# Mathematical Operations With Lua (FW version 1.7)

Mathematical operations can be applied to Lua signals as in any other protocol. This can be done by configuring WCC Lite according to solution needed. To create an example which would test multiple mathematical operations, Excel configuration and Lua script is required. Device sheet should look similar to this:

| name | device_alias | enable | protocol | execution_type | error_limit | ip | id | mode | scan_rate_ms | retry_count | timeout_ms | bind_address | host | port | asdu_size | cot_size | ioa_size | rwt | swt | t1 | t2 | t3 | time_sync | message_size | cache_size |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Modbus TCP device | Modbus_TCP | 1 | Modbus TCP | | | | 1 | tcp | 300 | 3 | 10000 | 0.0.0.0 | | 502 | | | | | | | | | | | |
| IEC104 SCADA system | IEC104_SCADA | 1 | IEC 60870-5-104 slave | | | | | | | | | 0.0.0.0 | | 2404 | 2 | 2 | 3 | 8 | 12 | 15 | 10 | 20 | 1 | 249 | 100 |
| LUA device | LUA0 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA1 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA2 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA3 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA4 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA5 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA6 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA7 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA8 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA9 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA10 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA11 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |
| LUA device | LUA12 | 1 | Lua runner | signal | 0 | | | | | | | | | | | | | | | | | | | | |

Each Lua device is created to send result values to Modbus TCP signals with different mathematical functions applied. This way the same Lua script can be reused and is more optimal since signal alias for each device can stay the same. There are many other solutions but this one allows to observe results more clearly.

In the field  "ip" for Modbus TCP master, enter IP address of Wi-Fi connection for computer in use. In the field "host" for IEC104 slave protocol enter IP address of WCC Lite device.

Signals for these devices should be mapped in example to this:

