WÜRTH ELEKTRONIK

PROTEUS-III REFERENCE MANUAL

2611011024000

VERSION 0.11

NOVEMBER 26, 2019

Revision history

Manual version	FW version	HW version	Notes	Date
0.11	1.0	1.1	• Initial release	November 2019

⋆ For firmware history see chapter `Firmware history`

Abbreviations and abstract

Abbreviation	Name	Description
BTMAC		Bluetooth® conform MAC address of the module used on the RF-interface.
CS	Checksum	Byte wise XOR combination of the preceding fields.
BLE	Bluetooth® Low Energy	According to Bluetooth® specification.
BT	Bluetooth®	According to Bluetooth® specification.
DSSS	Direct sequence spread spectrum	Technique to spread a message on the radio
DTM	Direct test mode	Mode to test Bluetooth® specific RF settings.
FEC	Forward error correction	Technique to correct received erroneous radio messages
GAP	Generic Access Profile	The GAP provides a basic level of functionality that all Bluetooth® devices must implement.
I/O	Input/output	Pinout description.
LESC	Low energy secure connection	Elliptic curve encryption method for BLE encryption and authentication
LPM	Low power mode	Mode for efficient power consumption.
LRM	Long range mode	Radio mode with higher range and lower throughput.
MAC		MAC address of the module.
MTU	Maximum transmission unit	Maximum packet size of the Bluetooth® connection.
Payload		The intended message in a frame / package.
RF	Radio frequency	Describes wireless transmission.
RSSI	Receive Signal Strength Indicator	The RSSI indicates the strength of the RF signal. Its value is always printed in two's complement notation.
Soft device		Operating system used by the nRF52 chip.
User settings		Settings to configure the module. Any relation to a specific entry in the user settings is marked in a special font and can be found in chapter 8.
UART	Universal Asynchronous Receiver Transmitter	Allows the serial communication with the module.
[HEX] 0xhh	Hexadecimal	All numbers beginning with 0x are hexadecimal numbers. All other numbers are decimal, unless stated otherwise.

Contents

1 Introduction

1.1 Operational description

The Proteus-III module is a radio sub module/device for wireless communication between devices such as control systems, remote controls, sensors etc. . On the basis of Bluetooth® Low Energy 5.0 it offers a fast and secure data transmission of data packages between two or more parties (point to point topology). A serial interface (UART) is available for communication with the host system.

The Proteus-III uses the BLE standard to provide general data transmission between several devices. The standard itself offers a wide range of configurations and possibilities to suit and optimize sophisticated customer applications. To fulfill the needs and specifications of such applications a tailored firmware can be developed on the basis of the Proteus-III hardware. This includes the connection and communication to custom sensors, custom BLE profiles, timing configurations, security configuration as well as power consumption optimizations.

1.1.1 Key features

The Proteus-III offers the following key features that are described in the manual in more detail:

SPP-like connection-based secured data transmission: The Proteus-III firmware implements an SPP-like BLE-profile that allows the bidirectional data transmission between several Proteus-III and/or to other BLE devices implementing the AMBER SPP profile. Any module in the network can initiate a connection setup. Secured connections allow the transmission of encrypted data.

Fast sensor data transmission via Beacons: The Proteus-III supports the transmission and reception of Beacons. Beacons are fast broadcast messages that allow the energy-efficient unidirectional transmission of data. Especially in sensor networks, this feature is suitable for the frequent transmission of measurement data as it avoids the need for a connection-based communication and therefore is more energy efficient.

Advanced customization capabilities: The configurable Device Information Service (DIS), the UUID and the appearance of the BLE profile, enable to personalize the Proteus-III to fuse with the user's end product.

Low power position sensing solutions: The current TX power of any Proteus-III is always transmitted with each advertising packet when the module is in command mode. With this, distance estimation and position sensing solutions can be realized conveniently by performing a passive scan.

Fast serial interface: The Proteus-III offers a UART-interface to communicate with a host using a user-defined baud rate and a simple command interface.

Latest microprocessor generation provided by Nordic Semiconductor nRF52 series: The heart of the Proteus-III is a BLE-chip of the nRF52 series offering high performance values combined with low power consumption. It is a 32 Bit ARM Cortex-M4F CPU with 1024kB flash + 256kB RAM and up to 8dBm output power.

Bluetooth® 5 stack: The Bluetooth® 5 stack enables fast and energy efficient data transmission using state-of-the-art technology of Nordic Semiconductors.

High throughput mode: The Proteus-III contains the so called "High throughput mode" that allows to send up to 4 data packets per connection interval. This increases significantly the throughput. This mode and its usage is described in Proteus-III - Application Note 4".

All BLE roles supported: The integrated BLE stack supports all BLE roles. Depending on the current state of operation the Proteus-III firmware automatically switches its role to execute the user's instructions.

Flexible wired interfacing: If custom hardware does not support UART communication or in case of a host less implementation, the Proteus-III is equipped with extra pins suited for custom device/sensor connection. With help of these, a tailored firmware can be developed which is optimized to the customer's needs. The pins can be configured to various functions such as UART, SPI, I2C, ADC, PWM, NFC and GPIO.

OTA firmware update: The Proteus-III firmware provides over the air firmware update capabilities. Firmware updates can be applied using the Nordic Apps for cell phones.

Peripheral only mode: The Proteus-III firmware provides the "peripheral only" operation mode (see chapter 10), that allows the easy adaption of already existing custom hardware with the BLE interface. By default, this mode offers the static passkey pairing method with bonding and a transparent UART interface. With this, custom hardware can be accessed by mobile BLE devices (like smart phones including a custom App) using an authenticated and encrypted BLE link without the need of configuring the module.

Additional Bluetooth® 5 radio modes: The Proteus-III provides the advanced radio modes (2 MBit and LE coded, see chapter 5.9) that have been introduced with Bluetooth® 5.0. The 2 MBit mode allows to use the double data rate. The LE coded mode enables higher transmission distances.

Long range connect: For backward compatibility reasons, a BLE connection is usually setup using the legacy 1MBit radio mode and can then be updated to long range mode. The Proteus-III allows in addition to setup the connection immediately using the long range mode, such that even connections can be initiated on high distances. More information can be found in chapter 5.10.

Remote GPIOs: The Proteus-III firmware allows to switch free module GPIOs via remote control. More information can be found in chapter 11.

1.1.2 Connectivity

The BLE standard allows to setup a network with various BLE devices from different manufacturers. To be able to communicate with Proteus-III devices, the AMBER SPP-like profile must be known and implemented by all network participants. Thus arbitrary BLE devices (like iOS or Android devices) must implement this profile, too. To do so, the Proteus-III application note 1 contains the design data of the AMBER SPP-like profile.

1.2 Block diagram
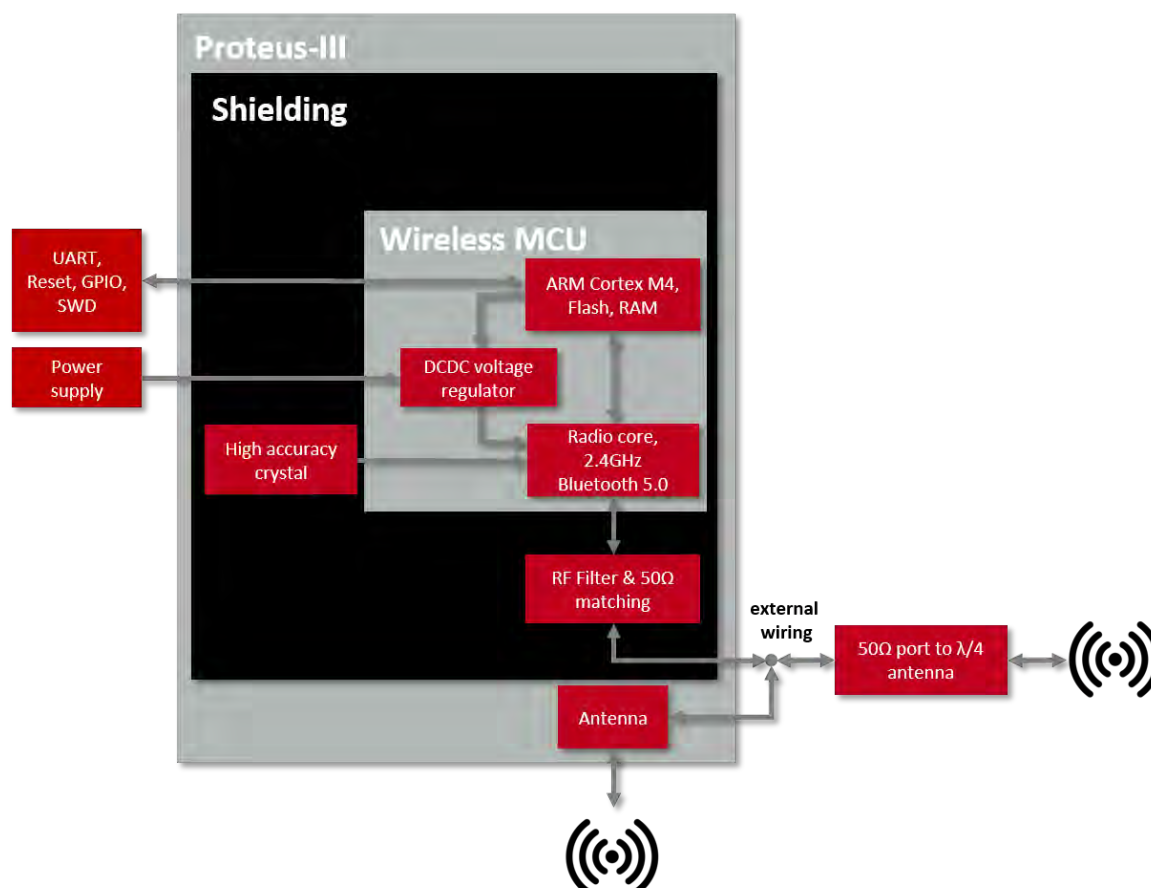
Figure 1: Block diagram of the module

1.3 Ordering information

WE order code	Description
2611011024000	Proteus-III Bluetooth® Smart Module, Tape & Reel
2611011024009	Development Kit including 3×Proteus-III
2611019224001	Bluetooth® Smart Evaluation Board with Proteus-III

Table 1: Ordering information

2 Electrical specifications

As not otherwise stated measured on the evaluation board Proteus-III-EV with T=25℃, VDDS=3V, f=2.44GHz, internal DC-DC converter in use.

2.1 Recommended operating conditions

Description	Min.	Typ.	Max.	Unit
Ambient temperature	-40	25	85	℃
Supply voltage (VDDS)	1.8	3	3.6	V
Supply rise time (0V to $\geq$ 1.7V)			60	ms

Table 2: Recommended operating conditions

> The on-chip power-on reset circuitry may not function properly for rise times longer than the specified maximum.

> A step in supply voltage of 300 mV or more, with rise time of 300 ms or less, within the valid supply range, may result in a system reset.

> An instable supply voltage may significantly decrease the radio performance and stability.

2.2 Absolute maximum ratings

Description	Min.	Typ.	Max.	Unit
Supply voltage (VDD)	-0.3		+3.9	V
Voltage on any digital pin, VDD$\leq$3.6V	-0.3		VDD+0.3	V
Voltage on any digital pin, VDD$\geq$3.6V	-0.3		3.9	V
Input RF level			10	dBm
Flash endurance	10 000			Write/erase cycles

Table 3: Absolute maximum ratings

2.3 Power consumption

2.3.1 Static

Continuous test mode	Min.	Typ.	Max.	Unit
TX current consumption at +8 dBm		16.4[1]		mA
TX current consumption at 0 dBm		6.4[1]		mA
RX current consumption		6.25[1]		mA
Sleep (system off mode)		0.4		µA
TX current consumption at +8 dBm		18.9[2]		mA
TX current consumption at 0 dBm		8[2]		mA
RX current consumption		7.7[2]		mA

Table 4: Power consumption for 100% transmission/reception

Due to the BLE time slot operation, the real operating currents are reduced significantly and depend on the user selectable advertising and connection interval settings.

[1]Transmitter only with DC/DC converter from nRF52 data sheet.
[2]Full module power consumption.
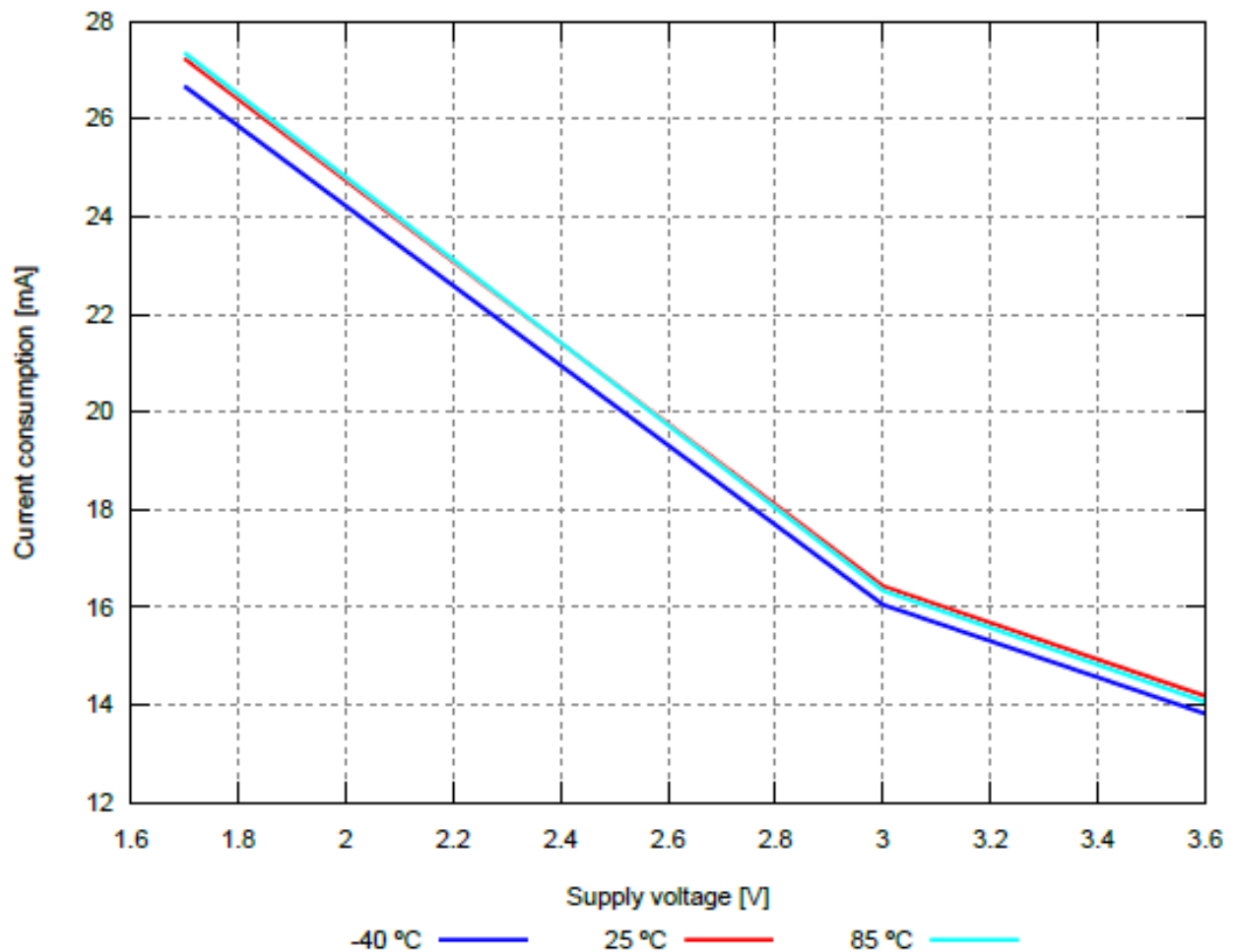
Figure 2: Radio transmitting @ 8 dBm output power, 1 Mbps BLE mode, Clock = HFXO, Regulator = DC/DC (typical values)

2.3.2 Dynamic

Besides the static TX, RX, idle and sleep current the average current is of interest. Here an example for a typical behavior of a peripheral device in advertising mode (see Figure 3). Currents and state durations are dependent on the configuration of the module. In this state the module transmits the advertising packets on the 3 advertising channels.

Nordic Semiconductor provides an online tool calculating the average current of a Bluetooth® connection. It can be accessed at *https://devzone.nordicsemi.com/power/*.

Test setup

Chip	nRF52840 QIAAC0
Softdevice	s140 6.1.0
Voltage	3.0 V
Regulator	DCDC

BLE event details

Interval	45.00 ms
Length	3.61 ms

Data transmission

On air data rate	1 Mbps

Current consumption

BLE event total charge	13.29 µC
LF clock calibration current	1.0 µA
Idle current	3.1 µA
Total average current	299 µA
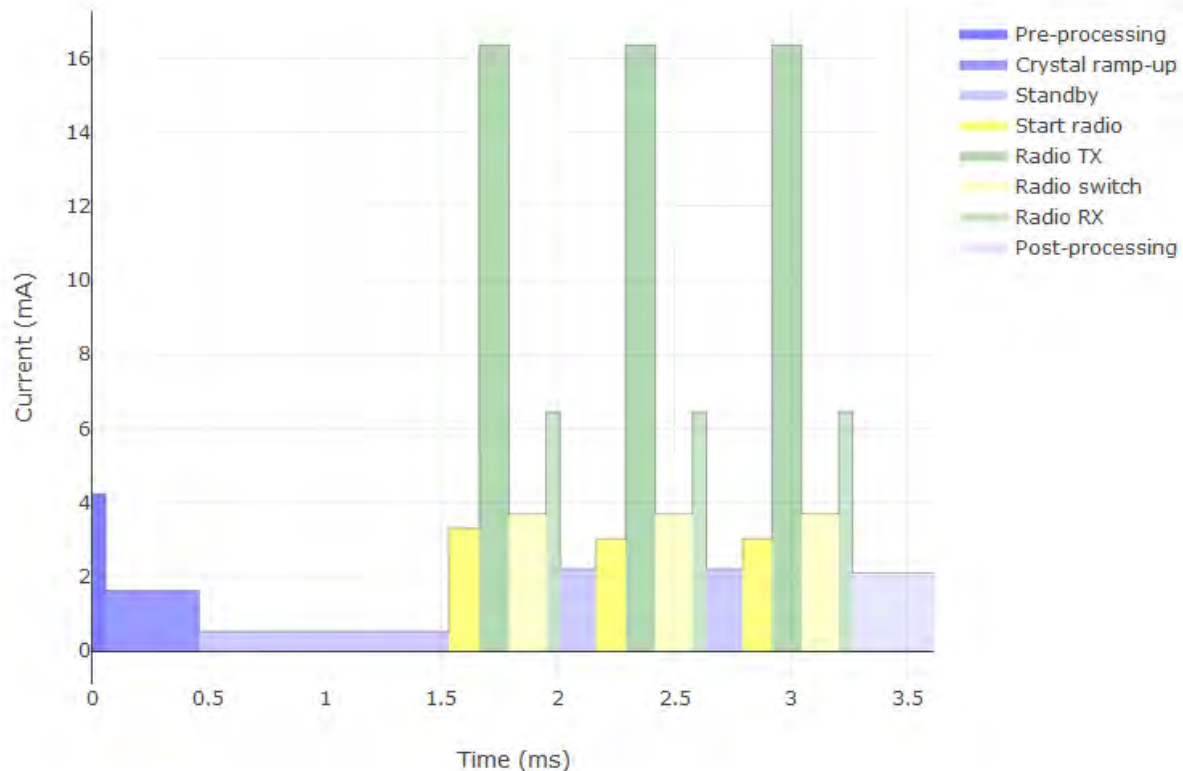
Figure 3: Current consumption calculation in advertising mode with 40ms advertising interval with 8 dBm output power, UART disabled

2.4 Radio characteristics

50Ω conducted measurements from nRF52 data sheet

Description	Min.	Typ.	Max.	Unit
Output power	-40		+8	dBm
Input sensitivity ($\leq$ 37 Bytes, BER=1E-3)		-95[1]		dBm
RSSI accuracy valid range (±2dB)	-90		-20	dBm
Enable TX or RX delay		140		μs
Enable TX or RX delay (fast mode)		40		μs
Disable TX delay		6		μs
Disable RX delay		0		μs

Table 5: Radio parameters

2.5 Pin characteristics

Measurements from nRF52 data sheet

Description	Min.	Typ.	Max.	Unit
Input high voltage	0.7 ×VCC		VCC	V
Input low voltage	VSS		0.3 ×VCC	V
Current at VSS+0.4 V, output set low, standard drive, VDD $\geq$1.7V	1	2	4	mA
Current at VSS+0.4 V, output set low, high drive, VDD $\geq$ 2.7 V	6	10	15	mA
Current at VSS+0.4 V, output set low, high drive, VDD $\geq$ 1.7 V	3			mA
Current at VDD-0.4 V, output set high, standard drive, VCC $\geq$1.7V	1	2	4	mA
Current at VDD-0.4 V, output set high, high drive, VDD $\geq$ 2.7 V	6	9	14	mA
Current at VDD-0.4 V, output set high, high drive, VDD $\geq$ 1.7 V	3			mA
Internal pull-up resistance	11	13	16	kΩ
Internal pull-down resistance	11	13	16	kΩ

Table 6: Pin characteristics

---

[1]nRF52840 Rev.1, with build code CKAA-DA, CSP package, in DC/DC mode

3 Pinout
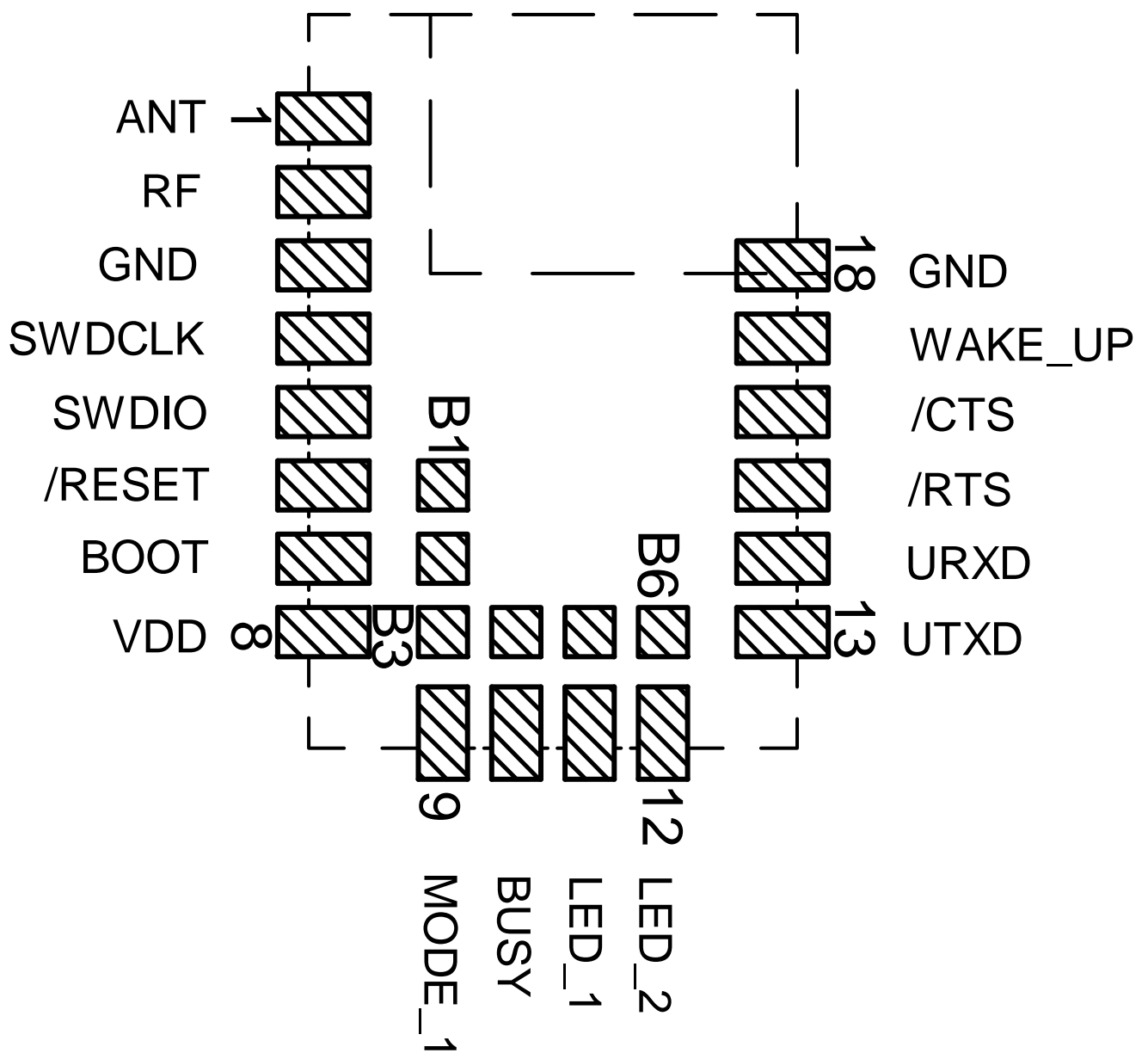
Figure 4: Pinout (top view)

No	µC Pin	Designation	I/O	Description
1		*ANT*	Input	RF connection to PCB antenna (see section 4.2)
2		*RF*	Output	50Ω RF connection to external antenna or on-board antenna via ANT (see section 4.2)
3		*GND*	Supply	Ground
4		*SWDCLK*	Input	Serial wire clock (SWD Interface). Uses internal pull down resistor. Do not connect if not needed.
5		*SWDIO*	Input	Serial wire input/output (SWD Interface). Uses internal pull up resistor. Do not connect if not needed.
6	P0.18	*/RESET*	Input	Reset pin. A low signal resets the module. Uses internal pull up resistor.
7	P0.02	*BOOT*	Input	Boot pin. A low signal during and short after reset starts the module in OTA bootloader mode. Uses internal pull up resistor[1]. Do not connect if not needed.
8		*VDD*	Supply	Supply voltage
9	P0.19	*MODE_1*	Input	Operation mode pin with internal pull down resistor[1] during start-up. Low level or open: Normal Mode. High level: Peripheral only Mode. Do not connect if not needed.
10	P0.22	*BUSY*	Output	Indicates if module is busy with data transmission when using Peripheral only Mode (see section 10.4.2). Do not connect, if not needed.
11	P0.00/XL1[3]	*LED_1*	Output	Indicates the module state (active high). Do not connect if not needed.
12	P0.01/XL2[3]	*LED_2*	Output	Indicates the module state (active high). Do not connect if not needed.
13	P1.08	*UTXD*	Output	UART (Transmission)
14	P1.09	*URXD*	Input	UART (Reception). Uses internal pull up resistor[1].
15	P0.11	*/RTS*	Output	/RTS signal, if flow control is enabled. Static low, otherwise. Do not connect if not needed.

Table 7: Pinout, first part

[1] Internal pull ups or pull downs are configured at startup by the firmware installed in the SoC. The pull up on the */RESET* pin cannot be disabled by firmware.
[2] NFC pins available for NFC function in custom firmware. The standard firmware of Proteus-III does not implement this function.
[3] Pins available to connect an external crystal in custom firmware. The standard firmware of Proteus-III does not implement this function.

No	µC Pin	Designation	I/O	Description
16	P0.12	/CTS	Input	/CTS signal, if flow control is enabled. Using internal pull down[1], otherwise. Do not connect if not needed.
17	P0.03	WAKE_UP	Input	Wake-up will allow leaving the system-off mode or re-enabling the UART. Uses internal pull up resistor[1]. Do not connect if not needed.
18		GND	Supply	Ground
B1	P0.09/NFC1[2]	B1	GPIO	Pin for remote GPIO access. Do not connect, if not needed.
B2	P0.10/NFC2[2]	B2	GPIO	Pin for remote GPIO access. Do not connect, if not needed.
B3	P0.23	B3	GPIO	Pin for remote GPIO access. Do not connect, if not needed.
B4	P1.00	B4	GPIO	Pin for remote GPIO access. Do not connect, if not needed.
B5	P0.21	B5	GPIO	Pin for remote GPIO access. Do not connect, if not needed.
B6	P0.07	B6	GPIO	Pin for remote GPIO access. Do not connect, if not needed.

Table 8: Pinout, second part

4 Quick start

4.1 Minimal pin connections
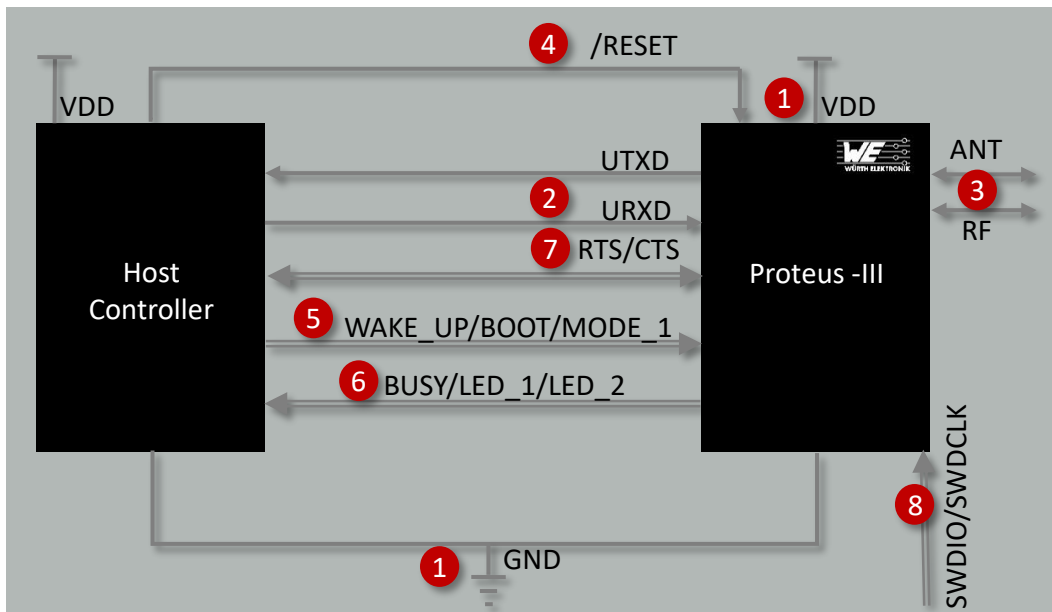
Figure 5: Minimal pin connections

The above image shows the steps to be performed to integrate the Proteus-III into a custom end device.

1. Supply voltage and ground
   First of all, connect the *VDD* and *GND* pins to supply the radio module with power.

2. UART serial interface to the host
   Connect the UART pins *UTXD* and *URXD* to the host to control the module via host.

3. Antenna connection
   Then the antenna configuration must be performed. See chapter `4.2`.

4. Reset
   Connect the */RESET* pin to the host to allow a hard reset of the module.

5. (Optional) Wakeup from sleep, FOTA and mode selection

   - Connect the *WAKE_UP* pin to the host controller to leave power saving modes, like the sleep mode.

   - Connect the *BOOT* pin to the host controller to set the module into boot mode to enable firmware updates via radio.

- Connect the *MODE_1* pin to the host controller to switch between command and peripheral only mode.

6. (Optional) Status indication
   Connect the *BUSY*, *LED_1* and *LED_2* pins to the host controller to allow easy indication of the status.

7. (Optional) UART flow control
   Connect the */RTS* and */CTS* pins to the host controller in case the UART flow control shall be used. This is mandatory for fast UART baud rates.

8. (Optional) Flash and debug interface
   We recommend to additionally have the pins *SWDIO* and *SWDCLK* accessible in order to support a fail-safe firmware update. A standard socket on the customer's PCB for connecting a flash adapter can be useful for debugging purposes (e.g. a JTAG 2*10 pin header with 2.54mm pin-to-pin distance).

If the module has to be connected to a PC, a converter (TTL to RS-232 or TTL to USB) has to be used. See chapter 3 for details on all pins. Please refer to the Proteus-III-EV schemes for a reference design.

> ⓘ The logic level of the module is based on 3V. A 5V logic level must not be connected directly to the module.

4.2 Antenna connection

Proteus-III's smart antenna configuration enables the user to choose between two antenna options:

4.2.1 On-board PCB antenna

The Proteus-III has an on-board PCB antenna optimized for operation in the 2.4 GHz frequency band. A simple short between the pins *RF* and *ANT* feeds the RF output of the module to the on-board antenna of the Proteus-III. In this configuration, the module does not require any additional RF circuitry.

4.2.2 External antenna

For applications that use an external antenna, the Proteus-III provides a 50Ω RF signal on pin *RF* of the module. In this configuration, pin *ANT* of the module has to be connected to ground and pin *RF* to the external antenna via 50Ω feed line. Refer to chapter 17 for further information.

4.3 Power up

After powering the module the */RESET* pin shall be hold for another Δt of 1ms after the *VDD* is stable to ensure a safe start-up. The module will send a `CMD_GETSTATE_CNF` to indicate "ready for operation" after the */RESET* pin was released.

> Applying a reset (e.g. a host temporarily pulling the */RESET* pin down for at least 1ms and releasing it again) after the VCC is stable will also be sufficient.
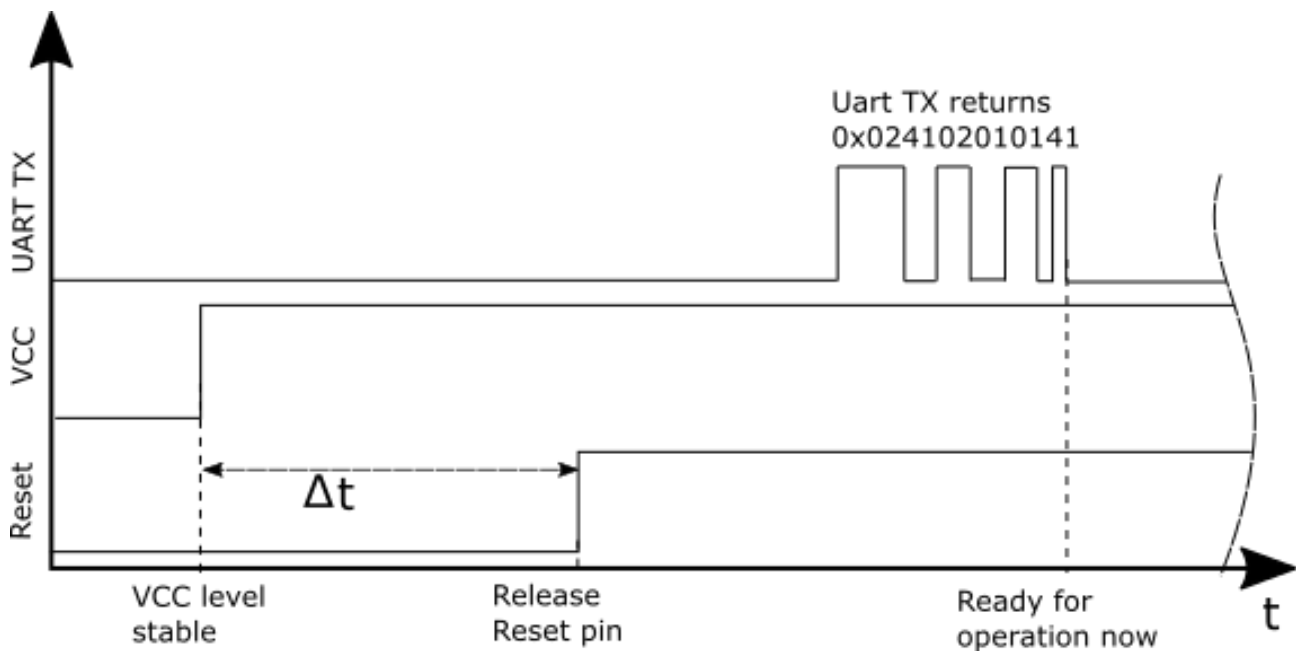
Figure 6: Power up

4.4 Quickstart example

This section describes how to quick start the data transmission between two Proteus-III modules. The goal is to setup a connection between module A and module B, transmit some data and close the connection again.

In this section, all packet data from or to the modules is given in **hexadecimal notation**. For quick testing, a pair of Proteus-III-EV is recommended.

Connect the two devices (modules, EV-boards or USB dongles) to a PC. A terminal program, for example *hterm*, is used to perform the communication via COM ports. The two corresponding COM ports have to be selected and opened with a default configuration of 115200 Baud, 8 data Bits, 1 stop Bit and parity set to none (8n1).

To reproduce the following sequence, note that, the `FS_BTMAC` of every module is different, thus it has to be replaced it in the commands below. In addition, the checksum has to be adjusted, when adapting any command. The command structure and checksum calculation is described in chapter 8.

Note that the module goes to `ACTION_SLEEP` mode if no connection is setup after `RF_AdvertisingTimeout` seconds. The module will indicate this using a `CMD_SLEEP_CNF`. In addition, the UART is disabled in `ACTION_SLEEP` mode. The default value is 0s, which means that it will run forever.

Connection setup and first data transmission

1. Power-up the modules and make their UARTs accessible by the host(s) (115200 Baud, 8n1). After the power-up or after reset the following sequence is sent from the module.

Info	Module A	Module B
⟸ Response `CMD_GETSTATE_CNF`: Module A started in `ACTION_IDLE` mode.	02 41 02 00 01 01 41	
⟸ Response `CMD_GETSTATE_CNF`: Module B started in `ACTION_IDLE` mode.		02 41 02 00 01 01 41

2. Request the `FS_BTMAC` of both modules.

Info	Module A	Module B
⟹ Request `CMD_GET_REQ` with settings index 4	02 10 01 00 04 17	
⟸ Response `CMD_GET_CNF`: FS_BTMAC of module A is 0x55 0x00 0x00 0xDA 0x18 0x00	02 50 07 00 00 55 00 00 DA 18 00 C2	
⟹ Request `CMD_GET_REQ` with settings index 4		02 10 01 00 04 17
⟸ Response `CMD_GET_CNF`: FS_BTMAC of module B is 0x11 0x00 0x00 0xDA 0x18 0x00		02 50 07 00 00 11 00 00 DA 18 00 86

3. Connect module A to module B via Bluetooth®.

Info	Module A	Module B
⇒ Request `CMD_CONNECT_REQ` with `FS_BTMAC` of module B	02 06 06 00 11 00 00 DA 18 00 D1	
⇐ Response `CMD_CONNECT_CNF`: Request understood, try to connect now	02 46 01 00 00 45	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 86 07 00 00 11 00 00 DA 18 00 50	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 86 07 00 00 55 00 00 DA 18 00 14
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet	02 C6 07 00 00 11 00 00 DA 18 00 F3 E3	
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet		02 C6 07 00 00 55 00 00 DA 18 00 F3 A7

4. Once the connection is active, data can be sent in each direction. Let us send a string "ABCD" from module B to module A.

