Documentation

EOL Test Adapter V1.40

TEST ADAPTER END OF LINE TEST

Module

Buzzer
CP and ADC (proximity and car state)
Contactor (5.10.0+)
Emergency Opener
FileIO
GPIO (Atmel and i2c)
GPIO (pic)
GSM functions
Heater
LEDs
Modbus Meter
Network
Phase Monitor (CC613, ICC1324)
PLC (Power Line Communication)
RCMB communication
RFID
SysInfo
Temperature
UART sending and receiving
USB Devices
Versions
Whitelist (RFID Cache)
WLAN
I2C
Settings

Buzzer

EOL Test Module Buzzer

The Buzzer Module is used to test the buzzer function.

start output

The buzzer will emit 3 frequencies from low to high, lasting approximately 1 second each for a total of 3 seconds.

- REQUEST

```
/diag/buzzer?cmd=test
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>Buzzer Started</value>
</response>
```

CP and ADC (proximity and car state)

EOL Test Modul Control Pilot

set pulse width of cp (control pilot) and relay. read adc channels in order determine proximity and car state

set pulse width

- REQUEST

```
/diag/cp?cmd=set_pw&target=[cp|relay]&value=[0..100]
```

Note: setting the pulse width to 100 will set the relay on. One should then reduce to 30 within 200ms. Otherwise some temp. protection mechanism will switch the relay off to avoid overheating of the coil. According to the relay datasheet, 30 will just keep the contacts closed

Note 2: ICC1324: Relay is controlled by Contactor / PowerCtrl.

- RESPONSE

```
<response>
  <result>success</result>
</response>
```

get adc channels

- REQUEST

  ```
  /diag/cp?cmd=get_adc_values
  ```

- RESPONSE

  ```
  <response>
  <result>success</result>
  <values>
    <value channel="0">139</value>
    <value channel="1">162</value>
    <value channel="2">0</value>
    <value channel="3">0</value>
  </values>
  </response>
  ```

  Beim ICC1324 / SAM9X60 mit ADC-IIO-Treiber arbeitet der ADC mit 6 Channels in 14 Bit. Der Wertebereich ist: 0 .. 16383

Note ICC1324: ADC-IIO-Driver needs PWM for trigger. PWM must be set by set_pw (cp) to a suitable value (e.g. 90 for CP+).

- Response (ICC1324)

  ```
  <response>
  <result>success</result>
  <values>
  <value channel="0">14208</value>
  <value channel="1">12416</value>
  <value channel="2">1124</value>
  <value channel="3">1416</value>
  <value channel="4">0</value>
  <value channel="5">1456</value>
  </values>
  </response>
  ```

- Channel Mapping ICC1324

  ```
  Channel 0: CP+
  Channel 1: PP
  Channel 2: CP-
  Channel 3: Temp1 (analog temperature sensor  1, power module)
  Channel 4: Temp2 (analog temperature sensor  2, power module)
  Channel 5: PT1000 (external analog temperature sensor)
  ```

Contactor (5.10.0+)

Manipulate the /dev/ebee_powerctrl device on the target. This allows you to open and close the contactor on systems 5.10 and above.

Open or close the contactor

- REQUEST

```
/diag/powerctrl?cmd=[open|close]
```

- RESPONSE

```
<response>
  <result>success</result>
</response>
```

Emergency Opener

Test functions for the Emergency Opener

get app or firmware version

- REQUEST

```
/diag/eo?cmd=get&what=[app_version|fw_version]
```

- RESPONSE

```
<response>
  <result>success</result>
  <value>D0653V1.0.1</value>
</response>
```

get actuator type

- REQUEST

```
/diag/eo?cmd=get&what=actuator_type
```

- RESPONSE

0 = Mennekes
1 = Fixed Cable

```
<response>
  <result>success</result>
  <value>0</value>
</response>
```

get actuator state

- REQUEST

```
/diag/eo?cmd=get&what=actuator_state
```

- RESPONSE

EO_ACTUATOR_STATE_UNKNOWN
EO_ACTUATOR_STATE_LOCKED
EO_ACTUATOR_STATE_UNLOCKED
EO_ACTUATOR_STATE_LOCKING
EO_ACTUATOR_STATE_UNLOCKING

```
<response>
  <result>success</result>
  <value>EO_ACTUATOR_STATE_UNLOCKING</value>
</response>
```

get voltage threshold

- REQUEST

```
/diag/eo?cmd=get&what=volt_threshold
```

- RESPONSE

```
<response>
  <result>success</result>
  <values>
    <value phase="1">10997</value>
    <value phase="2">11497</value>
    <value phase="3">9997</value>
  </values>
</response>
```

is cable locked

- REQUEST

  ```
  /diag/eo?cmd=get&what=cable_locked
  ```

- RESPONSE

yes/no

```
<response>
  <result>success</result>
  <value>yes</value>
</response>
```

get unlock time

- REQUEST

  ```
  /diag/eo?cmd=get&what=unlock_time
  ```

- RESPONSE

```
<response>
  <result>success</result>
  <value>2000</value>
</response>
```

get emergency opener state

- REQUEST

  ```
  /diag/eo?cmd=get&what=eo_state
  ```

- RESPONSE

EO_ERROR_STATE_POWER_SUPPY_UNDERVOLTAGE
EO_ERROR_STATE_CAPACITOR_UNDERVOLTAGE
EO_ERROR_STATE_STEP_UP_UNDERVOLTAGE
EO_ERROR_H_BRIDGE_DEFECTIVE
EO_ERROR_STACK_SHORT_OF_SPACE
Everything is ok

```
<response>
  <result>success</result>
  <value>Everything is ok</value>
</response>
```

get time threshold

- REQUEST

  ```
  /diag/eo?cmd=get&what=time_threshold
  ```

- RESPONSE

```
<response>
  <result>success</result>
  <values>
    <value phase="1">1000</value>
    <value phase="2">100</value>
    <value phase="3">10</value>
  </values>
</response>
```

get voltage levels

- REQUEST

  ```
  /diag/eo?cmd=get&what=voltage_level
  ```

- RESPONSE

```
<response>
  <result>success</result>
  <value>12220</value>
</response>
```

set voltage threshold

- REQUEST

  ```
  /diag/eo?cmd=set&what=volt_threshold&type=T&volts=V
  ```

T = 1, 2, or 3 V = Voltage

- RESPONSE

```
<response>
  <result>success</result>
</response>
```

set actuator type 0 = Mennekes 1 = ?

- REQUEST

  ```
  /diag/eo?cmd=set&what=actuator_type&val=[0|1]
  ```

- RESPONSE

```
<response>
  <result>success</result>
</response>
```

set time threshold

- REQUEST

  ```
  /diag/eo?cmd=set&what=time_threshold&type=T&time=M
  ```

T = 1, 2, or 3 M = milliseconds

- RESPONSE

```
<response>
  <result>success</result>
</response>
```

open or close

- REQUEST

```
/diag/eo?cmd=[open|close]
```

- RESPONSE

```
<response>
  <result>success</result>
</response>
```

go: Sends command requesting the EO to switch to program mode

- REQUEST

```
/diag/eo?cmd=go
```

- RESPONSE

```
<response>
  <result>success</result>
</response>
```

reset: restarts the EO

- REQUEST

```
/diag/eo?cmd=reset
```

- RESPONSE

```
<response>
  <result>success</result>
</response>
```

raw: RAW message request (generic transfer of messages supported by the EO)

- REQUEST

```
/diag/eo?cmd=raw&request=[R]&response_size=[S]&response_dely=[D]
```

[R] Message ID (Hex)
[S] expected response size (optional)
[D] response delay in µs (optional)

Example: Read Firmware Version RAW

- REQUEST

```
/diag/eo?cmd=raw&request=3&response_size=8
```

- RESPONSE

```
<response>
<result>success</result>
<value>000100020001028E</value>
</response>
```

testing: low level test functions

Note: EO (app) firmware V1.1.4 or higher is required for testing functions

- REQUEST

```
/diag/eo?cmd=testing&what=[W]&function=[F]&port=[Po]&pin=[Pi]&mode=[M]&type=[T]&speed=[S]&pupd=[Pu]&altfunc=[A]
&value=[V]
```

