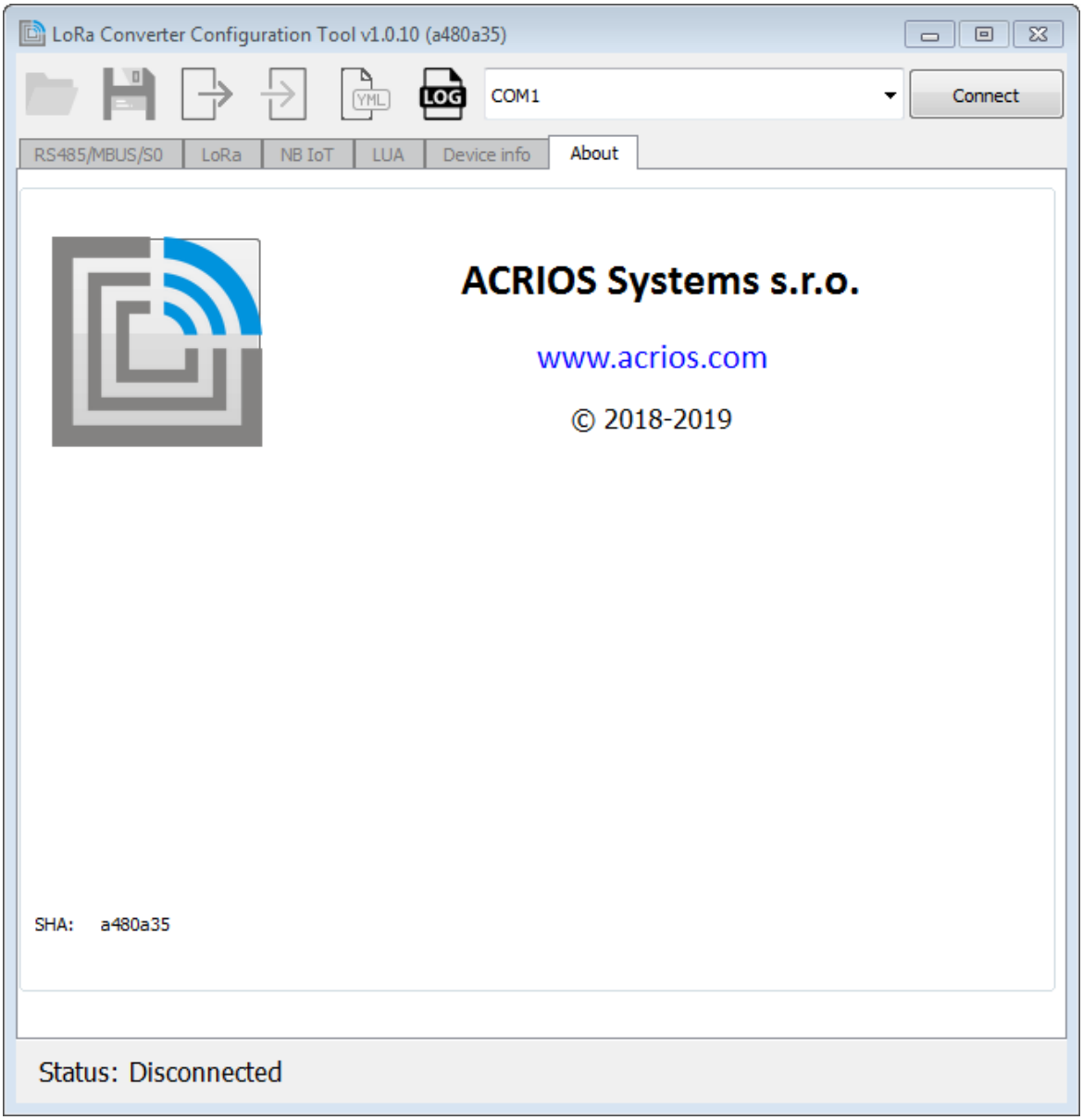
MBUS - NB-IoT converter Configuration SW v1.0.10

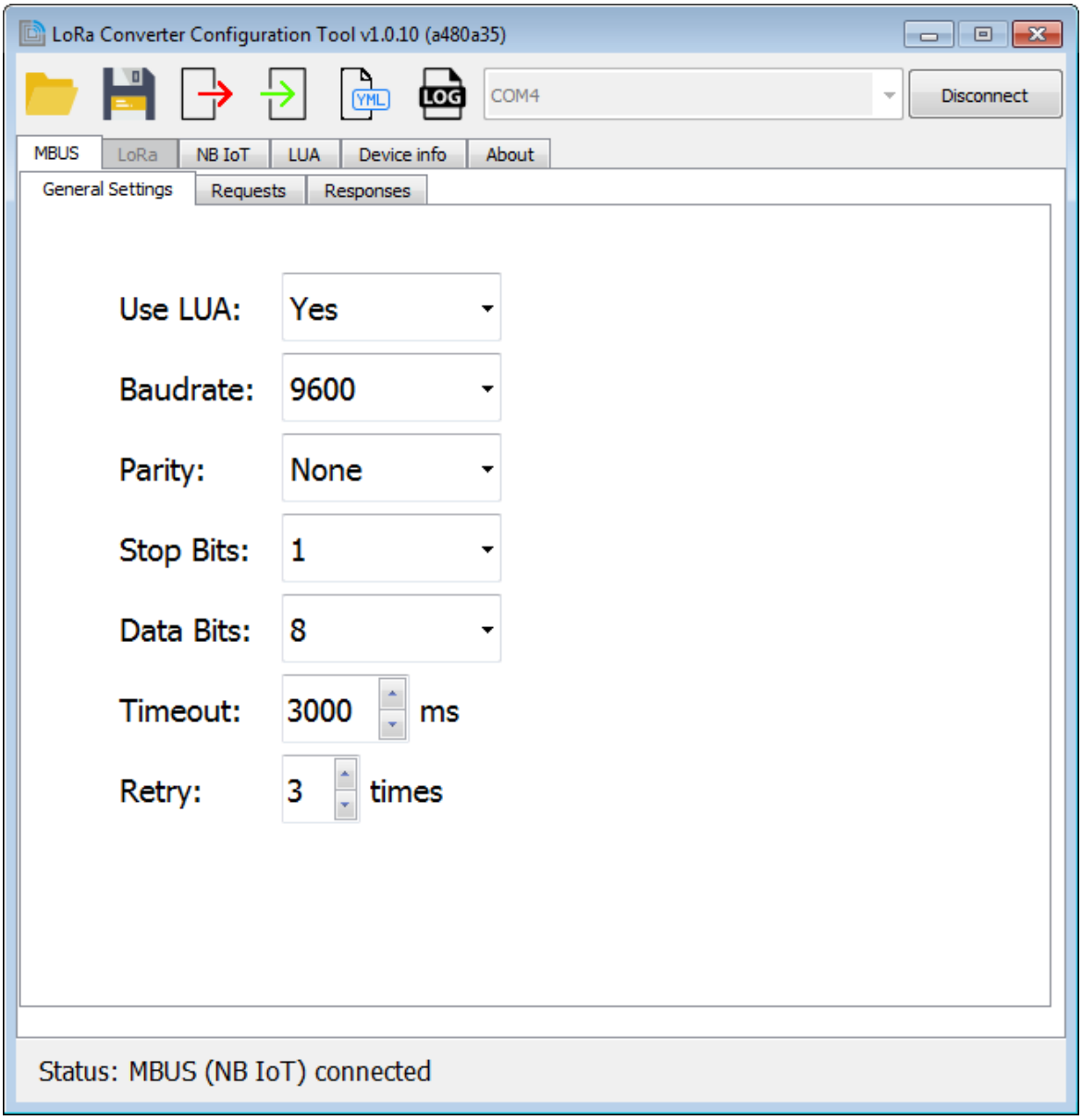
Connecting to a converter

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,
ctx = api.getGUIContext()\n\n    if err ~= 0 then\n        print
(\"Error occured on line\" .. tostring(err))\n        print(\"Sending
error code to NB-IoT\")\n        api.nbSend(ip,port, \"ERROR: \" ..