| signal_name | device_alias | signal_alias | source_device_alias | source_signal_alias | execute | enable | multiply | add | bit_select | min_value | max_value | absolute_threshold | threshold_units | suppression_values | suppression_time_ms | gi | log | number_type | job_todo | tag_job_todo | common_address | info_address | data_type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Result_modbus0 | Modbus_TCP | result0 | LUA0 | result | | 1 | | | | | | | | | | | 1 | SIGNED16 | 3;0;1 | 3;0;1 | | | |
| command LUA | LUA0 | command | IEC104_SCADA | Command0 | 1 | | | | | | | | | | | | 1 | | | | | | |
| result LUA | LUA0 | result | | | | | | | | | | | | | | | 1 | | | | | | |
| Command IEC104.0 | IEC104_SCADA | Command0 | LUA0 | command | | 1 | | 5 | | | | | | | | 1 | 1 | | | | 1 | 1 | 50 |
| Result_modbus1 | Modbus_TCP | result1 | LUA1 | result | | 1 | | | | | | | | | | | 1 | SIGNED16 | 3;1;1 | 3;1;1 | | | |
| command LUA | LUA1 | command | IEC104_SCADA | Command1 | 1 | | | | | | | | | | | | 1 | | | | | | |
| result LUA | LUA1 | result | | | | | | | | | | | | | | | 1 | | | | | | |
| Command IEC104.1 | IEC104_SCADA | Command1 | LUA1 | command | | 1 | | -5 | | | | | | | | 1 | 1 | | | | 1 | 2 | 50 |
| Result_modbus2 | Modbus_TCP | result2 | LUA2 | result | | 1 | | | | | | | | | | | 1 | SIGNED16 | 3;2;1 | 3;2;1 | | | |
| command LUA | LUA2 | command | IEC104_SCADA | Command2 | 1 | | | | | | | | | | | | 1 | | | | | | |
| result LUA | LUA2 | result | | | | | | | | | | | | | | | 1 | | | | | | |
| Command IEC104.2 | IEC104_SCADA | Command2 | LUA2 | command | | 1 | 5 | | | | | | | | | 1 | 1 | | | | 1 | 3 | 50 |
| Result_modbus3 | Modbus_TCP | result3 | LUA3 | result | | 1 | | | | | | | | | | | 1 | SIGNED16 | 3;3;1 | 3;3;1 | | | |
| command LUA | LUA3 | command | IEC104_SCADA | Command3 | 1 | | | | | | | | | | | | 1 | | | | | | |
| result LUA | LUA3 | result | | | | | | | | | | | | | | | 1 | | | | | | |
| Command IEC104.3 | IEC104_SCADA | Command3 | LUA3 | command | | 1 | | | 1 | | | | | | | 1 | 1 | | | | 1 | 4 | 50 |
| Result_modbus4 | Modbus_TCP | result4 | LUA4 | result | | 1 | | | | | | | | | | | 1 | SIGNED16 | 3;4;1 | 3;4;1 | | | |
| command LUA | LUA4 | command | IEC104_SCADA | Command4 | 1 | | | | | | | | | | | | 1 | | | | | | |
| result LUA | LUA4 | result | | | | | | | | | | | | | | | 1 | | | | | | |
| Command IEC104.4 | IEC104_SCADA | Command4 | LUA4 | command | | 1 | 6 | | 2 | | | | | | | 1 | 1 | | | | 1 | 5 | 50 |
| Result_modbus5 | Modbus_TCP | result5 | LUA5 | result | | 1 | | | | | | | | | | | 1 | SIGNED16 | 3;5;1 | 3;5;1 | | | |
| command LUA | LUA5 | command | IEC104_SCADA | Command5 | 1 | | | | | | | | | | | | 1 | | | | | | |
| result LUA | LUA5 | result | | | | | | | | | | | | | | | 1 | | | | | | |
| Command IEC104.5 | IEC104_SCADA | Command5 | LUA5 | command | | 1 | | 2 | 3 | | | | | | | 1 | 1 | | | | 1 | 6 | 50 |
| Result_modbus6 | Modbus_TCP | result6 | LUA6 | result | | 1 | | | | | | | | | | | 1 | SIGNED16 | 3;6;1 | 3;6;1 | | | |
| command LUA | LUA6 | command | IEC104_SCADA | Command6 | 1 | | | | | | | | | | | | 1 | | | | | | |
| result LUA | LUA6 | result | | | | | | | | | | | | | | | 1 | | | | | | |
| Command IEC104.6 | IEC104_SCADA | Command6 | LUA6 | command | | 1 | 5 | 34 | 4 | | | | | | | 1 | 1 | | | | 1 | 7 | 50 |
| Result_modbus7 | Modbus_TCP | result7 | LUA7 | result | | 1 | | | | | | | | | | | 1 | SIGNED16 | 3;7;1 | 3;7;1 | | | |
| command LUA | LUA7 | command | IEC104_SCADA | Command7 | 1 | | | | | | | | | | | | 1 | | | | | | |
| result LUA | LUA7 | result | | | | | | | | | | | | | | | 1 | | | | | | |
| Command IEC104.7 | IEC104_SCADA | Command7 | LUA7 | command | | 1 | | | | 2 | 20 | | | | | 1 | 1 | | | | 1 | 8 | 50 |