> ! The RSSI values will be different in your tests.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "ABCD" to module A		02 04 04 00 41 42 43 44 06
⇐ Response `CMD_DATA_CNF`: Request received, send data now		02 44 01 00 00 47
⇐ Indication `CMD_DATA_IND`: Received string "ABCD" from `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xCA (-54dBm)	02 84 0B 00 11 00 00 DA 18 00 CA 41 42 43 44 90	
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully		02 C4 01 00 00 C7

5. Reply with "EFGH" to module B.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "EFGH" to module B	02 04 04 00 45 46 47 48 0E	
⇐ Response `CMD_DATA_CNF`: Request received, send data now	02 44 01 00 00 47	
⇐ Indication `CMD_DATA_IND`: Received string "EFGH" from `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xC1 (-63dBm)		02 84 0B 00 55 00 00 DA 18 00 C1 45 46 47 48 D7
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully	02 C4 01 00 00 C7	

6. Now module A closes the connection, so both modules will get a disconnect indication.

Info	Module A	Module B
⇒ Request `CMD_DISCONNECT_REQ`: Disconnect	02 07 00 00 05	
⇐ Response `CMD_DISCONNECT_CNF`: Request received, disconnect now	02 47 01 00 00 44	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed	02 87 01 00 16 92	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed		02 87 01 00 F3 97

5 Functional description

The Proteus-III module acts as a slave and can be fully controlled by an external host that implements the command interface. The configuration as well as the operation of the module can be managed by predefined commands that are sent as telegrams over the UART interface of the module.

The Proteus-III can operate in different states. Depending on the active state several commands of the command interface (see chapter 7) are permitted to modify the state, configure the module or transmit data over the radio interface. An overview of the different states and the corresponding allowed commands can be found in Figure 7.

When the Proteus-III is powered up, it starts in `ACTION_IDLE` state. In this state the module advertises (BLE role "peripheral"), such that other devices in range (BLE role "central" or "observer") can detect it and connect to it. If no connection was setup after `RF_AdvertisingTimeout` seconds, the module goes to `ACTION_SLEEP` state which will stop advertising.

The `ACTION_IDLE` state also allows to switch to `ACTION_SCANNING` state, where the module stops advertising and scans for other advertising modules in range (BLE role "central").

When leaving the `ACTION_SCANNING` state with the corresponding command, the module is in `ACTION_IDLE` state and starts advertising again.

The `ACTION_CONNECTED` state can be entered either by getting a connection request from another module (BLE role "peripheral") or by setting up a connection itself (BLE role "central"). In this case it stops advertising and data can be transmitted and received to/from the connected module. This state remains active as long as the module does not disconnect itself (e.g. due to a timeout), no disconnection request from the connected device is received.

When disconnecting, the module goes to `ACTION_IDLE` state and starts advertising again.

ACTION_SCANNING

The module scans for other advertising Proteus modules in range.
Permitted commands:
CMD_RESET_REQ, CMD_GETSTATE_REQ,
CMD_SCANSTOP_REQ, CMD_GET_REQ,
CMD_UARTDISABLE_REQ

ACTION_DTM

The module is in direct test mode.

Permitted commands:
CMD_DTM_REQ,
CMD_RESET_REQ,
CMD_GETSTATE_REQ

CMD_SCANSTART_REQ

CMD_SCANSTOP_REQ

CMD_DTMSTART_REQ

ACTION_IDLE

The module advertises and waits for incoming connection indications. If no connection will be established, it goes to sleep.
The module can establish a connection to another module in range by itself.

Permitted commands:
CMD_RESET_REQ, CMD_FACTORYRESET_REQ,
CMD_GETSTATE_REQ, CMD_CONNECT_REQ,
CMD_SLEEP_REQ, CMD_SETBEACON_REQ,
CMD_SCANSTART_REQ, CMD_GET_REQ,
CMD_GETDEVICES_REQ, CMD_SET_REQ,
CMD_BOOTLOADER_REQ, CMD_UARTDISABLE_REQ,
CMD_DTMSTART_REQ, CMD_PASSKEY_REQ,
CMD_GETBONDS_REQ, CMD_DELETEBONDS_REQ,
CMD_NUMERIC_COMP, CMD_GPIO_LOCAL_CONFIG_REQ,
CMD_GPIO_REMOTE_WRITE_REQ, CMD_GPIO_REMOTE_READ_REQ,
CMD_GPIO_LOCAL_WRITE_REQ, CMD_GPIO_LOCAL_READ_REQ

ACTION_SLEEP

The module is in sleep mode and waits for a wakeup-event.

Permitted commands:
None

Sleep event
CMD_SLEEP_REQ

Wake-up event

Start

Reset

CMD_CONNECT_REQ
CMD_CONNECT_IND

no

Ok ?

yes

CMD_DISCONNECT_REQ
CMD_DISCONNECT_IND

Bootloader event
CMD_BOOTLOADER_REQ

ACTION_CONNECTED

The module is connected.
Now data can be transmitted.

Permitted commands:
CMD_RESET_REQ, CMD_GETSTATE_REQ,
CMD_DATA_REQ, CMD_DISCONNECT_REQ,
CMD_GETDEVICES_REQ, CMD_GET_REQ,
CMD_UARTDISABLE_REQ, CMD_PHYUPDATE_REQ,
CMD_GPIO_REMOTE_WRITE_REQ,
CMD_GPIO_REMOTE_READ_REQ,
CMD_GPIO_LOCAL_WRITE_REQ,
CMD_GPIO_LOCAL_READ_REQ

BOOTLOADER

The module can be updated over the air.

Permitted commands:
None

Figure 7: State overview

5.1 State indication using the LED pins

The pins *LED_1* and *LED_2* of the Proteus-III can be used to determine the module state. The states described in Figure 7 result in the following pin behavior. The pins on the Proteus-III are active high.

State	*LED_1*	*LED_2*
`ACTION_IDLE`	Blinking (On for 200ms, Off for 2800ms)	Off
`ACTION_SCANNING`	Blinking (On for 1000ms, Off for 1000ms)	Off
`ACTION_CONNECTED`	On	Off, On (as soon as the channel was opened successfully, see `CMD_CHANNELOPEN_RSP`)
`ACTION_SLEEP`	Off	Off
`ACTION_DTM`	Off	Off
`BOOTLOADER` waiting for connection	On	Off
`BOOTLOADER` connected, firmware update running	Off	On

Table 9: LED behavior of the Proteus-III

5.2 Sleep mode

Especially for battery-powered devices the `ACTION_SLEEP` mode (system-off mode) supports very low power consumption (<1µA). It can be entered by sending the command `CMD_SLEEP_REQ` to the module. If allowed (due to the current operating state) the module will then send a `CMD_SLEEP_CNF` and then enter the `ACTION_SLEEP` mode.

In `ACTION_SLEEP` mode the UART is disabled, so the module will not receive or transmit any data. To prevent leakage current, the host shall not pull the *URXD* to LOW level (as the module has an internal pull-up resistor enabled on this pin).

To leave the `ACTION_SLEEP` mode and enter `ACTION_IDLE` state again, the module has to be woken up by applying a low signal to the *WAKE_UP* pin for at least 5ms before releasing the signal back to high. The module then restarts completely, so that all volatile settings are set to default. A `CMD_GETSTATE_CNF` will be send when the module is ready for operation.

> Please note that the *WAKE_UP* pin has a second function. If the module is not in `ACTION_SLEEP` mode and the UART was disabled using the `CMD_UARTDISABLE_REQ`, the UART can be re-enabled by applying falling edge, holding the line low for at least 10ms before applying a rising edge and holding it high for at least 10ms. In this case the module answers with a `CMD_UARTENABLE_IND` message.

5.3 Identification of a Proteus-III device on the radio

The Proteus-III can be identified on the radio interface by its `FS_BTMAC`. This `FS_BTMAC` is a Bluetooth®-conform MAC address, which is part of the data package sent during advertising in `ACTION_IDLE` mode. A `FS_BTMAC` has the size of 6 Bytes.

In `ACTION_SCANNING` state a module listens to the data packets of all advertising modules in range and stores their `FS_BTMAC` to an internal data base. With help of a `FS_BTMAC` a connection to the corresponding device can then be established using the `CMD_CONNECT_REQ` command.

To simplify the identification of Proteus-III devices on the RF-interface a short user-defined name (see `RF_DeviceName`) can be given to the module, which is also part of the advertising packet.

> The `FS_BTMAC` consists of the company ID 0x0018DA followed by the `FS_SerialNumber` of the module.

5.4 Connection based data transmission, with or without security

In the BLE standard the transmission of data typically is connection based. A connection between two devices can be secured (with or without key exchange) or unsecured (default setting). In any case, each data packet transmitted is acknowledged on the link layer, such that it is resent as long as a packet is lost. The following lines describe how to run the connection setup and data transmission using the Proteus-III.

If module A is supposed to setup a connection with module B, module A can use the command `CMD_CONNECT_REQ` including the `FS_BTMAC` of module B. If the `FS_BTMAC` of module B is unknown, a radio scan can be run before by module A to discover all available modules in range.

After sending the command `CMD_CONNECT_REQ`, the module answers with a `CMD_CONNECT_CNF` to signal that the request has been understood and the module now tries to establish the connection.

If module B cannot be found on the air within a timeout, module A outputs a `CMD_CONNECT_IND` with "failed" as status. Otherwise, as soon as the physical connection has been set up successfully, module A and B print a `CMD_CONNECT_IND` with the status of the successful connection and *LED_1* turns on.

Next some security and authentication messages will follow, like `CMD_SECURITY_IND`, if security is enabled.

After the physical connection has been setup successfully the modules exchange their services. As soon as this has finished successfully, a `CMD_CHANNELOPEN_RSP` is given out to the UART indicating that the connection is ready for data transmission. Furthermore, *LED_2* turns on.

Now data can be transmitted in both directions using the command `CMD_DATA_REQ`. It is confirmed by a `CMD_DATA_CNF` (data will be processed) and a `CMD_TXCOMPLETE_RSP` (data transmitted successfully).

Each time data has been received a `CMD_DATA_IND` will be output containing the transmitted data.

As soon as one module closes the connection using a `CMD_DISCONNECT_REQ`, both modules

will inform their host by a `CMD_DISCONNECT_IND` message that the connection is no longer open. If one module is no longer within range, the `CMD_DISCONNECT_IND` message is triggered by a timeout.

For an example on setting up an unsecured connection, see chapter `4.4`. See also the Proteus-III application note 1 "advanced user guide" to get detailed information about the connection setup with foreign devices.

5.4.1 Further information for a secure connection setup

The `RF_SecFlags` parameter of the module determines the security mode. If a certain security mode of a Proteus-III peripheral device is set, its security level has to be met by the connecting central device to be able to exchange data. As long as the defined security level is not met by the central device, no access to the peripheral's profiles will be granted.

> When connecting from a Proteus-III to a Proteus-III, you shall not use different security modes.

> To get further information about the secured connection setup, when using a foreign device (i.e. mobile phone with a custom APP), please refer to the Proteus-III application note 1 "advanced user guide".

5.4.1.1 Just works mode

In case of the "Just works" mode, each time a connection is established, a new random key is exchanged in advance to be used for data encryption. Since no authentication will be performed, also devices without input and output capabilities (like keyboard or display) are able to connect to each other.

Example: Secured connection with LE Legacy security method "Just Works" without bonding

1. Power-up the modules and make their UARTs accessible by the host(s) (115200 Baud, 8n1). After the power-up or after reset the following sequence is sent from the module

Info	Module A	Module B
⇐ Response `CMD_GETSTATE_CNF`: Module A started in `ACTION_IDLE` mode.	02 41 02 00 01 01 41	
⇐ Response `CMD_GETSTATE_CNF`: Module B started in `ACTION_IDLE` mode.		02 41 02 00 01 01 41

2. Request the `FS_BTMAC` of both modules.

Info	Module A	Module B
⇒ Request `CMD_GET_REQ` with settings index 4	02 10 01 00 04 17	
⇐ Response `CMD_GET_CNF`: `FS_BTMAC` of module A is 0x55 0x00 0x00 0xDA 0x18 0x00	02 50 07 00 00 55 00 00 DA 18 00 C2	
⇒ Request `CMD_GET_REQ` with settings index 4		02 10 01 00 04 17
⇐ Response `CMD_GET_CNF`: `FS_BTMAC` of module B is 0x11 0x00 0x00 0xDA 0x18 0x00		02 50 07 00 00 11 00 00 DA 18 00 86

3. Configure the parameter `RF_SecFlags` to use "Just Works" pairing method for BT security.

Info	Module A	Module B
⇒Perform `CMD_SET_REQ` with settings index 12 and value 0x02 on module A	02 11 02 00 0C 02 1F	
⇐ Response `CMD_SET_CNF` (Module will restart to adopt the new value)	02 51 01 00 00 52	
⇐ Response `CMD_GETSTATE_CNF`	02 41 02 00 01 01 41	
⇒Perform `CMD_SET_REQ` with settings index 12 and value 0x02 on module B		02 11 02 00 0C 02 1F
⇐ Response `CMD_SET_CNF` (Module will restart to adopt the new value)		02 51 01 00 00 52
⇐ Response `CMD_GETSTATE_CNF`		02 41 02 00 01 01 41

4. Connect module A to module B via Bluetooth®.

Info	Module A	Module B
⇒ Request `CMD_CONNECT_REQ` with `FS_BTMAC` of module B	02 06 06 00 11 00 00 DA 18 00 D1	
⇐ Response `CMD_CONNECT_CNF`: Request understood, try to connect now	02 46 01 00 00 45	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 86 07 00 00 11 00 00 DA 18 00 50	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 86 07 00 00 55 00 00 DA 18 00 14
⇐ Indication `CMD_SECURITY_IND`, status 0x02 (encrypted link, pairing, no bonding), with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 88 07 00 02 11 00 00 DA 18 00 5C	
⇐ Indication `CMD_SECURITY_IND`, status 0x02 (encrypted link, pairing, no bonding), with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 88 07 00 02 55 00 00 DA 18 00 18
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet	02 C6 08 00 00 11 00 00 DA 18 00 F3 EC	
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet		02 C6 08 00 00 55 00 00 DA 18 00 F3 A8

5. Once the connection is active, data can be sent in each direction. Let us send a string "ABCD" from module B to module A.

The RSSI values will be different in your tests.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "ABCD" to module A		02 04 04 00 41 42 43 44 06
⇐ Response `CMD_DATA_CNF`: Request received, send data now		02 44 01 00 00 47
⇐ Indication `CMD_DATA_IND`: Received string "ABCD" from `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xCA (-54dBm)	02 84 0B 00 11 00 00 DA 18 00 CA 41 42 43 44 90	
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully		02 C4 01 00 00 C7

6. Reply with "EFGH" to module B.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "EFGH" to module B	02 04 04 00 45 46 47 48 0E	
⇐ Response `CMD_DATA_CNF`: Request received, send data now	02 44 01 00 00 47	
⇐ Indication `CMD_DATA_IND`: Received string "EFGH" from `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xC1 (-63dBm)		02 84 0B 00 55 00 00 DA 18 00 C1 45 46 47 48 D7
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully	02 C4 01 00 00 C7	

7. Now module A closes the connection, so both modules will get a disconnect indication.

Info	Module A	Module B
⇒ Request `CMD_DISCONNECT_REQ`: Disconnect	02 07 00 00 05	
⇐ Response `CMD_DISCONNECT_CNF`: Request received, disconnect now	02 47 01 00 00 44	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed	02 87 01 00 16 92	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed		02 87 01 00 13 97

8. You may want to perform a `CMD_FACTORYRESET_REQ` to restore default settings.

5.4.1.2 StaticPasskey mode

In case of the "StaticPasskey" mode, a pass key has to be entered at the central side that has to match the pass key of the peripheral. Here the Proteus-III uses a static pass key in the peripheral role that is stored in the parameter `RF_StaticPasskey`. When using this method,

the central device requests its host to enter the correct pass key (see `CMD_PASSKEY_IND`). In this case the pass key of the peripheral has to be entered on central side using the `CMD_PASSKEY_REQ` command. If the entered pass key is correct, the channel will be opened for data transmission. Otherwise, the connection will be rejected.

Example: Secured connection with security method "StaticPasskey"

1. Power-up the modules and make their UARTs accessible by the host(s) (115200 Baud, 8n1). After the power-up or after reset the following sequence is sent from the module

Info	Module A	Module B
⇐ Response `CMD_GETSTATE_CNF`: Module A started in `ACTION_IDLE` mode.	02 41 02 00 01 01 41	
⇐ Response `CMD_GETSTATE_CNF`: Module B started in `ACTION_IDLE` mode.		02 41 02 00 01 01 41

2. Request the `FS_BTMAC` of both modules.

Info	Module A	Module B
⇒ Request `CMD_GET_REQ` with settings index 4	02 10 01 00 04 17	
⇐ Response `CMD_GET_CNF`: FS_BTMAC of module A is 0x55 0x00 0x00 0xDA 0x18 0x00	02 50 07 00 00 55 00 00 DA 18 00 C2	
⇒ Request `CMD_GET_REQ` with settings index 4		02 10 01 00 04 17
⇐ Response `CMD_GET_CNF`: FS_BTMAC of module B is 0x11 0x00 0x00 0xDA 0x18 0x00		02 50 07 00 00 11 00 00 DA 18 00 86

3. Configure the parameter `RF_SecFlags` to use "StaticPasskey" pairing method for BT security.

Info	Module A	Module B
⇒Perform `CMD_SET_REQ` with settings index 12 and value 0x03 on module A	02 11 02 00 0C 03 1E	
⇐ Response `CMD_SET_CNF` (Module will restart to adopt the new value)	02 51 01 00 00 52	
⇐ Response `CMD_GETSTATE_CNF`	02 41 02 00 01 01 41	
⇒Perform `CMD_SET_REQ` with settings index 12 and value 0x03 on module B		02 11 02 00 0C 03 1E
⇐ Response `CMD_SET_CNF` (Module will restart to adopt the new value)		02 51 01 00 00 52
⇐ Response `CMD_GETSTATE_CNF`		02 41 02 00 01 01 41

4. Connect module A to module B via Bluetooth®.

Info	Module A	Module B
⇒ Request `CMD_CONNECT_REQ` with `FS_BTMAC` of module B	02 06 06 00 11 00 00 DA 18 00 D1	
⇐ Response `CMD_CONNECT_CNF`: Request understood, try to connect now	02 46 01 00 00 45	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 86 07 00 00 11 00 00 DA 18 00 50	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 86 07 00 00 55 00 00 DA 18 00 14
⇐ Indication `CMD_PASSKEY_IND` to ask for the pass key	02 8D 07 00 00 11 00 00 DA 18 00 5B	
⇒ Answer with the `CMD_PASSKEY_REQ` and the pass key "123123"	02 0D 06 00 31 32 33 31 32 33 09	
⇐ Response `CMD_PASSKEY_CNF`: Pass key ok	02 4D 01 00 00 4E	
⇐ Indication `CMD_SECURITY_IND`, status 0x02 (encrypted link, pairing, no bonding), with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 88 07 00 02 11 00 00 DA 18 00 5C	
⇐ Indication `CMD_SECURITY_IND`, status 0x02 (encrypted link, pairing, no bonding), with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 88 07 00 02 55 00 00 DA 18 00 18
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet	02 C6 08 00 00 11 00 00 DA 18 00 F3 EC	
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet		02 C6 08 00 00 55 00 00 DA 18 00 F3 A8

5. Once the connection is active, data can be sent in each direction. Let us send a string "ABCD" from module B to module A.

⓵ The RSSI values will be different in your tests.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "ABCD" to module A		02 04 04 00 41 42 43 44 06
⇐ Response `CMD_DATA_CNF`: Request received, send data now		02 44 01 00 00 47
⇐ Indication `CMD_DATA_IND`: Received string "ABCD" from `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xCA (-54dBm)	02 84 0B 00 11 00 00 DA 18 00 CA 41 42 43 44 90	
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully		02 C4 01 00 00 C7

6. Reply with "EFGH" to module B.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "EFGH" to module B	02 04 04 00 45 46 47 48 0E	
⇐ Response `CMD_DATA_CNF`: Request received, send data now	02 44 01 00 00 47	
⇐ Indication `CMD_DATA_IND`: Received string "EFGH" from `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xC1 (-63dBm)		02 84 0B 00 55 00 00 DA 18 00 C1 45 46 47 48 D7
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully	02 C4 01 00 00 C7	

7. Now module A closes the connection, so both modules will get a disconnect indication.

Info	Module A	Module B
⇒ Request `CMD_DISCONNECT_REQ`: Disconnect	02 07 00 00 05	
⇐ Response `CMD_DISCONNECT_CNF`: Request received, disconnect now	02 47 01 00 00 44	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed	02 87 01 00 16 92	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed		02 87 01 00 13 97

8. You may want to perform a `CMD_FACTORYRESET_REQ` to restore default settings.

5.4.1.3 LescPasskey mode

In case of the "LescPasskey" mode, a pass key has to be entered at the central side that has to match the pass key of the peripheral. Here the Proteus-III uses a pass key in the peripheral role that is generated by the LESC method (low energy secure connection) when

a connection is initiated. When using this method, the peripheral device outputs the new generated pass key (see `CMD_DISPLAY_PASSKEY_IND`) when a connection setup has been initiated. At the same time the central device requests its host to enter this pass key (see `CMD_PASSKEY_IND`). In this case the pass key of the peripheral has to be entered on central side using the `CMD_PASSKEY_REQ` command. If the entered pass key is correct, the channel will be opened for data transmission. Otherwise, the connection will be rejected.

Example: Secured connection with security method "LescPasskey"

1. Power-up the modules and make their UARTs accessible by the host(s) (115200 Baud, 8n1). After the power-up or after reset the following sequence is sent from the module

Info	Module A	Module B
⇐ Response `CMD_GETSTATE_CNF`: Module A started in `ACTION_IDLE` mode.	02 41 02 00 01 01 41	
⇐ Response `CMD_GETSTATE_CNF`: Module B started in `ACTION_IDLE` mode.		02 41 02 00 01 01 41

2. Request the `FS_BTMAC` of both modules.

Info	Module A	Module B
⇒ Request `CMD_GET_REQ` with settings index 4	02 10 01 00 04 17	
⇐ Response `CMD_GET_CNF`: FS_BTMAC of module A is 0x55 0x00 0x00 0xDA 0x18 0x00	02 50 07 00 00 55 00 00 DA 18 00 C2	
⇒ Request `CMD_GET_REQ` with settings index 4		02 10 01 00 04 17
⇐ Response `CMD_GET_CNF`: FS_BTMAC of module B is 0x11 0x00 0x00 0xDA 0x18 0x00		02 50 07 00 00 11 00 00 DA 18 00 86

3. Configure the parameter `RF_SecFlags` to use "LescPasskey" pairing method for BT security.

Info	Module A	Module B
⇒Perform `CMD_SET_REQ` with settings index 12 and value 0x05 on module A	02 11 02 00 0C 05 18	
⇐ Response `CMD_SET_CNF` (Module will restart to adopt the new value)	02 51 01 00 00 52	
⇐ Response `CMD_GETSTATE_CNF`	02 41 02 00 01 01 41	
⇒Perform `CMD_SET_REQ` with settings index 12 and value 0x05 on module B		02 11 02 00 0C 05 18
⇐ Response `CMD_SET_CNF` (Module will restart to adopt the new value)		02 51 01 00 00 52
⇐ Response `CMD_GETSTATE_CNF`		02 41 02 00 01 01 41

4. Connect module A to module B via Bluetooth®.

Info	Module A	Module B
⇒ Request `CMD_CONNECT_REQ` with `FS_BTMAC` of module B	02 06 06 00 11 00 00 DA 18 00 D1	
⇐ Response `CMD_CONNECT_CNF`: Request understood, try to connect now	02 46 01 00 00 45	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 86 07 00 00 11 00 00 DA 18 00 50	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 86 07 00 00 55 00 00 DA 18 00 14
⇐ Indication `CMD_DISPLAY_PASSKEY_IND` to display the new generated pass key "429943"		02 A4 0D 00 00 55 00 00 DA 18 00 34 32 39 39 34 33 3D
⇐ Indication `CMD_PASSKEY_IND` to ask for the pass key	02 8D 07 00 00 11 00 00 DA 18 00 5B	
⇒ Answer with the `CMD_PASSKEY_REQ` and the pass key "429943"	02 0D 06 00 34 32 39 39 34 33 08	
⇐ Response `CMD_PASSKEY_CNF`: Pass key ok	02 4D 01 00 00 4E	
⇐ Indication `CMD_SECURITY_IND`, status 0x02 (encrypted link, pairing, no bonding), with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 88 07 00 02 11 00 00 DA 18 00 5C	
⇐ Indication `CMD_SECURITY_IND`, status 0x02 (encrypted link, pairing, no bonding), with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 88 07 00 02 55 00 00 DA 18 00 18
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet	02 C6 08 00 00 11 00 00 DA 18 00 F3 EC	
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet		02 C6 08 00 00 55 00 00 DA 18 00 F3 A8

5. Once the connection is active, data can be sent in each direction. Let us send a string "ABCD" from module B to module A.

The RSSI values will be different in your tests.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "ABCD" to module A		02 04 04 00 41 42 43 44 06
⇐ Response `CMD_DATA_CNF`: Request received, send data now		02 44 01 00 00 47
⇐ Indication `CMD_DATA_IND`: Received string "ABCD" from `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xCA (-54dBm)	02 84 0B 00 11 00 00 DA 18 00 CA 41 42 43 44 90	
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully		02 C4 01 00 00 C7

6. Reply with "EFGH" to module B.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "EFGH" to module B	02 04 04 00 45 46 47 48 0E	
⇐ Response `CMD_DATA_CNF`: Request received, send data now	02 44 01 00 00 47	
⇐ Indication `CMD_DATA_IND`: Received string "EFGH" from `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xC1 (-63dBm)		02 84 0B 00 55 00 00 DA 18 00 C1 45 46 47 48 D7
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully	02 C4 01 00 00 C7	

7. Now module A closes the connection, so both modules will get a disconnect indication.

Info	Module A	Module B
⇒ Request `CMD_DISCONNECT_REQ`: Disconnect	02 07 00 00 05	
⇐ Response `CMD_DISCONNECT_CNF`: Request received, disconnect now	02 47 01 00 00 44	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed	02 87 01 00 16 92	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed		02 87 01 00 13 97

8. You may want to perform a `CMD_FACTORYRESET_REQ` to restore default settings.

5.4.1.4 LescNumComp mode

In case of the "LescNumComp" mode, a pass key is displayed on peripheral and central side. Both, the central and peripheral device must confirm that both keys are equal. Here the Proteus-III uses a pass key that is generated by the LESC method (low energy secure

connection) when a connection is initiated. When using this method, the peripheral and central device output the new generated pass key (see `CMD_DISPLAY_PASSKEY_IND`) when a connection setup has been initiated. Both, the central and peripheral device request their hosts to confirm that both keys coincide (see `CMD_NUMERIC_COMP_REQ`). If both devices confirmed the key, the channel will be opened for data transmission. Otherwise, the connection will be rejected.

Example: Secured connection with security method "LescNumComp"

1. Power-up the modules and make their UARTs accessible by the host(s) (115200 Baud, 8n1). After the power-up or after reset the following sequence is sent from the module

Info	Module A	Module B
⇐ Response `CMD_GETSTATE_CNF`: Module A started in `ACTION_IDLE` mode.	02 41 02 00 01 01 41	
⇐ Response `CMD_GETSTATE_CNF`: Module B started in `ACTION_IDLE` mode.		02 41 02 00 01 01 41

2. Request the `FS_BTMAC` of both modules.

Info	Module A	Module B
⇒ Request `CMD_GET_REQ` with settings index 4	02 10 01 00 04 17	
⇐ Response `CMD_GET_CNF`: FS_BTMAC of module A is 0x55 0x00 0x00 0xDA 0x18 0x00	02 50 07 00 00 55 00 00 DA 18 00 C2	
⇒ Request `CMD_GET_REQ` with settings index 4		02 10 01 00 04 17
⇐ Response `CMD_GET_CNF`: FS_BTMAC of module B is 0x11 0x00 0x00 0xDA 0x18 0x00		02 50 07 00 00 11 00 00 DA 18 00 86

3. Configure the parameter `RF_SecFlags` to use "LescPasskey" pairing method for BT security.

Info	Module A	Module B
⇒Perform `CMD_SET_REQ` with settings index 12 and value 0x04 on module A	02 11 02 00 0C 04 19	
⇐ Response `CMD_SET_CNF` (Module will restart to adopt the new value)	02 51 01 00 00 52	
⇐ Response `CMD_GETSTATE_CNF`	02 41 02 00 01 01 41	
⇒Perform `CMD_SET_REQ` with settings index 12 and value 0x04 on module B		02 11 02 00 0C 04 19
⇐ Response `CMD_SET_CNF` (Module will restart to adopt the new value)		02 51 01 00 00 52
⇐ Response `CMD_GETSTATE_CNF`		02 41 02 00 01 01 41

4. Connect module A to module B via Bluetooth®.

Info	Module A	Module B
⇒ Request `CMD_CONNECT_REQ` with `FS_BTMAC` of module B	02 06 06 00 11 00 00 DA 18 00 D1	
⇐ Response `CMD_CONNECT_CNF`: Request understood, try to connect now	02 46 01 00 00 45	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 86 07 00 00 11 00 00 DA 18 00 50	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 86 07 00 00 55 00 00 DA 18 00 14
⇐ Indication `CMD_DISPLAY_PASSKEY_IND` to display the new generated pass key "234939"	02 A4 0D 00 01 11 00 00 DA 18 00 32 33 34 39 33 39 7F	
⇐ Indication `CMD_DISPLAY_PASSKEY_IND` to display the new generated pass key "234939"		02 A4 0D 00 01 55 00 00 DA 18 00 32 33 34 39 33 39 3B
⇒ Answer with the `CMD_NUMERIC_COMP_REQ` to confirm that both keys are equal	02 24 01 00 00 27	
⇐ Response `CMD_NUMERIC_COMP_CNF`: Request understood, going on with the connection setup process	02 64 01 00 00 67	
⇒ Answer with the `CMD_NUMERIC_COMP_REQ` to confirm that both keys are equal		02 24 01 00 00 27
⇐ Response `CMD_NUMERIC_COMP_CNF`: Request understood, going on with the connection setup process		02 64 01 00 00 67
⇐ Indication `CMD_SECURITY_IND`, status 0x02 (encrypted link, pairing, no bonding), with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 88 07 00 02 11 00 00 DA 18 00 5C	
⇐ Indication `CMD_SECURITY_IND`, status 0x02 (encrypted link, pairing, no bonding), with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 88 07 00 02 55 00 00 DA 18 00 18
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet	02 C6 08 00 00 11 00 00 DA 18 00 F3 EC	
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet		02 C6 08 00 00 55 00 00 DA 18 00 F3 A8

5. Once the connection is active, data can be sent in each direction. Let us send a string "ABCD" from module B to module A.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "ABCD" to module A		02 04 04 00 41 42 43 44 06
⇐ Response `CMD_DATA_CNF`: Request received, send data now		02 44 01 00 00 47
⇐ Indication `CMD_DATA_IND`: Received string "ABCD" from `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xCA (-54dBm)	02 84 0B 00 11 00 00 DA 18 00 CA 41 42 43 44 90	
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully		02 C4 01 00 00 C7

6. Reply with "EFGH" to module B.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "EFGH" to module B	02 04 04 00 45 46 47 48 0E	
⇐ Response `CMD_DATA_CNF`: Request received, send data now	02 44 01 00 00 47	
⇐ Indication `CMD_DATA_IND`: Received string "EFGH" from `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xC1 (-63dBm)		02 84 0B 00 55 00 00 DA 18 00 C1 45 46 47 48 D7
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully	02 C4 01 00 00 C7	

7. Now module A closes the connection, so both modules will get a disconnect indication.

Info	Module A	Module B
⇒ Request `CMD_DISCONNECT_REQ`: Disconnect	02 07 00 00 05	
⇐ Response `CMD_DISCONNECT_CNF`: Request received, disconnect now	02 47 01 00 00 44	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed	02 87 01 00 16 92	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed		02 87 01 00 13 97

8. You may want to perform a `CMD_FACTORYRESET_REQ` to restore default settings.

5.4.1.5 Bonding

The `SECFLAGS_BONDING_ENABLE` flag in the `RF_SecFlags` user setting allows enabling the bonding feature. This feature stores the keys that are exchanged during the pairing phase in a connection setup. With this, subsequent connections to bonded devices can be established without renegotiation.

The commands `CMD_GETBONDS_REQ` and `CMD_DELETEBONDS_REQ` allow to display and remove certain or all entries of the list of bonded devices.

Example: Secured connection with LE Legacy security method "Just Works" using bonding

1. Power-up the modules and make their UARTs accessible by the host(s) (115200 Baud, 8n1). After the power-up or after reset the following sequence is sent from the module

Info	Module A	Module B
⇐ Response `CMD_GETSTATE_CNF`: Module A started in `ACTION_IDLE` mode.	02 41 02 00 01 01 41	
⇐ Response `CMD_GETSTATE_CNF`: Module B started in `ACTION_IDLE` mode.		02 41 02 00 01 01 41

2. Request the `FS_BTMAC` of both modules.

Info	Module A	Module B
⇒ Request `CMD_GET_REQ` with settings index 4	02 10 01 00 04 17	
⇐ Response `CMD_GET_CNF`: FS_BTMAC of module A is 0x55 0x00 0x00 0xDA 0x18 0x00	02 50 07 00 00 55 00 00 DA 18 00 C2	
⇒ Request `CMD_GET_REQ` with settings index 4		02 10 01 00 04 17
⇐ Response `CMD_GET_CNF`: FS_BTMAC of module B is 0x11 0x00 0x00 0xDA 0x18 0x00		02 50 07 00 00 11 00 00 DA 18 00 86

3. Configure the parameter `RF_SecFlags` to use "Just Works with bonding" pairing method for BT security.

Info	Module A	Module B
⟹Perform `CMD_SET_REQ` with settings index 12 and value 0x0A (Just works with `SECFLAGS_BONDING_ENABLE` flag set) on module A	02 11 02 00 0C 0A 17	
⟸ Response `CMD_SET_CNF` (Module will restart to adopt the new value)	02 51 01 00 00 52	
⟸ Response `CMD_GETSTATE_CNF`	02 41 02 00 01 01 41	
⟹Perform `CMD_SET_REQ` with settings index 12 and value 0x0A (Just works with `SECFLAGS_BONDING_ENABLE` flag set) on module B		02 11 02 00 0C 0A 17
⟸ Response `CMD_SET_CNF` (Module will restart to adopt the new value)		02 51 01 00 00 52
⟸ Response `CMD_GETSTATE_CNF`		02 41 02 00 01 01 41

4. Connect module A to module B via Bluetooth®.

Info	Module A	Module B
⟹ Request `CMD_CONNECT_REQ` with `FS_BTMAC` of module B	02 06 06 00 11 00 00 DA 18 00 D1	
⟸ Response `CMD_CONNECT_CNF`: Request understood, try to connect now	02 46 01 00 00 45	
⟸ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 86 07 00 00 11 00 00 DA 18 00 50	
⟸ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 86 07 00 00 55 00 00 DA 18 00 14
⟸ Indication `CMD_SECURITY_IND`, status 0x01 (encrypted link, bonding established), with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 88 07 00 01 11 00 00 DA 18 00 5F	
⟸ Indication `CMD_SECURITY_IND`, status 0x01 (encrypted link, bonding established), with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 88 07 00 01 55 00 00 DA 18 00 1B
⟸ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet	02 C6 08 00 00 11 00 00 DA 18 00 F3 EC	
⟸ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet		02 C6 08 00 00 55 00 00 DA 18 00 F3 A8

5. Now module A closes the connection, so both modules will get a disconnect indication.

Info	Module A	Module B
⇒ Request `CMD_DISCONNECT_REQ`: Disconnect	02 07 00 00 05	
⇐ Response `CMD_DISCONNECT_CNF`: Request received, disconnect now	02 47 01 00 00 44	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed	02 87 01 00 16 92	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed		02 87 01 00 13 97

6. Connect module A to module B a second time. Now, since both devices have been bonded before, the exchanged keys are reused.

Info	Module A	Module B
⇒ Request `CMD_CONNECT_REQ` with `FS_BTMAC` of module B	02 06 06 00 11 00 00 DA 18 00 D1	
⇐ Response `CMD_CONNECT_CNF`: Request understood, try to connect now	02 46 01 00 00 45	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 86 07 00 00 11 00 00 DA 18 00 50	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 86 07 00 00 55 00 00 DA 18 00 14
⇐ Indication `CMD_SECURITY_IND`, status 0x00 (encrypted link to bonded device), with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 88 07 00 00 11 00 00 DA 18 00 5E	
⇐ Indication `CMD_SECURITY_IND`, status 0x00 (encrypted link to bonded device), with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 88 07 00 00 55 00 00 DA 18 00 1A
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet	02 C6 08 00 00 11 00 00 DA 18 00 F3 EC	
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet		02 C6 08 00 00 55 00 00 DA 18 00 F3 A8

7. You may want to perform a `CMD_FACTORYRESET_REQ` to restore default settings.

5.5 Unidirectional connectionless data transmission using Beacons

Besides the connection-based type of data transmission described in the previous section, there exists a second method that uses so called Beacons. In this case, a limited amount of user data can be placed in the BLE scan response packet, which is broadcasted frequently without acknowledgement and without security.

If a Proteus-III is supposed to broadcast some data, the command `CMD_SETBEACON_REQ` can be used to place user data in the scan response packet.

If a second Proteus-III, which has its Beacon-function (see `RF_BeaconFlags`) enabled, is in the operating state `ACTION_SCANNING`, then the scan response packet is requested as soon as an advertising packet from the beacon module has been detected. Filtering the beacon messages can be enabled or disabled using `RF_BeaconFlags`.

After the reception of the scan response packet the included user data is interpreted and given out to the UART using a `CMD_BEACON_IND` message.