tostring(err), 5000, proto)\n        print(\"Done sending\")\n    else
\n        print(\"Sending to NB-IoT\")\n        api.nbSend(ip,port,
buf, 5000, proto)\n        print(\"Done sending\")\n        print(\"No
error, sent to NB-IoT\")\n    end\n\n    api.wakeUpIn(0,0,wake,0)\nend
\nfunction onStartup()\n    print(\"Starting up NB-IoT default script!
\")\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.

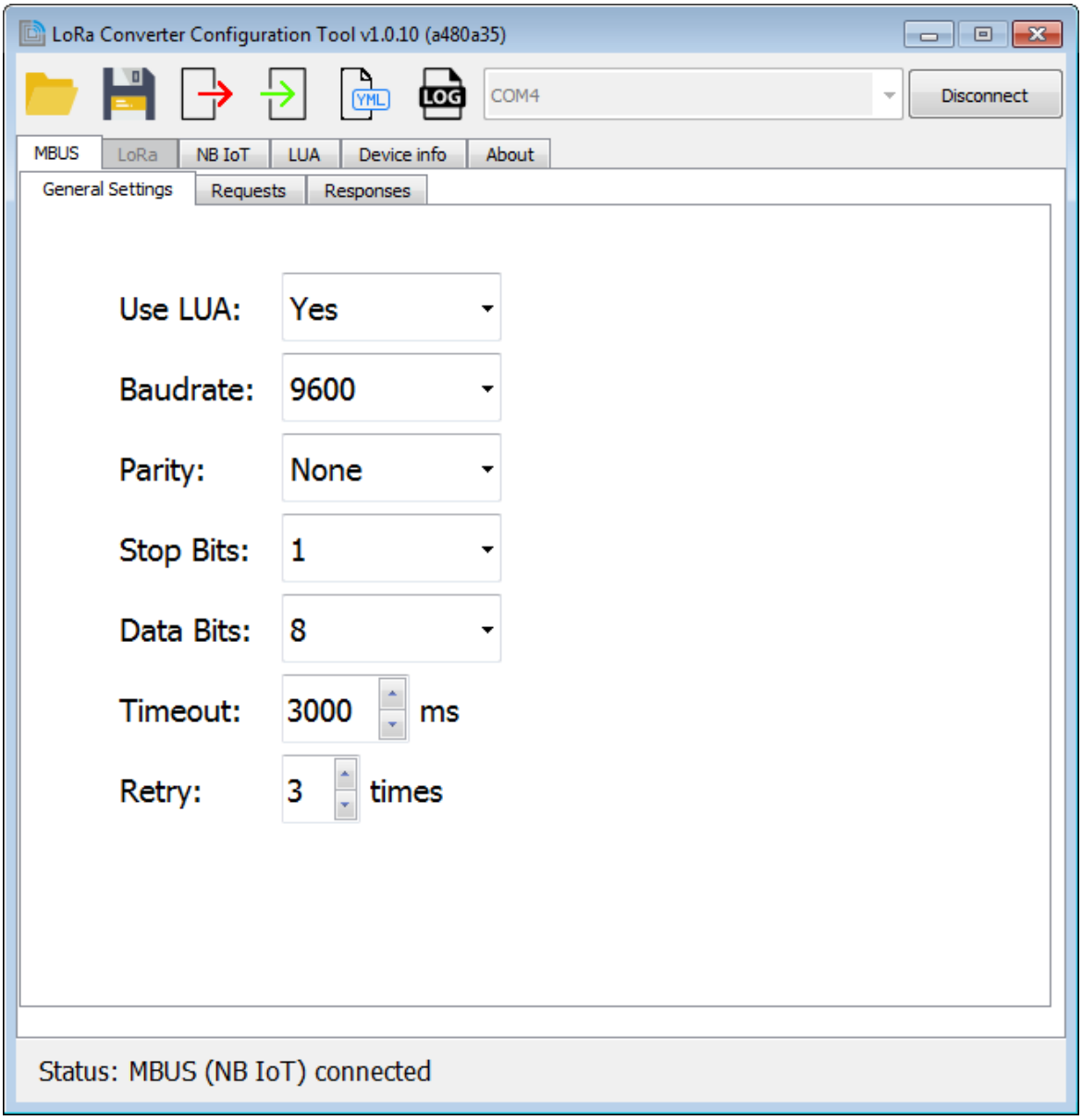
```
# ################
# General settings
# Baudrate: 9600 (80250000, Byte 7:10)
# Parity: None (00, Byte 11)
# Stop bits: 1 (01, Byte 12)
# Timeout: 3000 (B80B, Byte 13:14)
# Retry: 3 (03, Byte 15)
# Data bits: 8 (08, Byte 16)
# ################
settings:
434F4E46494701802500000001B80B030875FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Tabs description

MBUS setting

