PROTEUS-I REFERENCE MANUAL

AMB2621 / 2608011X2400X

VERSION 3.10

DECEMBER 21, 2020

Revision history

Manual version	FW version	HW version	Notes	Date
1.2	1.0.0 - 1.1.0	2.0	• Added more detailed electrical specifications	August 2016
2.1	2.0.0 - 2.1.0	2.1	• Better description of device states	November 2016
2.6	3.0.0	2.1	• Adapted certification chapters	July 2017
2.10	3.3.0 - 3.3.6	2.1	• Adapted footprint and antenna free area and measured TX power	March 2018
2.17	3.4.0	2.1	• New corporate design and structure • Added description of new features of firmware version 3.4.0 and more information about certification and Bluetooth® declaration	June 2018
2.20	3.4.0	2.1	• Added ARIB certification details and labeling information	September 2018
3.0	3.4.0	2.1	• Updated name from AMB2621 to Proteus-I	November 2018
3.1	3.4.0	2.1	• Corrected `CMD_SET_CNF` message in chapter 8 • Updated description of firmware update using the OTA bootloader	January 2019
3.2	3.4.0	2.1	• Added chapter `Reference design` and `Information for Ex protection` • Updated decription of chapter `Peripheral only mode`	March 2019

Manual version	FW version	HW version	Notes	Date
3.3	3.5.0	2.1	• Added information concerning firmware version 3.5.0 in chapter `Firmware history` • Updated chapter `Important notes`	May 2019
3.4	3.5.0	2.1	• Update references to new AppNote name structure.	June 2019
3.5	3.5.0	2.1	• Corrected information on brownout and maximum TX power in chapter `Electrical specifications` • Updated label in chapter `General labeling information` • Corrected example of DTM RX test in chapter `CMD_DTM_REQ` • Updated address of Division Wireless Connectivity & Sensors location	October 2019
3.6	3.5.0	2.1	• Removed -30dBm as valid `RF_TXPower` value	December 2019
3.7	3.5.0	2.1	• Correction of Value amount of inductivity for ex protection	February 2020
3.8	3.5.0	2.1	• Limitation of the `RF_DeviceName` to a maximum of 31 bytes • Added Annex `Additional CRC8 Information` and `Example codes for host integration`	June 2020
3.9	3.5.0	2.1	• Updated Declaration of EU conformity to latest Version of EN 300 328 after successfully passing corresponding delta test in chapter `Regulatory compliance information`. • Added package name in chapter `Footprint WE-FP-4`.	October 2020

Manual version	FW version	HW version	Notes	Date
3.10	3.5.0	2.1	• Updated Declaration of EU conformity `Regulatory compliance information`.	December 2020

⋆ 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.
DTM	Direct test mode	Mode to test Bluetooth® specific RF settings.
GAP	Generic Access Profile	The GAP provides a basic level of functionality that all Bluetooth® devices must implement.
I/O	Input/output	Pinout description.
LPM	Low power mode	Mode for efficient power consumption.
MAC		MAC address of the module.
MTU	Maximum transmission unit	Maximum packet size of the Bluetooth® connection.
Payload		The intended message in a frame / package.
RF	Radio frequency	Describes wireless transmission.
RSSI	Receive Signal Strength Indicator	The RSSI indicates the strength of the RF signal. Its value is always printed in two's complement notation.
Soft device		Operating system used by the nRF52 chip.
User settings		Settings to configure the module. Any relation to a specific entry in the user settings is marked in a special font and can be found in chapter 8.
UART	Universal Asynchronous Receiver Transmitter	Allows the serial communication with the module.
[HEX] 0xhh	Hexadecimal	All numbers beginning with 0x are hexadecimal numbers. All other numbers are decimal, unless stated otherwise.

Contents

1. Introduction

1.1. Operational description

The Proteus-I exists in two variants, one variant with integrated PCB-antenna, and the other variant with 50Ω connection to an external antenna. For the general functionality there is no difference between the variants.
The Proteus-I module is a radio sub module/device for wireless communication between devices such as control systems, remote controls, sensors etc. . On the basis of Bluetooth® LE 4.2 it offers a fast and secure data transmission of data packages between two or more parties (point to point topology). A serial interface (UART) is available for communication with the host system.
The Proteus-I uses the Bluetooth® LE standard to provide general data transmission between several devices. The standard itself offers a wide range of configurations and possibilities to suit and optimize sophisticated customer applications. To fulfill the needs and specifications of such applications a tailored firmware can be developed on the basis of the Proteus-I hardware. This includes the connection and communication to custom sensors, custom Bluetooth® LE profiles, timing configurations, security configuration as well as power consumption optimizations.

1.1.1. Key features

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

SPP-like connection-based secured data transmission: The Proteus-I firmware implements an SPP-like Bluetooth® LE profile that allows the bidirectional data transmission between several Proteus-I and/or to other Bluetooth® LE devices implementing the AMBER SPP profile. Any module in the network can initiate connection setup. Secured connections allow the transmission of encrypted data (user-defined key or pairing).

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

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

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

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

Latest microprocessor generation provided by Nordic Semiconductor nRF52 series: The heart of the Proteus-I is a Bluetooth® LE chip of the nRF52 series offering high performance values combined with low power consumption. It is a 32 Bit ARM Cortex-M4F CPU with 512kB flash + 64kB RAM and up to 4dBm output power.

Bluetooth® 4.2 stack: The Bluetooth® 4.2 stack enables fast and energy efficient data transmission using state-of-the-art technology of Nordic Semiconductors.

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

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

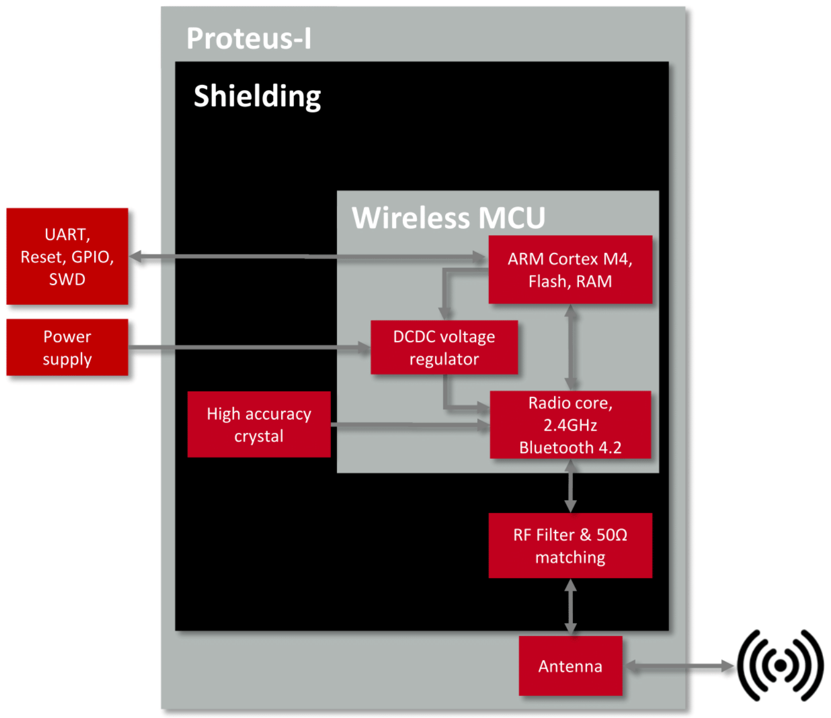
OTA firmware update: The Proteus-I firmware provides over the air firmware update capabilities. Firmware updates can be applied using the Nordic Apps for cell phones.

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

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-I devices, the AMBER SPP-like profile must be known and implemented by all network participants. Thus arbitrary Bluetooth® LE devices (like iOS or Android devices) must implement this profile, too. To do so, the Proteus-I application note 1 contains the design data of the AMBER SPP-like profile.

1.2. Block diagram



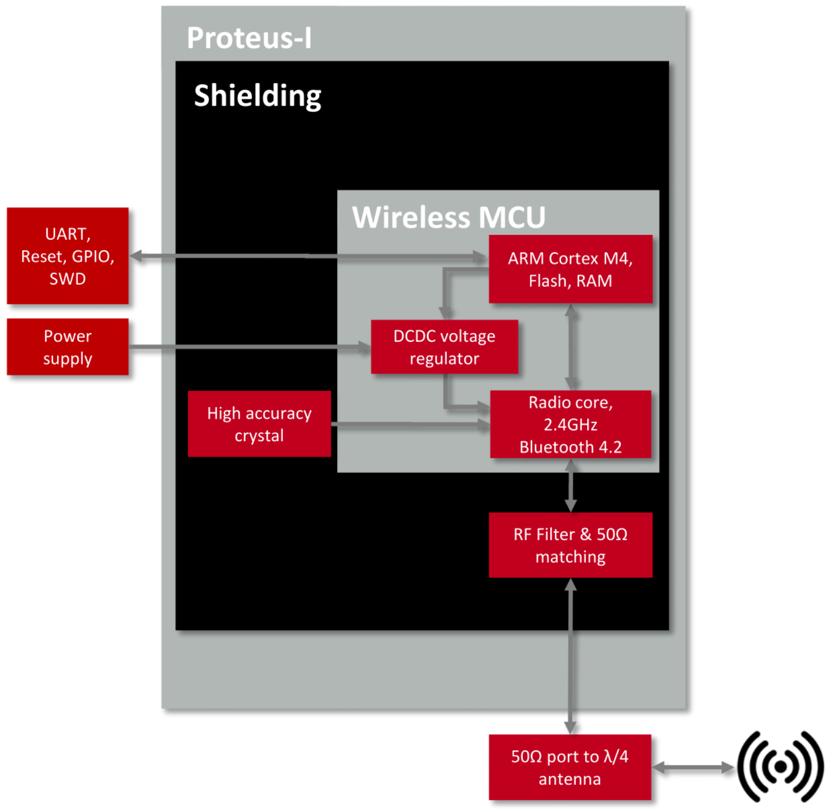


Figure 1: Block diagram of the module with internal PCB antenna and antenna pad

1.3. Ordering information

WE order code	Former order code	Description
2608011024000	AMB2621-TR	Same as AMB2621-TR, but not Tape & Reel
2608011124000	AMB2621-1-TR	Same as AMB2621-1-TR, but not Tape & Reel
2608011024009	AMB2621-DEV	Development Kit including 3×AMB2621
2608011124009	AMB2621-1-DEV	Development Kit including 3×AMB2621-1
2608019024001	AMB2621-EV	Bluetooth® Smart Evaluation Board with AMB2621
2608019324001	AMB2621-1-EV	Bluetooth® Smart Evaluation Board with AMB2621-1

Table 1: Ordering information

2. Electrical specifications

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

2.1. Recommended operating conditions

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

Table 2: Recommended operating conditions

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

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

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

2.2. Absolute maximum ratings

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

Table 3: Absolute maximum ratings

2.3. Power consumption

2.3.1. Static

Continuous test mode	Min.	Typ.	Max.	Unit
TX current consumption at +4 dBm		7.5[1]		mA
TX current consumption at 0 dBm		5.3[1]		mA
RX current consumption		5.4[1]		mA
Sleep (system off mode)		0.4		µA
TX current consumption at +4 dBm		11[2]		mA
TX current consumption at 0 dBm		8[2]		mA
RX current consumption		8[2]		mA

Table 4: Power consumption for 100% transmission/reception

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.

[1] Transmitter only with DC/DC converter from nRF52 data sheet.
[2] Full module power consumption.

