Lua4RC Custom Programming

Lua Programming for **SpaceLogic™** SE8000 Room Controllers

Version 6

Schneider Electric

Contents

Safety Information

Important Information

Read these instructions carefully and inspect the equipment to become familiar with the device before trying to install, operate, service or maintain it. The following special messages may appear throughout this bulletin or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER
DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING
WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

⚠ CAUTION
CAUTION indicates a hazardous situation which, if not avoided, **could result in** minor or moderate injury.

NOTICE
NOTICE is used to address practices not related to physical injury. The safety alert symbol shall not be used with this signal word.

Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction, installation, and operation of electrical equipment and has received safety training to recognize and avoid the hazards involved.

Lua4RC Programming for SE8000

SE8000 Room Controllers can run custom applications designed to meet specific customer requirements. These scripts, referred to as Lua4RC scripts, can be developed for Integrators, or by qualified Integrators. Lua4RC adds a layer of programming on top of the embedded control logic of a SE8000 Room Controller.

The script running on the Room Controller has the ability to override parameters set by the embedded application. With this added flexibility, you can adapt the control logic of the SE8000 Room Controllers to meet the specific requirements of your projects.

This section gives an overview of the basic functions of the Lua language. It is not an exhaustive and complete tutorial on the Lua language.

Accessing SE8000 Room Controller Database

When writing a Lua4RC script, the keyword "**ME**" is used to access the objects in the local database. For example, a line that reads "**ME.AV25 = 10**" in a Lua script would set the value of the **AV25** object as 10.

The following object types are available:

1. AI (Analog Input)
2. AO (Analog Output)
3. AV (Analog Value)
4. BI (Binary Input)
5. BO (Binary Output)
6. BV (Binary Value)
7. MSI (Multi-state Input)
8. MV (Multi-state Value)
9. CSV (Character String Value)

Notes:

- A list of each point available, along with the description and possible values can be found in the SE8000 Room Controllers BACnet Integration Guide.
- When accessing Binary Objects (BI, BO, BV), the return values are limited to 0 and 1, and not 'true' or 'false'.

Lua4RC Scripts

There are 2 distinct locations to store custom Lua scripts in a SE8000 Room Controller, each having different characteristics and limitations:

Before 2.5.0	2.5.0 and Later
1. Flash memory: • Single script • Run time 1 second • Maximum script size = 80 kB • Script loaded via USB using the SE8000 Uploader tool 2. PG objects on BACnet: • Up to 10 scripts running in a single thread • Run time 1 second for thread • Maximum script size = 420 bytes / script It is possible to load scripts using both methods on the same Room Controller according to the following: 1. A script loaded to any PG object disables a script loaded to flash 2. A script loaded to flash will only be available if all PG objects are empty	Flash memory: • Single script • Run time 1 second • Maximum script size = 80 kB • Script loaded via: ▪ USB using the SE8000 Uploader tool (write only), or ▪ BACnet using a file object (read/white)

Standard Lua Library

The SE8000 Lua environment uses LUA 5.1. Refer to the following for additional information:

http://www.lua.org/manual/5.1/manual.html#5 for an introduction to the basics of Lua programming.
http://www.lua.org/manual/5.1/ provides more general details on the Lua language.

Only basic functions and mathematical functions are implemented. Other libraries (advanced string manipulation, table manipulation, input and output facilities, operating system facilities, and debug libraries) are not available.

Lua Functions and Tools

These functions offer basic control over common Room Controller applications frequently used in most installations.

tools.switch()

Switch function (on-off with deadband). Simulates the operation of a conventional ON-OFF thermostat. It also provides a deadband function, so an Object does not continuously switch ON and OFF based on a specific value. output (0 or 1) =tools. switch(output, input-expr, off-expr, on-expr).

```
ME.BO28 = tools.switch(ME.BO28, ME.AO21, 0, 15)    --W1(BO28) will go (ON at 15% PI_Heat) then (OFF at 0% PI_Heat)
```

The “switch” function is the equivalent of:

```
if ME.AO21>15 then
ME.BO28=1
end

if ME.AO21==0 then
ME.BO28=0
end
```

tools.scale()

tools.scale(variable,offset,x1,y1,x2,y2). This function returns the linear interpolation between two points. The function can also add the offset value to the final result if desired.

```
ME.AO123 = tools.scale(ME.AO21, 0, 0, 2, 100, 10)    --UO11(AO123) 2-10Vdc will follow the 0-100% PI_Heat (AO21)
```

Scripting Best Practices

This section provides an overview of best practices when writing Lua4RC scripts.