To set the module into `ACTION_SCANNING` mode the command `CMD_SCANSTART_REQ` has to be used. Enable the Beacon-function before by setting the corresponding value in the `RF_BeaconFlags` parameter.

> This method is very suitable for sensor networks, which frequently send their data to data collectors. Especially when using a slow `RF_ScanTiming` mode, data can be transmitted in a more energy efficient way.

> Please check the settings `RF_AdvertisingTimeout` and the advertising interval in `RF_ScanTiming` to configure the frequency and interval of transmissions which will have an influence on the current consumption of the module.

Info	Module A	Module B
⇐ Reset both modules using /RESET pin, `CMD_GETSTATE_CNF`	02 41 02 00 01 01 41	02 41 02 00 01 01 41
⇒ Configure `RF_BeaconFlags` using `CMD_SET_REQ` to "beacon rx enabled, no filter"		02 11 02 00 0E 01 1E
⇐ `CMD_SET_CNF` from module B		02 51 01 00 00 52
⇐ Module B reset such that the change in the user setting takes effect (`CMD_GETSTATE_CNF`)		02 41 02 00 01 01 41
⇒ Activate scanning on module B		02 09 00 00 0B
⇐ Response `CMD_SCANSTART_CNF`		02 49 01 00 00 4A
⇒ `CMD_SETBEACON_REQ`, content "Hallo"	02 0C 05 00 48 61 6C 6C 6F 4D	
⇐ `CMD_SETBEACON_CNF`	02 4C 01 00 00 4F	
⇐ receiving multiple `CMD_BEACON_IND`		02 8C 0C 00 02 00 00 DA 18 00 B5 48 61 6C 6C 6F B1 02 8C 0C 00 02 00 00 DA 18 00 B1 48 61 6C 6C 6F B5
⋮	⋮	⋮
⇒ Deactivate scanning on module B, `CMD_SCANSTOP_REQ`		02 0A 00 00 08
⇐ Response `CMD_SCANSTOP_CNF`		02 4A 01 00 00 49
⇒ Reset module A (disable sending beacons), `CMD_RESET_REQ`	02 00 00 00 02	
⇐ Response `CMD_RESET_CNF`	02 40 01 00 00 43	
⇐ Response `CMD_GETSTATE_CNF`	02 41 02 00 01 01 41	

5.6 Energy-efficient distance estimation solutions

The Proteus-III advertising packet contains the TX power value of the transmitting device. This value in combination with the RSSI value of the received advertising packet can be used to estimate the distance between the modules. Using a suitable triangulation algorithm and multiple receivers or transmitters, a position can be approximated.

The advertising packets can be received by performing a passive scan that will not request the scan response. Thus only one frame, instead of three frames, is transmitted per advertising interval.

Besides the `FS_BTMAC` of the sending module, the RSSI value and the TX power is output in format of a `CMD_RSSI_IND` message via UART when an advertising packet of another Proteus-III has been received.

To enable this function, the corresponding value in the `RF_BeaconFlags` has to be set.

5.7 Configure the module for low power consumption

Depending on the application environment of the Proteus-III, the goal is to find the optimal trade-off between the module's performance and its power consumption. Therefore, the main settings and operation modes that affect the current consumption are listed below:

- `CMD_SLEEP_REQ`: This command puts the module into `ACTION_SLEEP` mode, where it consumes the lowest current (<1µA). In this case, both the UART and the BLE interface are shut down.

- `CMD_UARTDISABLE_REQ`: This command disables the UART interface. It is enabled again as soon as the module is reset/woken or when the module outputs a message e.g. when a connection request has been received or the *WAKE_UP* pin of the module was used.

- `RF_TXPower`: This setting can be used to configure the output power of the module. Reducing the output power saves energy.

- `RF_ScanTiming` and `RF_ScanFactor`: These settings define the timing behavior of the module, when advertising or scanning. The less often the module sends advertising packets or scans, the less current is consumed.

- `RF_ConnectionTiming`: This setting defines the timing behavior of the module during connection setup and an established connection. The less often the connected modules communicate with each other, the less current is consumed.

- The on-board nRF52 SoC is running in debug mode. This will not occur if the pins are connected as described in this manual.

- The 2 MBit radio modes transmits data packets faster. Using it reduces the current consumption slightly.

> ⚠ For optimum energy efficiency a user and application specific firmware may be required.

5.8 Start the direct test mode (DTM)

The direct test mode (DTM) enables the test functions described in Bluetooth® Specification. The purpose of DTM is to test the operation of the radio at the physical level, such as:

- transmission power and receiver sensitivity

- frequency offset and drift

- modulation characteristics

- packet error rate

- inter modulation performance

Conformance tests of the nRF52 with the DTM application are carried out by dedicated test equipment. To get access to the test functions the `CMD_DTMSTART_REQ` shall be used first. This command restarts the module in direct test mode. A `CMD_GETSTATE_CNF` message confirms that the DTM has been started successfully. Now the `CMD_DTM_REQ` can be used to start and stop the test functions. After a test has been started, it has to be stopped before a next test can be run.

Example: Transmission test on channel 0 with Bit pattern 0x0F
The goal of this example is to show how the DTM, and in specific the transmission/reception test, can be run. Here fore we need to connect two modules, start the transmission test on one module and start the reception test on the second module. In this section, all packet data from or to the modules is given in **hexadecimal notation**.
All steps are described in the following:

- First, restart the modules in DTM mode.

Info	Module A	Module B
⇒ Request `CMD_DTMSTART_REQ` to enable the DTM on module A	02 1D 00 00 1F	
⇐ Response `CMD_DTMSTART_CNF`: Request understood, try to start DTM now	02 5D 01 00 00 5E	
⇐ Indication `CMD_GETSTATE_CNF`: Restarted module with DTM enabled	02 41 02 00 10 05 54	
⇒ Request `CMD_DTMSTART_REQ` to enable the DTM on module B		02 1D 00 00 1F
⇐ Response `CMD_DTMSTART_CNF`: Request understood, try to start DTM now		02 5D 01 00 00 5E
⇐ Indication `CMD_GETSTATE_CNF`: Restarted module with DTM enabled		02 41 02 00 10 05 54

- Now both modules are ready for the DTM. Start the transmission test first.

Info	Module A	Module B
⇒ Request `CMD_DTM_REQ` to start the transmission test on module A with channel 0 and Bit pattern 16 times 0x0F	02 1E 04 00 02 00 10 01 0B	
⇐ Response `CMD_DTM_CNF`: Started test successfully	02 5E 03 00 00 00 00 5F	

- Start the reception test.

Info	Module A	Module B
⇒ Request `CMD_DTM_REQ` to start the reception test on module B with channel 0		02 1E 04 00 01 00 00 00 19
⇐ Response `CMD_DTM_CNF`: Started test successfully		02 5E 03 00 00 00 00 5F

- Stop both tests again.

Info	Module A	Module B
⇒ Request `CMD_DTM_REQ` to stop the transmission test	02 1E 04 00 03 00 00 01 1A	
⇐ Response `CMD_DTM_CNF`: Stopped test successfully	02 5E 03 00 00 80 00 DF	
⇒ Request `CMD_DTM_REQ` to stop the reception test		02 1E 04 00 03 00 00 01 1A
⇐ Response `CMD_DTM_CNF`: Stopped test successfully, received 0x14FE (5374_{dec}) packets		02 5E 03 00 00 94 FE 35

During the time the reception and transmission tests were running 5374 data packets have been received by module B, which were transmitted by module A.

5.9 Using the 2 MBit and LE Coded phy

Bluetooth® 5 allows to transmit data with 2 MBit data rate as well as in LE Coded mode. The LE Coded mode is the so called "Long range mode" that has been invented with Bluetooth® 5. It uses the Direct Sequence Spread Spectrum (DSSS) technique that spreads the signal and thus generates redundant informations. On the receiver side, it uses the Forward Error Correction (FEC) technique to use the redundancy to correct a received perturbed signal. The combination of both, the DSSS and FEC, enable higher ranges in data transmission.

To be backward compatible to Bluetooth® 4.x devices, Bluetooth® connections must still be setup using the 1 MBit phy. As soon as a connection has been setup, the connection can be updated to the 2 MBit or LE Coded mode. To switch the phy after the connection has been setup the Proteus-III offers the command `CMD_PHYUPDATE_REQ`. As response to this request a `CMD_PHYUPDATE_IND` is returned from the Proteus-III, that gives feedback if the connection was switched to the new phy, or if the connection partner rejected the request.

> Please note that the 2 MBit and LE Coded phy is an optional feature of Bluetooth® 5 devices and therefore must not be supported.

5.10 Connection setup using LE Coded phy

Due to backward compatibility reasons the Bluetooth® Low Energy standard expects to setup a Bluetooth® connection in the 1 MBit legacy mode and then to update the connection to long range mode, if requested. Thus, at connection setup time the distance between the two BLE devices must be within the standard range.

To avoid this situation, the Proteus-III allows to setup a connection directly in long range mode. To enable this feature, set the corresponding bit in the user setting `CFG_Flags`.

As soon as this feature is enabled, the Proteus-III sends only advertising packets in long range mode. Furthermore, when scanning, only advertising packets in long range mode are received. Thus only devices using this special mode can be found on the radio.

> The Proteus-III advertises and scans in long range mode and thus is radio incompatible to BLE devices acting in legacy 1 Mbit mode. Using this feature, all BLE enabled devices, like smart phones, do not find the Proteus-III on radio as they are scanning in legacy 1 Mbit mode, by default.

Example: Configure the device for long range connection setup

The goal of this example is to demonstrate how to configure two Proteus-III radio module to be able to setup connections in long range mode. Further, a connection setup and data transmission is shown.

1. Power-up the modules and make their UARTs accessible by the host(s) (115200 Baud, 8n1). After the power-up or after reset the following sequence is sent from the module

Info	Module A	Module B
⇐ Response `CMD_GETSTATE_CNF`: Module A started in `ACTION_IDLE` mode.	02 41 02 00 01 01 41	
⇐ Response `CMD_GETSTATE_CNF`: Module B started in `ACTION_IDLE` mode.		02 41 02 00 01 01 41

2. Request the `FS_BTMAC` of both modules.

Info	Module A	Module B
⇒ Request `CMD_GET_REQ` with settings index 4	02 10 01 00 04 17	
⇐ Response `CMD_GET_CNF`: FS_BTMAC of module A is 0x55 0x00 0x00 0xDA 0x18 0x00	02 50 07 00 00 55 00 00 DA 18 00 C2	
⇒ Request `CMD_GET_REQ` with settings index 4		02 10 01 00 04 17
⇐ Response `CMD_GET_CNF`: FS_BTMAC of module B is 0x11 0x00 0x00 0xDA 0x18 0x00		02 50 07 00 00 11 00 00 DA 18 00 86

3. Configure the parameter `CFG_Flags` to use "Long range connection mode".

Info	Module A	Module B
⇒Perform `CMD_SET_REQ` with settings index 28 and value 0x02 (Long range connection mode) on module A	02 11 03 00 1C 02 00 0E	
⇐ Response `CMD_SET_CNF` (Module will restart to adopt the new value)	02 51 01 00 00 52	
⇐ Response `CMD_GETSTATE_CNF`	02 41 02 00 01 01 41	
⇒Perform `CMD_SET_REQ` with settings index 28 and value 0x02 (Long range connection mode) on module B		02 11 03 00 1C 02 00 0E
⇐ Response `CMD_SET_CNF` (Module will restart to adopt the new value)		02 51 01 00 00 52
⇐ Response `CMD_GETSTATE_CNF`		02 41 02 00 01 01 41

4. Connect module A to module B via Bluetooth®.

Info	Module A	Module B
⇒ Request `CMD_CONNECT_REQ` with `FS_BTMAC` of module B	02 06 06 00 11 00 00 DA 18 00 D1	
⇐ Response `CMD_CONNECT_CNF`: Request understood, try to connect now	02 46 01 00 00 45	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 86 07 00 00 11 00 00 DA 18 00 50	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 86 07 00 00 55 00 00 DA 18 00 14
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet	02 C6 08 00 00 11 00 00 DA 18 00 F3 EC	
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0xF3 (243 Bytes) per packet		02 C6 08 00 00 55 00 00 DA 18 00 F3 A8

5. Once the connection is active, data can be sent in each direction. Let us send a string "ABCD" from module B to module A.

> The RSSI values will be different in your tests.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "ABCD" to module A		02 04 04 00 41 42 43 44 06
⇐ Response `CMD_DATA_CNF`: Request received, send data now		02 44 01 00 00 47
⇐ Indication `CMD_DATA_IND`: Received string "ABCD" from `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xCA (-54dBm)	02 84 0B 00 11 00 00 DA 18 00 CA 41 42 43 44 90	
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully		02 C4 01 00 00 C7

6. Reply with "EFGH" to module B.

Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "EFGH" to module B	02 04 04 00 45 46 47 48 0E	
⇐ Response `CMD_DATA_CNF`: Request received, send data now	02 44 01 00 00 47	
⇐ Indication `CMD_DATA_IND`: Received string "EFGH" from `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xC1 (-63dBm)		02 84 0B 00 55 00 00 DA 18 00 C1 45 46 47 48 D7
⇐ Response `CMD_TXCOMPLETE_RSP`: Data transmitted successfully	02 C4 01 00 00 C7	

7. Now module A closes the connection, so both modules will get a disconnect indication.

Info	Module A	Module B
⇒ Request `CMD_DISCONNECT_REQ`: Disconnect	02 07 00 00 05	
⇐ Response `CMD_DISCONNECT_CNF`: Request received, disconnect now	02 47 01 00 00 44	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed	02 87 01 00 16 92	
⇐ Indication `CMD_DISCONNECT_IND`: Connection closed		02 87 01 00 13 97

8. You may want to perform a `CMD_FACTORYRESET_REQ` to restore default settings.

6 Host connection

6.1 Serial interface: UART

The configuration in factory state of the UART is 115200 Baud without flow control and with data format of 8 data Bits, no parity and 1 stop Bit ("8n1"). The baud rate and flow control of the UART can be configured by means of the UserSetting `UART_ConfigIndex`. The data format is fixed to 8n1.

The output of characters on the serial interface runs with secondary priority. For this reason, short interruptions may occur between the outputs of individual successive Bytes. The host must not implement too strict timeouts between two Bytes to be able to receive packets that have interruptions in between.

6.1.1 Reset behaviour

When holding the module's *RESET* pin LOW, the radio chip states are undefined. In this case the modules *UTXD* pin may be pulled LOW by the radio module, such that the connected host controller's UART may detect a 0x00-byte with frame error.

To guarantee a clean UART communication, the host controller may not accept bytes with frame errors and flush its RX buffer, after pulling the module's *RESET* pin LOW.

7 The command interface

The module acts as a slave and can be fully controlled by an external host. The configuration as well as the operation of the module can be managed by predefined commands that are sent as telegrams over the UART interface of the module.
The commands of the command interface can be divided into 3 groups:

- Requests: The host requests the module to trigger any action, e.g. in case of the request `CMD_RESET_REQ` the host asks the module to perform a reset.

- Confirmations: On each request, the module answers with a confirmation message to give a feedback on the requested operation status. In case of a `CMD_RESET_REQ`, the module answers with a `CMD_RESET_CNF` to tell the host whether the reset will be performed or not.

- Indications and Responses: The module indicates spontaneously when a special event has occurred. The `CMD_CONNECT_IND` indicates for example that a connection has been established.

Start signal	Command	Length	Payload	CS
0x02	1 Byte	2 Byte, LSB first	Length Bytes	1 Byte

Start signal: 0x02 (1 Byte)

Command: One of the predefined commands (1 Byte).

Length: Specifies the length of the data that follows. Length is a 16 Bit field with LSB first.

Payload: Variable number (defined by the length field) of data or parameters.

Checksum: Byte wise XOR combination of all preceding Bytes including the start signal, i.e. 0x02 ˆ Command ˆ Length ˆ Payload = CS

> If the transmission of the UART command has not finished within the packet transmission duration (depending on the currently selected UART Baud rate + 5ms after having received the start signal), the module will discard the received Bytes and wait for a new command. This means that the delay between 2 successive Bytes in a frame must be kept as low as possible.

> Please note that the different commands are only valid in specific module states (see Figure 7). If a command is not permitted in the current state, the command confirmation returns "Operation not permitted" as a response.

7.1 Scan for other modules in range

7.1.1 CMD_SCANSTART_REQ

This command starts the scan operation to find other Proteus-III in range. All found devices that fit the Proteus-III specification (i.e. devices that support AMBER SPP service UUID) are saved in an internal data base. Before outputting the data base content using the command `CMD_GETDEVICES_REQ`, the scan has to be stopped using `CMD_SCANSTOP_REQ`.
Format:

Start signal	Command	Length	CS
0x02	0x09	0x00 0x00	0x0B

Response (`CMD_SCANSTART_CNF`):

Start signal	Command	0x40	Length	Status	CS
0x02	0x49	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, will start scan now

0x01: Operation failed

0xFF: Operation not permitted

7.1.2 CMD_SCANSTOP_REQ

This command stops the scan operation that was started using `CMD_SCANSTART_REQ`. It stores the detected Proteus-III `FS_BTMAC` addresses in an internal database, which can be output using the `CMD_GETDEVICES_REQ`.
Format:

Start signal	Command	Length	CS
0x02	0x0A	0x00 0x00	0x08

Response (`CMD_SCANSTOP_CNF`):

Start signal	Command	0x40	Length	Status	CS
0x02	0x4A	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, will stop scan now

0x01: Operation failed

0xFF: Operation not permitted

7.1.3 CMD_GETDEVICES_REQ

This command returns the information about the devices found during the last scan operation. `#Devices` determines the number of devices that have been detected. The corresponding information will be output one after the other in the field behind `#Devices` in the `CMD_GETDEVICES_CNF` response. The RSSI and TXPower values are transmitted in the two's complement notation.

Format:

Start signal	Command	Length	CS
0x02	0x0B	0x00 0x00	0x09

Response (`CMD_GETDEVICES_CNF`):

Start signal	Command | 0x40	Length	Status	#Devices	Payload	CS
0x02	0x4B	2 Bytes	1 Byte	1 Byte	(Length - 2) Bytes	1 Byte

The Payload sequentially lists the data of the detected `#Devices` devices. It consists of `#Devices` times the following telegram (see example below).

BTMAC	RSSI	TXPower	Device name length	Device name
6 Bytes	1 Byte	1 Byte	1 Byte	Device name length Bytes

Status:

0x00: Request received

0x01: Operation failed

0xFF: Operation not permitted

If there are too many devices found to be output, the response of the `CMD_GETDEVICES_REQ` is split into several `CMD_GETDEVICES_CNF` messages.

The detected device name is the content of the device name field of the received advertising packet. Thus, in case of the "Complete Local Name" is too long to fit into the device name field of the advertising packet, this could be the "Shortened Local Name" of the device.

If RSSI = 0x80, there is no value available.

> If TXPower = 0x80, there is no value available.

> If Device name length = 0, then there is no device name available.

7.1.3.1 Example 1

Request for the `FS_BTMAC` of the devices found during the last scan.

Start signal	Command	Length	CS
0x02	0x0B	0x00 0x00	0x09

Response:

Start signal	Command | 0x40	Length	Status	#Devices	Payload	CS
0x02	0x4B	0x1E 0x00	0x00	0x02	0x11 0x00 0x00 0xDA 0x18 0x00 0xE2 0x04 0x05 0x4D 0x4F 0x44 0x20 0x31 0x55 0x00 0x00 0xDA 0x18 0x00 0xE5 0x00 0x05 0x4D 0x4F 0x44 0x20 0x32	0x11

During the last scan two devices have been detected:

- Device 1 with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00, RSSI value of 0xE2 (-30 dBm), TXPower of 0x04 (=+4 dBm) and device name of length 5 with the value of 0x4D4F442031 ("MOD 1").

- Device 2 with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 and RSSI value of 0xE5 (-27 dBm), TXPower of 0x00 (0 dBm) and device name 0x4D4F442032 ("MOD 2") of length 5.

7.1.4 CMD_RSSI_IND

This telegram indicates the reception of an advertising packet sent by another Proteus-III module. It can be used to realize a position sensing application. This data can only be received, when the module is in `ACTION_SCANNING` mode (passive scan is sufficient) and the corresponding value in the `RF_BeaconFlags` is set.
Besides the `FS_BTMAC`, the RSSI value of the advertising packet and the transmission power of the sending device are output. Both, the RSSI value and the TX power are in two's

complement notation.

The accuracy is ±2dB when inside the RSSI range of -90 to -20 dBm.

The value of the parameter TX power is read from the content of the received advertise packet.

Format:

Start signal	Command	Length	BTMAC	RSSI	TX Power	CS
0x02	0x8B	2 Bytes	6 Byte	1 Byte	1 Byte	1 Byte

7.1.5 CMD_BEACON_RSP

This telegram indicates the reception of an advertising packet. This data can only be received, when the module is in `ACTION_SCANNING` mode and the corresponding value in the `RF_BeaconFlags` is set. If active scanning is enabled by setting the `RF_ScanFlags`, received scan response packets are output in addition.

Besides the `FS_BTMAC`, the RSSI value of the advertising packet and the raw data is output. The accuracy is ±2dB when inside the RSSI range of -90 to -20 dBm.

> The format of the raw advertising data is described in the Bluetooth® specification version 5.0 | Vol 3, Part C, section "Advertising and scan response data format".

Format:

Start signal	Command	Length	BTMAC	RSSI	Raw advertising data	CS
0x02	0xCC	2 Bytes	6 Byte	1 Byte	(Length - 7) Bytes	1 Byte

7.2 Setup connections

7.2.1 CMD_CONNECT_REQ

This command tries to setup a connection to the Proteus-III, which is identified by the `FS_BTMAC` used in the command. After the module prints a `CMD_CONNECT_CNF` to confirm that the request was received, the indication message `CMD_CONNECT_IND` follows which determines whether the connection request was accepted by the other device.
In case of enabled security features (see the setting `RF_SecFlags`) a `CMD_SECURITY_IND` is output in addition.
As soon as the connection setup has been completed and all services have been discovered successfully a `CMD_CHANNELOPEN_RSP` is printed by the UART. Now data may be sent using the `CMD_DATA_REQ`.
Format:

Start signal	Command	Length	BTMAC	CS
0x02	0x06	0x06 0x00	6 Bytes	1 Byte

Response (`CMD_CONNECT_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x46	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, try to connect to the device with the `FS_BTMAC`

0x01: Operation failed

0xFF: Operation not permitted

7.2.2 CMD_CONNECT_IND

This telegram indicates the connection status and the `FS_BTMAC` of the connected device. This indication message is the result of a connection request (`CMD_CONNECT_REQ`).
Format:

Start signal	Command	Length	Status	BTMAC	CS
0x02	0x86	0x07 0x00	1 Byte	6 Bytes	1 Byte

Status:

0x00: Physical connection established successfully

0x01: Connection failed, e.g. due to a timeout (as defined by `RF_ScanTiming`)

7.2.3 CMD_SECURITY_IND

This telegram indicates the security status and the `FS_BTMAC` of the connected device. This indication message is the result of a connection request (`CMD_CONNECT_REQ`).
Format:

Start signal	Command	Length	Status	BTMAC	CS
0x02	0x88	0x07 0x00	1 Byte	6 Bytes	1 Byte

Status:

0x00: Encrypted link to previously bonded device established

0x01: Bonding successful, encrypted link established

0x02: No bonding, pairing successful, encrypted link established

7.2.4 CMD_CHANNELOPEN_RSP

This command is printed on the UART as soon as connection setup and service discovery has been completed successfully. Now data can be transmitted using the `CMD_DATA_REQ`. Next to the `FS_BTMAC` of the connected device, the maximum payload size that is supported by the link is part of this telegram. This indication message is the result of a connection request (`CMD_CONNECT_REQ`).
Format:

Start signal	Command	Length	Status	BTMAC	Max payload	CS
0x02	0xC6	0x08 0x00	1 Byte	6 Bytes	1 Byte	1 Byte

Status:

0x00: Success

7.2.5 CMD_DISCONNECT_REQ

This command shuts down the existing connection. Thereafter the module prints a `CMD_DISCONNECT_CNF` to confirm that the request has been received, the indication message `CMD_DISCONNECT_IND` follows which determines whether the disconnection operation has been performed successfully or not.
Format:

Start signal	Command	Length	CS
0x02	0x07	0x00 0x00	0x05

Response (`CMD_DISCONNECT_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x47	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, try to disconnect

0x01: Operation failed

0xFF: Operation not permitted

7.2.6 CMD_DISCONNECT_IND

This telegram indicates that the connection has shut down successfully. This indication message is the result of a disconnection request (`CMD_DISCONNECT_REQ`).
Format:

Start signal	Command	Length	Reason	CS
0x02	0x87	0x01 0x00	1 Byte	1 Byte

Reason:

0x08: Connection timeout

0x13: User terminated connection

0x16: Host terminated connection

0x3B: Connection interval unacceptable

0x3D: Connection terminated due to MIC failure (Not able to connect due to bad link quality, or connection request ignored due to wrong key)

0x3E: Connection setup failed

7.2.7 CMD_PHYUPDATE_REQ

This command allows to update the PHY of the current BLE connection. After the module prints a `CMD_PHYUPDATE_CNF` it tries to update the PHY. The result is indicated by `CMD_PHYUPDATE_IND` message.

Format:

Start signal	Command	Length	PHY	CS
0x02	0x1A	0x01 0x00	1 Byte	1 Byte

PHY:

0x01: 1 MBit PHY

0x02: 2 MBit PHY

0x04: LE Coded mode (1 MBit PHY with DSSS and FEC for higher ranges)

Response (`CMD_PHYUPDATE_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x5A	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received. Try to update PHY of current connection

0x01: Operation failed, e.g. due to invalid PHY

0xFF: Operation not permitted

7.2.8 CMD_PHYUPDATE_IND

This command indicates that there was an attempt to update the PHY of the existing connection. If the PHY update was successful, the command includes the new PHY for receiving and transmitting direction, as well as the BTMAC of the device connected to. This command is the result of the `CMD_PHYUPDATE_REQ`.
Format in case of success:

Start signal	Command	Length	Status	PHY Rx	PHY Tx	BTMAC	CS
0x02	0x9A	0x09 0x00	0x00	1 Byte	1 Byte	6 Bytes	1 Byte

PHY Rx/PHY Tx:

0x01: Using 1 MBit PHY now

0x02: Using 2 MBit PHY now

0x04: Using LE Coded mode (1 MBit PHY with DSSS and FEC for higher ranges) now

Format in case of failure:

Start signal	Command	Length	Status	Info	CS
0x02	0x9A	0x02 0x00	0x01	1 Byte	1 Byte

Info:

0x1A: Unsupported feature of remote device

7.2.9 CMD_PASSKEY_REQ

When receiving a `CMD_PASSKEY_IND` during connection setup, the peripheral requests for a pass key to authenticate the connecting device. To answer this request the `CMD_PASSKEY_REQ` message has to be sent to the Proteus-III central including the passkey of the peripheral. The permissible characters of the passkey are ranging from 0x30 to 0x39 (both included) which are ASCII numbers (0-9).
Format:

Start signal	Command	Length	Pass key	CS
0x02	0x0D	0x06 0x00	6 Bytes	1 Byte

Response (`CMD_PASSKEY_CNF`):

Start signal	Command	0x40	Length	Status	CS
0x02	0x4D	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Pass key accepted and pass key request answered

0x01: Operation failed, due to invalid pass key

0xFF: Operation not permitted

7.2.10 CMD_PASSKEY_IND

Depending on the security settings of the peripheral, a passkey has to be entered on the central side to authenticate the central device. When such a pass key authentication request is received on the central side this `CMD_PASSKEY_IND` message is send to the host. In this case, the passkey has to be entered using the `CMD_PASSKEY_REQ` to successfully finish the connection procedure.

Format:

Start signal	Command	Length	Status	BTMAC	CS
0x02	0x8D	0x07 0x00	1 Byte	6 Bytes	1 Byte

Status:

0x00: Success

7.2.11 CMD_DISPLAY_PASSKEY_IND

Depending on the security settings of the peripheral, a passkey is displayed to show it to the connection partner or to confirm/reject it.

In case of the "LescPasskey" mode, the peripheral outputs the pass key using this message to enter it on the central side for authentication.

In case of the "LescNumComp" mode, the central and peripheral output the passkey to their host. Both hosts must reply with a `CMD_NUMERIC_COMP_REQ` message to confirm that both keys coincide.

Format:

Start signal	Command	Length	Action	BTMAC	Pass key	CS
0x02	0xA4	0x0D 0x00	1 Byte	6 Bytes	6 Bytes	1 Byte

Action:

0x00: Key is displayed to enter it on the central device, no action needed in this device

0x01: Key is displayed, please confirm/reject it using a `CMD_NUMERIC_COMP_REQ`

7.2.12 CMD_NUMERIC_COMP_REQ

Depending on the security settings of the peripheral, a passkey is displayed to confirm or reject it.

In case of the "LescNumComp" mode, the central and peripheral output the passkey to their host. Both hosts must reply with a `CMD_NUMERIC_COMP_REQ` message to confirm that both keys coincide.

Format:

Start signal	Command	Length	Status	CS
0x02	0x24	0x01 0x00	1 Byte	1 Byte

Status:

0x00: The keys displayed on the central and peripheral device coincide, thus connection setup can be continued

0x01: The keys displayed on the central and peripheral device do not coincide, thus connection setup shall the canceled

Response (`CMD_NUMERIC_COMP_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x64	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Answer accepted

0xFF: Operation not permitted

7.2.13 CMD_GETBONDS_REQ

This command requests the MAC addresses of all bonded devices.
Format:

Start signal	Command	Length	CS
0x02	0x0F	0x00 0x00	0x0D

Response (`CMD_GETBONDS_CNF`):

Start signal	Command | 0x40	Length	Status	#Devices	Payload	CS
0x02	0x4F	2 Bytes	1 Byte	1 Byte	(Length - 2) Bytes	1 Byte

The Payload sequentially lists the data of the bonded `#Devices` devices. It consists of `#Devices` times the following telegram (see example below).

Bond_ID	BTMAC
2 Bytes	6 Bytes

Status:

0x00: Request successfully processed

0x01: Operation failed

0xFF: Operation not permitted

> (!) If there are too many devices, the response of the `CMD_GETBONDS_REQ` is split into several `CMD_GETBONDS_CNF` messages.

7.2.13.1 Example 1

Request for the bonding data of the devices in database.

Start signal	Command	Length	CS
0x02	0x0F	0x00 0x00	0x0D

Response:

Start signal	Command | 0x40	Length	Status	#Devices	Payload	CS
0x02	0x4F	0x12 0x00	0x00	0x02	0x00 0x00 0x82 0x5C 0xA7 0xE2 0x87 0xD0 0x01 0x00 0x01 0x00 0x00 0xDA 0x18 0x00	0x53

Two devices have been bonded before:

- Device 1 (Bond_ID 0x0000) with `FS_BTMAC` 0x82 0x5C 0xA7 0xE2 0x87 0xD0

- Device 2 (Bond_ID 0x0001) with `FS_BTMAC` 0x01 0x00 0x00 0xDA 0x18 0x00

7.2.14 CMD_DELETEBONDS_REQ

This command removes the bonding information of all or single bonded devices. Enter Bond_ID to remove the bonding data of a certain Bond_ID. To remove all bonding data, choose Length equals 0 and leave Bond_ID empty.
Format:

Start signal	Command	Length	Bond_ID	CS
0x02	0x0E	2 Bytes	0 or 2 Bytes	1 Byte

Response (`CMD_DELETEBONDS_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x4E	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request successfully processed

0x01: Operation failed (e.g. Bond_ID not found)

0xFF: Operation not permitted

7.2.14.1 Example 1

Request to remove all bonding data.

Start signal	Command	Length	CS
0x02	0x0E	0x00 0x00	0x0C

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x4E	0x01 0x00	0x00	0x4D

Successfully removed all bonding information.

7.2.14.2 Example 2

Request to remove the bonding of the device corresponding to Bond_ID 0.

Start signal	Command	Length	Bond_ID	CS
0x02	0x0E	0x02 0x00	0x00 0x00	0x0E

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x4E	0x01 0x00	0x00	0x4D

Successfully removed the bonding information.

7.3 Transmit and receive data

7.3.1 CMD_DATA_REQ

This command provides the simple data transfer between two connected modules. Transmission takes place to the previously connected device(s). This command is suitable for transmission for a point-to-point connection. The number of payload data Bytes is negotiated during the connection phase. It can be maximal 243 Bytes, but at least 19 Bytes.
When the data is processed by the module a `CMD_DATA_CNF` is output by the UART. Additionally a `CMD_TXCOMPLETE_RSP` will follow as soon as the data has been sent.
The receiving Proteus-III will get a `CMD_DATA_IND` message containing the transmitted payload data.
In "high throughput mode" the length of data packets may be up to 964 Bytes. Format:

Start signal	Command	Length	Payload	CS
0x02	0x04	2 Bytes	Length Bytes	1 Byte

Response (`CMD_DATA_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x44	2 Bytes	Length Bytes	1 Byte

Status:

0x00: Request received, will send data now

0x01 + 0xXX: Operation failed + 0xXX maximum payload size (if it was exceeded)

0xFF: Operation not permitted

7.3.2 CMD_TXCOMPLETE_RSP

This command is output to the UART as soon as the data, which was requested by a `CMD_DATA_REQ` has been transmitted successfully.
Format:

Start signal	Command	Length	Status	CS
0x02	0xC4	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Data transmitted successfully

0x01: Data transmission failed

7.3.3 CMD_DATA_IND

This telegram indicates the reception of data sent by the previously connected device. This indication message is the result of a data request (`CMD_DATA_REQ`) sent to the associated device within a connection.

The `CMD_DATA_IND` returns the `FS_BTMAC` of the sending device, the RSSI value of the received data packet and the data received via the RF-interface, which can be found in the payload. The RSSI value is printed in two's complement notation.

Format:

Start signal	Command	Length	BTMAC	RSSI	Payload	CS
0x02	0x84	2 Bytes	6 Bytes	1 Byte	(Length - 7) Bytes	1 Byte

7.3.4 CMD_SETBEACON_REQ

This command is used to place user data in the scan response packet. The data is broadcasted frequently without acknowledgement and security. No connection is needed for this mode of operation.

It can be received by any scanning Proteus-III with Beacon-function enabled (see `RF_BeaconFlags`). The receiving module will output a `CMD_BEACON_IND` indication message containing the transmitted data. See chapter `5.5` for more information.

Choosing 0x00 as Length and leaving the Payload field empty will remove the data from the scan response packet. The number of payload data Bytes is limited to 19.

Format:

Start signal	Command	Length	Payload	CS
0x02	0x0C	2 Bytes	Length Bytes	1 Byte

Response (`CMD_SETBEACON_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x4C	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, will place data now

0x01: Operation failed

0xFF: Operation not permitted

7.3.5 CMD_BEACON_IND

This telegram indicates the reception of data Bytes that have been transmitted in a beacon-packet. This data can only be received, when the module is in `ACTION_SCANNING` mode and the beacon-function is enabled (see `RF_BeaconFlags`).

The data received via the RF-interface can be found in the payload of the `CMD_BEACON_IND` telegram. Besides this, the `FS_BTMAC` of the sending device and the RSSI value of the data packet are output as well. The RSSI value is output in two's complement notation.

Format:

Start signal	Command	Length	BTMAC	RSSI	Payload	CS
0x02	0x8C	2 Bytes	6 Bytes	1 Byte	(Length - 7) Bytes	1 Byte

7.4 Configuring the module and modifying the device settings

> **[STOP]** It is strongly recommended to have identical settings on all devices, which have to open a connection with each other or are to be used in Beacon mode.

The module's parameters are stored in flash, but have a local copy in RAM. The flash parameters can be modified by the `CMD_SET_REQ`, read by the `CMD_GET_REQ` and retain their content even when resetting the module.

7.4.1 CMD_SET_REQ

This command enables direct manipulation of the parameters in the module's settings in flash. The respective parameters are accessed by means of the corresponding settings index, which can be found in Table 20.
Parameters of 2 or more Bytes have to be transferred with the LSB first unless noted differently in the corresponding description.

> **[!]** The modified parameters only take effect after a restart of the module. This may be done by a `CMD_RESET_REQ` if the module does not restart automatically.

> **[!]** The flash memory used to store these settings has a limited count of write cycles. Try to avoid performing periodic `CMD_SET_REQ` as each command will use one write cycle.

> **[STOP]** The validity of the specified parameters is not verified. Incorrect values can result in device malfunction!

> **[STOP]** To save the parameters in the flash memory of the module, the particular memory segment must first be flushed entirely and then restored from RAM. If a reset occurs during this procedure, the entire memory area may be corrupted (e.g. due to supply voltage fluctuations).
> Recommendation: First, verify the configuration of the module with `CMD_GET_REQ` and only then apply a `CMD_SET_REQ` if required to avoid unnecessary flash cycles.

Format:

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	2 Bytes	1 Byte	(Length - 1) Bytes	1 Byte

Response (`CMD_SET_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, settings set successfully

0x01: Operation failed due to invalid parameter

0x04: Serious error, when writing flash. Try to factory reset or re-flash the device

0x05: Supply voltage to low. Please apply correct supply voltage, reset and retry.

0xFF: Operation not permitted

7.4.1.1 Example 1

Setting the advertising time `RF_AdvertisingTimeout` to 180 seconds.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x07	0xB4 0x00	0xA3

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

Setting was set successfully.

7.4.1.2 Example 2

Setting the static pass key `RF_StaticPasskey` to "123456".

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x07 0x00	0x12	0x31 0x32 0x33 0x34 0x35 0x36	0x01

Response:

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

Setting was set successfully.

7.4.2 CMD_GET_REQ

This command can be used to query individual setting parameters in flash. The respective parameters are accessed by means of the corresponding settings index, which can be found in Table 20.

Parameters of 2 or more Bytes have to be transferred with the LSB first unless noted differently in the corresponding description.

Read access to the memory area outside the setting is blocked.

Format:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	1 Byte	1 Byte

Response (CMD_GET_CNF):

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	2 Bytes	1 Byte	(Length - 1) Bytes	1 Byte

Status:

0x00: Request received, read out of setting successful

0x01: Operation failed

0xFF: Operation not permitted

7.4.2.1 Example 1

Request the current static pass key RF_StaticPasskey.

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x12	0x01

Response: The current RF_StaticPasskey in flash is "123123" (0x31 0x32 0x33 0x31 0x32 0x33).