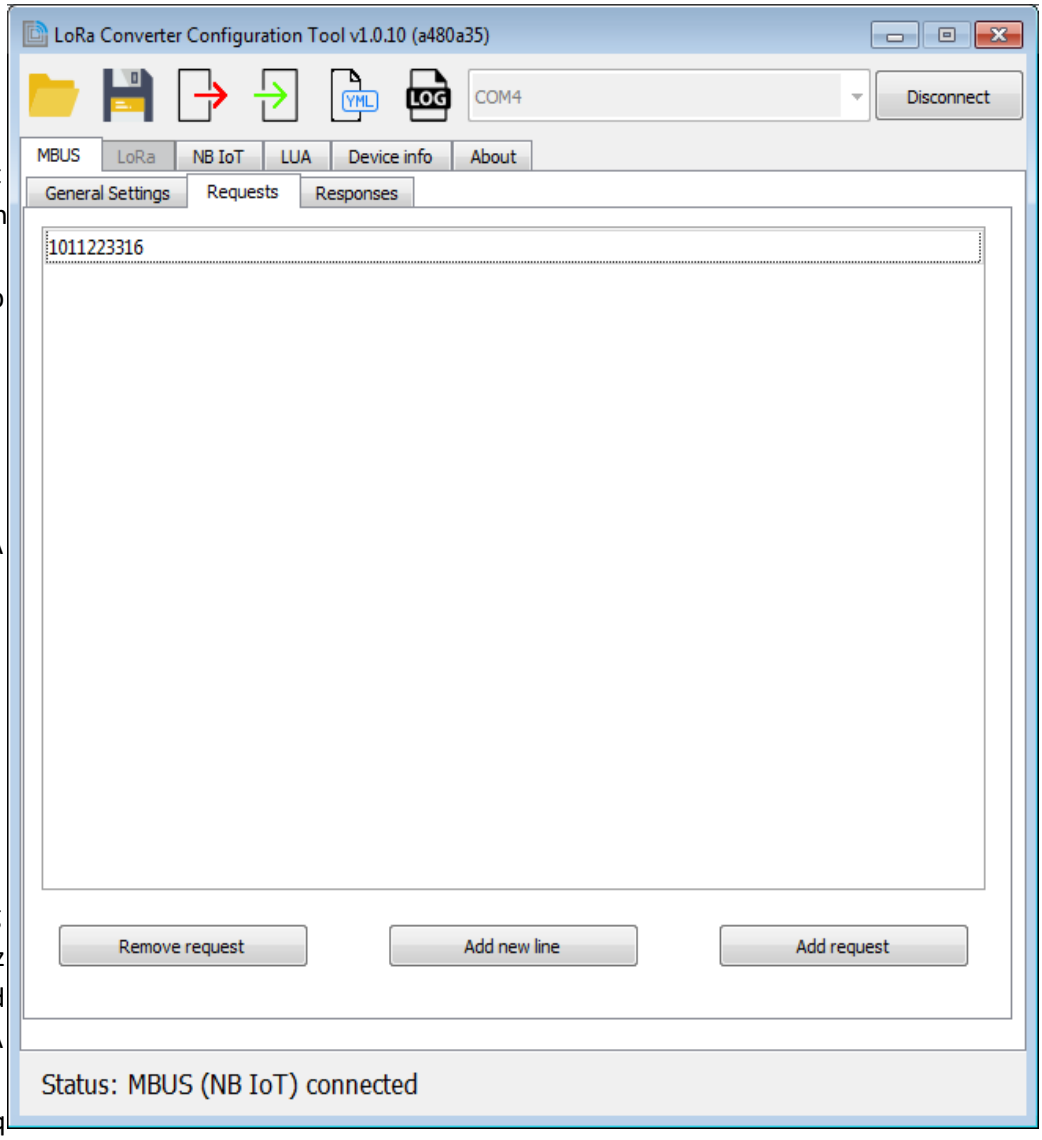
General setting

Settings of communication protocol with MBUs device and converter:



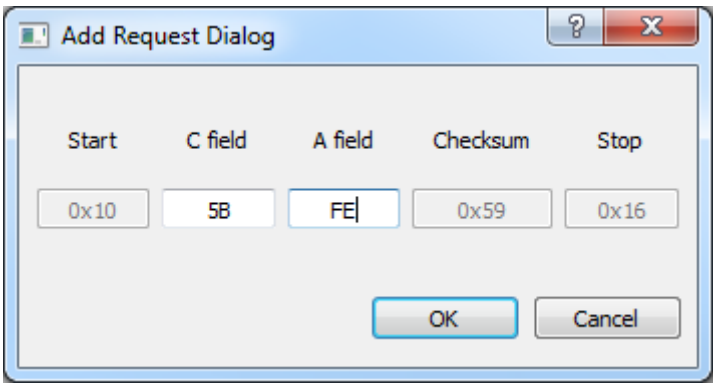
List of MBUS requests

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



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

MBUS

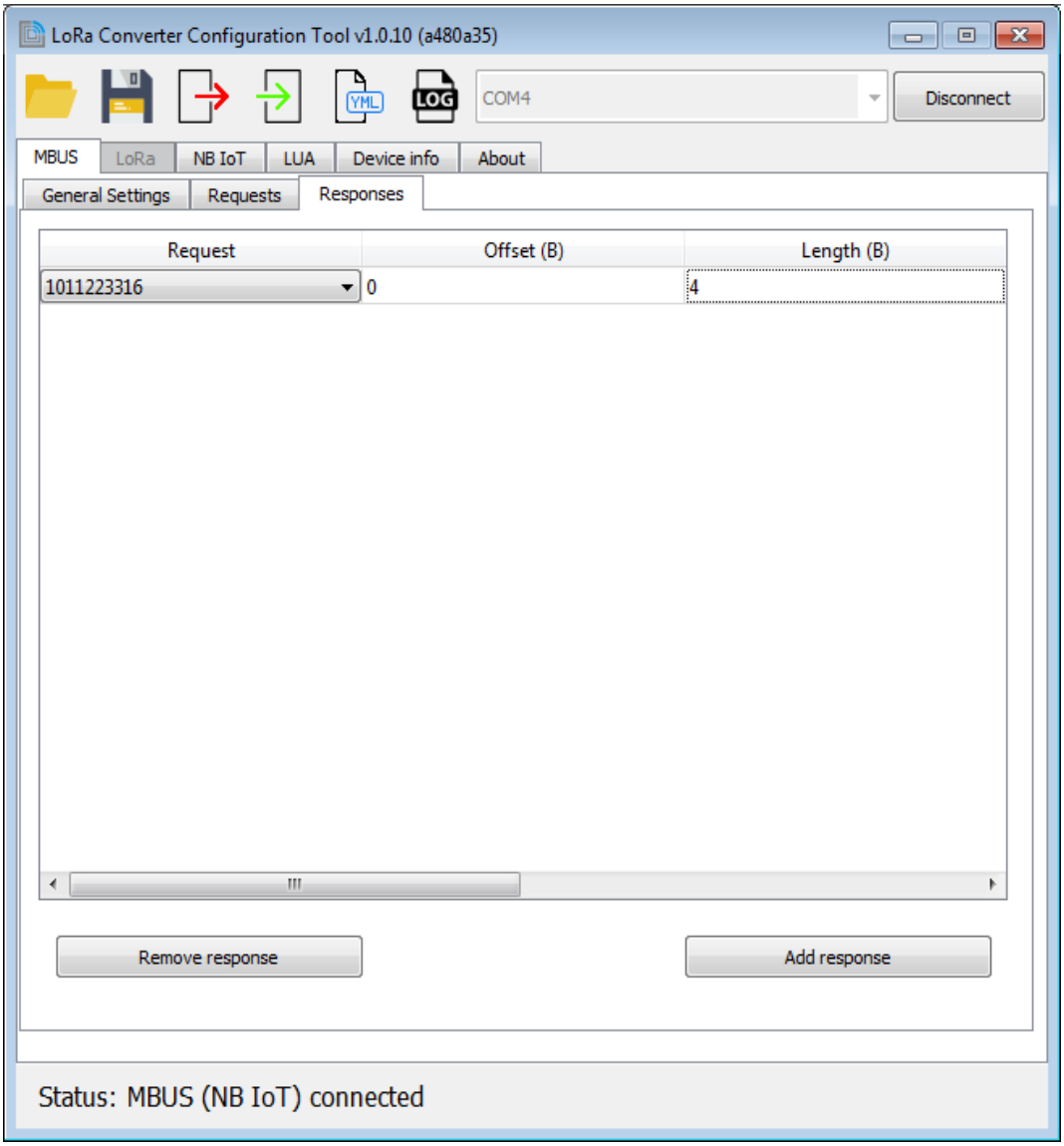


Dialog automatically calculates checksum and fills in start and stop bytes.

