PROTEUS-E REFERENCE MANUAL

2612011024000

VERSION 0.3

NOVEMBER 22, 2021

****

MUST READ

Check for firmware updates

Before using the product make sure you use the most recent firmware version, data sheet and user manual. This is especially important for Wireless Connectivity products that were not purchased directly from Würth Elektronik eiSos. A firmware update on these respective products may be required.

We strongly recommend to include in the customer system design, the possibility for a firmware update of the product.

Revision history

Manual version	FW version	HW version	Notes	Date
1.0	1.0.0	1.0	• Initial release	December 2021

⋆ 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.
DSSS	Direct sequence spread spectrum	Technique to spread a message on the radio
DTM	Direct test mode	Mode to test Bluetooth® specific RF settings.
EV (Board)	Evaluation (Board)	Proteus-e populated on motherboard with USB interface for test and evaluation purpose.
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 Bluetooth® LE encryption and authentication
LPM	Low power mode	Mode for efficient power consumption.
MAC		MAC address of the module.
MPS	Maximum payload size	The maximum size of the payload, that can be transmitted/received using one Bluetooth® LE transaction.
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.
SPI	Serial Peripheral Interface	Allows the serial communication with the module.
UART	Universal Asynchronous Receiver Transmitter	Allows the serial communication with the module.
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.
[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-e is a radio module for wireless communication between devices such as control systems, remote controls, sensors etc. On the basis of Bluetooth® LE 5.1, it offers a fast and secure data transmission of data packages between two parties (point to point topology).

The Bluetooth® LE standard itself offers very high configurability, making the module suitable for a wide range of applications. For example, the module can be configured to communicate with external sensors, can use security configurations, and can be optimized for low power consumption. To fulfill the specific requirements of the application, a tailored firmware can be developed on the basis of the Proteus-e hardware (see chapter 13).

Ultra small dimensions of 7 x 9 mm including a strongly miniaturized PCB antenna make the Proteus-e ideal for small form factor design. It is possible to connect an external antenna, in case higher radio ranges are required.
A serial interface (UART) is available for communication with the host system.

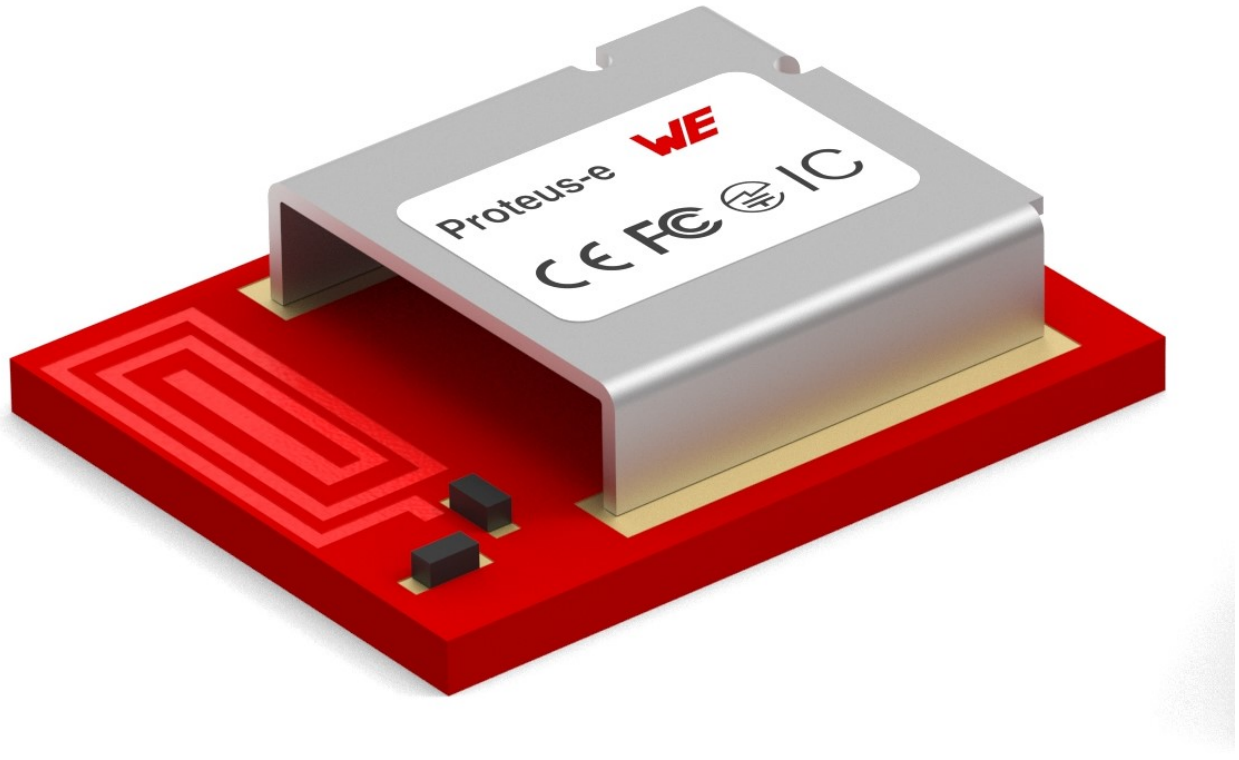


Figure 1: Proteus-e

1.1.1. Key features

The Proteus-e offers the following key features:

SPP-like connection-based secured data transmission: The Proteus-e firmware implements an SPP-like Bluetooth® LE profile that allows the bidirectional data transmission between two Proteus-e and/or to other Bluetooth® LE devices implementing the SPP-like profile. Secured connections allow the transmission of encrypted and authenticated data.

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

Advanced customization capabilities: The configurable UUID and the appearance of the Bluetooth® LE profile, enable to personalize the Proteus-e to fuse with the user's end product. For more information see chapter 12.

Custom advertising data: Furthermore, the complete content of the advertising and scan response packet can be customer defined. With this, any beacon function can be realized. For more information see chapter 5.7.

Low power position sensing solutions: The TX power of any Proteus-e is always transmitted during advertising. With this, distance estimation and position sensing solutions can be realized conveniently.

Fast serial interface: The Proteus-e offers a UART-interface to communicate with a host using a user-defined baud rate and a simple command interface. Furthermore, the firmware provides the so called "transparent mode" (see chapter 10), which is formerly known as "peripheral only mode" in Proteus-I,-II and -III. This mode offers a transparent UART interface such that an easy adaption of already existing custom hardware with the Bluetooth® LE interface is enabled.

Latest microprocessor generation provided by Nordic Semiconductor nRF52 series: The heart of the Proteus-e is a Bluetooth® LE chip of the nRF52 series offering high performance values combined with low power consumption. It is a 64 MHz ARM Cortex-M4 CPU with 192kB flash + 24kB RAM and up to 4 dBm 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.

Flexible wired interfacing: The Proteus-e 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 and GPIO.

Additional Bluetooth® 5 radio modes: The Proteus-e provides the advanced radio modes 2 MBit mode for faster data transmission. For more information, see chapter 5.11.

1.1.2. Connectivity

The Bluetooth® LE standard allows to setup a network with various Bluetooth® LE devices from different manufacturers. To be able to communicate with Proteus-e devices, the SPP-like profile must be known and implemented by all network participants.
The advanced developer guide of Proteus-e (application note ANR024 [2]) contains the design data of the SPP-like profile, to implement it for example in smart phone apps.

1.2. Block diagram

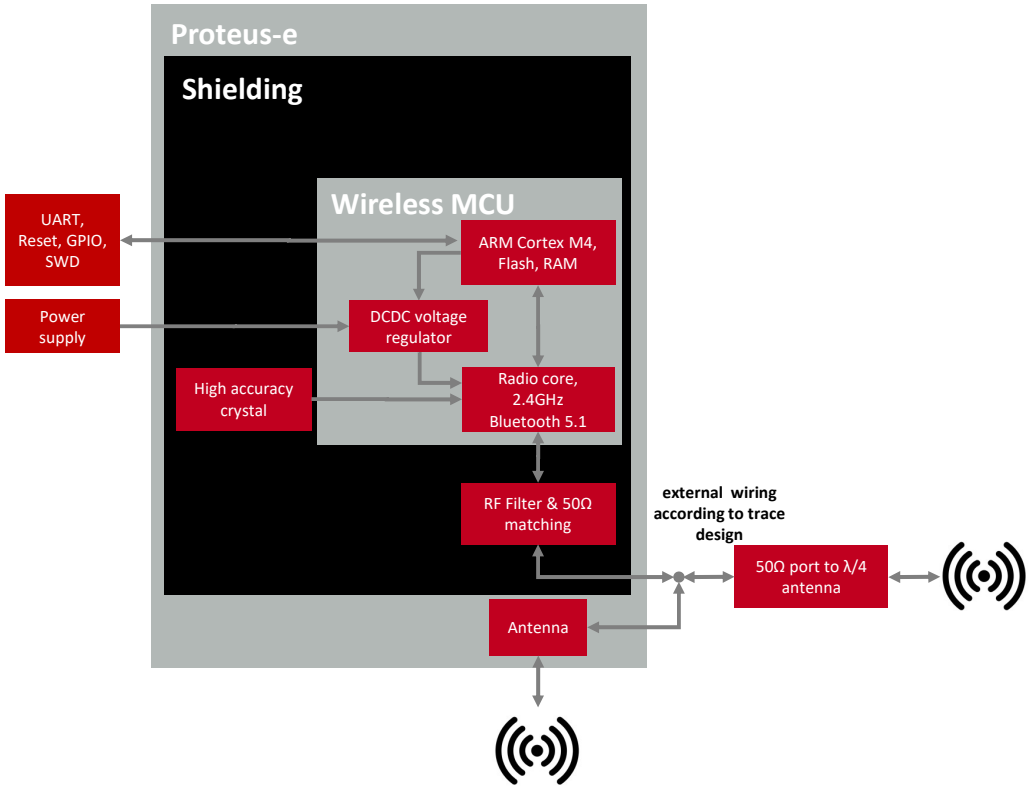


Figure 2: Block diagram of the module

1.3. Ordering information

WE order code	Description
2612011024000	Proteus-e Bluetooth® LE Module, Tape & Reel
2612019024001	Proteus-e Evaluation Board

Table 1: Ordering information

2. Electrical specifications

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

2.1. Recommended operating conditions

Parameter	Min.	Typ.	Max.	Unit
Ambient temperature	-40	25	85	℃
Supply voltage (VDD)	1.8[1]	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.

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

2.2. Absolute maximum ratings

Parameter	Min.	Max.	Unit
Supply voltage (VDD)	-0.3	+3.9	V
Voltage on any digital pin (VDD<3.6V)	-0.3	VDD+0.3	V
Voltage on any digital pin (VDD≥3.6V)	-0.3	3.9	V
Input RF level		10	dBm
Flash endurance	10 000		Write/erase cycles

Table 3: Absolute maximum ratings

[1] Power fail comparator is set to 1.8V to avoid flash fail due to voltage drop.

2.3. Power consumption

2.3.1. Static

Parameter	Power	Test conditions	Value	Unit
TX Current consumption	RF_TXPower = 4	Transmitter only with DC/DC converter from nRF52 data sheet, CPU current not included	8	mA
		Full module current consumption	9.3	mA
	RF_TXPower = 0	Transmitter only with DC/DC converter from nRF52 data sheet, CPU current not included	5.8	mA
		Full module current consumption	7.1	mA

Table 4: Current consumption - transmitting

Parameter	Test conditions	Value	Unit
RX Current consumption	Transmitter only with DC/DC converter from nRF52 data sheet, CPU current not included	6.1	mA
	Full module current consumption	6.8	mA

Table 5: Current consumption - receiving

Parameter	Test conditions	Value	Unit
Current consumption	Sleep (system off mode)	0.3	µA
Current consumption reduction through CMD_UARTDISABLE_REQ		TBD	µA

Table 6: Current consumption - low power

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

Sleep current is significantly increasing for temperatures above 30 °C.

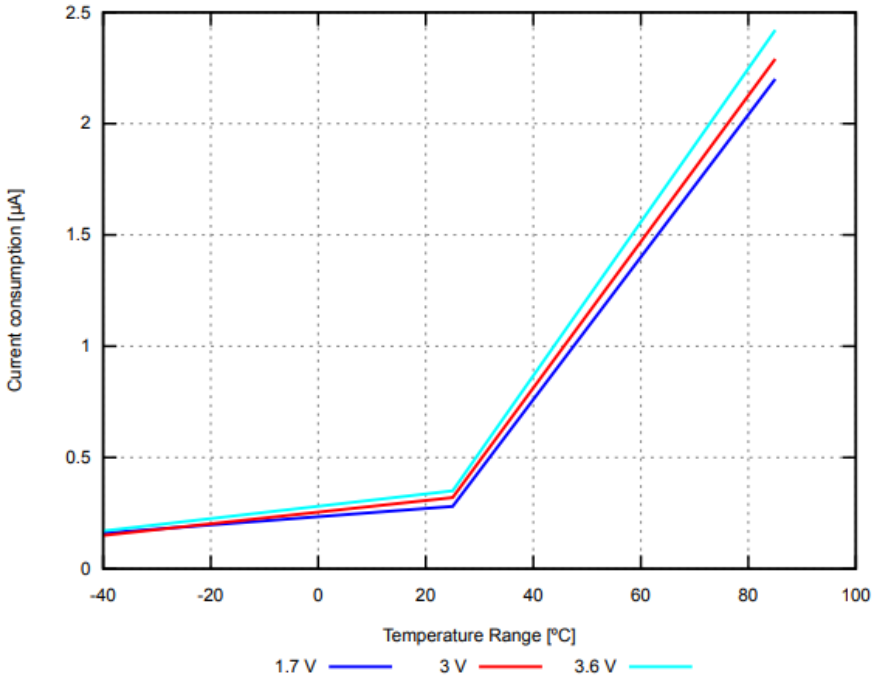


Figure 3: Sleep current (no RAM retention, wake on reset) over operating temperature range

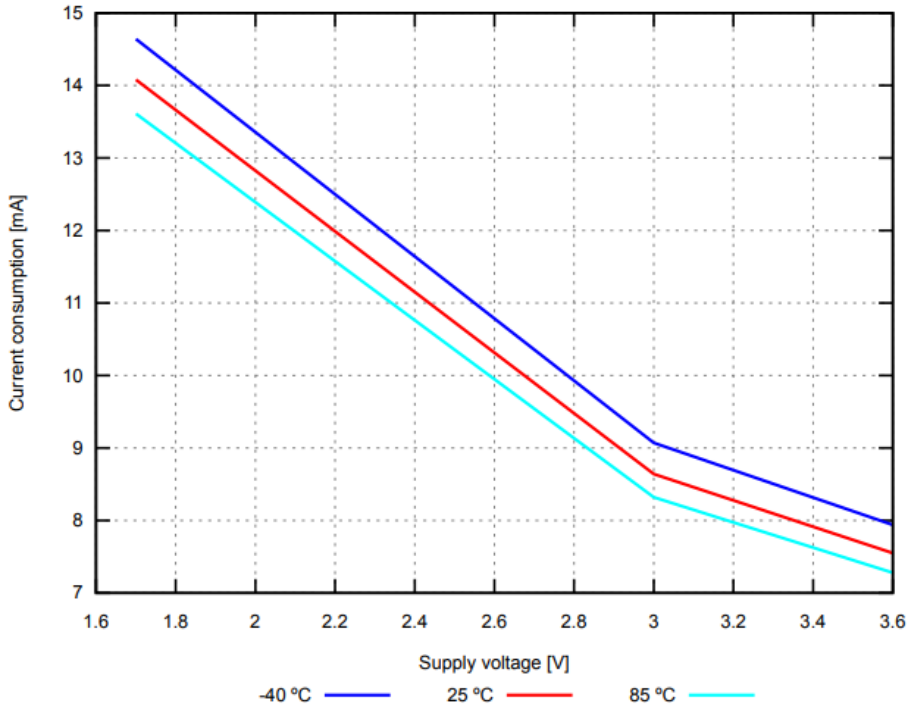


Figure 4: Radio transmitting @ 4 dBm output power, 1 Mbps Bluetooth® LE mode, Clock = HFXO, Regulator = DC/DC (typical values)

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 5). Currents and state durations are dependent on the configuration of the module. In this state the module transmits the advertising packets on the three 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/*.

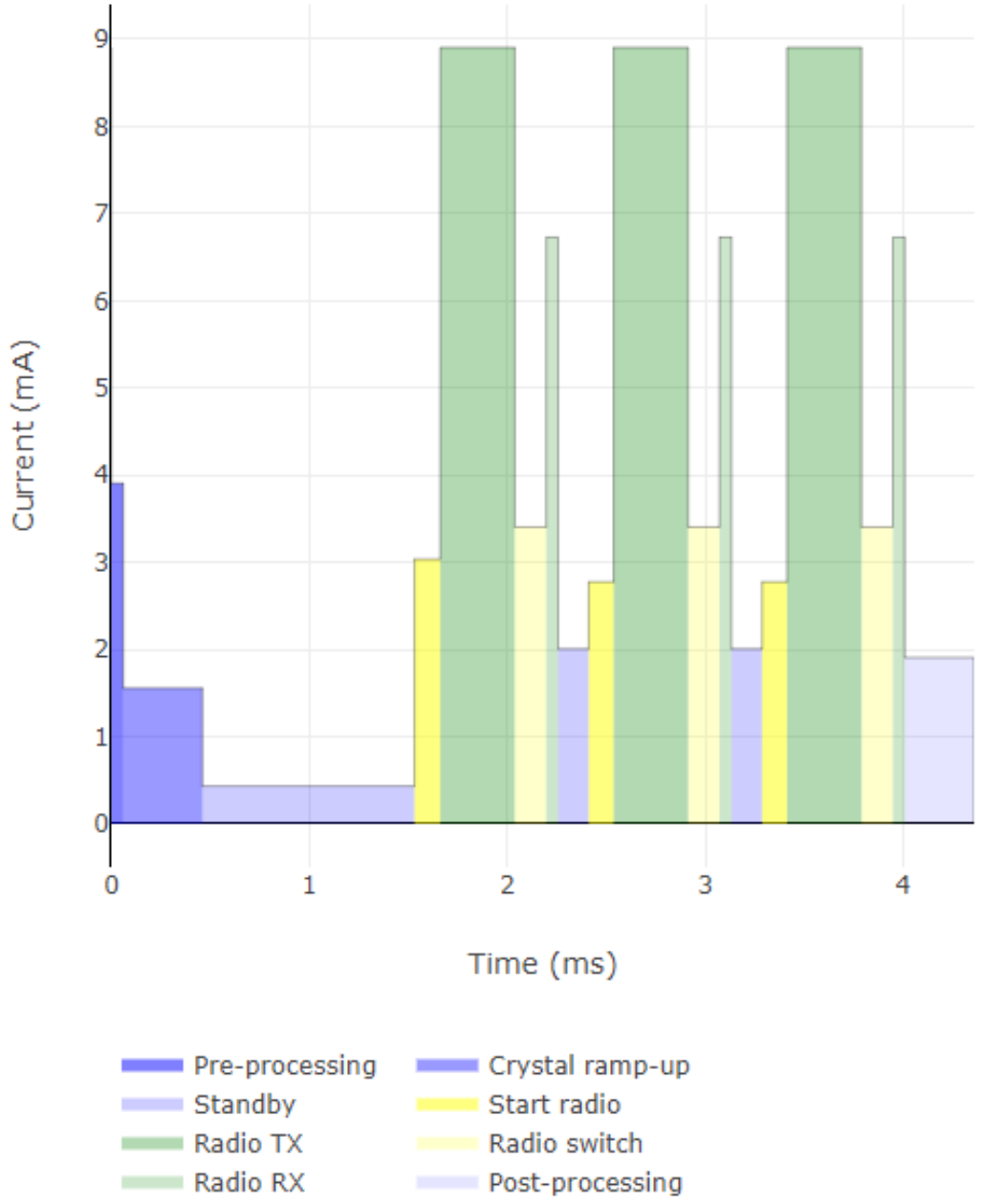


Figure 5: Current consumption calculation in advertising mode with 40ms advertising interval with 4 dBm output power, UART disabled

2.4. Radio characteristics

Parameter	Min.	Max.	Unit
RSSI accuracy valid range (±2dB)	-90	-20	dBm

Table 7: RSSI accuracy

Parameter	Value	Unit
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 8: Timing

Parameter	Test conditions	Value	Unit
Output power	`RF_TXPower` = 4, conducted	+4	dBm
	`RF_TXPower` = 4, radiated	-4	dBm
Input sensitivity	Conducted, BER=1E-3, 1Mbps	-93	dBm
	Conducted, BER=1E-3, 1Mbps, LDO mode	-97	dBm
	Radiated, BER=1E-3, 1Mbps	-85	dBm

Table 9: Transmit and receive power

All transmit and receive power levels are measured on the evaluation board. The values already include losses of transitions from module to motherboard to SMA or modules PCB antenna. They are realistic values for the end application.
Sensitivity in the table above is stated for the common used bit error rate of 0.1%. In the table below the sensitivity is stated for a packet error rate of 1% with a payload length of 38 byte at different data rates. The PER 1% is a harder criteria resulting in 2 dBm less sensitivity.

Parameter	Test conditions	Value	Unit
Input sensitivity	1Mbit Phy, PER 1%	-91	dBm
	2Mbit Phy, PER 1%	-88	dBm

Table 10: Sensitivity at different data rates

2.5. Pin characteristics

Specifications from nRF52 data sheet are reported here below.