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x07 0x00	0x00	0x31 0x32 0x33 0x31 0x32 0x33	0x55

Setting was read successfully.

7.5 Manage the device state

7.5.1 CMD_GETSTATE_REQ

This command returns the current state of the module.

> Please refer to chapter 5 for details on the states of the module.

Format:

Start signal	Command	Length	CS
0x02	0x01	0x00 0x00	0x03

Response (CMD_GETSTATE_CNF):

Start signal	Command | 0x40	Length	Module role	Module actions	More info	CS
0x02	0x41	2 Bytes	1 Byte	1 Byte	(Length - 2) Bytes	1 Byte

Module role:

0x00: No role

0x01: Peripheral

0x02: Central

0x10: Direct test mode (DTM)

Other: Reserved

Module action:

0x00: No action

0x01: Idle (advertising)

0x02: Scanning

0x03: Connected (More info is the 6 Bytes FS_BTMAC address of the connected device)

0x04: Sleep (system-off mode)

0x05: Direct test mode

7.5.1.1 Example 1

Get the current state of the module.

Start signal	Command	Length	CS
0x02	0x01	0x00 0x00	0x03

Response:

Start signal	Command | 0x40	Length	Module role	Module actions	More info	CS
0x02	0x41	0x08 0x00	0x02	0x03	0x11 0x00 0x00 0xDA 0x18 0x00	0x99

The module is connected to another module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00.

7.5.2 CMD_RESET_REQ

This command triggers a software reset of the module.
Format:

Start signal	Command	Length	CS
0x02	0x00	0x00 0x00	0x02

Response (`CMD_RESET_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x40	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, will perform reset now

0x01: Operation failed

0xFF: Operation not permitted

7.5.3 CMD_SLEEP_REQ

This command is used to start the system-off mode (`ACTION_SLEEP`). After entering this mode, the module has to be woken up using the *WAKE_UP* pin (apply a low signal at this for at least 5ms and release it to high again) before any other action can be performed. The UART interface as well as the BLE interface are shut down in this mode. For more details, please see also chapter 5.2.
Format:

Start signal	Command	Length	CS
0x02	0x02	0x00 0x00	0x00

Response (`CMD_SLEEP_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x42	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, will go to sleep now

0x01: Operation failed

0xFF: Operation not permitted

> Please note that the *WAKE_UP* pin has a second function. If the module is not in `ACTION_SLEEP` mode and the UART was disabled using the `CMD_UARTDISABLE_REQ`, the UART can be re-enabled by applying falling edge, holding the line low for at least 10ms before applying a rising edge and holding it high for at least 10ms. In this case the module answers with a `CMD_UARTENABLE_IND` message.

7.5.4 CMD_SLEEP_IND

This indication is send by the module when the `RF_AdvertisingTimeout` has expired without a connection to the module.
Format:

Start signal	Command	Length	Status	CS
0x02	0x82	0x01 0x00	0x00	1 Byte

Status:

0x00: Advertising timeout detected, will go to sleep now

7.5.5 CMD_FACTORYRESET_REQ

This command triggers a factory reset of the module. First, the default User Settings are restored, then the module is reset.

> Please note that also the GPIO configuration specified in chapter 11 is reset to default.

Format:

Start signal	Command	Length	CS
0x02	0x1C	0x00 0x00	0x1E

Response (`CMD_FACTORYRESET_CNF`):

Start signal	Command	0x40	Length	Status	CS
0x02	0x5C	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, will perform factory reset now

0x01: Operation failed

0xFF: Operation not permitted

> To save the parameters in the flash memory of the module, the particular memory segment must first be flushed entirely and then restored from RAM. If a reset occurs during this procedure (e.g. due to supply voltage fluctuations), the entire memory area may be destroyed.

> During start-up of the device, the user settings memory is checked for consistency. In case of inconsistency (e.g. the memory was erased) the device will perform a factory reset.

> This command also removes all bonding data.

7.5.6 CMD_UARTDISABLE_REQ

This command disables the UART of the module. It will be re-enabled when the module has to send data to the host (e.g. data was received via RF or a state is indicated) or if the *WAKE_UP* pin is used (apply a falling edge, hold low for at least 10ms before applying a rising edge and hold high for at least 10ms). In this case, either the received data or a `CMD_UARTENABLE_IND` is transmitted by the module. Afterwards the UART will stay active until another `CMD_UARTDISABLE_REQ` or `CMD_SLEEP_REQ` or a timer triggered sleep event occurs.
Format:

Start signal	Command	Length	CS
0x02	0x1B	0x00 0x00	0x19

Response (`CMD_UARTDISABLE_CNF`):

Start signal	Command	0x40	Length	Status	CS
0x02	0x5B	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, will disable UART now

0x01: Operation failed

0xFF: Operation not permitted

> We insistently recommend disabling the UART using this command only, if it is foreseeable that there will be no UART communication for several seconds! Use cases could be during advertising phase to wait for connecting BLE devices or when broadcasting data via Beacons.

> Disabling the UART peripheral of the module results in a reduction of current consumption of about 550µA.

> Please note that the *WAKE_UP* pin has a second function. If the module is in `ACTION_SLEEP` mode, this pin wakes up the module by applying a low signal at this for at least 5ms and releasing it to high. In this case, the module answers with a `CMD_GETSTATE_CNF`.

7.5.7 CMD_UARTENABLE_IND

This indication is shown when the UART of the module is re-enabled (after performing a `CMD_UARTDISABLE_REQ` followed by using the *WAKE_UP* pin). After receiving this message the UART can be used for any operation again.
Format:

Start signal	Command	Length	Status	CS
0x02	0x9B	0x01 0x00	1 Byte	1 Byte

Status:

0x00: UART has been re-enabled successfully

7.5.8 CMD_BOOTLOADER_REQ

This command resets the module and starts the OTA bootloader.

> Please refer to chapter 14 on how to use the bootloader for a firmware update.

Format:

Start signal	Command	Length	CS
0x02	0x1F	0x00 0x00	0x1D

Response (`CMD_BOOTLOADER_CNF`):

Start signal	Command	0x40	Length	Status	CS
0x02	0x5F	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, will start bootloader now

0x01: Operation failed

0xFF: Operation not permitted

7.6 Run the Bluetooth test modes

The test modes "DTM" as specified by the Bluetooth® SIG are defined in the Bluetooth® Core specification.

7.6.1 CMD_DTMSTART_REQ

This command restarts the module in direct test mode (DTM). When starting in DTM mode, a `CMD_GETSTATE_CNF` message follows which indicates that the test mode has been enabled successfully. Now the `CMD_DTM_REQ` can be used to start and stop various test modes.
As soon as the module is reset, the DTM will be left again and normal operations can be performed.
Performing a reset will leave the DTM and restart the module in the `ACTION_IDLE` state.
Format:

Start signal	Command	Length	CS
0x02	0x1D	0x00 0x00	0x1F

Response (`CMD_DTMSTART_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x5D	0x01 0x00	1 Byte	1 Byte

Status:

0x00: Request received, will enable the direct test mode now

0x01: Operation failed

0xFF: Operation not permitted

7.6.2 CMD_DTM_REQ

This command starts and stops various test modes. To be able to run these test modes, the DTM has to be enabled first using the `CMD_DTMSTART_REQ`. After a test has been started, it has to be stopped first before a next test can be run.
The default TX power value is 8 dBm, the allowed range is from -40 up to +8 dBm (see chapter 8.17 for valid TX power values). The valid range for channel is 0...39.
Format:

Start signal	Command	Length	Command code	Channel / Vendor option	Length / Vendor command	Payload	CS
0x02	0x1E	0x04 0x00	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte

Command code:

0x00: DTM setup

Vendor option	Vendor command	Payload
0x00: Reset DTM	0x00	0x00
0x02: Set phy	New phy 1. 0x01: 1MBit 2. 0x02: 2MBit 3. 0x03: S8 LE Coded 4. 0x04: S2 LE Coded	0x00

0x01: Start RX test

Channel	Length	Payload
Frequency = (2402 + Channel * 2) MHz to be used for RX	0x00	0x00

0x02: Start TX test

Channel	Length	Payload
Frequency = (2402 + Channel * 2) MHz to be used for TX	Length of the packet to send	Bit pattern **0x00:** PRBS9 **0x01:** 0x0F **0x02:** 0x55

Vendor option	Vendor command	Payload
Frequency = (2402 + Channel * 2) MHz to be used for TX	0x00: Carrier test	**0x03:** Vendor specific
TX power -40 up to +8 dBm (see chapter 8.17 for valid TX power values)	0x02: Set TX power	**0x03:** Vendor specific

0x03: Stop last test

Channel	Length	Payload
0x00	0x00	0x00

Response (`CMD_DTM_CNF`):

Start signal	Command | 0x40	Length	Status	Result	CS
0x02	0x5E	2 Bytes	1 Byte	0-2 Bytes	1 Byte

Status:

0x00: Request received

0x01: Operation failed

0x03: Busy

0xFF: Operation not permitted

Result:

0x0000: Test success

0x0001: Test failed

0x8000 + n: Received n packets during RX test

See also the example in chapter 5.8.

7.6.2.1 Example: Transmission, 16 times 0x0F, channel 0

Start the transmission test on channel 0 (2402 MHz). The packets consist of 16 times 0x0F:

Start signal	Command	Length	Command code	Channel / Vendor option	Length / Vendor command	Payload	CS
0x02	0x1E	0x04 0x00	0x02	0x00	0x10	0x01	0x0B

Response:

Start signal	Command | 0x40	Length	Status	Result	CS
0x02	0x5E	0x03 0x00	0x00	0x00 0x00	0x5F

Test started successfully. Now stop the test again.

Start signal	Command	Length	Command code	Channel / Vendor option	Length / Vendor command	Payload	CS
0x02	0x1E	0x04 0x00	0x03	0x00	0x00	0x01	0x1A

Response:

Start signal	Command	0x40	Length	Status	Result	CS
0x02	0x5E	0x03 0x00	0x00	0x80 0x00	0xDF

Test stopped successfully and received 0 packets.

7.6.2.2 Example: Receiver, channel 0

Start the reception test on channel 0 (2402MHz):

Start signal	Command	Length	Command code	Channel / Vendor option	Length / Vendor command	Payload	CS
0x02	0x1E	0x04 0x00	0x01	0x00	0x00	0x00	0x19

Response:

Start signal	Command	0x40	Length	Status	Result	CS
0x02	0x5E	0x03 0x00	0x00	0x00 0x00	0x5F

Test started successfully. In between we started the transmission test on a second module. When we stop RX test now, we can count the received packets from the transmitting module.

Start signal	Command	Length	Command code	Channel / Vendor option	Length / Vendor command	Payload	CS
0x02	0x1E	0x04 0x00	0x03	0x00	0x00	0x01	0x0B

Response:

Start signal	Command	0x40	Length	Status	Result	CS
0x02	0x5E	0x03 0x00	0x00	0x8E 0x67	0xB6

Test stopped successfully and received 0x0E67 (3687) packets.

7.6.2.3 Example: Transmission, carrier test, channel 0

Start the carrier test on channel 0 (2402MHz). We need to use a vendor specific command:

Start signal	Command	Length	Command code	Channel / Vendor option	Length / Vendor command	Payload	CS
0x02	0x1E	0x04 0x00	0x02	0x00	0x00	0x03	0x19

Response:

Start signal	Command | 0x40	Length	Status	Result	CS
0x02	0x5E	0x03 0x00	0x00	0x00 0x00	0x5F

See the previous example to stop the test again.

7.6.2.4 Example: Set TX power to -4 dBm

Set the TX power to -4dBm (0xFC in two's complement notation):

Start signal	Command	Length	Command code	Channel / Vendor option	Length / Vendor command	Payload	CS
0x02	0x1E	0x04 0x00	0x02	0xFC	0x02	0x03	0xE7

Response:

Start signal	Command | 0x40	Length	Status	Result	CS
0x02	0x5E	0x03 0x00	0x00	0x00 0x00	0x5F

7.6.2.5 Example: Set PHY to 2MBit mode

Set the phy to 2MBit mode:

Start signal	Command	Length	Command code	Channel / Vendor option	Length / Vendor command	Payload	CS
0x02	0x1E	0x04 0x00	0x00	0x02	0x02	0x00	0x18

Response:

Start signal	Command	0x40	Length	Status	Result	CS
0x02	0x5E	0x03 0x00	0x00	0x00 0x00	0x5F

7.7 Switching GPIOs by remote control

This chapter contains the commands to use the GPIO feature of the Proteus-III. Please refer to chapter `11` for a detailed description.

7.7.1 CMD_GPIO_LOCAL_WRITECONFIG_REQ

This command configures the free GPIOs of the radio module. This is necessary to allow local and remote GPIO control. As the configuration is stored in flash, it is retained after restarting the device.

> The flash memory used to store these settings has a limited count of write cycles. Try to avoid performing periodic `CMD_GPIO_LOCAL_WRITECONFIG_REQ` as each command will use one write cycle.

Format:

Start signal	Command	Length	Block$_1$	. . .	Block$_n$	CS
0x02	0x25	2 Bytes	x Bytes		x Bytes	1 Byte

Response (`CMD_GPIO_LOCAL_WRITECONFIG_CNF`):

Start signal	Command | 0x40	Length	Status	Block$_1$	. . .	Block$_n$	CS
0x02	0x65	2 Bytes	1 Byte	x Bytes		x Bytes	1 Byte

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted

CMD_GPIO_LOCAL_WRITECONFIG_REQ block structure

Each **Block** has the following format:

Length	GPIO_ID	Function	Value
0x03	1 Byte	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Function:

0x01: GPIO works as input

0x02: GPIO works as output

Value:

- if **Function** is input:

    0x00: GPIO has no pull resistor

    0x01: GPIO has pull down resistor

    0x02: GPIO has pull up resistor
- if **Function** is output:

    0x00: GPIO is output low

    0x01: GPIO is output high

CMD_GPIO_LOCAL_WRITECONFIG_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Status
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Status:

0x00: Success

0x01: Failed

7.7.1.1 Example: Configure two GPIOs to output high

Configure the GPIOs with ID **0x01** and **0x02** to output high:

Start signal	Command	Length	Block$_1$	Block$_2$	CS
0x02	0x25	0x08 0x00	0x03 **0x01** 0x02 0x01	0x03 **0x02** 0x02 0x01	0x2C

Response:

Start signal	Command | 0x40	Length	Status	Block$_1$	Block$_2$	CS
0x02	0x65	0x07 0x00	0x00	0x02 **0x01** 0x00	0x02 **0x02** 0x00	0x63

Configured both GPIOs with success.

7.7.2 CMD_GPIO_LOCAL_READCONFIG_REQ

This command reads the current configuration of the free GPIOs of the radio module.
Format:

Start signal	Command	Length	CS
0x02	0x2B	0x00 0x00	0x29

Response (`CMD_GPIO_LOCAL_READCONFIG_CNF`):

Start signal	Command | 0x40	Length	Status	Block₁	. . .	Blockₙ	CS
0x02	0x6B	2 Bytes	1 Byte	x Bytes		x Bytes	1 Byte

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted

CMD_GPIO_LOCAL_READCONFIG_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Function	Value
0x03	1 Byte	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Function:

> **0x00:** GPIO is not configured yet (Length is 0x02 and **Value** is empty)
> **0x01:** GPIO works as input
> **0x02:** GPIO works as output

Value:

> • if **Function** is input:
> > **0x00:** GPIO has no pull resistor
> > **0x01:** GPIO has pull down resistor
> > **0x02:** GPIO has pull up resistor
> • if **Function** is output:
> > **0x00:** GPIO is output low
> > **0x01:** GPIO is output high

7.7.2.1 Example: Read the current GPIO configuration

Read the current configuration:

Start signal	Command	Length	CS
0x02	0x2B	0x00 0x00	0x29

Response:

Start signal	Command | 0x40	Length	Status	Blocks	CS
0x02	0x6B	0x15 0x00	0x00	0x03 **0x01** 0x02 0x01 0x03 **0x02** 0x02 0x01 0x02 **0x03** 0x00 0x02 **0x04** 0x00 0x02 **0x05** 0x00 0x02 **0x06** 0x00	0x7B

The GPIOs with GPIO_ID **0x01** and **0x02** are output high. The remaining GPIOs with GPI-O_ID **0x03**,**0x04**,**0x05** and **0x06** are not configured.

7.7.3 CMD_GPIO_REMOTE_WRITECONFIG_REQ

This command configures the free GPIOs of the connected remote device. This is necessary to allow remote GPIO control. As the configuration is stored in flash, it is retained after restarting the device. This command can be run successfully only if the remote device is connected via BLE.

> (!) The flash memory used to store these settings has a limited count of write cycles. Try to avoid performing periodic CMD_GPIO_REMOTE_WRITECONFIG_REQ as each command will use one write cycle.

Format:

Start signal	Command	Length	Block$_1$	. . .	Block$_n$	CS
0x02	0x28	2 Bytes	x Bytes		x Bytes	1 Byte

Response (CMD_GPIO_REMOTE_WRITECONFIG_CNF):

Start signal	Command | 0x40	Length	Status	Block$_1$	. . .	Block$_n$	CS
0x02	0x68	2 Bytes	1 Byte	x Bytes		x Bytes	1 Byte

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted

CMD_GPIO_REMOTE_WRITECONFIG_REQ block structure

Each **Block** has the following format:

Length	GPIO_ID	Function	Value
0x03	1 Byte	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 11.1

Function:

  0x01: GPIO works as input
  0x02: GPIO works as output

Value:

- if **Function** is input:

    0x00: GPIO has no pull resistor

    0x01: GPIO has pull down resistor

    0x02: GPIO has pull up resistor

- if **Function** is output:

    0x00: GPIO is output low

    0x01: GPIO is output high

CMD_GPIO_REMOTE_WRITECONFIG_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Status
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Status:

0x00: Success

0x01: Failed

0xFF: Remote configuration not allowed (blocked by the user setting `CFG_Flags` of the remote device)

7.7.3.1 Example: Configure two GPIOs of the connected remote device to output high

Configure the GPIOs with ID **0x01** and **0x02** to output high:

Start signal	Command	Length	Block₁	Block₂	CS
0x02	0x28	0x08 0x00	0x03 **0x01** 0x02 0x01	0x03 **0x02** 0x02 0x01	0x21

Response:

Start signal	Command | 0x40	Length	Status	Block₁	Block₂	CS
0x02	0x68	0x07 0x00	0x00	0x02 **0x01** 0x00	0x02 **0x02** 0x00	0x6E

Configured both GPIOs with success.

7.7.4 CMD_GPIO_REMOTE_READCONFIG_REQ

This command reads the current configuration of the free GPIOs of the connected remote device.
Format:

Start signal	Command	Length	CS
0x02	0x2C	0x00 0x00	0x2E

Response (`CMD_GPIO_REMOTE_READCONFIG_CNF`):

Start signal	Command | 0x40	Length	Status	Block$_1$	. . .	Block$_n$	CS
0x02	0x6C	2 Bytes	1 Byte	x Bytes		x Bytes	1 Byte

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted

CMD_GPIO_REMOTE_READCONFIG_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Function	Value
0x03	1 Byte	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Function:

    0x00: GPIO is not configured yet (Length is 0x02 and **Value** is empty)

    0x01: GPIO works as input

    0x02: GPIO works as output

Value:

- if **Function** is input:
   0x00: GPIO has no pull resistor
   0x01: GPIO has pull down resistor
   0x02: GPIO has pull up resistor
- if **Function** is output:
   0x00: GPIO is output low
   0x01: GPIO is output high

7.7.4.1 Example: Read the current GPIO configuration of the connected remote device

Read the current GPIO configuration of the connected remote device:

Start signal	Command	Length	CS
0x02	0x2C	0x00 0x00	0x2E

Response:

Start signal	Command | 0x40	Length	Status	Blocks	CS
0x02	0x6C	0x15 0x00	0x00	0x03 **0x01** 0x02 0x01 0x03 **0x02** 0x02 0x01 0x02 **0x03** 0x00 0x02 **0x04** 0x00 0x02 **0x05** 0x00 0x02 **0x06** 0x00	0x7C

The GPIOs with GPIO_ID **0x01** and **0x02** are output high. The remaining GPIOs with GPIO_ID **0x03**,**0x04**,**0x05** and **0x06** are not configured.

7.7.5 CMD_GPIO_REMOTE_WRITE_REQ

This command writes the free GPIOs of the remote device. This command can be only run successfully if the respective pins of the remote device have been configured as output pins before and the remote device is connected via BLE.
Format:

Start signal	Command	Length	Block$_1$	. . .	Block$_n$	CS
0x02	0x29	2 Bytes	x Bytes		x Bytes	1 Byte

Response (`CMD_GPIO_REMOTE_WRITE_CNF`):

Start signal	Command | 0x40	Length	Status	Block$_1$	. . .	Block$_n$	CS
0x02	0x69	2 Bytes	1 Byte	x Bytes		x Bytes	1 Byte

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted (i.e. no device connected via BLE)

CMD_GPIO_REMOTE_WRITE_REQ block structure

Each **Block** has the following format:

Length	GPIO_ID	Value
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Value:

    0x00: Set GPIO to low
    0x01: Set GPIO to high

CMD_GPIO_REMOTE_WRITE_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Status
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Status:

      0x00: Success

      0x01: Failed

7.7.5.1 Example: Set a remote output GPIO to low

Set the output GPIO (GPIO_ID **0x01**) of the connected remote device to low:

Start signal	Command	Length	Block$_1$	CS
0x02	0x29	0x03 0x00	0x02 **0x01** 0x00	0x2B

Response:

Start signal	Command | 0x40	Length	Status	Block$_1$	CS
0x02	0x69	0x04 0x00	0x00	0x02 **0x01** 0x00	0x6C

Successfully set GPIO with GPIO_ID **0x01** to low.

7.7.6 CMD_GPIO_REMOTE_READ_REQ

This command reads the free GPIOs of the remote device. This command can be only run successfully if the respective pins of the remote device have been configured as output or input pins before and the remote device is connected via BLE.

Format:

Start signal	Command	Length	Block$_1$	. . .	Block$_n$	CS
0x02	0x2A	2 Bytes	x Bytes		x Bytes	1 Byte

Response (`CMD_GPIO_REMOTE_READ_CNF`):

Start signal	Command | 0x40	Length	Status	Block$_1$	. . .	Block$_n$	CS
0x02	0x6A	2 Bytes	1 Byte	x Bytes		x Bytes	1 Byte

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted (i.e. no device connected via BLE)

CMD_GPIO_REMOTE_READ_REQ block structure

Each **Block** has the following format:

Length	GPIO_ID$_1$	. . .	GPIO_ID$_n$
1 Bytes	1 Byte		1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

CMD_GPIO_REMOTE_READ_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Value
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Value:

    0x00: The remote GPIO is low.

    0x01: The remote GPIO is high.

    0xFF: Failed reading remote GPIO value.

7.7.6.1 Example: Read the values of remote GPIOs

Read the value of the GPIOs with GPIO_ID **0x01** and **0x02** of the connected remote device:

Start signal	Command	Length	Block$_1$	CS
0x02	0x2A	0x03 0x00	0x02 **0x01 0x02**	0x2A

Response:

Start signal	Command | 0x40	Length	Status	Block$_1$	Block$_2$	CS
0x02	0x6A	0x07 0x00	0x00	0x02 **0x01** 0x00	0x02 **0x02** 0x01	0x6D

Successfully read the values of the remote GPIOs with GPIO_ID **0x01** (GPIO is low) and **0x02** (GPIO is high).

7.7.7 CMD_GPIO_LOCAL_WRITE_REQ

This command writes the free GPIOs of the local device. This command can be only run successfully if the respective pins of the local device have been configured as output pins before.

Format:

Start signal	Command	Length	Block₁	. . .	Blockₙ	CS
0x02	0x26	2 Bytes	x Bytes		x Bytes	1 Byte

Response (`CMD_GPIO_LOCAL_WRITE_CNF`):

Start signal	Command | 0x40	Length	Status	Block₁	. . .	Blockₙ	CS
0x02	0x66	2 Bytes	1 Byte	x Bytes		x Bytes	1 Byte

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted (i.e. no device connected via BLE)

CMD_GPIO_LOCAL_WRITE_REQ block structure

Each **Block** has the following format:

Length	GPIO_ID	Value
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Value:

    0x00: Set GPIO to low

    0x01: Set GPIO to high

CMD_GPIO_LOCAL_WRITE_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Status
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Status:

    0x00: Success

    0x01: Failed

7.7.7.1 Example: Set a local output GPIO to low

Set the output GPIO (GPIO_ID **0x01**) of the local device to low:

Start signal	Command	Length	Block$_1$	CS
0x02	0x26	0x03 0x00	0x02 **0x01** 0x00	0x24

Response:

Start signal	Command | 0x40	Length	Status	Block$_1$	CS
0x02	0x66	0x04 0x00	0x00	0x02 **0x01** 0x00	0x63

Successfully set GPIO with GPIO_ID **0x01** to low.

7.7.8 CMD_GPIO_LOCAL_READ_REQ

This command reads the free GPIOs of the local device. This command can be only run successfully if the respective pins of the local device have been configured as output or input pins before.
Format:

Start signal	Command	Length	Block$_1$	. . .	Block$_n$	CS
0x02	0x27	2 Bytes	x Bytes		x Bytes	1 Byte

Response (`CMD_GPIO_LOCAL_READ_CNF`):

Start signal	Command | 0x40	Length	Status	Block$_1$	. . .	Block$_n$	CS
0x02	0x67	2 Bytes	1 Byte	x Bytes		x Bytes	1 Byte

Status:

0x00: Request received and processed

0x01: Operation failed

0xFF: Operation not permitted (i.e. no device connected via BLE)

CMD_GPIO_LOCAL_READ_REQ block structure

Each **Block** has the following format:

Length	GPIO_ID$_1$	. . .	GPIO_ID$_n$
1 Bytes	1 Byte		1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

CMD_GPIO_LOCAL_READ_CNF block structure

Each **Block** has the following format:

Length	GPIO_ID	Value
0x02	1 Byte	1 Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter `11.1`

Value:

    0x00: The remote GPIO is low.

    0x01: The remote GPIO is high.

    0xFF: Failed reading remote GPIO value.

7.7.8.1 Example: Read the values of local GPIOs

Read the value of the GPIOs with GPIO_ID **0x01** and **0x02** of the local device:

Start signal	Command	Length	Block₁	CS
0x02	0x27	0x03 0x00	0x02 **0x01 0x02**	0x27

Response:

Start signal	Command | 0x40	Length	Status	Block₁	Block₂	CS
0x02	0x67	0x07 0x00	0x00	0x02 **0x01** 0x00	0x02 **0x02** 0x01	0x60

Successfully read the values of the local GPIOs with GPIO_ID **0x01** (GPIO is low) and **0x02** (GPIO is high).

7.7.9 CMD_GPIO_REMOTE_WRITECONFIG_IND

This command indicates that the remote device has written the free GPIOs of the radio module.

> ⚠ Please note that only the GPIOs are part of this message, that have been configured successfully. Failed attempts of GPIO configurations will not be indicated by this message.

Format:

Start signal	Command	Length	Block$_1$	. . .	Block$_n$	CS
0x02	0xA8	2 Bytes	x Bytes		x Bytes	1 Byte

The **Block** structure is as defined in `CMD_GPIO_REMOTE_WRITECONFIG_REQ` block structure.

7.7.9.1 Example: Two GPIOs have been configured by the connected remote device to output high

Start signal	Command	Length	Block$_1$	Block$_2$	CS
0x02	0xA8	0x08 0x00	0x03 **0x01** 0x02 0x01	0x03 **0x02** 0x02 0x01	A1

The two GPIOs with ID **0x01** and **0x02** have been configured by the connected remote device to output high.

7.7.10 CMD_GPIO_REMOTE_WRITE_IND

This command indicates that the remote device has written the free GPIOs of the radio module.

Please note that only the GPIOs are part of this message, that have been updated successfully. Failed attempts of GPIO updates will not be indicated by this message.

Format:

Start signal	Command	Length	Block$_1$	. . .	Block$_n$	CS
0x02	0xA9	2 Bytes	x Bytes		x Bytes	1 Byte

The **Block** structure is as defined in `CMD_GPIO_LOCAL_READ_CNF block structure.`

7.7.10.1 Example: GPIOs have been written via remote access

Start signal	Command	Length	Status	Block$_1$	Block$_2$	CS
0x02	0xA9	0x07 0x00	0x00	0x02 **0x01** 0x00	0x02 **0x02** 0x01	0xAE

The remote device has written the GPIOs with GPIO_ID **0x01** (GPIO is low) and **0x02** (GPIO is high).

7.7.11 CMD_GPIO_LOCAL_WRITE_IND

This command indicates that the GPIOs of the remote device have been written by its local host.

> **!** Please note that only the GPIOs are part of this message, that have been updated successfully. Failed attempts of GPIO updates will not be indicated by this message.

Format:

Start signal	Command	Length	Block$_1$	...	Block$_n$	CS
0x02	0xA6	2 Bytes	x Bytes		x Bytes	1 Byte

The **Block** is of structure as defined in `CMD_GPIO_LOCAL_READ_CNF block structure`.

7.7.11.1 Example: GPIOs of the remote device have been written by its local host

Start signal	Command	Length	Status	Block$_1$	Block$_2$	CS
0x02	0xA6	0x07 0x00	0x00	0x02 **0x01** 0x00	0x02 **0x02** 0x01	0xA1

The GPIOs with GPIO_ID **0x01** (GPIO is low) and **0x02** (GPIO is high) of the radio module have been written by its local host.

7.8 Other messages

7.8.1 CMD_ERROR_IND

This indication is shown when the module entered an error state.
Format:

Start signal	Command	Length	Status	CS
0x02	0xA2	0x01 0x00	1 Byte	1 Byte

Status:

0x01: UART_COMMUNICATION_ERROR The UART had a buffer overflow. Thus, UART TX and RX was aborted and UART has restarted. Please restart module if UART is still malfunctioning.

7.9 Message overview

Start signal	CMD	Message name	Short description	Chapter
0x02	0x00	`CMD_RESET_REQ`	Reset the module	7.5.2
0x02	0x01	`CMD_GETSTATE_REQ`	Request the current module state	7.5.1
0x02	0x02	`CMD_SLEEP_REQ`	Go to sleep	7.5.3
0x02	0x04	`CMD_DATA_REQ`	Send data to the connected device	7.3.1
0x02	0x06	`CMD_CONNECT_REQ`	Setup a connection with another device	7.2.1
0x02	0x07	`CMD_DISCONNECT_REQ`	Close the connection	7.2.5
0x02	0x09	`CMD_SCANSTART_REQ`	Start scan	7.1.1
0x02	0x0A	`CMD_SCANSTOP_REQ`	Stop scan	7.1.2
0x02	0x0B	`CMD_GETDEVICES_REQ`	Request the scanned/detected devices	7.1.3
0x02	0x0C	`CMD_SETBEACON_REQ`	Place data in scan response packet	7.3.4
0x02	0x0D	`CMD_PASSKEY_REQ`	Respond to a pass key request	7.2.9
0x02	0x0E	`CMD_DELETEBONDS_REQ`	Delete bonding information	7.2.14
0x02	0x0F	`CMD_GETBONDS_REQ`	Read the MACs of bonded devices	7.2.13
0x02	0x10	`CMD_GET_REQ`	Read the module settings in flash	7.4.2
0x02	0x11	`CMD_SET_REQ`	Modify the module settings in flash	7.4.1
0x02	0x1A	`CMD_PHYUPDATE_REQ`	Update the PHY	7.2.7
0x02	0x1B	`CMD_UARTDISABLE_REQ`	Disable the UART	7.5.6
0x02	0x1C	`CMD_FACTORYRESET_REQ`	Perform a factory reset	7.5.5
0x02	0x1D	`CMD_DTMSTART_REQ`	Enable the direct test mode	7.6.1
0x02	0x1E	`CMD_DTM_REQ`	Start/stop a test of the direct test mode	7.6.2
0x02	0x1F	`CMD_BOOTLOADER_REQ`	Switch to the bootloader	7.5.8
0x02	0x24	`CMD_NUMERIC_COMP_REQ`	Confirm/reject the displayed pass key	7.2.12

Table 10: Message overview: Requests 1

Start signal	CMD	Message name	Short description	Chapter
0x02	0x25	`CMD_GPIO_LOCAL_WRITECONFIG_REQ`	Configure the free GPIOs for remote control	7.7.1
0x02	0x26	`CMD_GPIO_LOCAL_WRITE_REQ`	Set the output value of a output GPIO of the current device	7.7.7
0x02	0x27	`CMD_GPIO_LOCAL_READ_REQ`	Read the value of a GPIO of the current device	7.7.8
0x02	0x28	`CMD_GPIO_REMOTE_WRITECONFIG_REQ`	Configure the free GPIOs of the remote device for remote control	7.7.3
0x02	0x29	`CMD_GPIO_REMOTE_WRITE_REQ`	Set the output value of a output GPIO of a remote device	7.7.5
0x02	0x2A	`CMD_GPIO_REMOTE_READ_REQ`	Read the value of a GPIO of a remote device	7.7.6
0x02	0x2B	`CMD_GPIO_LOCAL_READCONFIG_REQ`	Read the GPIO configuration	7.7.2
0x02	0x2C	`CMD_GPIO_REMOTE_READCONFIG_REQ`	Read the GPIO configuration of the connected remote device	7.7.4

Table 11: Message overview: Requests 2

Start signal	CMD	Message name	Short description	Chapter
0x02	0x40	`CMD_RESET_CNF`	Reset request received	`7.5.2`
0x02	0x41	`CMD_GETSTATE_CNF`	Return the current module state	`7.5.1`
0x02	0x42	`CMD_SLEEP_CNF`	Sleep request received	`7.5.3`
0x02	0x44	`CMD_DATA_CNF`	Data transmission request received	`7.3.1`
0x02	0x46	`CMD_CONNECT_CNF`	Connection setup request received	`7.2.1`
0x02	0x47	`CMD_DISCONNECT_CNF`	Disconnection request received	`7.2.5`
0x02	0x49	`CMD_SCANSTART_CNF`	Scan started	`7.1.1`
0x02	0x4A	`CMD_SCANSTOP_CNF`	Scan stopped	`7.1.2`
0x02	0x4B	`CMD_GETDEVICES_CNF`	Output the scanned/detected devices	`7.1.3`
0x02	0x4C	`CMD_SETBEACON_CNF`	Data is placed in scan response packet	`7.3.4`
0x02	0x4D	`CMD_PASSKEY_CNF`	Received the pass key	`7.2.9`
0x02	0x4E	`CMD_DELETEBONDS_CNF`	Deleted bonding information	`7.2.14`
0x02	0x4F	`CMD_GETBONDS_CNF`	Return the MAC of all bonded devices	`7.2.13`
0x02	0x50	`CMD_GET_CNF`	Return the requested module flash settings	`7.4.2`
0x02	0x51	`CMD_SET_CNF`	Module flash settings have been modified	`7.4.1`
0x02	0x5A	`CMD_PHYUPDATE_CNF`	Update Phy request received	`7.2.7`
0x02	0x5B	`CMD_UARTDISABLE_CNF`	Disable UART request received	`7.5.6`
0x02	0x5C	`CMD_FACTORYRESET_CNF`	Factory reset request received	`7.5.5`
0x02	0x5D	`CMD_DTMSTART_CNF`	Enable the direct test mode now	`7.6.1`
0x02	0x5E	`CMD_DTM_CNF`	Test of direct test mode started/stopped	`7.6.2`
0x02	0x5F	`CMD_BOOTLOADER_CNF`	Will switch to bootloader now	`7.5.8`
0x02	0x64	`CMD_NUMERIC_COMP_CNF`	`CMD_NUMERIC_COMP_REQ` accepted	`7.2.12`
0x02	0x65	`CMD_GPIO_LOCAL_WRITECONFIG_CNF`	Configuration of a local GPIO for remote control done	`7.7.1`
0x02	0x66	`CMD_GPIO_LOCAL_WRITE_CNF`	Output value of a local GPIO set	`7.7.7`
0x02	0x67	`CMD_GPIO_LOCAL_READ_CNF`	Value of a local GPIO read	`7.7.8`
0x02	0x68	`CMD_GPIO_REMOTE_WRITECONFIG_CNF`	Configuration of a remote GPIO for remote control done	`7.7.3`
0x02	0x69	`CMD_GPIO_REMOTE_WRITE_CNF`	Output value of a remote GPIO set	`7.7.5`
0x02	0x6A	`CMD_GPIO_REMOTE_READ_CNF`	Value of a remote GPIO read	`7.7.6`

Table 12: Message overview: Confirmations 1

Start signal	CMD	Message name	Short description	Chapter
0x02	0x6B	`CMD_GPIO_LOCAL_READCONFIG_CNF`	Returns the GPIO configuration	7.7.2
0x02	0x6C	`CMD_GPIO_REMOTE_READCONFIG_CNF`	Returns the GPIO configuration of the connected remote device	7.7.4

Table 13: Message overview: Confirmations 2

Start signal	CMD	Message name	Short description	Chapter
0x02	0x82	`CMD_SLEEP_IND`	State will be changed to `ACTION_SLEEP`	7.5.4
0x02	0x84	`CMD_DATA_IND`	Data has been received	7.3.3
0x02	0x86	`CMD_CONNECT_IND`	Connection established	7.2.2
0x02	0x87	`CMD_DISCONNECT_IND`	Disconnected	7.2.6
0x02	0x88	`CMD_SECURITY_IND`	Secured connection established	7.2.3
0x02	0x8B	`CMD_RSSI_IND`	Proteus-III advertising package detected	7.1.4
0x02	0x8C	`CMD_BEACON_IND`	Received Beacon data	7.3.5
0x02	0x8D	`CMD_PASSKEY_IND`	Received a pass key request	7.2.10
0x02	0x9A	`CMD_PHYUPDATE_IND`	PHY has been updated	7.2.8
0x02	0x9B	`CMD_UARTENABLE_IND`	UART was re-enabled	7.5.7
0x02	0xA2	`CMD_ERROR_IND`	Entered error state	7.8.1
0x02	0xA4	`CMD_DISPLAY_PASSKEY_IND`	Display pass key	7.2.11
0x02	0xA6	`CMD_GPIO_LOCAL_WRITE_IND`	Local host has written the GPIOs of the remote device	7.7.11
0x02	0xA8	`CMD_GPIO_REMOTE_WRITECONFIG_IND`	Remote device has configured the GPIOs of the module	7.7.9
0x02	0xA9	`CMD_GPIO_REMOTE_WRITE_IND`	Remote device has written the GPIOs of the module	7.7.10
0x02	0xC4	`CMD_TXCOMPLETE_RSP`	Data has been sent	7.3.2
0x02	0xC6	`CMD_CHANNELOPEN_RSP`	Channel open, data transmission possible	7.2.4
0x02	0xCC	`CMD_BEACON_RSP`	Advertising package detected	7.1.5

Table 14: Message overview: Indications

8 UserSettings - Module configuration values

The settings described in this chapter are stored permanently in the module's flash memory. Depending on their corresponding permissions, their current values can be read out by the `CMD_GET_REQ` command or modified by the `CMD_SET_REQ` command. To do so the corresponding settings index is used, which can be found in the primary table of each setting description.