List of responses

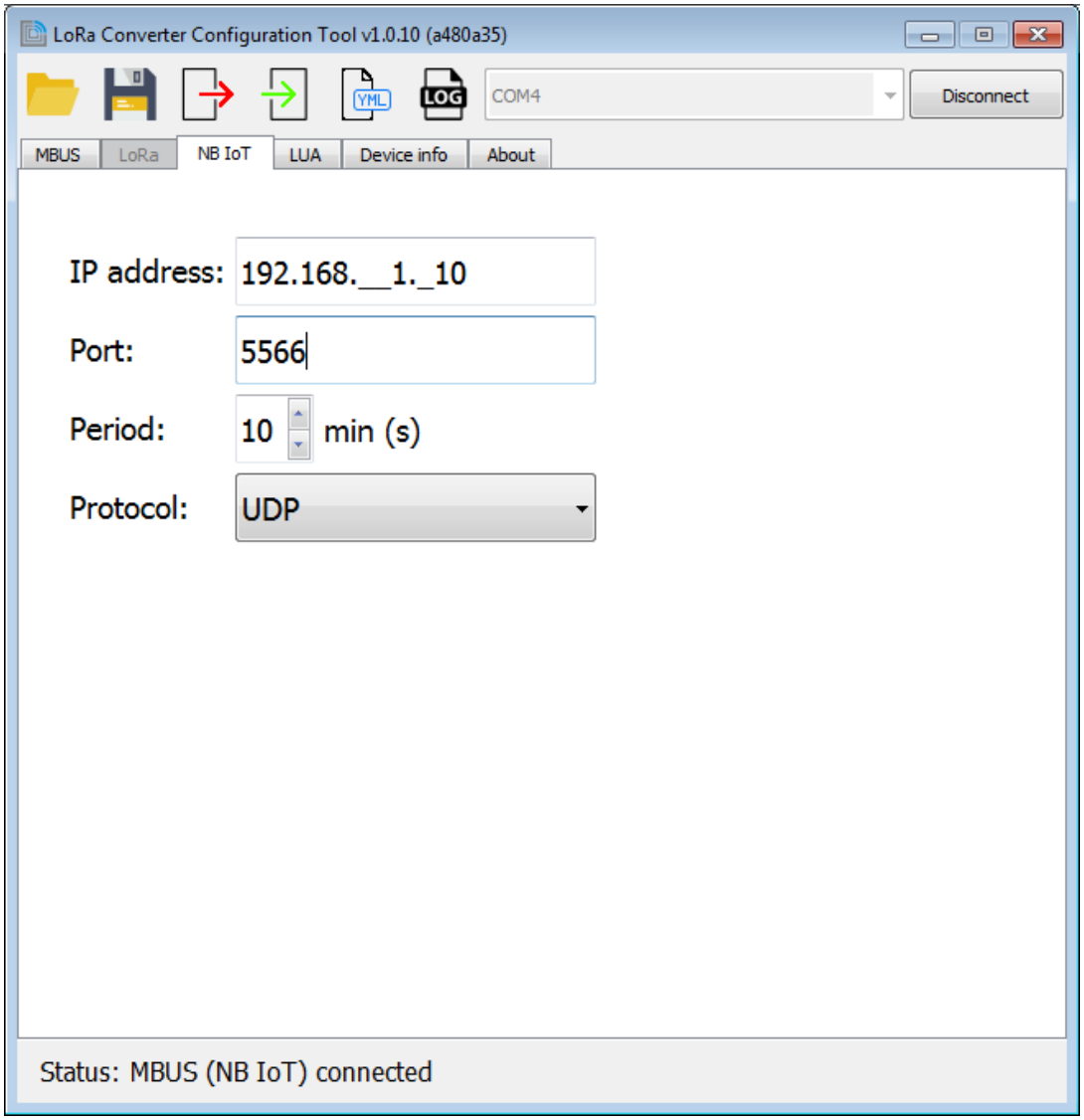
User can set different offsets and length which are taken from responses to selected requests.

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



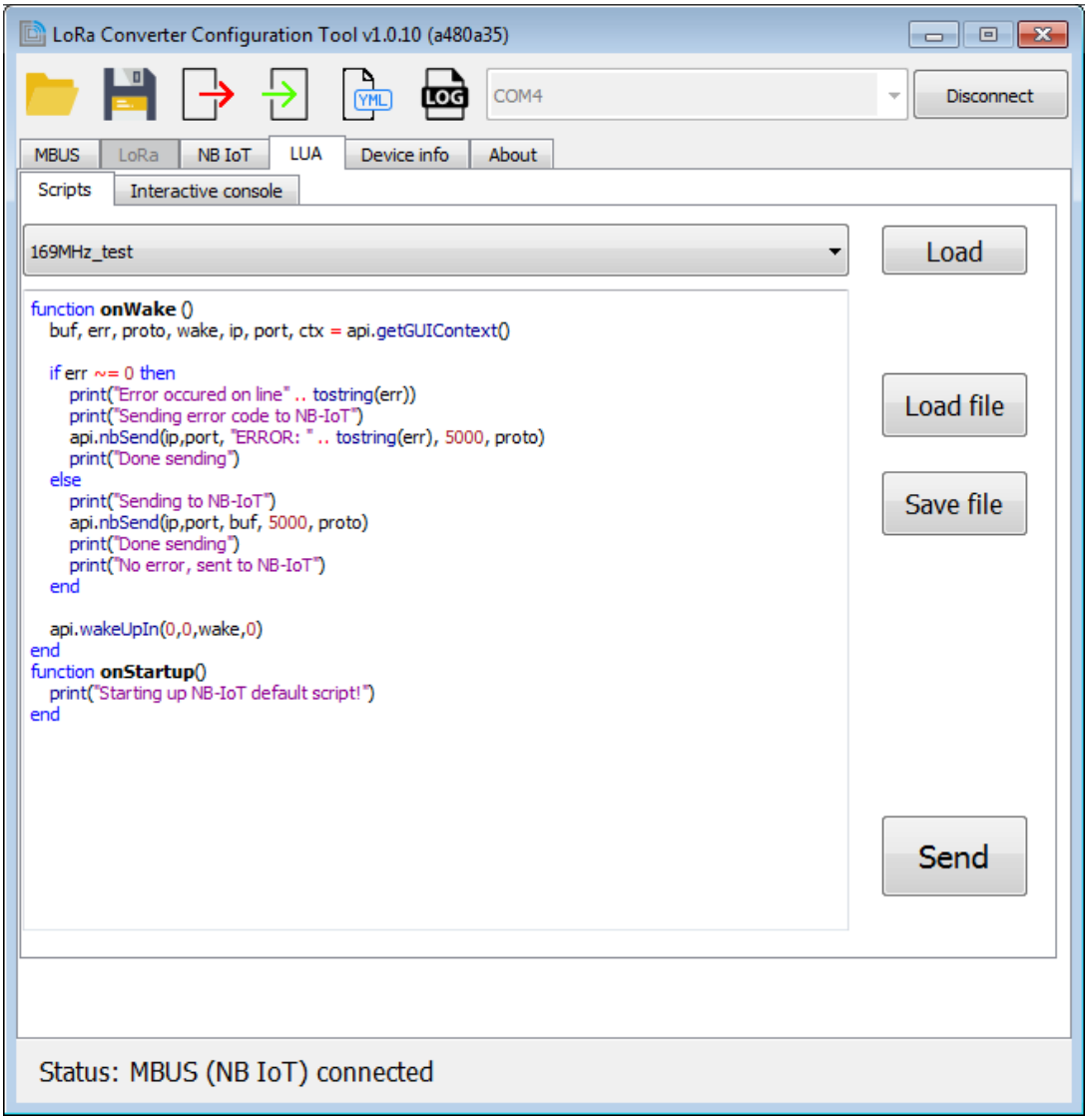