Parameter	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 ≥1.7V	1	2	4	mA
Current at VSS+0.4 V, output set low, high drive, VDD ≥ 2.7 V	6	10	15	mA
Current at VSS+0.4 V, output set low, high drive, VDD ≥ 1.7 V	3			mA
Current at VDD-0.4 V, output set high, standard drive, VCC ≥1.7V	1	2	4	mA
Current at VDD-0.4 V, output set high, high drive, VDD ≥ 2.7 V	6	9	14	mA
Current at VDD-0.4 V, output set high, high drive, VDD ≥ 1.7 V	3			mA
Internal pull-up resistance	11	13	16	kΩ
Internal pull-down resistance	11	13	16	kΩ

Table 11: Pin characteristics

3. Pinout

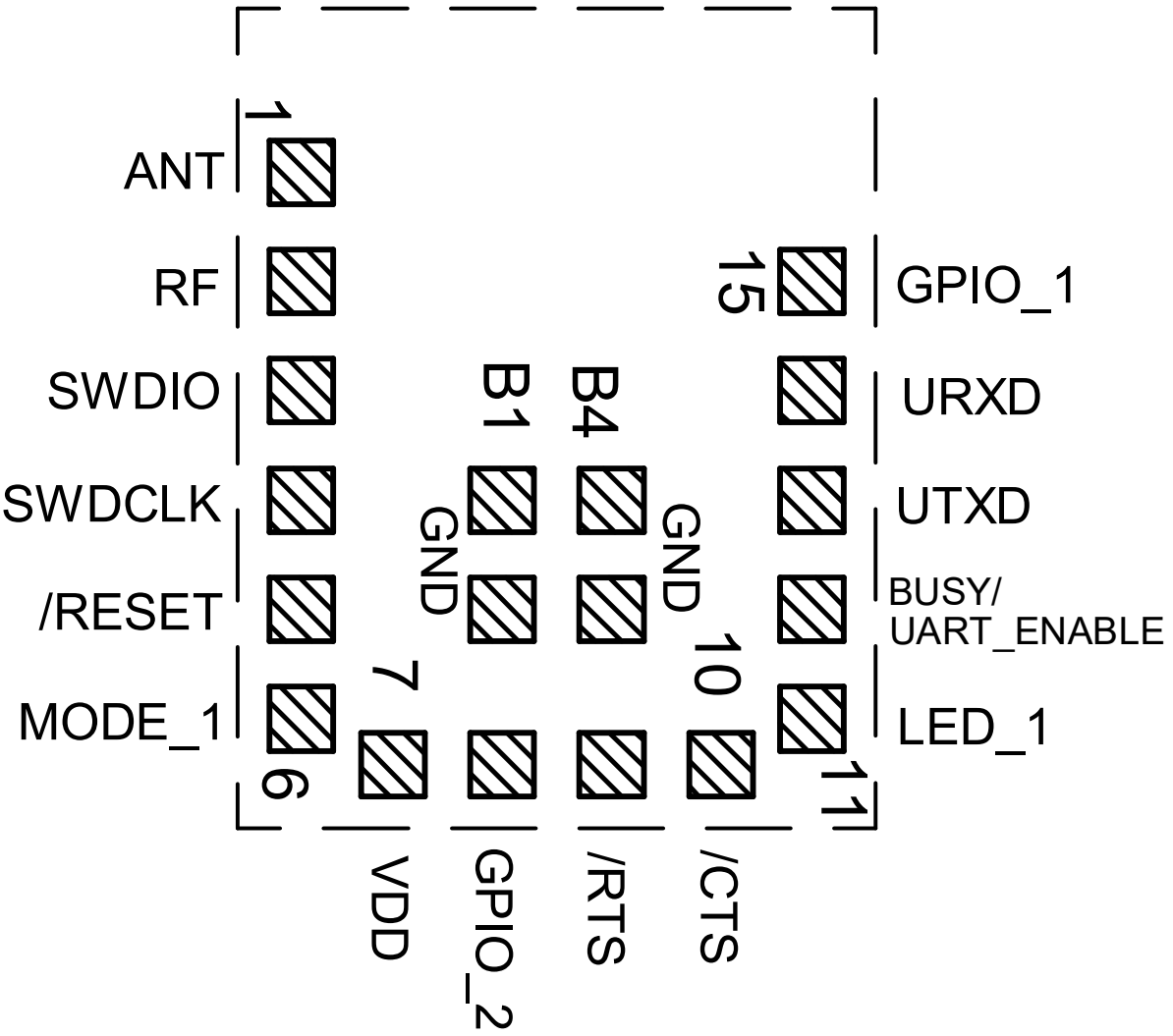


Figure 6: Pinout (top view)

No	µC Pin	Designation	I/O	Description
1		*ANT*	I/O	RF connection to PCB antenna. (see chapter 4.2)
2		*RF*	I/O	50Ω RF connection through radio front end to transceiver part of chipset. (see chapter 4.2)
3		*SWDIO*	Input	Serial wire input/output (SWD Interface). Uses internal pull up resistor. Do not connect if not needed.
4		*SWDCLK*	Input	Serial wire clock (SWD Interface). Uses internal pull down resistor. Do not connect if not needed.
5	P0.21	*/RESET*	Input	Reset pin. A low signal resets the module. Uses internal pull up resistor.
6	P0.12	*MODE_1*	Input	Operation mode pin with internal pull down resistor[1] during start-up. Low level or open: Command mode. High level: Transparent mode. Do not connect if not needed.
7		*VDD*	Supply	Supply voltage
8	P0.05	*GPIO_2*	GPIO	Pin for remote GPIO access. Do not connect, if not needed.
9	P0.04	*/RTS*	Output	/RTS signal, if flow control is enabled. Static low, otherwise. Do not connect if not needed.
10	P0.14	*/CTS*	Input	/CTS signal, if flow control is enabled. Using internal pull down[1], otherwise. Do not connect if not needed.
11	P0.00/XL1[2]	*LED_1*	Output	Indicates the module state (active high). Do not connect if not needed.
12	P0.01/XL2[2]	*BUSY / UART_ENABLE*	Input / Output	Shared pin. In Transparent mode (see chapter 10.3.1) this is an output pin, that indicates if the module is busy with data transmission. In Command mode this is an input pin with pull-up, that wakes up the UART, in case the UART has been disabled before using the CMD_UARTDISABLE_REQ command. To do so, apply a falling edge, holding the line low for at least 10ms before applying a rising edge and holding it high for at least 10ms. Do not connect if not needed.

Table 12: 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] Pins available to connect an external crystal in custom firmware. The standard firmware of Proteus-e does not implement this function.

No	µC Pin	Designation	I/O	Description
13	P0.16	*UTXD*	Output	UART (Transmission)
14	P0.18	*URXD*	Input	UART (Reception). Uses internal pull up resistor[1].
15	P0.20	*GPIO_1*	GPIO	Pin for remote GPIO access. Do not connect, if not needed.
B1		*GND*	Supply	Ground
B2		*GND*	Supply	Ground
B3		*GND*	Supply	Ground
B4		*GND*	Supply	Ground

Table 13: Pinout, second part

4. Quick start

4.1. Minimal pin connections

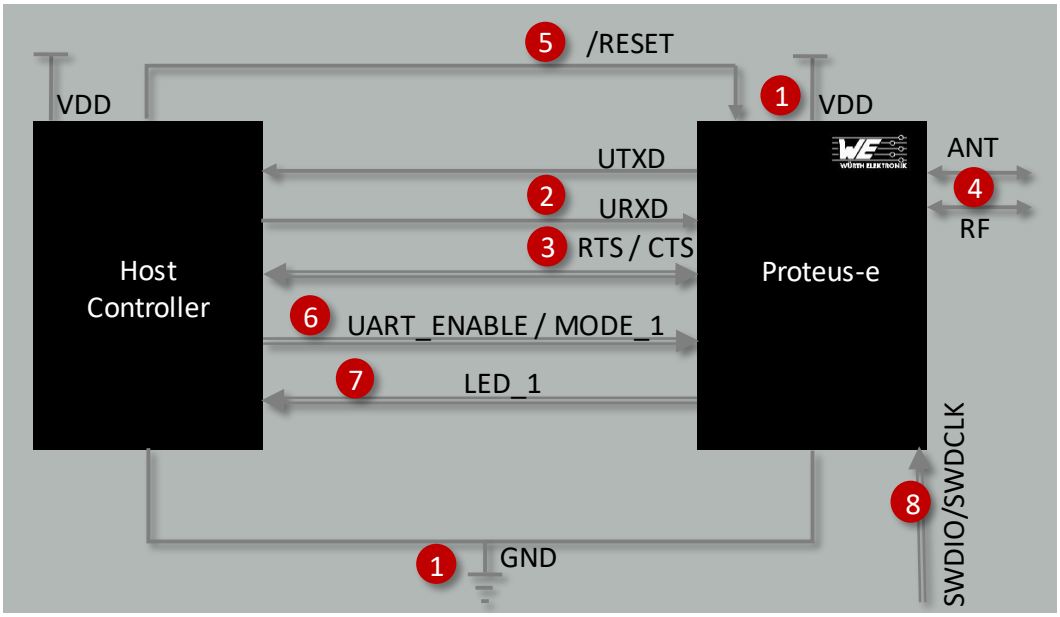


Figure 7: Minimal pin connections

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

1. Supply voltage and ground
   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. UART flow control
   In case of UART baudrates higher than 115.2 kbaud, the UART flow control is activated automatically. For lower data rates, the flow control is inactive per default. If activated the */RTS* and */CTS* pins must be connected to the host controller.

4. Antenna connection
   The antenna configuration must be performed. See chapter 4.2.

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

6. (Optional) Enable/Disable UART and mode selection

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

- In case the transparent mode or the feature of switching off the UART in command mode shall be used, connect the shared pin *BUSY/UART_ENABLE* pin to the host controller.

7. (Optional) Status indication
   Connect the *LED_1* pin to the host controller to allow easy indication of the status.

8. (Optional) Flash and debug interface
   In case of custom firmware development, it is recommended to additionally have the pins *SWDIO* and *SWDCLK* accessible in order to support a fail-safe update of firmware. 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 2.54 mm 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 `Pinout` for details on all pins. Please refer to the Proteus-e evaluation board schemes for a reference design `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-e's smart antenna configuration allows the user to choose between two antenna options. Detailed description on how to use them and what to consider for certification aspects is described in chapter 17:

4.2.1. On-board PCB antenna

The Proteus-e has an on-board PCB antenna optimized for strong miniaturization operating in the 2.4 GHz frequency band. To use this integrated antenna, it has to be connected to the radio chip by connecting the pins *RF* and *ANT*.

4.2.2. External antenna

For applications that use an external antenna, the Proteus-e provides a 50 Ω RF signal on pin *RF* of the module. In this configuration, pin *ANT* of the module has to be left open and pin *RF* has to be connected to the external antenna via 50 Ω feed line.

The use cases for the integrated antenna are miniaturization and re-use of module certifications for the end-application. An external antenna is normally used to increase the achievable radio range, at the cost of more space needed in the device/application. Also, an external antenna could be needed to fit the specific application environment of the module, for example when a metal housing is used.

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` (0x02 41 02 00 01 01 41) 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.

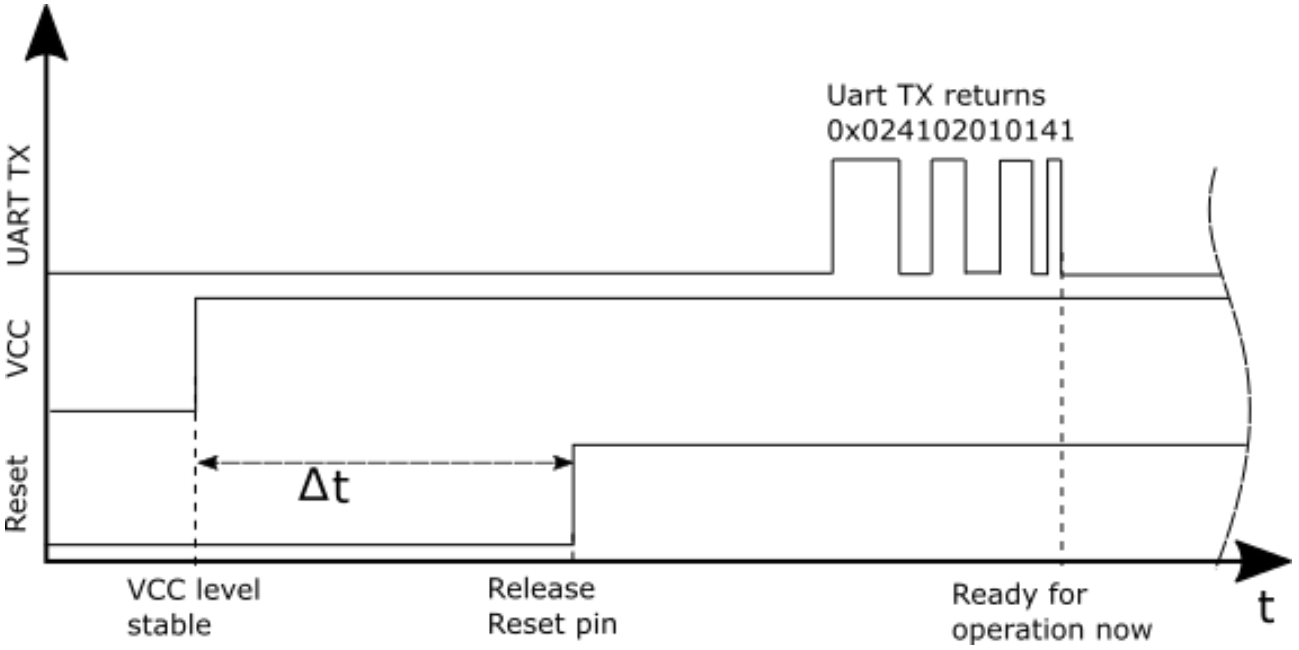


Figure 8: Power up

4.4. Quickstart example

The quick start example in this chapter demonstrates how the UART communication with the Proteus-e works in command mode. In case the host sends a request message (i.e. `CMD_GET_REQ`) to the radio module, it responds with an confirmation message (i.e. `CMD_GET_CNF`). In this example several parameters of the radio module are requested and the device name is configured.

Quick start examples demonstrating the connection setup and data transmission via radio can be found in the application note ANR025 [3].

The below commands are in hexadecimal notation. The arrow in the left column describes, whether it's a message from host to radio module, or vice versa. A request command is always sent from host to module (⇒). An indication, confirmation or response message is always sent from module to host (⇐).

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

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

2. Request the `FS_BTMAC` and `FS_SerialNumber` of the module.

Info	Module
⇒ Request `CMD_GET_REQ` with settings index 4	02 10 01 00 04 17
⇐ Response `CMD_GET_CNF`: `FS_BTMAC` 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 16	02 10 01 00 10 03
⇐ Response `CMD_GET_CNF`: `FS_SerialNumber` is 0x55 0x00 0x00 0xDA 0x18 0x00	02 50 04 00 00 55 00 00 57

3. Set the `RF_DeviceName` of the module to "Hello World" (0x48 65 6C 6C 6F 20 57 6F 72 6C 64).

Info	Module
⇒ Request CMD_SET_REQ with settings index 2	02 11 0C 00 02 48 65 6C 6C 6F 20 57 6F 72 6C 64 3D
⇐ Response CMD_SET_CNF: Successfully modified the setting	02 51 01 00 00 52
⇐ Response CMD_GETSTATE_CNF: Module re-started in ACTION_IDLE mode	02 41 02 00 01 01 41

5. Functional description

5.1. Operation modes

The Proteus-e module acts as a slave and can be fully controlled by an external host. The Proteus-e supports the following operating modes:

- The **command mode**, where the Proteus-e can be controlled by the host controller via commands. The command mode allows to use all functions of the radio module. Functions, like data transmission or configuration tasks, can be triggered by predefined commands (see chapter 7) that are sent as telegrams over the UART interface.
- The **transparent mode** (see chapter Transparent mode) provides a transparent UART interface. Data transmission can be done by the host without using any commands.

5.2. Radio module states

The Proteus-e 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 9.
When the Proteus-e is powered up, it starts in ACTION_IDLE state. In this state the module advertises, such that other devices in range can detect it and connect to it.
The ACTION_CONNECTED state can be entered by getting a connection request from another Bluetooth® LE device. In this case, it stops advertising and data can be transmitted and received to/from the connected Bluetooth® LE device. This state remains active until the module disconnects itself, or a disconnection request from the connected remote device is received.
When disconnecting, the module goes to ACTION_IDLE state and starts advertising again to be ready for the next connection setup.

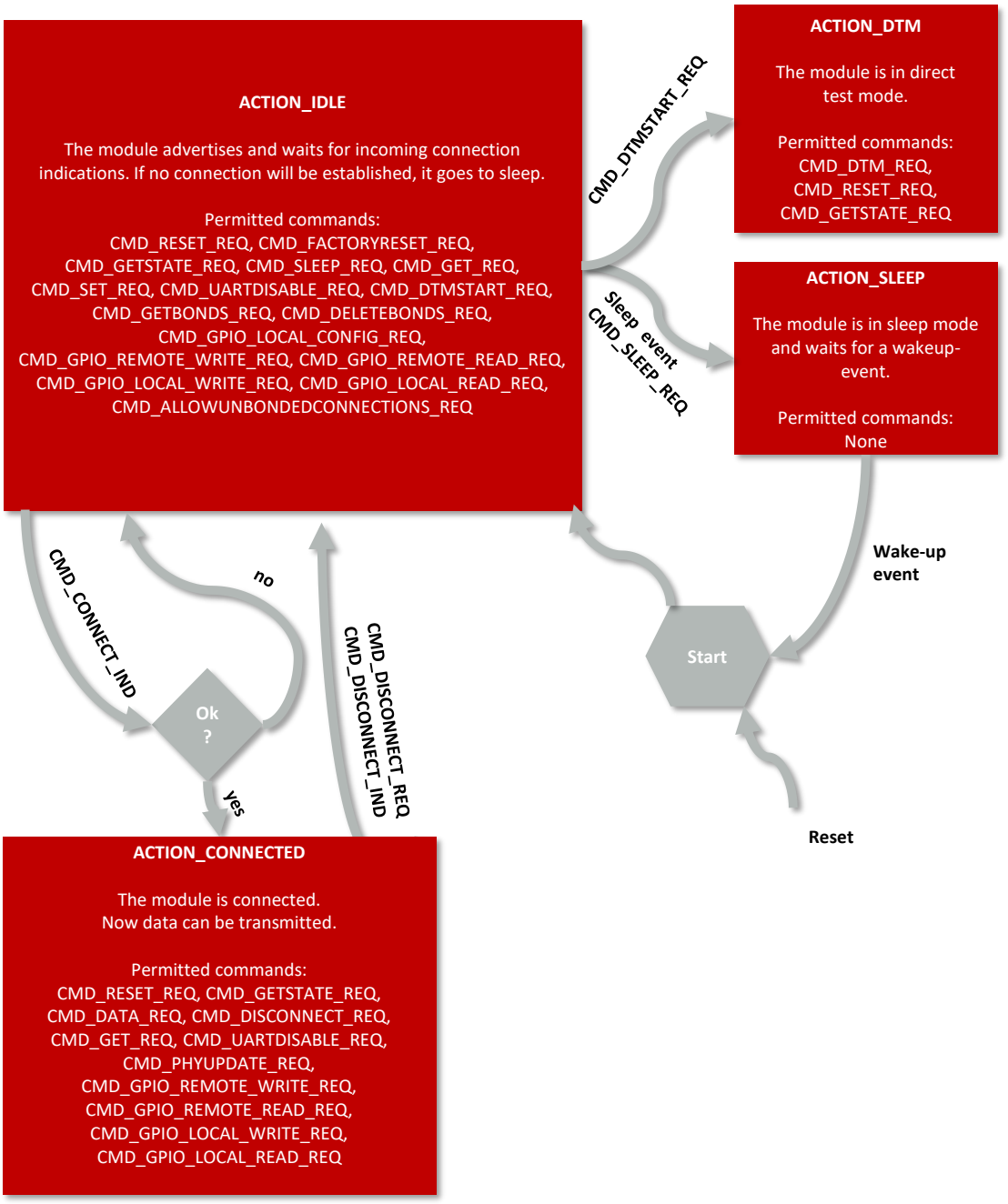


Figure 9: State overview

5.3. State indication using the LED pins

The pin *LED_1* of the Proteus-e can be used to determine the module state. The states described in Figure 9 result in the following pin behavior. The pins on the Proteus-e are active high.

State	*LED_1*
ACTION_IDLE	Blinking (On for 200 ms, Off for 2800 ms)
ACTION_CONNECTED	Blinking (On for 250 ms, Off for 250 ms)
	Static On (as soon as the channel was opened successfully, see CMD_CHANNELOPEN_RSP)
ACTION_SLEEP	Off
ACTION_DTM	Off

Table 14: LED behavior of the Proteus-e

5.4. Sleep mode

Especially for battery-powered devices the ACTION_SLEEP mode (system-off mode) supports very low power consumption. It can be entered by sending the command CMD_SLEEP_REQ to the module. As response, the module will send a CMD_SLEEP_CNF and then enter the ACTION_SLEEP mode.
In ACTION_SLEEP mode the UART is disabled. Thus 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. The GPIO pins *GPIO_1* and *GPIO_2* are set to input without pull resistor during the sleep period.
The ACTION_SLEEP mode can be entered only if the module is in state ACTION_IDLE, that means no peer device is connected via radio.
To leave the ACTION_SLEEP mode and enter ACTION_IDLE state again, the module has to be woken up by a pin reset; apply a low signal to the *RESET* pin for at least 5 ms 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 again.

5.5. Identification of a Proteus-e device on the radio

The Proteus-e can be identified on the radio interface by its 6 Bytes long Bluetooth®-conform MAC address FS_BTMAC, which is part of the data package sent during advertising in ACTION_IDLE mode.
To simplify the identification of Proteus-e devices on the Bluetooth® LE interface a short user-defined name (see user setting 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 module's serial number FS_SerialNumber.

5.6. Connection based data transmission

In the Bluetooth® LE standard the data transmission typically is connection based. A connection between two devices can be secured or unsecured (see user setting `RF_SecFlags`). The connection setup is triggered by the central device and consists of several steps, that must be run sequentially:

1. Physical connection establishment
   A physical connection has to be established first. Therefore, a central device (i.e. smart phone) has to connect to the Proteus-e which runs as peripheral.