[W] what to do (active / inactive / pin)

- active: set test mode active (required to enable pin test functions in EO firmware)
- inactive: set test mode inactive (switch to normal operation of EO firmware)
- pin: pin test

[F] pin test function (init / set / get)

- init: pin init
- set: set pin output
- get: get pin input

Parameters for function "init"

- port: 0..7
- pin: 0..15
- mode: 0..3 (0: GPIO input, 1: GPIO output, 2: alternative function, 3: analog)
- type (optional): 0..1 (0: push-pull (default), 1: open drain)
- speed (optional): 0..3 (0: low, 1: medium (default), 2: high, 3: very high)
- pupd (optional): 0..3 (0: no pull (default), 1: weak pull-up, 2: weak pull-down, 3: reserved)
- altfunc (optional): 0..7

Parameters for function "set"

- port: 0..7
- pin: 0..15
- value: 0..1 (0: low (GND), 1: high (3,3 V))

Parameters for function "get"

- port: 0..7
- pin: 0..15
- type: 0..1 (0: GPIO, 1: ADC)

Example: Get input value of GPIO PC14 (ICC1324 weld check)
Note: set test mode active and pin init must be done before get input value

- REQUEST

```
/diag/eo?cmd=testing&what=pin&function=get&port=2&pin=14&type=0
```

- RESPONSE

```
<response>
<result>success</result>
<value>1</value>
</response>
```

FileIO

EOL Test Module FileIO

The FileIO Module is used to test reading and writing of files.

write to file

- REQUEST

  ```
  /diag/fileio?cmd=set&fname=[F]&value=[V]
  ```

[F] Path **to file**. Sub directories must exist, they will not **be** created. File will **be** created or overwritten.
[V] Characters **to write to** the **file**

- RESPONSE

Contains the written value in the response.

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>[V]</value>
</response>
```

read from file

- REQUEST

  ```
  /diag/fileio?cmd=get&fname=[F]
  ```

[F] Path **to** file **from** which **to** read

- RESPONSE

Contains the value that was read in the response.

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>[V]</value>
</response>
```

[V] Value **that** was read **from the** file.

delete file

- REQUEST

  ```
  /diag/fileio?cmd=del&fname=[F]
  ```

[F] **Path to** file **to delete**.

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

Example using the write/read/delete functions:

```
/diag/fileio?cmd=set&fname=/root/persistency/test.txt&value=test1
/diag/fileio?cmd=get&fname=/root/persistency/test.txt
/diag/fileio?cmd=del&fname=/root/persistency/test.txt
```

GPIO (Atmel and i2c)

EOL Test Modul GPIO

The gpio Modul is used to set direction or set/get logic value of any gpio pin on the hardware components. You have the urls to set direction, write level or read level of a port pin.

set direction

- REQUEST

```
/diag/gpio?cmd=set_direction&pin=[P]&direction=[D]
```

```
[P]  0 - xx    PortPin-ID
[D]  out|in    PortPin direction
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

get logic value

- REQUEST

```
/diag/gpio?cmd=get_pin&pin=[P]
```

```
[P]  0 - xx    PortPin-ID
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>[L]</value>
</response>
```

```
[L]  on|off    PortPin level (off=low; on=high)
```

set logic value

- REQUEST

```
/diag/gpio?cmd=set_pin&pin=[P]&value[L]
```

```
[P]  0 - xx    PortPin-ID
[L]  on|off    PortPin level (off=low; on=high)
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

unload module (ICC1324: unload kernel module to release GPIOs occupied by kernel module)

- REQUEST

```
/diag/gpio?cmd=unload_module&module=[M]
```

[M] kernel **module name**

Example ICC1324: unload phasemon module to release GPIOs PB22, PB23, PB24, PC12

- REQUEST

```
/diag/gpio?cmd=unload_module&module=ebee_phasemon
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

load module (ICC1324: load kernel module)

- REQUEST

```
/diag/gpio?cmd=load_module&module=[M]
```

[M] kernel **module name**

Example ICC1324: load phasemon module

- REQUEST

```
/diag/gpio?cmd=load_module&module=ebee_phasemon
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

PortPin ID Table

- Port pins AT91SAM9260 (CC613): PA0..PA31, PB0..PB31, PC0..C31
- Port pins SAM9X60 (ICC1324): PA0..PA31, PB0..PB25, PC0..C31, PD0..PD21

ID	µC	Pin		ID	µC	Pin		ID	µC	Pin		ID	µC	Pin
0	AT	PA0		32	AT	PB0		64	AT	PC0		96	AT	PD0
1	AT	PA1		33	AT	PB1		65	AT	PC1		97	AT	PD1
2	AT	PA2		34	AT	PB2		66	AT	PC2		98	AT	PD2
3	AT	PA3		35	AT	PB3		67	AT	PC3		99	AT	PD3
4	AT	PA4		36	AT	PB4		68	AT	PC4		100	AT	PD4
5	AT	PA5		37	AT	PB5		69	AT	PC5		101	AT	PD5
6	AT	PA6		38	AT	PB6		70	AT	PC6		102	AT	PD6
7	AT	PA7		39	AT	PB7		71	AT	PC7		103	AT	PD7
8	AT	PA8		40	AT	PB8		72	AT	PC8		104	AT	PD8
9	AT	PA9		41	AT	PB9		73	AT	PC9		105	AT	PD9
10	AT	PA10		42	AT	PB10		74	AT	PC10		106	AT	PD10

ID	µC	Pin		ID	µC	Pin		ID	µC	Pin		ID	µC	Pin
11	AT	PA11		43	AT	PB11		75	AT	PC11		107	AT	PD11
12	AT	PA12		44	AT	PB12		76	AT	PC12		108	AT	PD12
13	AT	PA13		45	AT	PB13		77	AT	PC13		109	AT	PD13
14	AT	PA14		46	AT	PB14		78	AT	PC14		110	AT	PD14
15	AT	PA15		47	AT	PB15		79	AT	PC15		111	AT	PD15
16	AT	PA16		48	AT	PB16		80	AT	PC16		112	AT	PD16
17	AT	PA17		49	AT	PB17		81	AT	PC17		113	AT	PD17
18	AT	PA18		50	AT	PB18		82	AT	PC18		114	AT	PD18
19	AT	PA19		51	AT	PB19		83	AT	PC19		115	AT	PD19
20	AT	PA20		52	AT	PB20		84	AT	PC20		116	AT	PD20
21	AT	PA21		53	AT	PB21		85	AT	PC21		117	AT	PD21
22	AT	PA22		54	AT	PB22		86	AT	PC22				
23	AT	PA23		55	AT	PB23		87	AT	PC23				
24	AT	PA24		56	AT	PB24		88	AT	PC24				
25	AT	PA25		57	AT	PB25		89	AT	PC25				
26	AT	PA26		58	AT	PB26		90	AT	PC26				
27	AT	PA27		59	AT	PB27		91	AT	PC27				
28	AT	PA28		60	AT	PB28		92	AT	PC28				
29	AT	PA29		61	AT	PB29		93	AT	PC29				
30	AT	PA30		62	AT	PB30		94	AT	PC30				
31	AT	PA31		63	AT	PB31		95	AT	PC31				

GPIO (pic)

EOL Test Modul GPIO

The gpio Modul is used to set direction or set/get logic value of any gpio pin on the pic. You have the urls to set direction, write level or read level of a port pin.

set direction

- REQUEST

```
/diag/picgpio?cmd=set_direction&pin[P]&direction=[D]
```

```
[P]  0 - xx    PortPin-ID
[D]  out|in    PortPin direction
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

get logic value

- REQUEST