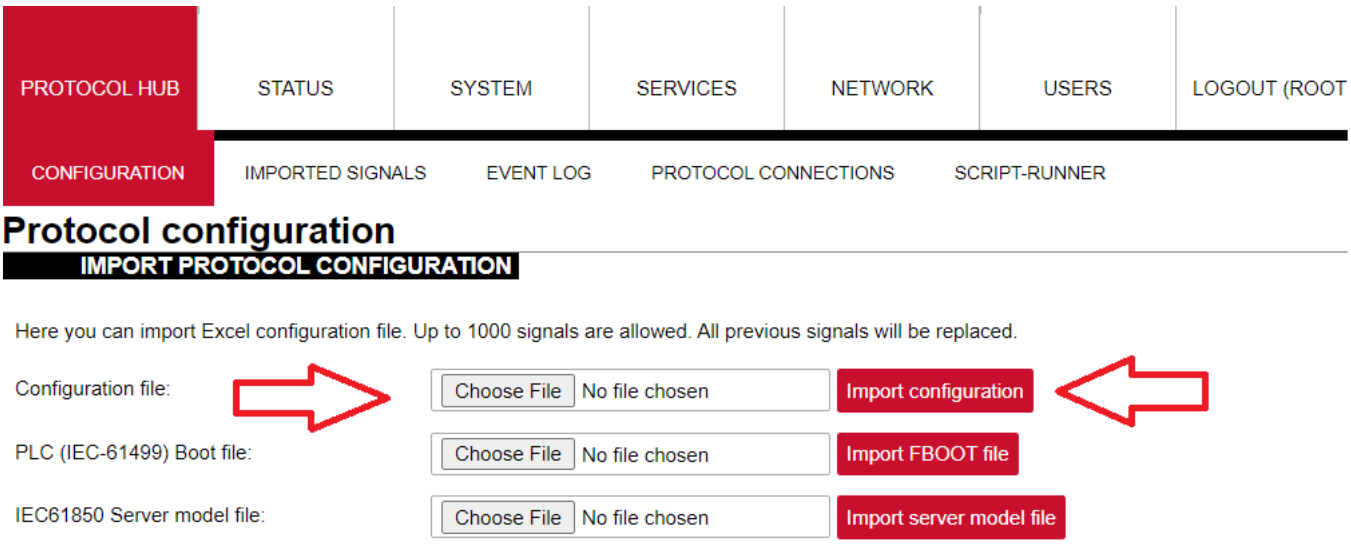
Each Lua device has command and result signals. Command received from IEC104 protocol is sent to Lua command signal and then this signal sends back a response for IEC104 protocol. If the response does not have negative cot

attributes, value is then sent to Lua result signal which sends value to Modbus TCP result signal. Mathematical operations are applied to IEC104 protocol signals since it is the one sending the commands.

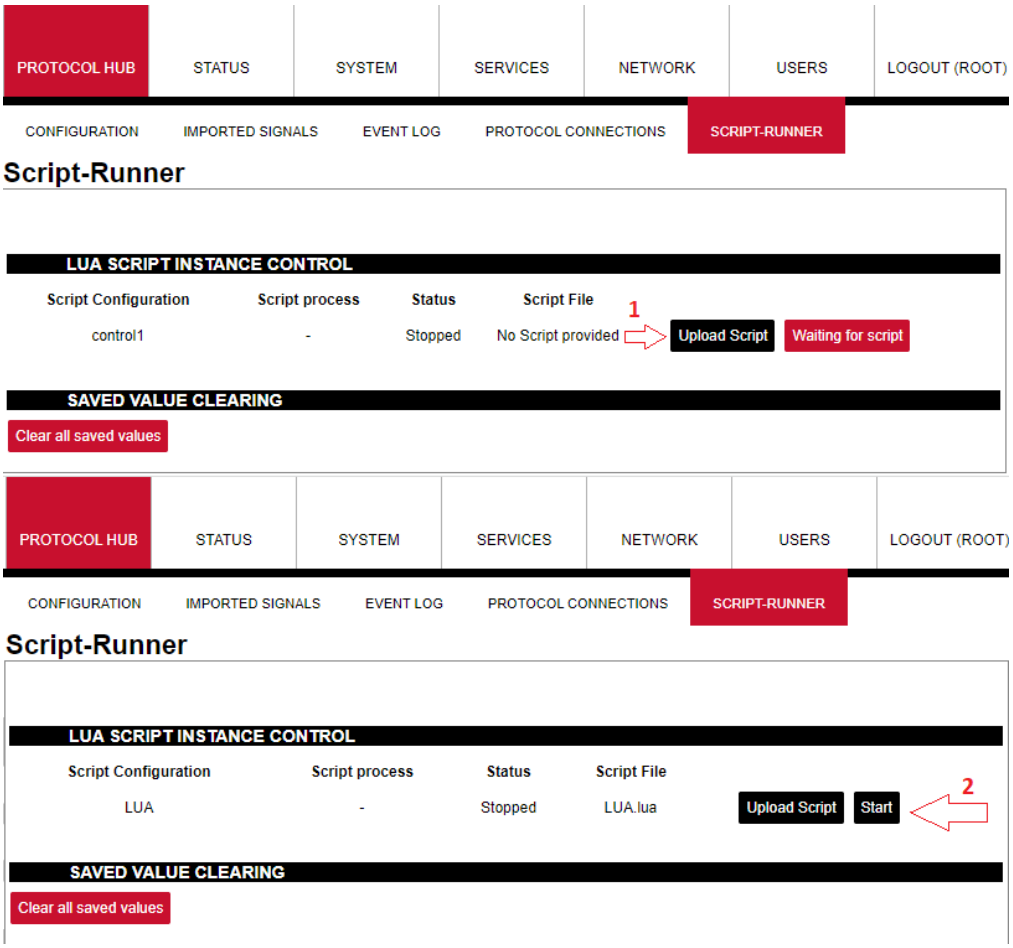
As mentioned before Lua script for each Lua device is going to be unchanged and should look like this:

```
local saved = get(signals.result) --getting result signal which is equated to new variable 'saved'
local command = get(signals.command) --getting command signal which is equated to new variable 'command'

--get() function returns nill if there is no valid value

if not command then --if command is not nill
    if saved then --if signal is not nill
        publish(signals.result, saved.value) --this value is published to result signals and saved value
    end
    return 0
end


local time_diff = time_ms() - tonumber(command.time) --compares command time and real time
local is_command = time_diff < 30000 and time_diff > -30000 --if command time differs from
--real time more than 30s it will not be executed


if string.find(command.attributes, "nt") or string.find(command.attributes,
    "iv") or string.find(command.attributes, "ov") then
  --searching if signal has negative attributes
    if is_command then --if command execution time is not exceeding the limits then
        command.attributes = "cot=7,cotn" --equates negative cot values to response signal attributes
        publish(signals.command, command) -- and publishes value to command signals and value
        if saved then --if there is saved value then
            publish(signals.result, saved.value) --restores saved value to result signals
        end
        return 0
    end
else
    if is_command then
        command.attributes = "cot=7"
        publish(signals.command, command)   --in this cycle command value is being returned as well as
    --cot7 and cot10 values in case given signal is command and has no negative attributes
        command.attributes = "cot=10"
        publish(signals.command, command) --publishes response to the command
        save(signals.result, command.value) --command value is being saved to result signal
    end
    publish(signals.result, command.value) --in this row command value is being published to result signals
end
```