2. Pairing process (optional, in case the user setting `RF_SecFlags` has been set)
   The authentication and exchange of encryption information is part of the pairing process. The central device must request at least the same security level to access the characteristics of the Proteus-e.

In case the peripheral device has enabled a security mode, but the central device goes on with the next steps without placing the pairing request, the peripheral device disconnects immediately as the required security level is not achieved. The same holds, if the central device places a bonding request with lower security level than required by the peripheral device.

3. Exchange of the maximum transmission unit (MTU) (optional)
   The maximum transmission unit can be increased to allow the transmission of larger data packets. The Proteus-e allows an MTU of up to 247 bytes, which results in a maximum payload size (MPS) of 243 bytes. Not selecting a higher MTU will use the Bluetooth® LE 4.0 default MTU which results in a MPS of 19 bytes, but will be compatible to pre Bluetooth® LE 4.2 devices.

4. Discover the characteristics of the Proteus-e SPP-like profile
   The characteristics offered by the Proteus-e have to be discovered by the central.

5. Notification enable
   To transmit data from the peripheral to the central, the central must enable the notifications on the peripheral's characteristics. After this step, the channel is open and data transmission can start. In case of transparent mode, the UART is enabled at this time.

As soon as the connection has been established data can be transmitted in both directions.

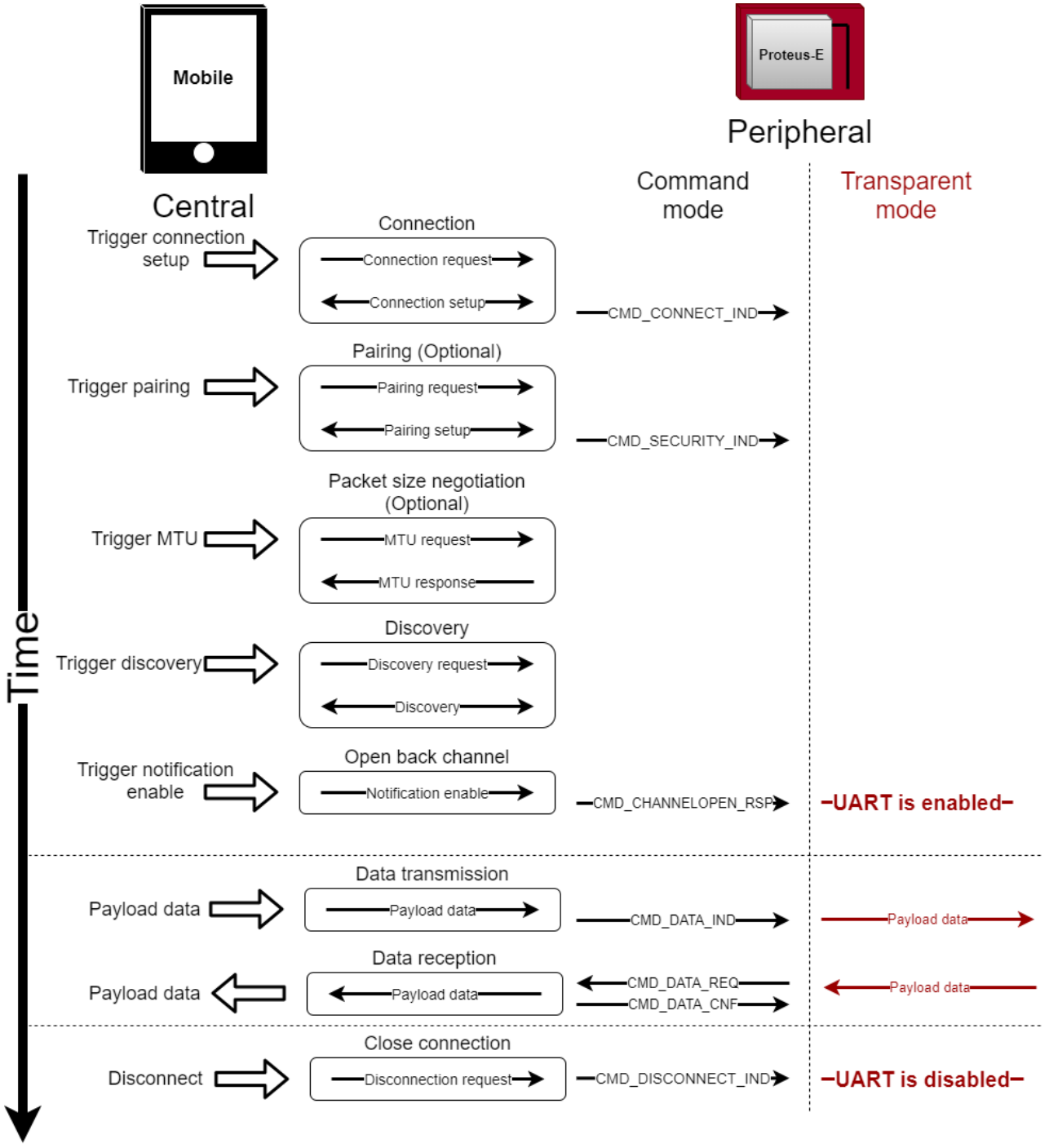


Figure 10: Steps for the connection setup

For more details about the connection setup and data transmission, please refer to application note ANR025 [3].

5.7. Advertising of custom data

The content of standard advertising and scan response packets of the Proteus-e is automatically defined. To place custom data in the advertising and scan response packet, the Proteus-e implements the user settings `RF_AdvertisingData` and `RF_ScanResponseData`. Both settings contain the raw data that is to be placed in the advertising packet and scan response packet respectively, 31 bytes maximum per packet.

The format of the raw data is defined in the Bluetooth specification [1] chapter "11 ADVERTISING AND SCAN RESPONSE DATA FORMAT". Placing other content in the advertising and scan response packets can result in malfunctioning.

5.7.1. Restrictions

In case custom data shall be placed in the advertising and scan response packet, the following restrictions must be respected:

1. It is not allowed to place "Flags" (header byte 0x01) in the `RF_ScanResponseData`.
2. In case the device name (header byte 0x08 or 0x09) shall be added, it must be placed only in `RF_AdvertisingData` or `RF_ScanResponseData`, not in both at the same time.

If one or several of these restrictions are not respected, custom data can not be written to the user settings `RF_AdvertisingData` and `RF_ScanResponseData`.

5.7.2. Application of custom advertising and scan response data

The content of the standard advertising and scan response packet is only cleared, if the user setting `RF_AdvertisingData` contains at least one byte. In this case the content of `RF_AdvertisingData` is placed in the advertising packet.
If the user setting `RF_ScanResponseData` contains at least one byte in addition, this content is placed in the scan response packet.
Thus, to place custom data in the scan response packet, custom data must be placed in the advertising packet first.

5.8. Energy-efficient distance estimation solutions

The transmitted scan response packet contains the TX power value used by the Proteus-e. This value in combination with the RSSI value of the received advertising packet can be used to estimate the distance to the module. Using a suitable triangulation algorithm and multiple receivers or transmitters, a position can be approximately determined.

The scan response packet is only received by the scanner, if it performs an active scan.

5.9. Configure the module for low power consumption

Depending on the application environment of the Proteus-e, 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 Bluetooth® LE 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 *UART_ENABLE* 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_AdvertisingInterval` and `RF_ConnectionInterval`: These parameters define the timing behaviour of the Proteus-e when advertising or during an open connection. The larger these intervals are, the less often data is transmitted via radio and thus power consumption descreases.
- The 2 MBit radio mode transmits data packets faster and thus reduces the power consumption slightly.

5.10. 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 03 00 10 05 01 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 03 00 10 05 01 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.

5.11. Using the 2 MBit phy

Bluetooth® 5 allows to transmit data with 2 MBit data rate.
To be backward compatible to Bluetooth® LE 4.x devices, Bluetooth® LE 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. To switch the phy after the connection has been setup the Proteus-e offers the command `CMD_PHYUPDATE_REQ`. As response to this request a `CMD_PHYUPDATE_IND` is returned from the Proteus-e, 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 phy is an optional feature of Bluetooth® 5 devices and therefore must not be supported by any connection partner.

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: Command byte identifying the command (1 Byte).

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

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

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

Host integration example codes for checksum calculation and command frame structure can be found in annex `A` and `B`, as well as in the Wireless Connectivity SDK [6][7].

If the transmission of the UART command has not finished within the packet transmission duration (depending on the currently selected UART baud rate + 5 ms after having received the start signal), the module discards the received Bytes and waits 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 9). If a command is not permitted in the current state, the command confirmation returns "Operation not permitted" as a response.

7.1. Setup connections

7.1.1. CMD_CONNECT_IND

This telegram indicates that a remote device starts the connection process. It contains the status and the FS_BTMAC of the connecting device.
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

7.1.2. CMD_SECURITY_IND

This telegram indicates the security status and the FS_BTMAC of the connected device.
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

7.1.3. CMD_CHANNELOPEN_RSP

This command is sent to the host as soon as connection setup 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 (MPS) that is supported by the link is part of this telegram.
Format:

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

Status:

0x00: Success

7.1.4. CMD_DISCONNECT_REQ

This command closes 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

7.1.5. 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

7.1.6. CMD_PHYUPDATE_REQ

This command allows to update the PHY of the current Bluetooth® LE 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

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

7.1.7. 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

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

7.1.8. 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.

7.1.8.1. 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

7.1.9. 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

7.1.9.1. 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.

7.1.9.2. 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.

7.1.10. CMD_ALLOWUNBONDEDCONNECTIONS_REQ

In case the `SECFLAGS_BONDEDCONNECTIONSONLY_ENABLE` bit has been set in the `RF_SecFlags` user setting, this command temporarily allows the connection setup of unbonded devices until the radio module is reset.
Format:

Start signal	Command	Length	CS
0x02	0x2D	0x00 0x00	0x2F

Response (`CMD_ALLOWUNBONDEDCONNECTIONS_CNF`):

Start signal	Command | 0x40	Length	Status	CS
0x02	0x6D	2 Bytes	1 Byte	1 Byte

Status:

0x00: Request successfully processed

0x01: Operation failed

0xFF: Operation not permitted

7.2. Transmit and receive data

7.2.1. CMD_DATA_REQ

This command provides the simple data transfer to the previously connected device. This command is suitable for transmission for a point-to-point connection. The number of payload data bytes (MPS) 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 sent to the host. Additionally a CMD_TXCOMPLETE_RSP will follow as soon as the data has been sent.
The receiving Proteus-e will get a CMD_DATA_IND message containing the transmitted payload data.
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

7.2.2. CMD_TXCOMPLETE_RSP

This command is sent to the host as soon as the data, which was requested by a CMD_DATA_REQ has been transmitted.
Format:

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

Status:

0x00: Data transmitted successfully

0x01: Data transmission failed

7.2.3. CMD_DATA_IND

This telegram indicates the reception of data sent by the previously connected device. The CMD_DATA_IND returns the FS_BTMAC of the sending device, the RSSI value of the received data packet and the payload data received. 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. Configuring the module and modifying the device settings

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.

7.3.1. 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 21.
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 of minimum 10.000. Try to avoid performing periodic CMD_SET_REQ as each command will use one write cycle.

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

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 too low. Please apply correct supply voltage, reset and retry.

0xFF: Operation not permitted

7.3.1.1. 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.

7.3.1.2. 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.

7.3.2. 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 21.
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

7.3.2.1. 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.

7.4. Manage the device state

7.4.1. 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

0x10: Direct test mode (DTM)

Other: Reserved

Module action:

0x00: No action

0x01: Idle (advertising)

0x03: Connected (More info is the 6 Bytes FS_BTMAC address of the connected device, followed by the MPS of the current connection)

0x04: Sleep (system-off mode)

0x05: Direct test mode (More info is 1 reserved byte)

7.4.1.1. 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	0x09 0x00	0x01	0x03	0x11 0x00 0x00 0xDA 0x18 0x00 0xF3	0x68

The module is connected to another module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 and an MPS of 243 Bytes (0xF3).

7.4.2. 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

7.4.3. CMD_SLEEP_REQ

This command is used to start the system-off mode (`ACTION_SLEEP`). For more details, see chapter `5.4`.
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

7.4.4. 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

7.4.5. CMD_FACTORYRESET_REQ

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

This command also removes all bonding data and GPIO configurations.

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.

7.4.6. 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 radio or a state is indicated) or if the *UART_ENABLE* pin is used (apply a falling edge, hold low for at least 10 ms before applying a rising edge and hold high for at least 10 ms). 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` is sent to the Proteus-e.
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

It is strongly recommended to disable the UART 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 Bluetooth® LE devices.

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

7.4.7. 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 *UART_ENABLE* pin for wake-up). 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. Run the Bluetooth test modes

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

7.5.1. 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. 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

7.5.2. 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 8.14 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	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 +4 dBm (see chapter 8.14 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 5.10.

7.5.2.1. 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.

7.5.2.2. Example: Receiver, channel 0

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

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.

7.5.2.3. Example: Transmission, carrier test, channel 0

Start the carrier test on channel 0 (2402 MHz). 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.

7.5.2.4. 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

7.5.2.5. 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

7.6. Switching GPIOs by remote control

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

7.6.1. 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 of minimum 10.000. 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	Values
1 Byte	1 Byte	1 Byte	(Length - 2) Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 11.1

Function:

0x00: GPIO disconnected

0x01: GPIO works as input

0x02: GPIO works as output

Values:

- if **Function** is disconnected, Length is 0x03:
  0x00: value field must use 0x00.
- if **Function** is input, Length is 0x03:
  0x00: GPIO has no pull resistor
  0x01: GPIO has internal pull down resistor
  0x02: GPIO has internal pull up resistor
- if **Function** is output, Length is 0x03:
  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

7.6.1.1. 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.

7.6.2. 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	$Block_1$	...	$Block_n$	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	Values
1 Byte	1 Byte	1 Byte	(Length - 2) Byte

Length: Length of the subsequent bytes in this block

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

Function:

0x00: GPIO is disconnected
0x01: GPIO works as input
0x02: GPIO works as output

Values:

- if **Function** is disconnected, Length is 0x02:
  Values field is not used in this Block
- if **Function** is input, Length is 0x03:
  0x00: GPIO has no pull resistor
  0x01: GPIO has pull down resistor
  0x02: GPIO has pull up resistor
- if **Function** is output, Length is 0x03:
  0x00: GPIO is output LOW
  0x01: GPIO is output HIGH

7.6.2.1. 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	0x09 0x00	0x00	0x03 **0x01** 0x02 0x01 0x03 **0x02** 0x02 0x01	0x63

The GPIOs with GPIO_ID **0x01** and **0x02** are output high.

7.6.3. 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 Bluetooth® LE.

The flash memory used to store these settings has a limited count of write cycles of minimum 10.000. 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	Values
1 Byte	1 Byte	1 Byte	(Length - 2) Byte

Length: Length of the subsequent bytes in this block

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

Function:

0x00: GPIO disconnected

0x01: GPIO works as input

0x02: GPIO works as output

Values:

- if **Function** is disconnected, Length is 0x03:

0x00: value field must use 0x00.

- if **Function** is input, Length is 0x03:

  0x00: GPIO has no pull resistor

  0x01: GPIO has internal pull down resistor

  0x02: GPIO has internal pull up resistor

- if **Function** is output, Length is 0x03:

  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)

7.6.3.1. 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	$Block_1$	$Block_2$	CS
0x02	0x28	0x08 0x00	0x03 **0x01** 0x02 0x01	0x03 **0x02** 0x02 0x01	0x21

Response:

Start signal	Command | 0x40	Length	Status	$Block_1$	$Block_2$	CS
0x02	0x68	0x07 0x00	0x00	0x02 **0x01** 0x00	0x02 **0x02** 0x00	0x6E

Configured both GPIOs with success.

7.6.4. 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	Values
1 Byte	1 Byte	1 Byte	(Length - 2) Byte

Length: Length of the subsequent bytes in this block

GPIO_ID: ID of the GPIO, see chapter 11.1

Function:

0x00: GPIO is disconnected
0x01: GPIO works as input
0x02: GPIO works as output

Values:

- if **Function** is disconnected, Length is 0x02:
  Values field is not used in this Block
- if **Function** is input, Length is 0x03:
  0x00: GPIO has no pull resistor
  0x01: GPIO has pull down resistor
  0x02: GPIO has pull up resistor
- if **Function** is output, Length is 0x03:
  0x00: GPIO is output LOW
  0x01: GPIO is output HIGH

7.6.4.1. 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	0x09 0x00	0x00	0x03 **0x01** 0x02 0x01 0x03 **0x02** 0x02 0x01	0x64

The GPIOs with GPIO_ID **0x01** and **0x02** are output high.

7.6.5. 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 Bluetooth® LE.
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 Bluetooth® LE)

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:

- if **Function** is output

  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

7.6.5.1. 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.

7.6.6. 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 Bluetooth® LE.
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 Bluetooth® LE)

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:

- if **Function** is output or input

  0x00: The remote GPIO is LOW.

  0x01: The remote GPIO is HIGH.

  0xFF: Failed reading remote GPIO value.

7.6.6.1. 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).

7.6.7. 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	$Block_1$	...	$Block_n$	CS
0x02	0x26	2 Bytes	x Bytes		x Bytes	1 Byte

Response (CMD_GPIO_LOCAL_WRITE_CNF):

Start signal	Command | 0x40	Length	Status	$Block_1$	...	$Block_n$	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 Bluetooth® LE)

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:

- if **Function** is output

  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

7.6.7.1. 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.

7.6.8. 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 Bluetooth® LE)

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:

- if **Function** is output or input
  0x00: The remote GPIO is LOW.
  0x01: The remote GPIO is HIGH.
  0xFF: Failed reading remote GPIO value.

7.6.8.1. 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	$Block_1$	CS
0x02	0x27	0x03 0x00	0x02 **0x01 0x02**	0x27

Response:

Start signal	Command | 0x40	Length	Status	$Block_1$	$Block_2$	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).

7.6.9. 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`.

7.6.9.1. 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.

7.6.10. 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`.

7.6.10.1. 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).

7.6.11. 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`.

7.6.11.1. 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.

7.7. Other messages

7.7.1. 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.

7.8. Message overview

Start signal	CMD	Message name	Short description	Chapter
0x02	0x00	CMD_RESET_REQ	Reset the module	7.4.2
0x02	0x01	CMD_GETSTATE_REQ	Request the current module state	7.4.1
0x02	0x02	CMD_SLEEP_REQ	Go to sleep	7.4.3
0x02	0x04	CMD_DATA_REQ	Send data to the connected device	7.2.1
0x02	0x07	CMD_DISCONNECT_REQ	Close the connection	7.1.4
0x02	0x0E	CMD_DELETEBONDS_REQ	Delete bonding information	7.1.9
0x02	0x0F	CMD_GETBONDS_REQ	Read the MACs of bonded devices	7.1.8
0x02	0x10	CMD_GET_REQ	Read the module settings in flash	7.3.2
0x02	0x11	CMD_SET_REQ	Modify the module settings in flash	7.3.1
0x02	0x1A	CMD_PHYUPDATE_REQ	Update the PHY	7.1.6
0x02	0x1B	CMD_UARTDISABLE_REQ	Disable the UART	7.4.6
0x02	0x1C	CMD_FACTORYRESET_REQ	Perform a factory reset	7.4.5
0x02	0x1D	CMD_DTMSTART_REQ	Enable the direct test mode	7.5.1
0x02	0x1E	CMD_DTM_REQ	Start/stop a test of the direct test mode	7.5.2

Table 15: 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	7.6.1
0x02	0x26	CMD_GPIO_LOCAL_WRITE_REQ	Set the output value of a output GPIO of the current device	7.6.7
0x02	0x27	CMD_GPIO_LOCAL_READ_REQ	Read the value of a GPIO of the current device	7.6.8
0x02	0x28	CMD_GPIO_REMOTE_WRITECONFIG_REQ	Configure the free GPIOs of the remote device for remote control	7.6.3
0x02	0x29	CMD_GPIO_REMOTE_WRITE_REQ	Set the output value of a output GPIO of a remote device	7.6.5
0x02	0x2A	CMD_GPIO_REMOTE_READ_REQ	Read the value of a GPIO of a remote device	7.6.6
0x02	0x2B	CMD_GPIO_LOCAL_READCONFIG_REQ	Read the GPIO configuration	7.6.2
0x02	0x2C	CMD_GPIO_REMOTE_READCONFIG_REQ	Read the GPIO configuration of the connected remote device	7.6.4
0x02	0x2D	CMD_ALLOWUNBONDEDCONNECTIONS_REQ	Temporarily allow the connection setup from unbonded peer devices	7.1.10

Table 16: Message overview: Requests 2

Start signal	CMD	Message name	Short description	Chapter
0x02	0x40	CMD_RESET_CNF	Reset request received	7.4.2
0x02	0x41	CMD_GETSTATE_CNF	Return the current module state	7.4.1
0x02	0x42	CMD_SLEEP_CNF	Sleep request received	7.4.3
0x02	0x44	CMD_DATA_CNF	Data transmission request received	7.2.1
0x02	0x47	CMD_DISCONNECT_CNF	Disconnection request received	7.1.4
0x02	0x4E	CMD_DELETEBONDS_CNF	Deleted bonding information	7.1.9
0x02	0x4F	CMD_GETBONDS_CNF	Return the MAC of all bonded devices	7.1.8
0x02	0x50	CMD_GET_CNF	Return the requested module flash settings	7.3.2
0x02	0x51	CMD_SET_CNF	Module flash settings have been modified	7.3.1
0x02	0x5A	CMD_PHYUPDATE_CNF	Update Phy request received	7.1.6
0x02	0x5B	CMD_UARTDISABLE_CNF	Disable UART request received	7.4.6
0x02	0x5C	CMD_FACTORYRESET_CNF	Factory reset request received	7.4.5
0x02	0x5D	CMD_DTMSTART_CNF	Enable the direct test mode now	7.5.1
0x02	0x5E	CMD_DTM_CNF	Test of direct test mode started/stopped	7.5.2
0x02	0x65	CMD_GPIO_LOCAL_WRITECONFIG_CNF	Configuration of a local GPIO for remote control done	7.6.1
0x02	0x66	CMD_GPIO_LOCAL_WRITE_CNF	Output value of a local GPIO set	7.6.7
0x02	0x67	CMD_GPIO_LOCAL_READ_CNF	Value of a local GPIO read	7.6.8
0x02	0x68	CMD_GPIO_REMOTE_WRITECONFIG_CNF	Configuration of a remote GPIO for remote control done	7.6.3
0x02	0x69	CMD_GPIO_REMOTE_WRITE_CNF	Output value of a remote GPIO set	7.6.5
0x02	0x6A	CMD_GPIO_REMOTE_READ_CNF	Value of a remote GPIO read	7.6.6
0x02	0x6B	CMD_GPIO_LOCAL_READCONFIG_CNF	Returns the GPIO configuration	7.6.2
0x02	0x6C	CMD_GPIO_REMOTE_READCONFIG_CNF	Returns the GPIO configuration of the connected remote device	7.6.4
0x02	0x6D	CMD_ALLOWUNBONDEDCONNECTIONS_CNF	Temporarily allowed the connection setup from unbonded peer devices	7.1.10

Table 17: Message overview: Confirmations

Start signal	CMD	Message name	Short description	Chapter
0x02	0x82	CMD_SLEEP_IND	State will be changed to ACTION_SLEEP	7.4.4
0x02	0x84	CMD_DATA_IND	Data has been received	7.2.3
0x02	0x86	CMD_CONNECT_IND	Connection established	7.1.1
0x02	0x87	CMD_DISCONNECT_IND	Disconnected	7.1.5
0x02	0x88	CMD_SECURITY_IND	Secured connection established	7.1.2
0x02	0x9A	CMD_PHYUPDATE_IND	PHY has been updated	7.1.7
0x02	0x9B	CMD_UARTENABLE_IND	UART was re-enabled	7.4.7
0x02	0xA2	CMD_ERROR_IND	Entered error state	7.7.1
0x02	0xA6	CMD_GPIO_LOCAL_WRITE_IND	Local host has written the GPIOs of the remote device	7.6.11
0x02	0xA8	CMD_GPIO_REMOTE_WRITECONFIG_IND	Remote device has configured the GPIOs of the module	7.6.9
0x02	0xA9	CMD_GPIO_REMOTE_WRITE_IND	Remote device has written the GPIOs of the module	7.6.10
0x02	0xC4	CMD_TXCOMPLETE_RSP	Data has been sent	7.2.2
0x02	0xC6	CMD_CHANNELOPEN_RSP	Channel open, data transmission possible	7.1.3

Table 18: 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.

