and GetVar() functions, as well as G-code programs, using the #nnnn syntax.

### **Arguments:**

VarNum is Integer the number of the Mach variable to be set.

Val is the Double value to which the variable will be set

### **Return Value:**



None

**Example:**

```
` Set a variable 1234 to our target position of 2.3456
SetVar(1234, 2.3456)
` Now move X to our target position
Code "G0 X #1234"
```

**See also:**

GetVar()

## **SingleVerify**

Sub SingleVerify(Axis As Integer)

This function performs a position verify on a single axis by homing that axis, and, once homed, zeroing the work offset, then returning to the initial offset. This is commonly used at the beginning of a program to ensure the fixture offset is set properly, or after a crash to restore the correct work offset.

**Arguments:**

Axis in an Integer value that specifies the first axis to verify. 0=X, 1=Y, 2=Z, 3=A, etc.

**Return Value:**

None

**Example:**

```
Xaxis = 0
Yaxis = 1
Zaxis = 2
` Verify Z axis work offset is properly set
SingleVerify(Zaxis)
```

**See also:**

SingleVerifyReport(), VerifyAxis(), RefCombination()

## **SingleVerifyReport**

Function SingleVerifyReport(Axis As Integer)

This function performs a position verify on a single axis by homing that axis, and, once homed, zeroing the work offset, then returning to the initial offset. When complete, a message is displayed showing how far each axis was offset from its correct home position. This function is identical to the SingleVerify function except for this message display. This is commonly used at the beginning of a program to ensure the fixture offset is set properly, or after a crash to restore the correct work offset.

**Arguments:**

Axis in an Integer value that specifies the first axis to verify. 0=X, 1=Y, 2=Z, 3=A, etc.

**Return Value:**

None

**Example:**

```
Xaxis = 0
Yaxis = 1
Zaxis = 2
` Verify Z axis work offset is properly set, and report error, if
any
SingleVerifyReport(Zaxis)
```

**See also:**

SingleVerifyReport(), VerifyAxis(), RefCombination()

## **Sleep**

Sub Sleep(Time As Integer)

This function causes the VB program to pause for the specified period of time, in mSec. During this time, other processes can have access to the CPU. A Sleep() call should always be inserted in any loop which might remain active for any period of time, for example the While loop used to wait for a move command to complete.

**Arguments:**

Time, an Integer value expressing the sleep time in milliseconds.

**Return Value:**

None

**Example:**

```
` Move to zero position
Code "G0 X0.000 Y0.000"
` Wait for move to complete
While IsMoving()
    ` Let other processes have CPU while we wait
    Sleep 100
Wend
```

**See also:**

## **Speak**

Sub Speak(TextToSpeak As String)

This function uses speech synthesis to “speak” the String argument. Note that your PC must have a working audio system, and speech must first be enabled by checking the “Allow Speech” checkbox in Config->GeneralConfig->GeneralConfiguration.

**Arguments:**

TextToSpeak is a String to be spoken

**Return Value:**

None

**Example:**

```
` Tell the user to load the next workpiece
Speak("Please load next workpiece.  Hit CycleStart When Ready")
Code "M00"
```

**See also:****StartTHC**

Sub StartTHC()

This function turns on torch height control. It is functionally identical to THCon().

**Arguments:**

None

**Return Value:**

None

**Example:**

```
StartTHC() ` Turn on torch height control
...         ` Do some cutting here
EndTHC()   ` Turn off torch height control
```

**See also:**

THCon(), THCOff(), EndTHC(), ZeroTHC(), ResetTHC()

**StraightFeed**

Sub StraightFeed(X As Double, Y As Double, Z As Double, A As Double, B As Double, C As Double)

This function performs a feedrate move to the specified position. Note that all axis positions must be specified. This is exactly equivalent to Code "G1 Xn.nnn Yn.nnn Zn.nnn An.nnn Bn.nnn Cn.nnn".