Upload Excel configuration to WCC Lite:




After uploading configuration no errors should appear and all signal should be represented on the web. To upload Lua script go to *Script-Runner*, select *upload script* and then *start* (for each Lua device):




Open Vinci as IEC104 master, enter IP address of WCC Lite and start communication. Then open another Vinci window and connect Modbus TCP master – select Modbus TCP slave in Vinci and enter the same IP address as set in Excel configuration for Modbus device. With both communications running check **Protocol connections** on WCC Lite web interface, it should show *Connected*. From IEC104 Vinci window go to System tab. Select command determined in the Excel configuration (50), IOA (different for each signal) and value. After executing command, each signal (IEC104 command, Lua command, Lua result and Modbus result) will have the same value, which now will be with math applied. For example, command with IOA=1 and value 1 is being executed. In Excel configuration for this signal **add** column has a value of 5, which means that this value is going to be added to the value sent and the result will be 6.

There can be multiple mathematical operations for one signal. For example add, multiply, bit select etc. If that is the case, math will be applied in typical order (eg. first bit select, then multiply, then add). More detailed explanation about mathematical operations in Excel configuration can be found here: Optional parameters for signals

A user can also apply mathematical condition for the signal value. For example minimum or maximum value, threshold, suppression time for specific value etc. Minimum and maximum values can be applied to set the range of the signal, if the value is smaller or larger signal state will show invalid or overflow. Thresholds can be used in many ways. It can be a specific value or a percentage. If the signal value passes set threshold it will be represented on imported signals window. Threshold works by comparing old value with new value and then applying the condition of either representing the value or suppressing it, depending on the value change. Suppression value and suppression time is best used together, because suppression time determines how long the specific value should be suppressed. There could be multiple values set for suppression. In Excel configuration those values should be separated by comma.

Mathematical operations combined with Lua script is useful for may cases. They can be used for filtering data, converting units, applying specific mathematical logic or other solutions.

Configuration --> Download

Lua script --> Download

Revision #7
★Created 10 August 2023 10:19:48 by Gabriele
Updated 10 December 2024 08:15:16 by Gabriele