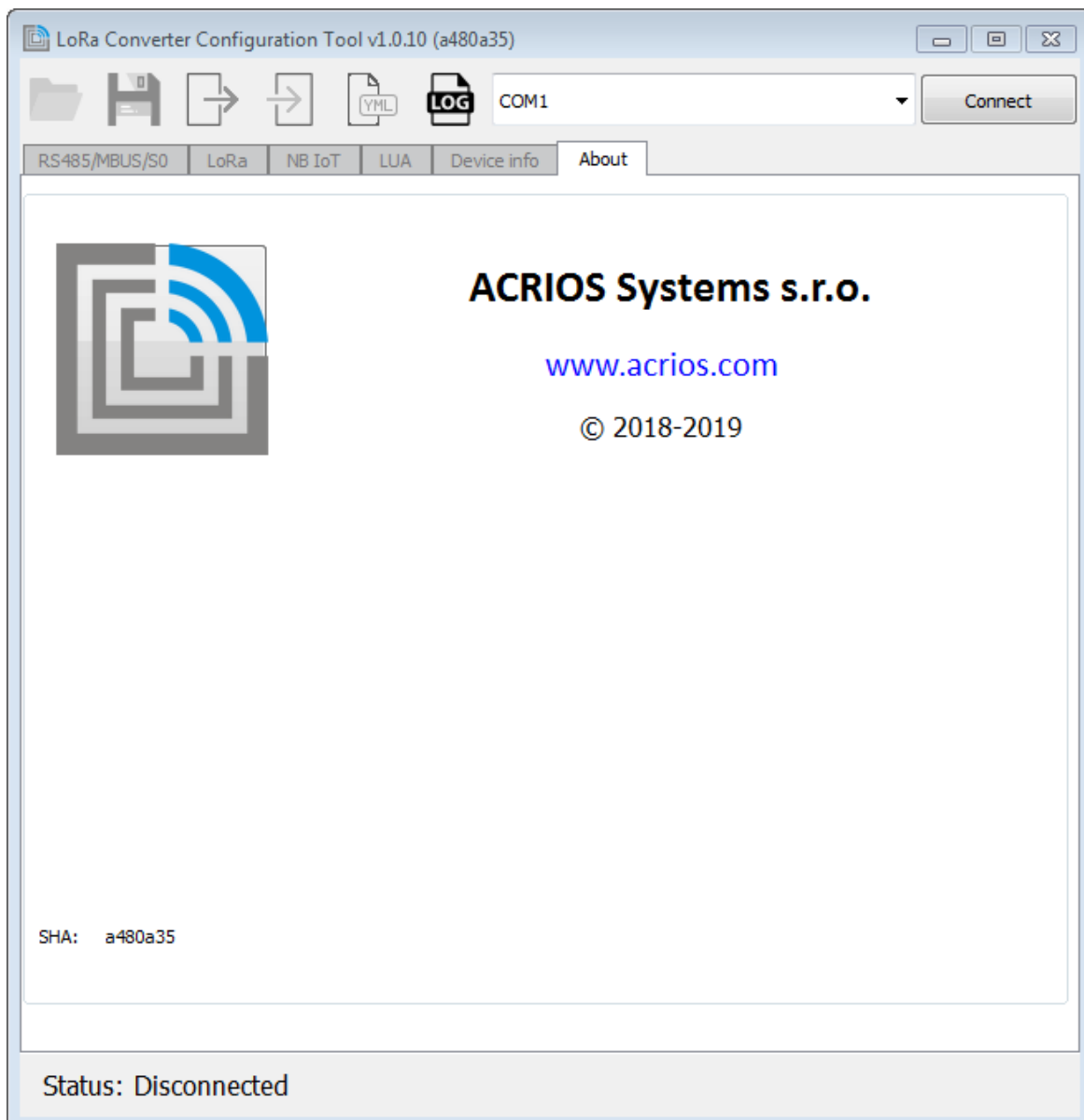


# MBUS - NB-IoT converter Configuration SW v1.0.10

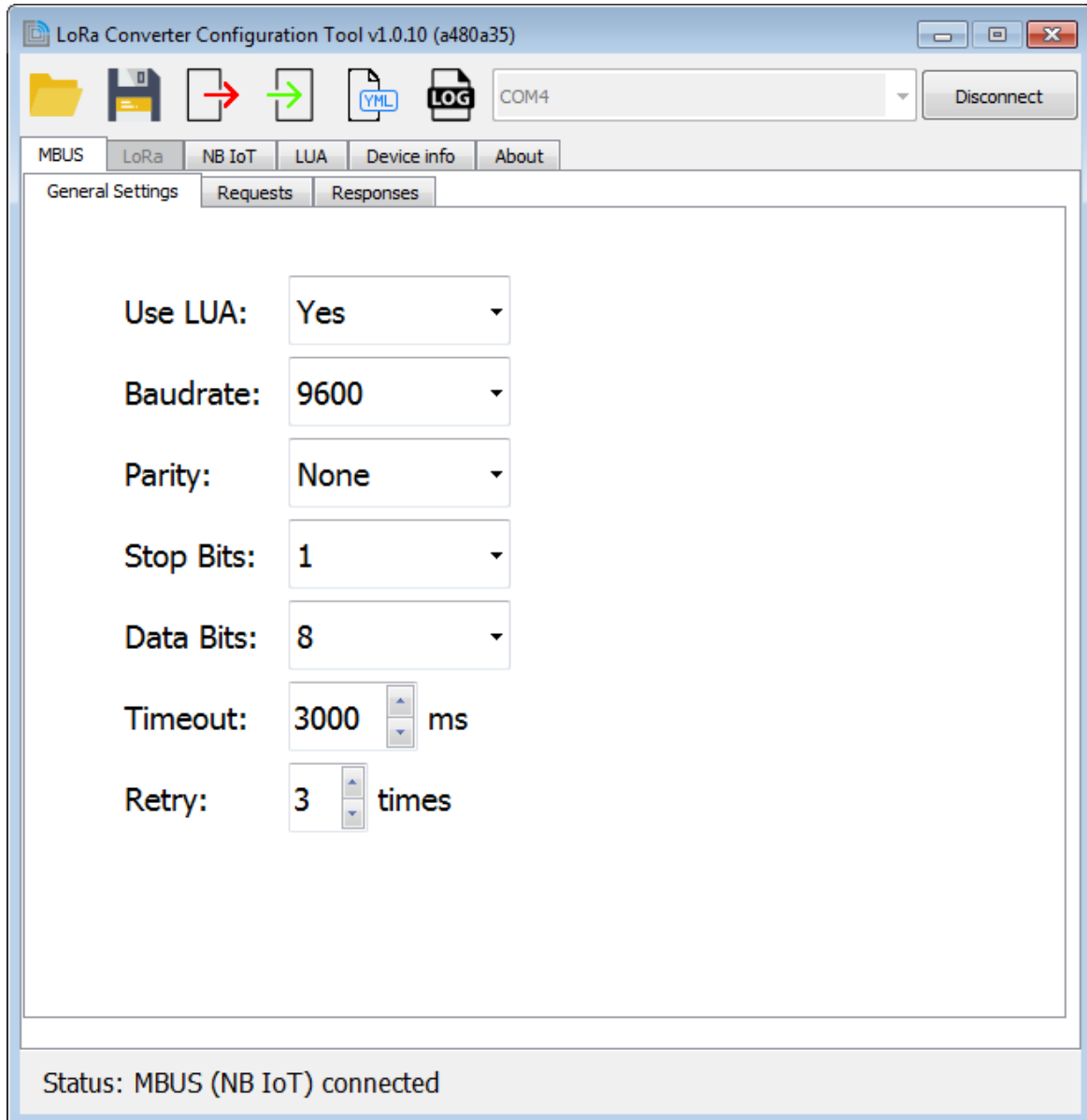
## Connecting to a converter

Converter is connected to a PC via USB/UART converter (3.5 mm jack connector).

After selecting port of USB/UART converter and clicking „Connect“ the connection with LoRa converter should be established (it may take a few seconds if converter is taking a measurement).

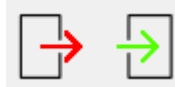


Current configuration of converter will be loaded upon connecting and GUI defaults to MBUS page:



## Buttons description

### Save and load configuration into/from a converter



Current configuration of converter can be saved into a EEPROM memory and loaded from it.

### Save and load configuration into/from a file



Current configuration of converter can be saved into a text file (.cfg) in JSON format and loaded from it.

```
{
  "connection": {
    "baudrate": 9600,
    "data_bits": 8,
    "parity": "None",
    "retry": 3,
    "stop_bits": 1,
    "timeout": 3000,
    "use_lua": "Yes"
  },
  "device_info": {},
  "lua": "function onwake ()\n  buf, err, proto, wake, ip, port,\n  ctx = api.getGUIContext()\n  if err ~= 0 then\n    print\n    (\"Error occured on line\" .. tostring(err))\n    print(\"Sending\n    error code to NB-IoT\")\n    api.nbSend(ip,port, \"ERROR: \" ..\n    tostring(err), 5000, proto)\n    print(\"Done sending\")\n  else\n    print(\"Sending to NB-IoT\")\n    api.nbSend(ip,port,\n    buf, 5000, proto)\n    print(\"Done sending\")\n    print(\"No\n    error, sent to NB-IoT\")\n    end\n    api.wakeupIn(0,0,wake,0)\nend\n\nfunction onStartUp()\n  print(\"Starting up NB-IoT default script!\n  \")\nend\n",
  "nbiot": {
    "ip": "1.2.3.4",
    "port": 5566,
    "protocol": 0
  },
  "period": 10,
  "requests": [
    "1011223316"
  ],
  "responses": [
    [
      0,
      0,
      4
    ]
  ]
}
```