> **STOP** The validity of the specified parameters is not verified. Incorrect values can result in device malfunction.

> **STOP** After the modification of the non-volatile parameters, a reset will be necessary for the changes to be applied.

8.1 FS_DeviceInfo: Read the chip type and OS version

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
15	FS_DeviceInfo	-	-	read	12

This setting contains information about the chip type and the OS version. The value of `FS_DeviceInfo` is composed of the following 4 sub parameters (ordered by appearance in the response):

OS version	Build code	Package variant	Chip ID
2 Bytes	4 Bytes	2 Bytes	4 Bytes

OS version:

0x00B6 : Softdevice S140 6.1.1.

0x00C1 : Softdevice S140 7.0.0.

0x00CA : Softdevice S140 7.0.1.

Package variant:

0x2004: QFN - QI

0x2005: WLCSP - CK

Chip ID:

0x00052840: nRF52840

Packet variant	Package	Flash size	RAM size
QFN	QFN73	1024 kB	256 kB
WLCSP	WLCSP	1024 kB	256 kB

Table 15: nRF52840 IC revision overview

8.1.1 Example 1

Request the device info of the module using `CMD_GET_REQ` with settings index 15

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x0F	0x1C

Response `CMD_GET_CNF`: Successfully read out the device info (with Byte order changed to MSB first):
OS version = 0x00B6 (Softdevice S140 6.1.1)
Build code = 0x41414300 (AAC0)
Package variant = 0x2004 (QFN)
Chip ID = 0x00052840
Please note that LSB is transmitted first in case of parameters with more than 1 Byte length.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x0D 0x00	0x00	0xB6 0x00 0x30 0x43 0x41 0x41 0x04 0x20 0x40 0x28 0x05 0x00	0xD3

8.2 FS_FWVersion: Read the firmware version

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
1	FS_FWVersion	-	-	read	3

This setting contains the firmware version of the module.

8.2.1 Example 1

Request the firmware version of the module using CMD_GET_REQ with settings index 1

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x01	0x12

Response CMD_GET_CNF: Successfully read out the firmware version, for this example it is 0x000001 so "1.0.0" (with the parameter reverted to MSB first).

Start signal	Command	0x40	Length	Status	Parameter	CS
0x02	0x50		0x04 0x00	0x00	0x00 0x00 0x01	0x57

8.3 FS_MAC: Read the MAC address

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
3	FS_MAC	-	-	read	8

This setting contains the unique MAC address of the module.

8.3.1 Example 1

Request the MAC address of the module using CMD_GET_REQ with settings index 3

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x03	0x10

Response CMD_GET_CNF: Successfully read out the MAC address 0x55 0x93 0x19 0x6E 0x5B 0x87 0x01 0x38

Start signal	Command	0x40	Length	Status	Parameter	CS
0x02	0x50		0x09 0x00	0x00	0x55 0x93 0x19 0x6E 0x5B 0x87 0x01 0x38	0x0F

8.4 FS_BTMAC: Read the BLE conform MAC address

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
4	FS_BTMAC	-	-	read	6

This setting contains the BLE conform MAC address of the module. The FS_BTMAC is intro-duced and used to find the respective device on the RF-interface. It consists of the company ID 0x0018DA followed by the FS_SerialNumber of the module. Please note that LSB is transmitted first in all commands.

8.4.1 Example 1

Request the Bluetooth®-conform MAC address of the module using CMD_GET_REQ with set-tings index 4

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x04	0x17

Response CMD_GET_CNF: Successfully read out the BLE conform MAC address 0x11 0x00 0x00 0xDA 0x18 0x00.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x07 0x00	0x00	0x11 0x00 0x00 0xDA 0x18 0x00	0x86

8.5 FS_SerialNumber: Read the serial number of the module

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
16	FS_SerialNumber	-	-	read	3

This setting contains the serial number of the module.

8.5.1 Example 1

Request the serial number of the module using CMD_GET_REQ with settings index 16

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x10	0x03

Response CMD_GET_CNF: Successfully read out the serial number, it is 0.0.11

Start signal	Command	0x40	Length	Status	Parameter	CS
0x02	0x50	0x04 0x00	0x00	0x11 0x00 0x00	0x57

8.6 RF_DeviceName: Modify the device name

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
2	`RF_DeviceName`	See description	"Proteus-III"	read/write	1-32

> **!** This parameter is using MSB first notation.

This parameter determines the name of the module, which is used in the advertising packets as well as in the Generic Access Profile (GAP). The permissible characters are in the range of 0x20 - 0x7E which are special characters (see ASCII table), alphabetic characters (a-z and A-Z), numbers (0-9) and whitespace.

> **!** The maximum size of the device name that fits into an advertising packet is 5 Bytes. Thus longer device names will be shortened to 5 Bytes and declared as "Shortened Local Name" in the advertising packet. The full device name is included in the GAP.

8.6.1 Example 1

Set the device name of the module to 0x4D 0x4F 0x44 0x20 0x31 = "MOD 1" using `CMD_SET_REQ` with settings index 2.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x06 0x00	0x02	0x4D 0x4F 0x44 0x20 0x31	0x40

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.6.2 Example 2

Request the device name of the module using `CMD_GET_REQ` with settings index 2:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x02	0x11

Response `CMD_GET_CNF`: Successfully read out the module as 0x41 0x32 0x37 0x32 0x31 = "A2721".

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x06 0x00	0x00	0x41 0x32 0x37 0x32 0x31	0x13

8.7 RF_StaticPasskey: Modify the static passkey

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
18	`RF_StaticPasskey`	See description	"123123"	read/write	6

This setting determines the static pass key of the peripheral device used for authentication. If the static pass key security mode is enabled by the peripheral, this key must be entered in the central device. In case of a Proteus-III central, the command to enter this pass key during connection setup is the `CMD_PASSKEY_REQ`.
The permissible characters are ranging from 0x30 to 0x39 (both included) which are ASCII numbers (0-9). This is due to the fact that mobile phones prefer numbers only for the passkey.

8.7.1 Example 1

Set the static pass key of the module to 0x31 0x32 0x33 0x34 0x35 0x36 = "123456" using `CMD_SET_REQ` with settings index 18

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x07 0x00	0x12	0x31 0x32 0x33 0x34 0x35 0x36	0x01

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.7.2 Example 2

Request the static pass key of the module using `CMD_GET_REQ` with settings index 18

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x12	0x01

Response `CMD_GET_CNF`:Successfully read out the key as 0x31 0x32 0x33 0x34 0x35 0x36 = "123456"

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x07 0x00	0x00	0x31 0x32 0x33 0x34 0x35 0x36	0x52

8.8 RF_SecFlags: Modify the security settings

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
12	`RF_SecFlags`	See description	0	read/write	1

This 8-Bit field configures security settings of the module. Chapter `5.4` contains further information about secure connections.

> When connecting from a Proteus-III to another Proteus-III, be sure that the same security mode is used.

> When connecting from a foreign device to a Proteus-III, the peripheral (Proteus-III) determines the minimum security level needed for communication. So configure the `RF_SecFlags` of the peripheral to set the desired security level.

> When updating this user setting (like enabling bonding or changing the security mode) please remove all existing bonding data using the command `CMD_DELETEBONDS_REQ`.

Bit no.	Description			
2 : 0	Security mode configuration. Depending on its value, different modes are chosen when setting up a secure connection. In firmware version 2.1.0 and newer the peripheral decides which is the minimum security level to access its data.			
		0x0	No security	Data is transmitted without authentication and encryption.
		0x2	Just works level 1.2	Each time a connection is established, new random keys are exchanged in advance to use them for data encryption. This mode uses the "just works" method.
		0x3	Static pass key level 1.3	For authentication, the `RF_StaticPasskey` is used. If the peripheral uses this method, the central device must enter the correct passkey to finalize the connection.
		0x4	Lesc numeric comparison level 1.4	For authentication, the peripheral and central device output a pass key. The central and peripheral device must confirm the pass key to finalize the connection in case both keys coincide. Otherwise reject it to cancel the connection. The pass key is generated using the LESC method (low energy elliptic curve).
		0x5	Lesc pass key level 1.4	For authentication, the peripheral outputs a passkey. The central device must enter this pass key to finalize the connection. The pass key is generated using the LESC method (low energy elliptic curve).
		others		Reserved
3	`SECFLAGS_BONDING_ENABLE`: If this Bit is set, bonding is enabled when using one of the pairing methods. Bonding data of up to 32 devices will be stored in the flash.			
7 : 4	Reserved			

Table 16: Security configuration flags

> Since the security modes "Lesc pass key" and "Lesc numeric comparison" output the LESC key on the UART, it is essential to use the command mode. In other words, the peripheral only mode does not support the lesc security modes.

8.8.1 Example 1

Set the security flags to 0x0B, to use the static passkey pairing and with bonding enabled, using `CMD_SET_REQ` with settings index 12

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x0C	0x0B	0x16

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.8.2 Example 2

Request the security flags of the module using `CMD_GET_REQ` with settings index 12

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x0C	0x1F

Response `CMD_GET_CNF`: Successfully read out the value 2, which means that the just works pairing mode is enabled.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x02	0x52

8.9 RF_SecFlagsPerOnly: Modify the security settings (Peripheral only mode)

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
44	RF_SecFlagsPerOnly	See description	11	read/write	1

This user setting determines the security setting of the peripheral only mode. Please refer to the setting RF_SecFlags for more details.

> **STOP** Since the security modes "Lesc pass key" and "Lesc numeric comparison" output the LESC key on the UART, it is essential to use the command mode. In other words, the peripheral only mode does not support the lesc security modes.

8.9.1 Example 1

Set the security flags to 0x02 to use the just works pairing, using CMD_SET_REQ with settings index 44

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x2C	0x02	0x3F

Response CMD_SET_CNF: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.9.2 Example 2

Request the security flags of the module using CMD_GET_REQ with settings index 44

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x2C	0x3F

Response CMD_GET_CNF: Successfully read out the value 2, which means that the just works pairing mode is enabled.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x02	0x52

8.10 RF_ScanFlags: Modify the scan behavior

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
13	`RF_ScanFlags`	See description	0	read/write	1

This 8-Bit field configures the scan behavior of the module. To use multiple settings, add the Bit numbers and choose the result as value for `RF_ScanFlags`.

Bit no.	Description
0	If this Bit is set, an active scan is performed when using `CMD_SCANSTART_REQ`. In this case, after receiving an advertising packet a scan request is send to the advertising module that returns a scan response containing additional information. For the communication of Proteus-III modules, active scanning is only needed when using Beacons. In this case, it is enabled automatically by the firmware. Please note that active scanning increases the current consumption.
15 : 1	Reserved

Table 17: Scan configuration flags

8.10.1 Example 1

Set the scan flags to 0x01 to enable active scanning using `CMD_SET_REQ` with settings index 13

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x0D	0x01	0x1D

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.10.2 Example 2

Request the scan flags of the module using `CMD_GET_REQ` with settings index 13

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x0D	0x1E

Response `CMD_GET_CNF`: Successfully read out the value 0, which means that active scan is disabled.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x00	0x50

8.11 RF_BeaconFlags: Interprete the advertising data

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
14	`RF_BeaconFlags`	See description	0	read/write	1

This field configures the reception of Beacons.

Value	Description
0x01	Receive all Beacons from Proteus-III devices in range. Each received packet is interpreted and output by the UART via a `CMD_BEACON_IND` message.
0x02	All received advertising data / beacons is output in raw format via a `CMD_BEACON_RSP` message.
0x03	Same as '0x01' plus additional filter. This filter discards redundant packets that contain the same content.
0x04	A `CMD_RSSI_IND` message is output each time when an advertising packet with AMBER SPP-like UUID is received. This feature can be used to realize a position sensing application, since the `CMD_RSSI_IND` contains the current TX power level and the current RSSI value besides the `FS_BTMAC` of the sending device.
Others	Reserved.

Table 18: Beacon configuration flags

The internal database of the module may host the advertising data of 25 different devices. If the data base is full, the oldest entry is removed.

To decrease the work load of the receiving module, use a sufficiently high UART baud rate at the receiving device and slow advertising intervals at the sending devices.

If the reception of beacons is configured, active scanning is performed which increases the current consumption.

8.11.1 Example 1

Set the Beacon flags to 0x04 using `CMD_SET_REQ` with settings index 14. Thus when an advertising packet with AMBER SPP-like UUID is received, a `CMD_RSSI_IND` message is printed.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x0E	0x04	0x1B

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.11.2 Example 2

Request the Beacon flags of the module using `CMD_GET_REQ` with settings index 14

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x0E	0x1D

Response `CMD_GET_CNF`: Successfully read out the value 3, which means that the reception of Beacons is enabled and double packets are filtered by the module.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x03	0x53

8.12 RF_AdvertisingTimeout: Modify the advertising timeout

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
7	`RF_AdvertisingTimeout`	0 (infinite), 1 - 650	0	read/write	2

This parameter defines the time in seconds after which the advertising of the module stops. If no peer connects before this timeout, advertising stops and the module goes to system-off mode. If the `RF_AdvertisingTimeout` is set to 0, the module advertises infinitely.

> To ensure that the module sends a sufficient amount of advertising packets per `RF_AdvertisingTimeout`, please also check the `RF_ScanTiming` parameter, which defines the frequency of advertising packets.

8.12.1 Example 1

Set the advertising timeout parameter to 0x00 0xB4 (180s) using `CMD_SET_REQ` with settings index 7.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x07	0xB4 0x00	0xA3

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command	0x40	Length	Status	CS
0x02	0x51		0x01 0x00	0x00	0x52

8.12.2 Example 2

Request the advertising timeout of the module using `CMD_GET_REQ` with settings index 7

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x07	0x14

Response `CMD_GET_CNF`: Successfully read out the value 0x00 0x00 = 0s, which indicates indefinite advertising.

Start signal	Command	0x40	Length	Status	Parameter	CS
0x02	0x50		0x03 0x00	0x00	0x00 0x00	0x51

8.13 RF_AdvertisingFlags: Configure the advertising packet

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
29	RF_AdvertisingFlags	0,1	0	read/write	1

The user setting RF_AdvertisingFlags specifies the content of the advertising packet.

Bit no.	Description	
0	Define the content of the advertising packet	
	0x0	Command mode: Advertising packet contains 5 bytes device name RF_DeviceName and the TX power
		PeripheralOnly mode: Advertising packet contains only 8 bytes device name generated by the FS_BTMAC (A-123456 in case the FS_BTMAC is 0x0018DA123456)
	0x1	All modes: Advertising packet contains only 8 bytes device name RF_DeviceName
	others	Reserved.
7 : 1	Reserved.	

Table 19: Advertising packet configuration flags

> **STOP** To use the beacon feature (see chapter 5.5) the TX power must be included in the advertising packet.

8.13.1 Example 1

Set the advertising flags to 1 such that command mode and PeripheralOnly mode use the same advertising packet content using the CMD_SET_REQ with settings index 29

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x1D	0x01	0x0D

Response CMD_SET_CNF: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.13.2 Example 2

Request the RF_AdvertisingFlags using CMD_GET_REQ with settings index 29:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x1D	0x0E

Response CMD_GET_CNF: Successfully read out the value 0x00.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x00	0x50

8.14 RF_ScanFactor: Modify the scan factor

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
10	`RF_ScanFactor`	1 - 10	2	read/write	1

This parameter defines the factor between the scan window and the scan interval. See `RF_ScanTiming` for more information.

Example: Let's assume that the scan window is 50ms, the `RF_ScanFactor` is 3, then the module scans for 50ms on a fixed channel, enters a suspend mode (system-on mode) for 100ms (3×50ms - 50ms), switches the channel, again scans for 50ms and so on. The larger the `RF_ScanFactor`, the less time the module scans and thus the less power is consumed, but also the more difficult it is to detect other BLE devices on air.

8.14.1 Example 1

Set the scan factor to 0x03 using `CMD_SET_REQ` with settings index 10.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x0A	0x03	0x18

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.14.2 Example 2

Request the scan factor of the module using `CMD_GET_REQ` with settings index 10

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x0A	0x19

Response `CMD_GET_CNF`: Successfully read out the value 2.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x02	0x52

8.15 RF_ScanTiming: Modify the scan timing

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
9	`RF_ScanTiming`	0 - 11	1	read/write	1

The `RF_ScanTiming` enables the possibility to configure the timing behavior of the module's RF interface during advertising and scanning state. Using this parameter several predefined configurations can be chosen, which include timing parameters, such as the frequency of advertising packets and the length of a scan window.

The choice of the `RF_ScanTiming` primarily affects the latency of device detection on air as well as the current consumption. The lower the `RF_ScanTiming`, the faster the modules can find each other for communication, but also the more power will be consumed.

`RF_ScanTiming`	0	1	2	3	4	5	6	7	8[1]	9[1]	10[1]	11[1]
Advertising interval [ms]	20	40	60	80	100	120	250	500	1000	2000	5000	10240
Scan window [ms]	25	50	80	100	120	150	312	600	1250	2500	6250	10240
Scan interval [ms]	Defined by the `RF_ScanFactor`.											
Connection setup timeout [s]	1	2	2	2	2	2	2	4	5	10	20	35
Current consumption	High	. . .										Low

Further information:

- In `ACTION_SCANNING` mode, the scan interval defines the time after which the module switches channel to detect other BLE devices in range. See also `RF_ScanFactor`.

- In `ACTION_SCANNING` mode, the scan window defines the section of the scan interval, where the module is scanning. During the remaining time, the module enters a suspend mode (system-on mode). See also `RF_ScanFactor`.

- In `ACTION_IDLE` mode, the advertising interval defines the time after which the module periodically sends its advertising packet. In between, the module enters a suspend mode (system-on mode).

- The connection setup timeout defines the time after which a connection request has to be answered by the peripheral.

---

[1]Mainly suitable for transmitting data using Beacons without consuming much energy.

8.15.1 Example 1

Set the scan timing parameter to 0x00 using `CMD_SET_REQ`with settings index 9.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x09	0x00	0x18

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.15.2 Example 2

Request the scan timing parameter of the module using `CMD_GET_REQ` with settings index 9

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x09	0x1A

Response `CMD_GET_CNF`: Successfully read out the value 4.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x04	0x54

8.16 RF_ConnectionTiming: Modify the connection timing

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
8	RF_ConnectionTiming	0 - 8	2	read/write	1

The RF_ConnectionTiming enables the possibility to configure the timing behavior of the module's RF interface during an established connection. Using this parameter several pre-defined configurations can be chosen, which include the minimum and maximum connection interval, as well as the connection supervision timeout.

The choice of the RF_ConnectionTiming primarily determines how rapidly the connection is established and data is transmitted. The lower the RF_ConnectionTiming, the more frequently the connected devices communicate with each other and thus, the more power is consumed.

RF_ConnectionTiming	0	1	2	3	4	5	6	7	8
Minimum connection interval [ms]	8	8	20	30	50	100	200	750	2000
Maximum connection interval [ms]	8	30	75	100	250	500	1000	2250	4000
Connection supervision timeout [s]	4	4	4	4	4	8	8	15	25
Maximum throughput[1] [kB/s]	Up to 10.4	Up to 8	Up to 3.2	Up to 1	-				
Current consumption	High	. . .							Low

> The throughput can be increased by using the so called high throughput mode.

Further information:

- The minimum and maximum connection interval parameters specify the borders of the connection interval as determined in the negotiation procedure between the central and the peripheral during connection setup. The connection interval defines the frequency of communication during connection setup and data transmission. If a Proteus-III module A (central) connects to a Proteus-III module B (peripheral), the connection interval settings of the central are used for connection setup. If both modules have different connection interval settings the peripheral requests the central to accept the peripheral's settings after 5s. The central accepts these settings, and thus the peripheral's

---

[1]Measured with 921600 Baud UART baud rate, enabled flow control and payload size of 243Bytes between two Proteus-III modules in command mode. Please note that data transmission to/from mobile phones does not achieve this speed due to old mobile phone chips and BLE stacks.

connection interval is used.

If now another BLE device (e.g. a smart phone) connects as central to a Proteus-III module (peripheral) and the connection interval settings do not coincide, the Proteus-III requests the smart phone to accept its settings after 5s. If the cell phone does not accept the settings, it will be requested a further 3 times with a delay of 10s. If the peripheral's settings request have been rejected in all cases the connection will be shut down. If the smart phone itself requests to update the connection interval of the Proteus-III, the module accepts the request. Reversely, if a Proteus-III (central) connects to another BLE device (peripheral) and the connection interval settings do not coincide, the Proteus-III accepts all requests of the peripheral to update the connection parameter settings.

- The connection supervision timeout defines the time after which an already established connection is considered as lost, when no further communication has occurred.

> **STOP** Please ensure that all members (Proteus-III, cell phones and other BLE devices) of a network use the same connection timing parameters to avoid connection problems and changes of the connection interval during an opened connection.

> **STOP** Please check the minimum connection interval that is supported by iOS. Former iOS devices do not support connection intervals shorter than 30ms!

8.16.1 Example 1

Set the connection timing parameter to 0x00 using `CMD_SET_REQ` with settings index 8.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x08	0x00	0x19

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command	0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.16.2 Example 2

Request the connection timing parameter of the module using `CMD_GET_REQ` with settings index 8

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x08	0x1B

Response CMD_GET_CNF: Successfully read out the value 1.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x01	0x51

8.17 RF_TXPower: Modify the output power

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
17	RF_TXPower	See description	8	read/write	1

This setting determines the output power in dBm of the module. The value has to be entered in hexadecimal and as two's complement. The permissible values are listed in the following table.

Permissible values								
Decimal [dBm]	-40	-30	-20	-16	-12	-8	-4	0
Two's complement, hexadecimal	0xD8	0xE2	0xEC	0xF0	0xF4	0xF8	0xFC	0x00

Decimal [dBm]	2	3	4	5	6	7	8	
Two's complement, hexadecimal	0x02	0x03	0x04	0x05	0x06	0x07	0x08	

8.17.1 Example 1

Set the output power of the module to -8 dBm, which is 0xF8 in two's complement notation, using CMD_SET_REQ with settings index 17

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x11	0xF8	0xF8

Response CMD_SET_CNF: Successfully modified the setting.

Start signal	Command	0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.17.2 Example 2

Request the output power of the module using CMD_GET_REQ with settings index 17

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x11	0x02

Response CMD_GET_CNF: Successfully read out the value 0x04 = 4dBm

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x04	0x54

8.18 RF_SPPBaseUUID: Configure the SPP base UUID

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
26	`RF_SPPBaseUUID`	See description	0x6E400000C352 11E5953D0002 A5D5C51B	read/write	16

The AMBER SPP-like profile consists of the 16 Bytes base UUID 0x6E40xxxx-C352-11E5-953D-0002A5D5C51B plus the 2 Bytes UUIDs for the underlying characteristics:

Characteristic	2 Bytes UUID
Primary service	0x0001
TX_CHARACTERISTIC	0x0002
RX_CHARACTERISTIC	0x0003

With this the TX characteristic can be identified by the resulting 16 Bytes UUID 0x6E400002-C352-11E5-953D-0002A5D5C51B on the radio. With help of the `RF_SPPBaseUUID` parameter we have to possibility to update the 16Byte base UUID of the AMBER SPP-like profile.

8.18.1 Example 1

Set the base UUID to 0xEFEEEDEC-EBEA-E9E8-E7E6-E5E4E3E2E1E0 using `CMD_SET_REQ` with settings index 26

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x11 0x00	0x1A	0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF	0x18

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command	0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.18.2 Example 2

Request the base UUID of the module using `CMD_GET_REQ`:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x1A	0x09

Response `CMD_GET_CNF`: Successfully read out the value 0x6E400000-C352-11E5-953D-0002A5D5C51B.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x11 0x00	0x00	0x1B 0xC5 0xD5 0xA5 0x02 0x00 0x3D 0x95 0xE5 0x11 0x52 0xC3 0x00 0x00 0x40 0x6E	0x0C

8.19 RF_Appearance: Configure the appearance of the device

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
25	`RF_Appearance`	0-65535	0	read/write	2

The user setting `RF_Appearance` specifies the appearance of the Bluetooth® devices. It's a 2 Bytes field defined by the Bluetooth® SIG. Please check the Bluetooth® Core Specification:Core Specification Supplement, Part A, section 1.12 for permissible values.

8.19.1 Example 1

Set the appearance to "Generic computer" (0x0080) using `CMD_SET_REQ` with settings index 25

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x19	0x80 0x00	0x89

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.19.2 Example 2

Request the `RF_Appearance` using `CMD_GET_REQ`:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x19	0x0A

Response `CMD_GET_CNF`: Successfully read out the value 0x0000, meaning that the appearance is unknown.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x03 0x00	0x00	0x00 0x00	0x51

8.20 UART_ConfigIndex: Modify the UART speed

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
11	UART_ConfigIndex	See description	22	read/write	1

This parameter defines the baud rate used by the module's UART. The permissible values are listed in the following table. If flow control is enabled the pins */RTS* and */CTS* are used.

UART_ConfigIndex	Rate [Baud]	Real rate [Baud]	Flow control	Parity
0	1200	1205	no	none
1	1200	1205	yes	none
2	2400	2396	no	none
3	2400	2396	yes	none
4	4800	4808	no	none
5	4800	4808	yes	none
6	9600	9598	no	none
7	9600	9598	yes	none
8	14400	14414	no	none
9	14400	14414	yes	none
10	19200	19208	no	none
11	19200	19208	yes	none
12	28800	28829	no	none
13	28800	28829	yes	none
14	38400	38462	no	none
15	38400	38462	yes	none
16	56000	55944	no	none
17	56000	55944	yes	none
18	57600	57762	no	none
19	57600	57762	yes	none
20	76800	76923	no	none
21	76800	76923	yes	none
22	115200	115942	no	none
23	115200	115942	yes	none
25	230400	231884	yes	none
27	250000	250000	yes	none
29	460800	470588	yes	none
31	921600	941176	yes	none

UART_ConfigIndex	Rate [Baud]	Real rate [Baud]	Flow control	Parity
33	1000000	1000000	yes	none
64	1200	1205	no	even
65	1200	1205	yes	even
66	2400	2396	no	even
67	2400	2396	yes	even
68	4800	4808	no	even
69	4800	4808	yes	even
70	9600	9598	no	even
71	9600	9598	yes	even
72	14400	14414	no	even
73	14400	14414	yes	even
74	19200	19208	no	even
75	19200	19208	yes	even
76	28800	28829	no	even
77	28800	28829	yes	even
78	38400	38462	no	even
79	38400	38462	yes	even
80	56000	55944	no	even
81	56000	55944	yes	even
82	57600	57762	no	even
83	57600	57762	yes	even
84	76800	76923	no	even
85	76800	76923	yes	even
86	115200	115942	no	even
87	115200	115942	yes	even
89	230400	231884	yes	even
91	250000	250000	yes	even
93	460800	470588	yes	even
95	921600	941176	yes	even
97	1000000	1000000	yes	even

After changing the baud rate using the `CMD_SET_REQ` the module restarts using the new baud rate. Therefore don't forget to update the baud rate of the connected host to be able to further use the module's UART.

Please note that due to the HF-activity of the chip, single Bytes on the UART can get lost, when using a very fast UART data rate. In case of corrupted UART communication the module cannot interpret the sent request and thus does not return a confirmation.

8.20.1 Example 1

Set the baud rate index to 0x1F (921600 Baud with flow control and parity none) using `CMD_SET_REQ` with settings index 11

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x0B	0x1F	0x05

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.20.2 Example 2

Request the baud rate index of the module using `CMD_GET_REQ` with settings index 11

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x0B	0x18

Response `CMD_GET_CNF`: Successfully read out the value 0x16, which equals 115200 Baud without flow control and parity none.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x16	0x46

8.21 CFG_Flags: Configure the module

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
28	CFG_Flags	See description	0	read/write	2

The user setting CFG_Flags specifies various module features.

Bit no.	Name	Description
0	High-throughput mode	Set this Bit to 1 to enable the high-throughput mode.
1	Long range connection mode	Set this Bit to 1 to enable the mode using the LE Coded mode during connection setup.
2	GPIO remote config.	Set this Bit to 1 to block the GPIO configuration via remote device.
3-7	Reserved	Reserved.

The high-throughput mode and its usage is described in the corresponding Proteus-III application note.

8.21.1 Example 1

Enable the high-throughput mode using CMD_SET_REQ with settings index 28

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x1C	0x01 0x00	0x0D

Response CMD_SET_CNF: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.21.2 Example 2

Request the CFG_Flags using CMD_GET_REQ:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x1C	0x0F

Response CMD_GET_CNF: Successfully read out the value 0x00, meaning that all of the specified features are disabled.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x03 0x00	0x00	0x00 0x00	0x51

8.22 DIS_ManufacturerName: Configure the manufacturer name

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
20	DIS_ManufacturerName	See description	"Default"	read/write	1-64

The user setting `DIS_ManufacturerName` specifies the content of the manufacturer name field of the Device Information Service. The permissible characters are in the range of 0x20 - 0x7E which are special characters (see ASCII table), alphabetic characters (a-z and A-Z), numbers (0-9) and whitespace.

> To add the content of the `DIS_ManufacturerName` to the DIS profile, please set the corresponding Bit in the `DIS_Flags`.

8.22.1 Example 1

Set the manufacturer name to "Manufacturer1" using `CMD_SET_REQ` with settings index 20

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x0E 0x00	0x14	0x4D 0x61 0x6E 0x75 0x66 0x61 0x63 0x74 0x75 0x72 0x65 0x72 0x31	0x0F

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.22.2 Example 2

Request the manufacturer name of the DIS profile using `CMD_GET_REQ`:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x14	0x07

Response `CMD_GET_CNF`: Successfully read out the value "Default".

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x08 0x00	0x00	0x44 0x65 0x66 0x61 0x75 0x6C 0x74	0x11

8.23 DIS_ModelNumber: Configure the model number

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
21	DIS_ModelNumber	See description	"Default"	read/write	1-64

The user setting `DIS_ModelNumber` specifies the content of the model number field of the Device Information Service. The permissible characters are in the range of 0x20 - 0x7E which are special characters (see ASCII table), alphabetic characters (a-z and A-Z), numbers (0-9) and whitespace.

> ⊘ To add the content of the `DIS_ModelNumber` to the DIS profile, please set the corresponding Bit in the `DIS_Flags`.

8.23.1 Example 1

Set the model number to "Model1" using `CMD_SET_REQ` with settings index 21

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x07 0x00	0x15	0x4D 0x6F 0x64 0x65 0x6C 0x31	0x7F

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.23.2 Example 2

Request the model number of the DIS profile using `CMD_GET_REQ`:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x15	0x06

Response `CMD_GET_CNF`: Successfully read out the value "Default".

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x08 0x00	0x00	0x44 0x65 0x66 0x61 0x75 0x6C 0x74	0x11

8.24 DIS_SerialNumber: Configure the serial number

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
22	DIS_SerialNumber	See description	"Default"	read/write	1-64

The user setting `DIS_SerialNumber` specifies the content of the serial number field of the Device Information Service. The permissible characters are in the range of 0x20 - 0x7E which are special characters (see ASCII table), alphabetic characters (a-z and A-Z), numbers (0-9) and whitespace.

> ⓘ To add the content of the `DIS_SerialNumber` to the DIS profile, please set the corresponding Bit in the `DIS_Flags`.

8.24.1 Example 1

Set the serial number to "1.2.3" using `CMD_SET_REQ` with settings index 22

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x06 0x00	0x16	0x31 0x2E 0x32 0x2E 0x33	0x33

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.24.2 Example 2

Request the serial number of the DIS profile using `CMD_GET_REQ`:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x16	0x05

Response `CMD_GET_CNF`: Successfully read out the value "Default".

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x08 0x00	0x00	0x44 0x65 0x66 0x61 0x75 0x6C 0x74	0x11

8.25 DIS_HWVersion: Configure the HW version

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
23	DIS_HWVersion	See description	"Default"	read/write	1-16

The user setting `DIS_HWVersion` specifies the content of the hardware version field of the Device Information Service. The permissible characters are in the range of 0x20 - 0x7E which are special characters (see ASCII table), alphabetic characters (a-z and A-Z), numbers (0-9) and whitespace.

> To add the content of the `DIS_HWVersion` to the DIS profile, please set the corresponding Bit in the `DIS_Flags`.

8.25.1 Example 1

Set the hardware version to "1.2.3" using `CMD_SET_REQ` with settings index 23

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x06 0x00	0x17	0x31 0x2E 0x32 0x2E 0x33	0x32

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command | 0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.25.2 Example 2

Request the hardware version of the DIS profile using `CMD_GET_REQ`:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x17	0x04

Response `CMD_GET_CNF`: Successfully read out the value "Default".

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x08 0x00	0x00	0x44 0x65 0x66 0x61 0x75 0x6C 0x74	0x11

8.26 DIS_SWVersion: Configure the SW version

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
24	DIS_SWVersion	See description	"Default"	read/write	1-16

The user setting `DIS_SWVersion` specifies the content of the software version field of the Device Information Service. The permissible characters are in the range of 0x20 - 0x7E which are special characters (see ASCII table), alphabetic characters (a-z and A-Z), numbers (0-9) and whitespace.

> To add the content of the `DIS_SWVersion` to the DIS profile, please set the corresponding Bit in the `DIS_Flags`.

8.26.1 Example 1

Set the software version to "1.2.3" using `CMD_SET_REQ` with settings index 24

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x06 0x00	0x18	0x31 0x2E 0x32 0x2E 0x33	0x3D

Response `CMD_SET_CNF`: Successfully modified the setting.

Start signal	Command	0x40	Length	Status	CS
0x02	0x51		0x01 0x00	0x00	0x52

8.26.2 Example 2

Request the software version of the DIS profile using `CMD_GET_REQ`:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x18	0x0B

Response `CMD_GET_CNF`: Successfully read out the value "Default".

Start signal	Command	0x40	Length	Status	Parameter	CS
0x02	0x50		0x08 0x00	0x00	0x44 0x65 0x66 0x61 0x75 0x6C 0x74	0x11

8.27 DIS_Flags: Configure the device information service

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
19	DIS_Flags	0-255	0	read/write	1

The user setting DIS_Flags specifies the content of the Device Information Service. To add a specific field, like DIS_ModelNumber to the Device Information Service, the corresponding Bit has to be set in the DIS_Flags.

Bit no.	Description
0	Set this Bit to 1 to add the DIS_ManufacturerName to the Device Information Service.
1	Set this Bit to 1 to add the DIS_ModelNumber to the Device Information Service.
2	Set this Bit to 1 to add the DIS_SerialNumber to the Device Information Service.
3	Set this Bit to 1 to add the DIS_HWVersion to the Device Information Service.
4	Set this Bit to 1 to add the DIS_SWVersion to the Device Information Service.
5-7	Reserved.

8.27.1 Example 1

Add the manufacturer name and model number (Bit0|Bit1 = 0x03) to the Device Information Service using CMD_SET_REQ with settings index 19

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x13	0x03	0x01

Response CMD_SET_CNF: Successfully modified the setting.

Start signal	Command	0x40	Length	Status	CS
0x02	0x51	0x01 0x00	0x00	0x52

8.27.2 Example 2

Request the DIS_Flags using CMD_GET_REQ:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x13	0x00

Response CMD_GET_CNF: Successfully read out the value 0x00, meaning that the Device Information Service is disabled, since no field was added.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x02 0x00	0x00	0x00	0x50

Settings index	Designation	Summary	Permissible values	Default value	Permission	Number of Bytes
1	FS_FWVersion	Version of the firmware	-	-	read	3
2	RF_DeviceName	Name of the module	See de-scription	"A2721"	read / write	1-32
3	FS_MAC	MAC address of the module	-	-	read	6
4	FS_BTMAC	BLE conform MAC address of the module	-	-	read	6
7	RF_AdvertisingTimeout	Time [s] after advertising stops. LSB first	0 (infinite), 1 - 65535	0	read / write	2
8	RF_ConnectionTiming	Module connection timing configuration	0 - 8	2	read / write	1
9	RF_ScanTiming	Module advertising and scanning timing configuration	0 - 11	1	read / write	1
10	RF_ScanFactor	Factor between scan interval and scan window	1 - 10	2	read / write	1
11	UART_ConfigIndex	Baud rate of the UART	See de-scription	22	read / write	1
12	RF_SecFlags	Security settings of the module	See de-scription	0	read / write	1
13	RF_ScanFlags	Scan settings of the module	See de-scription	0	read / write	1
14	RF_BeaconFlags	Beacon settings of the module	See de-scription	0	read / write	1
15	FS_DeviceInfo	Information about the chip	-	-	read	12
16	FS_SerialNumber	Serial number of the module	-	-	read	3
17	RF_TXPower	Output power [dBm] Two's complement	See de-scription	8	read / write	1

Table 20: Table of settings (Part 1)

Settings index	Designation	Summary	Permissible values	Default value	Permission	Number of Bytes
18	RF_StaticPasskey	6 digit pass key	See de-scription	"123123"	read / write	6
19	DIS_Flags	Flags for the DIS	0 - 255	0	read / write	1
20	DIS_ ManufacturerName	Manufacturer name field of the DIS	See de-scription	"Default"	read / write	1-64
21	DIS_ModelNumber	Model number field of the DIS	See de-scription	"Default"	read / write	1-64
22	DIS_SerialNumber	Serial number field of the DIS	See de-scription	"Default"	read / write	1-64
23	DIS_HWVersion	HW version field of the DIS	See de-scription	"Default"	read / write	1-16
24	DIS_SWVersion	SW version field of the DIS	See de-scription	"Default"	read / write	1-16
25	RF_Appearance	Appearance	0-65535	0	read / write	2
26	RF_SPPBaseUUID	Base UUID of the AMBER SPP-like profile	See de-scription	See de-scrip-tion	read / write	16
28	CFG_Flags	CFG Flags	See de-scription	0	read / write	2
29	RF_ AdvertisingFlags	Advertising Flags	0,1	0	read / write	1
44	RF_SecFlagsPerOnly	Security settings of the module (peripheral only mode only)	See de-scription	11	read / write	1

Table 21: Table of settings (Part 2)

9 Timing parameters

9.1 Reset and sleep

After power-up, resetting the module or waking the module from sleep a `CMD_GETSTATE_CNF` is sent to the serial interface as soon as the module is ready for operation.