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

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:

0x0126 : Softdevice S112 7.3.0.

Package variant:

0x2005: WLCSP

Chip ID:

0x00052805: nRF52805

Packet variant	Package	Flash size	RAM size
WLCSP	WLCSP	196 kB	24 kB

Table 19: nRF52805 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 = 0x0126 (Softdevice S112 7.3.0)
Build code = 0x41414300 (AAC0)
Package variant = 0x2005 (WLCSP)
Chip ID = 0x00052805
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	0x26 0x01 0x30 0x43 0x41 0x41 0x05 0x20 0x05 0x28 0x05 0x00	0x06

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 Bluetooth conform MAC address

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

This setting contains the Bluetooth® LE conform MAC address of the module. The FS_BTMAC is introduced 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 settings index 4

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

Response CMD_GET_CNF: Successfully read out the Bluetooth® LE 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-E"	read/write	1-31

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 26 Bytes. Thus longer device names will be shortened to 26 Bytes and declared as "Shortened Local Name" in the advertising packet. The full device name is included in the GAP.

In case the device name is not changed from default value, it is attached by the ASCII serial number of the module.

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.
The permissible characters are ranging from 0x30 to 0x39 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 5.6 contains further information about secure connections.

When connecting from a foreign device to a Proteus-e, the peripheral (Proteus-e) determines the minimum security level needed for communication. So configure the RF_SecFlags of the peripheral to set the desired security level. When connecting from a Proteus-I,-II,-III to a Proteus-e, be sure that the same security mode is used.

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.

<table>
<tr><th>Bit no.</th><th>Description</th></tr>
<tr><td>2 : 0</td><td>Security mode configuration. Depending on its value, different modes are chosen when setting up a secure connection.
<table>
<tr><td>0x0</td><td>No security</td><td>Data is transmitted without authentication and encryption.</td></tr>
<tr><td>0x2</td><td>Just works level 1.2</td><td>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.</td></tr>
<tr><td>0x3</td><td>Static pass key level 1.3</td><td>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.</td></tr>
<tr><td>others</td><td></td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>3</td><td>SECFLAGS_BONDING_ENABLE: If this Bit is set, bonding is enabled when using one of the pairing methods. Bonding data of up to 12 devices will be stored in the flash. If bonding storage is full, the bonding information that has not been used for the longest period will be removed.</td></tr>
<tr><td>4</td><td>SECFLAGS_BONDEDCONNECTIONSONLY_ENABLE: If this Bit is set, only bonded peer devices are allowed to connect. All connection requests from any unbonded peer device are rejected. In case this restriction shall be disabled temporarily to setup a bonding to a new peer device, use the command CMD_ALLOWUNBONDEDCONNECTIONS_REQ, which temporarily disables this restriction. If this feature is enabled, the maximum number of bonded devices is reduced to 8.</td></tr>
<tr><td>7 : 5</td><td>Reserved</td></tr>
</table>

Table 20: Security configuration flags

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_ScanResponseData: Modify the content of the scan response packet

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
14	RF_ScanResponseData	See description	None	read/write	31

The standard content of the scan response packet of the Proteus-e is automatically defined. This setting allows to put user content in the scan response packet. The value of this user setting is the raw data which is placed without modification in the scan response packet, after the standard content has been removed before.

Please check chapter Advertising of custom data for details and restrictions before modifying this user setting.

Please ensure that the raw data is compliant to the Bluetooth® specification [1] chapter "11 ADVERTISING AND SCAN RESPONSE DATA FORMAT". Otherwise it can result in malfunctioning.

8.9.1. Example 1

Set the data of the scan response packet to:

- TX power is 4 (0x02 0x0A 0x04)

using CMD_SET_REQ with settings index 14

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x04 0x00	0x0E	0x02 0x0A 0x04	0x15

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 static pass key 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 content as:

- TX power is 4 (0x02 0x0A 0x04)

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x04 0x00	0x00	0x02 0x0A 0x04	0x5A

8.10. RF_AdvertisingData: Modify the content of the advertising packet

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
13	RF_AdvertisingData	See description	None	read/write	31

The standard content of the advertising packet of the Proteus-e is automatically defined. This setting allows to put user content in the advertising packet. The value of this user setting is the raw data which is placed without modification in the advertising packet, after the standard content has been removed before.

Please check chapter Advertising of custom data for details and restrictions before modifying this user setting.

Please ensure that the raw data is compliant to the Bluetooth® specification [1] chapter "11 ADVERTISING AND SCAN RESPONSE DATA FORMAT". Otherwise it can result in malfunctioning.

8.10.1. Example 1

Set the data of the advertising packet to:

- full device name is "Hello" (0x06 0x09 0x48 0x65 0x6C 0x6C 0x6F)
- TX power is 4 (0x02 0x0A 0x04)

using CMD_SET_REQ with settings index 13

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x0B 0x00	0x0D	0x06 0x09 0x48 0x65 0x6C 0x6C 0x6F 0x02 0x0A 0x04	0x54

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 static pass key 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 content as:

- full device name is "Hello" (0x06 0x09 0x48 0x65 0x6C 0x6C 0x6F)
- TX power is 4 (0x02 0x0A 0x04)

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x0B 0x00	0x00	0x06 0x09 0x48 0x65 0x6C 0x6C 0x6F 0x02 0x0A 0x04	0x18

8.11. 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 sleep 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_AdvertisingInterval parameter, which defines the frequency of advertising packets.

8.11.1. 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

8.11.2. 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

8.12. RF_AdvertisingInterval: Modify the advertising interval

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
9	RF_AdvertisingInterval	20 - 10240	40	read/write	2

The RF_AdvertisingInterval defines how often advertising packets are transmitted. The value is the interval in milliseconds.
The choice of the RF_AdvertisingInterval primarily affects the latency of device detection on air as well as the current consumption. A lower value of the RF_AdvertisingInterval results in a shorter pause between the advertising packets. Thus the radio module can be detected earlier, but also needs more power.

8.12.1. Example 1

Set the advertising interval to 100 ms using CMD_SET_REQ with settings index 9.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x09	0x64 0x00	0x7D

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 interval 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 0x0028 (40 ms).

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x03 0x00	0x00	0x28 0x00	0x79

8.13. RF_ConnectionInterval: Modify the connection interval

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
8	RF_ConnectionInterval	See description	15-50	read/write	4

The user setting RF_ConnectionInterval defines the minimum and maximum connection interval, which is used to negotiate the connection interval during connection setup.

The 4 byte value of the user setting RF_ConnectionInterval consists of the 2 byte value of the minimum connection interval, followed by the 2 byte value of the maximum connection interval (see the examples below).
The value for the minimum connection interval, and the value for the maximum connection interval must be within 8 ms and 4000 ms. Only settings are accepted where the minimum connection interval is lower or equal to the maximum connection interval.

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. The lower the connection interval is, the more frequently the connected devices communicate with each other and thus the more power is consumed.
  If a Bluetooth® LE device (e.g. a smart phone) connects as central to a Proteus-e module (peripheral) and the connection interval settings do not coincide, the Proteus-e requests the smart phone to accept its settings after 5 s. If the cell phone does not accept the settings, it will be requested a further 3 times with a delay of 10 s. 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-e, the module accepts the request.

Please ensure that all members (Proteus-e, cell phones and other Bluetooth® LE devices) of a network use the same connection timing parameters to avoid connection problems and changes of the connection interval during an opened connection.

Please check the minimum connection interval that is supported by iOS is 15 ms. Furthermore the minimum connection interval for Apple devices must be modulo 15 ms!

8.13.1. Example 1

Set the RF_ConnectionInterval to 16-40 ms (0x0010-0x0028) using CMD_SET_REQ with settings index 8.

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x05 0x00	0x08	0x10 0x00 0x28 0x00	0x26

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 minimum and maximum connection interval 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 20-75 ms (0x0014-0x004B).

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x05 0x00	0x00	0x14 0x00 0x4B 0x00	0x08

8.14. RF_TXPower: Modify the output power

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
17	RF_TXPower	See description	4	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	-20	-16	-12	-8
Two's complement, hexadecimal	0xD8	0xEC	0xF0	0xF4	0xF8

Decimal [dBm]	-4	0	3	4	
Two's complement, hexadecimal	0xFC	0x00	0x03	0x04	

Please note that this setting defines the TX power value the radio chip provides. The radio signal will be damped on its way along the RF path and the antenna. For maximum output signal at the *RF* pin or integrated antenna, please refer to the chapter Radio characteristics.

8.14.1. 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

8.14.2. 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

8.15. 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

Set the base UUID of the SPP-like profile. For more information about the UUID definition, please refer to chapter 12.1.

8.15.1. 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

8.15.2. 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

8.16. RF_SPPServiceUUID: Configure the SPP service UUID

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
32	RF_SPPServiceUUID	See description	0x0001	read/write	2

Set the service UUID of the SPP-like profile. For more information about the UUID definition, please refer to chapter 12.1.
The service UUID can be every value, but must be different from RF_SPPTXUUID and RF_SPPRXUUID.

8.16.1. Example 1

Set the service UUID to 0x1122 using CMD_SET_REQ with settings index 32

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x20	0x22 0x11	0x03

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 service UUID of the module using CMD_GET_REQ:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x20	0x33

Response CMD_GET_CNF: Successfully read out the value 0x1234.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x03 0x00	0x00	0x34 0x12	0x77

8.17. RF_SPPRXUUID: Configure the SPP RX UUID

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
33	RF_SPPRXUUID	See description	0x0002	read/write	2

Set the RX UUID of the SPP-like profile. This characteristics has the function to transmit data from the connected remote peer to the radio module via write command. For more information about the UUID definition, please refer to chapter 12.1.

The RX UUID can be every value, but must be different from RF_SPPServiceUUID and RF_SPPTXUUID.

8.17.1. Example 1

Set the RX UUID to 0x1122 using CMD_SET_REQ with settings index 33

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x21	0x22 0x11	0x02

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 service UUID of the module using CMD_GET_REQ:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x21	0x32

Response CMD_GET_CNF: Successfully read out the value 0x1234.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x03 0x00	0x00	0x34 0x12	0x77

8.18. RF_SPPTXUUID: Configure the SPP TX UUID

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
34	RF_SPPTXUUID	See description	0x0003	read/write	2

Set the TX UUID of the SPP-like profile. This characteristics has the function to transmit data from the radio module to the connected remote peer via notification. For more information about the UUID definition, please refer to chapter 12.1.

The TX UUID can be every value, but must be different from RF_SPPServiceUUID and RF_SPPRXUUID.

8.18.1. Example 1

Set the TX UUID to 0x1122 using CMD_SET_REQ with settings index 34

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x22	0x22 0x11	0x01

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 service UUID of the module using CMD_GET_REQ:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x22	0x31

Response CMD_GET_CNF: Successfully read out the value 0x1234.

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x03 0x00	0x00	0x34 0x12	0x77

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 [1] 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. To avoid loosing single bytes, please enable the UART flow control.

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	24	read/write	2

The user setting CFG_Flags specifies various module features.

Bit no.	Name	Description
0-1	Reserved	Reserved.
2	GPIO remote config.	Set this Bit to 1 to block the GPIO configuration via remote device.
3	DCDC enable	Set this Bit to 1 to use the internal DCDC instead of LDO.
4	Disconnect enable	Set this Bit to disconnect in case the central is forcing the use of invalid connection parameters.
5-15	Reserved	Reserved.

8.21.1. Example 1

Block the GPIO configuration via remote and enable the DCDC access using CMD_SET_REQ with settings index 28

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x03 0x00	0x1C	0x0C 0x00	0x00

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

Settings index	Designation	Summary	Permissible values	Default value	Permissi	Number of Bytes
1	FS_FWVersion	Version of the firmware	-	-	read	3
2	RF_DeviceName	Name of the module	See description	"Prot3"	read / write	1-31
3	FS_MAC	MAC address of the module	-	-	read	6
4	FS_BTMAC	Bluetooth® LE 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_ConnectionInterval	Connection interval	See description	15-50	read / write	4
9	RF_AdvertisingInterval	Advertising interval	20-10240	40	read / write	2
11	UART_ConfigIndex	Baud rate of the UART	See description	22	read / write	1
12	RF_SecFlags	Security settings of the module	See description	0	read / write	1
13	RF_AdvertisingData	Custom advertising packet data	See description	-	read / write	31
14	RF_ScanResponseData	Custom scan response packet data	See description	-	read / write	31
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 description	4	read / write	1

Table 21: Table of settings (Part 1)

Settings index	Designation	Summary	Permissible values	Default value	Permissi	Number of Bytes
18	RF_StaticPasskey	6 digit pass key	See description	"123123"	read / write	6
25	RF_Appearance	Appearance	0-65535	0	read / write	2
26	RF_SPPBaseUUID	Base UUID of the SPP-like profile	See description	See description	read / write	16
28	CFG_Flags	CFG Flags	See description	24	read / write	2
32	RF_SPPServiceUUID	Service UUID of the SPP-like profile	See description	0x0001	read / write	2
33	RF_SPPRXUUID	RX UUID of the SPP-like profile	See description	0x0002	read / write	2
34	RF_SPPTXUUID	TX UUID of the SPP-like profile	See description	0x0003	read / write	2

Table 22: 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		ms

9.2. Bluetooth LE timing parameters

The timing parameters for sending advertising packets are determined by the user settings `RF_AdvertisingInterval` and `RF_AdvertisingTimeout`. Furthermore, the user setting `RF_ConnectionInterval` allows to configure the timing parameters used during connection setup and data transmission.

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, i.e. smart phone) waits for an advertising packet, which was sent by the peripheral (Proteus-e) 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 (see `RF_AdvertisingInterval`) of the peripheral as well as on the scan interval and scan window of the central.

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_ConnectionInterval`).

Connection type	Estimated number of exchanged messages	Negotiation time for a connection interval of 50ms
Unsecured connection	12-14	600-700 ms
Secured connection using the pairing method	22-24	1100-1200 ms
Secured connection to already bonded device	19-20	950-1000 ms

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.

9.4. Connection based data transmission

After connection has been setup, 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 to the host. The time needed for this coincides with the connection interval that was negotiated during connection setup. The `RF_ConnectionInterval` parameter defines the minimum and maximum connection interval, which is supported by the module.
The following image shows the command sequence when sending data:

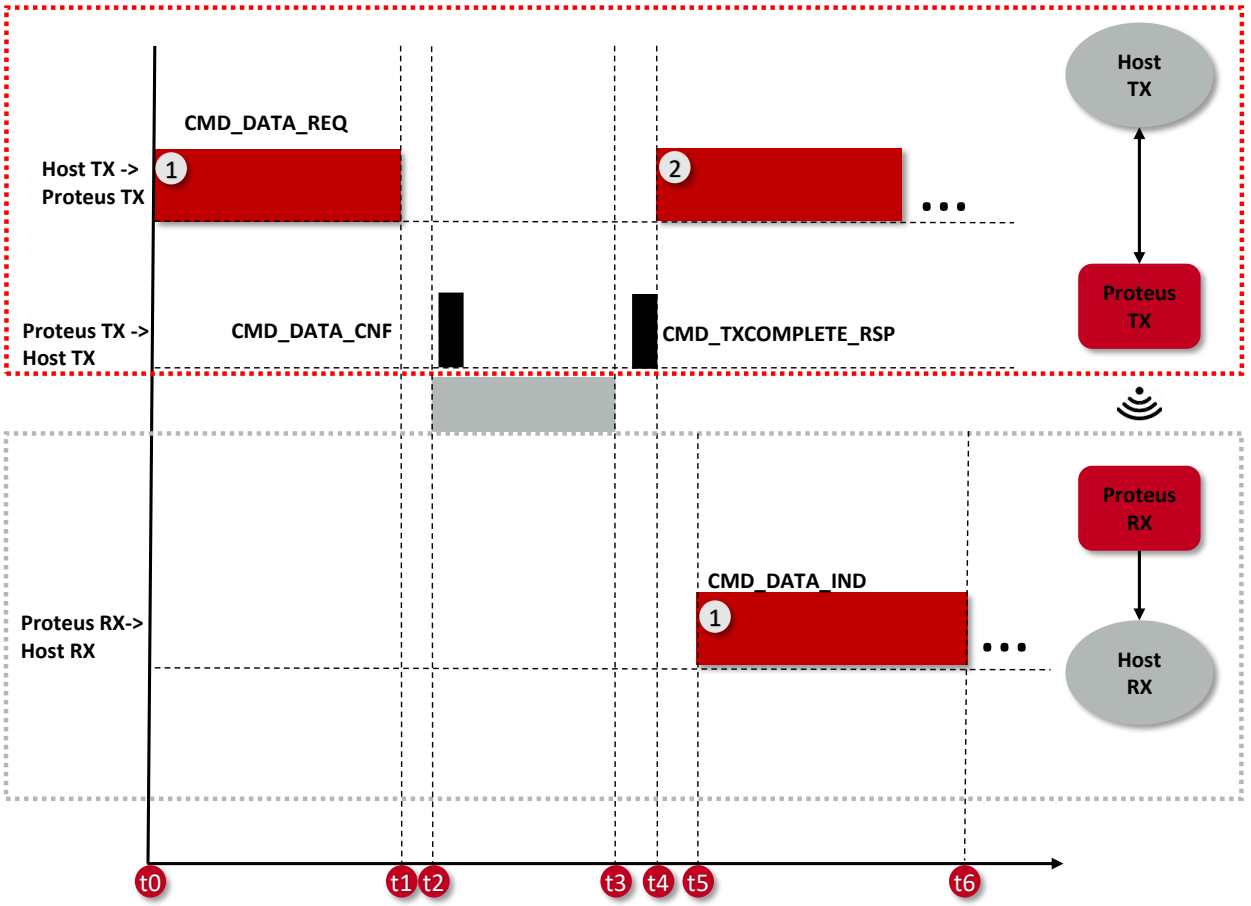


Figure 11: Command sequence when transmitting data

9.4.1. Maximum data throughput

The following table contains the measured maximum throughput values for user payload. The test setup is as follows:

- A Proteus-e radio module and a Proteus-III radio module connected to a fast micro controller (STM32 on NUCLEO-L476RG)
- Radio mode as specified in the table below (1 Mbit/s or 2 Mbit/s)
- Fastest connection interval of 8 ms (`RF_ConnectionInterval` equals 8-8ms)
- Fastest UART baud rate of 1000000 Baud (`UART_ConfigIndex` equals 33) with HW flow control

- 243 bytes per `CMD_DATA_REQ`

Radio mode	t1-t0 [ms] (UART RX)	t4-t0 [ms] (Host TX Period)	t6-t5 [ms] (UART TX)	t6-t0 [ms] (End-to-end delay)	243/(t4-t0) [kByte/s] (Throughput)
1 Mb/s	3.85	22.5	-	-	**10.8**
2 Mb/s	3.85	15	-	-	**16.2**

Table 23: Maximum throughput timings, packet error rate = 0%

Please note that data transmission to/from smart devices typically do not achieve this speed due to latency caused by the smart device and it's software and apps or even missing hardware features such as Bluetooth® LE 5.1 full feature support.