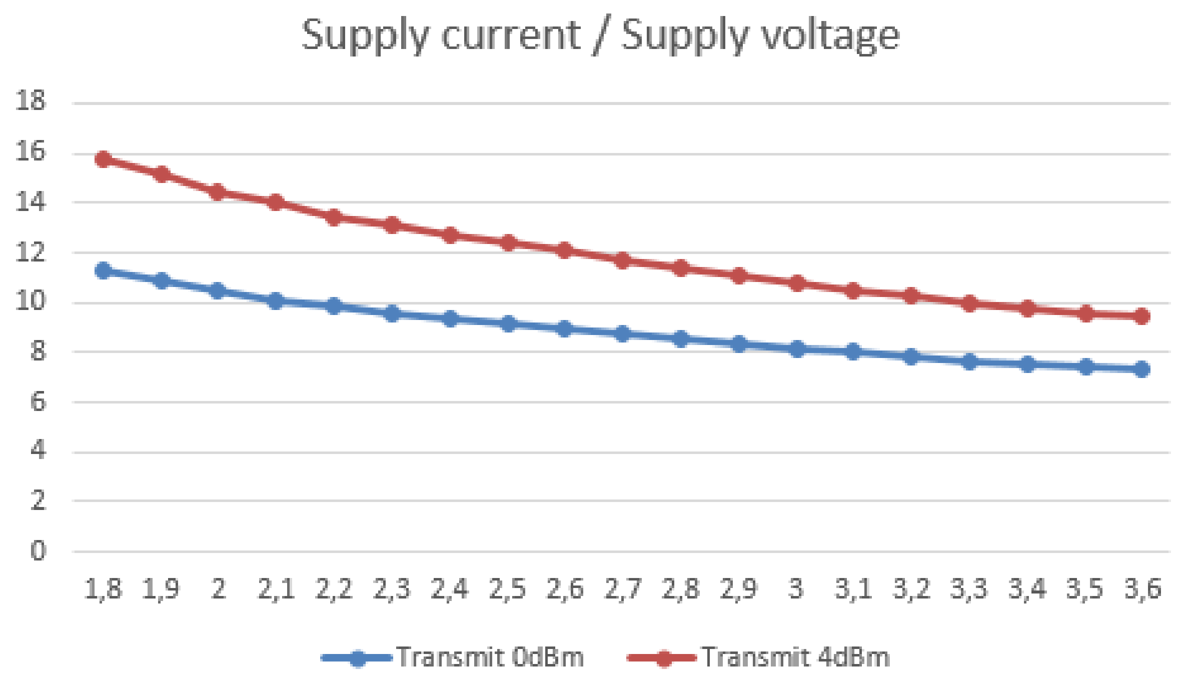
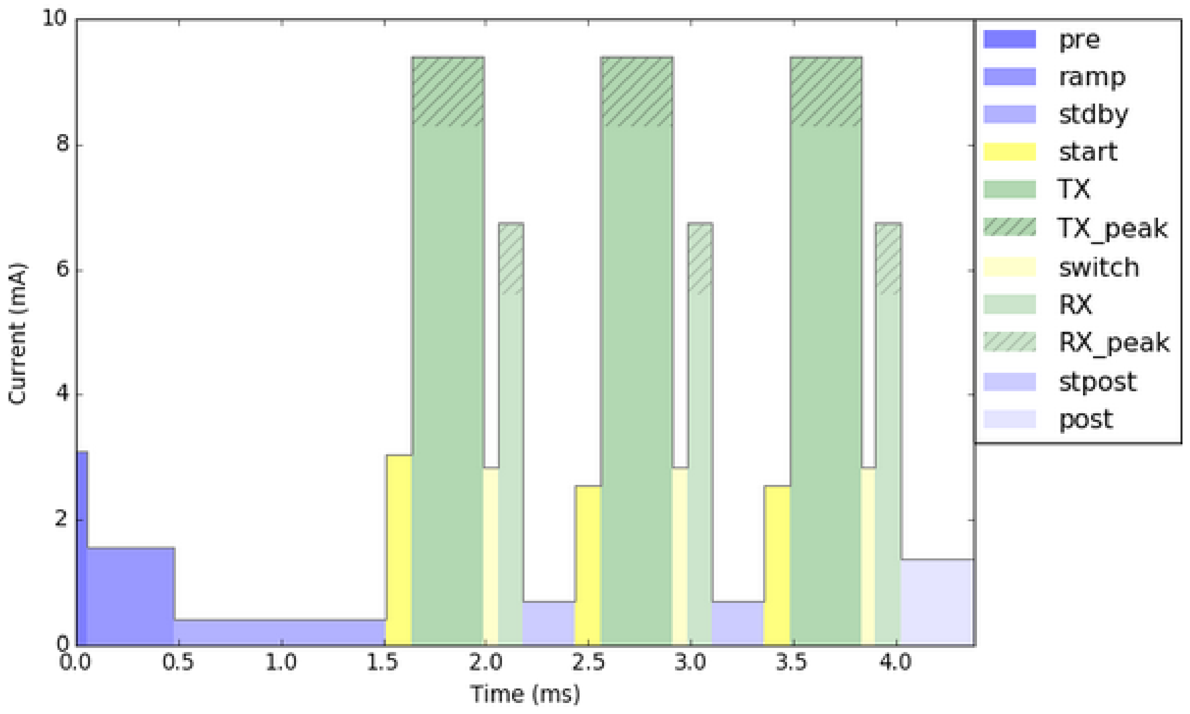


Figure 2: TX Current consumption vs. VCC

2.3.2. Dynamic

Besides the static TX, RX, idle and sleep current the average current is of interest. Here an example for a typical behavior of a peripheral device in advertising mode (see Figure 3 and Figure 4). Currents and state durations are dependent on the configuration of the module. In this state the module transmits the advertising packets on the 3 advertising channels.

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



Stage	Description	Time (ms)	Length (us)	Avg. current (mA)	Peak current (mA)
pre	Pre-processing	0.0	56	3.1	
ramp	Standby + HFXO ramp	0.1	420	1.6	
stdby	Standby	0.5	1034	0.4	
start	Radio startup + CPU	1.5	128	3.1	
TX	Radio TX	1.6	353	8.3	9.4
switch	Radio switch	2.0	67	2.8	
RX	Radio RX	2.1	123	5.6	6.7
stpost	Standby + Post-processing	2.2	256	0.7	
start	Radio startup	2.4	123	2.6	
TX	Radio TX	2.6	353	8.3	9.4
switch	Radio switch	2.9	67	2.8	
RX	Radio RX	3.0	123	5.6	6.7
stpost	Standby + Post-processing	3.1	256	0.7	
start	Radio startup	3.4	123	2.6	
TX	Radio TX	3.5	353	8.3	9.4
switch	Radio switch	3.8	67	2.8	
RX	Radio RX	3.9	123	5.6	6.7
post	Post-processing	4.0	358	1.4	
	System On IDLE	4.4	40.6 ms	1.9 uA	
Total			45.0 ms	325 uA	

Figure 3: Current consumption calculation in advertising mode with 40ms advertising interval, UART disabled

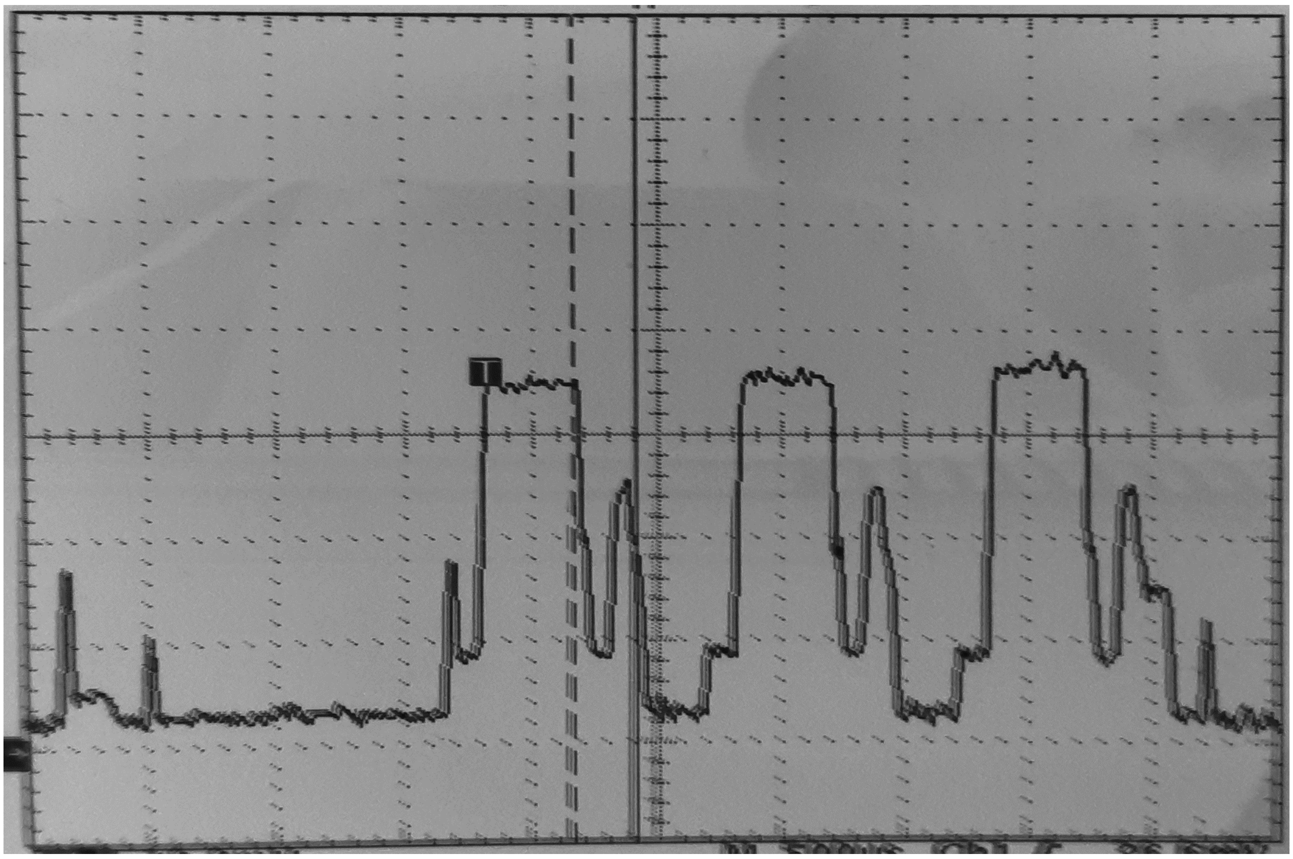


Figure 4: Measured Proteus-I transient current consumption in advertising mode with 40ms advertising interval, excerpt of 5ms

2.4. Radio characteristics

50Ω conducted measurements from nRF52 data sheet

Description	Min.	Typ.	Max.	Unit
Output power	-40	+3	+4	dBm
Input sensitivity ($\leq$ 37 Bytes, BER=1E-3)		-92[1]		dBm
RSSI accuracy valid range (±2dB)	-90		-20	dBm
Enable TX or RX delay		140		µs
Enable TX or RX delay (fast mode)		40		µs
Disable TX delay		6		µs
Disable RX delay		0		µs

Table 5: Radio parameters

Output power `RF_TXPower` = 4	Min.	Typ.	Max.	Unit
Proteus-I external antenna (50Ω conducted)		3	4	dBm
Proteus-I integrated pcb antenna (e.i.r.p.)		-2	0	dBm

Table 6: Output power

[1] nRF52832 Rev.1, with build code CIAA-B00, CSP package, in DC/DC Mode

2.5. Pin characteristics

Measurements from nRF52 data sheet

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

Table 7: Pin characteristics

3. Pinout

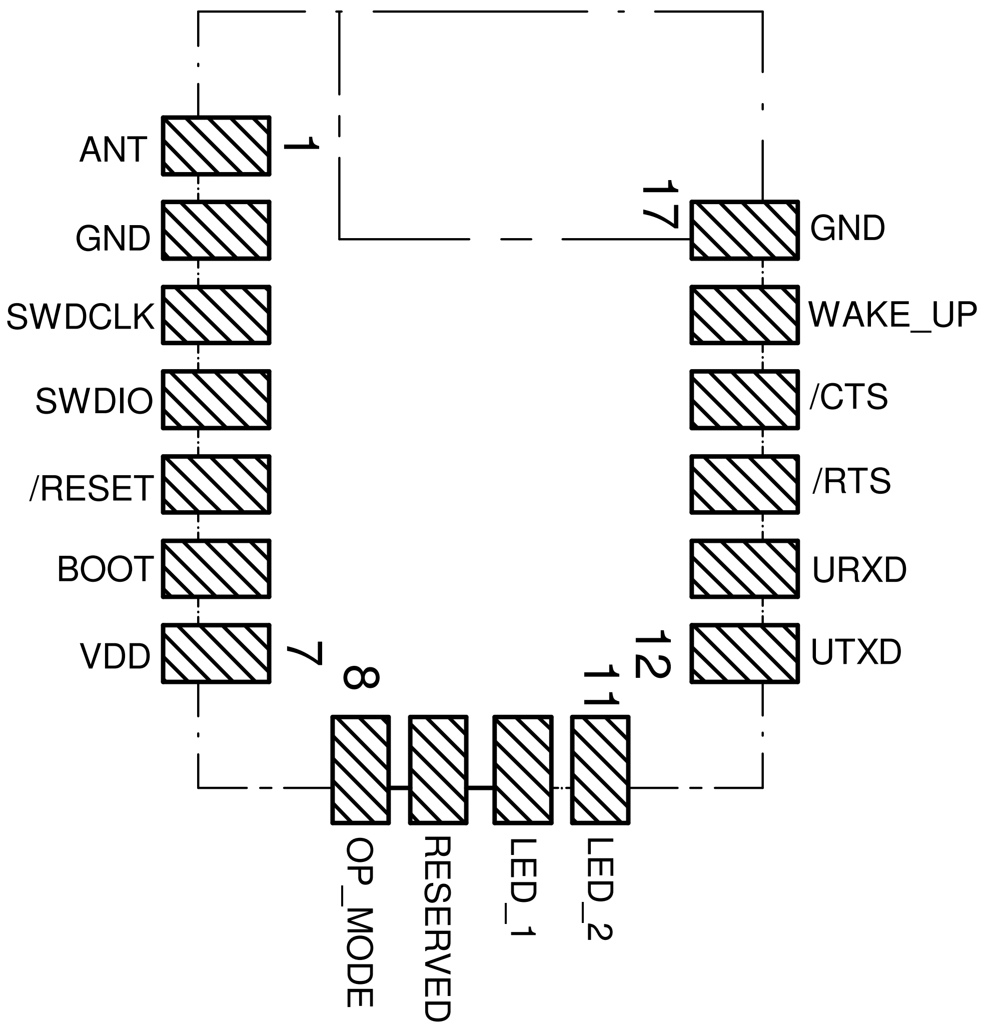


Figure 5: Pinout (top view)

No	µC Pin	Designation	I/O	Description
1		*ANT*	RF	Antenna connection in case of module variant with external antenna. In case of module with integrated antenna, do not connect.
2		*GND*	Supply	Ground
3		*SWDCLK*	Input	Serial wire clock. Uses internal pull down resistor. Do not connect if not needed.
4		*SWDIO*	Input	Serial wire input/output. Uses internal pull up 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	P0.05/AIN3	*BOOT*	Input	Boot pin. A low signal during and short after reset starts the module in OTA bootloader mode. Uses internal pull up resistor[1]. Do not connect if not needed.
7		*VDD*	Supply	Supply voltage
8	P0.10/NFC2[2]	*OP_MODE*	Input	Operation mode pin with internal pull down resistor[1] during start-up. Low level or open: Normal Mode. High level: Peripheral only Mode. Do not connect if not needed.
9	P0.09/NFC1[2]	RESERVED	I/O	Do not connect.
10	P0.00/XL1[3]	*LED_1*	Output	Indicates the module state (active high). Do not connect if not needed.
11	P0.01/XL2[3]	*LED_2*	Output	Indicates the module state (active high). Do not connect if not needed.
12	P0.02/AIN0	*UTXD*	Output	UART(Transmission)
13	P0.03/AIN1	*URXD*	Input	UART (Reception). Uses internal pull up resistor[1].
14	P0.04/AIN2	*/RTS*	Output	Only used if flow control is enabled. Do not connect if not needed.
15	P0.28/AIN4	*/CTS*	Input	Only used if flow control is enabled. Do not connect if not needed.
16	P0.29/AIN5	*WAKE_UP*	Input	Wake-up will allow leaving the system-off mode or re-enabling the UART. Uses internal pull up resistor[1]. Do not connect if not needed.
17		*GND*	Supply	Ground

Table 8: Pinout

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

not implement this function.