```
/diag/picgpio?cmd=get_pin&pin=[P]
```

```
[P]  0 - xx    PortPin-ID
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>[L]</value>
</response>
```

```
[L]  on|off    PortPin level (off=low; on=high)
```

set logic value

- REQUEST

```
/diag/picgpio?cmd=set_pin&pin=[P]&value[L]
```

```
[P]  0 - xx    PortPin-ID
[L]  on|off    PortPin level (off=low; on=high)
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

PortPin ID Table - Version < 1.13

Pin [P]	Name	

Pin [P]	Name	
0	PIC24_Pin1	BCD1 - RE5, input only
1	PIC24_Pin2	BCD2 - RE6, input only
2	PIC24_Pin3	BCD3 - RE7, input only
3	PIC24_Pin5	BCD4 - RG7, input only
4	PIC24_Pin6	LED2 green new - RG8
5	PIC24_Pin29	LED3 yellow new - RB14
6	PIC24_Pin31	Weld check - RF4. input only
7	PIC24_Pin35	Tilt sensor - RF6, input only
8	PIC24_Pin40	Version2 - RC15, input only
...		
14	PIC24_Pin50	Version1 - RD2, input only
15	PIC24_Pin53	Version0 - RD5, input only
17	PIC24_Pin55	Relay - RD7
...		
23	PIC24_Pin63	Controller5 - RE3, input only, ex in 1
24	PIC24_Pin64	Controller6 - RE4, input only, ex in 2

Since Version 1.13

Pin [P]	Name	16xx	15xx
0	PIC24_Pin1	RE5 BCD1, input	Rotation detection L1, input
1	PIC24_Pin2	RE6 BCD2, input	Rotation detection L2, input
2	PIC24_Pin3	RE7 BCD3, input	Rotation detection L3, input
3	PIC24_Pin5	RG7 BCD4, input	
4	PIC24_Pin6	RG8 LED2, green new	
5	PIC24_Pin29	RB14 LED3, yellow new	
6	PIC24_Pin31	RF4 Weld check, input	Weld check, input
7	PIC24_Pin35	RF6 Tilt sensor, input	
8	PIC24_Pin37		CP monitor, input
9	PIC24_Pin40	RC15, input only	
10	PIC24_Pin43	RD9 Relay7	Relay on/off
11	PIC24_Pin44	RD10 Relay6	
12	PIC24_Pin45	RD11 Relay5	
13	PIC24_Pin46	RD0 Relay4	
14	PIC24_Pin49	RD1	USB chip reset

Pin [P]	Name	16xx	15xx
15	PIC24_Pin50	RD2 Version1, input	WLAN module reset
16	PIC24_Pin53	RD5 Version0, input	USB over-current, input
17	PIC24_Pin54	RD6	WLAN power on/off
18	PIC24_Pin55	RD7 Relay8	
19	PIC24_Pin58	RF0	Version0, input
20	PIC24_Pin59	RF1 Controller1, input	Temperature mon. 1, input
21	PIC24_Pin60	RE0 Controller2, input	Temperature mon. 2, input
22	PIC24_Pin61	RE1 Controller3, input	Version1, input
23	PIC24_Pin62	RE2 Controller4, input	Diagnostics 1, input
24	PIC24_Pin63	RE3 Controller5, input	Diagnostics 2, input
25	PIC24_Pin64	RE4 Controller6, input	Version2, input

GSM functions

GSM Module communicates with the QUECTEL GSM modem.

Request IMEI

- REQUEST

```
/diag/gsm?cmd=getimei
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>385627154035637</value>
</response>
```

Request IMSI

- REQUEST

```
/diag/gsm?cmd=getimsi
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>405627154035637</value>
</response>
```

Check if SIM PIN is needed

- REQUEST

```
/diag/gsm?cmd=pinstatus
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>Not Required</value>
</response>
```

Connect to gsm network

- REQUEST

```
/diag/gsm?cmd=connect[&apn=[A]]
```

[A] optionally **set** the APN. Default **is** "internet.t-mobile"

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>started</value>
</response>
```

Disconnect from GSM network

- REQUEST

```
/diag/gsm?cmd=disconnect
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>Disconnected</value>
</response>
```

Receive information including connection status and signal strength

- REQUEST

```
/diag/gsm?cmd=status
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<values>
<value name="Connected">Connected</value>
<value name="IMEI">385627154035637</value>
<value name="IMSI">405627154035637</value>
<value name="Manufacturer">Quectel</value>
<value name="Model">EC21</value>
<value name="Revision">EC21EFAR02A03M4G</value>
<value name="Signal">72dBm</value>
</values>
</response>
```

Get modem and manufacturer information

- REQUEST

```
/diag/gsm?cmd=info
```

This information is also included in the status call

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<values>
<value key="Manufacturer">Quectel</value>
<value key="Model">EC21</value>
<value key="Revision">EC21EFAR02A03M4G</value>
</values>
</response>
```

Check modem availability

- REQUEST

```
/diag/gsm?cmd=check_modem_on
```

- RESPONSE (if modem is switched on and driver available)

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>Modem is On</value>
</response>
```

- RESPONSE (if modem is switched off / driver not available)

```
<?xml version="1.0"?>
<response>
<result>fail</result>
<cause>Modem is Off</cause>
</response>
```

Get information about RSSI and network (works also without SIM card)

- REQUEST

```
/diag/gsm?cmd=status2
```

- RESPONSE (example)

```
<response>
<result>success</result>
<values>
<value name="COPS">+COPS: 0,0,"o2 - de Things Mobile",7</value>
<value name="QNWINFO">+QNWINFO: "FDD LTE","26203","LTE BND 20",1600</value>
<value name="RSSI">18</value>
</values>
</response>
```

Execute AT command

- REQUEST

```
/diag/gsm?cmd=AT&atcmd=[C]
```

[C] **AT command** (without "AT")

- RESPONSE (example for [C]="+QNWINFO")

```
<response>
<result>success</result>
<values>
<value name="line 0">AT+QNWINFO</value>
<value name="line 1">+QNWINFO: "FDD LTE","26203","LTE BAND 20",1600</value>
<value name="line 2">OK</value>
</values>
</response>
```

Check EG91 modem firmware version and USB setting

Notes: The EG91 modem with default settings shows problems with broken USB communication (ICC1324). The controller software enables the setting (+QCFG: "usbenum/seoctl",1) to avoid these problems. EG91 FW Version must be >= 5 to support this setting. The response must be "success", otherwise the EG91 FW is not up to date or the USB setting is not correct.

- REQUEST

```
/diag/gsm?cmd=EG91check
```

- RESPONSE

```
<response>
<result>success</result>
<value>EG91 firmware >= V05 and USB settings OK</value>
</response>
```

Check if modem Firmware is EC21EFAR06A01M4G or higher and has the QRF Test AT commands

Note: If you send the QRFTEST AT commands to a modem that does not support them, the modem will not respond until reset.

- REQUEST

```
/diag/gsm?cmd=qrftest&mode=hasqrf
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>True</value>
</response>

<?xml version="1.0"?>
<response>
<result>fail</result>
<value>False</value>
</response>
```

The returned value may be "True" or "False"

Start QRF Test

Note: If you send the QRFTEST AT commands to an EC21 modem with a firmware that does not support them, the modem will not respond until reset. Therefore we check if the firmware version is up to date. You can optionally force the execution of the command if you think that your modems firmware supports this command.

- REQUEST

```
/diag/gsm?cmd=qrftest&mode=start&band=[B]&channel=[C]&txenable=[E]&txgain=[G]&waveform=[W][&force=[1|true]]
```

```
[B] see the the pdf for possible values*
[C]
[E]
[G]
[W]
optional: force=1 or force=true overrides and sends the command anyways, even if hasqrf test returns negative
```

*ATQRFTEST.pdf

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>fail</result>
<cause>Your Modem does not support the QRFTESTMODE commands</cause>
</response>
```

```
<?xml version="1.0"?>
<response>
<result>success</result>
<values>
```

Stop the QRF Test

Note: If you send the QRFTEST AT commands to an EC21-E modem with a firmware that does not support them, the modem will not respond until reset. Therefore we check if the firmware version is up to date. You can optionally force the execution of the command if you think that your modems firmware supports this command.