## Save configuration into a YAML file



Current configuration of converter can be saved into a YAML file (.yaml) with prepared individual configuration frames. User can select only needed frames and send them into a converter.

```
# #####  
# General settings  
# Baudrate: 9600 (80250000, Byte 7:10)  
# Parity: None (00, Byte 11)  
# Stop bits: 1 (01, Byte 12)  
# Timeout: 3000 (B80B, Byte 13:14)  
# Retry: 3 (03, Byte 15)  
# Data bits: 8 (08, Byte 16)  
# #####  
settings:  
434F4E4649470180250000001B80B030875FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF  
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

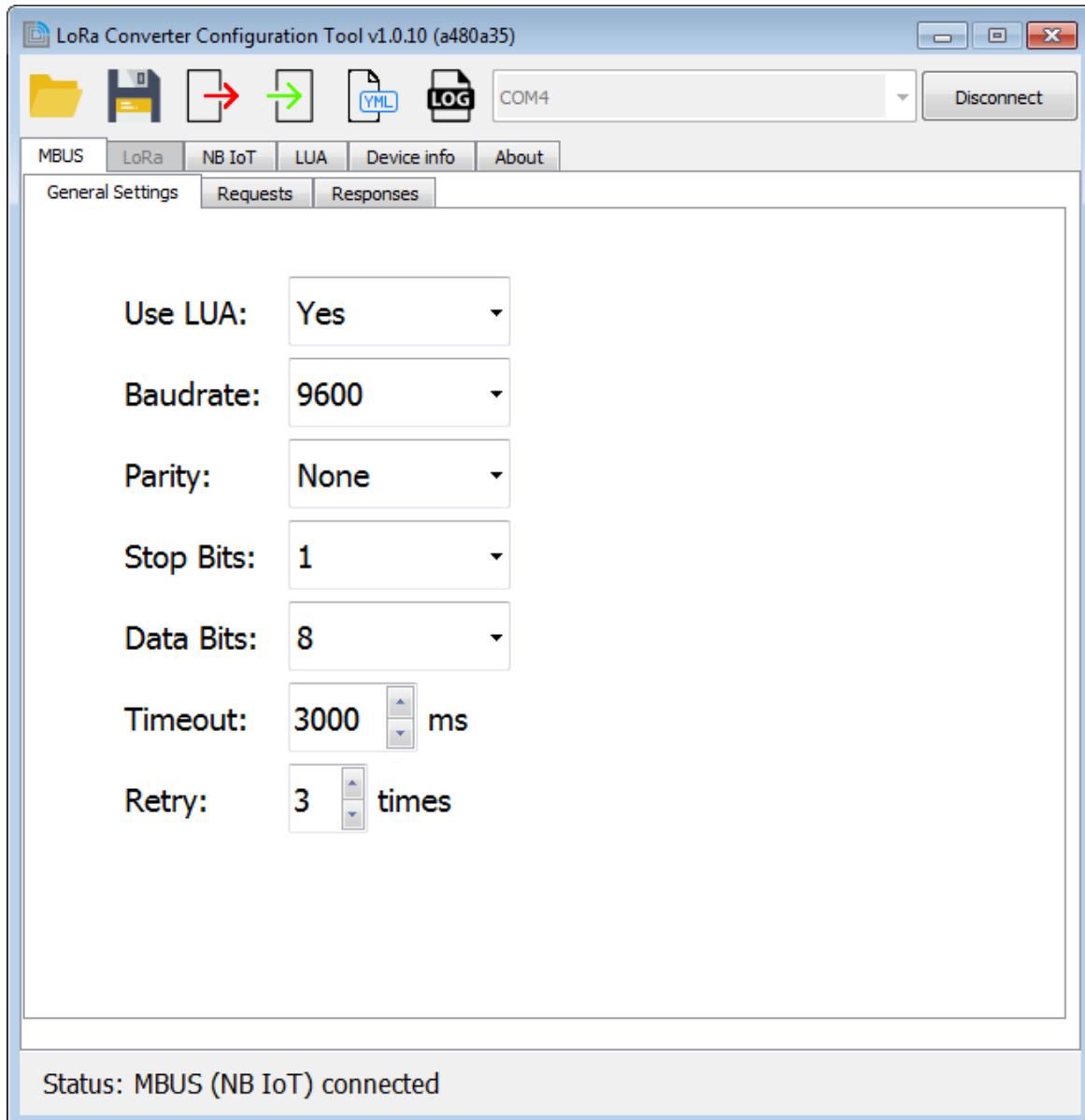


## Tabs description

### MBUS setting

#### General setting

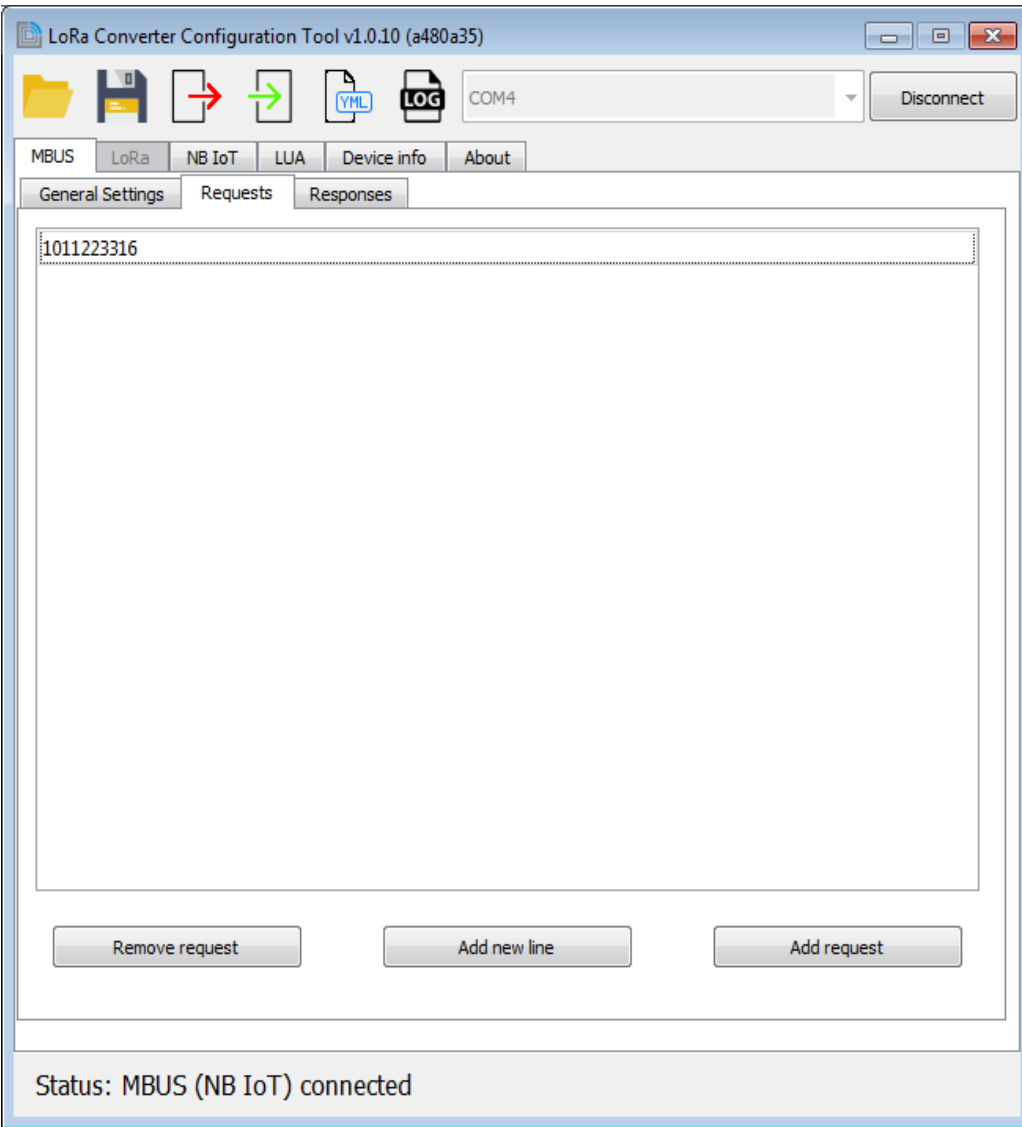
Settings of communication protocol with MBUs device and converter:



## List of MBUS requests

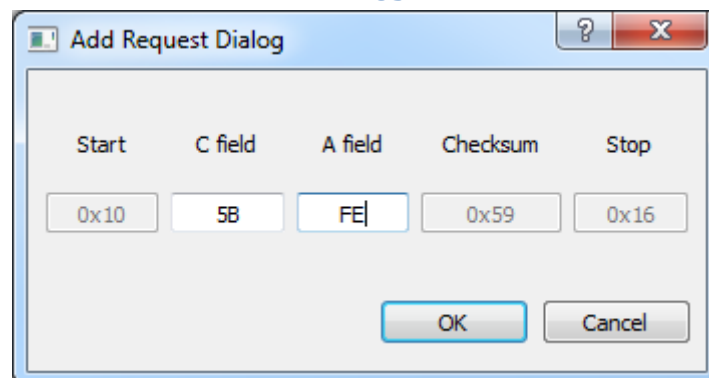
List of MBUs request to be made from converter to a MBUS device:

Request can be typed in by hand („Address new line“) or by dialog wizard („Address request“):



The screenshot shows the 'LoRa Converter Configuration Tool v1.0.10 (a480a35)' window. The 'Requests' tab is selected, displaying a list with the request '1011223316'. Below the list are three buttons: 'Remove request', 'Add new line', and 'Add request'. The status bar at the bottom indicates 'Status: MBUS (NB IoT) connected'.

### MBUS



The 'Add Request Dialog' window shows the following fields and values:

Start	C field	A field	Checksum	Stop
0x10	5B	FE	0x59	0x16

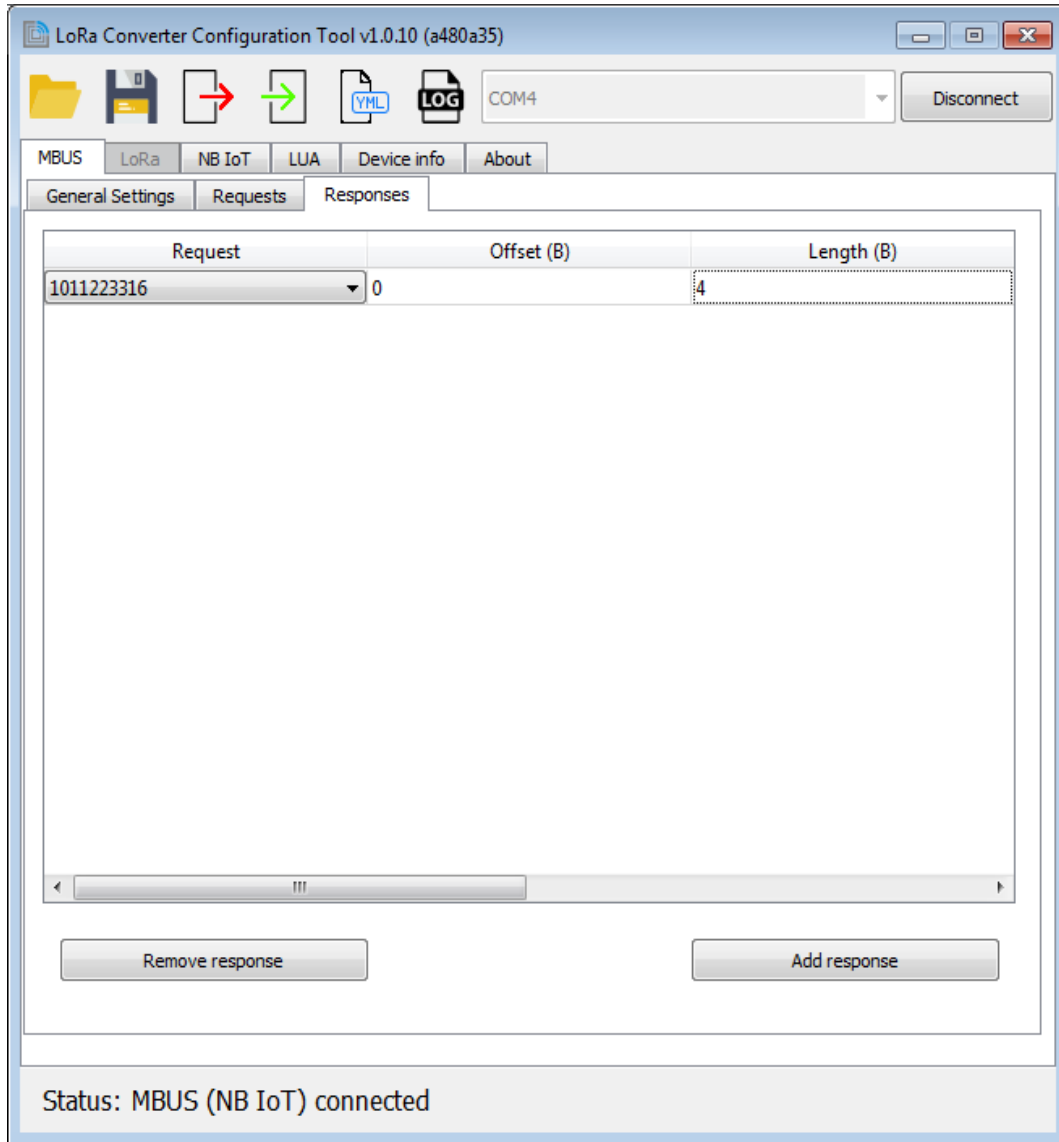
Buttons: OK, Cancel

Dialog automatically calculates checksum and fills in start and stop bytes.

## List of responses

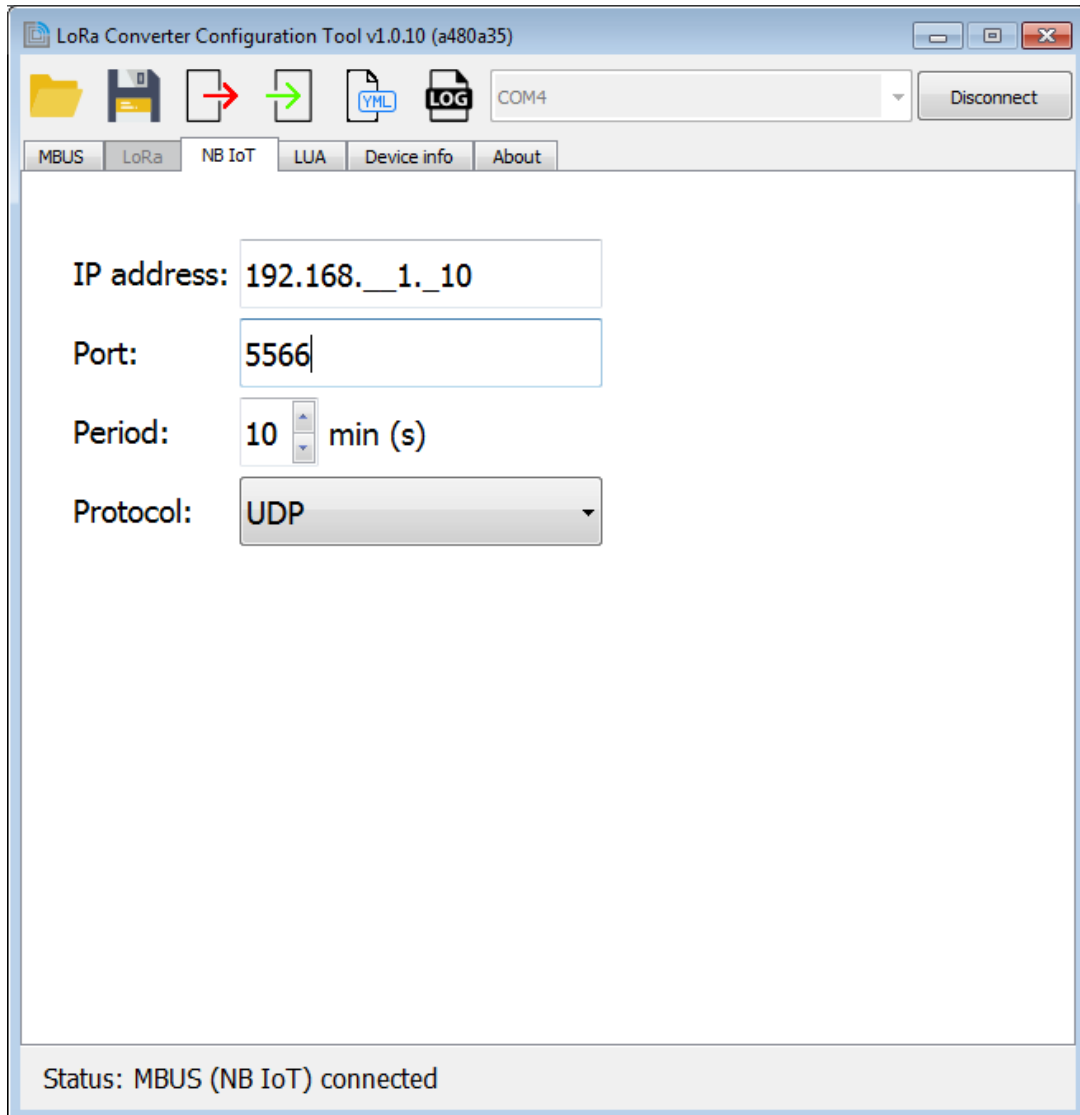