4. Quick start

4.1. Minimal pin configuration

In factory state the modules are immediately ready for operation; the following pins are required in the minimal configuration:
VDD, *GND*, *UTXD*, *URXD*, */RESET*

If the flow control is enabled additionally the pins */RTS* and */CTS* shall be connected.

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

Implementing the fail-safe firmware update method using the SWD interface is recommended. Without having the SWD interface available a fail-safe firmware update on a customer PCB cannot be guaranteed.

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

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

4.2. Power up

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

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

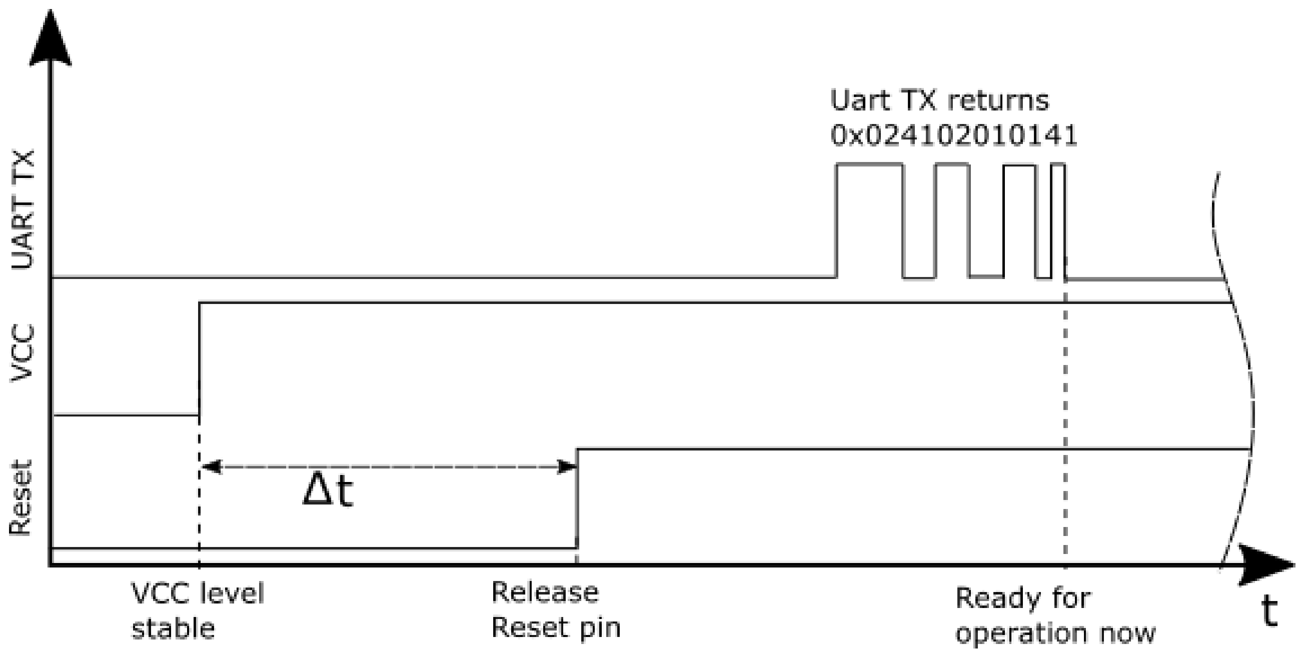


Figure 6: Power up

4.3. Quickstart example

This section describes how to quick start the data transmission between two Proteus-I modules. The goal is to setup a connection between module A and module B, transmit some data and close the connection again.
In this section, all packet data from or to the modules is given in **hexadecimal notation**. For quick testing, a pair of Proteus-I-EV is recommended.
Connect the two devices (modules, EV-boards or USB dongles) to a PC. A terminal program, for example *hterm*, is used to perform the communication via COM ports. The two corresponding COM ports have to be selected and opened with a default configuration of 115200 Baud, 8 data Bits, 1 stop Bit and parity set to none (8n1).

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

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

Connection setup and first data transmission

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

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

2. Request the `FS_BTMAC` of both modules.

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

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

This example is taken from an older firmware. Using newer firmwares with the optional Bluetooth® 4.2 feature "LE Packet Length Extension", the maximum supported payload per packet may be higher than 0x13.

Info	Module A	Module B
⇒ Request `CMD_CONNECT_REQ` with `FS_BTMAC` of module B	02 06 06 00 11 00 00 DA 18 00 D1	
⇐ Response `CMD_CONNECT_CNF`: Request understood, try to connect now	02 46 01 00 00 45	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00	02 86 07 00 00 11 00 00 DA 18 00 50	
⇐ Indication `CMD_CONNECT_IND`: Physical connection established successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00		02 86 07 00 00 55 00 00 DA 18 00 14
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x11 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0x13 (19 Bytes) per packet	02 C6 08 00 00 11 00 00 DA 18 00 13 C3	
⇐ Indication `CMD_CHANNELOPEN_RSP`: Channel opened successfully to module with `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 and maximum payload size of 0x13 (19 Bytes) per packet		02 C6 08 00 00 55 00 00 DA 18 00 13 87

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

The RSSI values will be different in your tests.

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

5. Reply with "EFGH" to module B.

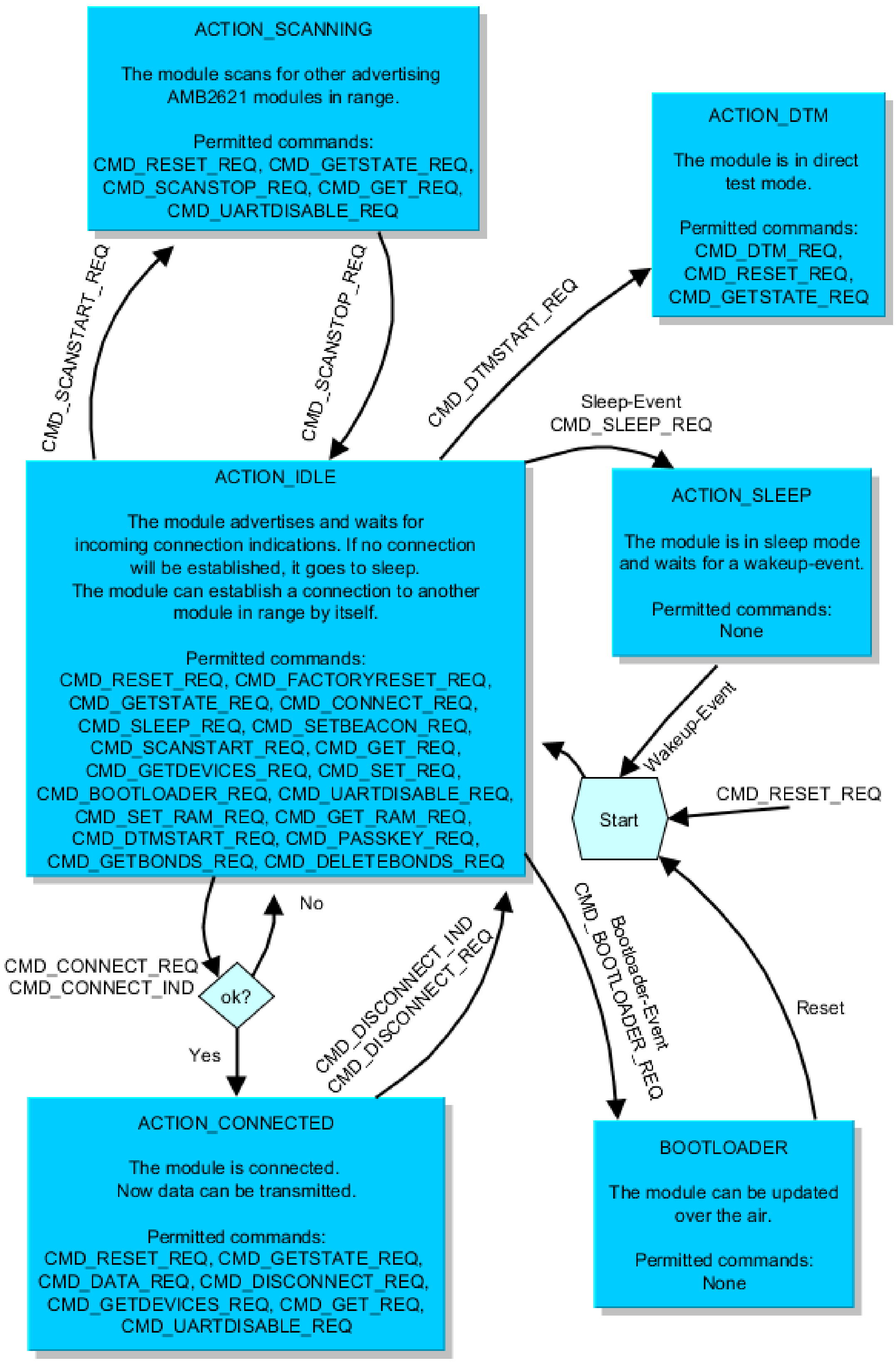
Info	Module A	Module B
⇒ Request `CMD_DATA_REQ`: Send "EFGH" to module B	02 04 04 00 45 46 47 48 0E	
⇐ Response `CMD_DATA_CNF`: Request received, send data now	02 44 01 00 00 47	
⇐ Indication `CMD_DATA_IND`: Received string "EFGH" from `FS_BTMAC` 0x55 0x00 0x00 0xDA 0x18 0x00 with RSSI of 0xC1 (-63dBm)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

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

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

5. Functional description

The Proteus-I module acts as a slave and can be fully controlled by an external host that implements the command interface. The configuration as well as the operation of the module can be managed by predefined commands that are sent as telegrams over the UART interface of the module.
The Proteus-I can operate in different states. Depending on the active state several commands of the command interface (see chapter7) are permitted to modify the state, configure the module or transmit data over the radio interface. An overview of the different states and the corresponding allowed commands can be found in Figure 7.
When the Proteus-I is powered up, it starts in `ACTION_IDLE` state. In this state the module advertises (Bluetooth® LE role "peripheral"), such that other devices in range (Bluetooth® LE role "central" or "observer") can detect it and connect to it. If no connection was setup after `RF_AdvertisingTimeout` seconds, the module goes to `ACTION_SLEEP` state which will stop advertising.
The `ACTION_IDLE` state also allows to switch to `ACTION_SCANNING` state, where the module stops advertising and scans for other advertising modules in range (Bluetooth® LE role "central").
When leaving the `ACTION_SCANNING` state with the corresponding command, the module is in `ACTION_IDLE` state and starts advertising again.
The `ACTION_CONNECTED` state can be entered either by getting a connection request from another module (Bluetooth® LE role "peripheral") or by setting up a connection itself (Bluetooth® LE role "central"). In this case it stops advertising and data can be transmitted and received to/from the connected module. This state remains active as long as the module does not disconnect itself (e.g. due to a timeout), no disconnection request from the connected device is received.
When disconnecting, the module goes to `ACTION_IDLE` state and starts advertising again.



ACTION_SCANNING
The module scans for other advertising AMB2621 modules in range.
Permitted commands: CMD_RESET_REQ, CMD_GETSTATE_REQ, CMD_SCANSTOP_REQ, CMD_GET_REQ, CMD_UARTDISABLE_REQ
ACTION_DTM
The module is in direct test mode.
Permitted commands: CMD_DTM_REQ, CMD_RESET_REQ, CMD_GETSTATE_REQ
CMD_SCANSTART_REQ
CMD_SCANSTOP_REQ
CMD_DTMSTART_REQ
Sleep-Event
CMD_SLEEP_REQ
ACTION_IDLE
The module advertises and waits for incoming connection indications. If no connection will be established, it goes to sleep.
The module can establish a connection to another module in range by itself.
Permitted commands: CMD_RESET_REQ, CMD_FACTORYRESET_REQ, CMD_GETSTATE_REQ, CMD_CONNECT_REQ, CMD_SLEEP_REQ, CMD_SETBEACON_REQ, CMD_SCANSTART_REQ, CMD_GET_REQ, CMD_GETDEVICES_REQ, CMD_SET_REQ, CMD_BOOTLOADER_REQ, CMD_UARTDISABLE_REQ, CMD_SET_RAM_REQ, CMD_GET_RAM_REQ, CMD_DTMSTART_REQ, CMD_PASSKEY_REQ, CMD_GETBONDS_REQ, CMD_DELETEBONDS_REQ
ACTION_SLEEP
The module is in sleep mode and waits for a wakeup-event.
Permitted commands: None
Wakeup-Event
Start
CMD_RESET_REQ
No
CMD_CONNECT_REQ
CMD_CONNECT_IND
ok?
Yes
CMD_DISCONNECT_IND
CMD_DISCONNECT_REQ
CMD_BOOTLOADER_REQ
Bootloader-Event
Reset
ACTION_CONNECTED
The module is connected. Now data can be transmitted.
Permitted commands: CMD_RESET_REQ, CMD_GETSTATE_REQ, CMD_DATA_REQ, CMD_DISCONNECT_REQ, CMD_GETDEVICES_REQ, CMD_GET_REQ, CMD_UARTDISABLE_REQ
BOOTLOADER
The module can be updated over the air.
Permitted commands: None

Figure 7: State overview

5.1. State indication using the LED pins

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

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

Table 9: LED behavior of the Proteus-I

5.2. Sleep mode

Especially for battery-powered devices the `ACTION_SLEEP` mode (system-off mode) supports very low power consumption (<1µA). It can be entered by sending the command `CMD_SLEEP_REQ` to the module. If allowed (due to the current operating state) the module will then send a `CMD_SLEEP_CNF` and then enter the `ACTION_SLEEP` mode.
In `ACTION_SLEEP` mode the UART is disabled, so the module will not receive or transmit any data. To prevent leakage current, the host shall not pull the *UART_RX* to LOW level (as the module has an internal pull-up resistor enabled on this pin).
To leave the `ACTION_SLEEP` mode and enter `ACTION_IDLE` state again, the module has to be woken up by applying a low signal to the *WAKE_UP* pin for at least 5ms before releasing the signal back to high. The module then restarts completely, so that all volatile settings are set to default. A `CMD_GETSTATE_CNF` will be send when the module is ready for operation.