NB-IoT settings

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



LUA interface

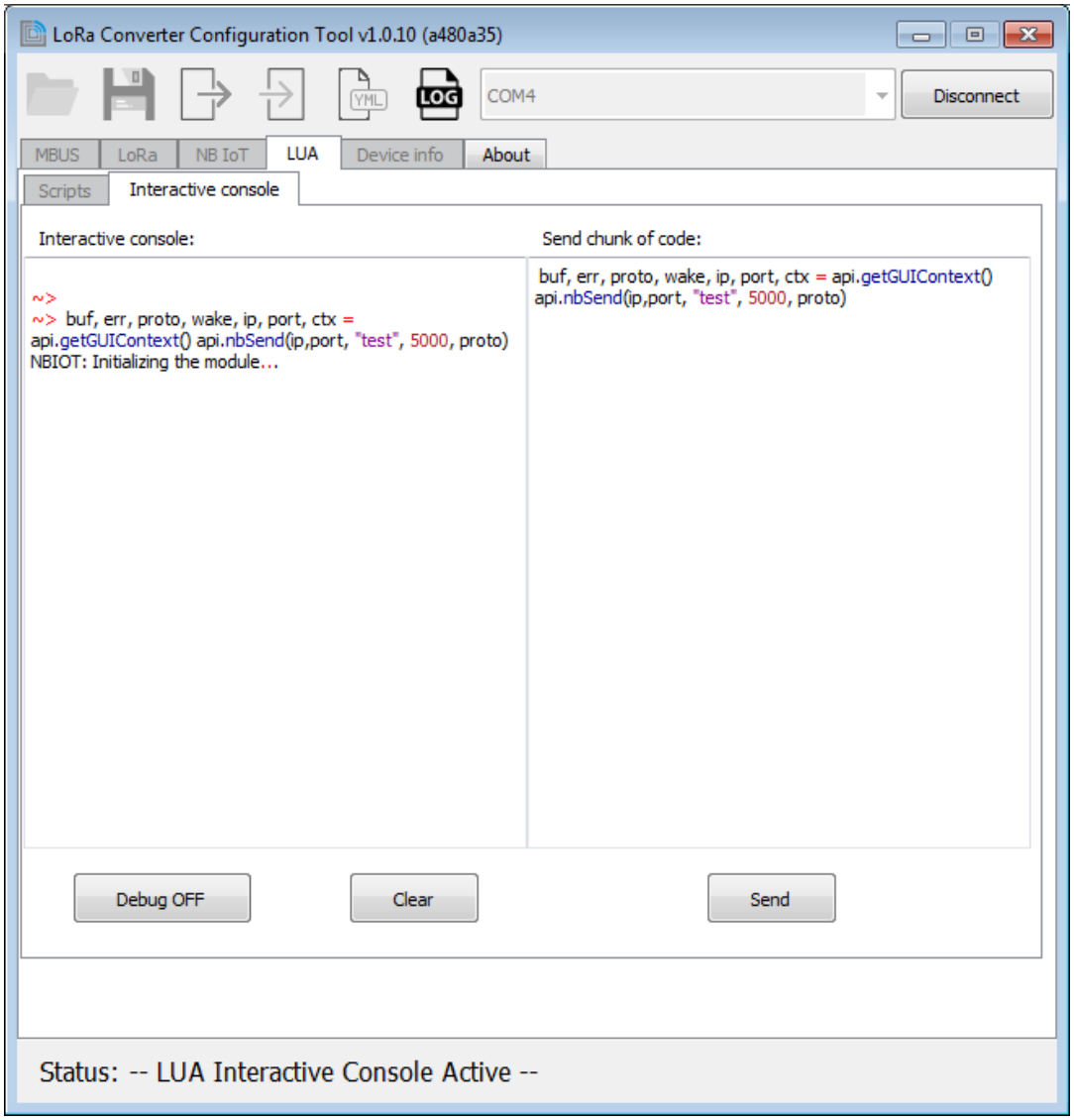
LUA interface adds ability to program converter with user's script with use of converter's API (separate documentation).