Variable Declaration

Variable declarations should always be made at the beginning of a script and contained within an **“init”** statement.

This is done to optimize CPU usage, processing time, proper initialization of values and is a general scripting good practice.

The below shows an example.

```
if not init then
    ME.MV6 = 2        --Network units = °F
    ME.MV145 = 2    --Set Room_Temp sensor = Local
    init = true
end
-- rest of control script here, if applicable.
```

Priority Management

Lua4RC accesses various points of the Room Controller using BACnet naming convention and priorities. The default priority of the Room Controller’s internal control sequence is 17. When writing a Lua4RC script, the default write priority is priority 16. As a result, the internal control is overridden by LUA commands such as “**ME.AV25 = 10**”.

Apply caution when using priorities other than the default value as this could result in some values being permanently overridden. To write to another priority, an array is used.

The example below would write a value of 20 to AV25, using priority 8:

```
ME.AV25_PV[8] = 20
```

To release this specific priority, you can set it to nil:

```
ME.AV25_PV[8] = nil
```

To access the relinquish default (the Room Controller’s internal logic application priority), priority 17 can be used.

```
ME.AV25_PV[17] = 30
```

NOTICE

PRIORITY LEVELS

Priority levels 1, 2, 3 and 17 (Relinquish Default) are stored in the non-volatile (EEPROM) memory of the Room Controller.

- This means the stored values will remain in the memory after a power cycle. It is necessary to perform a factory reset of the Room Controller to reset these values.
- Each location in the EEPROM memory is limited to 1 million (1,000,000) writes, so care must be taken to minimize changes written to priorities stored in EEPROM. Failure to do so will damage the device.
  - Do not write values that change regularly to priority levels stored in EEPROM. Use other priority levels that are stored in RAM only and support infinite read/write cycles.
  - Do not write to EEPROM locations on every cycle of the script (every 1 second). If you must write data in priority levels stored in EEPROM, do so less frequently (e.g. every 15 minutes).
  - Do not “NIL” a value before writing to it. This is unnecessary and causes two writes to the EEPROM location. For example, avoid the following:
    - ME.MV16_PV[17] = nil
    - ME.MV16_PV[17] = 2

Failure to follow these instructions can result in equipment damage.