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

5.3. Identification of a Proteus-I device on the radio

The Proteus-I can be identified on the radio interface by its `FS_BTMAC`. This `FS_BTMAC` is a Bluetooth®-conform MAC address, which is part of the data package sent during advertising in `ACTION_IDLE` mode. A `FS_BTMAC` has the size of 6 Bytes.
In `ACTION_SCANNING` state a module listens to the data packets of all advertising modules in range and stores their `FS_BTMAC` to an internal data base. With help of a `FS_BTMAC` a connection to the corresponding device can then be established using the `CMD_CONNECT_REQ` command.
To simplify the identification of Proteus-I devices on the RF-interface a short user-defined name (see `RF_DeviceName`) can be given to the module, which is also part of the advertising packet.

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

5.4. Connection based data transmission, with or without security

In the Bluetooth® LE standard the transmission of data typically is connection based. A connection between two devices can be secured (with or without key exchange) or unsecured (default setting). In any case, each data packet transmitted is acknowledged on the link layer, such that it is resent as long as a packet is lost. The following lines describe how to run the connection setup and data transmission using the Proteus-I.
If module A is supposed to setup a connection with module B, module A can use the command `CMD_CONNECT_REQ` including the `FS_BTMAC` of module B. If the `FS_BTMAC` of module B is unknown, a scan can be run before by module A to discover all available modules in range.
After sending the command `CMD_CONNECT_REQ`, the module answers with a `CMD_CONNECT_CNF` to signal that the request has been understood and the module now tries to establish the connection.
If module B cannot be found on the air within a timeout, module A outputs a `CMD_CONNECT_IND` with "failed" as status. Otherwise, as soon as the physical connection has been set up successfully, module A and B print a `CMD_CONNECT_IND` with the status of the successful connection and *LED_1* turns on.
Next some security and authentication messages will follow, like `CMD_SECURITY_IND`, if security is enabled.
After the physical connection has been setup successfully the modules exchange their services. As soon as this has finished successfully, a `CMD_CHANNELOPEN_RSP` is given out to the UART indicating that the connection is ready for data transmission. Furthermore, *LED_2* turns on.
Now data can be transmitted in both directions using the command `CMD_DATA_REQ`. It is confirmed by a `CMD_DATA_CNF` (data will be processed) and a `CMD_TXCOMPLETE_RSP` (data transmitted successfully).
Each time data has been received a `CMD_DATA_IND` will be output containing the transmitted data.
As soon as one module closes the connection using a `CMD_DISCONNECT_REQ`, both modules will inform their host by a `CMD_DISCONNECT_IND` message that the connection is no longer

open. If one module is no longer within range, the `CMD_DISCONNECT_IND` message is triggered by a timeout.
For an example on setting up an unsecured connection, see chapter 4.3. See also the Proteus-I application note "advanced user guide" to get detailed information about the connection setup with foreign devices.

5.4.1. Further information for a secure connection setup

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

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

To get further information about the secured connection setup, when using a foreign device (i.e. mobile phone with a custom APP), please refer to the Application Note "advanced user guide" (Proteus-I: ANR002, Proteus-II: ANR005).

5.4.1.1. Just works mode

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

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

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

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

2. Request the `FS_BTMAC` of both modules.

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

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

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

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

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

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

The RSSI values will be different in your tests.

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

6. Reply with "EFGH" to module B.

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

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

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

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

5.4.1.2. StaticPasskey mode

In case of the "StaticPasskey" mode, a pass key has to be entered at the central side that has to match the pass key of the peripheral. Here the Proteus-I uses a static pass key in the peripheral role that is stored in the parameter `RF_StaticPasskey`. When using this method, the central device requests its host to enter the correct pass key (see CMD_PASSKEY_IND). In this case the pass key of the peripheral has to be entered on central side using the CMD_PASSKEY_REQ command. If the entered pass key is correct, the channel will be opened for data transmission. Otherwise, the connection will be rejected.

Example: Secured connection with security method "StaticPasskey"

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

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

2. Request the FS_BTMAC of both modules.

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

3. Configure the parameter RF_SecFlags to use "StaticPasskey" pairing method for Bluetooth® security.

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

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

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

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

The RSSI values will be different in your tests.

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

6. Reply with "EFGH" to module B.

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

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

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

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

5.4.1.3. Bonding

The `SECFLAGS_BONDING_ENABLE` flag in the `RF_SecFlags` user setting allows enabling the bonding feature. This feature stores the keys that are exchanged during the pairing phase in a connection setup. With this, subsequent connections to bonded devices can be established without renegotiation. Bonding data of up to 32 devices will be stored in the flash. The commands CMD_GETBONDS_REQ and CMD_DELETEBONDS_REQ allow to display and remove certain or all entries of the list of bonded devices.

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

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

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

2. Request the FS_BTMAC of both modules.

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

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

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

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

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

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

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

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

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

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

5.5. Unidirectional connectionless data transmission using Beacons

Besides the connection-based type of data transmission described in the previous section there exists a second method that uses so called Beacons. In this case, a limited amount of user data can be placed in the Bluetooth® LE scan response packet, which is broadcasted frequently without acknowledgement and without security.
If a Proteus-I is supposed to broadcast some data, the command `CMD_SETBEACON_REQ` can be used to place user data in the scan response packet.
If a second Proteus-I, which has its Beacon-function (see `RF_BeaconFlags`) enabled, is in the operating state `ACTION_SCANNING`, then the scan response packet is requested as soon as an advertising packet from the beacon module has been detected. Filtering the beacon messages can be enabled or disabled using `RF_BeaconFlags`.
After the reception of the scan response packet the included user data is interpreted and given out to the UART using a `CMD_BEACON_IND` message.
To set the module into `ACTION_SCANNING` mode the command `CMD_SCANSTART_REQ` has to be used. Enable the Beacon-function before by setting the corresponding Bit in the `RF_BeaconFlags` parameter.

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

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

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

5.6. Energy-efficient distance estimation solutions

The Proteus-I advertising packet contains the TX power value of the transmitting device. This value in combination with the RSSI value of the received advertising packet can be used to estimate the distance between the modules. Using a suitable triangulation algorithm and multiple receivers or transmitters, a position can be approximated.
The advertising packets can be received by performing a passive scan that will not request the scan response. Thus only one frame, instead of three frames, is transmitted per advertising interval.
Besides the `FS_BTMAC` of the sending module, the RSSI value and the TX power is output in format of a `CMD_RSSI_IND` message via UART when an advertising packet of another Proteus-I has been received.
To enable this function, the corresponding Bit in the `RF_BeaconFlags` has to be set.

5.7. Configure the module for low power consumption

Depending on the application environment of the Proteus-I, 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 *WAKE_UP* pin of the module was used.
- `RF_TXPower`: This setting can be used to configure the output power of the module. Reducing the output power saves energy.
- `RF_ScanTiming` and `RF_ScanFactor`: These settings define the timing behavior of the module, when advertising or scanning. The less often the module sends advertising packets or scans, the less current is consumed.
- `RF_ConnectionTiming`: This setting defines the timing behavior of the module during connection setup and an established connection. The less often the connected modules communicate with each other, the less current is consumed.
- The on-board nRF52 SoC is running in debug mode. This will not occur if the pins are connected as described in this manual.

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

5.8. Start the direct test mode (DTM)

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

- transmission power and receiver sensitivity
- frequency offset and drift
- modulation characteristics
- packet error rate
- inter modulation performance

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

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

- First, restart the modules in DTM mode.

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

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

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

- Start the reception test.

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

- Stop both tests again.

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

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

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 of the UART can be configured by means of the UserSetting `UART_BaudrateIndex`. The data format is fixed to 8n1. The flow control can be enabled by means of the UserSetting `UART_Flags`.
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.

7. The command interface

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

- Requests: The host requests the module to trigger any action, e.g. in case of the request `CMD_RESET_REQ` the host asks the module to perform a reset.
- Confirmations: On each request, the module answers with a confirmation message to give a feedback on the requested operation status. In case of a `CMD_RESET_REQ`, the module answers with a `CMD_RESET_CNF` to tell the host whether the reset will be performed or not.
- Indications and Responses: The module indicates spontaneously when a special event has occurred. The `CMD_CONNECT_IND` indicates for example that a connection has been established.

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

Start signal: 0x02 (1 Byte)

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

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

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

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

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*.

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

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

7.1. Scan for other modules in range

7.1.1. CMD_SCANSTART_REQ

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

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

Response (`CMD_SCANSTART_CNF`):

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

Status:

0x00: Request received, will start scan now

0x01: Operation failed

0xFF: Operation not permitted

7.1.2. CMD_SCANSTOP_REQ

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

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

Response (`CMD_SCANSTOP_CNF`):

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

Status:

0x00: Request received, will stop scan now

0x01: Operation failed

0xFF: Operation not permitted

7.1.3. CMD_GETDEVICES_REQ

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

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

Response (CMD_GETDEVICES_CNF):

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

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

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

Status:

0x00: Request received

0x01: Operation failed

0xFF: Operation not permitted

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

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

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

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

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

7.1.3.1. Example 1

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

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

Response:

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

During the last scan two devices have been detected:

- Device 1 with FS_BTMAC 0x11 0x00 0x00 0xDA 0x18 0x00, RSSI value of 0xE2 (-30 dBm), TXPower of 0x04 (=+4 dBm) and device name of length 5 with the value of 0x4D4F442031 ("MOD 1").
- Device 2 with FS_BTMAC 0x55 0x00 0x00 0xDA 0x18 0x00 and RSSI value of 0xE5 (-27 dBm), TXPower of 0x00 (0 dBm) and device name 0x4D4F442032 ("MOD 2") of length 5.

7.1.4. CMD_RSSI_IND

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

complement notation.
The accuracy is ±2dB when inside the RSSI range of -90 to -20 dBm.
The value of the parameter TX power is read from the content of the received advertise packet.
Format:

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

7.2. Setup connections

7.2.1. CMD_CONNECT_REQ

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

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

Response (`CMD_CONNECT_CNF`):

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

Status:

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

0x01: Operation failed

0xFF: Operation not permitted

7.2.2. CMD_CONNECT_IND

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

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

Status:

0x00: Physical connection established successfully

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

7.2.3. CMD_SECURITY_IND

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

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

Status:

0x00: Encrypted link to previously bonded device established

0x01: Bonding successful, encrypted link established

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

7.2.4. CMD_CHANNELOPEN_RSP

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

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

Status:

0x00: Success

7.2.5. CMD_DISCONNECT_REQ

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

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

Response (`CMD_DISCONNECT_CNF`):

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

Status:

0x00: Request received, try to disconnect

0x01: Operation failed

0xFF: Operation not permitted

7.2.6. CMD_DISCONNECT_IND

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

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

Reason:

0x08: Connection timeout

0x13: User terminated connection

0x16: Host terminated connection

0x3B: Connection interval unacceptable

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

0x3E: Connection setup failed

7.2.7. CMD_PASSKEY_REQ

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

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

Response (`CMD_PASSKEY_CNF`):

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

Status:

0x00: Pass key accepted and pass key request answered

0x01: Operation failed, due to invalid pass key

0xFF: Operation not permitted

7.2.8. CMD_PASSKEY_IND

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

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

Status:

0x00: Success

7.2.9. CMD_GETBONDS_REQ

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

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

Response (`CMD_GETBONDS_CNF`):

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

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

Bond_ID	BTMAC
2 Bytes	6 Bytes

Status:

0x00: Request successfully processed

0x01: Operation failed

0xFF: Operation not permitted

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

7.2.9.1. Example 1

Request for the bonding data of the devices in database.

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

Response:

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

Two devices have been bonded before:

- Device 1 (Bond_ID 0x0000) with `FS_BTMAC` 0x82 0x5C 0xA7 0xE2 0x87 0xD0
- Device 2 (Bond_ID 0x0001) with `FS_BTMAC` 0x01 0x00 0x00 0xDA 0x18 0x00

7.2.10. CMD_DELETEBONDS_REQ

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

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

Response (`CMD_DELETEBONDS_CNF`):

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

Status:

0x00: Request successfully processed

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

0xFF: Operation not permitted

7.2.10.1. Example 1

Request to remove all bonding data.

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

Response:

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

Successfully removed all bonding information.

7.2.10.2. Example 2

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

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

Response:

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

Successfully removed the bonding information.

7.3. Transmit and receive data

7.3.1. CMD_DATA_REQ

This command provides the simple data transfer between two connected modules. Transmission takes place to the previously connected device(s). This command is suitable for transmission for a point-to-point connection. The number of payload data Bytes is negotiated during the connection phase. It can be maximal 243 Bytes, but at least 19 Bytes.
When the data is processed by the module a `CMD_DATA_CNF` is output by the UART. Additionally a `CMD_TXCOMPLETE_RSP` will follow as soon as the data has been sent.
The receiving Proteus-I 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

7.3.2. CMD_TXCOMPLETE_RSP

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

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

Status:

0x00: Data transmitted successfully

0x01: Data transmission failed

7.3.3. CMD_DATA_IND

This telegram indicates the reception of data sent by the previously connected device. This indication message is the result of a data request (`CMD_DATA_REQ`) sent to the associated device within a connection.
The `CMD_DATA_IND` returns the `FS_BTMAC` of the sending device, the RSSI value of the received data packet and the data received via the RF-interface, which can be found in the payload. The RSSI value is printed in two's complement notation.
Format:

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

7.3.4. CMD_SETBEACON_REQ

This command is used to place user data in the scan response packet. The data is broadcasted frequently without acknowledgement and security. No connection is needed for this mode of operation.
It can be received by any scanning Proteus-I with Beacon-function enabled (see `RF_BeaconFlags`).
The receiving module will output a `CMD_BEACON_IND` indication message containing the transmitted data. See chapter `5.5` for more information.
Choosing 0x00 as Length and leaving the Payload field empty will remove the data from the scan response packet. The number of payload data Bytes is limited to 19.
Format:

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

Response (`CMD_SETBEACON_CNF`):

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

Status:

0x00: Request received, will place data now

0x01: Operation failed

0xFF: Operation not permitted

7.3.5. CMD_BEACON_IND

This telegram indicates the reception of data Bytes that have been transmitted in a beacon-packet. This data can only be received, when the module is in `ACTION_SCANNING` mode and the beacon-function is enabled (see `RF_BeaconFlags`).
The data received via the RF-interface can be found in the payload of the `CMD_BEACON_IND` telegram. Besides this, the `FS_BTMAC` of the sending device and the RSSI value of the data packet are output as well. The RSSI value is output in two's complement notation.
Format:

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

7.4. Configuring the module and modifying the device settings

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

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

7.4.1. CMD_SET_REQ

This command enables direct manipulation of the parameters in the module's settings in flash. The respective parameters are accessed by means of the corresponding settings index, which can be found in Table 17.
Parameters of 2 or more Bytes have to be transferred with the LSB first unless noted differently in the corresponding description.