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.

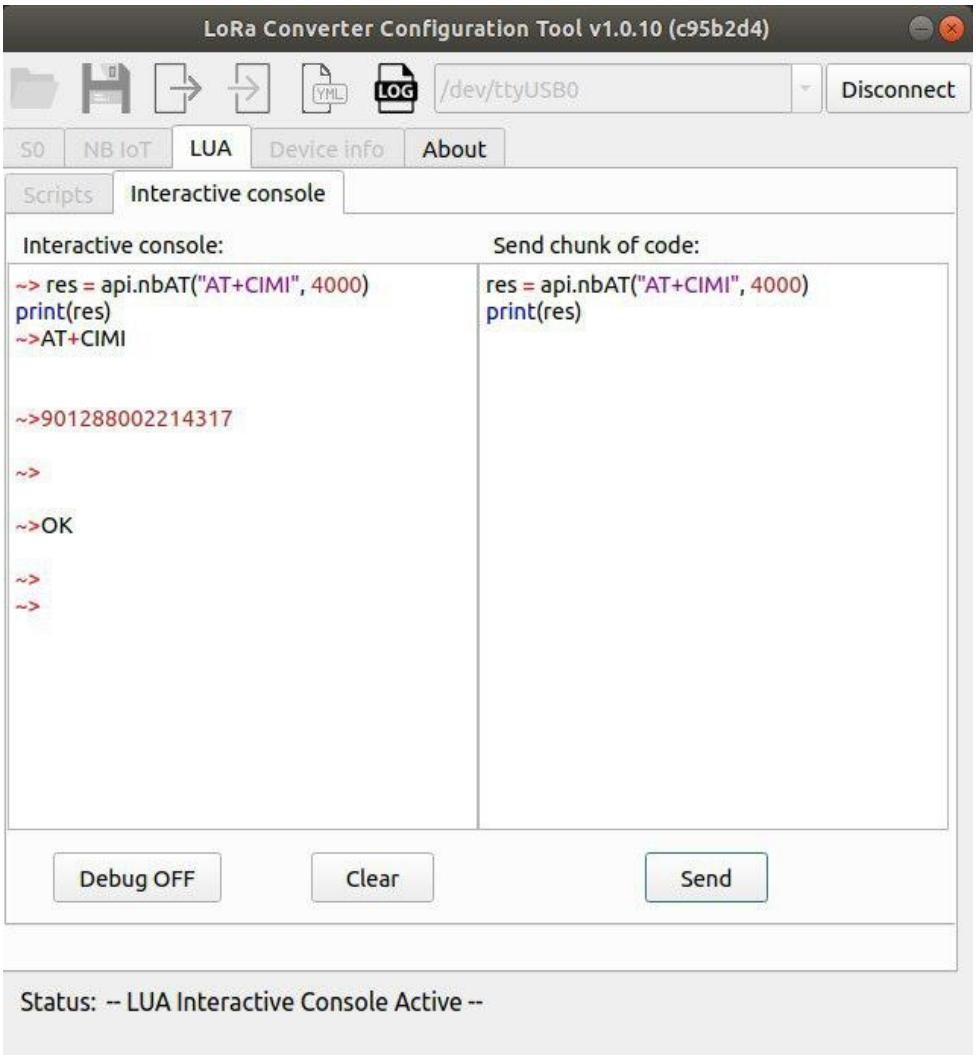
Example usage of Lua interface

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

api.nbAT()

IMSI SIM card number:

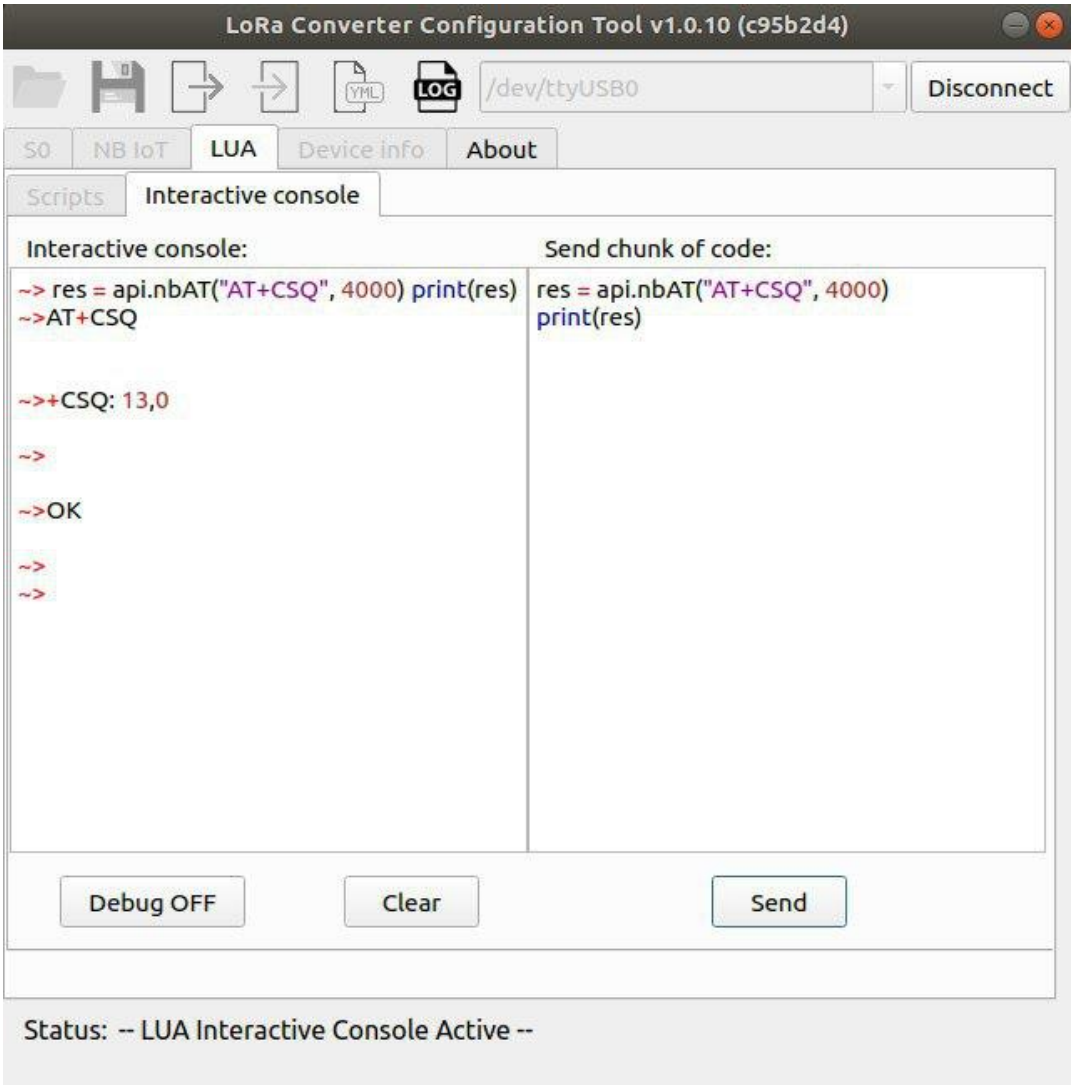
Command: *AT+CIMI*



Quality of signal:

Command: AT+CSQ