- REQUEST

```
/diag/gsm?cmd=qrftest&mode=stop[&force=[1|true]]
```

```
optionally force execution even when hasqrf test responds negative
if force is set to anything other than 1 or true, it is ignored as if it hadn't been sent
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>fail</result>
<cause>Your Modem does not support the QRFTESTMODE commands</cause>
</response>
```

```
<?xml version="1.0"?>
<response>
<result>success</result>
<values>
```

Heater

EOL Test Module Heater

This module allows testing the heater. The heater can be set using a percentage. The function passes the percentage to the PIC, which controls the heater.

set heater

- REQUEST

```
/diag/heater?percent=[p]
```

```
[p] value 0..100
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>100</value>
</response>
```

LEDs

EOL Test Module LEDs (ICC1522)

The LEDs Module is used to test the LEDs. You should inspect the LEDs after turning them on to see if they are working.

turn leds on

- REQUEST

```
/diag/leds?cmd=on
```

```
/diag/leds?cmd=on&r=[R]&g=[G]&b=[B]
```

```
You may optionally set the rgb values.
The default value of 255 is used if not set

[R] red    0..255
[G] green  0..255
[B] blue   0..255
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>All LEDS are ON[with manual RGB]</value>
</response>
```

turn leds off

- REQUEST

```
/diag/leds?cmd=off
```

which is the same as calling

```
/diag/leds?cmd=on&r=0&g=0&b=0
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>All LEDS are OFF</value>
</response>
```

progress bar

Only turn on a certain number of LED's. There are 8 LED's in the ICC1522. They can be turned on incrementally in pairs of two.

- REQUEST

```
/diag/leds?cmd=bar&step=[S]
```

```
[S] step from 0..4
S=1 turns on the first two LED's
S=2 turns on the first four LED's
S=3 turns on the first six LED's
S=4 turns on all of the LED's
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>OK</value>
</response>
```

EOL Test Module LEDs (ICC1324)

The ICC1324 LEDs Module is used control the LEDs on the ICC1324 main PCB or the LEDs on a connected RFID120 HMI PCB.

- REQUEST

  ```
  /diag/leds?cmd=on&device=[D]&number=[N]&pattern=[P]&r=[R]&g=[G]&b=[B]
  ```

[D] device (optional): ICC1324 / RFID120
[N] number of LED to control (optional, depending on device)

- ICC1324: 1..4
- RFID120: 1..12

  [P] test pattern (optional): 1 - LEDs with odd number on / 2 - LEDs with even number on
  [R][G][B] RGB values (optional): 0..255

Notes

- if no device [D] is set the software tries to detect ICC1234 (LP5012) or RFID120 (LP5036)
- if no [R] [G] [B] values are set the LEDs will be switched to white color
- if no number is set all LEDs are switched on
- pattern and number must not be set at the same time

EOL Test Module LEDs (CC613)

The CC613 LEDs Module is used control the LEDs of the USB Ethernet Phy.

- REQUEST

  ```
  /diag/leds?cmd=[C]&device=usb_eth&number=[N]
  ```

[C] command: on|off
[N] number of LED to control

- 1|2 (1: LINK LED, 2: SPEED LED)

Notes

- if no number is set both LEDs are switched on|off

Modbus Meter

EOL Test Module Modbus Meter

This module lets you read Modbus Meters registers

init

Init will always respond with success the first time and fail if called again.
To see if the communication works you must read a value.

- REQUEST

```
/diag/meter?cmd=init&dev=[D]&baud=[B]&parity=[P]
```

```
[D] device to read
[B] baud rate for communication
[P] parity of communication
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

```
<?xml version="1.0"?>
<response>
<result>fail</result>
<cause>Already initialized</cause>
</response>
```

uninit

- REQUEST

```
/diag/meter?cmd=uninit
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

```
<?xml version="1.0"?>
<response>
<result>fail</result>
<cause>not inited</cause>
</response>
```

read

- REQUEST

```
/diag/meter?cmd=read&id=[I]&address=[A]&length=[L]
```

```
[I] id of device (ex. 1 or 2)
[A] address of register in hexadecimal
[L] number of registers to read
```

- RESPONSE

Each register is 16 bit long and is represented in the result of the response by 4 hexadecimal values. The result is concatenated as it is read so any parsing and adjusting of the value must be done afterwards.

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>0000</value>
</response>
```

```
<?xml version="1.0"?>
<response>
<result>fail</result>
<cause>Failed to read</cause>
</response>
```

SAIA meter init parameters

```
[D] /dev/ttyS1
[B] 9600
[P] E
```

SAIA meter register Table

Name	Address [A]	Length [L]
IMPORT ENERGY	1b	2
TOTAL POWER	32	1
PHASE 1 CURRENT	24	1
PHASE 2 CURRENT	29	1
PHASE 3 CURRENT	2e	1
SERIAL NUMBER	f	2

Network

EOL Test Modul Network

The EOL Network module can be used for any network interface as it matches the URL to the interface. That means if you have a network interface

```
ppp99
```

you should be able to adjust its settings using the URL

```
/network/ppp99
```

get ifconfig status

- REQUEST

  ```
  /network/[I]?cmd=ifconfig
  ```

[I] **Interface**. Example: eth0 **or** wlan0

- RESPONSE

```
<?xml version="1.0"?>
<response>
 <result>success</result>
 <values>
  <value ifconfig="0">wlan0     Link encap:Ethernet  HWaddr D4:CA:6E:90:20:13</value>
  <value ifconfig="1">inet addr:192.168.10.154  Bcast:192.168.10.255  Mask:255.255.255.0</value>
  <value ifconfig="2">inet6 addr: fe80::d6ca:6eff:fe90:2013/64 Scope:Link</value>
  <value ifconfig="3">UP BROADCAST MULTICAST  MTU:1500  Metric:1</value>
  <value ifconfig="4">RX packets:2235 errors:0 dropped:4 overruns:0 frame:0</value>
  <value ifconfig="5">TX packets:25 errors:0 dropped:0 overruns:0 carrier:0</value>
  <value ifconfig="6">collisions:0 txqueuelen:1000</value>
  <value ifconfig="7">RX bytes:181803 (177.5 KiB)  TX bytes:3462 (3.3 KiB)</value>
  <value ifconfig="8"/>
 </values>
</response>
```

get MAC address of interface

- REQUEST

  ```
  /network/[I]?cmd=getifmac
  ```

[I] **Interface**. Example: wlan0

- RESPONSE

```
<?xml version="1.0"?>
<response>
 <result>success</result>
 <value>D4:CA:6E:A1:F3:9C</value>
</response>
```

set ip, net mask and gateway

- REQUEST

  ```
  /network/[I]?cmd=ifconfig&addr=[A]&subnet=[S]&gw=[G]
  ```

[I] **Interface**. **Example**: **eth0**
[A] **IP Address**. **Example**: **192**.168.5.1
[S] **Subnet Mask**. **Example**: **255**.255.255.0
[G] **Default Gateway**. **Example**: **192**.168.5.5

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

example /network/eth0?cmd=ifconfig&addr=192.168.5.1&subnet=255.255.255.0&gw=192.168.5.5

You may omit arguments such as addr, if it has already been set. Not all combinations work. The call will fail if the gateway is inaccessible.

interface up/down

Calls ifconfig up or ifdown -f on the interface

- REQUEST

```
/network/[I]?cmd=ifconfig&status=[S]
```

```
[I]  Interface to configure. Example: eth0
[S]  "up" or "down"
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

ping

- REQUEST

```
/network/[I]?cmd=ping&addr=[P]&count=[C]
```

```
[I]  Interface to use. Example: eth0
[P]  ipv4 address to ping
[C]  number of pings to send
```

```
Shell command to be executed: ping -I [I] -c [C] [P]
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<values>
<value line="1">PING 192.168.111.1 (192.168.111.1): 56 data bytes</value>
<value line="2">64 bytes from 192.168.111.1: seq=0 ttl=64 time=8.797 ms</value>
<value line="3"/>
<value line="4">--- 192.168.111.1 ping statistics ---</value>
<value line="5">1 packets transmitted, 1 packets received, 0% packet loss</value>
<value line="6">round-trip min/avg/max = 8.797/8.797/8.797 ms</value>
</values>
</response>
```

save MAC address to NOR storage

Write the MAC address for the given interface to persistent NOR storage.
Writes MAC address to NOR storage key "[I]_MAC"
It will overwrite the key if it exists.
Note: The NOR storage is only a storage and the key/value pair is only written. The system must have a mechanism in place to read the value and set the MAC address.