User can set different offsets and length which are taken from responses to selected requests.

E.g.: for first request (1011223316) take data of length 4 bytes with offset 0.



## NB-IoT settings

User can set host IP address, port, communication protocol and period of sending a data.



The screenshot shows the 'LoRa Converter Configuration Tool v1.0.10 (a480a35)' window. The 'NB IoT' tab is selected. The settings are as follows:

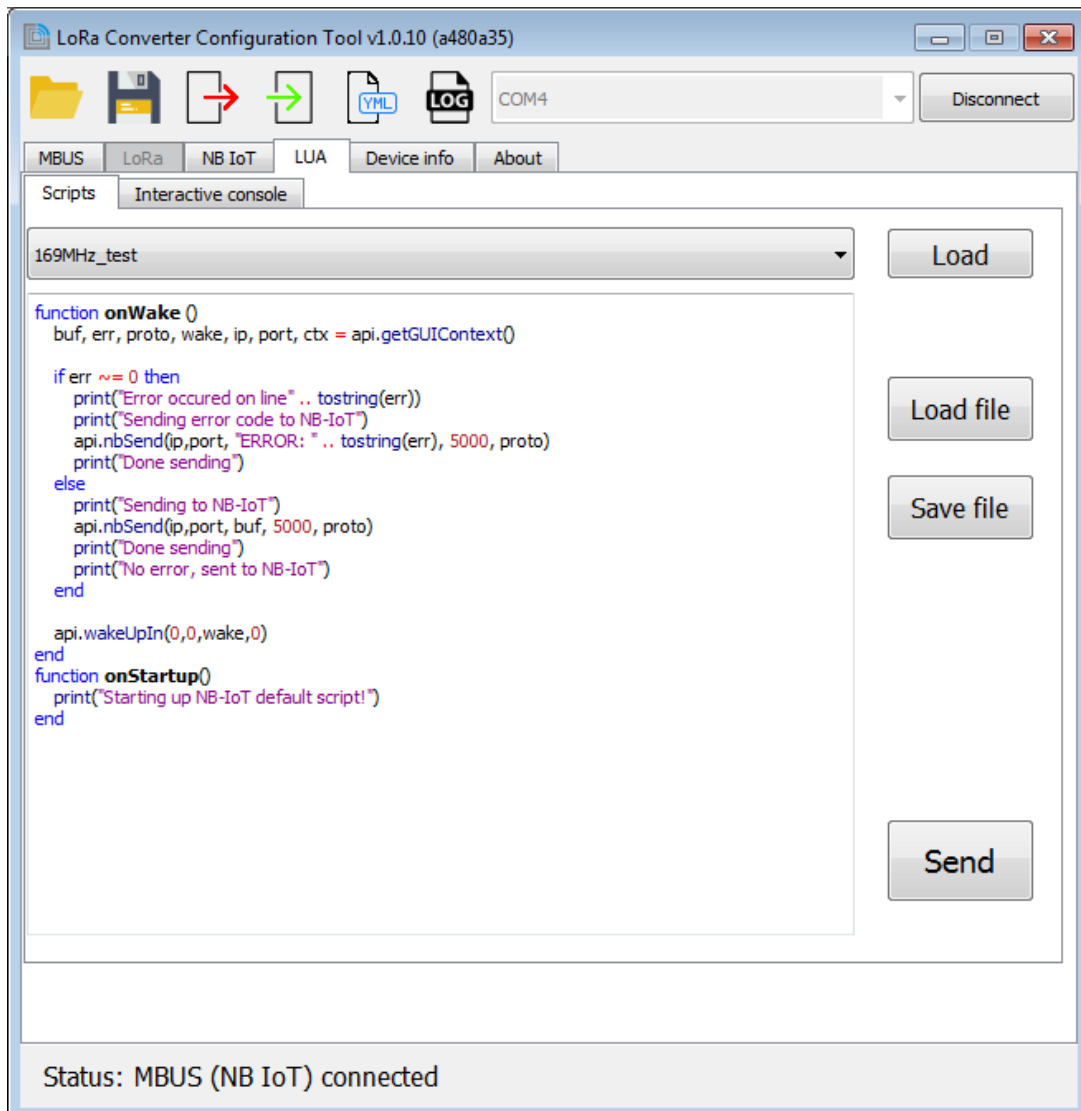
- IP address: 192.168.\_\_1\_\_10
- Port: 5566
- Period: 10 min (s)
- Protocol: UDP

Status: MBUS (NB IoT) connected



## LUA interface

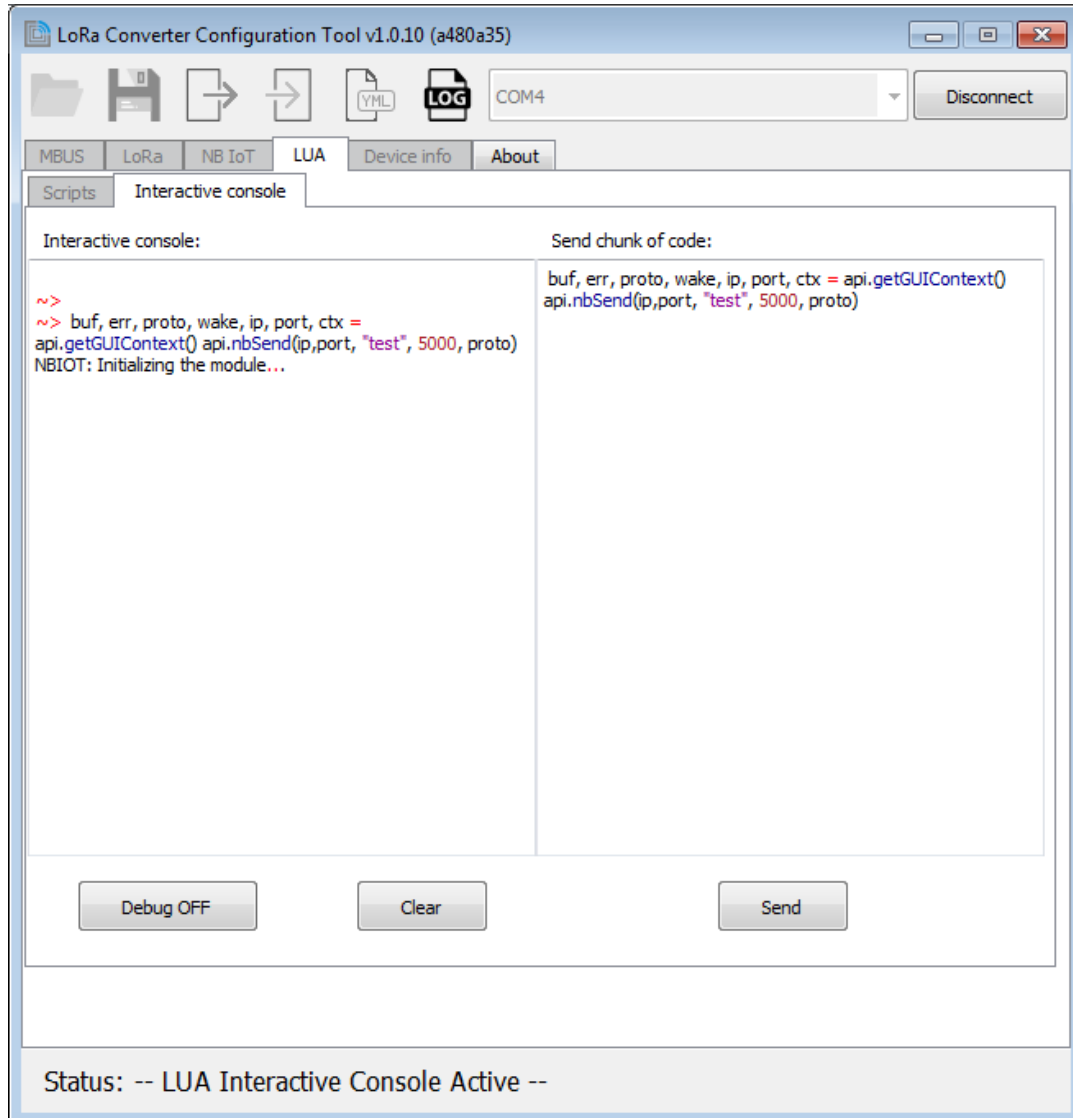
LUA interface adds ability to program converter with user's script with use of converter's API (separate documentation).