Description	Typ.	Unit
Ready after reset/sleep	77	ms

9.2 BLE timing parameters

The timing parameters for sending advertising packets or scanning are determined by the user settings `RF_ScanTiming`, `RF_ScanFactor` and `RF_AdvertisingTimeout`. Using these settings, the advertising interval, the advertising timeout, the scan interval and the scan window can be configured. Furthermore, the user setting `RF_ConnectionTiming` allows to configure the timing parameters used during connection setup and connection retention, as well as the connection interval and the connection supervision timeout.

9.3 Connection establishment

The time needed to establish a connection sums up as the time needed to detect the selected peripheral on air and the time needed for connection parameter negotiation and service discovery.

Peripheral detection To establish a connection, the initiating device (central) waits for an advertising packet, which was sent by the peripheral to which it wants to connect to. As soon as such an advertising packet has been received, the central sends a connection request to the chosen peripheral. The time needed to receive this advertising packet strongly depends on the advertising interval of the peripheral as well as on the scan interval and scan window of the central (see `RF_ScanTiming`).

Connection parameter negotiation After the connection request has been sent the central and peripheral negotiate the timing and security parameters of the connection. To finish this procedure and discover the services of the peripheral several messages have to be sent, whereby only one is sent per connection interval (see `RF_ConnectionTiming`).

Connection type	Estimated number of exchanged messages	Negotiation time for a connection interval of 50ms
Unsecured connection	12-14	600-700ms
Secured connection using the pairing method	22-24	1100-1200ms
Secured connection to already bonded device	19-20	950-1000ms

Knowing the connection interval and the number of messages that will be sent, the time necessary to setup a connection can be estimated by multiplying the number of messages with the connection interval.

> In case the Device Information Service is enabled, the number of messages and thus the time consumption of the connection setup may be increased.

9.4 Connection based data transmission

After setting up a connection, data can be transmitted using the `CMD_DATA_REQ`. It buffers the data in the module and sends it with the next connection interval event. As soon as the data has been transmitted successfully, a `CMD_TXCOMPLETE_RSP` is returned by the UART. The time needed for this coincides with the connection interval that was negotiated during connection setup. The `RF_ConnectionTiming` parameter defines the minimum and maximum connection interval, which is supported by the module.

10 Peripheral only mode

The Proteus-III implements a new feature that allows the easy integration of the Proteus-III BLE module to an already existing host. The peripheral only mode offers a plug and play installation without previous configuration of the Proteus-III. It is tailored for easy communication with mobile BLE devices like smart phones.

The peripheral only mode is a special operation mode, that uses the user settings and the peripheral functions of the normal mode described in the previous chapters. It has to be enabled during the module start-up and contains the following key features:

- Peripheral only functions: The Proteus-III only contains the functions of a peripheral device. Thus, it is advertising until another BLE enabled device connects to it. In this case, the UART of the Proteus-III is enabled, the *LED_2* pin shows that the channel is open and bidirectional data transmission can start. As soon as the connection is closed, the UART is disabled again to save power. Since all central functions are no longer valid, the module cannot initiate any connection or run scans.

- Transparent UART interface: The serial interface of the Proteus-III is no longer driven by commands. This means, when the UART of the module is enabled (i.e. only when a channel is open, indicated by both LEDs active), data sent to the UART is transmitted by the Proteus-III to the connected BLE enabled device. On the other hand, all data received by radio is send from the Proteus-III to the connected host without additional header Bytes. The UART is only running, when a channel is open. Thus, power is saved during the advertising period. Depending on the configured connection interval, only one packet per interval is allowed to be transmitted. Since the commands of the command interface are no longer valid, a Proteus-III cannot be configured when running in peripheral only mode.

- Pairing: The default security mode is the static passkey pairing method (see `RF_SecFlagsPerOnly`), with the default key "123123". The bonding feature is enabled by default.

10.1 Reasons to use the peripheral only mode

The Proteus-III peripheral only mode equips custom applications with a BLE interface (to be accessible by other BLE devices) without installation effort.

To setup a connection to the Proteus-III in peripheral only mode the central device has to insert the Proteus-III's static passkey. As soon as the channel to a connected BLE central device is open, the *LED_2* pin switches on to signalize that data can be exchanged now. When the connection was shut down by the BLE central device, the *LED_2* pin switches off again.

Due to the transparent UART interface, data can be exchanged without additional headers. Furthermore, the peripheral only mode allows an energy efficient operation of the BLE interface, since the UART is only enabled when it is really used.

10.2 Restrictions

In peripheral only mode, it is not possible to use the security modes "Lesc pass key" and "Lesc numeric comparison", as these security features require to output the LESC key on the UART.

10.3 How to use the peripheral only mode

The peripheral only mode is enabled, when a high signal is present on the *MODE_1* pin during device start-up or reset.

No configuration of the module is needed for this operating mode. The module shall be set to factory settings if reconfigured before so it uses the default user settings. In this case, the UART uses 115200 Baud 8n1 and static passkey pairing is used as authentication method.

If a configuration of the module is still needed (e.g. when another UART baud rate needs to be chosen), the module has to be started in normal mode and the `CMD_SET_REQ` may be used to update the user settings.

It is permitted to modify any user setting to change the behavior of the peripheral only mode. Nevertheless, we recommend to update only the following parameters to run the device in factory state with minimal adaptions:

- `UART_ConfigIndex` (change the UART baud rate, default value "115200")

- `RF_StaticPasskey` (change the default static passkey, default value "123123")

- `RF_AdvertisingFlags` and `RF_DeviceName` (determine the content of the advertising packet)

10.4 More information

10.4.1 Radio

In factory state the following holds.

- In peripheral only mode a new 8-digit device name is automatically generated by the `FS_BTMAC`. In case of the `FS_BTMAC` equals 0x0018DA123456 the device name is "A-123456". This is a workaround for iOS, which does not allow access to the BTMAC for received BT frames.

- The content of the advertising packet was changed in peripheral only mode. The TX power information block was removed, as the device name was extended to 8 digits.

See also the user setting `RF_AdvertisingFlags` to adapt the content of the advertising packet.

10.4.2 UART

- The data sent to the UART is buffered in the Proteus-III up to a maximum payload of 1015 Bytes. When no new Byte was received for 20ms, the data will be transmitted by radio to the connected BLE device. If the data is larger than the MTU of the connection, the data is sent via radio in several packets, with one packet per connection interval.

- To enable a fast transmission of data packets a large MTU of the BLE connection is helpful. The Proteus-III supports up to 243 Bytes BLE packet payload (corresponding to a MTU of 247 Bytes), which may be negotiated by the central device (using a MTU request). If no MTU request is requested by the connecting central device the value of 19 Bytes payload per BLE packet and connection interval as given by the BT 4.0 standard is used (compatibility mode to BLE 4.0 devices).

- The pin *BUSY* can be used as a kind of flow control for the data transmission during the peripheral only mode. By default the pin level is LOW. As soon as the 20ms timeout was detected or too much data was received via UART, the pin switches to HIGH and data transmission starts via BLE. The pin switches LOW again, as soon as BLE data transmission has finished and the transmission of new data is feasible again. In case the pin is HIGH, no more data is accepted on the UART.
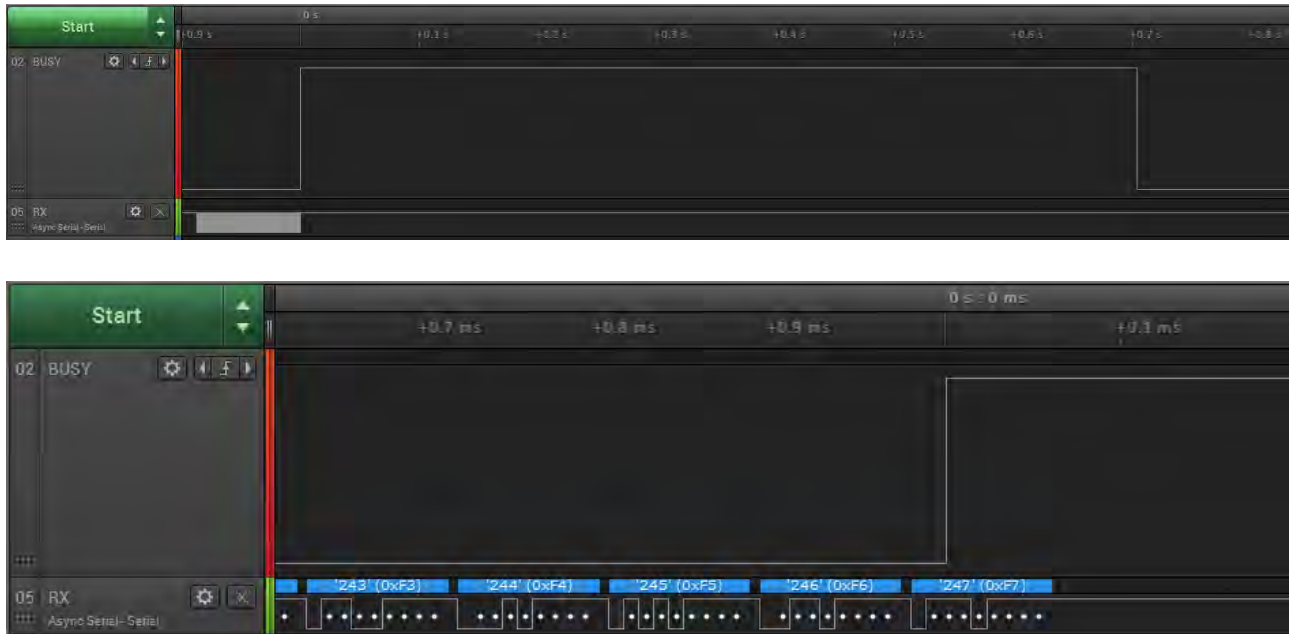
Figure 8: Switch of the *BUSY* pin when transmitting data

11 Remote GPIO control

The Proteus-III allows to control free GPIOs via remote access. Chapter `7.7` contains the description of the necessary commands.

To use the remote GPIO control feature of the Proteus-III, the GPIOs of interest must be configured first. This can be done in two ways. Either by the local host (see figure `9`), when the radio module is in idle mode (not connected via BLE), or via the connected remote device (see figure `10`).
In case of the local host, it must send a `CMD_GPIO_LOCAL_WRITECONFIG_REQ` command to the radio module via UART. In case of the remote device, it must setup a BLE connection to the remote device first and send a `CMD_GPIO_REMOTE_WRITECONFIG_REQ` command to the radio module via BLE afterwards.
The configuration is stored in flash memory, such that it is retained also after a device restart. It can be reset to default by using the `CMD_FACTORYRESET_REQ` command.
The configuration can be also read out using the respective commands, `CMD_GPIO_LOCAL_READCONFIG_REQ` via local host or `CMD_GPIO_REMOTE_READCONFIG_REQ` via remote device.

If the configuration has been done, the configured GPIOs can be controlled by the local host controller or by any remote device.

To control a GPIO via local host controller just send the respective commands, `CMD_GPIO_LOCAL_WRITE_REQ` for setting GPIO output values (see figure `13`), or `CMD_GPIO_LOCAL_READ_REQ` for reading GPIO values (see figure `14`). Each time the GPIOs are written via local host, the connected remote device is informed using a `CMD_GPIO_LOCAL_WRITE_IND` message.

To control a GPIO via remote device, first setup a BLE connection to the radio module and send the respective commands, `CMD_GPIO_REMOTE_WRITE_REQ` for setting GPIO output values (see figure `15`), or `CMD_GPIO_REMOTE_READ_REQ` for reading GPIO values (see figure `16`). Each time the GPIOs are written via remote connection, the local host is informed using a `CMD_GPIO_REMOTE_WRITE_IND` message.
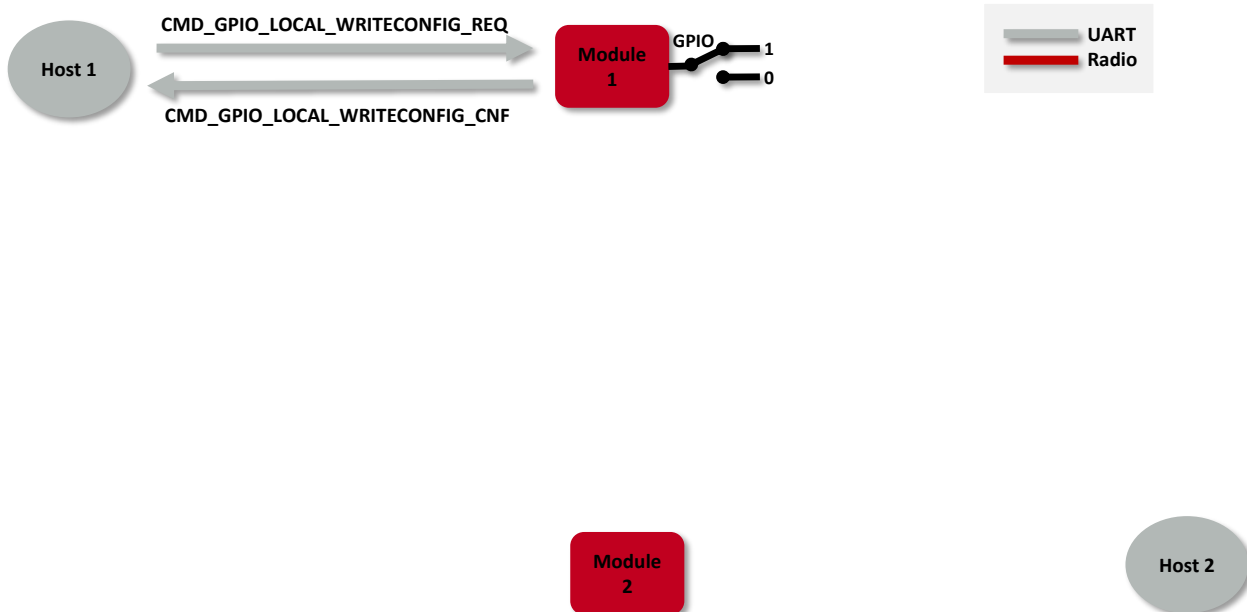
Figure 9: Configure the local GPIOs via local host
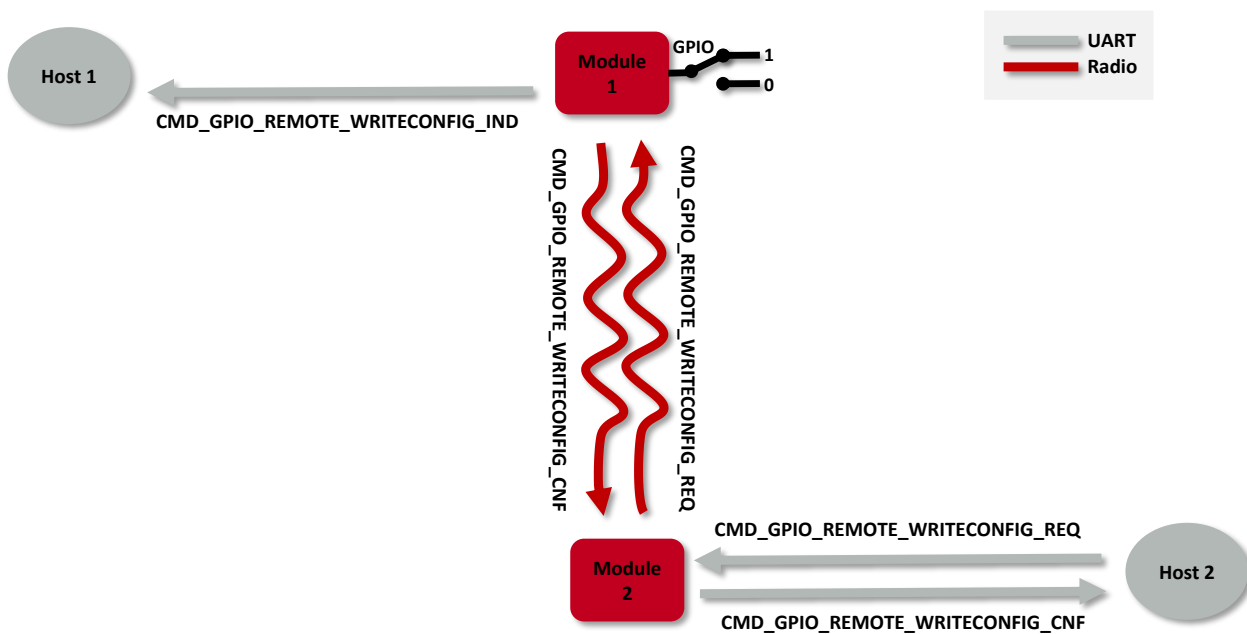
Figure 10: Configure the local GPIOs via remote device host
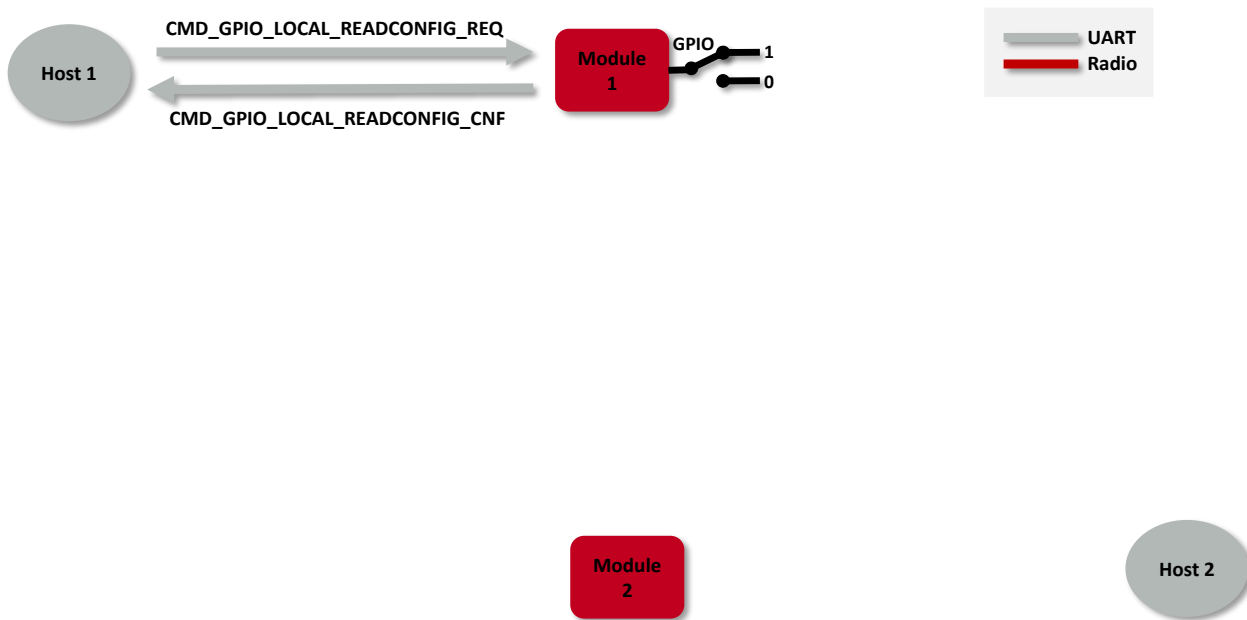
Figure 11: Read the configuration of the local GPIOs via local host
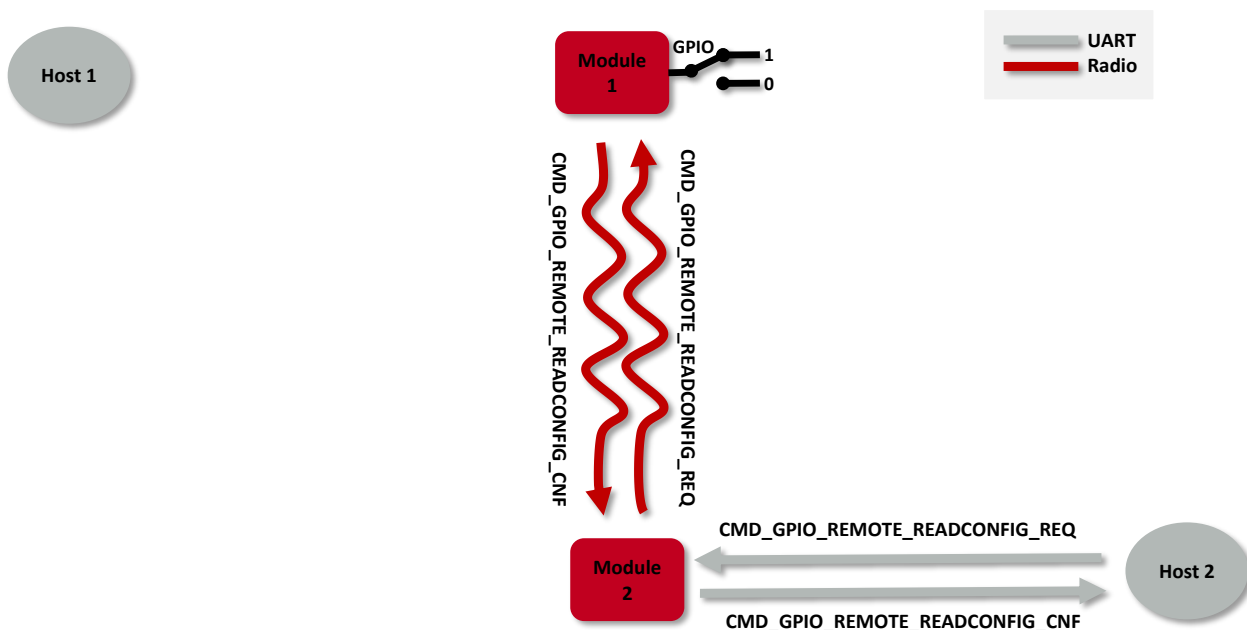
Figure 12: Read the configuration of the local GPIOs via remote device host
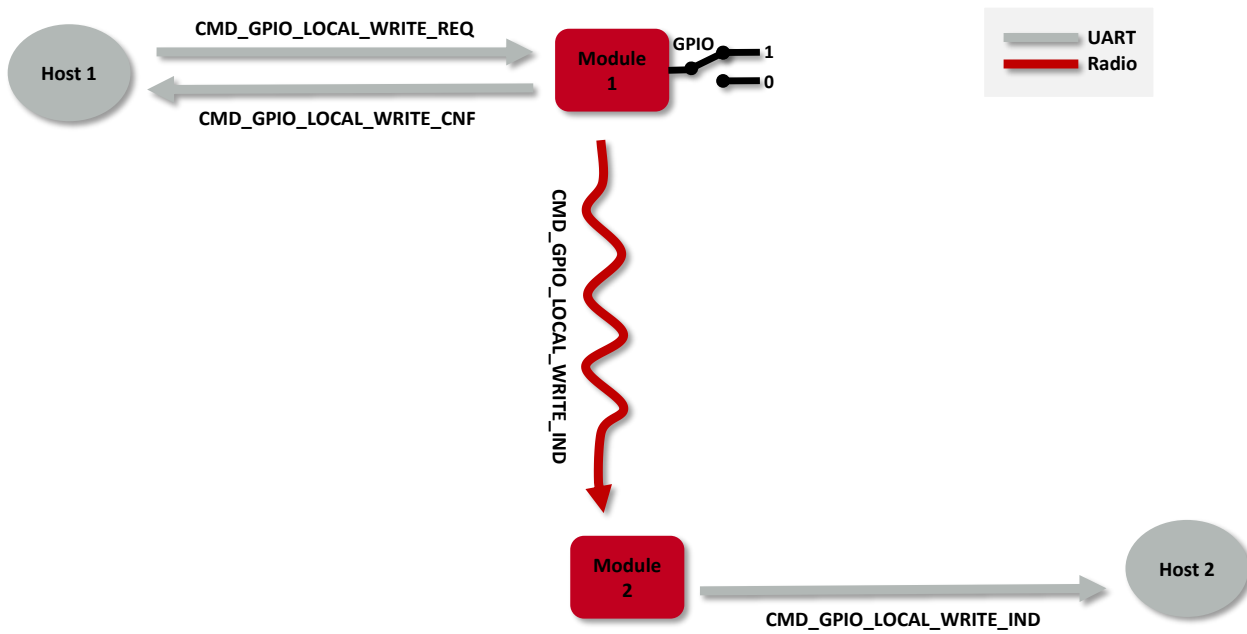
Figure 13: Set the output value of a GPIO via host controller
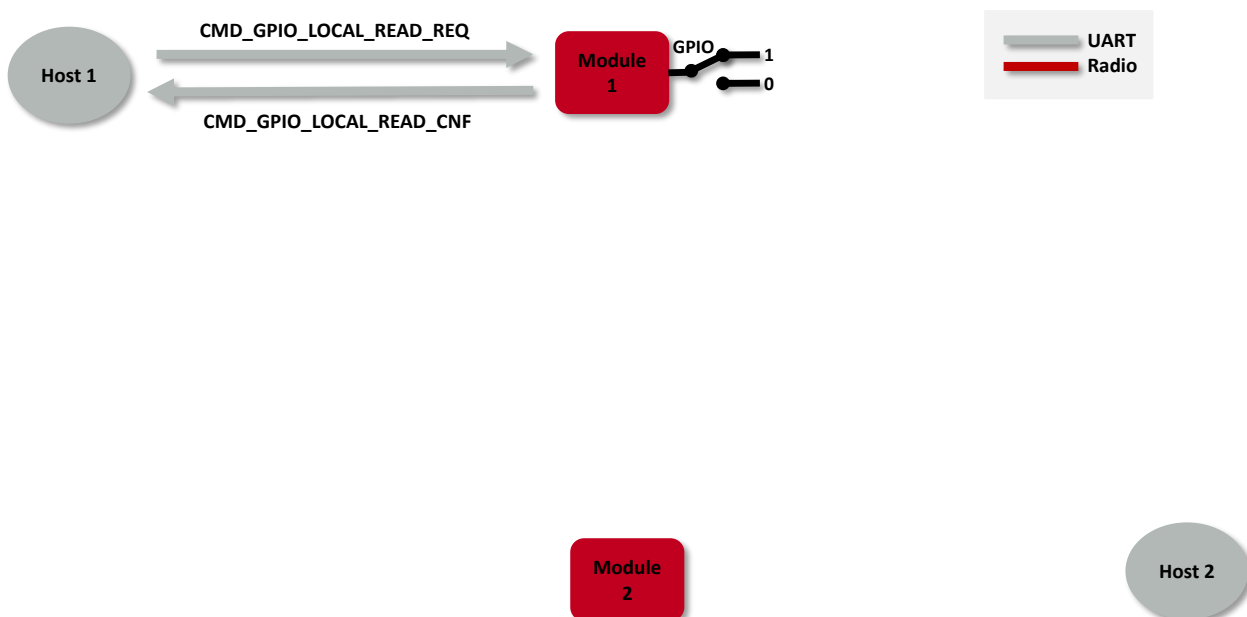
Figure 14: Read the input value of a GPIO via host controller
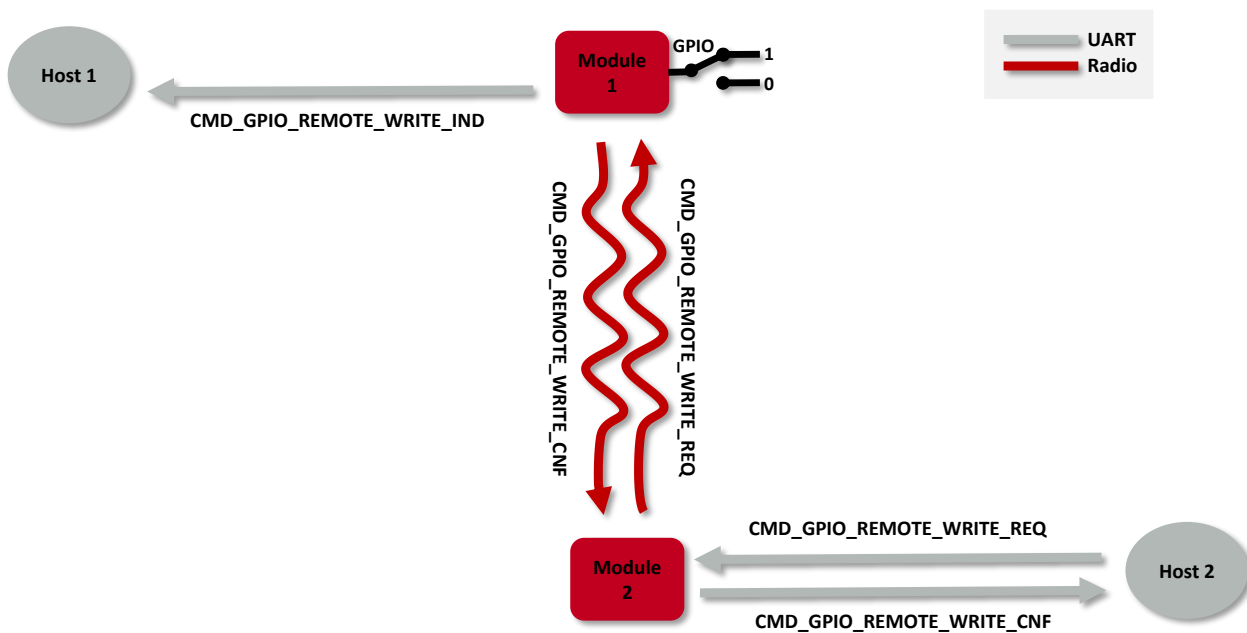
Figure 15: Set the output value of a GPIO via remote device
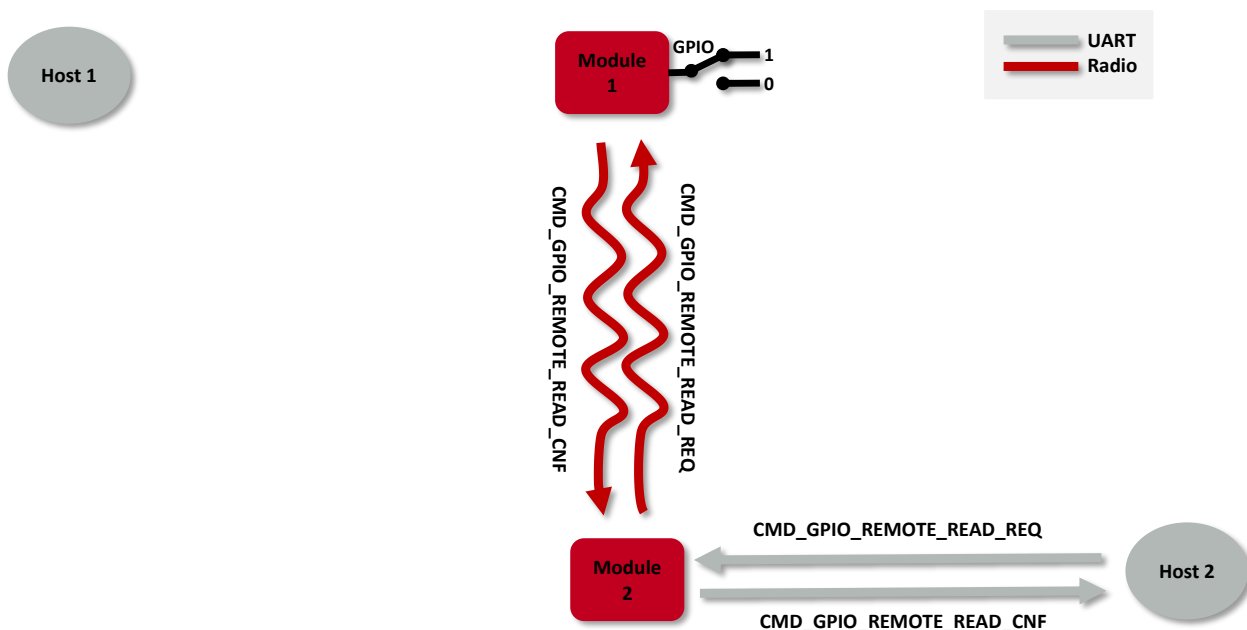
Figure 16: Read the input value of a GPIO via remote device

11.1 Supported GPIO_IDs for remote and local control

The following GPIOs of the Proteus-III are supported for remote and local access.
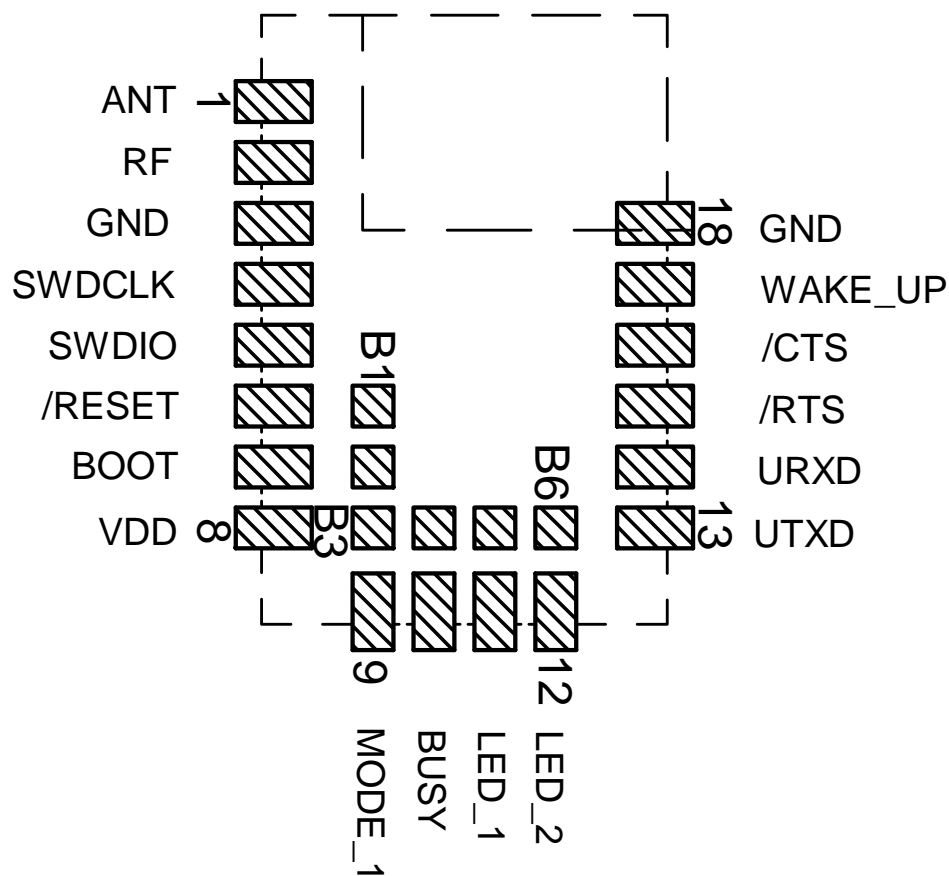
No	GPIO_ID	Supported functions
B1	1	Input, Output
B2	2	Input, Output
B3	3	Input, Output
B4	4	Input, Output
B5	5	Input, Output
B6	6	Input, Output

Table 22: Supported GPIO_IDs

12 Customizing the Proteus-III

12.1 DIS - Device information service

Besides the AMBER SPP-like profile for data transmission, the Proteus-III contains the so called Device Information Service. This profile exposes manufacturer information about a device and is used to personalize the Proteus-III to fuse with the custom product. The Device Information Service is a standard BLE profile that is recognized by all devices with Bluetooth® capabilities. It contains the following fields, that can only be modified by updating the respective user setting using the `CMD_SET_REQ` command:

Field name	User setting	Maximum length
Manufacturer Name String	`DIS_ManufacturerName`	64
Model Number String	`DIS_ModelNumber`	64
Serial Number String	`DIS_SerialNumber`	64
Hardware Revision String	`DIS_HWVersion`	16
Software Revision String	`DIS_SWVersion`	16

Furthermore, the user setting `DIS_Flags` defines which of the described DIS fields are finally placed in the DIS profile. Thus after adding content to the a DIS field user setting, like `DIS_ManufacturerName`, the user setting `DIS_Flags` has to be adapted such that the content is added to the profile.

12.2 UUID

The UUID is a unique number identifying a BLE profile and thus describing its functions. The Proteus-III using its standard UUID is compatible to all devices that implement the AMBER SPP-like profile, whichever device it is integrated. To suspend this interoperability, the user setting `RF_SPPBaseUUID` can be used to modify the UUID of the AMBER SPP-like profile. With this, a new custom SPP-like profile is defined that is solely known to those that chose the new UUID.
To generate a custom UUID the Bluetooth® SIG recommends to use the tool:
http://www.uuidgenerator.net/

12.3 Appearance

The appearance of the Bluetooth® device is a 2 Bytes value defined by the Bluetooth® SIG. It can be configured by adapting the parameter `RF_Appearance`.

13 Custom firmware

13.1 Custom configuration of standard firmware

The configuration of standard firmware includes adoption of the non-volatile Usersettings (see chapter 8) to customer requirements and creating a customized product on base of the standard product with a unique ordering number for a specific customer that needs this configuration.

For example if the UART baud rate shall be changed from the default value to another value. This variant will result in a customer exclusive module with a unique ordering number. This will also fix the firmware version to a specific and customer tested version and thus results in a customer exclusive module with a unique ordering number.

Further scheduled firmware updates of the standard firmware will not be applied to this variant automatically. Applying updates or further functions require a customer request and customer release procedure.

13.2 Customer specific firmware

A customer specific firmware may include "Custom configuration of standard firmware" plus additional options or functions and tasks that are customer specific and not part of the standard firmware.

Further scheduled firmware updates of the standard firmware will not be applied to this variant automatically. Applying updates or further functions require a customer request and customer release procedure.

This also results in a customer exclusive module with a unique ordering number.

An example for this level of customization are functions like host-less operation where the module will perform data generation (e.g. by reading a SPI or I^2C sensor) and cyclic transmission of this data to a data collector while sleeping or being passive most of the time.

Also replacing UART with SPI as host communication interface is classified such a custom specific option.

Certification critical changes need to be re-evaluated by an external qualified measurement laboratory. These critical changes may occur when e.g. changing radio parameters, the channel access method, the duty-cycle or in case of various other functions and options possibly used or changed by a customer specific firmware.

