Down and Dirty “mcLua” scripting, quick ref guide.

By Scott “Poppa Bear” Shafer

Note, this is just for common things, that most folks want to do, It does NOT show initializing the variables and all that just the code, to do this or that. (see the McLua scripting list that is on here for more details on specific calls, ALSO there are MANY, MANY, MANY functions that do specific things. I will not go into those here, since they are in the current mcLua scripting reference in the “Tool Box”. The thought here is just the basic foundational stuff……. Error checking on rc is NOT done in the examples.

Show an error on the mach error line:

```
mc.mcCntlSetLastError(inst, "WHAT THE HECK DID YOU DO?!");
```

To Get an Input: --## = 0-63

```
hsig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_INPUT##);
SigState = mc.mcSignalGetState(hsig);
if SigState == 1 then
    -- your code here to do whatever about it.
end
```

To get an axis is home: --## = 0-31

```
hsig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_MOTOR##_HOME);
SigState = mc.mcSignalGetState(hsig);
if SigState == 1 then
    -- your code here to do whatever about it.
end
```

To get an axis is Positive Limit switch: --## = 0-31

```
hsig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_MOTOR##_PLUS);
SigState = mc.mcSignalGetState(hsig);
if SigState == 1 then
    -- your code here to do whatever about it.
end
```

To get an axis is Negative Limit switch: --## = 0-31

```
hsig, rc = mc.mcSignalGetHandle(inst, mc.ISIG_MOTOR##_MINUS);
SigState = mc.mcSignalGetState(hsig);
if SigState == 1 then
    -- your code here to do whatever about it.
end
```

To get Other (specific named input):

```
--SpecificNamedInputs =
        mc.ISIG_DIGITIZE,
        mc.ISIG_INDEX,
        mc.ISIG_LIMITOVER,
        mc.ISIG_EMERGENCY,
        mc.ISIG_THCON,
        mc.ISIG_THCUP,
        mc.ISIG_THCDOWN,
        mc.ISIG_TIMING,
        mc.ISIG_JOGXP,
        mc.ISIG_JOGXN,
        mc.ISIG_JOGYP,
        mc.ISIG_JOGYN,
        mc.ISIG_JOGZP,
        mc.ISIG_JOGZN,
        mc.ISIG_JOGAP,
        mc.ISIG_JOGAN,
        mc.ISIG_JOGBP,
        mc.ISIG_JOGBN,
        mc.ISIG_JOGCP,
        mc.ISIG_JOGCN,
        mc.ISIG_SPINDLE_AT_SPEED,
        mc.ISIG_SPINDLE_AT_ZERO
hsig, rc = mc.mcSignalGetHandle(inst, SpecificNamedInputs);
SigState = mc.mcSignalGetState(hsig);
if SigState == 1 then
    -- your code here to do whatever about it.
end
```

To Set an Output: --## = 0-63

```
hsig, rc = mc.mcSignalGetHandle(inst, mc.OSIG_OUTPUT##);
local state  = false;
local VarIamTestingFor = SomeThingIamWatchingOrSettingOrValue;
if VarIamTestingFor == WhatSetsTheOutputToOn then
        state  = true;
else
        state = false;
end
mc.mcSignalSetState(hsig, state);
```

To Set an "Enable" Output: --## = 0-31

```
hsig, rc = mc.mcSignalGetHandle(inst, mc.OSIG_ENABLE##);
local state  = false;
local VarIamTestingFor = SomeThingIamWatchingOrSettingOrValue;
if VarIamTestingFor == WhatSetsTheOutputToOn then
        state  = true;
else
        state = false;
end
mc.mcSignalSetState(hsig, state);
```

To Set an axis switch status: "Is on a Limit" or "Is on a Home" for an Output:

$ = Axis Letter: X, Y, Z, A, B, or C

LHswtich = mc.OSIG_$LIMITPLUS, mc.OSIG_$LIMITMINUS, or mc.OSIG_$HOME

```
hsig, rc = mc.mcSignalGetHandle(inst,  LHswtich);
local state  = false;
local VarIamTestingFor = SomeThingIamWatchingOrSettingOrValue;
if VarIamTestingFor == WhatSetsTheOutputToOn then
        state  = true;
else
        state = false;
end
mc.mcSignalSetState(hsig, state);
```