User can either select from available scripts (from combo box), load script from a file (.lua), edit loaded script and save it. Or user can type in own custom script.

Script is send to a converter by "Send" button or by saving configuration into a converter.

Another way of using LUA interface is interactive console where can user type LUA commands in real time. It is useful mainly for testing purposes.



Interactive console is activated with button “Debug ON” (can be exited from with “Debug OFF” or “Disconnect” buttons).

Left panel is for real time Lua scripting (e.g.: everything typed in is immediately send to converter’s Lua interface) and right panel is for sending a few lines of Lua code. Responses are seen in left panel.

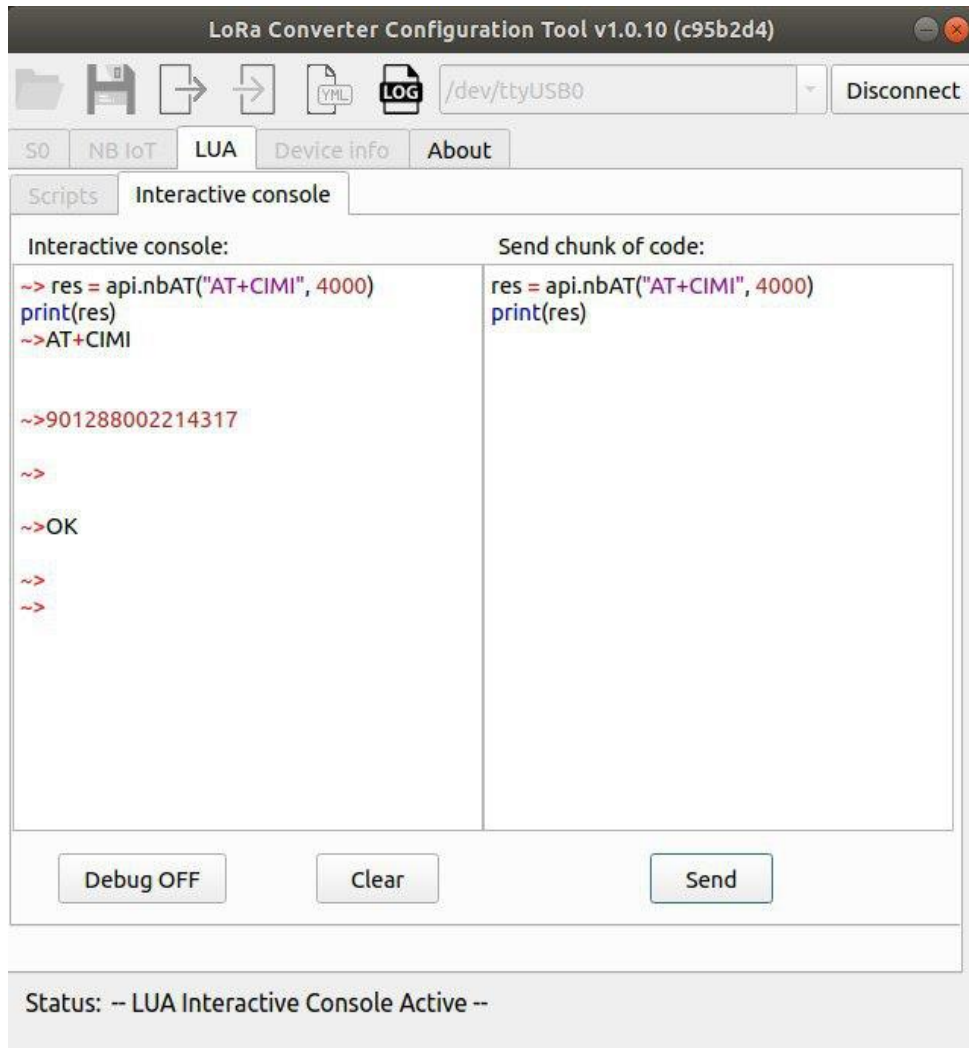
## Example usage of Lua interface

Lua interactive console is great for communication with NB IoT module. It supports AT commands via api call:

```
api.nbAT()
```

### IMSI SIM card number:

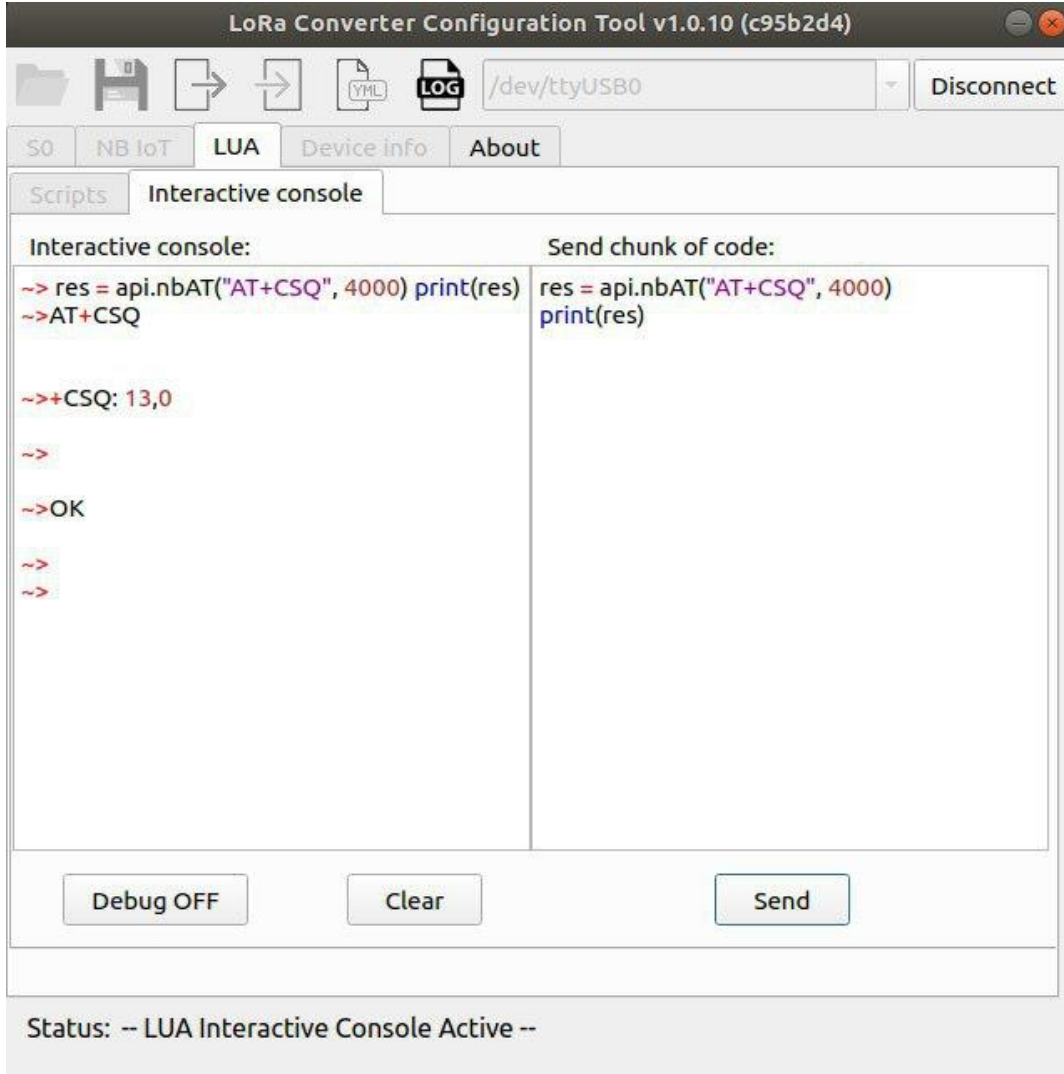
Command: *AT+CIMI*



### Quality of signal:

Command: AT+CSQ

Response: 13 means 85 dBm RSSI (normal values 9-14, bigger number = stronger signal)



The screenshot shows the 'LoRa Converter Configuration Tool v1.0.10 (c95b2d4)' interface. The 'LUA' tab is active, and the 'Interactive console' is open. The console shows the following interaction:

```
~> res = api.nbAT("AT+CSQ", 4000) print(res)
~> AT+CSQ

~> +CSQ: 13,0

~>

~> OK

~>

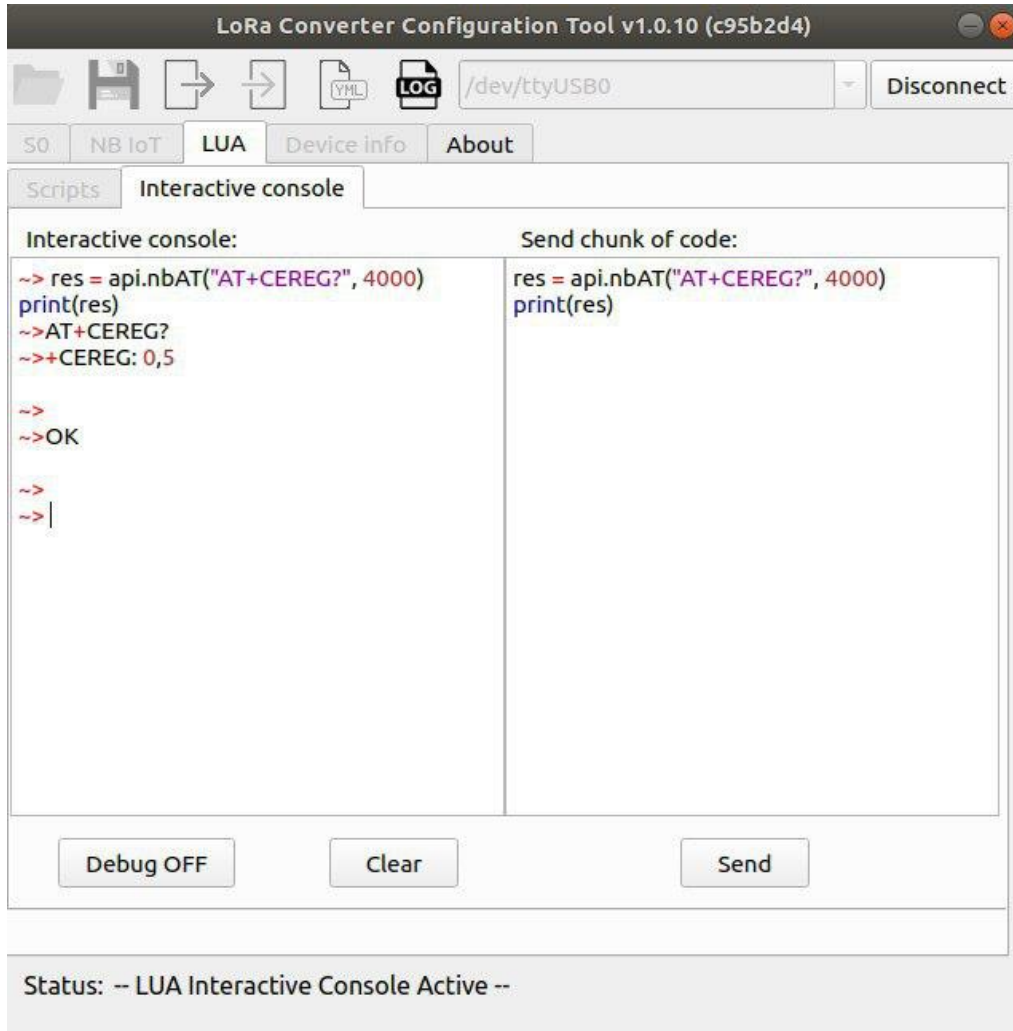
~>
```

Buttons at the bottom of the console include 'Debug OFF', 'Clear', and 'Send'. The status bar at the bottom indicates: 'Status: -- LUA Interactive Console Active --'.

## Status of network registration

Command: AT+CEREG?

Response: 5 means that module is registered into a roaming network



The screenshot shows the 'LoRa Converter Configuration Tool v1.0.10 (c95b2d4)' interface. The 'Interactive console' tab is active, displaying the following Lua code and its output:

```
Interactive console:
--> res = api.nbAT("AT+CEREG?", 4000)
print(res)
--> AT+CEREG?
--> +CEREG: 0,5

-->
--> OK

-->
--> |
```

The 'Send chunk of code:' area contains the same Lua code:

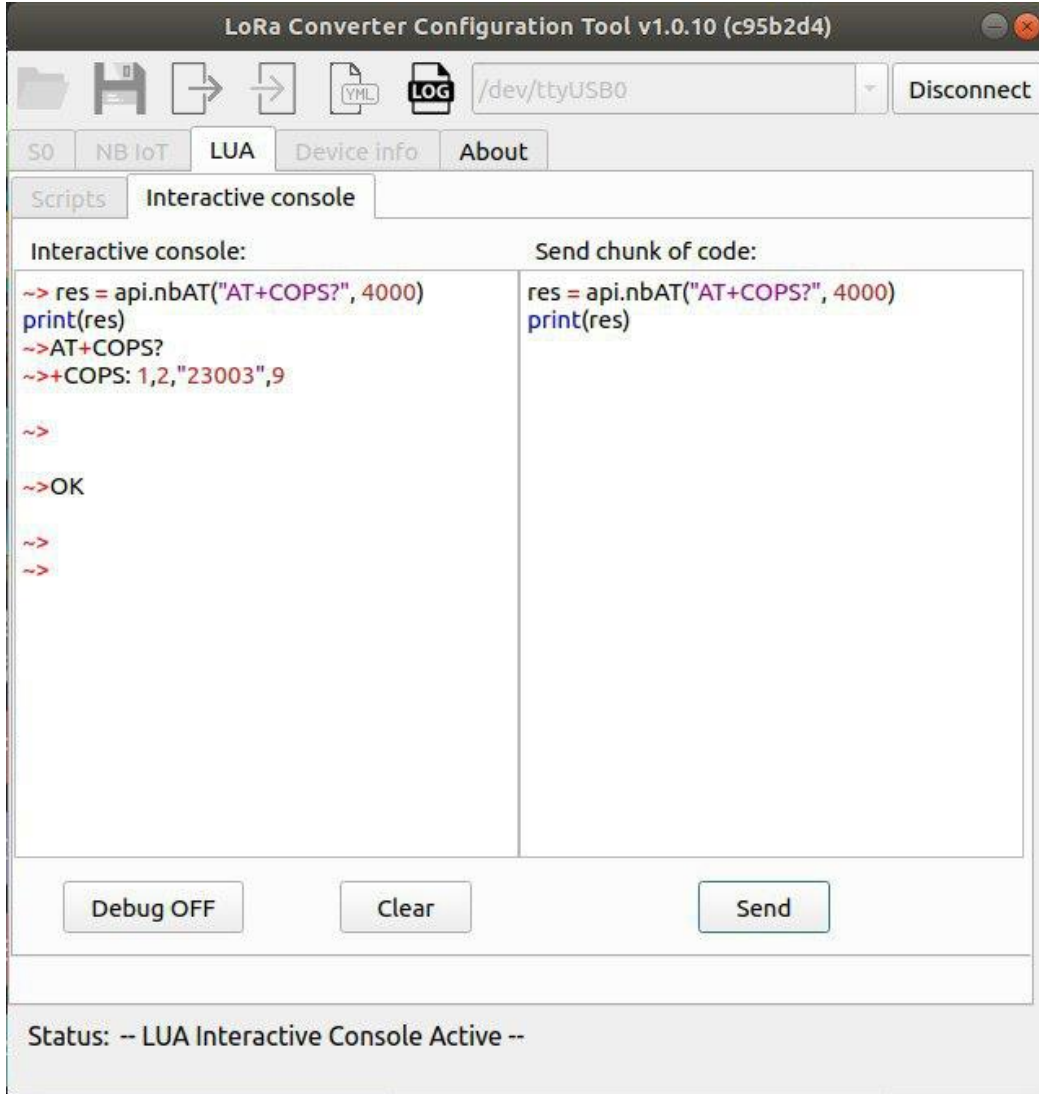
```
Send chunk of code:
res = api.nbAT("AT+CEREG?", 4000)
print(res)
```

At the bottom of the console, there are buttons for 'Debug OFF', 'Clear', and 'Send'. The status bar at the bottom of the window reads: 'Status: -- LUA Interactive Console Active --'.

**Current operator:**

Command: AT+COPS?

Response: 23003 is unique operator identifier(in this case Vodafone CZ)



```
LoRa Converter Configuration Tool v1.0.10 (c95b2d4)
/dev/ttyUSB0 Disconnect
S0 NB IoT LUA Device info About
Scripts Interactive console
Interactive console:
~> res = api.nbAT("AT+COPS?", 4000)
print(res)
~> AT+COPS?
~> +COPS: 1,2,"23003",9
~>
~> OK
~>
~>
Send chunk of code:
res = api.nbAT("AT+COPS?", 4000)
print(res)
Debug OFF Clear Send
Status: -- LUA Interactive Console Active --
```