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

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

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

0xFF: Operation not permitted

7.4.1.1. Example 1

Setting the advertising time `RF_AdvertisingTimeout` to 180 seconds.

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

Response:

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

Setting was set successfully.

7.4.1.2. Example 2

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

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

Response:

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

Setting was set successfully.

7.4.2. CMD_GET_REQ

This command can be used to query individual setting parameters in flash. The respective parameters are accessed by means of the corresponding settings index, which can be found in Table 17.
Parameters of 2 or more Bytes have to be transferred with the LSB first unless noted differently in the corresponding description.
Read access to the memory area outside the setting is blocked.
Format:

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

Response (`CMD_GET_CNF`):

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

Status:

0x00: Request received, read out of setting successful

0x01: Operation failed

0xFF: Operation not permitted

7.4.2.1. Example 1

Request the current static pass key `RF_StaticPasskey`.

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

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

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

Setting was read successfully.

7.5. Manage the device state

7.5.1. CMD_GETSTATE_REQ

This command returns the current state of the module.

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

Format:

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

Response (`CMD_GETSTATE_CNF`):

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

Module role:

0x00: No role

0x01: Peripheral

0x02: Central

0x10: Direct test mode (DTM)

Other: Reserved

Module action:

0x00: No action

0x01: Idle (advertising)

0x02: Scanning

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

0x04: Sleep (system-off mode)

0x05: Direct test mode

7.5.1.1. Example 1

Get the current state of the module.

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

Response:

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

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

7.5.2. CMD_RESET_REQ

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

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

Response (`CMD_RESET_CNF`):

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

Status:

0x00: Request received, will perform reset now

0x01: Operation failed

0xFF: Operation not permitted

7.5.3. CMD_SLEEP_REQ

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

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

Response (`CMD_SLEEP_CNF`):

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

Status:

0x00: Request received, will go to sleep now

0x01: Operation failed

0xFF: Operation not permitted

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

7.5.4. CMD_SLEEP_IND

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

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

Status:

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

7.5.5. CMD_FACTORYRESET_REQ

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

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

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

This command also removes all bonding data.

7.5.6. CMD_UARTDISABLE_REQ

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

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

Response (`CMD_UARTDISABLE_CNF`):

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

Status:

0x00: Request received, will disable UART now

0x01: Operation failed

0xFF: Operation not permitted

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

Disabling the UART peripheral of the module results in a reduction of current consumption of about 1.15mA.

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

7.5.7. CMD_UARTENABLE_IND

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

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

Status:

0x00: UART has been re-enabled successfully

7.5.8. CMD_BOOTLOADER_REQ

This command resets the module and starts the OTA bootloader.

Please refer to chapter 13 on how to use the bootloader for a firmware update.

Please note that you can only exit the bootloader mode by performing a hardware reset using the respective pin.

The bootloader mode will also be enabled if the firmware image is marked "invalid" or if the *BOOT* pin logic level (set by the host) is set to start the bootloader during start-up of the module.

Format:

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

Response (`CMD_BOOTLOADER_CNF`):

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

Status:

0x00: Request received, will start bootloader now

0x01: Operation failed

0xFF: Operation not permitted

7.6. Run the Bluetooth® test modes

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

7.6.1. CMD_DTMSTART_REQ

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

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

Response (`CMD_DTMSTART_CNF`):

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

Status:

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

0x01: Operation failed

0xFF: Operation not permitted

7.6.2. CMD_DTM_REQ

This command starts and stops various test modes. To be able to run these test modes, the DTM has to be enabled first using the `CMD_DTMSTART_REQ`. After a test has been started, it has to be stopped first before a next test can be run.
The default TX power value is 0 dBm, the allowed range is from -40 up to +4 dBm in steps of 4dB (see chapter 8.16).
The valid range for Channel (or Vendor option in case of Vendor Command = Carrier Test) 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 reset (note: this command does not perform a module reset.

0x01: Start RX test

0x02: Start TX test

0x03: Stop last test

Payload:

0x00: Bit pattern PRBS9

0x01: Bit pattern 0x0F

0x02: Bit pattern 0x55

0x03: Vendor specific

Payload $\neq$ Vendor specific (0x00, 0x01 or 0x02)	Payload = Vendor specific (0x03)
Length / Vendor Command: Length of the packet to send	Length / Vendor Command: 0x00: Carrier test 0x02: Set transmission power
Channel: Frequency = (2402 + Channel * 2) MHz to be used for RX/TX	Vendor option: (dependent on used "Vendor command") Frequency = (2402 + [Vendor option] * 2) MHz or [Vendor option] := TXPower (in two's complement notation) in steps of 4dB

Response (`CMD_DTM_CNF`):

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

Status:

0x00: Request received

0x01: Operation failed

0x03: Busy

0xFF: Operation not permitted

Result:

0x0000: Test success

0x0001: Test failed

0x8000 + n: Received n packets during RX test

See also the example in chapter 5.8.

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

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

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

Response:

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

Test started successfully. Now stop the test again.

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

Response:

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

Test stopped successfully and received 0 packets.

7.6.2.2. Example: Receiver, channel 0

Start the reception test on channel 0 (2402MHz) with Bit pattern 0x0F:

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	0x1A

Response:

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

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

7.6.2.3. Example: Transmission, carrier test, channel 0

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

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

Response:

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

See the previous example to stop the test again.

7.6.2.4. Example: Set TX power to -4 dBm

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

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

Response:

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

See the previous example to stop the test again.

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	7.5.2
0x02	0x01	CMD_GETSTATE_REQ	Request the current module state	7.5.1
0x02	0x02	CMD_SLEEP_REQ	Go to sleep	7.5.3
0x02	0x04	CMD_DATA_REQ	Send data to the connected device	7.3.1
0x02	0x06	CMD_CONNECT_REQ	Setup a connection with another device	7.2.1
0x02	0x07	CMD_DISCONNECT_REQ	Close the connection	7.2.5
0x02	0x09	CMD_SCANSTART_REQ	Start scan	7.1.1
0x02	0x0A	CMD_SCANSTOP_REQ	Stop scan	7.1.2
0x02	0x0B	CMD_GETDEVICES_REQ	Request the scanned/detected devices	7.1.3
0x02	0x0C	CMD_SETBEACON_REQ	Place data in scan response packet	7.3.4
0x02	0x0D	CMD_PASSKEY_REQ	Respond to a pass key request	7.2.7
0x02	0x0E	CMD_DELETEBONDS_REQ	Delete bonding information	7.2.10
0x02	0x0F	CMD_GETBONDS_REQ	Read the MACs of bonded devices	7.2.9
0x02	0x10	CMD_GET_REQ	Read the module settings in flash	7.4.2
0x02	0x11	CMD_SET_REQ	Modify the module settings in flash	7.4.1
0x02	0x1B	CMD_UARTDISABLE_REQ	Disable the UART	7.5.6
0x02	0x1C	CMD_FACTORYRESET_REQ	Perform a factory reset	7.5.5
0x02	0x1D	CMD_DTMSTART_REQ	Enable the direct test mode	7.6.1
0x02	0x1E	CMD_DTM_REQ	Start/stop a test of the direct test mode	7.6.2
0x02	0x1F	CMD_BOOTLOADER_REQ	Switch to the bootloader	7.5.8

Table 10: Message overview: Requests

Start signal	CMD	Message name	Short description	Chapter
0x02	0x40	CMD_RESET_CNF	Reset request received	7.5.2
0x02	0x41	CMD_GETSTATE_CNF	Return the current module state	7.5.1
0x02	0x42	CMD_SLEEP_CNF	Sleep request received	7.5.3
0x02	0x44	CMD_DATA_CNF	Data transmission request received	7.3.1
0x02	0x46	CMD_CONNECT_CNF	Connection setup request received	7.2.1
0x02	0x47	CMD_DISCONNECT_CNF	Disconnection request received	7.2.5
0x02	0x49	CMD_SCANSTART_CNF	Scan started	7.1.1
0x02	0x4A	CMD_SCANSTOP_CNF	Scan stopped	7.1.2
0x02	0x4B	CMD_GETDEVICES_CNF	Output the scanned/detected devices	7.1.3
0x02	0x4C	CMD_SETBEACON_CNF	Data is placed in scan response packet	7.3.4
0x02	0x4D	CMD_PASSKEY_CNF	Received the pass key	7.2.7
0x02	0x4E	CMD_DELETEBONDS_CNF	Deleted bonding information	7.2.10
0x02	0x4F	CMD_GETBONDS_CNF	Return the MAC of all bonded devices	7.2.9
0x02	0x50	CMD_GET_CNF	Return the requested module flash settings	7.4.2
0x02	0x51	CMD_SET_CNF	Module flash settings have been modified	7.4.1
0x02	0x5B	CMD_UARTDISABLE_CNF	Disable UART request received	7.5.6
0x02	0x5C	CMD_FACTORYRESET_CNF	Factory reset request received	7.5.5
0x02	0x5D	CMD_DTMSTART_CNF	Enable the direct test mode now	7.6.1
0x02	0x5E	CMD_DTM_CNF	Test of direct test mode started/stopped	7.6.2
0x02	0x5F	CMD_BOOTLOADER_CNF	Will switch to bootloader now	7.5.8

Table 11: Message overview: Confirmations

Start signal	CMD	Message name	Short description	Chapter
0x02	0x82	`CMD_SLEEP_IND`	State will be changed to `ACTION_SLEEP`	7.5.4
0x02	0x84	`CMD_DATA_IND`	Data has been received	7.3.3
0x02	0x86	`CMD_CONNECT_IND`	Connection established	7.2.2
0x02	0x87	`CMD_DISCONNECT_IND`	Disconnected	7.2.6
0x02	0x88	`CMD_SECURITY_IND`	Secured connection established	7.2.3
0x02	0x8B	`CMD_RSSI_IND`	Advertising package detected	7.1.4
0x02	0x8C	`CMD_BEACON_IND`	Received Beacon data	7.3.5
0x02	0x8D	`CMD_PASSKEY_IND`	Received a pass key request	7.2.8
0x02	0x9B	`CMD_UARTENABLE_IND`	UART was re-enabled	7.5.7
0x02	0xA2	`CMD_ERROR_IND`	Entered error state	7.7.1
0x02	0xC4	`CMD_TXCOMPLETE_RSP`	Data has been sent	7.3.2
0x02	0xC6	`CMD_CHANNELOPEN_RSP`	Channel open, data transmission possible	7.2.4

Table 12: 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.
These settings cannot be accessed (read, write) from the Peripheral only mode introduced in a follow-up chapter.

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

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

8.1. FS_DeviceInfo: Read the chip type and OS version

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

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

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

OS version:

0x0081 : Softdevice S132 2.0.0

0x0088 : Softdevice S132 2.0.1

0x008C : Softdevice S132 3.0.0

0x0091 : Softdevice S132 3.1.0

Build code:

- ASCII coded (see the following table 13)

Package variant:

0x2000: QFN

0x2002: CI

Chip ID:

0x00052832 : nRF52832

Packet variant	Build code	Package	Flash size	RAM size
QF	AAB0	QFN48	512 kB	64 kB
QF	ABB0	QFN48	256 kB	32 kB
CI	AABA, AAB0, AAB1, AAE0, AAE1	WLCSP	512 kB	64 kB

Table 13: nRF52832 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 = 0x0088 (Softdevice S132 2.0.1)
Build code = 0x41414241 (AABA)
Package variant = 0x2002 (CI)
Chip ID = 0x00052832
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	0x88 0x00 0x41 0x42 0x41 0x41 0x02 0x20 0x32 0x28 0x05 0x00	0xE9

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	6

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

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x07 0x00	0x00	0x55 0x93 0x19 0x6E 0x5B 0x87	0x38

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	"A2621"	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 5 Bytes. Thus longer device names will be shortened to 5 Bytes and declared as "Shortened Local Name" in the advertising packet. The full device name is included in the GAP.

8.6.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.6.2. Example 2

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

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

Response `CMD_GET_CNF`: Successfully read out the module as 0x41 0x32 0x36 0x32 0x31 = "A2621".

Start signal	Command | 0x40	Length	Status	Parameter	CS
0x02	0x50	0x06 0x00	0x00	0x41 0x32 0x36 0x32 0x31	0x12

8.7. RF_StaticPasskey: Modify the static passkey

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

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

8.7.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.7.2. Example 2

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

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

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

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

8.8. RF_SecFlags: Modify the security settings

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

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

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

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

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

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

Table 14: 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_SecFlagsPerOnly: Modify the security settings (Peripheral only mode)

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

Please refer to the setting `RF_SecFlags` for more details.

8.9.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.9.2. Example 2

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

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

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

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

8.10. RF_ScanFlags: Modify the scan behavior

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

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

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

Table 15: Scan configuration flags

8.10.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.10.2. Example 2

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

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

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

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

8.11. RF_BeaconFlags: Interprete the advertising data

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

This 8-Bit field enables/disables the reception of Beacons. To use multiple settings, add the Bit numbers and choose the result as value for `RF_BeaconFlags`.