Spindle Control:

--//Spindle CW button:

```
hsig, rc = mc.mcSignalGetHandle(inst, mc.OSIG_SPINDLEON);
mc.mcSignalSetState(hsig, 1); --enable the spindle
rc = mc.mcSpindleSetDirection(inst, 1) --dir is:  -1ccw, 1cw, 0 stop
```

--//Spindle CCW button:

```
hsig, rc = mc.mcSignalGetHandle(inst, mc.OSIG_SPINDLEON);
mc.mcSignalSetState(hsig, 1); --enable the spindle
rc = mc.mcSpindleSetDirection(inst, -1) --dir is:  -1ccw, 1cw, 0 stop
```

--//Spindle OFF button:

```
rc = mc.mcSpindleSetDirection(inst, 0) --dir is:  -1ccw, 1cw, 0 stop
```

--//Spindle Set the RPM:

```
rc = mc.mcSpindleSetCommandRPM(inst, RPMvalue) --set the speed
```

To Set an "Other function" (unique functions) Output:

```
OtherTypeOutPutFunctionName =
        mc.OSIG_RUNNING_GCODE,
        mc.OSIG_FEEDHOLD,
        mc.OSIG_BLOCK_DELETE,
        mc.OSIG_SINGLE_BLOCK,
        mc.OSIG_REVERSE_RUN,
        mc.OSIG_OPT_STOP,
        mc.OSIG_MACHINE_ENABLED,
        mc.OSIG_TOOL_CHANGE,
        mc.OSIG_DIST_TOGO,
        mc.OSIG_MACHINE_CORD,
        mc.OSIG_SOFTLIMITS_ON,
        mc.OSIG_JOG_INC,
        mc.OSIG_JOG_CONT,
        mc.OSIG_JOG_ENABLED,
        mc.OSIG_JOG_MPG,
        mc.OSIG_HOMED_X,
        mc.OSIG_HOMED_Y,
        mc.OSIG_HOMED_Z,
        mc.OSIG_HOMED_A,
        mc.OSIG_HOMED_B,
        mc.OSIG_HOMED_C,
        mc.OSIG_DWELL,
        mc.OSIG_TP_MOUSE_DN,
        mc.OSIG_LIMITOVER,
        mc.OSIG_CHARGE,
        mc.OSIG_CHARGE2,
        mc.OSIG_CURRENTHILOW,
        mc.OSIG_SPINDLEON,
        mc.OSIG_SPINDLEFWD,
        mc.OSIG_SPINDLEREV,
        mc.OSIG_COOLANTON,
        mc.OSIG_MISTON,
        mc.OSIG_DIGTRIGGER
hsig, rc = mc.mcSignalGetHandle(inst, OtherTypeOutPutFunctionName);
local state  = false;
local VarIamTestingFor = SomeThingIamWatchingOrSettingOrValue;
if VarIamTestingFor == WhatSetsTheOutputToOn then
        state  = true;
else
        state = false;
end
mc.mcSignalSetState(hsig, state);
```

To do/get a "Function" in Mach4:
There is a **MASSIVE list of functions** please see the mcLua reference guide in the Tool Box for the complete list…… again, it is huge. I will show the basic form and a few simple/commonly used funcs.

Basic form = mc.mc$$$$$$$$$$$$$$$$(Parameters);
$$$$$$$$$$$$$$$$ = TheFunctionName
Parameters = whatever that function takes as arguments.
AxisID = 0-5(X-C)
state = 1 (true) or 0 (false)
doubleVal = decimal number, i.e. 100.021
StringBuffer = your g-code line or complete g-code routine
StringPath = path to your g-code folder and including tap file name with extension

COMMON DO THINGS FUNCS:

- mc.mcAxisEnable(inst, AxisID, state); --enable an axis
- mc.mcAxisHome(inst, AxisID);--home specific axis
- mc.mcAxisSetMachinePos(inst, AxisID, doubleVal);--set Axis machine absolute position
- mc.mcAxisSetSoftlimitEnable(inst, AxisID, 1); --enable soft limits for an axis
- mc.mcAxisSetPos(inst, AxisID, doubleVal);--set an working axis position
- mc.mcToolPathGenerate(inst);--redraw tool path, used after zero axis above or other position change.
- mc.mcCntlCycleStart(inst);--cycle start
- mc.mcCntlCycleStop(inst);--cycle stop
- mc.mcCntlEStop(inst);--Estop
- mc.mcCntlFeedHold(inst);--Feedhold
- mc.mcCntlGcodeExecute(inst, StringBuffer);--run some g-code
- mc.mcCntlLoadGcodeFile(inst, StringPath);--load a g-code file
- mc.mcCntlCloseGCodeFile(inst);--close a g-code file that is loaded.
- mc.mcCntlRewindFile(inst);--rewind the g-code that is loaded
- mc.mcCntlMdiExecute(inst, StringCmds);--like executing from the MDI
- mc.mcCntlReset(inst);--Reset button
- mc.mcCntlSetOptionalStop(inst, state);--set optional stop M1
- mc.mcCntlSetPoundVar(inst, VarNum, doubleVal);--set a #(pound) variable
- mc.mcCntlSetSingleBlock(inst, state);--control single block

COMMON GET THINGS FUNCS:

- val, rc = mc.mcCntlGetState(inst);--Gets Machs state
- val, rc = mc.mcCntlGetFRO(inst);--get feed OVR value
- bool, rc = mc.mcCntlIsInCycle(inst);--Gcode file running?
- bool, rc = mc.mcCntlIsStill(inst);--anything moving?
- val, rc = mc.mcCtrlGetRunTime(inst);--how long has file been running?
- doubleVal, rc= mc.mcAxisGetPos(inst, AxisID);--what is axis position, doubleVal
- doubleVal, rc = mc.mcCntlGetPoundVar(inst, VarNum);--Get the Pound vars value
- doubleVal, rc = mc.mcAxisGetMachinePos(inst, AxisID);--what is absolute axis position?

Read a Register: --(this works for reading a Modbus registers also)
hReg, rc= mc.mcRegGetHandle(inst, RegisterPathAsString);
val = number or string depending on type of register you're reading from
val = mc.mcRegGetValue(hReg);

Write a Register: --(this works for writing a Modbus registers also)
hReg, rc= mc.mcRegGetHandle(inst, RegisterPathAsString);
val = number or string depending on type of register you're writing to
mc.mcRegSetValue(hReg, val);

Get Hardware(device) I/O state: --get a point of I/O
hReg, rc = mc.mcIoGetHandle(inst, DeviceNameAsString/ IoNameAsString);
BoolPointValue = mc.mcIoGetState(hReg);

Set Hardware(device) I/O state: --Set a point of I/O
hReg, rc = mc.mcIoGetHandle(inst, DeviceNameAsString/ IoNameAsString);
mc.mcIoSetState(hReg, BoolPointValue);

Screen Stuff: You can get the names of the object from within the screen designer to include property names and values. Further note, you can access screen objects from within macros or wizards.

The **SET** property for screen work has three properties, each of these properties are
Passed as 'Strings'.
'Param1' = string name of the object on your screen that you are accessing.
'Param2' = string name of the objects "property" you want to access
'Param3' = string name of the value your assigning to the "property" your changing.
scr.SetProperty('Param1','Param2','Param3');

The **GET** property for screen work has two properties, each of these properties are
Passed as 'Strings'.
'Param1' = string name of the object on your screen that you are accessing.
'Param2' = string name of the objects "property" you want to access its value
Val is what the value is of the property you getting.
Val = scr.GetProperty('Param1','Param2');

Here is a MACRO for testing/debugging a script that you can put your test code in.

```
function TestFunc()
   local mInst = 0;
   local rc = 0;
   local inst = mc.mcGetInstance(mInst);

   --Your Test Code Here

   local debugvar = 0;--if stepping through with breakpoints a place to stop
end

if (mc.mcInEditor() == 1) then
   TestFunc();
end
```