- REQUEST

```
/network/[I]?cmd=setmac&address=[A]
```

```
[I]  Interface name e.g. eth0
[A]  MAC address the eth0 adapter should use without the ":" delimiter
     ex: 123456789ABC writes 12:34:56:78:9A:BC
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

If the interface does not exist, the list of available interfaces is printed for reference.

get MAC address

Read the content of the [I]_MAC key from NOR and returns it.

- REQUEST

```
/network/[I]?cmd=getmac
```

```
[I]  Interface name e.g. eth0
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>12:34:55:78:9a:bc</value>
</response>
```

Phase Monitor (CC613, ICC1324)

Manipulate the /dev/ebee_phasemon device on the target.

read voltage frequency

- REQUEST

```
/diag/phasemon?cmd=read_frequency
```

- RESPONSE

```
<response>
  <result>success</result>
  <value>-1</value>
</response>
```

read state (read L1, L2, L3 frequency and rotation) (ICC1324)

- REQUEST

```
/diag/phasemon?cmd=read_state
```

- RESPONSE

```
<response>
<result>success</result>
<values>
<value key="L1 frequency">50.012</value>
<value key="L2 frequency">50.023</value>
<value key="L3 frequency">50.010</value>
<value key="rotation">0</value>
</values>
</response>
```

possible values for rotation:
0: normal
1: reverse
2: unknown

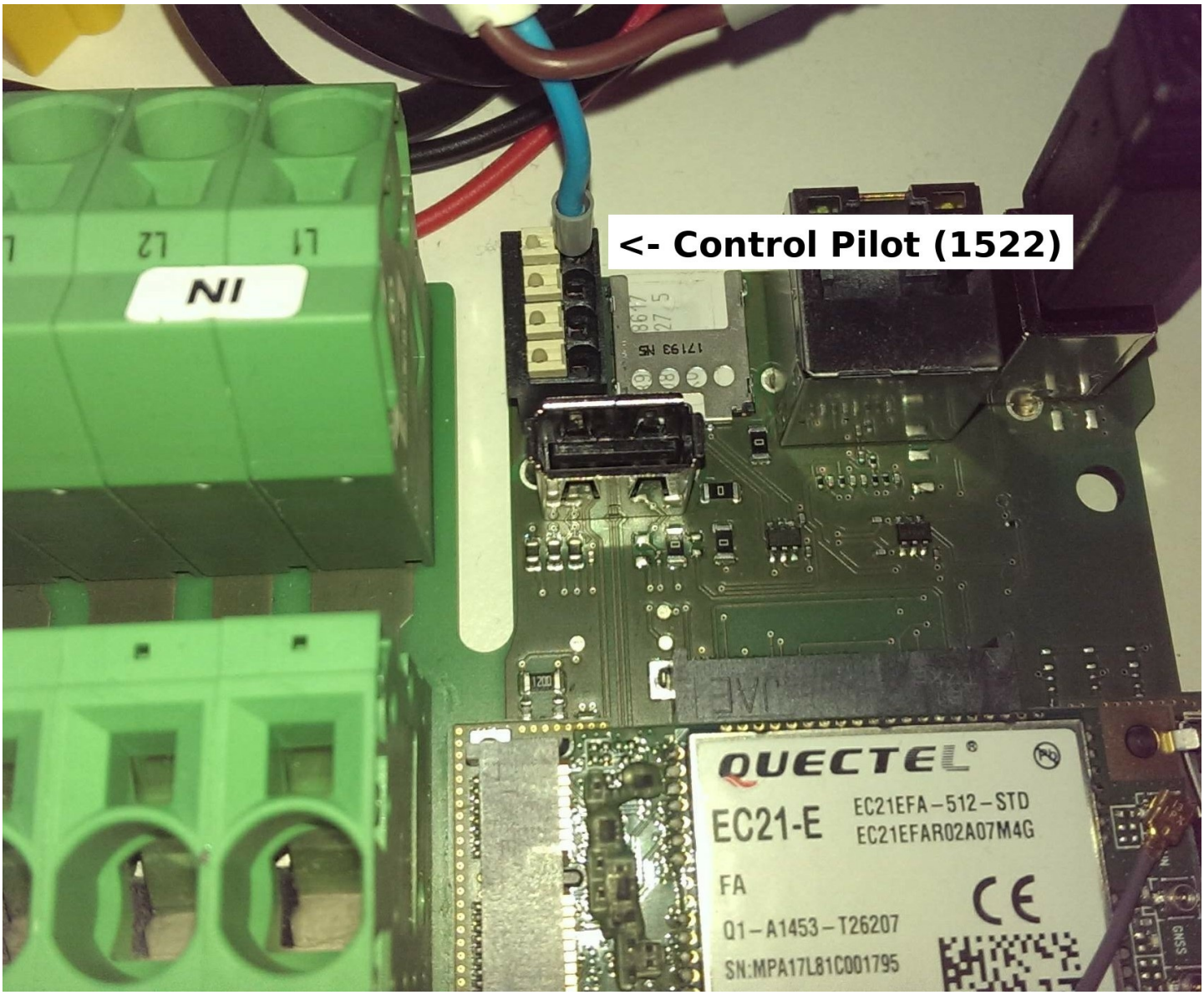
PLC (Power Line Communication)

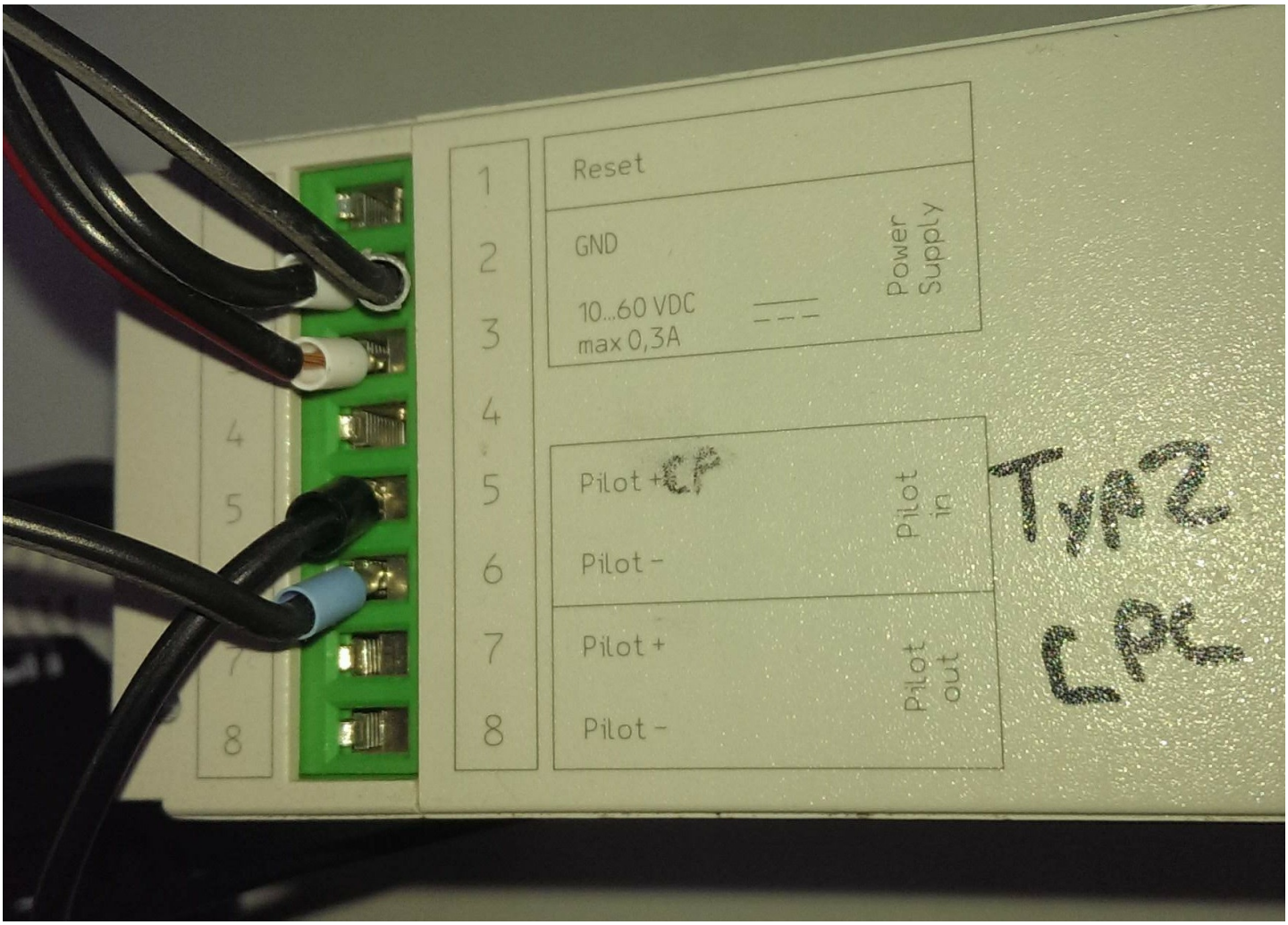
EOL Test Module PLC

This module allows testing the PLC. You will need a Modem from INSYS: INSYS Powerline GP

Here is a video tutorial that uses this documentation as a guide onYouTube

1. Connect an Ethernet cable (not crossover) from the modem directly to the PCs eth0 ethernet port.
2. Connect your Control Pilot (CP) to slot 5 (Pilot +)
3. Connect +12 V to slot 3,
4. Connect GND to slot 2 (GND) and slot 6 (Pilot -)
5. Slots 1, 4, 7, 8 should be unused on the modem.





- On the PC:
  Let's configure the modem!

LINUX:
Download and build the plctool program:

```
cd ~
git clone https://github.com/qca/open-plc-utils
cd open-plc-utils
make && sudo make install
```

download HPGP1.pib file from https://office.elinc.de/svn/ebee/XChange/Bender/97_Transfer/HPGP1.pib and copy it to a folder such as ~/Downloads

The HPGP1.pib file contains a driver that configures the modem for communication with the CP over power line.

Load the PIB

To load the drivers from your PC on linux run:

```
# ifconfig eth0 192.168.111.1
# plctool -i eth0 -P ~/Downloads/HPGP1.pib
eth0 00:B0:52:00:00:01 Start Module Write Session
eth0 00:B0:52:00:00:01 Flash HPGP1.pib
RESERVED 0x00000000
NUM_OP_DATA 1
MOD_OP 0x11
MOD_OP_DATA_LEN 1423
....
....
....
RESERVED 0x00000000
MODULE_ID 0x7002
MODULE_SUB_ID 0x0000
MODULE_LENGTH 928
MODULE_OFFSET 0x03A00000
eth0 00:B0:52:00:00:01 Close Session
eth0 00:B0:52:00:00:01 Reset Device
eth0 00:05:B6:01:88:71 Resetting ...
#
```

The output above is truncated.
The Program should complete after flashing the file.

If you receive this message:

```
eth0 00:B0:52:00:00:01 Start Module Write Session
plctool: ModuleSession: Read timeout or network error
```

then you probably do not have the cable connected correctly.

If you receive this message:

```
eth0 .... Close Session
eth0 .... Individual Module Error (0x31): Device refused request
```

Then you need to set the Device Access Key (DAK).

Newer modems do not allow you to flash just any pib. You will need to prepare your pib first by setting the DAK. To get this DAK read it from your device:

```
plctool -i eth0 -I
```

You should see a line like this:

```
...
DAK AA:BB:CC:DD:EE:FF:00:11:22:33:44:55:66:77:88:99
...
```

Now modify your HPGP1.pib using the modpib program.

```
# modpib -D AA:BB:CC:DD:EE:FF:00:11:22:33:44:55:66:77:88:99 ~/Downloads/HPGP1.pib
```

Now go back to: **Load the PIB** and try again.

...

NOTE Only the instructions for windows have been tested. If modifying and uploading the pib did not work, try the method described for windows, changing the commands as needed.

The PC should now be set up successfully.
Continue with flashing the firmware on the board...

WINDOWS:
To set up the Modem under Windows we use the "Powerline GP Tools Windows".

NOTE: I place a description for variable data in [ ].
INFO: For more info on the PLC Tool read its documentation.

setup:

1. Connect your PC to the modem using a LAN cable and set your network adapter to the IP address: "192.168.111.1/24".
2. Now download and unzip the open-plc-utils for Windows and navigate into the folder with CMD or PowerShell.
   "*TIPP: Hold shift and right click in the Explorer to get the "open PowerShell here" option*".
3. To use the GP Tools we need the number of the interface the modem is connected to. By typing `netsh interface ipv4 show interfaces` we get a neat list of all interfaces with their names, we need the numbers under Idx.
4. To check if the modem is reachable and get the MAC address we power it up and use the command `.\plctool.exe -i[interface number] -I`. This gives use a overview of the settings the modem currently operates on.
   If you get `plctool.exe: Identity2: Read timeout or network error` you need to check your interface settings or connection.
   If you get anything else, no problem we just want to check the connection and get the MAC address.

download the PIB:

1. The MAC address should now been displayed in some way or is written on the side of the modem (but that might be outdated or overwritten).
2. Now we will download the PIB File from the modem by typing `.\plctool.exe -i[interface number] [modem MAC] -p device.pib`.
   You should just get `nic7 [modem MAC] Read Module from Memory`
   If you get an error change the MAC address in the command to the correct one.

change the PIB:

1. You should get the HPGP1.pib and copy it into the folder with the GP Tools.
2. Now we use `.\modpib.exe HPGP1.pib -v` to get a list of its settings.
   more info on the displayed data here
3. We need the values NMK, NET, MFG and USR in our device.pib file. To write this data we use modpib.exe.
   Note: for more info about the modpib.exe tool type `.\modpib.exe -?`
   Use the command `./modpib.exe device.pib -N [target NMK]` to write the NMK Value.
   Use the command `./modpib.exe device.pib -T [target NET]` to write the NET Value.
   Use the command `./modpib.exe device.pib -S [target MFG]` to write the MFG Value.
   Use the command `./modpib.exe device.pib -U [target USR]` to write the USR Value.
4. Check your written data with `./modpib.exe device.pib -v`.

upload PIB:

1. To upload the PIB file type `.\plctool.exe -i[interface number] -P device.pib`

Now the modem is setup and waiting for devicess to link with.

flash qca0 firmware on the controller

- REQUEST

  ```
  /network/plc?cmd=setupqca0[&nmk=X]
  ```

```
X: 32 hex characters optionally delimited in pairs by the ':' character.
Example: aabbccddeeff11223344556677889900
or       aa:bb:cc:dd:ee:ff:11:22:33:44:55:66:77:88:99:00
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>ok</value>
</response>
```

Starting with version 1.16, this command starts a thread in the background. Previous versions of the software return only once

the flashing has completed.
Once the qca0 has been setup, you will also see the "Link" LED light up on the modem which was previously off.

check the status of the qca0 setup (V1.16+)

- REQUEST