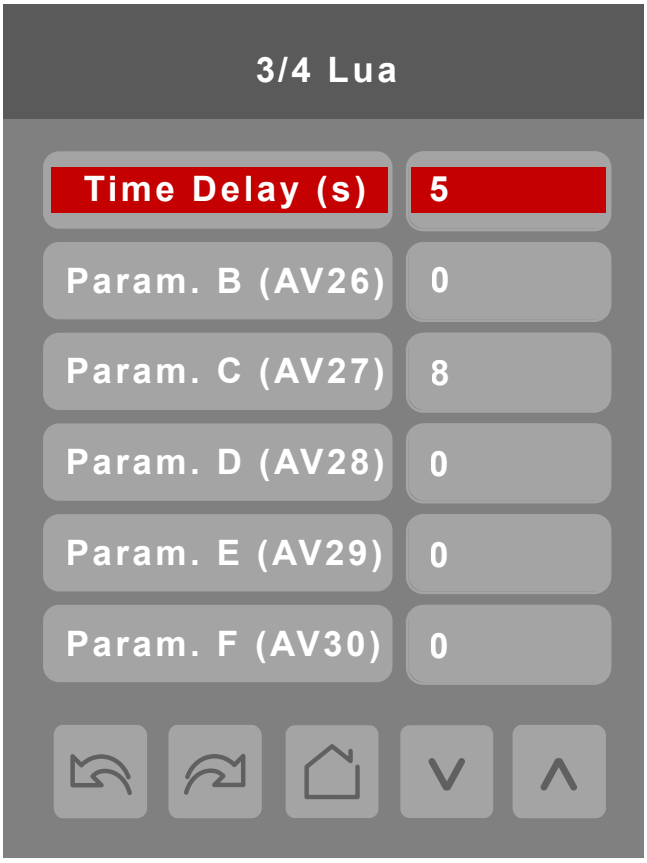
Script / BACnet Variables

To interface between the Lua engine of the Room Controller and the BACnet integration, 12 variables are made available: **AV25** to **AV30** and **AV225** to **AV230**. These can be read and written from the Lua engine and BACnet. Since these variables are also visible and configurable from the Room Controller's HMI, they can be exposed to the User for quick customization or parametrization of points.

To change the name of one or more of these variables, the **_Desc** property of each point can be modified from a Lua script. This will be visible both in the HMI and BACnet.

To change the name of **AV25** (for example) to **Time delay (s)** the following script can be used:

```
ME.AV25_Desc = "Time delay (s)"
```



Minimum / Maximum and Increment Values

To restrict the values of **AV25-AV30** to a certain range, the minimum and maximum acceptable values can be adjusted by modifying the **_Min** and **_Max** properties of each object.

Note: For Firmware version 1.4.2, **AV30** has a known issue where **_Min**, **_Max** and **_Inc** will not function.

It is also possible to modify the increment property of a point. This will be used when adjusting the point on the Room Controller. An increment of 10 means the Room Controller will cycle between 0,10,20, etc. when using the up/down arrows, while an increment of 5 will cycle between 0,5,10,15, etc.

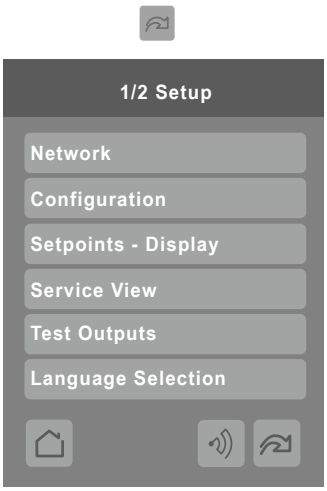
Note: If the increment is set to 0, the parameter is read-only. If the parameter is set to nil, the present value is not displayed.

User Interface

This section describes using the Lua screens on the SE8000 Room Controllers.

Access Lua Configuration Menu

1. Press and hold the top-center area of the Room Controller screen for 2 seconds.
2. Navigate to the Setup page 2/2 by pressing the arrow button and select Lua.



Lua Screen 1/4

Lua page 1/4 presents the first 10 lines of the script.

Lua Screen 2/4

This screen gives the option to run or stop the Lua script. When the Room Controller is started up and a script is loaded, the script is automatically in Running status, shown in the Program status field.

If the script is not in Running status, the Program error field shows if there is an error in the script. If the field shows anything other than No error, there are errors in the loaded script and it does not run. If there are errors, a message explaining the error(s) is shown in the Debug log.

The most frequent error is, for example: “Unable to write to AV39_Present_Value[16]” and most of the time it is because the value is out of range. Writing a temperature in °F like Heating Setpoint = 73 when the Network units are set to default degree Celcius results in 73°C being out of range. To resolve this issue, set Network Units to °F (MV6=2) in the INIT section before writing to any temperature points.

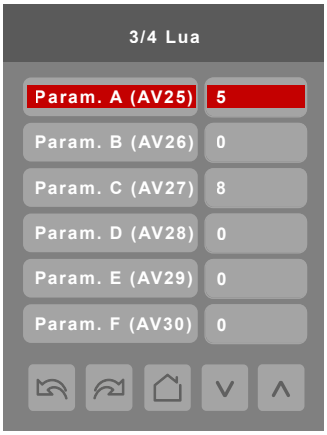
The Debug log can also be used for printing values from the script. The Debug log window can hold a maximum of 78 characters on 3 lines. The Debug log is refreshed every time the script loops back on itself.