Bit no.	Description	
1 : 0	Enable/disable the reception of Beacons. To avoid too much traffic on the UART, we recommend to use the filtered version.	
	0x0	Reception of Beacons disabled.
	0x1	Receive all Beacons from Proteus-I devices in range. Each received packet is interpreted and output by the UART. In this case, active scanning is performed which increases the current consumption. To decrease the work load of the receiving module, use a sufficiently high UART baud rate at the receiving device and slow advertising intervals at the sending devices.
	0x3	Same as '0x1' plus additional filter. This filter discards redundant packets that contain the same content.
	others	Reserved.
2	If this Bit is set, a `CMD_RSSI_IND` message is output each time when an advertising packet with AMBER SPP-like UUID is received. This feature can be used to realize a position sensing application, since the `CMD_RSSI_IND` contains the current TX power level and the current RSSI value besides the `FS_BTMAC` of the sending device. To decrease the work load of the receiving module, please use a sufficiently high UART baud rate at the receiving device and slow advertising intervals at the sending devices.	
15 : 3	Reserved.	

Table 16: Beacon configuration flags

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

To avoid too much traffic the usage of slow advertising intervals is recommended.

8.11.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.11.2. Example 2

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

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

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

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

8.12. RF_AdvertisingTimeout: Modify the advertising timeout

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

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

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

8.12.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.12.2. Example 2

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

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

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

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

8.13. RF_ScanFactor: Modify the scan factor

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

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

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

8.13.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.13.2. Example 2

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

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

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

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

8.14. RF_ScanTiming: Modify the scan timing

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
9	`RF_ScanTiming`	0 - 5	1	read/write	1

The `RF_ScanTiming` enables the possibility to configure the timing behavior of the module's RF interface during advertising and scanning state. Using this parameter several predefined configurations can be chosen, which include timing parameters, such as the frequency of advertising packets and the length of a scan window.
The choice of the `RF_ScanTiming` primarily affects the latency of device detection on air as well as the current consumption. The lower the `RF_ScanTiming`, the faster the modules can find each other for communication, but also the more power will be consumed.

`RF_ScanTiming`	0	1	2	3[1]	4[1]	5[1]
Advertising interval [ms]	20	40	250	1000	5000	10240
Scan window [ms]	25	50	312	1250	6250	10240
Scan interval [ms]	Defined by the `RF_ScanFactor`.					
Connection setup timeout [s]	1	2	2	5	20	35
Current consumption	High	...				Low

Further information:

- In `ACTION_SCANNING` mode, the scan interval defines the time after which the module switches channel to detect other Bluetooth® LE devices in range. See also `RF_ScanFactor`.
- In `ACTION_SCANNING` mode, the scan window defines the section of the scan interval, where the module is scanning. During the remaining time, the module enters a suspend mode (system-on mode). See also `RF_ScanFactor`.
- In `ACTION_IDLE` mode, the advertising interval defines the time after which the module periodically sends its advertising packet. In between, the module enters a suspend mode (system-on mode).
- The connection setup timeout defines the time after which a connection request has to be answered by the peripheral.

Please ensure that all members of a network support the same advertising and scan timing parameters.

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

To ensure that the module is allowed to send a sufficient amount of advertising packets, please also check the `RF_AdvertisingTimeout` parameter.

8.14.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.14.2. Example 2

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

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

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

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

8.15. RF_ConnectionTiming: Modify the connection timing

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
8	`RF_ConnectionTiming`	0 - 6	1	read/write	1

The `RF_ConnectionTiming` enables the possibility to configure the timing behavior of the module's RF interface during an established connection. Using this parameter several pre-defined configurations can be chosen, which include the minimum and maximum connection interval, as well as the connection supervision timeout.
The choice of the `RF_ConnectionTiming` primarily determines how rapidly the connection is established and data is transmitted. The lower the `RF_ConnectionTiming`, the more frequently the connected devices communicate with each other and thus, the more power is consumed.

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

Further information:

- The minimum and maximum connection interval parameters specify the borders of the connection interval as determined in the negotiation procedure between the central and the peripheral during connection setup. The connection interval defines the frequency of communication during connection setup and data transmission. If a Proteus-I module A (central) connects to a Proteus-I module B (peripheral), the connection interval settings of the central are used for connection setup. If both modules have different connection interval settings the peripheral requests the central to accept the peripheral's settings after 5s. The central accepts these settings, and thus the peripheral's connection interval is used.
  If now another Bluetooth® LE device (e.g. a smart phone) connects as central to a Proteus-I module (peripheral) and the connection interval settings do not coincide, the Proteus-I requests the smart phone to accept its settings after 5s. If the cell phone does not accept the settings, it will be requested a further 3 times with a delay of 10s. If the peripheral's settings request have been rejected in all cases the connection will

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

be shut down. If the smart phone itself requests to update the connection interval of the Proteus-I, the module accepts the request. Reversely, if a Proteus-I (central) connects to another Bluetooth® LE device (peripheral) and the connection interval settings do not coincide, the Proteus-I accepts all requests of the peripheral to update the connection parameter settings.

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

Please ensure that all members (Proteus-I, 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. Former iOS devices do not support connection intervals shorter than 30ms!

8.15.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.15.2. Example 2

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

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

Response `CMD_GET_CNF`: Successfully read out the value 1.

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

8.16. 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	-4	0	+4
Two's complement, hexadecimal	0xD8	0xEC	0xF0	0xF4	0xF8	0xFC	0x00	0x04

8.16.1. 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

8.16.2. 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

8.17. RF_SPPBaseUUID: Configure the SPP base UUID

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

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

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

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

8.17.1. 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

8.17.2. 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

8.18. RF_Appearance: Configure the appearance of the device

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

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

8.18.1. 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

8.18.2. 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

8.19. UART_BaudrateIndex: Modify the UART speed

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
11	`UART_BaudrateIndex`	See description	3	read/write	1

This parameter defines the baud rate used by the module's UART. The permissible values are listed in the following table.

Permissible values							
`UART_BaudrateIndex`	0	1	2	3	4	5	6
Rate [Baud]	9600	19200	38400	115200	230400	460800	921600

The flow control pins */RTS* and */CTS* can be enabled using the user setting `UART_Flags`. For `UART_BaudrateIndex` 5 and 6 the flow control pins are enabled independent of the `UART_Flags`.

For baud rates faster than 230400 Baud, the flow control pins */RTS* and */CTS* are enabled.
The evaluation board Proteus-I-EV version 2.0 does not provide the connection between the flow control pins of the module and the evaluation board's USB port. Thus in this version of the Proteus-I-EV the flow control can be only used, if the on-board UART is disconnected (remove respective jumpers on JP2) and all UART lines (*URXD*,*UTXD*,*/RTS* and */CTS*) are connected to an external FTDI cable.

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

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

8.19.1. Example 1

Set the baud rate index to 0x04 (230400 Baud) using `CMD_SET_REQ` with settings index 11

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x0B	0x04	0x1E

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 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 0x03, which equals 115200 Baud.

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

8.20. UART_Flags: Configure the UART

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
27	`UART_Flags`	0,1	0	read/write	1

The user setting `UART_Flags` specifies whether the UART uses flow control or not.

Bit no.	Description
0	Set this Bit to 1 to enable the flow control pins */RTS* and */CTS*.
1-7	Reserved.

For baud rates faster than 230400 Baud, the flow control pins */RTS* and */CTS* are enabled.
The evaluation board Proteus-I-EV version 2.0 does not provide the connection between the flow control pins of the module and the evaluation board's USB port. Thus in this version of the Proteus-I-EV the flow control can be only used, if the on-board UART is disconnected (remove respective jumpers on JP2) and all UART lines (*URXD*,*UTXD*,*/RTS* and */CTS*) are connected to an external FTDI cable.

8.20.1. Example 1

Enable the flow control using `CMD_SET_REQ` with settings index 27

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x1B	0x01	0x0B

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 `UART_Flags` using `CMD_GET_REQ` with settings index 27:

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

Response `CMD_GET_CNF`: Successfully read out the value 0x00, meaning that the flow control is disabled.

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

8.21. DIS_ManufacturerName: Configure the manufacturer name

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

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

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

8.21.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.21.2. Example 2

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

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

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

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

8.22. DIS_ModelNumber: Configure the model number

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

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

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

8.22.1. Example 1

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

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

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.22.2. Example 2

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

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

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

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

8.23. DIS_SerialNumber: Configure the serial number

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

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

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

8.23.1. Example 1

Set the serial number to "1.2.3" using `CMD_SET_REQ` with settings index 22

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x06 0x00	0x16	0x31 0x2E 0x32 0x2E 0x33	0x33

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.23.2. Example 2

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

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x16	0x05

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

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

8.24. DIS_HWVersion: Configure the HW version

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
23	`DIS_HWVersion`	See description	"Default"	read/write	1-16

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

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

8.24.1. Example 1

Set the hardware version to "1.2.3" using `CMD_SET_REQ` with settings index 23

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x06 0x00	0x17	0x31 0x2E 0x32 0x2E 0x33	0x32

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.24.2. Example 2

Request the hardware version of the DIS profile using `CMD_GET_REQ`:

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

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

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

8.25. DIS_SWVersion: Configure the SW version

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
24	`DIS_SWVersion`	See description	"Default"	read/write	1-16

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

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

8.25.1. Example 1

Set the software version to "1.2.3" using `CMD_SET_REQ` with settings index 24

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x06 0x00	0x18	0x31 0x2E 0x32 0x2E 0x33	0x3D

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.25.2. Example 2

Request the software version of the DIS profile using `CMD_GET_REQ`:

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

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

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

8.26. DIS_Flags: Configure the device information service

Settings index	Designation	Permissible values	Default value	Permissions	Number of Bytes
19	`DIS_Flags`	0-255	0	read/write	1

The user setting `DIS_Flags` specifies the content of the Device Information Service. To add a specific field, like `DIS_ModelNumber` to the Device Information Service, the corresponding Bit has to be set in the `DIS_Flags`.

Bit no.	Description
0	Set this Bit to 1 to add the `DIS_ManufacturerName` to the Device Information Service.
1	Set this Bit to 1 to add the `DIS_ModelNumber` to the Device Information Service.
2	Set this Bit to 1 to add the `DIS_SerialNumber` to the Device Information Service.
3	Set this Bit to 1 to add the `DIS_HWVersion` to the Device Information Service.
4	Set this Bit to 1 to add the `DIS_SWVersion` to the Device Information Service.
5-7	Reserved.

8.26.1. Example 1

Add the manufacturer name and model number (Bit0|Bit1 = 0x03) to the Device Information Service using `CMD_SET_REQ` with settings index 19

Start signal	Command	Length	Settings index	Parameter	CS
0x02	0x11	0x02 0x00	0x13	0x03	0x01

Response `CMD_SET_CNF`: Successfully modified the setting.

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

8.26.2. Example 2

Request the `DIS_Flags` using `CMD_GET_REQ`:

Start signal	Command	Length	Settings index	CS
0x02	0x10	0x01 0x00	0x13	0x00

Response `CMD_GET_CNF`: Successfully read out the value 0x00, meaning that the Device Information Service is disabled, since no field was added.

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

Settings index	Designation	Summary	Permissible values	Default value	Permission	Number of Bytes
1	`FS_FWVersion`	Version of the firmware	-	-	read	3
2	`RF_DeviceName`	Name of the module	See description	"A2621"	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_ConnectionTiming`	Module connection timing configuration	0 - 6	1	read / write	1
9	`RF_ScanTiming`	Module advertising and scanning timing configuration	0 - 5	1	read / write	1
10	`RF_ScanFactor`	Factor between scan interval and scan window	1 - 10	2	read / write	1
11	`UART_BaudrateIndex`	Baud rate of the UART	See description	3	read / write	1
12	`RF_SecFlags`	Security settings of the module	See description	0	read / write	1
13	`RF_ScanFlags`	Scan settings of the module	See description	0	read / write	1
14	`RF_BeaconFlags`	Beacon settings of the module	See description	0	read / write	1
15	`FS_DeviceInfo`	Information about the chip	-	-	read	12
16	`FS_SerialNumber`	Serial number of the module	-	-	read	3
17	`RF_TXPower`	Output power [dBm] Two's complement	See description	4	read / write	1

Table 17: Table of settings (Part 1)

Settings index	Designation	Summary	Permissible values	Default value	Permissio	Number of Bytes
18	`RF_StaticPasskey`	6 digit pass key	See description	"123123"	read / write	6
19	`DIS_Flags`	Flags for the DIS	0 - 255	0	read / write	1
20	`DIS_ManufacturerName`	Manufacturer name field of the DIS	See description	"Default"	read / write	1-64
21	`DIS_ModelNumber`	Model number field of the DIS	See description	"Default"	read / write	1-64
22	`DIS_SerialNumber`	Serial number field of the DIS	See description	"Default"	read / write	1-64
23	`DIS_HWVersion`	HW version field of the DIS	See description	"Default"	read / write	1-16
24	`DIS_SWVersion`	SW version field of the DIS	See description	"Default"	read / write	1-16
25	`RF_Appearance`	Appearance	0-65535	0	read / write	2
26	`RF_SPPBaseUUID`	Base UUID of the AMBER SPP-like profile	See description	See description	read / write	16
27	`UART_Flags`	UART Flags	0,1	0	read / write	1
44	`RF_SecFlagsPerOnly`	Security settings of the module (peripheral only mode only)	See description	11	read / write	1

Table 18: 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	4	ms

9.2. Bluetooth LE timing parameters

The timing parameters for sending advertising packets or scanning are determined by the user settings `RF_ScanTiming`, `RF_ScanFactor` and `RF_AdvertisingTimeout`. Using these settings, the advertising interval, the advertising timeout, the scan interval and the scan window can be configured. Furthermore, the user setting `RF_ConnectionTiming` allows to configure the timing parameters used during connection setup and connection retention, as well as the connection interval and the connection supervision timeout.

9.3. Connection establishment

The time needed to establish a connection sums up as the time needed to detect the selected peripheral on air and the time needed for connection parameter negotiation and service discovery.