```
/network/plc?cmd=qca0_was_setup
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>ok</value>
</response>
```

Proceed to call ifconfig down then up, then ping from thenetwork URLs:

http://192.168.123.123:8888/network/qca0?cmd=ifconfig&addr=192.168.111.123
http://192.168.123.123:8888/network/qca0?cmd=ifconfig&status=down
http://192.168.123.123:8888/network/qca0?cmd=ifconfig&status=up
http://192.168.123.123:8888/network/qca0?cmd=ping&addr=192.168.111.1&count=4

notes to ping

The controller software (>= V5.20) enables a firewall which blocks qca0 ping. The EOL Test Server (>= V1.38) tries to disable the firewall at startup. This requires the EOL Test Server running with root privileges.

RCMB communication

EOL Test Modul RCMB

The RCMB is the yellow ring shaped device that measures residual current of cables that pass through its center. This module allows reading it's values, version and resetting it.

get values

- REQUEST

```
/diag/rcmb?cmd=values
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<values>
  <value key="RMS">0</value>
  <value key="DC">0</value>
  <value key="R1">128</value>
  <value key="R2">32</value>
  <value key="R3">4</value>
  <value key="PE-byte-0">1</value>
  <value key="PE-byte-1">0</value>
  <value key="PE-byte-2">0</value>
  <value key="PE-byte-3">0</value>
  <value key="PE-byte-4">0</value>
</values>
</response>
```

get version

- REQUEST

```
/diag/rcmb?cmd=version
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>D0569V1.01</value>
</response>
```

reset controller or measurements

- REQUEST

```
/diag/rcmb?cmd=reset&what=[controller|measurement]
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>success</value>
</response>
```

set norm IEC62955/IEC62752

- REQUEST

```
/diag/rcmb?cmd=norm&set=[IEC62955|IEC62752]
```

- RESPONSE

```
<?xml version="1.0"?>
<response><result>success</result></response>
```

get current mode

- REQUEST

```
/diag/rcmb?cmd=norm&get=[IEC62955|IEC62752]
```

- RESPONSE

```
<response>
<result>success</result>
<value>false|true</value>
</response>
```

enable/disable PE check

- REQUEST

```
/diag/rcmb?cmd=pe_check&set=[enable|disable]
```

- RESPONSE

```
<response>
<result>success</result>
<value>enabled|disabled</value>
</response>
```

get 'PE check' state

- REQUEST

```
/diag/rcmb?cmd=pe_check&get=status
```

- RESPONSE

```
<response>
<result>success</result>
<value>enabled|disabled</value>
</response>
```

asic (Get asic settings)

- REQUEST

```
/diag/rcmb?cmd=asic
```

- RESPONSE

```
<response>
<result>success</result>
<values>
<value key="Gain">31890</value>
<value key="Offset">65480</value>
</values>
</response>
```

selftest (Start functional test)

- REQUEST

```
/diag/rcmb?cmd=selftest
```

- RESPONSE

```
<response>
<result>success</result>
</response>
```

RFID

EOL Test Module RFID

The RFID Module is used to test the RFID reader.

read from RFID reader

this function opens the rfid reader and tries to read a value 40 times, about once every second. If no tag was read during this time it returns an error. If a tag was read, that tag is returned. There is also an audible buzz from the buzzer when a tag was read successfully.

- REQUEST

```
/diag/rfid?cmd=read
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>[RFID TAG]</value>
</response>
```

SysInfo

EOL Test Module Sysinfo

The sysinfo module reports whether it's safe to reboot when doing an upgrade while testing.

The TA_EOLT software detects if the opkg-cl tool is installing new firmware and prevents the system from restarting automatically.

system is safe to reboot

This function returns successfully when the software update process has completed. It is not responsible for the installation, it only checks if the upgrade process has terminated.

- REQUEST

```
/diag/sysinfo?cmd=reboot_ready
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

Temperature

EOL Test Module Temperature

The Temperature module allows you to check the temperature on the board. Two values are read from files inside the /sys folder.

read temperature values

- REQUEST

```
/diag/temperature?cmd=values
```

The results are in millidegree celsius.

- RESPONSE

```
<?xml version="1.0"?>
<response>
  <result>success</result>
  <values>
    <value location="1">52500</value>
    <value location="2">25000</value>
    <value location="3">37.513618</value>
  </values>
</response>
```

UART sending and receiving

EOL Test Modul UART

The UART Modul is allows to send and receive hex strings on any uart device (modem, rcmb, ...)

open

- REQUEST

```
/diag/serial?cmd=open&dev=device[&baud=baud][&mode=mode]
```

```
device  device name, e.g. /dev/ttyUSB2
baud    baud rate (optional, 19200 default)
mode    data bits, parity, stop bits (optional, 8N1 default)
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
  <result>success</result>
</response>
```

close

- REQUEST

```
/diag/serial?cmd=close&dev=device
```

```
device  device name, e.g. /dev/ttyUSB2
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
  <result>success</result>
</response>
```

send

- REQUEST

```
/diag/serial?cmd=tx&dev=device&value=value
```

```
device  device name, e.g. /dev/ttyUSB2
value   hex string, e.g. 41540d0a for "AT\CR\LF"
crc     optional, ["crc16"], sends an additional crc16, can be used for RCMB testing
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
  <result>success</result>
</response>
```

receive

crc optional, ["crc16"], sends an additional crc16, can be used for RCMB testing

- REQUEST

```
/diag/serial?cmd=rx&dev=device[&timeout=timeout]
```

```
device  device name, e.g. /dev/ttyUSB2
timeout receive timeout in sec (optional, default 1 sec)
crc     optional, ["crc16"], expects the last rx'ed bytes to be a crc16can be used
        for RCMB testing
```

- RESPONSE

```
<response>
<result>success</result>
<value>41540d0d0a4f4b0d0a</value>
</response>
```

set_rts

opens device
calls `modbus_rtu_set_rts`
closes device

- REQUEST

```
/diag/serial?cmd=set_rts&dev=[D]&state=[S]
```

```
D device such as /dev/ttyS1
S state 0..2
   0 = MODBUS_RTU_RTS_NONE
   1 = MODBUS_RTU_RTS_UP
   2 = MODBUS_RTU_RTS_DOWN
```

- RESPONSE

```
<response>
<result>success</result>
</response>
```

USB Devices

EOL Test Module USB Devices

The Devices Module is used to test the usb devices

check if a usb block device is present

- REQUEST

```
/diag/usb?cmd=stickpresent
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>USB block device is present</value>
</response>
```

or

```
<?xml version="1.0"?>
<response>
<result>fail</result>
<cause>could not detect any usb block device</cause>
</response>
```

Detection may take up to 10 seconds before failing.

list usb device paths

- REQUEST

```
/diag/usb?cmd=list_present_sticks
```

- RESPONSE

```
<response>
<result>success</result>
<values>
<value block_devices="USB-0">/sys/bus/usb/devices/1-1.1</value>
<value block_devices="USB-1">/sys/bus/usb/devices/1-1.2</value>
</values>
</response>
```

count usb smart hub ports

- REQUEST

```
/diag/usb?cmd=uhubctl&action=count
```

- RESPONSE

```
<response>
  <result>success</result>
  <value>4</value>
</response>
```

list usb ports of the hub

- REQUEST

```
/diag/usb?cmd=uhubctl&action=list
```

- RESPONSE

```
<response>
  <result>success</result>
  <values>
    <value usb-ports="0"> Port 1: 0100 power</value>
    <value usb-ports="1"> Port 2: 0100 power</value>
    <value usb-ports="2"> Port 3: 0103 power enable connect [0424:9e00]</value>
    <value usb-ports="3"> Port 4: 0100 power</value>
  </values>
</response>
```

set power state of specified port and return state

- REQUEST

  ```
  /diag/usb?cmd=uhubctl&action=power&port=X&set=[on/off]
  ```

- RESPONSE

Example:

```
http://192.168.123.123:8888/diag/usb?cmd=uhubctl&action=power&port=3&set=off
```

```
<response>
  <result>success</result>
  <values>
    <value usb-ports="0"> Port 1: 0100 power</value>
    <value usb-ports="1"> Port 2: 0100 power</value>
    <value usb-ports="2"> Port 3: 0000 off</value>
    <value usb-ports="3"> Port 4: 0100 power</value>
  </values>
</response>
```

Versions

EOL Test Module Versions

The versions Module is used to get version information of different software and hardware.