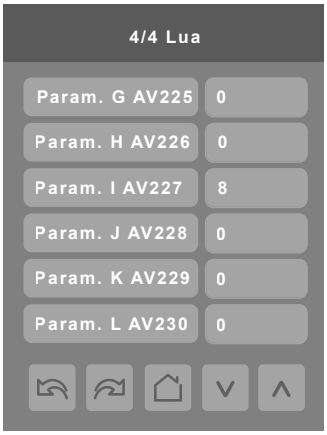
Lua Screens 3/4 and 4/4

The SE8000 Lua environment offers 12 AV objects that can be used in the scripts. These objects are regular BACnet AV objects and are accessible directly from the screen. The values can be set in the user interface, and the user interface sets the value at the lowest level of priority (relinquish default level 17).

Any value gets overridden by a value set in a script or via BACnet. When the value is overridden, the value’s text shows red and it is not possible to change it in the interface. To release the override, the script or the BACnet client must set the value priority to nil.

IMPORTANT: Even if the script is not currently running, the AV25 to AV30 and AV225 to AV230 variables, if used by the script, get overridden and cannot be changed by the user.





Lua4RC Applied Examples

The section shows some common applied examples using Lua4RC.

Pre-commission Points and Parameters

This allows to quickly set all the necessary configuration points without having to use the HMI of the Room Controller at the time of installation. In this example, the points that are likely to be modified by the installer (or end-user) are set at priority 17 (relinquish default), while others are left at the default priority of 16.

Note: The script will set the included values each time the room controller starts.

```
if not init then          --Open INIT

    ME.MV6=2              --Set network units to Imperial
    ME.MV13=1             --Disable Dehum
    ME.MV70=2             --Display RH
    ME.AV87=0             --Heat Stages = 0 (Mod. Heat)
    ME.MV119=2            --Application= HPU
    ME.MV117=1            --Rev. Valve= O
    ME.MV16_PV[17]=2      --System Mode = Auto [with release]

    require 'math'                  --Opens the math functions library

    print "2021.10.18 R0"           --Print a message like Date and Revision in LUA Debug page

    heat=0  --variable
    cool=0  --variable

init=true                 --Close INIT
end


if ME.MV6_PV[3]~=2 then      --Once in a lifetime presets

    ME.MV51_PV[17]=2      --Display deg.F
    ME.MV2_PV[17]=3       --Display color = Blue
    ME.AV44_PV[17]=80     --Unocc COOLING SETPOINT = 80°F
    ME.AV43_PV[17]=66     --Unocc HEATING SETPOINT = 66°F
    ME.MV103_PV[17]=1     --Remove Chinese language
    ME.MV104_PV[17]=1     --Remove Russian language

ME.MV6_PV[3]=2  --value saved in EEPROM (remains after a power-cycle)
end
```

Custom Button Action

The section shows how the icon and functionality of the 5th button of the Room Controller's home screen (bottom right) can be changed via MV115. This specific example will change the logo of the 5th button to Lighting and set the function to No Function. It will then toggle physical point Binary Output 8 (BACnet point BO98) when the button is pressed.

Note: The last button press variable (AV92) must be reset to 0 in the script.

The below tables show the various buttons along with descriptions.

Image	Icon MV114	Icon Description
	1	Default Button
No Button	2	No Button
	3	System Mode Heat/Cool
	4	System Mode On/Off
	5	Fan Mode
	6	Override Button
	7	Units Button
	8	Help Button
	9	Language Button
	10	Schedule button
	11	Lighting Button
	12	Blind Button
	13	Lamp button
	14	Energy Button
	15	Make-Up Room Button
	16	Settings Button
	17	Timer Button

Function MV115	Button Function
1	Default Function
2	No Function
3	System Mode Function
4	Fan Function
5	Override Function
6	Schedule Function
7	Units Function
8	Help Function
9	Language Button
10	Configuration Function
11	Custom Function
12	Standby Function (display the long message or standby image)

```
if not init then          --Open INIT

    ME.MV114_PV[3]=11         --Set 5th button icon to "Lighting"
    ME.MV115=2                --Set 5th button function to "No function"

init=true                 --Close INIT
end


if ME.AV92==5 then        --If the 5th button is pressed then

    ME.BO98=(1-ME.BO98)       --BO8 output toggles between 0 and 1
    ME.AV92_PV[17]=0          --Sets the 5th button value back to 0 [released to be pressed again]

end
```