10. Transparent mode

The Proteus-e implements a new feature that allows the easy integration of the Proteus-e Bluetooth® LE module to an already existing host. The transparent mode offers a plug and play installation without previous configuration of the Proteus-e. It is tailored for easy communication with mobile Bluetooth® LE devices like smart phones.
The transparent mode is a special operation mode, that uses the configuration of the command mode described in the previous chapters. It has to be enabled during the module start-up and contains the following key features:

- Automatic UART enabling: During advertising the UART is disabled. As soon as another Bluetooth® LE enabled device connects to it, the UART of the Proteus-e is enabled, the *LED_1* pin shows that the channel is open and bidirectional data transmission can start. As soon as the connection is closed, the UART is disabled.
- Transparent UART interface: The serial interface of the Proteus-e is no longer driven by commands. This means, when the UART of the module is enabled, all data sent to the UART is transmitted by the Proteus-e to the connected Bluetooth® LE enabled device. On the other hand, all data received by radio is send from the Proteus-e to the connected host without additional header data. Since the commands of the command interface are no longer valid, a Proteus-e cannot be configured when running in transparent mode.

10.1. Reasons to use the transparent mode

The Proteus-e transparent mode equips custom applications with a Bluetooth® LE interface (to be accessible by other Bluetooth® LE devices) without installation effort. Due to the transparent UART interface, data can be exchanged without additional headers. In addition to that, the transparent mode allows an energy efficient operation of the Bluetooth® LE interface, since the UART is only enabled when it is really used.

10.2. How to use the transparent mode

The transparent 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.

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 command 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 transparent 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_DeviceName` (change the default device name, default value "Proteus-E")
- `RF_SecFlags` (change the default security mode, default value "open - no security")

10.3. More information

10.3.1. UART

- The data sent to the UART is buffered in the Proteus-e up to a maximum payload of 243 Bytes. When no new byte was received for 20 ms, the data will be transmitted by radio to the connected Bluetooth® LE device. If the data is larger than the MPS 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 MPS of the Bluetooth® LE connection is helpful. The Proteus-e supports up to 243 Bytes Bluetooth® LE 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 Bluetooth® LE packet and connection interval as given by the Bluetooth® 4.0 standard is used (compatibility mode to Bluetooth® LE 4.0 devices).
- The pin *BUSY* can be used as a kind of flow control for the data transmission during the transparent 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 Bluetooth® LE. The pin switches LOW again, as soon as Bluetooth® LE 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.

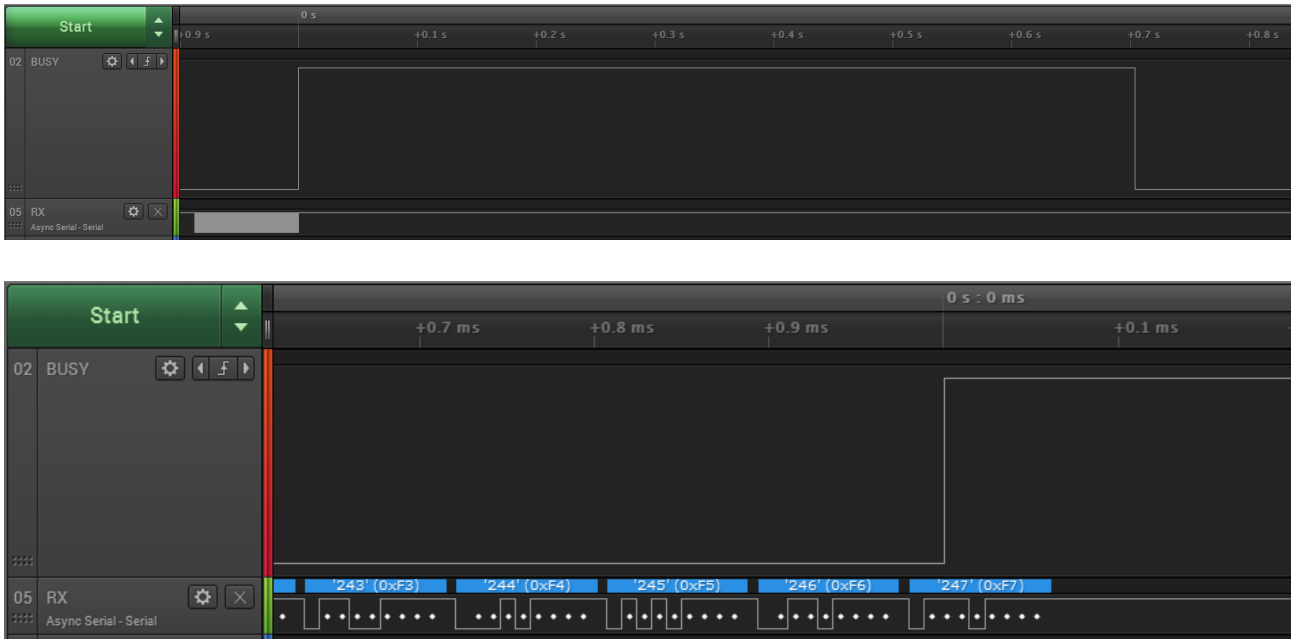


Figure 12: Switch of the *BUSY* pin when transmitting data

To use the signal on the *BUSY* pin as flow control on the host controller side, we recommend to use an OR gate to combine the */RTS* and *BUSY* pins' signals.

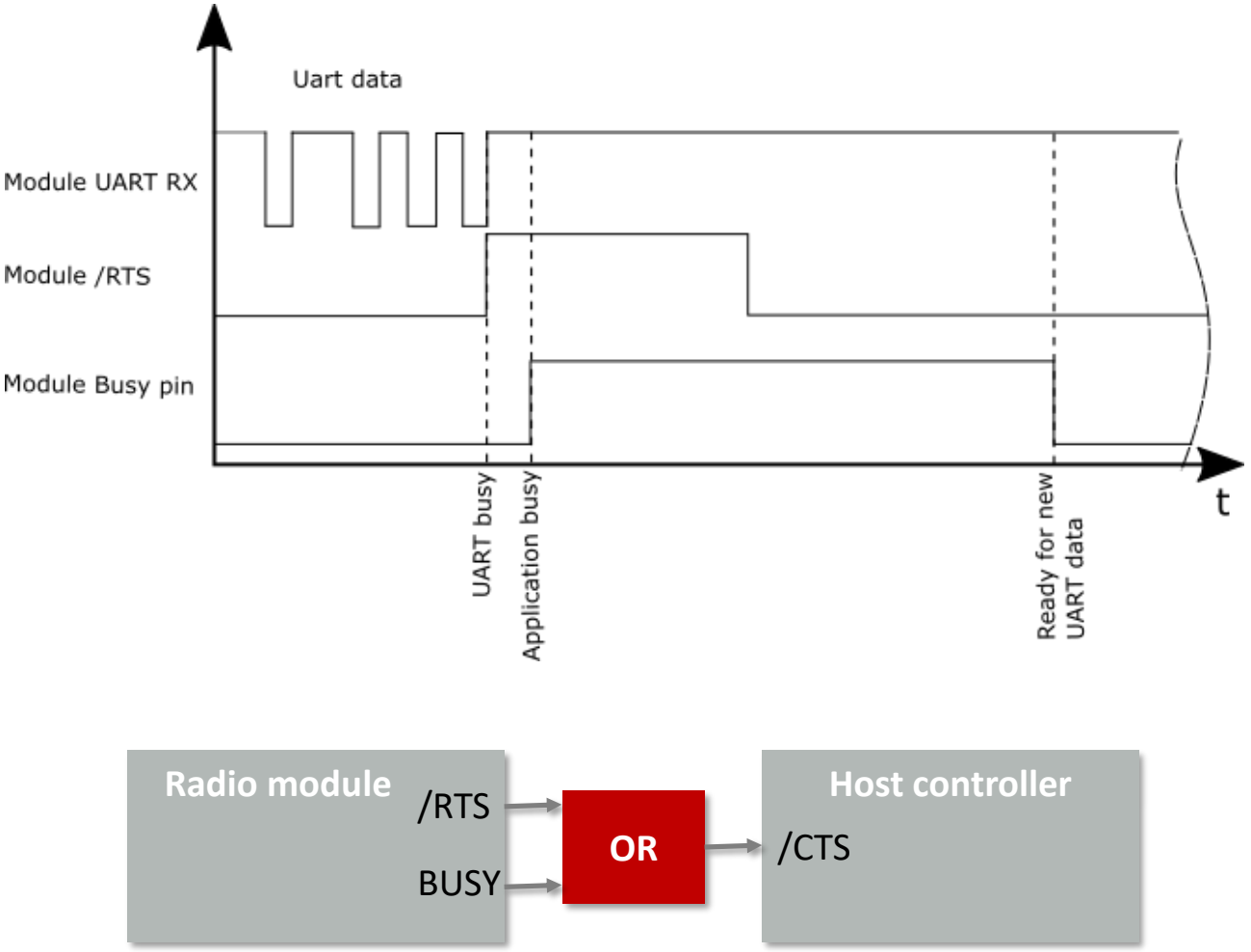


Figure 13: Handling the */RTS* and *BUSY* pin

11. Remote GPIO control

The Proteus-e allows to control free GPIOs via remote access. Chapter 7.6 contains the description of the necessary commands.

To use the remote GPIO control feature of the Proteus-e, the GPIOs of interest must be configured first. This can be done in two ways. Either by the local host (see figure 14), when the radio module is in idle mode (not connected via Bluetooth® LE), or via the connected remote device (see figure 15).
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 Bluetooth® LE connection to the remote device first and send a CMD_GPIO_REMOTE_WRITECONFIG_REQ command to the radio module via Bluetooth® LE 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 18), or CMD_GPIO_LOCAL_READ_REQ for reading GPIO values (see figure 19). 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 Bluetooth® LE connection to the radio module and send the respective commands, CMD_GPIO_REMOTE_WRITE_REQ for setting GPIO output values (see figure 20), or CMD_GPIO_REMOTE_READ_REQ for reading GPIO values (see figure 21).
Each time the GPIOs are written via remote connection, the local host is informed using a CMD_GPIO_REMOTE_WRITE_IND message.