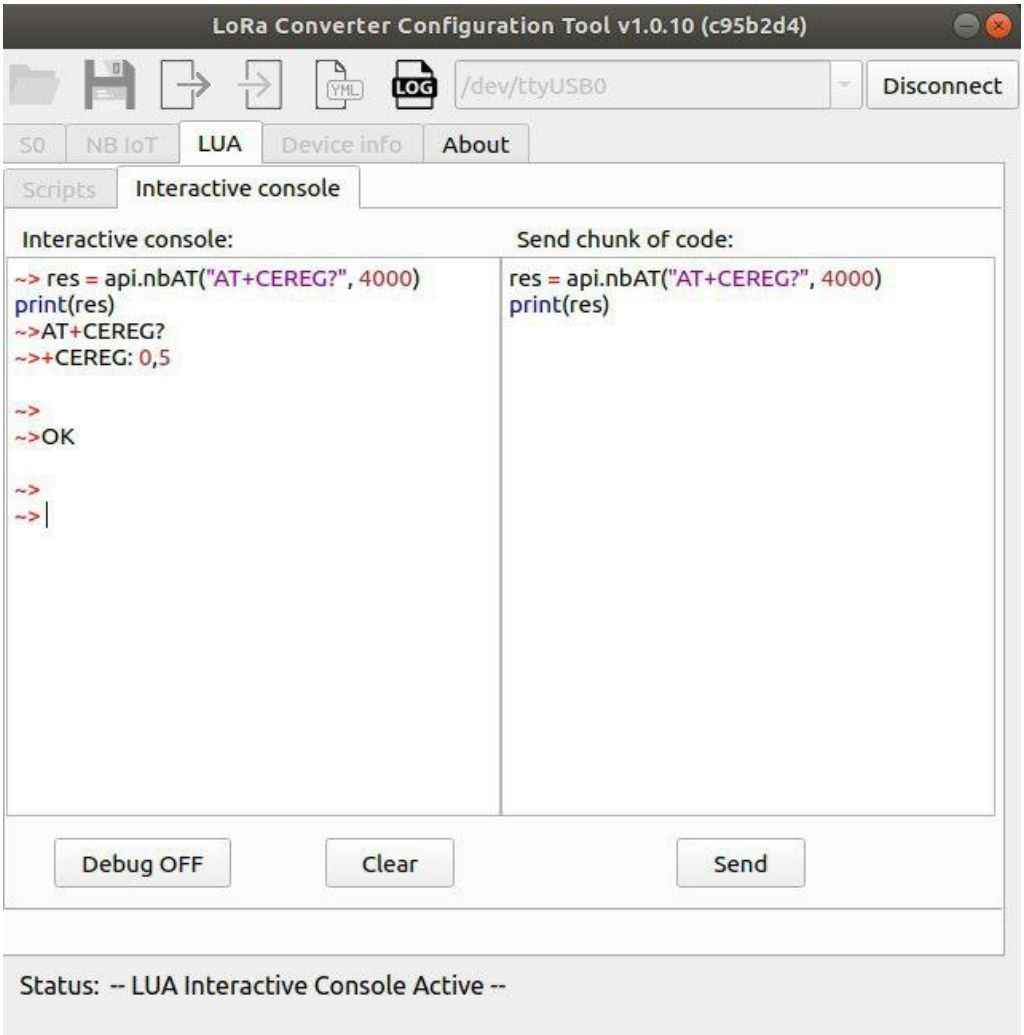
Response: 13 means 85 dBm RSSI (normal values 9-14, bigger number = stronger signal)



Status of network registration

Command: AT+CEREG?

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



Current operator:

Command: AT+COPS?

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