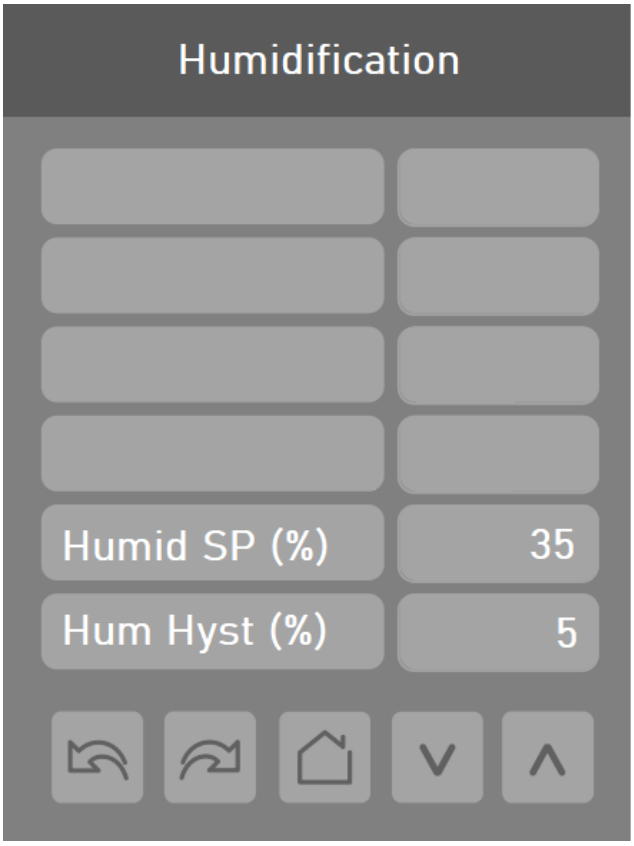
For reference purposes, the below shows the possible values for the last button pressed value (AV92):

- 1: Button 1 (lower-left)
- 2: Button 2
- 3: Button 3
- 4: Button 4
- 5: Button 5 (lower-right)
- 6: setpoint down-arrow
- 7: up-arrow
- 8: config (upper middle)
- 9: schedule (upper left or right corner)
- 10: another spot on the home screen
- 25 to 30: AV25..30 on the custom page
- 35: home button in custom page(s)

Configure Lua Parameter Page Title

This script will change the title of the Lua parameter page to “Humidification”. The name will appear when the custom button is pressed and requires the button function to be set to “custom”.

```
if not init then          --Open INIT

    ME.MV114_Desc= "Humidification" --Rename Custom page description
    ME.MV114_PV[3]=16               --Set 5th button to "Settings"
    ME.MV115=11                     --Set 5th button function to "Custom page"

    ME.AV25_Desc= " "
    ME.AV26_Desc= " "
    ME.AV27_Desc= " "
    ME.AV28_Desc= " "

    ME.AV29_Desc= "Humid SP (%)"    --Humidification Setpoint
    ME.AV29_Min=10
    ME.AV29_Max=75
    ME.AV29_Inc=5

    ME.AV30_Desc= "Hum Hyst (%)"    --Humidification Hysteresis
    ME.AV30_Min=5
    ME.AV30_Max=15

init=true                 --Close INIT
end
```



Remote Wired Humidity Sensor

All SE8000 Room Controllers have an internal Relative Humidity sensor. They can also use a remote wireless Zigbee sensor instead, but they do not have a dedicated input for a wired remote Relative Humidity sensor.