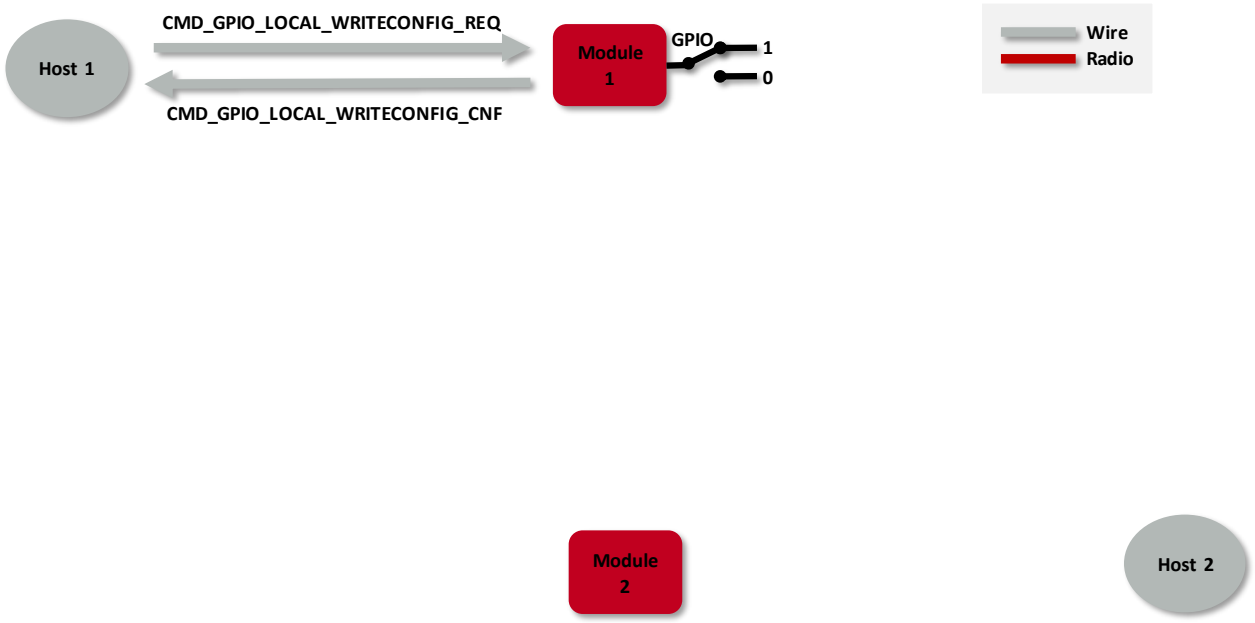


Figure 14: Configure the local GPIOs via local host

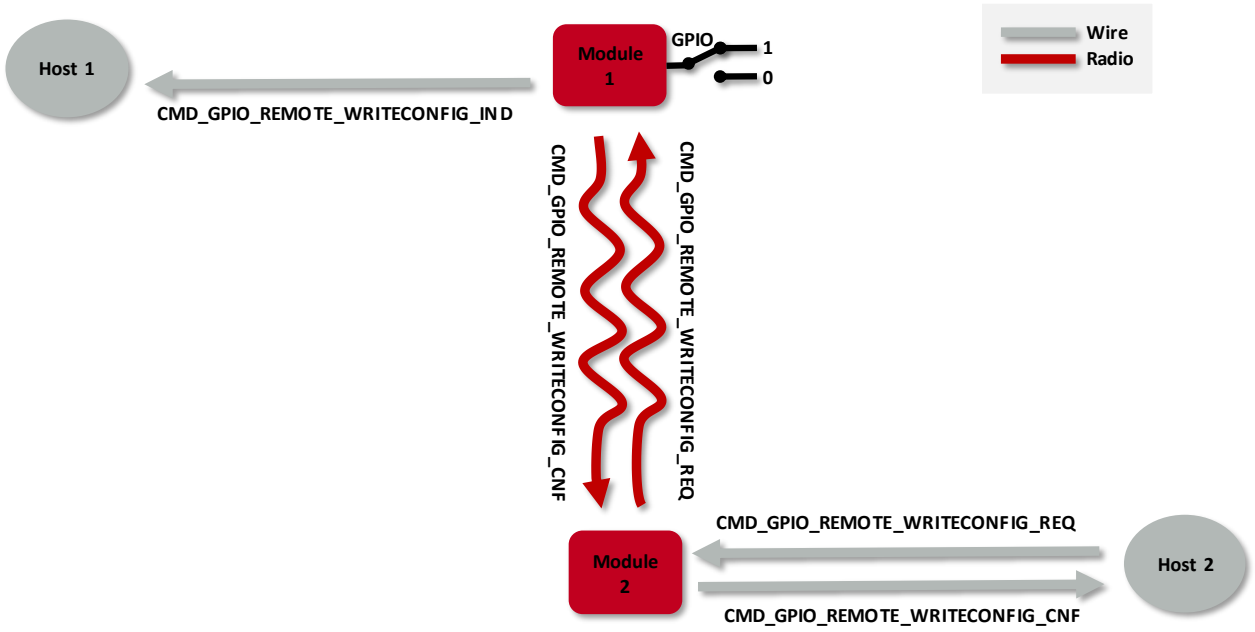


Figure 15: Configure the local GPIOs via remote device host

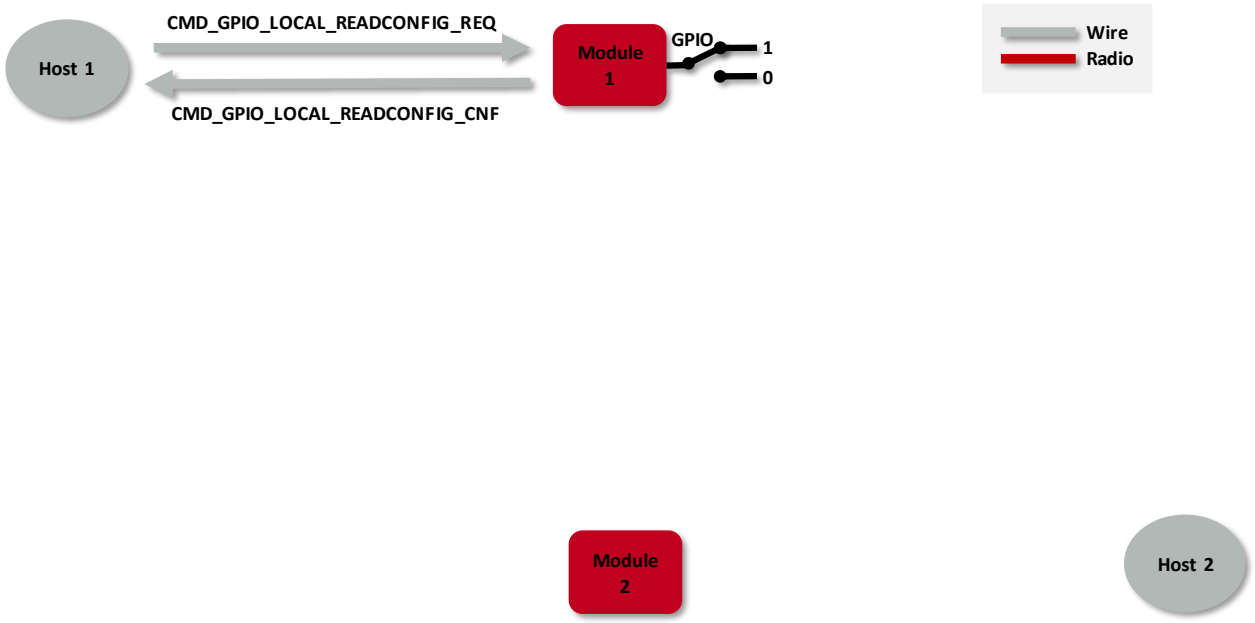


Figure 16: Read the configuration of the local GPIOs via local host

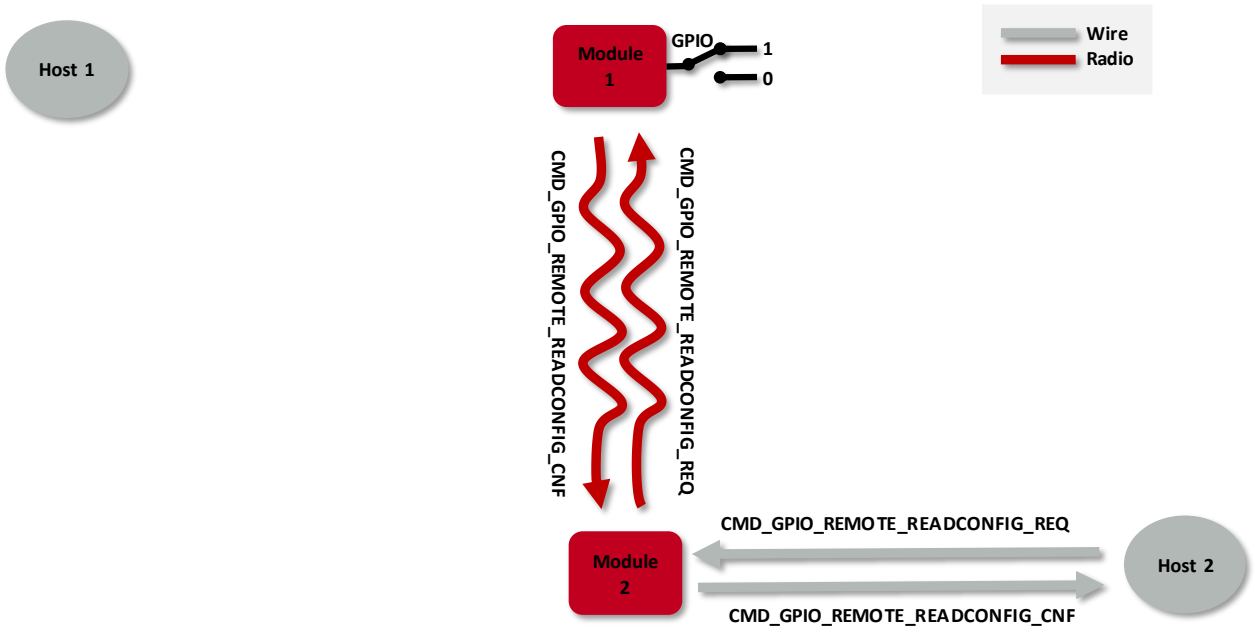


Figure 17: Read the configuration of the local GPIOs via remote device host

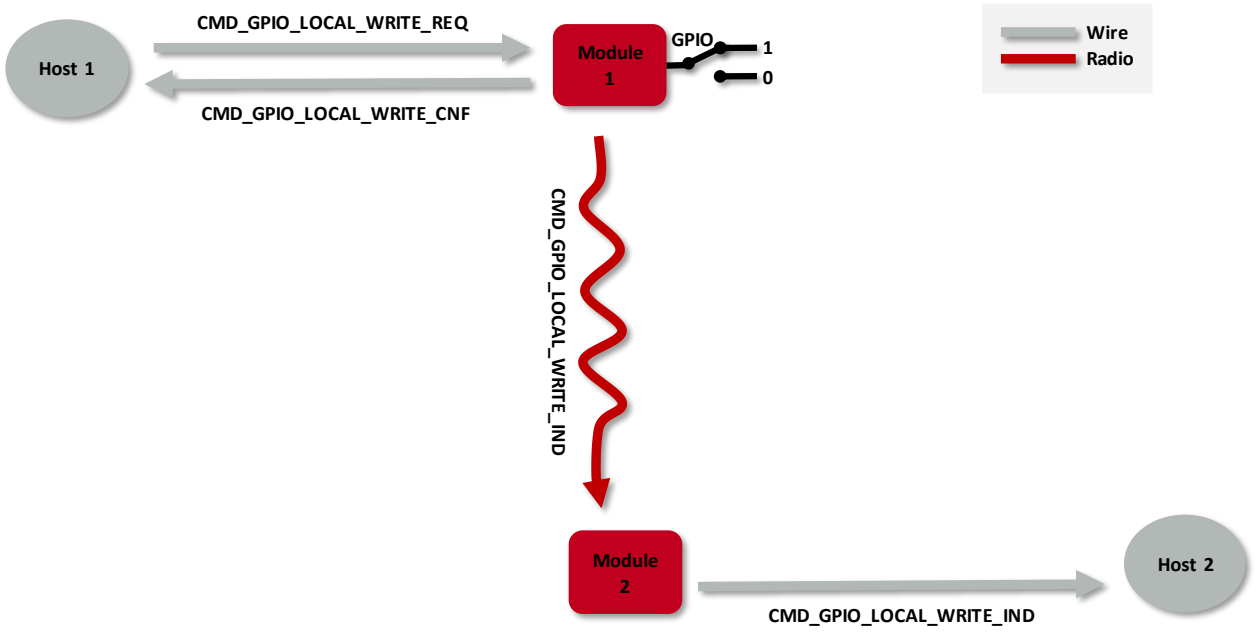


Figure 18: Set the output value of a GPIO via host controller

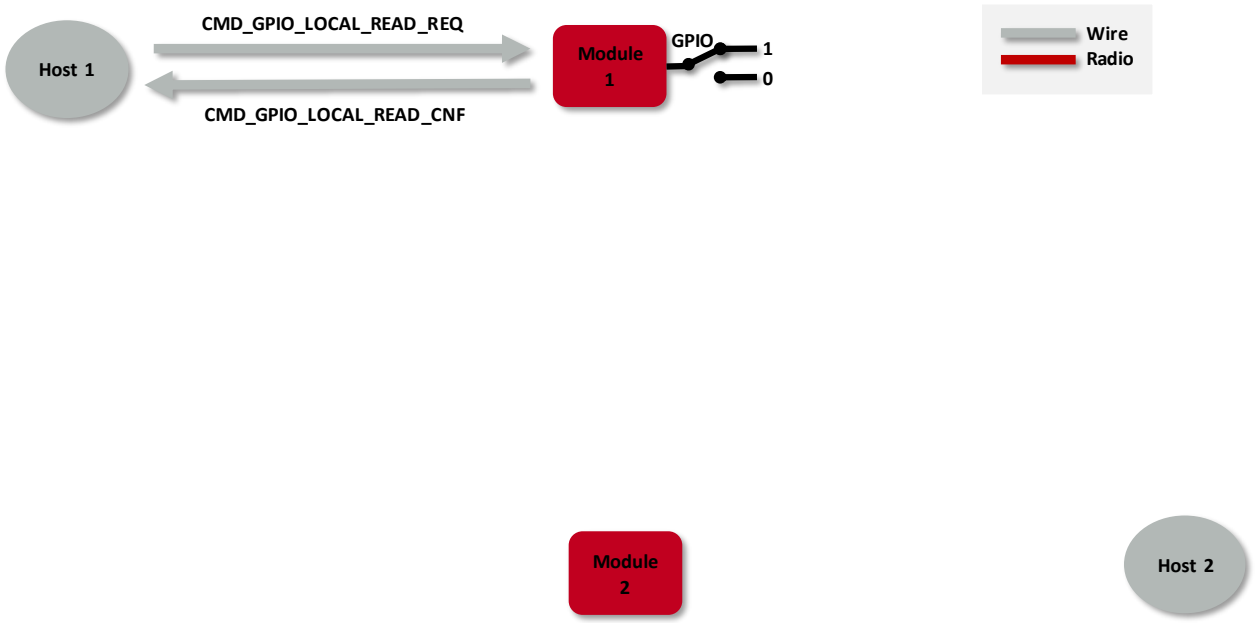


Figure 19: Read the input value of a GPIO via host controller

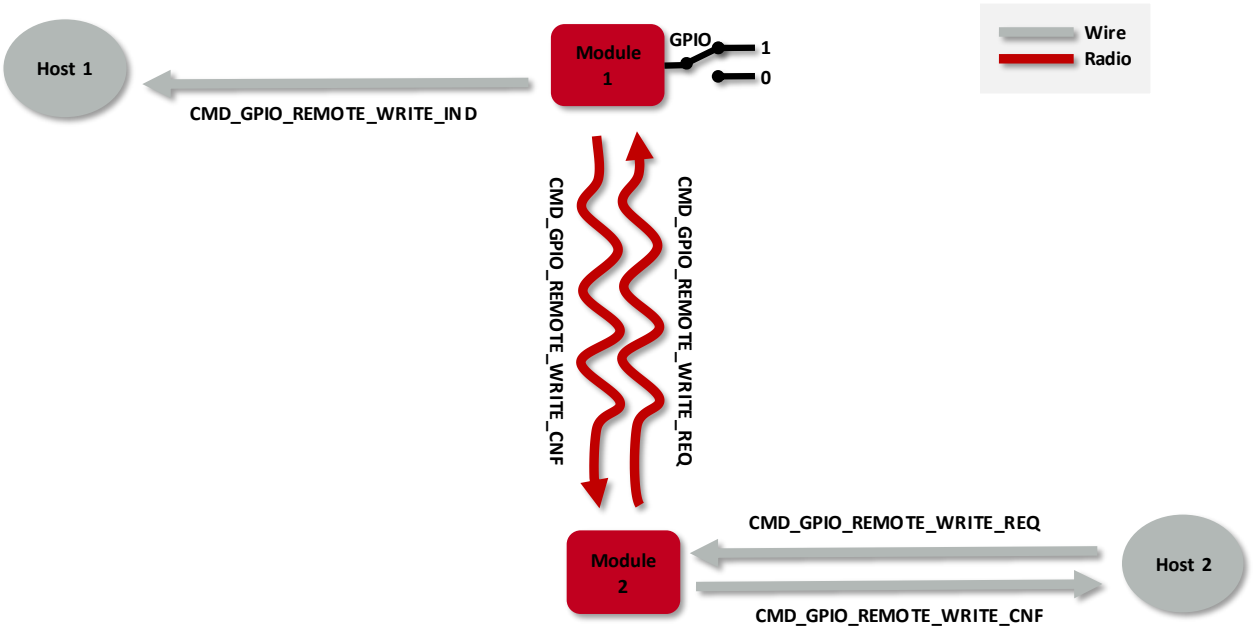


Figure 20: Set the output value of a GPIO via remote device

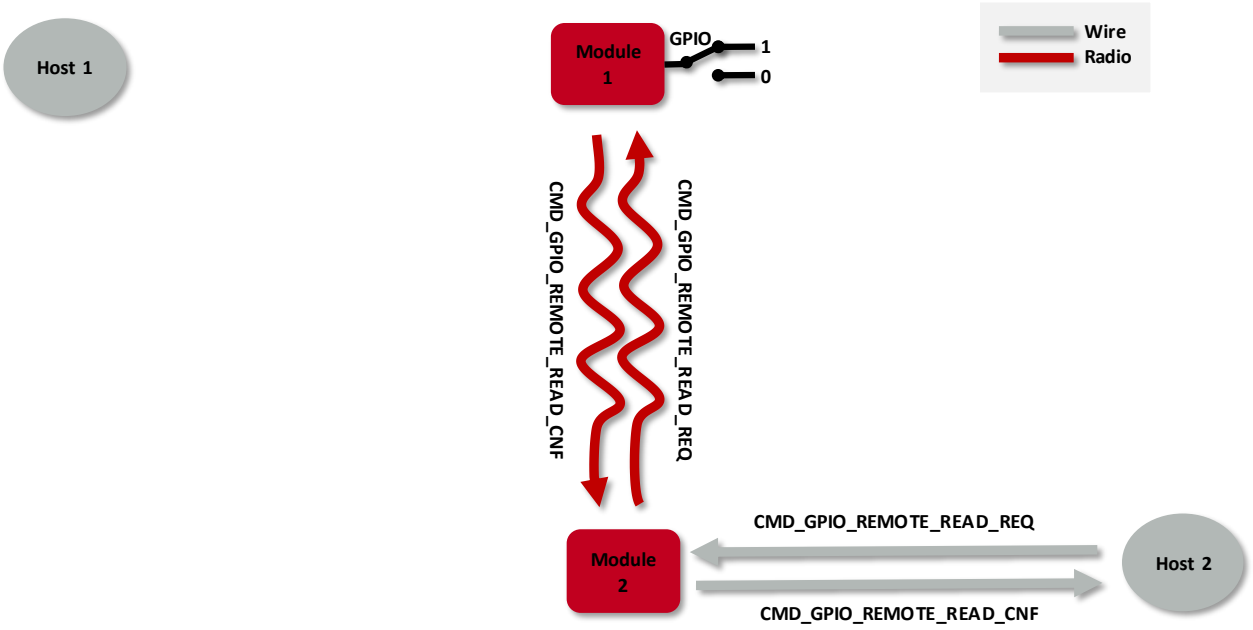
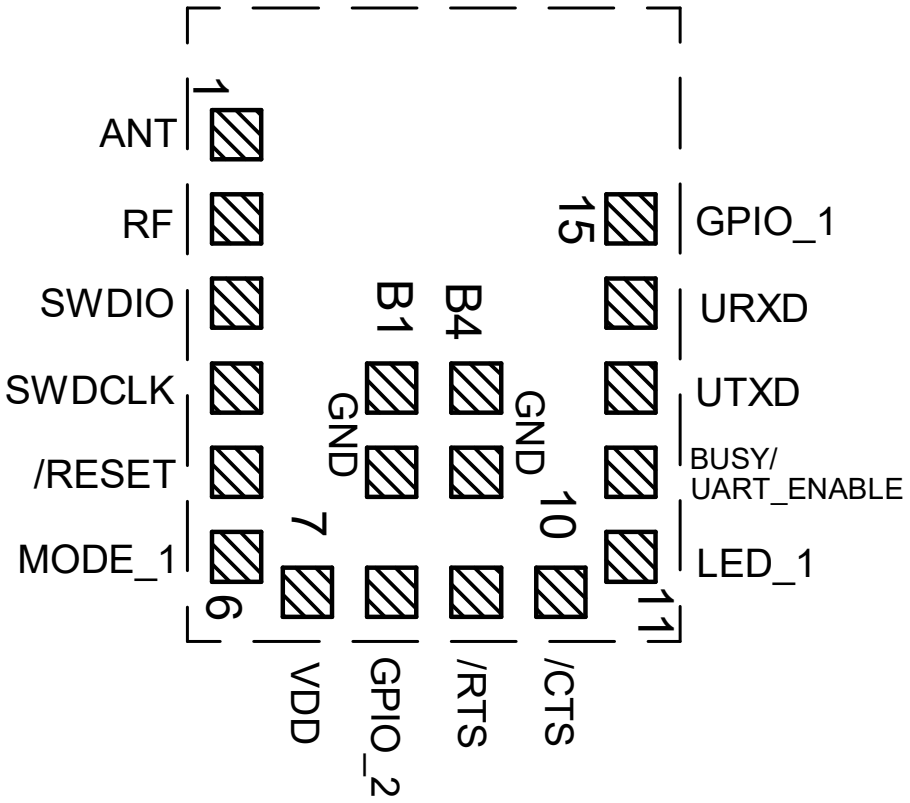


Figure 21: Read the input value of a GPIO via remote device

11.1. Supported GPIO_IDs for remote and local control

The following GPIOs of the Proteus-e are supported for remote and local access.



Pin	GPIO_ID	Supported functions
GPIO_1	1	Input, Output
GPIO_2	2	Input, Output

Table 24: Supported GPIO_IDs

12. Customizing the Proteus-e

12.1. UUID

The UUID is a unique number identifying a Bluetooth® LE profile and thus describing its functions. The Proteus-e using its standard UUID is compatible to all devices that implement the SPP-like profile, whichever device it is integrated.
To suspend this interoperability, the user settings `RF_SPPBaseUUID`, `RF_SPPServiceUUID`, `RF_SPPTXUUID` and `RF_SPPRXUUID` can be used to modify the UUID of the SPP-like profile. With this, a new custom SPP-like profile is defined that is solely known to those that chose the new UUID.

The SPP-like profile consists of the 128 bit base UUID plus the 16 bit UUIDs for the underlying characteristics and services:

Characteristic	UUID
128 Bit `RF_SPPBaseUUID`	0x6E40xxxx-C352-11E5-953D-0002A5D5C51B
16 Bit `RF_SPPServiceUUID`	0x0001
16 Bit `RF_SPPRXUUID`	0x0002
16 Bit `RF_SPPTXUUID`	0x0003

Table 25: UUID default values

Using these user settings, the UUIDs of all characteristics calculate as the base UUID, where byte 2 and 3 are replaced by the underlying service or characteristic UUID.

Example:
With the above mentioned default values, the full UUIDs calculate as

Direction	Characteristic	128 Bit UUID
	Primary service	0x6E40 **0001** -C352-11E5-953D-0002A5D5C51B
Remote peer to module	RX characteristic	0x6E40 **0002** -C352-11E5-953D-0002A5D5C51B
Module to remote peer	TX characteristic	0x6E40 **0003** -C352-11E5-953D-0002A5D5C51B

To generate a custom base UUID the Bluetooth® SIG recommends to use the tool:
http://www.uuidgenerator.net/

12.2. 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 the 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.
This variant will result in a customer exclusive module with a unique ordering number. It will also freeze 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 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 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.

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 updates

All products will experience maintenance, security and/or feature updates from time to time. For the standard products these are maintained via the PCN process.
Customers can request the creation of a customized product including a "firmware freeze" to ensure that they will receive their verified product even if the standard product is updated.

The Proteus-e does not support an update of the production firmware images by users.

14.1. Firmware flashing using the production interface

The Proteus-e offers a serial wire debug and programming interface (SWD) for module flash access. This interface can be used by customers to erase the entire chip and install their own firmware.

This interface is not intended to perform updates of Proteus-e firmware.

Production firmware images and binary files for Würth Elektronik eiSos wireless connectivity modules are not publicly available in general.

Customers performing any flashing of own firmware are fully responsible for any certification, declaration listing and qualification as the Proteus-e documentation becomes invalid. Details to perform a "change in ID" are described in chapter 24.7.3

15. Firmware history

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 particulary 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 malfunction. 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, when using an external antenna the RF-pin could be accessible and should be protected in this case.

- 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 8231606A 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.).

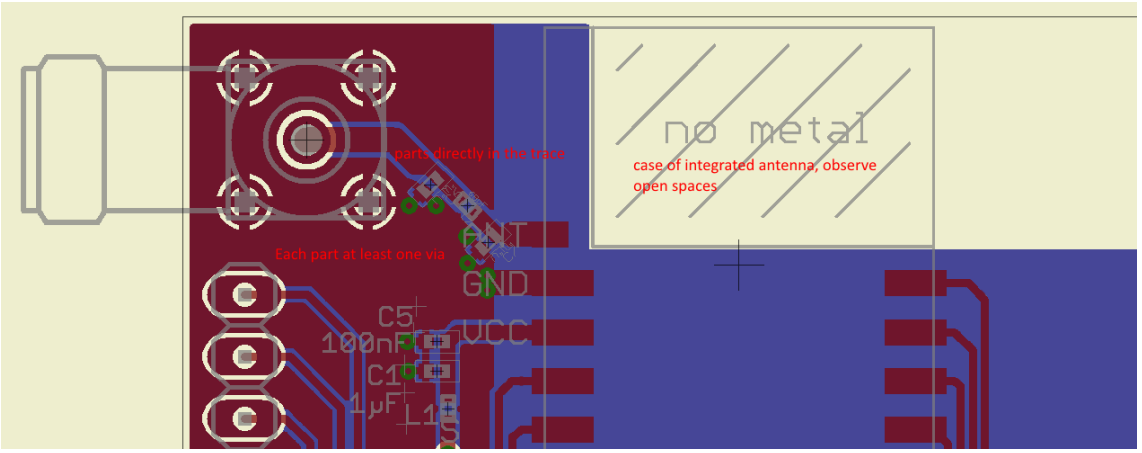


Figure 22: Layout example, Layout of the corresponding evaluation board is published in the evaluaton board manual

- 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.

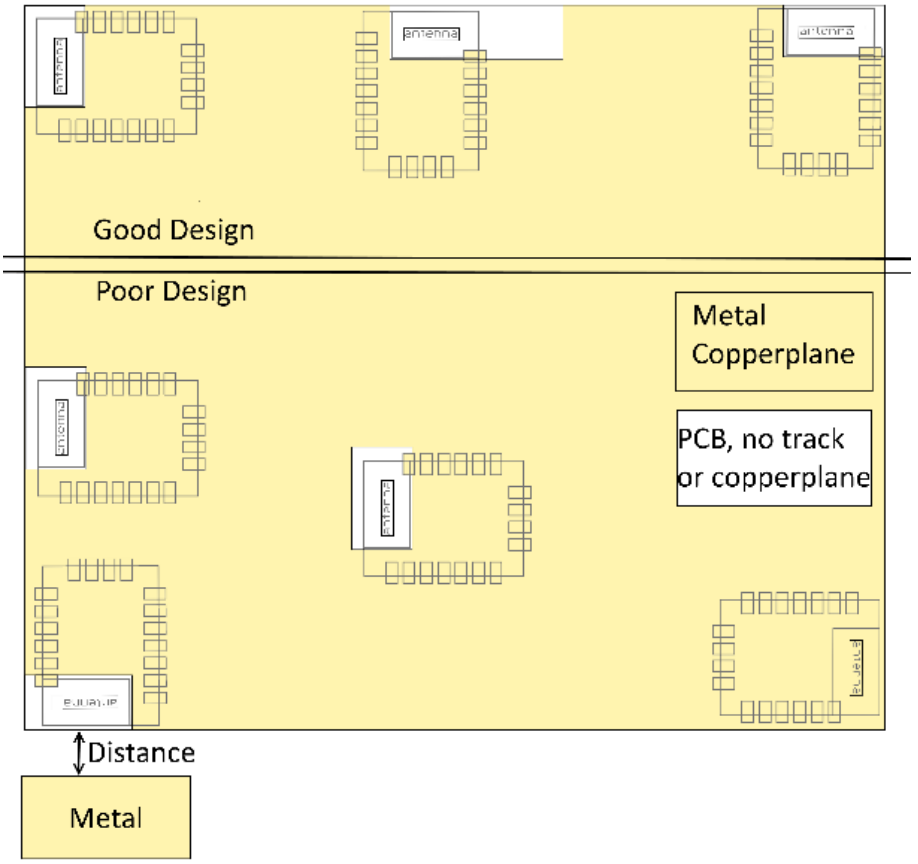


Figure 23: Placement of the module with integrated antenna

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

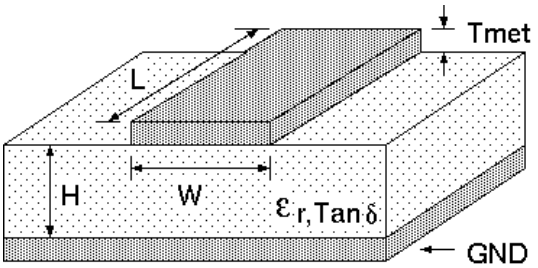


Figure 24: Dimensioning the antenna feed line as micro strip

be calculated using the following equation:

$$W = 1.25 \times \left(\frac{5.98 \times H}{e^{\frac{50 \times \sqrt{\epsilon_r} + 1.41}{87}}} - T_{met} \right) \quad (1)$$

Example:
A FR4 material with ε_r = 4.3, a height H = 1000 µm and a copper thickness of T_{met} = 18 µm will lead to a trace width of W ~ 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 λ/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 behaviour 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. 2600130021 - Himalia - 2.4 GHz dipole antenna

Figure 25: 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	$\leq$ 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. [°C]	-40 – +80

Special care must be taken for FCC certification when using this external antenna to fulfil 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-e was tested and certified on the corresponding Proteus-e evaluation board. For the European Conformity the evaluation board serves as reference design. When reusing Würth Elektronik eiSos FCC or IC certification it is mandatory to follow the trace design.

Complete layout and schematic information can be found in the manual of the Proteus-e evaluation board.