13.3 Customer firmware

A customer firmware is a firmware written and tested by the customer himself or a 3rd party as a customer representative specifically for the hardware platform provided by a module.

This customer firmware (e.g. in form of a Intel hex file) will be implemented into the module's production process at our production side.

This also results in a customer exclusive module with a unique ordering number.

The additional information needed for this type of customer firmware, such as hardware specific details and details towards the development of such firmware are not available for the public and can only be made available to qualified customers.

STOP The qualification(s) and certification(s) of the standard firmware cannot be applied to this customer firmware solution without a review and verification.

13.4 Contact for firmware requests

Please contact your local field sales engineer (FSE) or wireless-sales@we-online.com for quotes regarding this topics.

14 Firmware update

The Proteus-III offers two possibilities of updating its firmware.

14.1 Firmware update using the SWD interface

To update the firmware of the Proteus-III the SWD interface of the module and a supported flasher hardware (such as SEGGER J-Link plus) can be used. Therefore the pins *GND*, *VDD*, */RESET*, *SWDIO* and *SWDCLK* of the module have to be accessible and connected to the flasher hardware accordingly (corresponding documentation of flasher has to be read for further information). After the connection of a flash adapter to this SWD interface, the new firmware can be flashed using the corresponding PC software nrfjprog.exe available directly from Nordic Semiconductor.

nrfjprog.exe - -family NRF52 - -eraseall
nrfjprog.exe - -family NRF52 - -program Proteus-III.hex
nrfjprog.exe - -family NRF52 - -reset

For this reason a .hex-file can be provided, which contains all firmware parts (bootloader, softdevice, application). The name of the hex file has to be adopted accordingly in the command line above.

> **!** This is the only method by which the module could be recovered in the event of a serious software fault or corrupted memory. This method is fail-safe.

14.2 Firmware update using the Proteus-III OTA bootloader

The second method offers a possibility to update the firmware over the air (OTA). Therefore, the Nordic nRF52 BLE DFU Bootloader is integrated into the Proteus-III's firmware, which will communicate over the BLE interface. The OTA bootloader mode is a distinct operating mode besides the normal operating modes mentioned before. For this reason, a .zip-file can be provided, which contains all (bootloader, Softdevice, application) parts of the firmware in an encrypted and authenticated package.

Before starting any update procedure, please check whether the installed firmware can be updated to a new one:

Version of the firmware before the update	Version of the new firmware	App
1.x.x	1.x.x	Nordic nRF Toolbox 2.7.2 or newer

Table 23: Compatibility matrix

To start the bootloader, one of the following two conditions has to be satisfied:

1. send the command `CMD_BOOTLOADER_REQ` to the module to restart in bootloader mode

---

2. during a reset and while restarting, a low signal has to be present on the *BOOT* pin of the module to start it in bootloader mode

2. during a reset and while restarting, a low signal has to be present on the *BOOT* pin of the module to start it in bootloader mode

The bootloader mode has started successfully if *LED_1* has turned on.   After the boot-loader has started successfully, the module goes into the advertising mode using the name "DFUProteus-III".   Now, any BLE device hosting an application that understands the commands of the Nordic nRF52 BLE DFU Bootloader can connect in order to update the Proteus-III firmware.

The DFU application of the used App (see Table 23) is such an application. For more details, please refer to chapter 14.2.1. As soon as a connection has been set up, *LED_1* turns off again and *LED_2* turns on.

> The implemented Nordic nRF52 BLE DFU bootloader uses a dual bank method to update the firmware. Thus, the old firmware is only replaced once the new firmware has been transferred successfully. This prevents the module from being flashed with a faulty firmware.

> An OTA firmware update will take several minutes to be performed, the duration is also dependent how much of the firmware shall be updated (application only or complete update).

> The max connection interval of the update service is set to 30ms. Please check whether your mobile supports this speed.

> This method is only applicable if the Proteus-III still contains an intact boot-loader. In order to be able to recover a faulty module, we recommend to have access to the relevant JTAG pins required to perform a wired firmware update.

14.2.1 Firmware update steps using the Nordic nRF Toolbox app

If the radio module Proteus-III has been set to bootloader mode, the Nordic nRF Toolbox app can be used to perform the OTA firmware update.

- Open the app, select the DFU function and press "SELECT FILE"
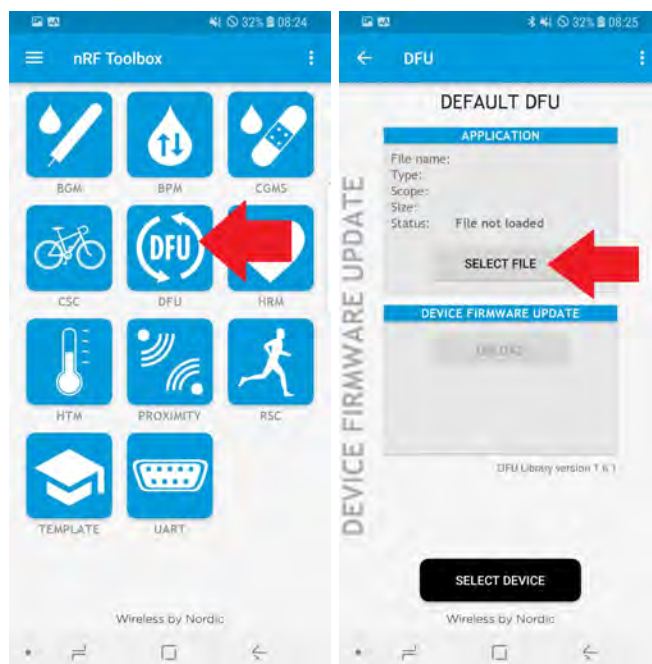
- Choose "Distribution packet (ZIP)", select the new firmware and choose "All".
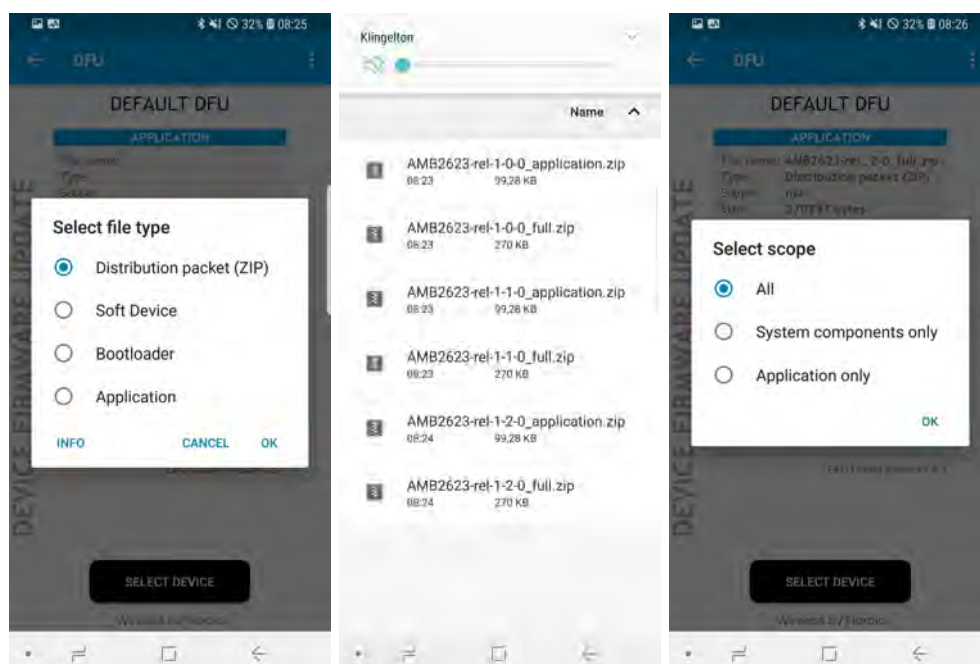
- Press "SELECT DEVICE" and choose the appropriate module in the list of displayed devices. In bootloader mode the module is named "DFUxxxx".

> ⓘ If there is no device named "DFUxxxx" on the radio, please check whether the module has been started in bootloader mode.
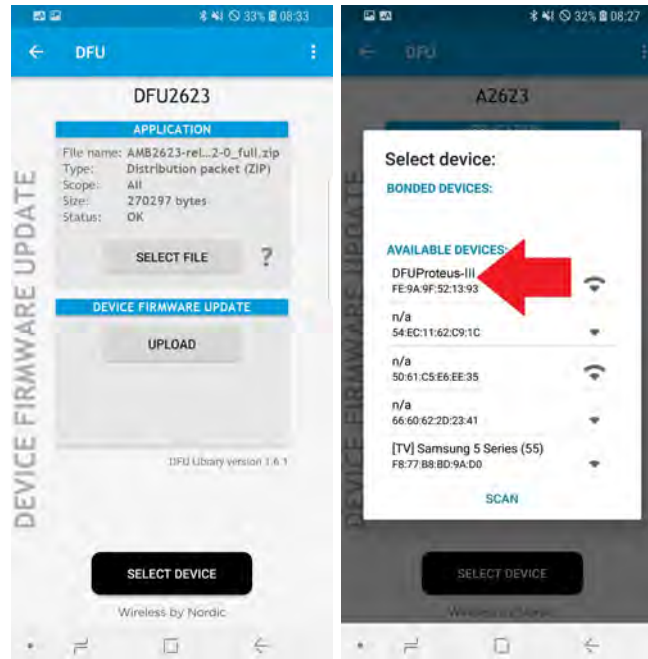
- Then press "UPLOAD" to transmit the selected firmware to the selected device.

15 Firmware history

Version 0.x.x "Engineering"

- Pre-release for test run

Version 1.0.0 "Release"

- First production release

16 Design in guide

16.1 Advice for schematic and layout

For users with less RF experience it is advisable to closely copy the relating evaluation board with respect to schematic and layout, as it is a proven design. The layout should be conducted with particular care, because even small deficiencies could affect the radio performance and its range or even the conformity.
The following general advice should be taken into consideration:

- A clean, stable power supply is strongly recommended. Interference, especially oscillation can severely restrain range and conformity.

- Variations in voltage level should be avoided.

- LDOs, properly designed in, usually deliver a proper regulated voltage.

- Blocking capacitors and a ferrite bead in the power supply line can be included to filter and smoothen the supply voltage when necessary.

No fixed values can be recommended, as these depend on the circumstances of the application (main power source, interferences etc.).

The use of an external reset IC should be considered if one of the following points is relevant:

- The slew rate of the power supply exceeds the electrical specifications.

- The effect of different current consumptions on the voltage level of batteries or voltage regulators should be considered. The module draws higher currents in certain scenarios like start-up or radio transmit which may lead to a voltage drop on the supply. A restart under such circumstances should be prevented by ensuring that the supply voltage does not drop below the minimum specifications.

- Voltage levels below the minimum recommended voltage level may lead to misfunction. The /Reset pin of the module shall be held on LOW logic level whenever the VCC is not stable or below the minimum operating Voltage.

- Special care must be taken in case of battery powered systems.

- Elements for ESD protection should be placed on all pins that are accessible from the outside and should be placed close to the accessible area. For example, the RF-pin is accessible when using an external antenna and should be protected.

- ESD protection for the antenna connection must be chosen such as to have a minimum effect on the RF signal. For example, a protection diode with low capacitance such as the LXES15AAA1-100 or a 68 nH air-core coil connecting the RF-line to ground give good results.

- Placeholders for optional antenna matching or additional filtering are recommended.

- The antenna path should be kept as short as possible.

> Again, no fixed values can be recommended, as they depend on the influencing circumstances of the application (antenna, interferences etc.).
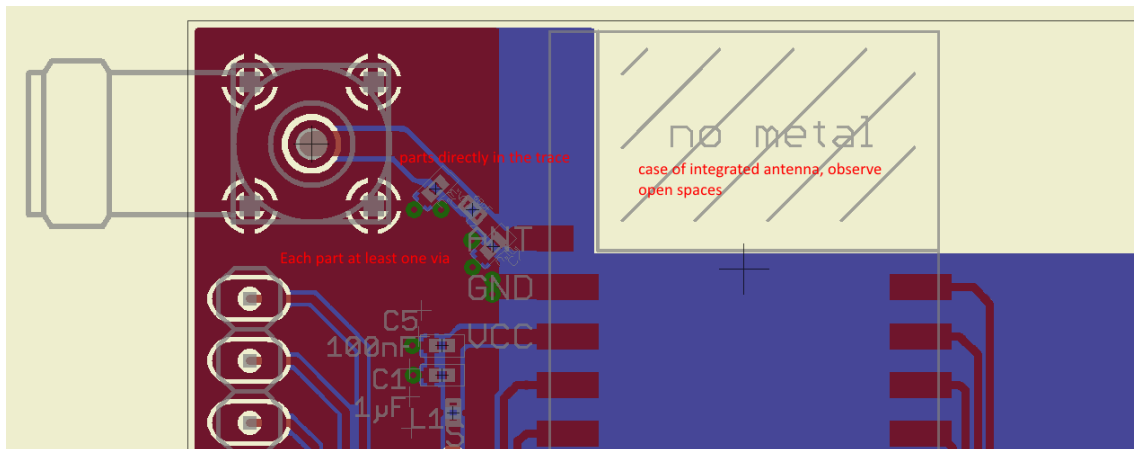
Figure 17: Layout

- To avoid the risk of short circuits and interference there should be no routing underneath the module on the top layer of the baseboard.

- On the second layer, a ground plane is recommended, to provide good grounding and shielding to any following layers and application environment.

- In case of integrated antennas it is required to have areas free from ground. This area should be copied from the evaluation board.

- The area with the integrated antenna must overlap with the carrier board and should not protrude, as it is matched to sitting directly on top of a PCB.

- Modules with integrated antennas should be placed with the antenna at the edge of the main board. It should not be placed in the middle of the main board or far away from the edge. This is to avoid tracks beside the antenna.

- Filter and blocking capacitors should be placed directly in the tracks without stubs, to achieve the best effect.

- Antenna matching elements should be placed close to the antenna / connector, blocking capacitors close to the module.

- Ground connections for the module and the capacitors should be kept as short as possible and with at least one separate through hole connection to the ground layer.

- ESD protection elements should be placed as close as possible to the exposed areas.
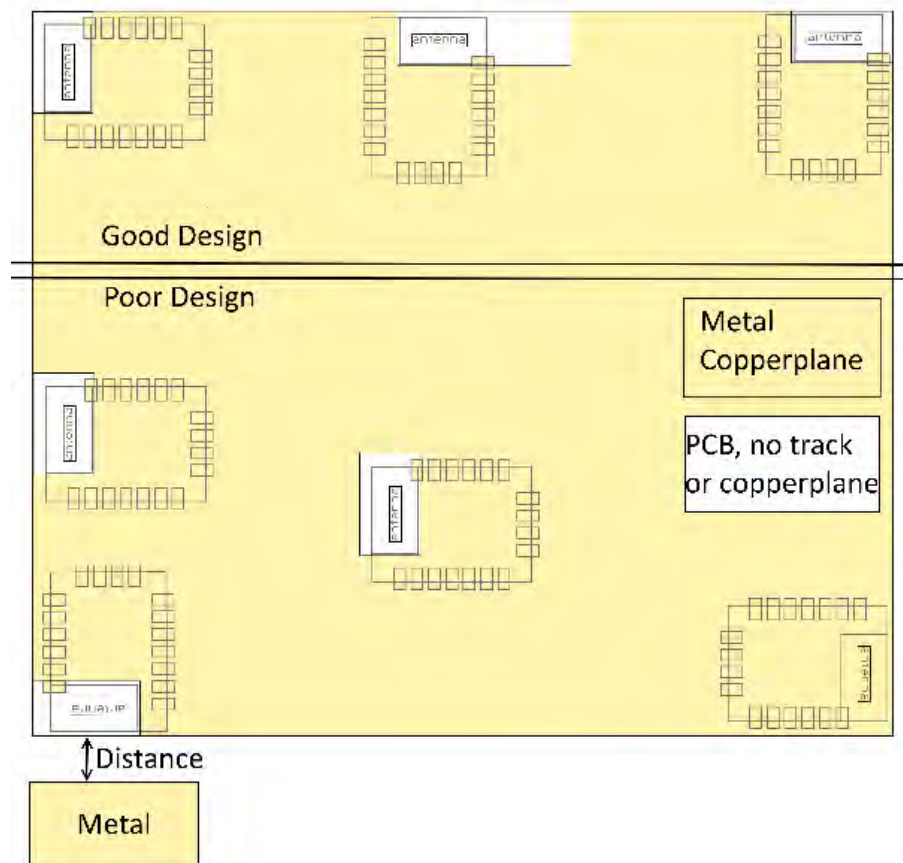
Figure 18: Placement of the module with integrated antenna

16.2 Dimensioning of the micro strip antenna line

The antenna track has to be designed as a 50Ω feed line. The width W for a micro strip can be calculated using the following equation:

$$W = 1.25 \times \left(\frac{5.98 \times H}{e^{\frac{50 \times \sqrt{\epsilon_r + 1.41}}{87}}} - T_{met} \right) \tag{1}$$

Example:
A FR4 material with ϵ_r = 4.3, a height H = 1000 μm and a copper thickness of T_{met} = 18 μm
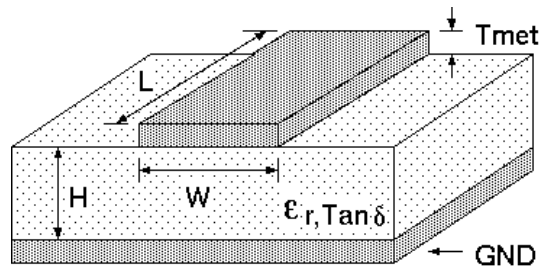
Figure 19: Dimensioning the antenna feed line as micro strip

will lead to a trace width of W $\sim$ 1.9 mm. To ease the calculation of the micro strip line (or e.g. a coplanar) many calculators can be found in the internet.

- As rule of thumb a distance of about 3×W should be observed between the micro strip and other traces / ground.

- The micro strip refers to ground, therefore there has to be the ground plane underneath the trace.

- Keep the feeding line as short as possible.

16.3 Antenna solutions

There exist several kinds of antennas, which are optimized for different needs. Chip antennas are optimized for minimal size requirements but at the expense of range, PCB antennas are optimized for minimal costs, and are generally a compromise between size and range. Both usually fit inside a housing.

Range optimization in general is at the expense of space. Antennas that are bigger in size, so that they would probably not fit in a small housing, are usually equipped with a RF connector. A benefit of this connector may be to use it to lead the RF signal through a metal plate (e.g. metal housing, cabinet).

As a rule of thumb a minimum distance of $\lambda/10$ (which is 3.5 cm @ 868 MHz and 1.2 cm @ 2.44 GHz) from the antenna to any other metal should be kept. Metal placed further away will not directly influence the behavior of the antenna, but will anyway produce shadowing.

> Keep the antenna away from large metal objects as far as possible to avoid electromagnetic field blocking.

> The choice of antenna might have influence on the safety requirements.

In the following chapters, some special types of antenna are described.

16.3.1 Wire antenna

An effective antenna is a λ/4 radiator with a suiting ground plane. The simplest realization is a piece of wire. It's length is depending on the used radio frequency, so for example 8.6 cm 868.0 MHz and 3.1 cm for 2.440 GHz as frequency. This radiator needs a ground plane at its feeding point. Ideally, it is placed vertically in the middle of the ground plane. As this is often not possible because of space requirements, a suitable compromise is to bend the wire away from the PCB respective to the ground plane. The λ/4 radiator has approximately 40 Ω input impedance, therefore matching is not required.

16.3.2 Chip antenna

There are many chip antennas from various manufacturers. The benefit of a chip antenna is obviously the minimal space required and reasonable costs. However, this is often at the expense of range. For the chip antennas, reference designs should be followed as closely as possible, because only in this constellation can the stated performance be achieved.

16.3.3 PCB antenna

PCB antenna designs can be very different. The special attention can be on the miniaturization or on the performance. The benefits of the PCB antenna are their small / not existing (if PCB space is available) costs, however the evaluation of a PCB antenna holds more risk of failure than the use of a finished antenna. Most PCB antenna designs are a compromise of range and space between chip antennas and connector antennas.

16.3.4 Antennas provided by Würth Elektronik eiSos

16.3.4.1 2600130011 - Helike - 169 MHz dipole antenna

Figure 20: 169 MHz dipole-antenna

Specification	Value
Frequency range [MHz]	169
Impedance [Ω]	50
VSWR	≤ 2.1
Gain [dBi]	1
Dimensions (L x d) [mm]	320 x 15
Weight [g]	42
Connector	SMA plug
Operating Temp. [℃]	-40 – +85

This antenna requires a ground plane which will influence the electrical parameters.

16.3.4.2 2600130041 - Herse - 434 MHz dipole antenna

Figure 21: 434 MHz dipole-antenna

Specification	Value
Frequency range [MHz]	433
Impedance [Ω]	50
VSWR	≤ 1.5
Polarization	Vertical
Radiation	Omni
Gain [dBi]	0
Antenna Cover	TPEE
Dimensions (L x d) [mm]	90 x 12
Weight [g]	9.6
Connector	SMA plug
Operating Temp. [°C]	-40 − +80

This antenna requires a ground plane which will influence the electrical parameters.

16.3.4.3 2600130081 - Hyperion-I - 868 MHz dipole antenna

Figure 22: 868 MHz dipole-antenna

Ideally suited for applications where no ground plane is available.

> The 2600130081 antenna can be also used for 902MHz - 928MHz range.

Specification	Value
Center frequency [MHz]	868
Frequency range [MHz]	853 – 883
Wavelength	0.5 wave
VSWR	≤ 2.0
Impedance [Ω]	50
Connector	SMA (Male)
Dimensions (L x d) [mm]	142 x 10
Peak gain [dBi]	-2.3
Operating temp. [℃]	-30 – +80

16.3.4.4 2600130082 - Hyperion-II - 868 MHz magnetic base antenna

Well suited for applications where the RF is lead through a metal wall that could serve as ground plane to the antenna.

Figure 23: 868 MHz magnet foot antenna with 1.5 m antenna cable

The 2600130082 is a kind of λ/4 radiator and therefore needs a ground plane at the feeding point.

Specification	Value
Frequency range [MHz]	824 − 894
VSWR	≤ 2.0
Polarisation	Vertical
Impedance [Ω]	50±5
Connector	SMA (Male)
Dimensions (L x d) [mm]	89.8 x 27
Weight [g]	50±5
Operating temp. [℃]	-30 − +60

16.3.4.5 2600130021 - Himalia - 2.4 GHz dipole antenna

Figure 24: 2.4 GHz dipole-antenna

Due to the fact, that the antenna has dipole topology there is no need for an additional ground plane. Nevertheless the specification was measured edge mounted and 90° bent on a 100 x 100 mm ground plane.

Specification	Value
Frequency range [GHz]	2.4 – 2.5
Impedance [Ω]	50
VSWR	≤ 2:1
Polarization	Linear
Radiation	Omni-Directional
Peak Gain [dBi]	2.8
Average Gain [dBi]	-0.6
Efficiency	85 %
Dimensions (L x d) [mm]	83.1 x 10
Weight [g]	7.4
Connector	SMA plug
Operating temp. [℃]	-40 – +80

Special care must be taken for FCC certification when using this external antenna to fulfill the requirement of permanently attached antenna or unique coupling for example by using the certified dipole antenna in a closed housing, so that only through professional installation it is possible to remove it.

17 Reference design

Proteus-III was tested and certified on the corresponding Proteus-III evaluation board. For the European Conformity the evaluation board serves as reference design, for the FCC it is mandatory to follow at least the trace design.

Complete layout and schematic information can be found in the manual of the Proteus-III evaluation board.

17.1 EV-Board
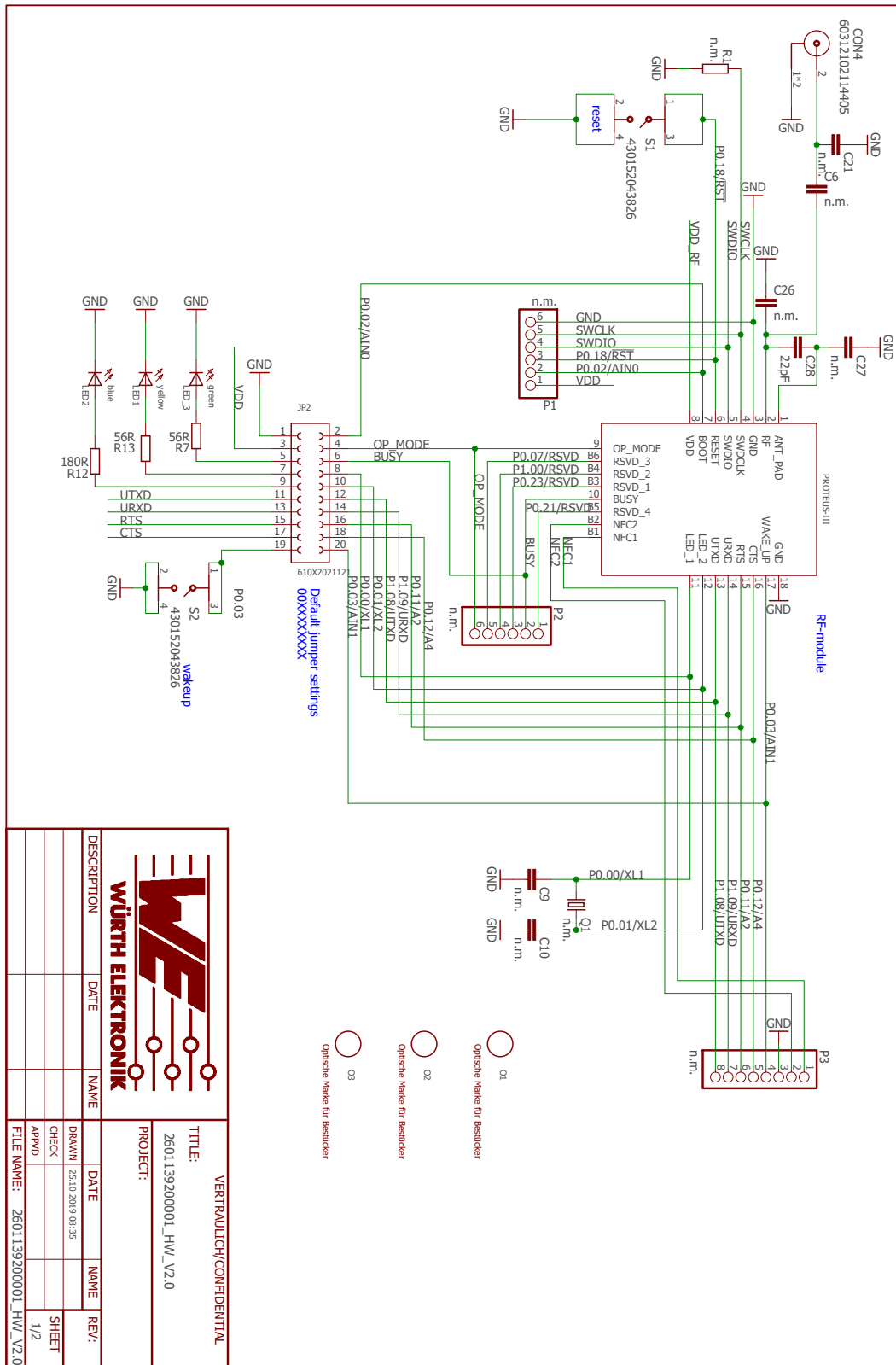
Figure 25: Reference design: Schematic page 1

Figure 26: Reference design: Schematic page 2
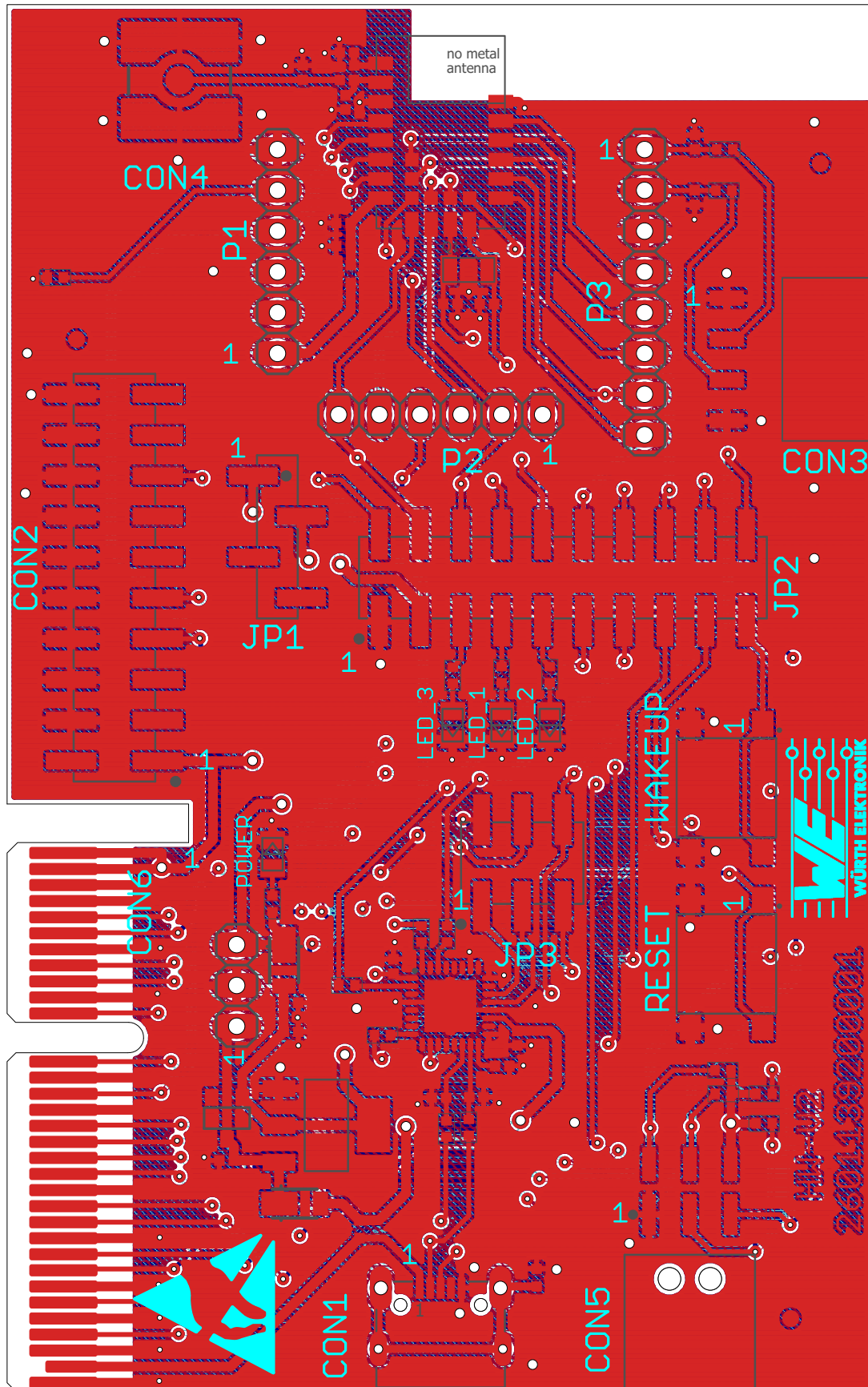
Figure 27: Reference design: Layout

17.2 Trace design
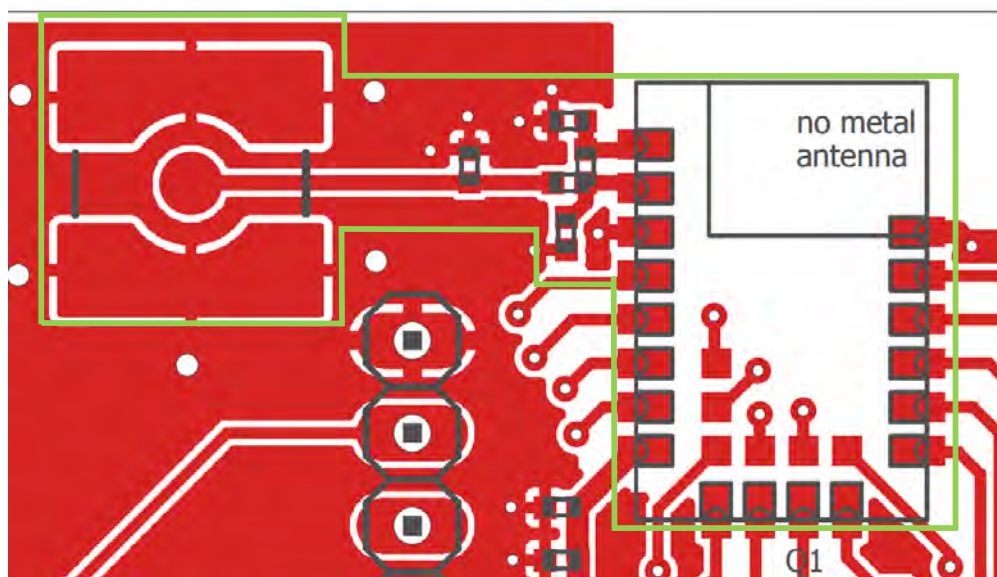
Figure 28: Trace design: Layout

Nr	Copper		Isolation
1	0.035mm		
			10mil
2	0.018mm		
			1mm
15	0.018mm		
			10mil
16	0.035mm		

Gesamt: 1.614mm

Figure 29: Reference design: Stack-up

- Top layer is used for routing, filled with ground plane except area under the module and antenna free area.

- Second layer is filled with ground plane, except the antenna free area.

- Third layer is the supply layer, except antenna free area. Some routing is allowed, not dividing the supply layer in to many or too small parts.

- Bottom layer is used for routing and filled with ground.
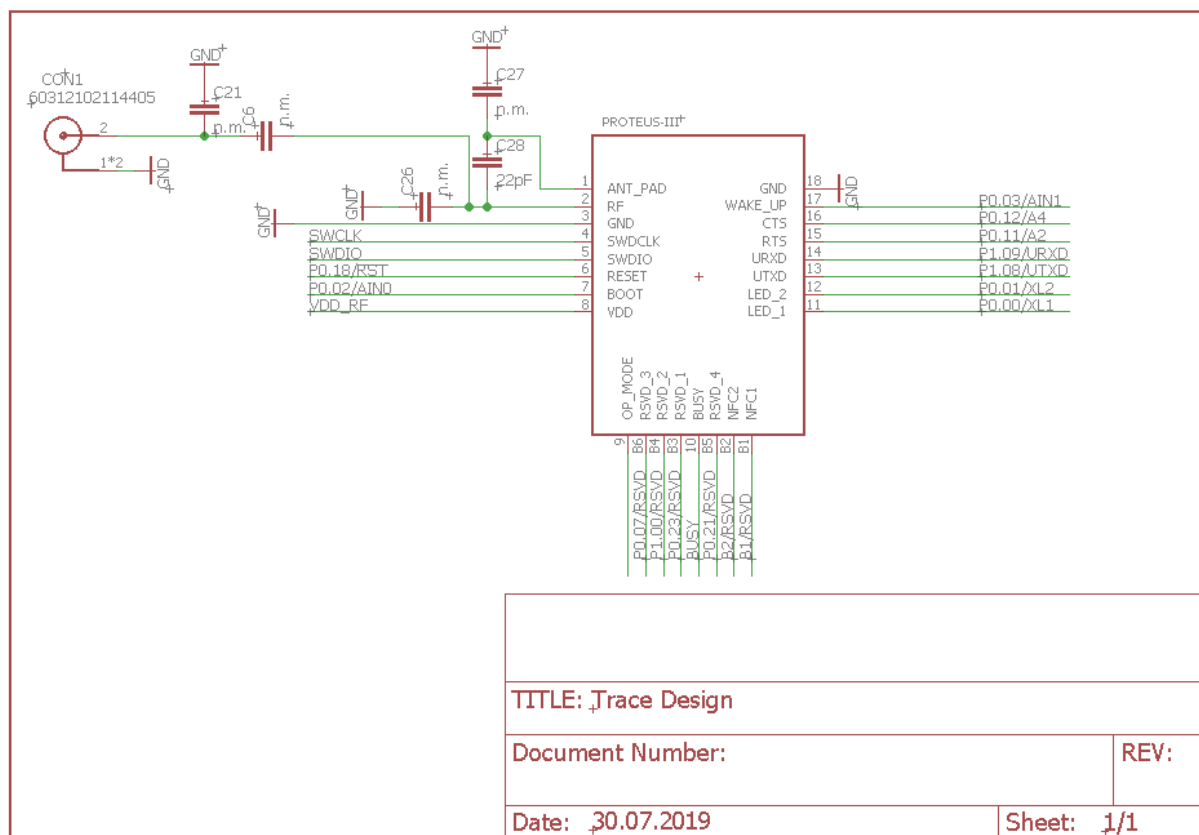
Figure 30: Trace design: Schematic

Two variants of the Proteus-III are certified:

- Using the on-board PCB antenna placing 22pF on C28 along with C27 and C28 as additional components to support 50Ω matched feed for the PCB antenna if needed in such case capacitors C6 and C21 would not be assembled.

- Using the external antenna connection by assembling 22pF on C6 along with C21 and C26 as additional components to support 50Ω matched feed for a quarter-wave-length whip antenna, in such case C27, and C28 would not be assembled.

> To reference to the Würth Elektronik eiSos' FCC ID it is mandatory to use the trace design.

18 Manufacturing information

18.1 Moisture sensitivity level

This wireless connectivity product is categorized as JEDEC Moisture Sensitivity Level 3 (MSL3), which requires special handling.

More information regarding the MSL requirements can be found in the IPC/JEDEC J-STD-020 standard on www.jedec.org.

More information about the handling, picking, shipping and the usage of moisture/re-flow and/or process sensitive products can be found in the IPC/JEDEC J-STD-033 standard on *www.jedec.org*.

18.2 Soldering

18.2.1 Reflow soldering

Attention must be paid on the thickness of the solder resist between the host PCB top side and the modules bottom side. Only lead-free assembly is recommended according to JEDEC J-STD020.

Profile feature		Value
Preheat temperature Min	$T_{S\ Min}$	150 °C
Preheat temperature Max	$T_{S\ Max}$	200 °C
Preheat time from $T_{S\ Min}$ to $T_{S\ Max}$	t_S	60 - 120 seconds
Ramp-up rate (T_L to T_P)		3 °C / second max.
Liquidous temperature	T_L	217 °C
Time t_L maintained above T_L	t_L	60 - 150 seconds
Peak package body temperature	T_P	see table below
Time within 5 °C of actual preak temperature	t_P	20 - 30 seconds
Ramp-down Rate (T_P to T_L)		6 °C / second max.
Time 20 °C to T_P		8 minutes max.