**Arguments:**

X is a Double specifying the target position for the X axis

Y is a Double specifying the target position for the Y axis

Z is a Double specifying the target position for the Z axis

A is a Double specifying the target position for the A axis

B is a Double specifying the target position for the B axis

C is a Double specifying the target position for the C axis

**Return Value:**

None

**Example:**

```
` Send all axes to zero position at current feedrate  
StraightTraverse(0.000, 0.000, 0.000, 0.000, 0.000, 0.000)
```

**See also:**

StraightTraverse(), Code()

## **StraightTraverse**

Sub StraightTraverse(X As Double, Y As Double, Z As Double, A As Double, B As Double, C As Double)

This function performs a rapid move to the specified position. Note that all axis positions must be specified. This is exactly equivalent to Code “G0 Xn.nnn Yn.nnn Zn.nnn An.nnn Bn.nnn Cn.nnn”.

**Arguments:**

X is a Double specifying the target position for the X axis

Y is a Double specifying the target position for the Y axis

Z is a Double specifying the target position for the Z axis

A is a Double specifying the target position for the A axis

B is a Double specifying the target position for the B axis

C is a Double specifying the target position for the C axis

**Return Value:**

None

**Example:**

```
` Send all axes to zero position  
StraightTraverse(0.000, 0.000, 0.000, 0.000, 0.000, 0.000)
```

**See also:**

StraightFeed(), Code()

## **SwapAxis**

Sub SwapAxis(FirstAxis As Integer, SecondAxis As Integer)

This function swaps the STEP and DIR pins for the two specified axes. This has precisely the same effect as changing the pin settings in Config->Ports&Pins. Note that no other axis parameters or settings are changed. If the two specified axes have different acceleration and velocity settings, unreliable operation will likely result. Note also that if you exit Mach3 while the swap is in effect, the swapped pins will be written to the XML configuration file, and the swap will still be in effect the next time you start Mach3. It is illegal to perform two consecutive swaps, without first executing a ResetAxisSwap(). Doing so will likely result in incorrect operation.

**Arguments:**

FirstAxis in an Integer value that specifies the first axis to swap. 0=X, 1=Y, 2=Z, etc.

SecondAxis in an Integer value that specifies the second axis to swap. 0=X, 1=Y, 2=Z, etc.

**Return Value:**

None

**Example:**

```
Xaxis = 0
Yaxis = 1
Code "G0 X0 Y0"
` Cut a 45 degree diagonal line from lower right to upper left
Code "G1 X-1 Y1"
Code "G0 X0 Y0"
` Now swap X and Y axes
SwapAxis(Xaxis, Yaxis)
` Cut a 45 degree diagonal line from upper left to lower right
Code "G1 X-1 Y1"
Code "G0 X0 Y0"
```

**See also:**

ResetAxisSwap()

## SystemWaitFor

Sub SystemWaitFor(Signal As Integer)

This function will halt macro execution until the specified signal becomes active. This can be used to synchronize macro execution to an external event, such as a switch closure. If the signal is defined in Config->Ports&Pins as ActiveHigh, it will wait until the signal is driven to a logic High level, otherwise it will wait until the signal is driven to a logic Low level.

**Arguments:**

SignalID must be one of the pre-defined Mach3 VB output signal constants (see VB Constants), or other value or expression that evaluates to one of those values.

**Return Value:**

None

**Example:**

```
` INPUT1 indicates adequate coolant level
If IsActive(INPUT1) Then
    ` Uh-oh! Coolant is low - Tell the user
    Message "Operation Halted Due To Low Coolant! Please Add
Coolant..."
    ` Now wait until he tops it off
    SystemWaitFor(INPUT1)
End If
```

*See also:*

## **THCOff**

Sub THCOff()

This function turns off the torch height control.

***Arguments:***

None

***Return Value:***

None

***Example:***

```
` Turn off THC  
THCOff( )
```

***See also:***

THCOn(), ZeroTHC()

## **THCOn**

Sub THCOn()

This function turns on the torch height control.

***Arguments:***

None

***Return Value:***

None

***Example:***

```
` Turn on THC  
THCOn( )
```

***See also:***

THCOff(), ZeroTHC()

## **ToggleScreens**

Sub ToggleScreens()

This function toggles the active screenset between the “complex” and “simple” (.set and .sset) screensets of the same base name. For example, if the current screenset is the default “1024.set”, this function will

load the “1024.sset” screenset, and vice-versa. This function can be used to toggle between any two screensets by simply giving both the same base filename, and giving one the .set extension, and the other the .sset extension.