1. Peripheral detection To establish a connection, the initiating device (central) waits for an advertising packet, which was sent by the peripheral to which it wants to connect to. As soon as such an advertising packet has been received, the central sends a connection request to the chosen peripheral. The time needed to receive this advertising packet strongly depends on the advertising interval of the peripheral as well as on the scan interval and scan window of the central (see `RF_ScanTiming`).

2. Connection parameter negotiation After the connection request has been sent the central and peripheral negotiate the timing and security parameters of the connection. To finish this procedure and discover the services of the peripheral several messages have to be sent, whereby only one is sent per connection interval (see `RF_ConnectionTiming`).

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

Knowing the connection interval and the number of messages that will be sent, the time necessary to setup a connection can be estimated by multiplying the number of messages with the connection interval.

In case the Device Information Service is enabled, the number of messages and thus the timing of the connection setup may be increased.

9.4. Connection based data transmission

After setting up a connection, data can be transmitted using the `CMD_DATA_REQ`. It buffers the data in the module and sends it with the next connection interval event. As soon as the data has been transmitted successfully, a `CMD_TXCOMPLETE_RSP` is returned by the UART. The time needed for this coincides with the connection interval that was negotiated during connection setup. The `RF_ConnectionTiming` parameter defines the minimum and maximum connection interval, which is supported by the module.

10. Peripheral only mode

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

- Peripheral only functions: The Proteus-I only contains the functions of a peripheral. Thus, it is advertising until another Bluetooth® LE device connects to it. In this case, the UART of the Proteus-I is enabled, the *LED_2* pin shows that the channel is open and bidirectional data transmission can start. As soon as the connection is closed, the UART is disabled again to save power. Since all central functions are no longer valid, the module cannot initiate any connection or run scans.
- Transparent UART interface: The serial interface of the Proteus-I is no longer driven by commands. This means, when the UART of the module is enabled (i.e. only when a channel is open, indicated by both LEDs active), data sent to the UART is transmitted by the Proteus-I to the connected Bluetooth® LE device. On the other hand, all data received by RF is send from the Proteus-I to the connected host without additional header Bytes. The UART is only running, when a channel is open. Thus, power is saved during the advertising period. Depending on the configured connection interval, only one packet per interval is allowed to be transmitted. Since the commands of the command interface are no longer valid, a Proteus-I cannot be configured when running in peripheral only mode.
- Pairing: The default security mode is the static passkey pairing method (see `RF_SecFlagsPerOnly`), with the default key "123123". The bonding feature is enabled by default.

10.1. Reasons to use the peripheral only mode

The Proteus-I peripheral only mode equips custom applications with a Bluetooth® LE interface (to be accessible by other Bluetooth® LE devices) without installation effort.
To setup a connection to the Proteus-I in peripheral only mode the central device has to insert the Proteus-I's static passkey. As soon as the channel to a connected Bluetooth® LE central device is open, the *LED_2* pin switches on to signalize that data can be exchanged now. When the connection was shut down by the Bluetooth® LE central device, the *LED_2* pin switches off again.
Due to the transparent UART interface, data can be exchanged without additional headers. Furthermore, the peripheral only mode allows an energy efficient operation of the Bluetooth® LE interface, since the UART is only enabled when it is really used.

10.2. How to use the peripheral only mode

The peripheral only mode is enabled, when a high signal is present on the *OP_MODE* pin during device start-up or reset.

No configuration of the module is needed for this operating mode. The module shall be set to factory settings if reconfigured before so it uses the default user settings. In this case, the UART uses 115200 Baud 8n1 and static passkey pairing is used as authentication method.

If a configuration of the module is still needed (e.g. when another UART baud rate needs to be chosen), the module has to be started in normal mode and the `CMD_SET_REQ` may be used to update the user settings.
It is permitted to modify any user setting to change the behavior of the peripheral only mode. Nevertheless, we recommend to update only the following parameters to run the device in factory state with minimal adaptions:

- `UART_BaudrateIndex` (change the UART baud rate, default value "115200")
- `UART_Flags` (enable or disable the flow control)
- `RF_StaticPasskey` (change the default static passkey, default value "123123")

10.3. More information

10.3.1. Radio

- In peripheral only mode a new 8-digit device name is automatically generated by the `FS_BTMAC`. In case of the `FS_BTMAC` equals 0x0018DA123456 the device name is "A-123456". This is a workaround for iOS, which does not allow access to the BTMAC for received Bluetooth® frames.
- The content of the advertising packet was changed in peripheral only mode. The TX power information block was removed, as the device name was extended to 8 digits.

10.3.2. UART

- The maximum payload per packet supported by an open channel depends on the connected central device. The Proteus-I supports up to 243 Bytes 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 packet and connection interval as given by the Bluetooth® 4.0 standard is used (compatibility mode to Bluetooth® LE 4.0 devices). Data received by the Proteus-I's UART, that exceeds the maximum payload size of the open channel, is discarded. In peripheral only mode, (due to the deactivated commands) the Proteus-I cannot inform its host about the maximum payload size or of payload discarding.
- The connecting device could implement a function to inform the host behind the Proteus-I which MTU the channel is capable of. Until this message is received, the host shall assume a payload capability of up to 19 Byte.
- The data sent to the UART is buffered in the Proteus-I up to a maximum payload depending on of the current channel MTU. When no new Byte was received for 20ms, the data will be transmitted by RF to the connected Bluetooth® LE device.

11. Customizing the Proteus-I

11.1. DIS - Device information service

Besides the AMBER SPP-like profile for data transmission, the Proteus-I contains the so called Device Information Service. This profile exposes manufacturer information about a device and is used to personalize the Proteus-I to fuse with the custom product. The Device Information Service is a standard Bluetooth® LE profile that is recognized by all devices with Bluetooth® capabilities. It contains the following fields, that can only be modified by updating the respective user setting using the `CMD_SET_REQ` command:

Field name	User setting	Maximum length
Manufacturer Name String	`DIS_ManufacturerName`	64
Model Number String	`DIS_ModelNumber`	64
Serial Number String	`DIS_SerialNumber`	64
Hardware Revision String	`DIS_HWVersion`	16
Software Revision String	`DIS_SWVersion`	16

Furthermore, the user setting `DIS_Flags` defines which of the described DIS fields are finally placed in the DIS profile. Thus after adding content to the a DIS field user setting, like `DIS_ManufacturerName`, the user setting `DIS_Flags` has to be adapted such that the content is added to the profile.

11.2. UUID

The UUID is a unique number identifying a Bluetooth® LE profile and thus describing its functions. The Proteus-I using its standard UUID is compatible to all devices that implement the AMBER SPP-like profile, whichever device it is integrated. To suspend this interoperability, the user setting `RF_SPPBaseUUID` can be used to modify the UUID of the AMBER SPP-like profile. With this, a new custom SPP-like profile is defined that is solely known to those that chose the new UUID.
To generate a custom UUID the Bluetooth® SIG recommends to use the tool:
http://www.uuidgenerator.net/

11.3. Appearance

The appearance of the Bluetooth® device is a 2 Bytes value defined by the Bluetooth® SIG. It can be configured by adapting the parameter `RF_Appearance`.

12. Custom firmware

12.1. 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.

12.2. 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^{2}C 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.

12.3. Customer firmware

A customer firmware is a firmware written and tested by the customer himself or a 3rd party as a customer representative specifically for the hardware platform provided by a module.
This customer firmware (e.g. in form of a Intel hex file) will be implemented into the module's production process at our production side.
This also results in a customer exclusive module with a unique ordering number.
The additional information needed for this type of customer firmware, such as hardware specific details and details towards the development of such firmware are not available for the public and can only be made available to qualified customers.

STOP The qualification(s) and certification(s) of the standard firmware cannot be applied to this customer firmware solution without a review and verification.

12.4. Contact for firmware requests

Please contact your local field sales engineer (FSE) or wireless-sales@we-online.com for quotes regarding this topics.

13. Firmware update

The Proteus-I offers two possibilities of updating its firmware.

13.1. Firmware flashing using the SWD interface

The SWD is a production interface and performing an update it will erase any pre-installed firmware and calibration values of the Proteus-I.

Any certification, declaration and qualification that has requirements in a combination of hard- and firmware is invalidated by this action.

13.2. Firmware update using the Proteus-I OTA bootloader

This method offers a possibility to update the firmware over the air (OTA). Therefore, the Nordic nRF52 Bluetooth® LE DFU Secure Bootloader is integrated into the Proteus-I's firmware, which will communicate over the Bluetooth® LE interface. The OTA bootloader mode is a distinct operating mode besides the normal operating modes mentioned before. For this reason, a .zip-file can be provided, which contains all (bootloader, Softdevice, application) parts of the firmware in an encrypted and authenticated package.

Before starting any update procedure, please check whether the installed firmware can be updated to a new one:

Version of the firmware before the update	Version of the new firmware	Version of the Proteus-I Toolbox App (Android)
1.0.0 - 1.1.0	1.0.0 - 1.1.0	1.16.2, 1.18.4
1.0.0 - 1.1.0	2.1.0	Not supported, due to S132 update and bootloader changes
2.1.0	2.1.0	1.18.4
2.1.0, 3.X.X	3.X.X	1.18.4, Nordic nRF Toolbox 2.2.1 or newer

Table 19: Compatibility matrix

To start the bootloader, one of the following two conditions has to be satisfied:

1. send the command `CMD_BOOTLOADER_REQ` to the module to restart in bootloader mode
2. during a reset and while restarting, a low signal has to be present on the *BOOT* pin of the module to start it in bootloader mode

The bootloader mode has started successfully if *LED_1* has turned on. After the bootloader has started successfully, the module goes into the advertising mode using the name

"DFU2621". Now, any Bluetooth® LE device hosting an application that understands the commands of the Nordic nRF52 Bluetooth® LE DFU Bootloader can connect in order to update the Proteus-I firmware.
The DFU application of the used App (see Table 19) is such an application. For more details, please refer to chapter 13.2.1. As soon as a connection has been set up, *LED_1* turns off again and *LED_2* turns on.

The implemented Nordic nRF52 Bluetooth® LE DFU bootloader uses a dual bank method to update the firmware. Thus, the old firmware is only replaced once the new firmware has been transferred and authenticated successfully. This prevents the module from being flashed with a faulty firmware.

An OTA firmware update will take several minutes to be performed, the duration is also dependent how much of the firmware shall be updated (application only or complete update).

The max connection interval of the update service is set to 30ms. Please check whether your mobile supports this speed.

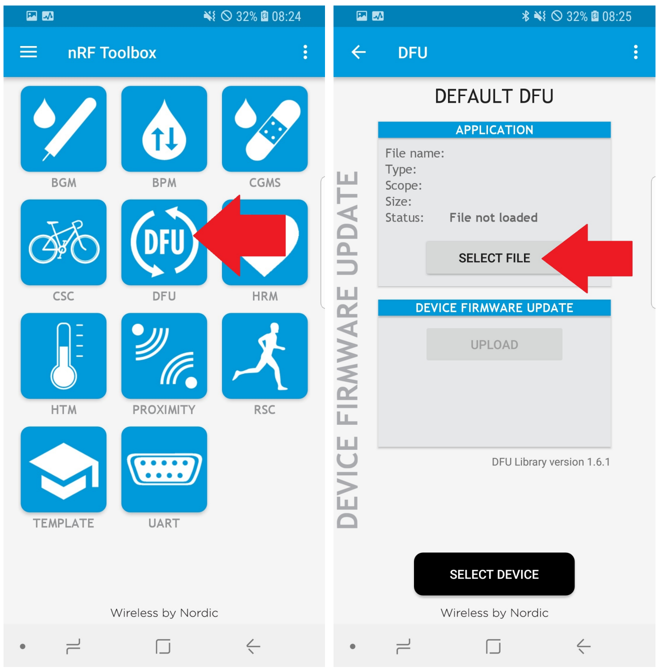
STOP

This method is only applicable if the Proteus-I still contains an intact bootloader.

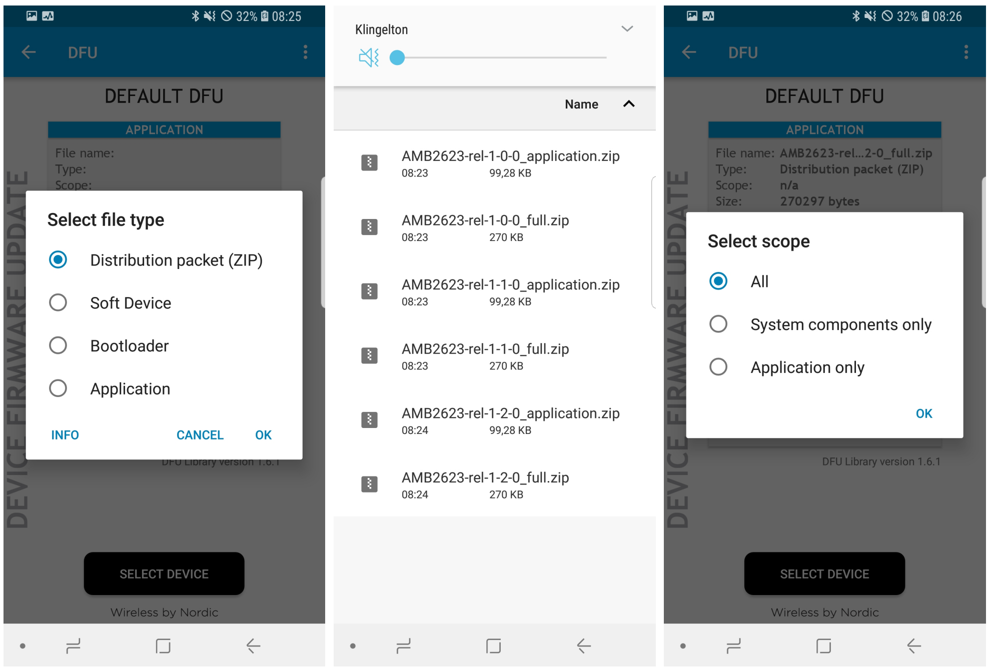
13.2.1. Firmware update steps using the Nordic nRF Toolbox app

If the radio module Proteus-I has been set to bootloader mode, the Nordic nRF Toolbox app can be used to perform the OTA firmware update.

