• First a little background on some things that I had a hard time with at the start.
• My wording of these may not be correct but I hope it at least gives the correct information in a way that people like me can understand.

• Mach4 is not exactly one big program. It is made of several smaller programs that work together. I have heard them referred to as “chunks” I believe. Sometimes things done in one part are not immediately seen in another part. Example: If running a macro the GUI LED’s and DRO’s are not updated until the macro finishes running.
• Something that works in a button may not work in a macro.
  Example: mcAxisHome works in a button but not in a macro.
  I have been told both it will and it won’t but I could not get it to work in a macro.
  Also, reading and writing to a DRO will work realtime in a button but not a macro.

• The mc calls can be found at C:\Mach4Hobby\Docs\Mach4CoreAPE.chm
• The scr calls can be found online at (thanks to Brett and Craig); https://www.machsupport.com/forum/index.php?topic=39762.0
• Registers are a great way to get information from one part of Mach4 to another. They are updated immediately across all parts of Mach4.
• The main part of Mach4 has a PLC built in. The PLC runs constantly in the background and can be used to do things depending on a register setting or a physical button push on the machine. Example: I have a physical button on one of my machines that is used as a pause button. I have also used it to look at a register and when the register changed to 1 it did an action.
• In the following sections I will explain how to get to the PLC and how to make new registers as well as use the registers.
• I am going to try starting with the simplest and working my way up.

• Macros go in the macros folder of whatever profile you are using.
• Here is the format of a macro.

```lua
function m#()
    your code
end
if (mc.mcInEditor() ==1) then
    m#()
end
```

The last 3 lines let you run or step through the macro while in the editor.

Example: (for this example theGetInstance is not needed but....)

```lua
Function m111()
    local inst = mc.mcGetInstance()
    mc.mcCntlGcodeExecute ("G0 X10.050")
end
if (mc.mcInEditor() ==1) then
    m111()
end
```

• Comments are very helpful in your code.
  Put - - in front of anything you want to comment out.

Example:

```lua
-- this is a comment
```

• Some things need numbers (math calculations) and others need a string (registers).
  To change to a number.

```lua
local variable = tonumber(value)
```

Example:

```lua
local Number = tonumber(3.0)
```

or

```lua
local Num = 3.0
local Number = tonumber(Num)
```
To change to a string.

   local variable = tostring(value)

Example:

   local String = tostring(3.0)

or

   local Str = 3.0
   local String = tostring(Str)

- **One thing that is needed at the start of any code is the following.**

   Local inst = mc.mcGetInstance()

- **How to make the program pause for a set amount of time in a button, macro or in the PLC.**

   I use this to allow air cylinders to complete their stroke.

   wx.wxSleep(seconds)
   wx.wxMilliSleep(milliseconds)

Example:

   wx.wxSleep(1)
   wx.wxMilliSleep(1000)

- **How to read a DRO’s value.**

   local variable = scr.GetProperty('DRO Name', 'Value')

Example:

   local StartPos = scr.GetProperty('dro Current Pos Y', 'Value')

- **How to write to a DRO.**

   scr.SetProperty('DRO Name', 'Value', tostring(variable))

Example:

   local Temp = 3.00
   scr.SetProperty('droTempPos', 'Value', tostring(Temp))
• How to get a LED’s state.
  local variable = scr.GetProperty('LED Name', 'Value')

  Example:
  local StartSet = scr.GetProperty('LEDStartComplete', 'Value')

• How to set a LED’s state.
  scr.SetProperty('LED Name', 'Value', 'State')

  Examples:
  scr.SetProperty('LEDStartComplete', 'Value', '1')
    LED set to on.
  scr.SetProperty('LEDStartComplete', 'Value', '0')
    LED set to off.

• How to get an input or output state.
  variable = mc.mcSignalGetHandle(inst, signal name)
  variable2 = mc.mcSignalGetState(variable)

  Examples:
  hsig = mc.mcSignalGetHandle(inst, mc.ISIG_MOTOR0_HOME)
  MatHome = mc.mcSignalGetState(hsig)
    Returns a 1 or 0 depending on if motor 0 has been homed.
  hsig2 = mc.mcSignalGetHandle(inst, mc.OSIG_OUTPUT8)
  ArmDown = mc.mcSignalGetState(hsig2)
    Returns a 1 or 0 depending on if output 8 is activated or not.

• How to set an output state.
  variable = mc.mcSignalGetHandle(inst, signal name)
  mc.mcSignalSetState(variable, state)

  Examples:
  notch = mc.mcSignalGetHandle(inst, mc.OSIG_OUTPUT8)
  mc.mcSignalSetState(notch, 1)
    Turns output 8 active or on.
  mc.mcSignalSetState(notch, 0)
    Turns output 8 inactive or off.
• How to run G Code in a button or macro.
  This can be done directly using G Code or using variables.
  `mc.mcCntlGcodeExecute(inst, "g code")`

Example:

```
mc.mcCntlGcodeExecute(inst, "G1 X22.375 F200.0")
```

or

```
local variable = "g code to run"
mc.mcCntlGcodeExecute(inst, variable)
```

Example:

```
local ClearPosition = "G1 X22.375 F200.0"
mc.mcCntlGcodeExecute(inst, ClearPosition)
```

or

```
local variable = value
local variable2 = stringformat("G0" .. variable .. "X" .. variable)
mc.mcCntlGcodeExecute(inst, variable2)
```

Example:

```
local zero = 0
local GoToZero = stringformat("G0 X" .. zero .. "Y" .. zero)
mc.mcCntlGcodeExecute(inst, GoToZero)
```

• How to turn on Registers.
  Go to Configure, Control, Plugins tab then place a green check next to Regfile.

• How to make a new Register.
  Go to Configure, Plugins then Regfile.
  Click on the green plus sign.
  Give the register a name. (no spaces)
  Give the register a starting value.
  Put in a longer description.
  Persistant
  A green check will keep value on exit from Mach4.
  A red x will start every new start of Mach4 with the starting value.
• **How to read a Register.**
  
  ```lua
  local variable = mc.mcRegGetHandle(inst, 'path')
  local variable2 = mc.mcRegGetValue(variable)
  ```

  **Examples:**
  
  ```lua
  local hreg = mc.mcRegGetHandle(inst, 'Encoder_0')
  local EncRawVal = mc.mcRegGetValue(hreg)
  ```

  or
  
  ```lua
  local hreg = mc.mcRegGetHandle(inst, 'ESS/EncRaw')
  local EncRawVal = mc.mcRegGetValue(hreg)
  ```

  or
  
  ```lua
  local hreg = mc.mcRegGetHandle(inst, 'iRegs0/NotchTime')
  local EncRawVal = mc.mcRegGetValue(hreg)
  ```

• **How to write to a Register. (this might be wrong)**
  
  ```lua
  local variable = mc.mcRegGetHandle(inst, 'path')
  local mc.mcRegSetValue(variable, value)
  ```

  or
  
  ```lua
  local variable = mc.mcRegGetHandle(inst, 'path')
  local mc.mcRegSetValue(variable, tostring(value))
  ```

  **Example:**
  
  ```lua
  local hreg = mc.mcRegGetHandle(inst, 'ESS/EncRaw')
  local mc.mcRegSetValue(hreg, 23.35)
  ```

  or
  
  ```lua
  local Num = 23.35
  local hreg = mc.mcRegGetHandle(inst, 'ESS/EncRaw')
  local mc.mcRegSetValue(hreg, tostring(Num))
  ```