***Arguments:***

None

***Return Value:***

None

***Example:***

```
` Toggle to simple screenset
ToggleScreens()
` Wait 5 seconds
Sleep 5000
` Toggle back to complex screenset
ToggleScreens()
```

***See also:***

## ***ToolLengthOffset***

Function ToolLengthOffset() As Double

This function returns the tool length offset for the current tool.

***Arguments:***

None

***Return Value:***

Tool length offset for current tool, as a Double

***Example:***

```
` Get the current tool length offset
LengthOffset = ToolLengthOffset
` Tell the user
Message "Tool length offset = " & LengthOffset
```

***See also:***

GetCurrentTool(), SetCurrentTool(), GetSelectedTool(), GetToolParam(), SetToolParam(),  
SetToolX(), SetToolZ(), GetToolDesc()

## ***VerifyAxis***

Sub VerifyAxis(Silent As Boolean)

This function verifies the positions of all axes by performing a home operation on all axes simultaneously, then moving all axes to position 0.0000 of the current fixture. If Safe\_Z is enabled, a move to Safe\_Z will be performed first. If the Silent argument is True, then upon completion, a message is displayed on the status line showing the offset errors of all axes from their home position. This function I used to ensure that no position loss has occurred, and/or to correct any position loss that may have occurred since the previous home or verify operation.

**Arguments:**

Silent is a Boolean value that, if true, causes a message to be displayed on completion of the verify indicating the offsets, if any, of each axis from its correct position.

**Return Value:**

None. All axes machine zeroes are reset to their correct position, and all axes are moved to position 0.0000 of the current fixture.

**Example:**

```
` Make sure all axes are in correct position
VerifyAxis()
` Load the G-code file
LoadFile("MyGCodeFile.nc")
` Wait for Load to Complete
While IsLoading()
    Sleep 100
Wend
` Run the File
DoOEMButton(CycleStartButton)
` Wait for it to complete
While (GetOEMLLED(StartLED))
    Sleep 100
Wend
```

**See also:**

SingleVerify(), SingleVerifyReport(), RefCombination()

## **ZeroTHC**

Sub ZeroTHC()

This function zeroes the torch height correction factor.

**Arguments:**

None

**Return Value:**

None

**Example:**

```
` Zero THC correction
```



`ZeroTHC( )`

***See also:***

`THCOn()`, `THCOff()`

## **Legacy Functions Grouped By Function**

### ***Digitizing***

CloseDigFile – Close digitization points file

OpenDigFile – Open digitization points file

### ***G-Code & G-code Files***

Code – Execute a line of G-code

FileName – Get current G-code filename

GetVar – Get a G-code Var value

IsLoading – Get current G-code file loading status

LoadFile – Load a G-code file

LoadRun – Load and run a G-code file

MaxX – Get maximum X extent for current G-code file

MaxY – Get maximum Y extent for current G-code file

MinX – Get minimum X extent for current G-code file

MinY – Get minimum Y extent for current G-code file

Param1 – Get M-macro P parameter value

Param2 – Get M-macro Q parameter value

Param3 – Get M-macro Q parameter value

RunFile – Run currently loaded G-code file

SetVar – Set a G-code Var value

### ***Lathe-only Functions***

GetTurretAng – Get current tool turret angle

IsDiameter – Get current diameter/radius mode status

### ***Mach3 Configuration & Status***

GetIJMode – Get current IJ mode

GetMainFolder – Get Mach3 main folder path

GetParam – Get a Mach3 named parameter value

GetScale – Get current scale factor for an axis

GetTimer – Get a timer value

HelpAbout – Get current VB version

QueueDepth – Get current Mach3 trajectory queue depth

ResetAxisSwap – Un-swap swapped axes  
RetractMode – Get current peck cycle retract mode  
SetFormula – Set a new axis formula  
SetIJMode – Set IJ mode  
SetParam – Set a Mach3 names parameter  
SetScale – Set a new axis scale factor  
SetTimer – Reset a timer  
SwapAxis – Swap axis outputs

### **Referencing, Verifying & Zeroing Axes**

GetABSPosition – Get machine position for an axis  
RefCombination – Reference any combination of axes  
SetMachZero – Set machine zero to current position  
SingleVerify – Verify position of a single axis  
SingleVerifyReport – Verify position of a single axis, and report if error  
VerifyAxis – Verify position of a single axis