17.1. EV-Board

17.1.1. Schematic

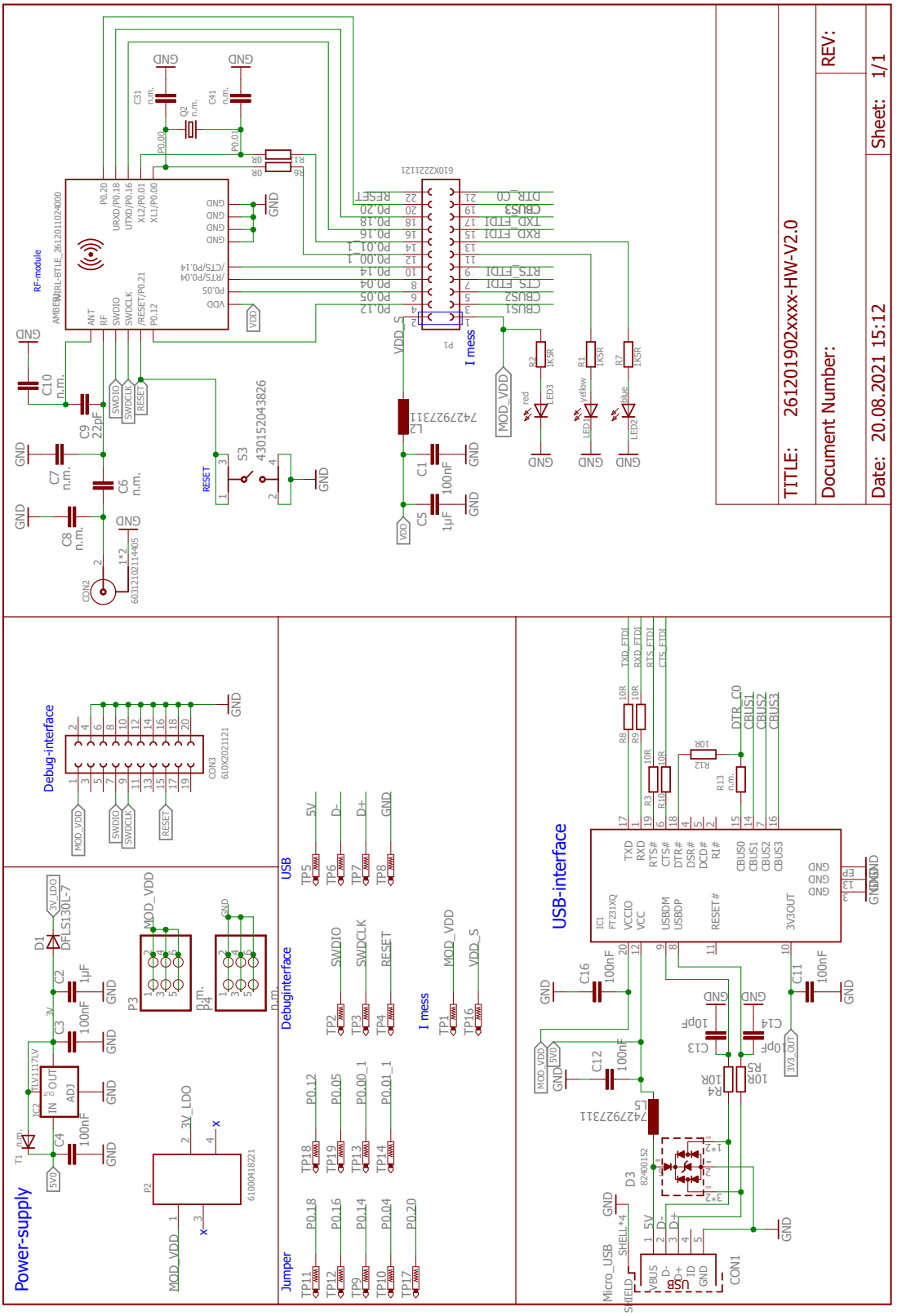


Figure 26: Reference design: Schematic

17.1.2. Layout

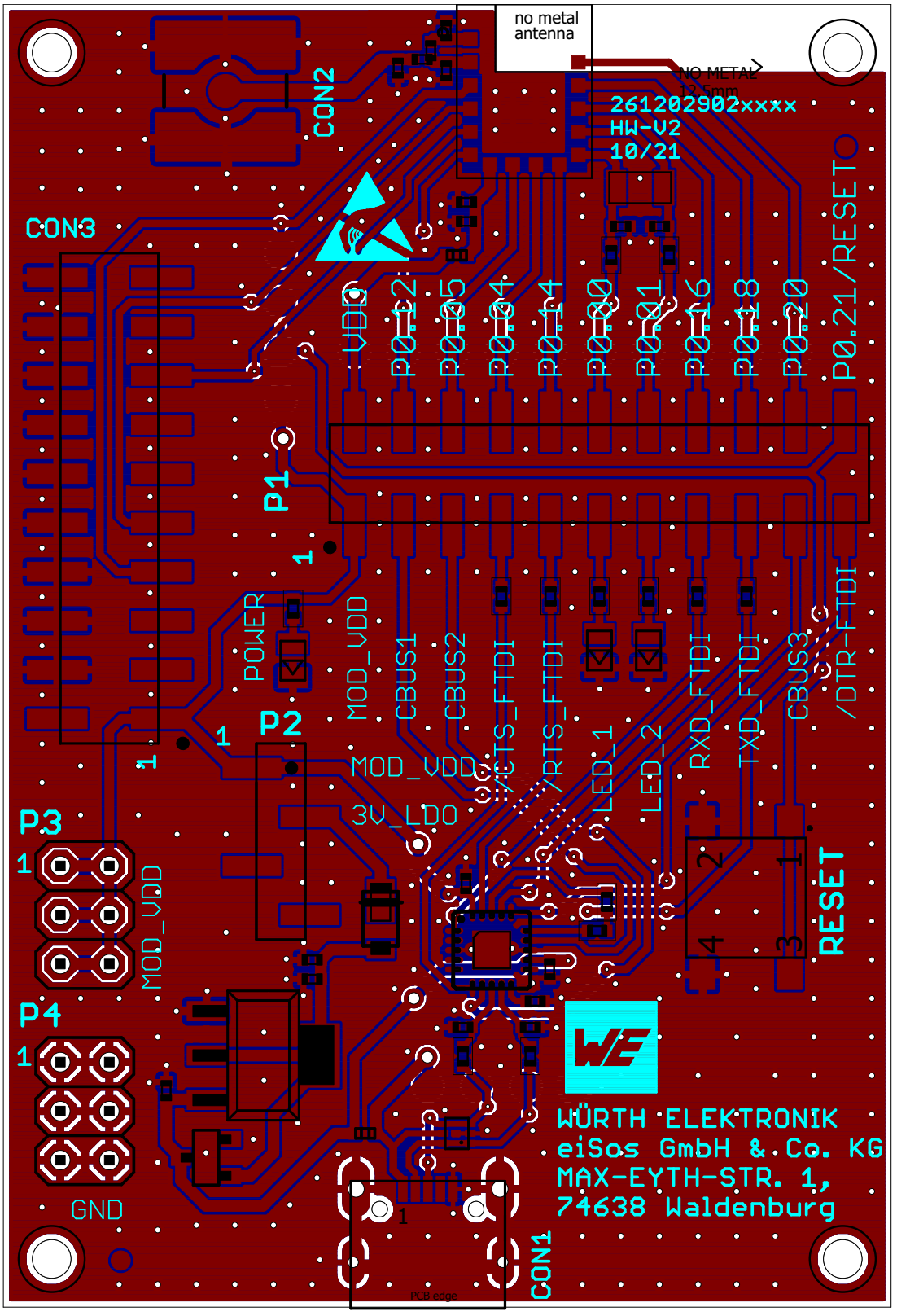


Figure 27: Reference design: Layout

17.2. Trace design

The trace design shown in this chapter the design proved for FCC and IC certification.

To reuse and reference to the Würth Elektronik eiSos' FCC ID it is mandatory to use the trace design.

To provide best options following trace designs are certified:

- A simple short between the pins *RF* and *ANT* pin to be used if size and price is critical and the range is uncritical.
- A 22 pF capacitor connecting *RF* and *ANT* pin to be used if size and price is less critical, but an assembly variant with external antenna is also be used.
- A 22 pF capacitor connecting *RF* pin to the external precertified dipola antenna (Himalia [5] , chapter `16.3.4.1`). This configuration suits best if size and price is less critical, but radio range should be optimized.

The trace designs use the same layer stack up.

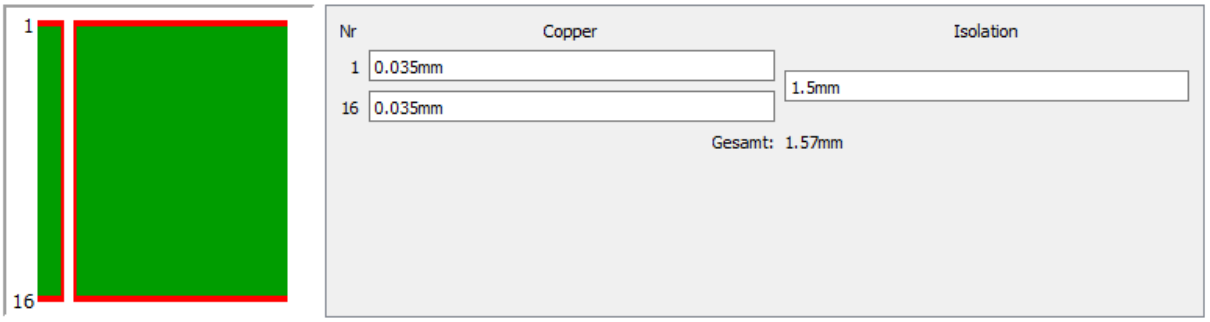


Figure 28: Stack-up

- Top layer is used for routing, filled with ground plane except the area under the module and the antenna free area.
- Bottom layer is the ground plane with as few as possible routing dividing it.

In the following the light green marked areas are the trace designs to be followed when reusing certifications.

17.2.1. Simple short using internal antenna

The simple short is a 50 Ω coplanar strip connecting *RF* and *ANT* pin. Figures 29 and 30 show this in detail.

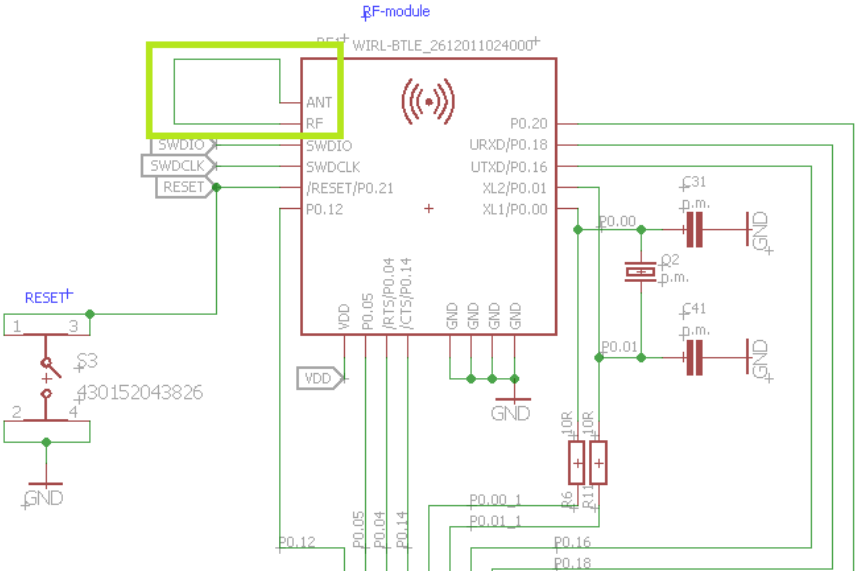


Figure 29: Simple short schematic

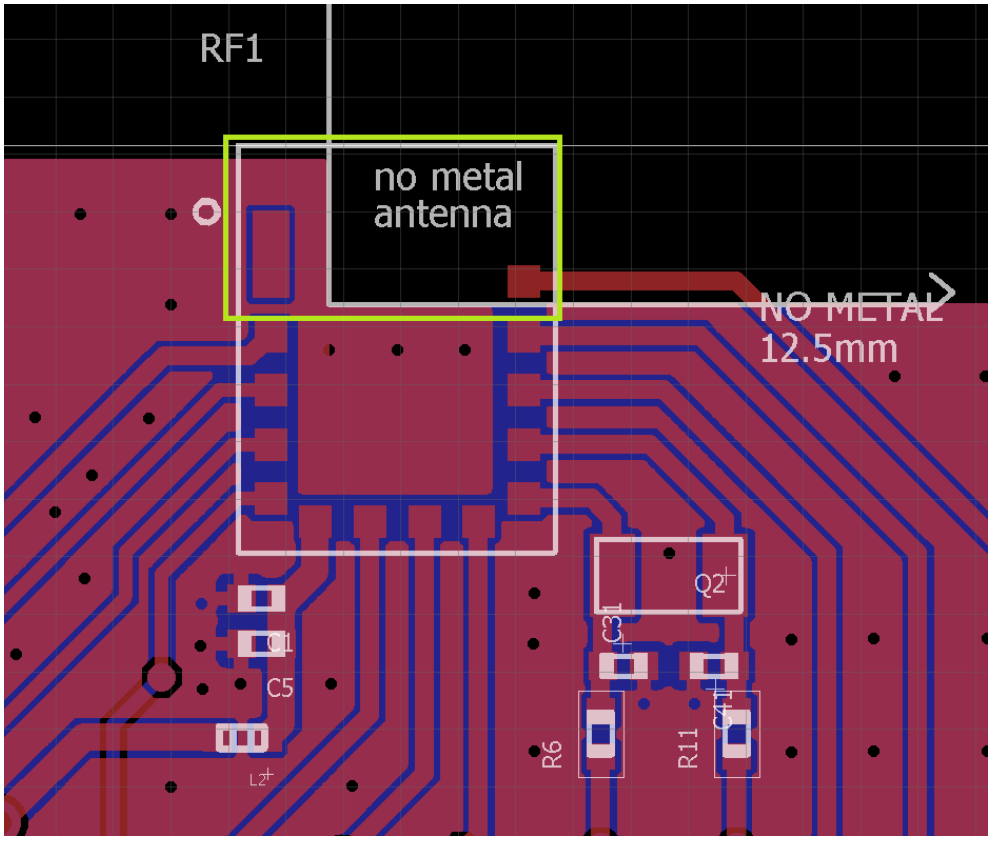


Figure 30: Simple short layout

17.2.2. 22 pF coupling capacitor using internal antenna

In this configuration, instead of the simple short, a 22 pF capacitor is used at C9 connecting *RF* and *ANT* pin. C6, C7, C8 and C10 are left unassembled. Figures 31 and 32 show this in detail.

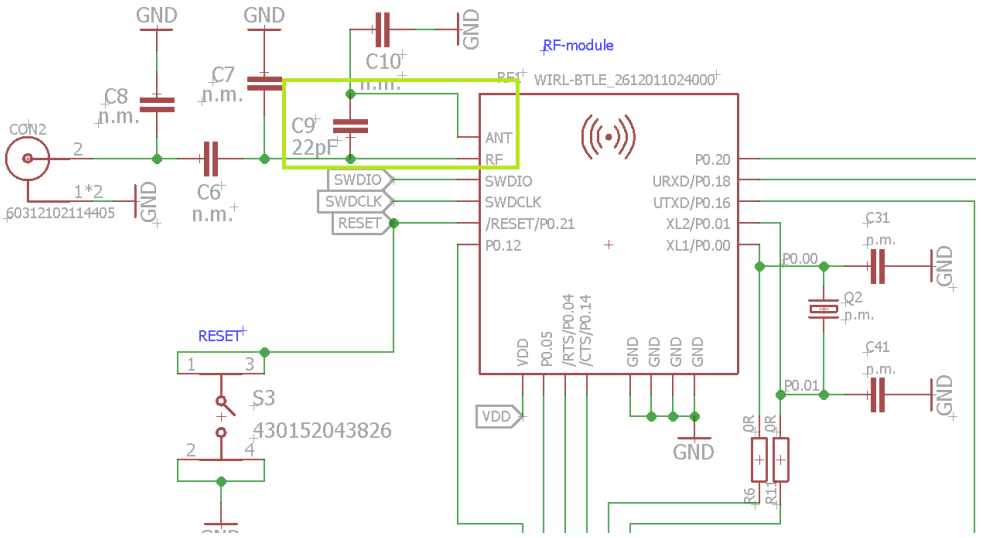


Figure 31: Capacitor internal antenna schematic

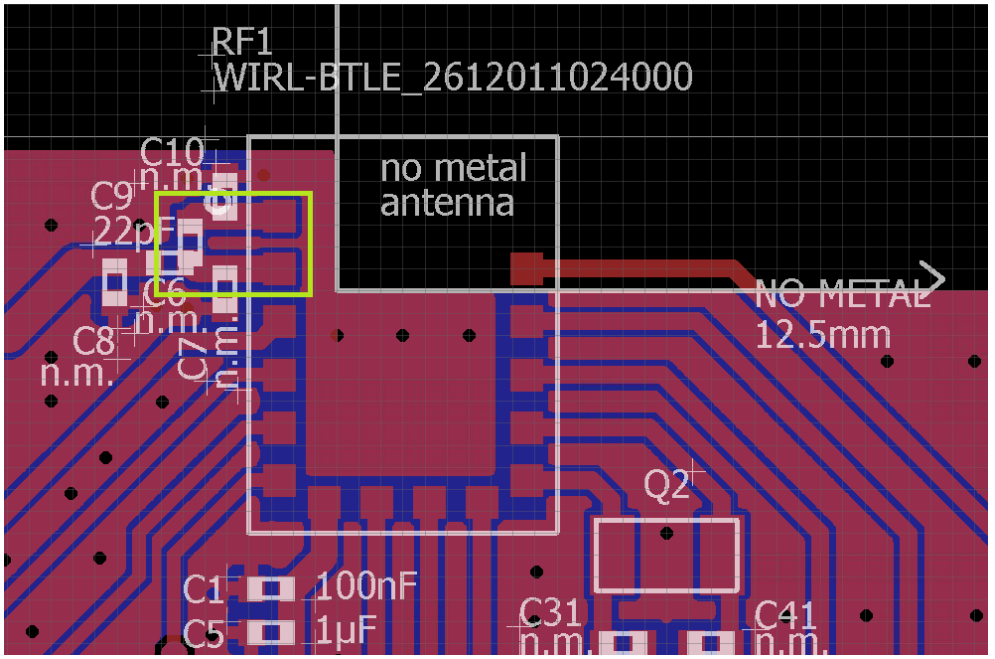


Figure 32: Capacitor internal antenna layout

17.2.3. 22 pF coupling capacitor using external antenna

In this configuration, the 22 pF capacitor is used at C6 connecting the *RF* pin to a dipol antenna (Himalia [5], chapter 16.3.4.1). C7, C8, C9 and C10 are left unassembled. Figures 33 and 34 show this in detail.

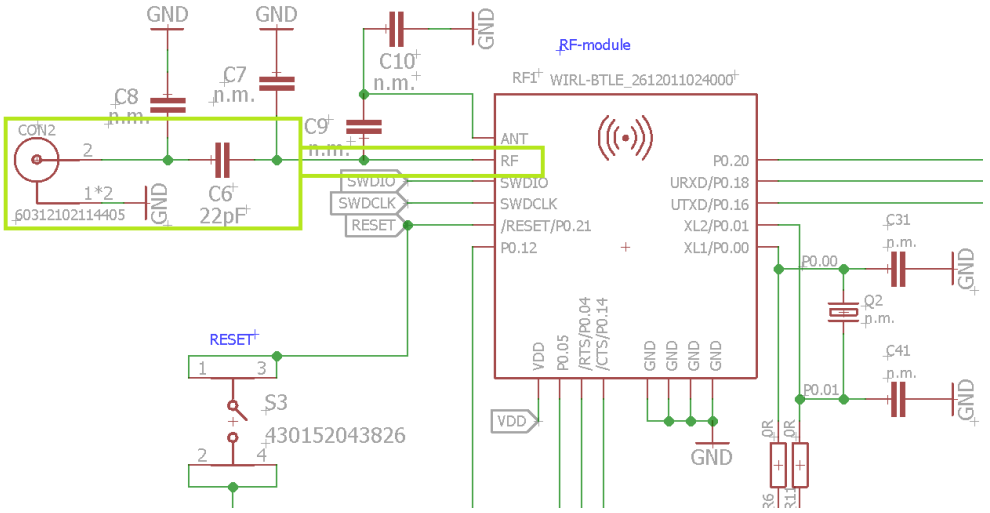


Figure 33: Capacitor external antenna schematic

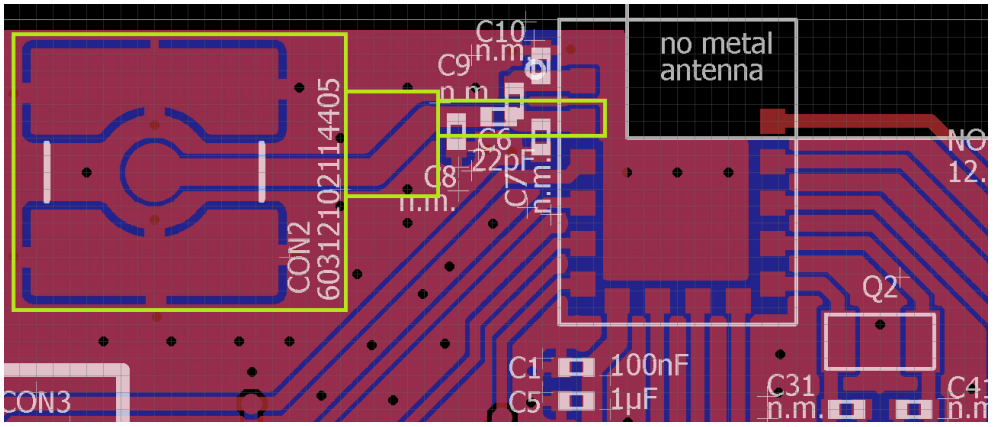


Figure 34: Capacitor external antenna layout

To fullfill §15.203 of FCC, the manufacturer of the end device must ensure that no antenna other than that furnished by the responsible party shall be used with the device. The use of a permanently attached antenna or an antenna, that uses a unique coupling to the end device, shall be considered sufficient to comply.

17.3. Antenna fine tuning

Engineers with experience in radio design and the needed measurement equipment should consider the possibility of antenna tuning. The smart antenna connection provides the possibility to even tune the antenna inside the module.
Due to the influence of mounting conditions as metallic objects close to the antenna, or the size of the mother-pcb and ground plane there might be some detuning of the antenna. Adjusting might be done by measuring the resulting antenna impedance and using corresponding values for C7, C9 and C10 see Figure 31.
This fine tuning is violating the trace design. A radio spot check measurement for the end device is needed.
Using the trace desing option 17.2.2 as implemented on the evaluation board gives the possibility to either follow the trace design or do fine tuning if needed without changing the PCB 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/reflow 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 peak 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 26: 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
< 1.6 mm	260 °C	260 °C	260 °C
1.6 mm - 2.5 mm	260 °C	250 °C	245 °C
> 2.5 mm	250 °C	245 °C	245 °C

Table 27: 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 reflow 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 reflow profile must be adjusted based on the thermal mass of the entire populated PCB, heat transfer efficiency of the reflow 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.