This LUA will configure UI24 Universal Input as 0-10 Vdc. Monitor the input to see if there is at least 0.5 Vdc (5% RH) on UI24. If that is the case, the LUA will automatically use this reading instead of the internal sensor (the LUA will update the value only if it is different from the previous value (COV)). If the reading is less than 0.2 Vdc (2% RH), the LUA will release the reading to the internal value as if the remote sensor is absent or defective.

```
--SE8000 Remote Relative Humidity sensor (RH Transmitter)
--(0-10 Vdc = 0-100% RH) on UI24 (auto-detect)


if not init then
ME.MV144=3       --UI24=Voltage
init=true
end


if ME.AV116 >= 0.5 then
    if ME.AV103~=ME.AV116*10 then
    ME.AV103=ME.AV116*10
    end
print "Use remote RH sensor"
end

if ME.AV116 < 0.2 then
ME.AV103=nil
print "Use internal RH sensor"
end
```

SE8650 Active Dehumidification

This LUA will do “active” dehumidification on a call for dehumidification, The Fan will be forced On as well as the Y1 and Y2 Cooling stages and W1 Heating stage. Dehumidification is also conditional to Occupancy, System Mode and Room Temperature.

```
--SE8650 RTU Dehum via Y1-Y2-W1 Documentation Example.lua

    --Dehumidification Setpoint: Use "Dehum SP" parameter in "Setpoints" menu


if not init then        --Open INIT

    ME.MV6=2                --Set network units to Imperial
    ME.MV13=1               --Disable Dehum
    ME.MV70=2               --Display RH
    ME.MV119=1              --Application= RTU
    ME.MV16_PV[17]=2        --System Mode = Auto [with release]
    print "2021.10.18 R0"   --Print a message like Date and Revision in LUA Debug page

init=true               --Close INIT
end


------------------------------------------------------------------------Once in a lifetime presets
if ME.MV6_PV[3]~=2 then

    ME.MV51_PV[17]=2    --Display deg.F
    ME.AV44_PV[17]=80   --Unocc COOLING SETPOINT = 80°F
    ME.AV43_PV[17]=66   --Unocc HEATING SETPOINT = 66°F

ME.MV6_PV[3]=2
end
```

```
------------------------------------------------------------------------------------- Dehum, ON conditions
--If (Occ or Override) and System Mode=(Auto or Cool) and Hum>(Hum_SP) and Room_Temp>(Occ_Heat_SP+0.5) and Cool_Demand<5%

if (ME.MSI33==1 or ME.MSI33==3) and (ME.MV16==2 or ME.MV16==3) and ME.AV103>ME.AV71 and ME.AV100>(ME.AV39+0.5) and ME.AO22<5 then
ME.BO25=1                       --G Fan = On
ME.BO26=1                       --Y1 = On
ME.BO27=1                       --Y2 = On
ME.BO28=1                       --W1 = On
ME.CSV1= "Deshumidification"    --Set CSV Dehum message to On
end


------------------------------------------------------------------------------------- Dehum, OFF conditions
--If (Unocc or Standby) or System Mode=(Off or Heat) or Room_Hum<(Hum_SP-Hum_Hyst.) or Room_Temp>(Occ_Heat_SP-0.5) or Cool_Demand>25%

if (ME.MSI33==2 or ME.MSI33==4) or (ME.MV16==1 or ME.MV16==4) or ME.AV103<(ME.AV71-ME.AV72) or ME.AV100<(ME.AV39-0.5) or ME.AO22>25 then
ME.BO28=nil     --W1 released to normal operation
ME.BO27=nil     --Y2 released to normal operation
ME.BO26=nil     --Y1 release to normal operation
ME.BO25=nil     --G Fan release to normal operation
ME.CSV1= " "    --Set CSV Dehum message to Off
end
```

Miscellaneous Examples

Reverse 0-100% input to 10-0 Vdc output

```
ME.AO123 = 10 - (ME.AO21/10)) -- Modulating based on Heating Demand
```

Convert a 0-100% input to 10-2 Vdc output

```
ME.AO123 = 10 - (0.8 * (ME.AO21/ 10)) -- Modulating based on Heating Demand
```

Reverse Y1 output

```
ME.BO26_PV[16] = (1 - ME.BO26_PV[17])
```

Average Room_Temp and Remote (use UI20 (RS) for the remote sensor)

```
if init == nil then
    ME.MV145 = 2    --Set Room_Temp sensor = Local (this line goes in the “init” section)
    init = true
end
```

```
ME.AV100_PV[16] = (ME.AV100_PV[17] + ME.AV105) / 2
```