### **SafeZ**

GetSafeZ – Get current SafeZ height  
GotoSafeZ – Go to current SafeZ height  
IsSafeZ – Find out if machine is currently at or above SafeZ height  
SetSafeZ – Set SafeZ height

### **Wizards& Plugins**

LoadWizard – Load a wizard by name  
NotifyPlugins – Send a notification to all plugins  
SaveWizard – Save current wizard settings

### ***Machine Status & Control***

IsEstop – Find out if Mach3 is currently in E-Stop  
CoupleSlave – Couple slave axis to its master for referencing

### **Motion Control**

FeedRate – Get current feedrate  
JogOff – Disable jogging for a single axis  
JogOn – Jog a single axis in a specified direction

SetFeedRate – Set a new feedrate  
StraightFeed – Perform a feedrate move  
StraightTraverse – Perform a rapid move  
IsMoving – Find out if machine is currently moving  
IsStopped – Find out if machine is currently stopped

### **Spindle Control**

DoSpinCCW – Turn spindle on CCW  
DoSpinCW – Turn spindle on CW  
DoSpinStop – Turn spindle off  
GetRPM – Get current spindle RPM  
SetPulley – Set a new spindle pulley  
SetSpinSpeed – Set a new spindle speed

### **Tool Parameters and Toolchanges**

GetCurrentTool – Get currently loaded tool  
GetSelectedTool – Get newly selected tool  
GetToolChangeStart – Get axis positions at start of toolchange  
GetToolDesc – Get the tool descriptor for the specified tool number  
GetToolParam – Get value of a named tool parameter for the specified tool number  
SetCurrentTool – Set the current tool to the specified value  
SetToolParam – Set value of a named tool parameter for the specified tool number  
SetToolX – Set the X axis offset  
SetToolZ – Set the Z axis offset  
ToolLengthOffset – Set the Z offset for the current tool

### **Torch Height Control**

EndTHC – Turn off THC  
ResetTHC – Reset THC  
StartTHC – Start THC  
THCOff – Turn off THC  
THCOn – Turn on THC  
ZeroTHC – Zero THC

### **Screensets**

DoMenu – Invoke a menu function

DoOEMButton – Invoke an OEM Button function  
GetDRO – Get current value for specified OEM DRO  
GetDROString – Get current value for specified OEM DRO as a String  
GetLED – Get current state of specified OEM LED  
GetOEMDRO – Get current value for specified OEM DRO  
GetOEMLED – Get current state specified OEM LED  
GetPage – Get current screenset page number  
GetUserDRO – Get current value of specified User DRO  
GetUserLabel – Get current value of specified User Label  
GetUserLED – Get current state of specified User LED  
LoadStandardLayout – Load default screenset  
Message – Display a message on the Error SmartLabel  
SetButtonText – Change the label text on the currently active button  
SetDRO – Set the value of the specified OEM DRO  
SetOEMDRO – Set the value of the specified OEM DRO  
SetPage – Select a new screenset page by number  
SetTicker – Set the value of a Ticker SmartLabel  
SetUserDRO – Set the value of the specified User DRO  
SetUserLabel – Set the value of the specified UserLabel SmartLabel  
SetUserLED – Set the state of the specified User LED  
ToggleScreens – Toggle between “complex”/.set and “simple”/.sset screensets

## **User Dialogs**

AskTextQuestion – Ask the user a question and get a String response  
GetACoor – Get A coordinate given by last GetCoord() call  
GetCoord – Get axis coordinates from user  
GetXCoor – Get X coordinate given by last GetCoord() call  
GetYCoor – Get Y coordinate given by last GetCoord() call  
GetZCoor – Get Z coordinate given by last GetCoord() call  
MachMsg – Display a dialog with one or more buttons on it  
Question – Ask the user a question, and get a numeric response

## **Signals and Port I/O**

ActivateSignal – Activate/Assert an output signal

DeactivateSignal – Deactivate/Deassert an output signal  
GetPortByte – Read a byte from an I/O port  
IsActive – Find out if the specified named input signal is in its active state  
IsOutputActive – Find out if the specified named output signal is in its active state  
IsSuchSignal – Find out if the specified named signal is defined  
PutPortByte – Write a byte to an I/O port  
SetTriggerMacro – Associate an M-macro with OEM code 301  
SystemWaitFor – Pause script execution until specified named input signal is asserted