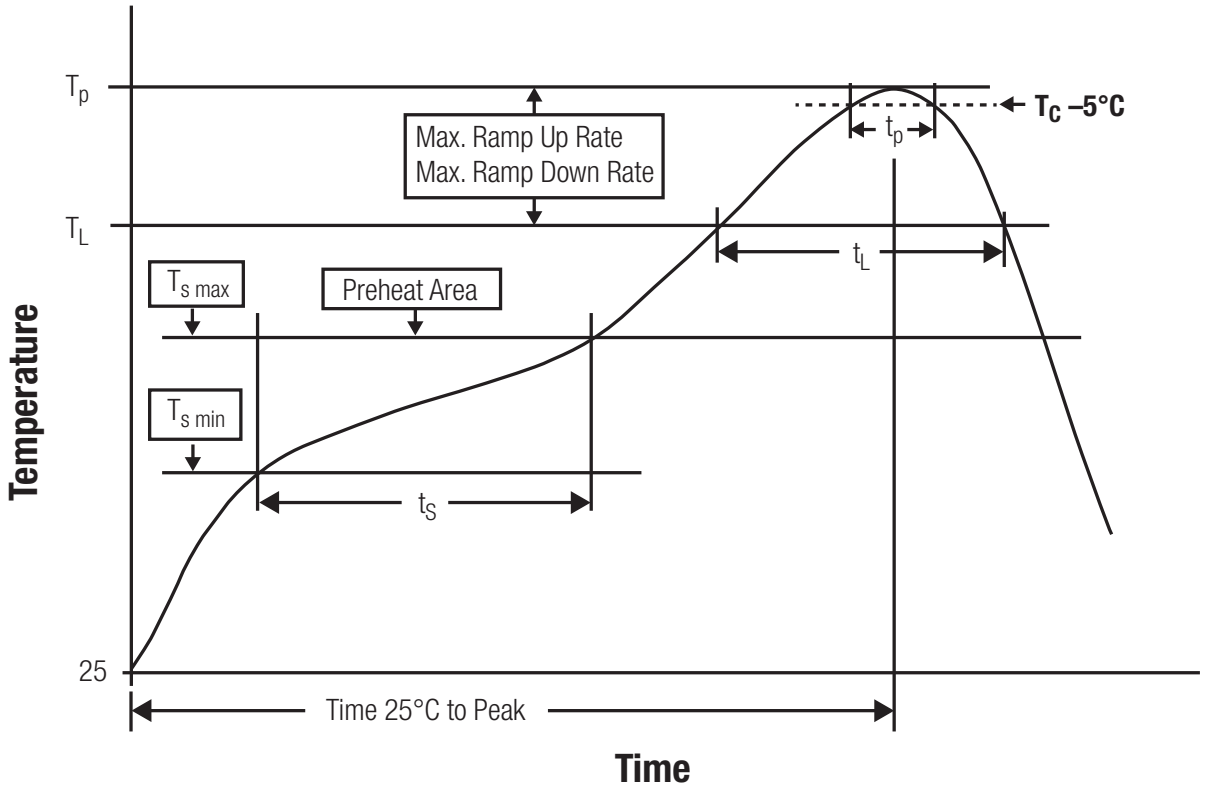


Figure 35: Reflow soldering profile

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. Potting and coating

- If the product is potted in the customer application, the potting material might shrink or expand during and after hardening. Shrinking could lead to an incomplete seal, allowing contaminants into the component. Expansion could damage components. We recommend a manual inspection after potting to avoid these effects.
- Conformal coating or potting results in loss of warranty.
- The RF shield will not protect the part from low-viscosity coatings and potting. An undefined amount of coating and potting will enter inside the shielding.
- Conformal coating and potting will influence the parts of the radio front end and consequently influence the radio performance.
- Potting will influence the temperature behaviour of the device. This might be critical for components with high power.

18.2.4. Other notations

- 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 specifications

19.1. Dimensions

Dimensions
9 x 7 x 2 mm

Table 28: Dimensions

19.2. Weight

Weight
<1g

Table 29: Weight

19.3. Light sensitivity

Inside the Proteus-e a light sensitive WLCSP package is used. This package is sensitive to visible and near infrared light. As the chip is not completely shielded on the sides, any mounting without enclosure could lead to malfunction. This should be taken into account when designing an enclosure for the end device.

19.4. Module drawing

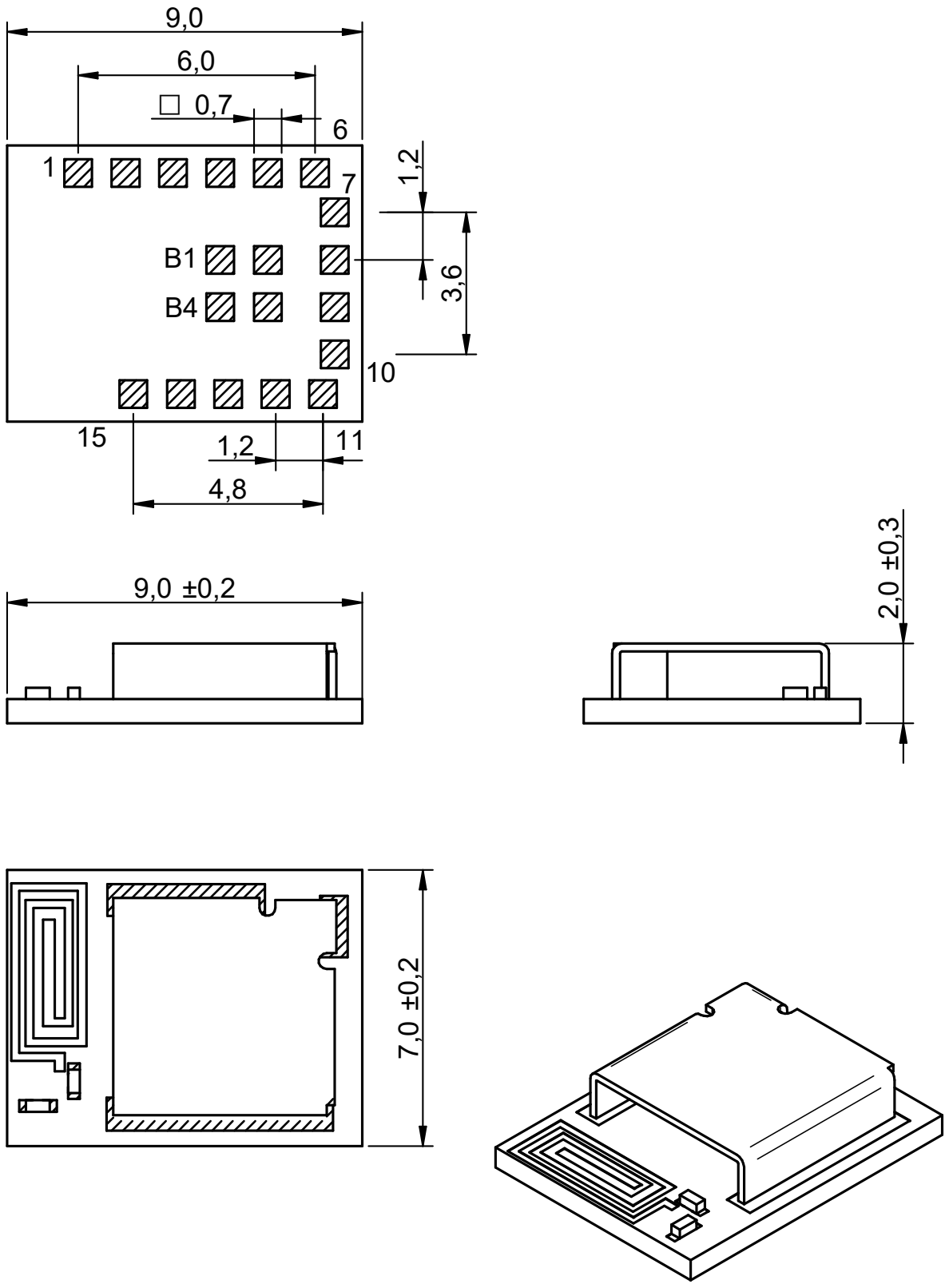


Figure 36: Module dimensions [mm]

19.5. Footprint WE-FP-4+

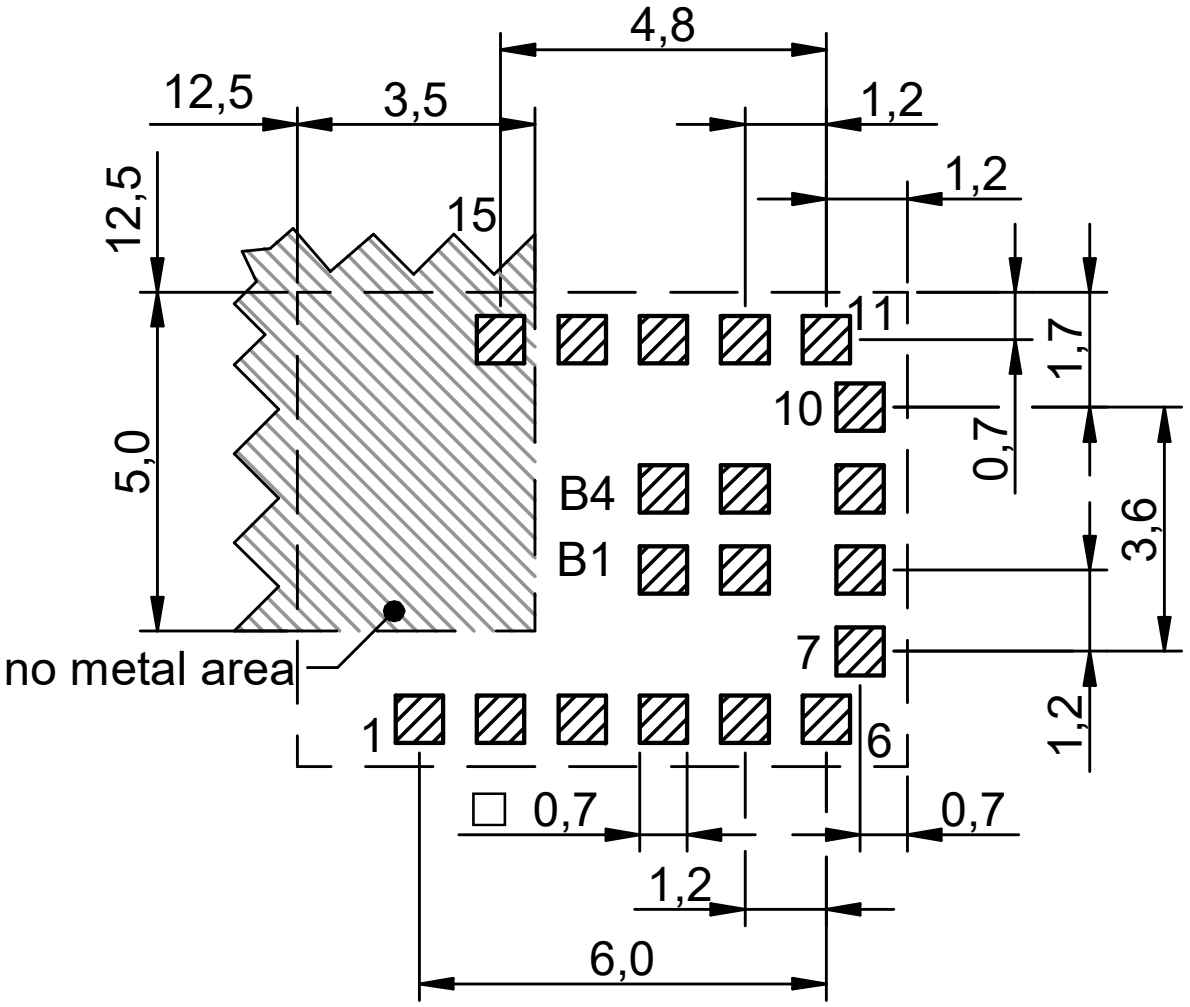


Figure 37: Footprint WE-FP-4+ [mm]

19.6. 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 λ/10 should be kept (see figure 37). 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.

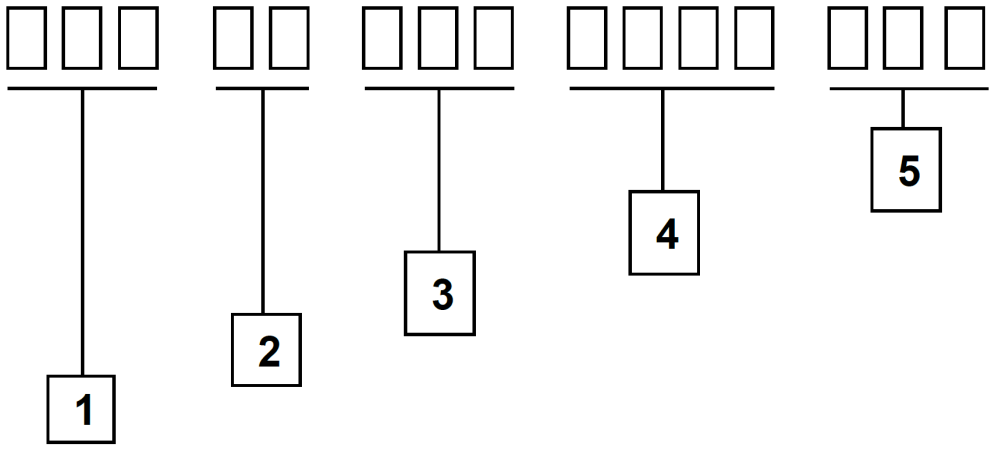


Figure 38: Lot number structure

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 30: 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

If the module is using a serial number, this serial number includes the product ID (PID) and an 6 digit number. The 6 rightmost digits represent the 6 digit number, followed by the product ID (2 or 3 digits). Some labels indicate the product ID with a "." as marker in-between the 2 fields. The PID and the 6 digit number form together a unique serial number for any wireless connectivity product.
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.

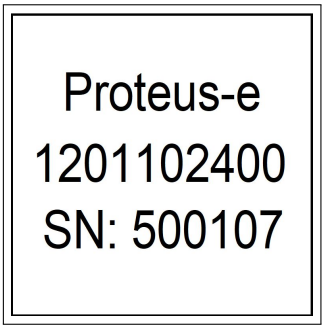


Figure 39: Label of the Proteus-e

21. Information for explosion protection

In case the end product should be used in explosion protection areas the following information can be used:

- The module itself is unfused.
- The maximum output power of the module is 5 dBm for radio pad .
- The total amount of capacitance of all capacitors is 6.8 µF.
- The total amount of inductance of all inductors is 10.009 µH.
- A DC/DC regulator is included in the chip set and used to obtain low power functionality.

22. References

[1] Bluetooth. Bluetooth Core Specification, version 5.1. https://www.bluetooth.com/specifications/specs/core-specification-5-1/.

[2] Würth Elektronik. Application note 24. http://www.we-online.com/ANR024.

[3] Würth Elektronik. Application note 25. http://www.we-online.com/ANR025.

[4] Würth Elektronik. Application note 27. http://www.we-online.com/ANR027.

[5] Würth Elektronik. Himalia. https://www.we-online.com/catalog/en/WIRL_ACCE_2600130021.

[6] Würth Elektronik. Wireless Connectivity SDK for Raspberry Pi - Radio module drivers in C-code. https://github.com/WurthElektronik/WirelessConnectivity-SDK.

[7] Würth Elektronik. Wireless Connectivity SDK for STM32 - Radio module drivers in C-code. https://github.com/WurthElektronik/WirelessConnectivity-SDK_STM32.

23. Bluetooth SIG listing/qualification

Type	Data
Design name	Proteus-e
Declaration ID	D057370
QDID	177585
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.
Due to the qualification of the Proteus-e as end product no further Bluetooth® tests are required. The only arising expenses are those for purchasing a Bluetooth® Declaration ID.
To obtain the Bluetooth® listing of the end device, please refer to the application note ANR027 [4].

24. Regulatory compliance information

24.1. Important notice EU

The use of RF frequencies is limited by national regulations. The Proteus-e 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-e 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.

24.2. Important notice FCC

The use of RF frequencies is limited by national regulations. The Proteus-e has been designed to comply with the FCC Part 15.
The Proteus-e 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.

24.3. Conformity assessment of the final product

The Proteus-e is a subassembly. It is designed to be embedded into other products (products incorporating the Proteus-e 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-e carried out by Würth Elektronik eiSos does not replace the required conformity assessment of the final product.

24.4. 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.

24.5. FCC Compliance Statement

FCC ID: R7T1201102

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.

24.6. IC Compliance Statement

Certification Number: 5136A-1201102
HVIN: 1201102
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.

24.7. 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: R7T1201102
  This equipment contains equipment certified under ICID: 5136A-1201102

- 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.

24.7.1. OEM requirements:

The OEM must ensure that the following conditions are met.

- The Proteus-e 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: R7T1201102
  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 sign the OEM Modular Approval Agreement.
  - The module must be used with only the following approved antenna(s).

24.7.2. Pre-certified antennas

The Proteus-e is pre-certified with the following antennas.

Product	Certified antenna
Proteus-e (2612011024000)	PCB antenna included in the Proteus-e

24.7.3. Change in ID / Multiple listing

Experienced developers have the possibility to write and flash their own firmware. This firmware will have full control of radio parameters. For an FCC/ISED approved radio module the ID holder, in this case Würth Elektronik eiSos, is solely responsible to ensure that the product complies with the regulations. Since the firmware application created by integrators of the Proteus-e cannot be controlled by Würth Elektronik eiSos, the module integrator will have to take over this responsibility. This process is called "change in ID" for FCC and "multiple listing" for ISED.
Test houses with corresponding accreditation can provide the service for this process. It is also possible to make an application by oneself to a Telecommunication Certification Body. Usually following documentations are needed:

- A permission letter, signed by Würth Elektronik eiSos, allowing the change in ID
- An application letter stating that there is no change in the design, circuitry, or construction of the Proteus-e, and that the original Proteus-e test results are still representative for the new product. Minor differences are allowed but must be described.
- Photos of the product showing the nameplate or label containing the new FCC ID.
- User manual for the new product.

25. Important notes

The following conditions apply to all goods within the wireless connectivity product range of Würth Elektronik eiSos GmbH & Co. KG:

25.1. 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.

25.2. 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.

25.3. 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.

25.4. 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.

25.5. 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 25.1 and 25.2 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.

25.6. 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.

25.7. 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.

25.8. 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*.

26. Legal notice

26.1. 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.

26.2. 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.

26.3. 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.

26.4. 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 a failure of the product is reasonably expected to cause severe personal injury or death,

unless the parties have executed an agreement specifically governing such use. Moreover, Würth Elektronik eiSos GmbH & Co. KG products 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. Würth Elektronik eiSos GmbH & Co. KG must be informed about the intent of such usage before the design-in stage. In addition, sufficient reliability evaluation checks for safety must be performed on every electronic component, which is used in electrical circuits that require high safety and reliability function or performance. By using Würth Elektronik eiSos GmbH & Co. KG products, the customer agrees to these terms and conditions.

27. 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.

27.1. 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.

27.2. 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.

27.3. Ownership

The incorporated Firmware created by Würth Elektronik eiSos is and will remain the exclusive property of Würth Elektronik eiSos.

27.4. 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.

27.5. 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.

27.6. 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.

27.7. 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.

27.8. 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.

27.9. 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

A. Additional CRC8 Information

This Annex gives an example CRC8 implementation and test vectors.

A.1. Example CRC8 Implementation

```
#include <stdint.h>

uint8_t  Get_CRC8(uint8_t * bufP, uint16_t len)
{
    uint8_t  crc = 0x00;
    for (uint16_t i = 0; i < len; i++)
    {
        crc ^= bufP[i];
    }
    return crc;
}
```

Code 1: Example CRC8 Implementation

A.2. CRC8 Test Vectors

Input data	Data length	Resulting CRC8
Null	0	0x00
0x02 0x01 0x00 0x00	4	0x03
0x02 0x87 0x01 0x00 0x16	5	0x92
0x02 0x04 0x04 0x00 0x41 0x42 0x43 0x44	8	0x06
0x02 0x88 0x07 0x00 0x00 0x55 0x00 0x00 0xDA 0x18 0x00	11	0x1A

Table 31: CRC8 Test Vectors

B. Example codes for host integration

The following code is an example implementation of a function to transmit data using a 2 Byte length field in the command frame. For demonstration reasons the Proteus-III has been taken. The full function codes of all radio modules are available in the Wireless Connectivity SDK (*www.we-online.de/wco-SDK*).

```
#define CMD_PAYLOAD_MAX 964
typedef struct {
    uint8_t   Stx;
    uint8_t   Cmd;
    uint16_t  Length;                    /* LSB first */
    uint8_t   Data[CMD_PAYLOAD_MAX+1]; /* +1 for CRC8 */
} CMD_Frame_t;
#define CMD_OFFSET_TO_DATAFIELD 4
#define CMD_OVERHEAD (CMD_OFFSET_TO_DATAFIELD+1)

bool ProteusIII_Transmit(uint8_t *PayloadP, uint16_t length)
{
    /* fill request message with STX, command byte and length field */
    CMD_Frame_t CMD_Frame;
    CMD_Frame.Stx = CMD_STX; /* 0x02 */
    CMD_Frame.Cmd = ProteusIII_CMD_DATA_REQ; /* 0x04 */
    CMD_Frame.Length = length;

    /* fill request message with user payload */
    memcpy(CMD_Frame.Data, PayloadP, length);

    /* fill request message with CRC8 */
    CMD_Frame.Data[CMD_Frame.Length] = Get_CRC8(&CMD_Frame, CMD_Frame.Length +
        CMD_OFFSET_TO_DATAFIELD);

    /* transmit full message via UART to radio module */
    UART_SendBytes(&CMD_Frame, (CMD_Frame.Length + CMD_OVERHEAD));

    /* wait for response message from radio module */
    return UART_Wait_for_Response(CMD_WAIT_TIME, ProteusIII_CMD_TXCOMPLETE_RSP,
        CMD_Status_Success, true);
}
```

Code 2: Example function implementation for radio modules with 2 byte length field