Table 24: Classification reflow soldering profile, Note: refer to IPC/JEDEC J-STD-020E

Package thickness	Volume mm^3 <350	Volume mm^3 350-2000	Volume mm^3 >2000
< 1.6mm	260°C	260°C	260°C
1.6mm - 2.5mm	260°C	250°C	245°C
> 2.5mm	250°C	245°C	245°C

Table 25: Package classification reflow temperature, PB-free assembly, Note: refer to IPC/-JEDEC J-STD-020E

It is recommended to solder this module on the last re-flow cycle of the PCB. For solder paste use a LFM-48W or Indium based SAC 305 alloy (Sn 96.5 / Ag 3.0 / Cu 0.5 / Indium 8.9HF / Type 3 / 89%) type 3 or higher.

The re-flow profile must be adjusted based on the thermal mass of the entire populated PCB, heat transfer efficiency of the re-flow oven and the specific type of solder paste used. Based on the specific process and PCB layout the optimal soldering profile must be adjusted and verified. Other soldering methods (e.g. vapor phase) have not been verified and have to be validated by the customer at their own risk. Rework is not recommended.
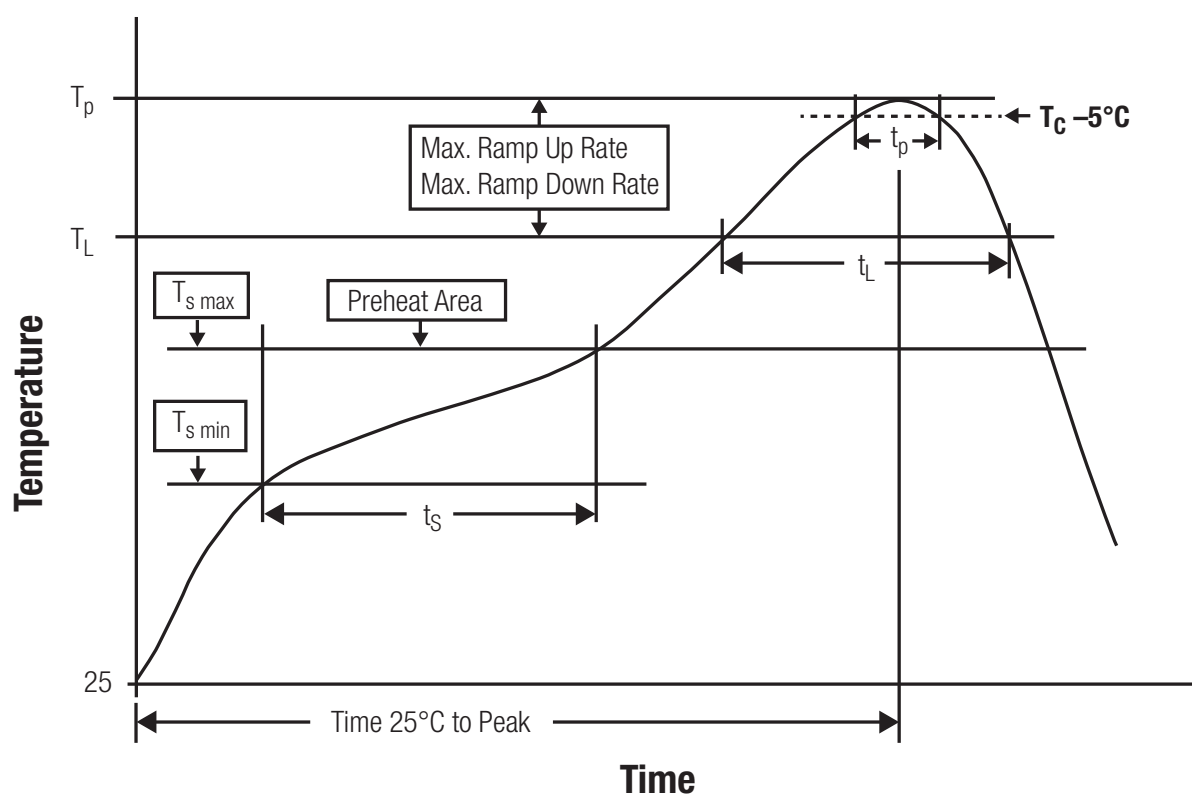
Figure 31: Reflow soldering profile

After reflow soldering, visually inspect the board to confirm proper alignment

18.2.2 Cleaning

Do not clean the product. Any residue cannot be easily removed by washing. Use a "no clean" soldering paste and do not clean the board after soldering.

- Do not clean the product with water. Capillary effects can draw water into the gap between the host PCB and the module, absorbing water underneath it. If water is trapped inside, it may short-circuit adjoining pads. The water may also destroy the label and ink-jet printed text on it.

- Cleaning processes using alcohol or other organic solvents may draw solder flux residues into the housing, which won't be detected in a post-wash inspection. The solvent may also destroy the label and ink-jet printed text on it.

- Do not use ultrasonic cleaning as it will permanently damage the part, particularly the crystal oscillators.

18.2.3 Other notations

- Conformal coating of the product will result in the loss of warranty. The RF shields will not protect the part from low-viscosity coatings.

- Do not attempt to improve the grounding by forming metal strips directly to the EMI covers or soldering on ground cables, as it may damage the part and will void the warranty.

- Always solder every pad to the host PCB even if some are unused, to improve the mechanical strength of the module.

- The part is sensitive to ultrasonic waves, as such do not use ultrasonic cleaning, welding or other processing. Any ultrasonic processing will void the warranty.

18.3 ESD handling

This product is highly sensitive to electrostatic discharge (ESD). As such, always use proper ESD precautions when handling. Make sure to handle the part properly throughout all stages of production, including on the host PCB where the module is installed. For ESD ratings, refer to the module series' maximum ESD section. For more information, refer to the relevant chapter 2. Failing to follow the aforementioned recommendations can result in severe damage to the part.

- the first contact point when handling the PCB is always between the local GND and the host PCB GND, unless there is a galvanic coupling between the local GND (for example work table) and the host PCB GND.

- Before assembling an antenna patch, connect the grounds.

- While handling the RF pin, avoid contact with any charged capacitors and be careful when contacting any materials that can develop charges (for example coaxial cable with around 50-80 pF/m, patch antenna with around 10 pF, soldering iron etc.)

- Do not touch any exposed area of the antenna to avoid electrostatic discharge. Do not let the antenna area be touched in a non ESD-safe manner.

- When soldering, use an ESD-safe soldering iron.

18.4 Safety recommendations

It is your duty to ensure that the product is allowed to be used in the destination country and within the required environment. Usage of the product can be dangerous and must be tested and verified by the end user. Be especially careful of:

- Use in areas with risk of explosion (for example oil refineries, gas stations).

- Use in areas such as airports, aircraft, hospitals, etc., where the product may interfere with other electronic components.

It is the customer's responsibility to ensure compliance with all applicable legal, regulatory and safety-related requirements as well as applicable environmental regulations. Disassembling the product is not allowed. Evidence of tampering will void the warranty.

- Compliance with the instructions in the product manual is recommended for correct product set-up.

- The product must be provided with a consolidated voltage source. The wiring must meet all applicable fire and security prevention standards.

- Handle with care. Avoid touching the pins as there could be ESD damage.

Be careful when working with any external components. When in doubt consult the technical documentation and relevant standards. Always use an antenna with the proper characteristics.

> Würth Elektronik eiSos radio modules with high output power of up to 500 mW, as for example the radio module Thebe-II, generate a high amount of warmth while transmitting. The manufacturer of the end device must take care of potentially necessary actions for his application.

19 Physical dimensions

19.1 Dimensions

Dimensions
12 x 8 x 2 mm

Table 26: Dimensions

19.2 Weight

Weight
<1g

Table 27: Weight

19.3 Module drawing
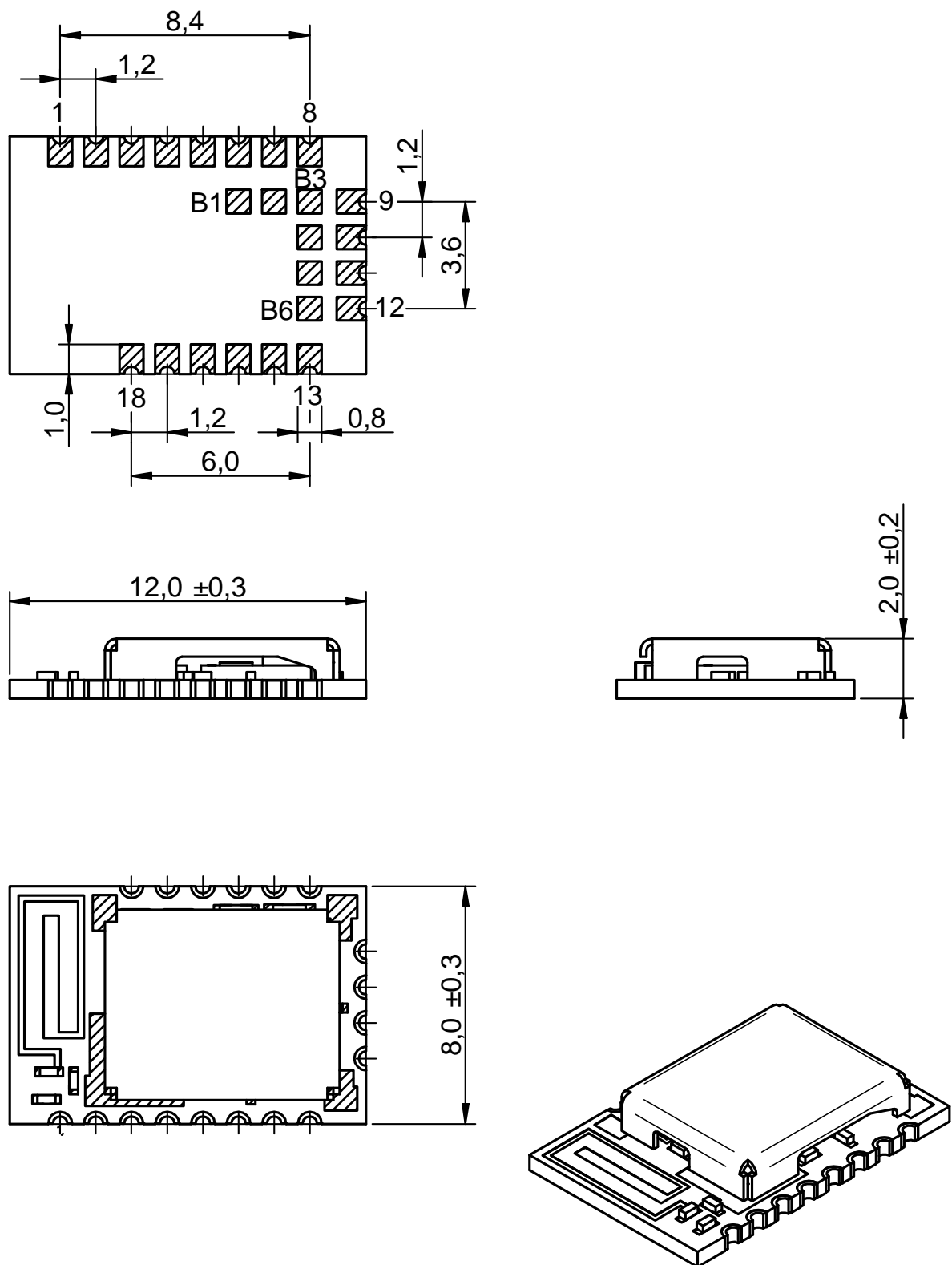
Figure 32: Module dimensions [mm]

19.4 Footprint
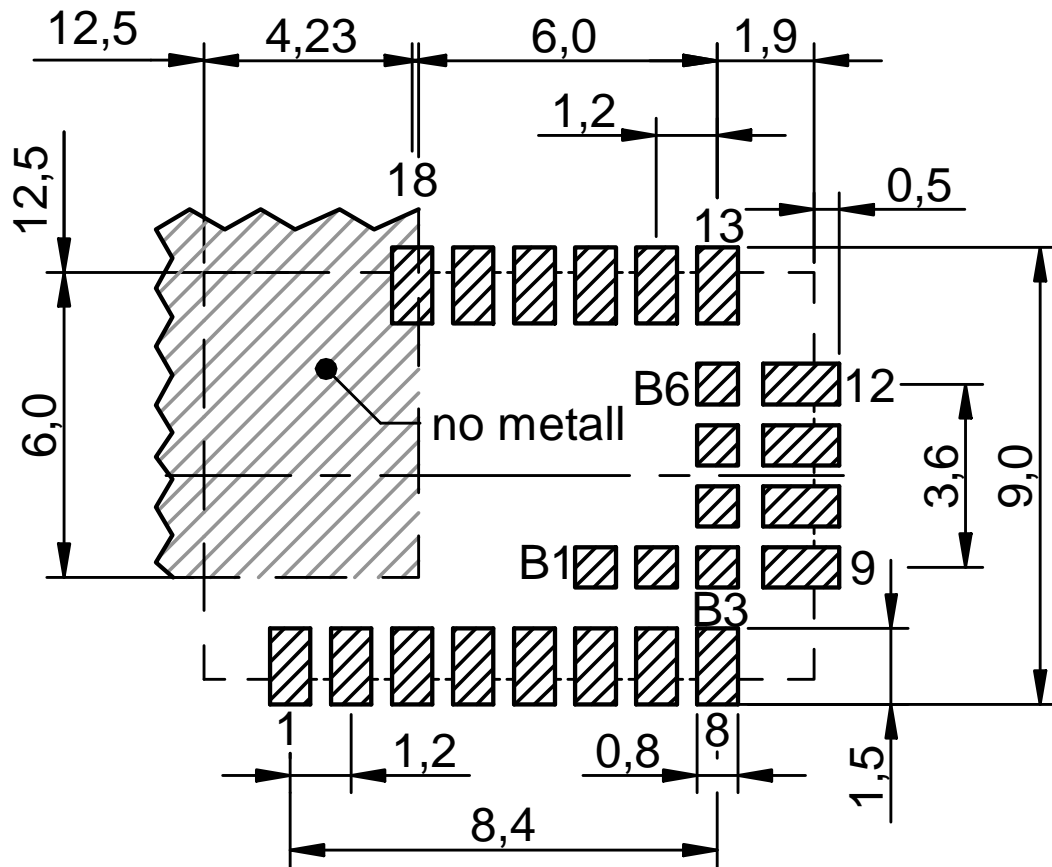
Figure 33: Footprint [mm]

19.5 Antenna free area

To avoid influence and mismatching of the antenna the recommended free area around the antenna should be maintained. As rule of thumb a minimum distance of metal parts to the antenna of $\lambda/10$ should be kept (see figure 33). Even though metal parts would influence the characteristic of the antenna, but the direct influence and matching keep an acceptable level.

20 Marking

20.1 Lot number

The 15 digit lot number is printed in numerical digits as well as in form of a machine readable bar code. It is divided into 5 blocks as shown in the following picture and can be translated according to the following table.
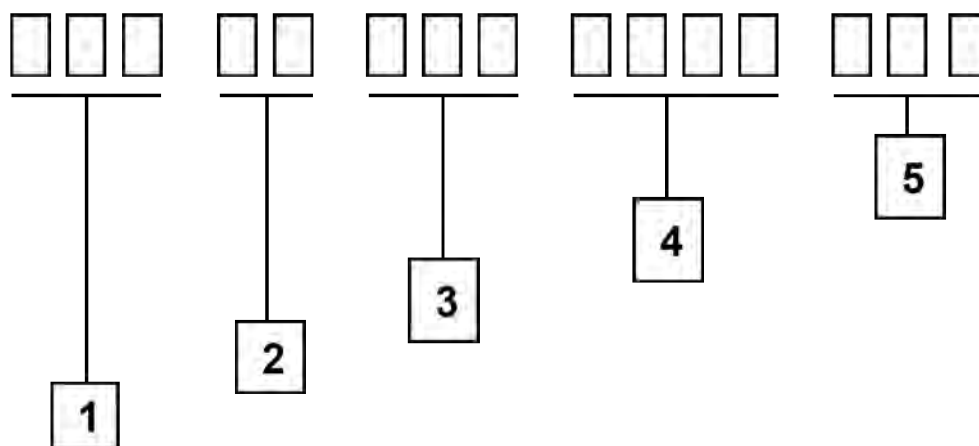
Figure 34: Lot number structure

Block	Information	Example(s)
1	eiSos internal, 3 digits	439
2	eiSos internal, 2 digits	01
3	Hardware version, 3 digits	V2.4 = 024, V12.2 = 122
4	Date code, 4 digits	1703 = week 03 in year 2017, 1816 = week 16 in year 2018
5	Firmware version, 3 digits	V3.2 = 302, V5.13 = 513

Table 28: Lot number details

As the user can perform a firmware update the printed lot number only shows the factory delivery state. The currently installed firmware can be requested from the module using the corresponding product specific command. The firmware version as well as the hardware version are restricted to show only major and minor version not the patch identifier.

20.2 General labeling information

The module labels may include the following fields:

- Manufacturer identification WE, Würth Elektronik or Würth Elektronik eiSos

- WE Order Code and/or article alias

- Serial number or MAC address

- Certification identifiers (CE, FCC ID, IC, ARIB,...)

- Bar code or 2D code containing the serial number or MAC address

The serial number includes the product ID (PID) and an unique 6 digit number. The first 1 to 3 digits represent the PID, then the "." marks the start of the 6 digit counter to create a unique product marking.

In case of small labels, the 3 byte manufacturer identifier (0x0018DA) of the MAC address is not printed on the labels. The 3 byte counter printed on the label can be used with this 0018DA to produce the full MAC address by appending the counter after the manufacturer identifier.

Figure 35: Label of the Proteus-III

21 Bluetooth SIG listing/qualification

Type	Data
Design name	Proteus-III
Declaration ID	D047845
QDID	141060
Specification name	5.1
Project type	End product

Each product containing intellectual property of the Bluetooth® Special Interest Group (SIG) must be qualified by the SIG to obtain the corresponding Declaration ID. Every new Bluetooth® design must pass the qualification process, even when linking to a Bluetooth® design that is already qualified. To go through the qualification process each company must register as a member of the Bluetooth® SIG:
https://www.bluetooth.org/login/register/

> Due to the qualification of the Proteus-III as end product no further Bluetooth® tests are required. The only arising expenses are those for purchasing a BT Declaration ID.

The fees for the Declaration ID depend on your membership status:
https://www.bluetooth.org/en-us/test-qualification/qualification-overview/fees

Refer to the testing laboratory of your choice for further more detailed information regarding the qualification of your product.

21.1 Qualification steps when referencing the Proteus-III

Due to the qualification of the Proteus-III as end product, it can be referenced when starting the qualification process of any product integrating the Proteus-III. To perform the qualification process in a row, we recommend purchasing a Declaration ID before starting the new qualification. Any of the following links was valid at the time this document was created but are subject to change without further notice.
Perform the following steps:

1. Visit *https://www.bluetooth.org/tpg/QLI_SDoc.cfm* .

2. Select "Manage Declarations IDs" .

3. Push the "Purchase a Declaration ID" button and fill in the form.

> The process can be finished once the invoice for the Declaration ID is paid.

To perform the qualification process of a product integrating Proteus-III, please go through the following steps:

1. Visit *https://www.bluetooth.org/tpg/QLI_SDoc.cfm* .

2. Select option "Start the Bluetooth® Qualification Process with NO Required Testing".

3. Enter the QDID (see above) and select the corresponding Proteus-III entry.

4. Select your pre-paid Declaration ID (it can be selected as soon as the Declaration ID has been paid).

5. Follow the subsequent steps to finish the qualification process.

After finishing the process, the product will be listed on the Bluetooth® website.

22 Regulatory compliance information

22.1 Important notice EU

The use of RF frequencies is limited by national regulations. The Proteus-III has been designed to comply with the R&TTE directive 1999/5/EC and the RED directive 2014/53/EU of the European Union (EU).
The Proteus-III can be operated without notification and free of charge in the area of the European Union. However, according to the R&TTE / RED directive, restrictions (e.g. in terms of duty cycle or maximum allowed RF power) may apply.

> Since the module itself is not fused the voltage supply shall be fed from a power source which is class PS2 according to EN 62368-1.

22.2 Important notice FCC

The use of RF frequencies is limited by national regulations. The Proteus-III has been designed to comply with the FCC Part 15.
The Proteus-III can be operated without notification and free of charge in the area of the United States of America. However, according to the FCC Part 15, restrictions (e.g. in terms of maximum allowed RF power and antenna) may apply.

22.3 Conformity assessment of the final product

The Proteus-III is a subassembly. It is designed to be embedded into other products (products incorporating the Proteus-III are henceforward referred to as "final products").
It is the responsibility of the manufacturer of the final product to ensure that the final product is in compliance with the essential requirements of the underlying national radio regulations. The conformity assessment of the subassembly Proteus-III carried out by Würth Elektronik eiSos does not replace the required conformity assessment of the final product.

22.4 Exemption clause

Relevant regulation requirements are subject to change. Würth Elektronik eiSos does not guarantee the accuracy of the before mentioned information. Directives, technical standards, procedural descriptions and the like may be interpreted differently by the national authorities. Equally, the national laws and restrictions may vary with the country. In case of doubt or uncertainty, we recommend that you consult with the authorities or official certification organizations of the relevant countries. Würth Elektronik eiSos is exempt from any responsibilities or liabilities related to regulatory compliance.

Notwithstanding the above, Würth Elektronik eiSos makes no representations and warranties of any kind related to their accuracy, correctness, completeness and/or usability for customer applications. No responsibility is assumed for inaccuracies or incompleteness.

22.5 FCC Compliance Statement

FCC ID: R7T1101102

This device complies with Part 15 of the FCC Rules.
Operation is subject to the following two conditions:
(1) this device may not cause harmful interference, and
(2) this device must accept any interference received, including interference that may cause undesired operation.
(FCC 15.19)

Modifications (FCC 15.21)

Caution: Changes or modifications for this equipment not expressly approved by Würth Elektronik eiSos may void the FCC authorization to operate this equipment.

22.6 IC Compliance Statement

Certification Number: 5136A-1101102
PMN: 1101102
This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.
Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

22.7 FCC and IC requirements to OEM integrators

This module has been granted modular approval. OEM integrators for host products may use the module in their final products without additional FCC/IC (Industry Canada) certification if they meet the following conditions. Otherwise, additional FCC/IC approvals must be obtained.
The host product with the module installed must be evaluated for simultaneous transmission requirements.

- The users manual for the host product must clearly indicate the operating requirements and conditions that must be observed to ensure compliance with current FCC/IC RF exposure guidelines.

- To comply with FCC/IC regulations limiting both maximum RF output power and human exposure to RF radiation, the maximum antenna gain including cable loss in a mobile-only exposure condition must not exceed 6dBi.

- A label must be affixed to the outside of the host product with the following statements:
This device contains FCCID: R7T1101102
This equipment contains equipment certified under ICID: 5136A-1101102

- The final host / module combination may also need to be evaluated against the FCC Part 15B criteria for unintentional radiators in order to be properly authorized for operation as a Part 15 digital device.

- If the final host / module combination is intended for use as a portable device (see classifications below) the host manufacturer is responsible for separate approvals for the SAR requirements from FCC Part 2.1093 and RSS-102.

OEM requirements:

The OEM must ensure that the following conditions are met.

- The Proteus-III will be used at a distance of at least 10 mm.

- End users of products, which contain the module, must not have the ability to alter the firmware that governs the operation of the module. The agency grant is valid only when the module is incorporated into a final product by OEM integrators.

- The end-user must not be provided with instructions to remove, adjust or install the module.

- The Original Equipment Manufacturer (OEM) must ensure that FCC labeling requirements are met. This includes a clearly visible label on the outside of the final product. Attaching a label to a removable portion of the final product, such as a battery cover, is not permitted.

- The label must include the following text:
Contains FCC ID: R7T1101102
The enclosed device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
(i.) this device may not cause harmful interference and
(ii.) this device must accept any interference received, including interference that may cause undesired operation.

  When the device is so small or for such use that it is not practicable to place the statement above on it, the information required by this paragraph shall be placed in a prominent location in the instruction manual or pamphlet supplied to the user or, alternatively, shall be placed on the container in which the device is marketed. However, the FCC identifier or the unique identifier, as appropriate, must be displayed on the device.

- The user manual for the end product must also contain the text given above.
  - Changes or modifications not expressly approved could void the user's authority to operate the equipment.
  - The OEM must ensure that timing requirements according to 47 CFR 15.231(a-c) are met.
  - The OEM must sign the OEM Modular Approval Agreement.
  - The module must be used with only the following approved antenna(s).

22.7.1 Pre-certified antennas

The Proteus-III is pre-certified with the following antennas.

Product	Certified antenna
Proteus-III (2611011024000)	PCB antenna included in the Proteus-III

22.8 ARIB Declaration of conformity

Japanese Radio Law Compliance.
This device is granted pursuant to the Japanese Radio Law.
This device should not be modified (otherwise the granted designation number will become invalid)

ID-Code (Interference provision)

The MAC address of the radio device maintains the format 00:18:DA:xx:xx:xx. The latter part xx:xx:xx of the MAC address coincides with the serial number of the device.

22.8.1 Label

Due to the size of the Proteus-III label, the certification label of the Proteus-III is not placed onto the module label.

2611011024000:

 After integration of the Proteus-III in the end device, the corresponding certification label must be recognized from the outside. Otherwise this information must be referenced on the housing as well as in the user manual. E labeling is allowed.

22.8.2 Certified antennas

The Proteus-III is pre-certified with the following antennas.

Product	Certified antenna
Proteus-III (2611011024000)	PCB antenna included in the Proteus-III

---

[1]Additional, not yet certified, antennas must be re-certified without retesting. Only antenna gain and antenna characteristic diagrams must be specified. Please contact your local field sales engineer (FSE) to get support in certifying your own antenna.

23 Important notes

The following conditions apply to all goods within the wireless connectivity product range of Würth Elektronik eiSos GmbH & Co. KG:

23.1 General customer responsibility

Some goods within the product range of Würth Elektronik eiSos GmbH & Co. KG contain statements regarding general suitability for certain application areas. These statements about suitability are based on our knowledge and experience of typical requirements concerning the areas, serve as general guidance and cannot be estimated as binding statements about the suitability for a customer application. The responsibility for the applicability and use in a particular customer design is always solely within the authority of the customer. Due to this fact, it is up to the customer to evaluate, where appropriate to investigate and to decide whether the device with the specific product characteristics described in the product specification is valid and suitable for the respective customer application or not. Accordingly, the customer is cautioned to verify that the documentation is current before placing orders.

23.2 Customer responsibility related to specific, in particular safety-relevant applications

It has to be clearly pointed out that the possibility of a malfunction of electronic components or failure before the end of the usual lifetime cannot be completely eliminated in the current state of the art, even if the products are operated within the range of the specifications. The same statement is valid for all software sourcecode and firmware parts contained in or used with or for products in the wireless connectivity and sensor product range of Würth Elektronik eiSos GmbH & Co. KG. In certain customer applications requiring a high level of safety and especially in customer applications in which the malfunction or failure of an electronic component could endanger human life or health, it must be ensured by most advanced technological aid of suitable design of the customer application that no injury or damage is caused to third parties in the event of malfunction or failure of an electronic component.

23.3 Best care and attention

Any product-specific data sheets, manuals, application notes, PCN's, warnings and cautions must be strictly observed in the most recent versions and matching to the products firmware revisions. This documents can be downloaded from the product specific sections on the wireless connectivity homepage.

23.4 Customer support for product specifications

Some products within the product range may contain substances, which are subject to restrictions in certain jurisdictions in order to serve specific technical requirements. Necessary information is available on request. In this case, the field sales engineer or the internal sales person in charge should be contacted who will be happy to support in this matter.

23.5 Product improvements

Due to constant product improvement, product specifications may change from time to time. As a standard reporting procedure of the Product Change Notification (PCN) according to the JEDEC-Standard, we inform about major changes. In case of further queries regarding the PCN, the field sales engineer, the internal sales person or the technical support team in charge should be contacted. The basic responsibility of the customer as per section `23.1` and `23.2` remains unaffected. All wireless connectivity module driver software ¨wireless connectivity SDK¨ and it's source codes as well as all PC software tools are not subject to the Product Change Notification information process.

23.6 Product life cycle

Due to technical progress and economical evaluation we also reserve the right to discontinue production and delivery of products. As a standard reporting procedure of the Product Termination Notification (PTN) according to the JEDEC-Standard we will inform at an early stage about inevitable product discontinuance. According to this, we cannot ensure that all products within our product range will always be available. Therefore, it needs to be verified with the field sales engineer or the internal sales person in charge about the current product availability expectancy before or when the product for application design-in disposal is considered. The approach named above does not apply in the case of individual agreements deviating from the foregoing for customer-specific products.

23.7 Property rights

All the rights for contractual products produced by Würth Elektronik eiSos GmbH & Co. KG on the basis of ideas, development contracts as well as models or templates that are subject to copyright, patent or commercial protection supplied to the customer will remain with Würth Elektronik eiSos GmbH & Co. KG. Würth Elektronik eiSos GmbH & Co. KG does not warrant or represent that any license, either expressed or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, application, or process in which Würth Elektronik eiSos GmbH & Co. KG components or services are used.

23.8 General terms and conditions

Unless otherwise agreed in individual contracts, all orders are subject to the current version of the "General Terms and Conditions of Würth Elektronik eiSos Group", last version available at *www.we-online.com*.

24 Legal notice

24.1 Exclusion of liability

Würth Elektronik eiSos GmbH & Co. KG considers the information in this document to be correct at the time of publication. However, Würth Elektronik eiSos GmbH & Co. KG reserves the right to modify the information such as technical specifications or functions of its products or discontinue the production of these products or the support of one of these products without any written announcement or notification to customers. The customer must make sure that the information used corresponds to the latest published information. Würth Elektronik eiSos GmbH & Co. KG does not assume any liability for the use of its products. Würth Elektronik eiSos GmbH & Co. KG does not grant licenses for its patent rights or for any other of its intellectual property rights or third-party rights.

Notwithstanding anything above, Würth Elektronik eiSos GmbH & Co. KG makes no representations and/or warranties of any kind for the provided information related to their accuracy, correctness, completeness, usage of the products and/or usability for customer applications. Information published by Würth Elektronik eiSos GmbH & Co. KG regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof.

24.2 Suitability in customer applications

The customer bears the responsibility for compliance of systems or units, in which Würth Elektronik eiSos GmbH & Co. KG products are integrated, with applicable legal regulations. Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of Würth Elektronik eiSos GmbH & Co. KG components in its applications, notwithstanding any applications-related in-formation or support that may be provided by Würth Elektronik eiSos GmbH & Co. KG. Customer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences lessen the likelihood of failures that might cause harm and take appropriate remedial actions. The customer will fully indemnify Würth Elektronik eiSos GmbH & Co. KGand its representatives against any damages arising out of the use of any Würth Elektronik eiSos GmbH & Co. KG components in safety-critical applications.

24.3 Trademarks

AMBER wireless is a registered trademark of Würth Elektronik eiSos GmbH & Co. KG. All other trademarks, registered trademarks, and product names are the exclusive property of the respective owners.

24.4 Usage restriction

Würth Elektronik eiSos GmbH & Co. KG products have been designed and developed for usage in general electronic equipment only. This product is not authorized for use in equipment where a higher safety standard and reliability standard is especially required or where

25 License terms

This License Terms will take effect upon the purchase and usage of the Würth Elektronik eiSos GmbH & Co. KG wireless connectivity products. You hereby agree that this license terms is applicable to the product and the incorporated software, firmware and source codes (collectively, "Software") made available by Würth Elektronik eiSos in any form, including but not limited to binary, executable or source code form.

The software included in any Würth Elektronik eiSos wireless connectivity product is purchased to you on the condition that you accept the terms and conditions of this license terms. You agree to comply with all provisions under this license terms.

25.1 Limited license

Würth Elektronik eiSos hereby grants you a limited, non-exclusive, non-transferable and royalty-free license to use the software and under the conditions that will be set forth in this license terms. You are free to use the provided Software only in connection with one of the products from Würth Elektronik eiSos to the extent described in this license terms. You are entitled to change or alter the source code for the sole purpose of creating an application embedding the Würth Elektronik eiSos wireless connectivity product. The transfer of the source code to third parties is allowed to the sole extent that the source code is used by such third parties in connection with our product or another hardware provided by Würth Elektronik eiSos under strict adherence of this license terms. Würth Elektronik eiSos will not assume any liability for the usage of the incorporated software and the source code. You are not entitled to transfer the source code in any form to third parties without prior written consent of Würth Elektronik eiSos.

You are not allowed to reproduce, translate, reverse engineer, decompile, disassemble or create derivative works of the incorporated Software and the source code in whole or in part. No more extensive rights to use and exploit the products are granted to you.

25.2 Usage and obligations

The responsibility for the applicability and use of the Würth Elektronik eiSos wireless connectivity product with the incorporated Firmware in a particular customer design is always solely within the authority of the customer. Due to this fact, it is up to you to evaluate and investigate, where appropriate, and to decide whether the device with the specific product characteristics described in the product specification is valid and suitable for your respective application or not.

You are responsible for using the Würth Elektronik eiSos wireless connectivity product with the incorporated Firmware in compliance with all applicable product liability and product safety laws. You acknowledge to minimize the risk of loss and harm to individuals and bear the risk for failure leading to personal injury or death due to your usage of the product.

Würth Elektronik eiSos' products with the incorporated Firmware are not authorized for use in safety-critical applications, or where a failure of the product is reasonably expected to cause severe personal injury or death. Moreover, Würth Elektronik eiSos' products with the incorporated Firmware are neither designed nor intended for use in areas such as military, aerospace, aviation, nuclear control, submarine, transportation (automotive control, train control, ship control), transportation signal, disaster prevention, medical, public information network etc. You shall inform Würth Elektronik eiSos about the intent of such usage before

design-in stage. In certain customer applications requiring a very high level of safety and in which the malfunction or failure of an electronic component could endanger human life or health, you must ensure to have all necessary expertise in the safety and regulatory ramifications of your applications. You acknowledge and agree that you are solely responsible for all legal, regulatory and safety-related requirements concerning your products and any use of Würth Elektronik eiSos' products with the incorporated Firmware in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by Würth Elektronik eiSos. YOU SHALL INDEMNIFY WÜRTH ELEKTRONIK EISOS AGAINST ANY DAMAGES ARISING OUT OF THE USE OF WÜRTH ELEKTRONIK EISOS' PRODUCTS WITH THE INCORPORATED FIRMWARE IN SUCH SAFETY-CRITICAL APPLICATIONS.

25.3 Ownership

The incorporated Firmware created by Würth Elektronik eiSos is and will remain the exclusive property of Würth Elektronik eiSos.

25.4 Firmware update(s)

You have the opportunity to request the current and actual Firmware for a bought wireless connectivity Product within the time of warranty. However, Würth Elektronik eiSos has no obligation to update a modules firmware in their production facilities, but can offer this as a service on request. The upload of firmware updates falls within your responsibility, e.g. via ACC or another software for firmware updates. Firmware updates will not be communicated automatically. It is within your responsibility to check the current version of a firmware in the latest version of the product manual on our website. The revision table in the product manual provides all necessary information about firmware updates. There is no right to be provided with binary files, so called "Firmware images", those could be flashed through JTAG, SWD, Spi-Bi-Wire, SPI or similar interfaces.

25.5 Disclaimer of warranty

THE FIRMWARE IS PROVIDED "AS IS". YOU ACKNOWLEDGE THAT WÜRTH ELEKTRONIK EISOS MAKES NO REPRESENTATIONS AND WARRANTIES OF ANY KIND RELATED TO, BUT NOT LIMITED TO THE NON-INFRINGEMENT OF THIRD PARTIES' INTELLECTUAL PROPERTY RIGHTS OR THE MERCHANTABILITY OR FITNESS FOR YOUR INTENDED PURPOSE OR USAGE. WÜRTH ELEKTRONIK EISOS DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT RELATING TO ANY COMBINATION, MACHINE, OR PROCESS IN WHICH THE WÜRTH ELEKTRONIK EISOS' PRODUCT WITH THE INCORPORATED FIRMWARE IS USED. INFORMATION PUBLISHED BY WÜRTH ELEKTRONIK EISOS REGARDING THIRD-PARTY PRODUCTS OR SERVICES DOES NOT CONSTITUTE A LICENSE FROM WÜRTH ELEKTRONIK EISOS TO USE SUCH PRODUCTS OR SERVICES OR A WARRANTY OR ENDORSEMENT THEREOF.

25.6 Limitation of liability

Any liability not expressly provided by Würth Elektronik eiSos shall be disclaimed.
You agree to hold us harmless from any third-party claims related to your usage of the Würth Elektronik eiSos' products with the incorporated Firmware, software and source code. Würth Elektronik eiSos disclaims any liability for any alteration, development created by you or your customers as well as for any combination with other products.

25.7 Applicable law and jurisdiction

Applicable law to this license terms shall be the laws of the Federal Republic of Germany. Any dispute, claim or controversy arising out of or relating to this license terms shall be resolved and finally settled by the court competent for the location of Würth Elektronik eiSos' registered office.

25.8 Severability clause

If a provision of this license terms is or becomes invalid, unenforceable or null and void, this shall not affect the remaining provisions of the terms. The parties shall replace any such provisions with new valid provisions that most closely approximate the purpose of the terms.

25.9 Miscellaneous

Würth Elektronik eiSos reserves the right at any time to change this terms at its own discretion. It is your responsibility to check at Würth Elektronik eiSos homepage for any updates. Your continued usage of the products will be deemed as the acceptance of the change.
We recommend you to be updated about the status of new firmware and software, which is available on our website or in our data sheet and manual, and to implement new software in your device where appropriate.
By ordering a wireless connectivity product, you accept this license terms in all terms.

List of Figures

List of Tables

WÜRTH ELEKTRONIK

more than you expect

Internet of Things

Monitoring & Control

Automated Meter Reading

Contact:
Würth Elektronik eiSos GmbH & Co. KG
Division Wireless Connectivity & Sensors

Max-Eyth-Straße 1
74638 Waldenburg
Germany

Tel.: +49 651 99355-0
Fax.: +49 651 99355-69
www.we-online.com/wireless-connectivity