### ***Teach Files***

AppendTeachFile – Open a Teach file for append  
CloseTeachFile – Close currently open Teach file  
LoadTeachFile – Load current Teach file for execution  
OpenTeachFile – Open a new Teach file for writing

### ***Miscellaneous***

nFmt – Round a Double value to a specified number of decimal places  
PlayWave – Play a .WAV file  
Random – Get a pseudo-random number  
Roun – Round a Double value to four decimal places  
Sleep – Pause script execution for specified length of time  
Speak – Use voice synthesis to “speak” a String

## Modbus Functions Grouped Alphabetically

### **GetInBit**

Function

This function

*Arguments:*

F

*Return Value:*

A

*Example:*

A

*See also:*

### **GetInput**

Function

This function

*Arguments:*

F

*Return Value:*

A

*Example:*

A

*See also:*

### **ResetOutBit**

Function

This function

*Arguments:*

F

*Return Value:*

A

*Example:*

A

*See also:*

## SetHomannString

Sub SetHomannString(Col As Integer, Row As Integer, Text As String)

This function writes the Text string to the LCD display on ModIO slave. Row and Col specify the row and column position of the first character of the string. This function is exactly equivalent to the SetModIOString function with a SlaveID of 1.

### *Arguments:*

Slave is the ModBus Slave ID of the device whose LCD the string is written to

Col is the 0-based column to which the first character of the Text will be written. Col **must** be an even number (multiple of 2).

Row is the 0-based row this which the Text will be written

Text is the String to be written to the LCD

### *Return Value:*

None

### *Example:*

‘ Write “Hello, world!” to the second line of the 2x16 LCD on Modbus Slave 2

SetModIOString(2, 0, 1, “Hello, world!”)

### *See also:*

FillFromCoil(), FillFromHolding(), FillFromInput(), FillFromStatus(), GetModWord(),  
ModGetInputWord(), SetModIOString(), SetModOutput(), SetModPlugString(), ResetOutBit(),  
GetInBit(), GetInput(), SetOutBit(), SetOutput(), WaitForPoll(),SetModIOString()

## SetModIOString

Sub SetModIOString(SlaveID As Integer, Col As Integer, Row As Integer, Text As String)

This function writes the Text string to the ModIO devices LCD display. Row and Col specify the row and column position of the first character of the string.

### *Arguments:*

Slave is the ModBus Slave ID of the device whose LCD the string is written to

Col is the 0-based column to which the first character of the Text will be written. Col **must** be an even number (multiple of 2).

Row is the 0-based row this which the Text will be written

Text is the String to be written to the LCD

### *Return Value:*

None

### *Example:*

‘ Write “Hello, world!” to the second line of the 2x16 LCD on Modbus Slave 2

SetModIOString(2, 0, 1, “Hello, world!”)



***See also:***

FillFromCoil(), FillFromHolding(), FillFromInput(), FillFromStatus(), GetModWord(),  
ModGetInputWord(), SetModIOString(), SetModOutput(), SetModPlugString(), ResetOutBit(),  
GetInBit(), GetInput(), SetOutBit(), SetOutput(), WaitForPoll(),SetHomannString()

## **SetModOutput**

Function

This function

***Arguments:***

F

***Return Value:***

A

***Example:***

A

***See also:***

## **SetOutBit**

Function

This function

***Arguments:***

F

***Return Value:***

A

***Example:***

A

***See also:***

## **WaitForPoll – Unreliable....**

Function

This function

***Arguments:***

F

***Return Value:***

A

***Example:***

A

*See also:*

## Serial Output Functions Grouped Alphabetically

There are a number of serial functions in the v3 code-base, some previously documented, some not. However, testing indicates the following is the only function that actually works. Serial input is not supported in Mach3 v3.

### SendSerial

Sub SendSerial(Data As String)

This function send the String Data to the serial port specified in the Config->GeneralConfig serial port configuration. This provides transmit-only capability, at any supported BAUD rate.

***Arguments:***

String message to be sent to configured serial device

***Return Value:***

None

***Example:***

```
` Send "Hello, world!" to serial device  
SendSerial("Hello, world!" & chr(10) & char(13))
```

***See also:***