Get Kernel Version

- REQUEST

```
/diag/version?sw=[S]
```

```
[S]  ebee|kernel|meta|sap|hardware|ta_eolt|all
```

- RESPONSE

ebee

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>5.30.0-15933-accede7f0</value>
</response>
```

kernel

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>
Linux cp 4.19.78-linux4sam-6.2-icp #1 Tue Aug 9 23:06:09 CEST 2022 armv5tejl GNU/Linux
</value>
</response>
```

meta

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>968-b3fefd6</value>
</response>
```

sap

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>530.0.146</value>
</response>
```

hardware (ICC1324)

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>ICC1324 AHM2B01117 (32)</value>
</response>
```

ta_eolt

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>1.39-259-04207aaa</value>
</response>
```

all

```
<?xml version="1.0"?>
<response>
<result>success</result>
<values>
<value software="ebee">5.30.0-15933-accede7f0</value>
<value software="hardware">ICC1324 AHM2B01117 (32)</value>
<value software="kernel">
Linux cp 4.19.78-linux4sam-6.2-icp #1 Tue Aug 9 23:06:09 CEST 2022 armv5tejl GNU/Linux
</value>
<value software="meta">968-b3fefd6</value>
<value software="sap">530.0.146</value>
<value software="ta_eolt">1.39-259-04207aaa</value>
</values>
</response>
```

Whitelist (RFID Cache)

EOL Test Module Whitelist

The Whitelist Module is used to add RFIDs to the fixed local and OCPP cache.

list entries

Show a list of cache entries

- REQUEST

  /diag/whitelist?cmd=list&list=[fixed|ocpp]

which list to **show**:
[fixed] **show** fixed **local** list cache
[ocpp] **show** OCPP cache

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<values>
  <value tag="0">abcdef</value>
  <value tag="1">1234567890</value>
  <value tag="2">3eba570</value>
</values>
</response>
```

add entries

add a list of rfid tags to the specified cache

- REQUEST

  /diag/whitelist?cmd=add&list=[fixed|ocpp]&tags=[T]

which list to **add** to:
[**fixed**] **fixed** local list cache
[**ocpp**] OCPP cache

[**T**] list of rfid tags separated **by** the colon character ":"
Ex. abcdef:1234567890:3eba570

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

clear entries

Delete all cache entries from specified list

- REQUEST

  /diag/whitelist?cmd=clear&list=[fixed|ocpp]

[**fixed**] **fixed** local list cache
[**ocpp**] OCPP cache

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

WLAN

EOL Test Modul Network

The EOL WLAN module tests the wlan0 interface

connect to an existing wifi network

Connect to SSID [S] using password [P] and start the dhcp client in the background.

- REQUEST

```
/network/wifi?cmd=connect&ssid=[S]&password=[P]
```

```
[S]  The SSID to connect to
[P]  The WPA Password
Note: certain characters may need to be converted to url encoding
so that they are not interpreted as special characters for URLs.
example: 'pass#word' must be changed to 'pass%23word'. If you do
not do this, 'pass' will be saved as the password and the connection
may fail.
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

Note: Please check the status of the interface to confirm that the device connects successfully to the network.

kill dhcp client

In case you want to set a manual IP address

- REQUEST

```
/network/wifi?cmd=killdhcpc
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

interface status

- REQUEST

```
/network/wifi?cmd=status
```

- RESPONSE

When the interface is not connected

```
<?xml version="1.0"?>
<response>
<result>success</result>
<values>
<value parameter="Access Point">Not-Associated</value>
<value parameter="ESSID">off/any</value>
</values>
</response>
```

When the interface is connected

```
<?xml version="1.0"?>
<response>
<result>success</result>
<values>
<value parameter="Access Point">78:8A:20:8A:2C:24</value>
<value parameter="ESSID">">Ebee-Smart<"</value>
<value parameter="Signal level">-23 dBm</value>
</values>
</response>
```

I2C

EOL Test Module I2C (ICC1324)

The I2C Module is used to test RAW communication with I2C devices.

I2C write

- REQUEST

  ```
  /diag/i2c?cmd=write&address=[A]&data=[D]
  ```

[A] I2C device address (hex) [D] data to write (hex)

Example: write data 0x38FF to I2C address 0x33 (RFID120: LP3036 Reset)

- REQUEST

  ```
  /diag/i2c?cmd=write&address=33&data=38FF
  ```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

I2C read

- REQUEST

  ```
  /diag/i2c?cmd=read&address=[A]&len=[L]
  ```

[A] I2C device address (hex) [L] number of bytes to read (decimal)

I2C writeread (write with subsequent read)

- REQUEST

  ```
  /diag/i2c?cmd=read&address=[A]&data=[D]&len=[L]
  ```

[A] I2C device address (hex) [D] data to write (hex) [L] number of bytes to read (decimal)

Example: write data 0x87 to I2C address 0x29 and read 1 byte (RFID120: read LTR329 manufacturer ID)

- REQUEST

  ```
  /diag/i2c?cmd=writeread&address=29&data=87&len=1
  ```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

Settings

EOL Test Module Settings

The Settings Module is used to write / read persistent settings.

set value

- REQUEST

  ```
  /diag/settings?cmd=set&what=[W]&key=[K]&value=[V]&line2=[V2]
  ```

[W] persistency / nor_store / otp
[K] key
[V] value
[V2] optional value (will be written into the 2nd line of the persistency file)

get value

- REQUEST

  ```
  /diag/settings?cmd=get&what=[W]&key=[K]
  ```

[W] persistency / nor_store / persistency_line2 (reads 2nd line of persistency file) / otp
[K] key

delete value

- REQUEST

  ```
  /diag/settings?cmd=delete&what=[W]&key=[K]
  ```

[W] persistency / nor_store
[K] key

Notes to "persistency"

- files with name [K] are read/written from/to the persistency directory (/home/charge/persistency)
- if the EOL Test server is running as user "root", the file owner is set to "charge"

Example: set controller SNR (persistency)

- REQUEST

  ```
  /diag/settings?cmd=set&what=persistency&key=SerialNumber_custom&value=12345
  ```

- RESPONSE

  ```
  <?xml version="1.0"?>
  <response>
  <result>success</result>
  </response>
  ```

Example: get controller SNR (persistency)

- REQUEST

  ```
  /diag/settings?cmd=get&what=persistency&key=SerialNumber_custom
  ```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>12345</value>
</response>
```

Example: set eth0 MAC address (nor_store)

- REQUEST

```
/diag/settings?cmd=set&what=nor_store&key=eth0_MAC&value=12:34:56:78:9A:BC
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
</response>
```

Example: get eth0 MAC address (nor_store)

- REQUEST

```
/diag/settings?cmd=get&what=nor_store&key=eth0_MAC
```

- RESPONSE

```
<?xml version="1.0"?>
<response>
<result>success</result>
<value>12:34:56:78:9A:BC</value>
</response>
```

Example: set bootconfig OTP (required for ICC1324 to boot from NOR flash)

- REQUEST

```
/diag/settings?cmd=set&what=otp&key=bootcfg
```

- RESPONSE

```
<response>
<result>success</result>
</response>
```

Example: get bootconfig OTP state (ICC1324)

- REQUEST

```
/diag/settings?cmd=get&what=otp&key=bootcfg
```

- RESPONSE

```
<response>
<result>success</result>
<value>BootCfg OTP OK</value>
</response>
```