- Open the app, select the DFU function and press "SELECT FILE"

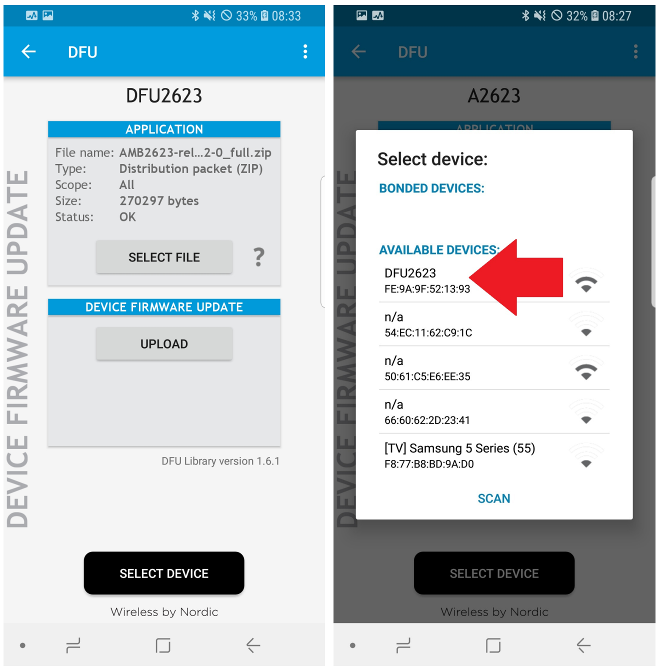


- Choose "Distribution packet (ZIP)", select the new firmware and choose "All".


DFU
DEFAULT DFU
APPLICATION
Select file type
Distribution packet (ZIP)
Soft Device
Bootloader
Application
INFO
CANCEL
OK
SELECT DEVICE
Wireless by Nordic
Klingelton
Name
AMB2623-rel-1-0-0_application.zip 08:23 99,28 KB
AMB2623-rel-1-0-0_full.zip 08:23 270 KB
AMB2623-rel-1-1-0_application.zip 08:23 99,28 KB
AMB2623-rel-1-1-0_full.zip 08:23 270 KB
AMB2623-rel-1-2-0_application.zip 08:24 99,28 KB
AMB2623-rel-1-2-0_full.zip 08:24 270 KB
File name: AMB2623-rel...2-0_full.zip
Type: Distribution packet (ZIP)
Scope: n/a
Size: 270297 bytes
Select scope
All
System components only
Application only
OK
DFU Library version 1.6.1
SELECT DEVICE
Wireless by Nordic

- Press "SELECT DEVICE" and choose the appropriate module in the list of displayed devices. In bootloader mode the module is named "DFUxxxx".

If there is no device named "DFUxxxx" on the radio, please check whether the module has been started in bootloader mode.



33% 08:33

DFU

DFU2623

APPLICATION

File name: AMB2623-rel...2-0_full.zip
Type: Distribution packet (ZIP)
Scope: All
Size: 270297 bytes
Status: OK

SELECT FILE ?

DEVICE FIRMWARE UPDATE

UPLOAD

DFU Library version 1.6.1

DEVICE FIRMWARE UPDATE

SELECT DEVICE

Wireless by Nordic

32% 08:27

DFU

A2623

Select device:

BONDED DEVICES:

AVAILABLE DEVICES:

DFU2623
FE:9A:9F:52:13:93

n/a
54:EC:11:62:C9:1C

n/a
50:61:C5:E6:EE:35

n/a
66:60:62:2D:23:41

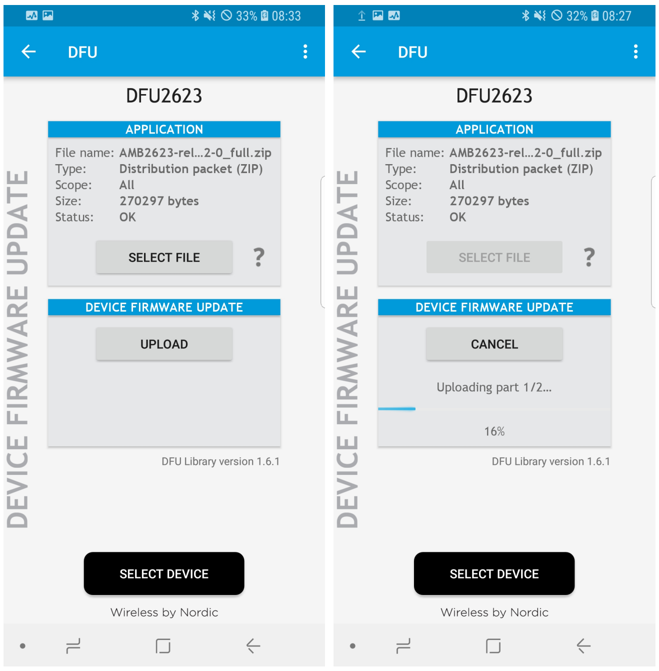
[TV] Samsung 5 Series (55)
F8:77:B8:BD:9A:D0

SCAN

SELECT DEVICE

Wireless by Nordic

- Then press "UPLOAD" to transmit the selected firmware to the selected device.



33% 08:33

DFU

DFU2623

APPLICATION

File name: AMB2623-rel...2-0_full.zip
Type: Distribution packet (ZIP)
Scope: All
Size: 270297 bytes
Status: OK

SELECT FILE ?

DEVICE FIRMWARE UPDATE

UPLOAD

DFU Library version 1.6.1

DEVICE FIRMWARE UPDATE

SELECT DEVICE

Wireless by Nordic

32% 08:27

DFU

DFU2623

APPLICATION

File name: AMB2623-rel...2-0_full.zip
Type: Distribution packet (ZIP)
Scope: All
Size: 270297 bytes
Status: OK

SELECT FILE ?

DEVICE FIRMWARE UPDATE

CANCEL

Uploading part 1/2...

16%

DFU Library version 1.6.1

DEVICE FIRMWARE UPDATE

SELECT DEVICE

Wireless by Nordic

14. Firmware history

Version 0.x.x "Engineering"

- Pre-Release for test run

Version 1.0.0 "Release"

- First production release
- New command interface with 2 Bytes length field
- using Softdevice 2.0.1 + SDK 11.0
- SPP-Like Protocol
- Known issues:
  - `CMD_SET_REQ` does not run with parameter `RF_AdvertisingTimeout`

Version 1.0.1 "Release"

- UART checks for max buffer size

Version 1.1.0 "Release"

- DCDC enabled for lowest power consumption

Version 2.1.0 "Release"

- Using Softdevice 3.0.0 and SDK 12.1.0
- Remove UART baud rates faster than 230400 baud to prevent lost Bytes
- due to DMA usage the UART current is increased (without DMA the UART data rate must be decreased further below 230400 baud)
- Introduced `CMD_ERROR_IND` message indicating internal error states
- Introduced support for transmission of large Bluetooth® LE packets (19 Bytes payload $\rightarrow$ 128Bytes payload). This is a non-mandatory Bluetooth® LE 4.2 feature. Use `CMD_DATA_REQ` command to send long packets if it is indicated by the `CMD_CHANNELOPEN_RSP`.
- Modified the `CMD_CHANNELOPEN_RSP` indication the max. supported payload size
- `CMD_DATA_REQ` returns maximum supported payload size if it was exceeded
- Modified `RF_ConnectionTiming` profile 0 and added profile 6
- Modified the `CMD_SECURITY_IND` message
- DTM uses max packet size for TX test packets (255 Bytes)
- Parameter `RF_SecLTK` replaced by `RF_OwnLTK` and `RF_PeerLTK`
- Added commands `CMD_SET_RAM_REQ` and `CMD_GET_RAM_REQ` to set/get volatile RAM parameter values (only `RF_PeerLTK` at the moment)
- Moved the settings index of parameter `RF_TXPower`
- New OTA bootloader
- AMB2621 Toolbox App version 1.18.4 must be used to update the firmware
- `CMD_DATAEX_REQ` removed due to incompatibilities with foreign Bluetooth® LE devices

- Added new security concept. Now the peripheral decides whether the security level is sufficient.
- Added new security mode in `RF_SecFlags` (Static pass key method was added)
- New user setting `RF_StaticPasskey` added
- New commands `CMD_PASSKEY_REQ` and `CMD_PASSKEY_IND` added
- Known issues:
  - OTA update from version 1.1.0 to 2.1.0 not supported
  - Compatibility of version 2.1.0 to older versions only given when no security mode is enabled
  - Choosing invalid value for UserSetting `RF_StaticPasskey` may lead to malfunction
  - The output power value `RF_TXPower` equals -30dBm is invalid and forces a firmware OTA update to recover the device

Version 3.0.0 "Release"

- Using Softdevice 3.1.0 and SDK 12.1.0
- Changed default value of user setting parameter `RF_AdvertisingTimeout` from 180s to 0s. This means, that in default configuration the module does not go to sleep after 180s as in the previous firmware versions.
- Function of *LED_2* has changed. Now it indicates whether a channel is open or not.
- Introduced new operation mode "Peripheral only mode" with special behavior
  - Introduced new Advertise format containing the LSBs of the MAC address (only for Peripheral only mode)
  - Introduced pin *OP_MODE* to enable the Peripheral only mode, as an internal pull-down is used the "do not connect if not needed" still applies for normal mode operation
  - Introduced a transparent UART interface (only available for this mode)
  - Introduced a new user setting `RF_SecFlagsPerOnly`
- Known issues:
  - OTA update from version 1.x.x to 3.0.0 not supported
  - Choosing invalid value for UserSetting `RF_StaticPasskey` may lead to malfunction
  - The output power value `RF_TXPower` equals -30dBm is invalid and forces a firmware OTA update to recover the device

Version 3.3.0 "Release"

- Using Softdevice 3.1.0 and SDK 12.3.0
- New compatible OTA bootloader
- Increased maximum payload from 128 Bytes to 243 Bytes
- Added bonding feature and new commands `CMD_DELETEBONDS_REQ` and `CMD_GETBONDS_REQ`

- Update of the user setting `RF_SecFlags`
  - Removed the direct LTK-encryption option
  - Added flag SECFLAGS_BONDING_ENABLE to switch on bonding
- Default value of user settings `RF_SecFlagsPerOnly` changed to 11 (Static pass key and bonding enabled)
- Removed user setting `RF_OwnLTK` and `RF_PeerLTK`
- Modified the message `CMD_SECURITY_IND`
- `CMD_SETBEACON_REQ` allows 0 as length
- Known issues:
  - OTA update from version 1.x.x to 3.3.0 not supported
  - Bluetooth® LE compatibility to version 3.0.0 is only given when bonding is not enabled
  - Choosing invalid value for UserSetting `RF_StaticPasskey` may lead to malfunction
  - The output power value `RF_TXPower` equals -30dBm is invalid and forces a firmware OTA update to recover the device

Version 3.3.6 "Release"

- Blink interval of *LED_1* changed in mode `ACTION_SCANNING`, now it is 1000ms on 1000ms off
- Known issues:
  - OTA update from version 1.x.x to 3.3.6 not supported
  - Bluetooth® LE compatibility to version 3.0.0 is only given when bonding is not enabled
  - The output power value `RF_TXPower` equals -30dBm is invalid and forces a firmware OTA update to recover the device

Version 3.4.0 "Release"

- Added possibilities for improved customization of the Proteus-I see chapter 11.
  - Added the user setting `RF_SPPBaseUUID` to modify the UUID of the AMBER SPP-like profile to generate a customized profile.
  - Added the user setting `RF_Appearance` to modify the appearance of the module
  - Added the Device Information Service (DIS) as second Bluetooth® LE profile to the Proteus-I to add customer specific data to the Bluetooth® LE interface of the Proteus-I
- Increased the maximum length of the user setting `RF_DeviceName` to 32 Bytes, that defines the name of the device on radio.
- Higher throughput due to faster UART baud rates.
  - Added the flow control pins */RTS* and */CTS* to the pinout.
  - Added 460800 and 921600 Baud to the user setting `UART_BaudrateIndex` using the flow control.

    - Added the user setting `UART_Flags` to enable/disable the flow control for UART baud rates slower than 430800 Baud.
- Reduced the detection time of pin *WAKE_UP* from 50ms to 10ms to re-enable the UART again, when the UART was switched off using the command `CMD_UARTDISABLE_REQ`.
- Known issues:
    - OTA update from version 1.x.x to 3.x.x not supported
    - Bluetooth® LE compatibility to version 3.0.0 is only given when bonding is not enabled
    - The output power value `RF_TXPower` equals -30dBm is invalid and forces a firmware OTA update to recover the device

Version 3.5.0 "Release"

- Integrated additional internal tests for better detection of production failures on our production sides
- Known issues:
    - Using 32 byte `RF_DeviceName` will result in a malfunctioning device. The device can be recovered by a FOTA (firmware over the air) update
    - OTA update from version 1.x.x to 3.x.x not supported
    - Bluetooth® LE compatibility to version 3.0.0 is only given when bonding is not enabled
    - The output power value `RF_TXPower` equals -30dBm is invalid and forces a firmware OTA update to recover the device

15. Design in guide

15.1. 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.).

STOP

The use of an external reset IC should be considered if one of the following points is relevant:

- The slew rate of the power supply exceeds the electrical specifications.
- The effect of different current consumptions on the voltage level of batteries or voltage regulators should be considered. The module draws higher currents in certain scenarios like start-up or radio transmit which may lead to a voltage drop on the supply. A restart under such circumstances should be prevented by ensuring that the supply voltage does not drop below the minimum specifications.
- Voltage levels below the minimum recommended voltage level may lead to misfunction. The /Reset pin of the module shall be held on LOW logic level whenever the VCC is not stable or below the minimum operating Voltage.
- Special care must be taken in case of battery powered systems.

- Elements for ESD protection should be placed on all pins that are accessible from the outside and should be placed close to the accessible area. For example, the RF-pin is accessible when using an external antenna and should be protected.
- ESD protection for the antenna connection must be chosen such as to have a minimum effect on the RF signal. For example, a protection diode with low capacitance such as the 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.).

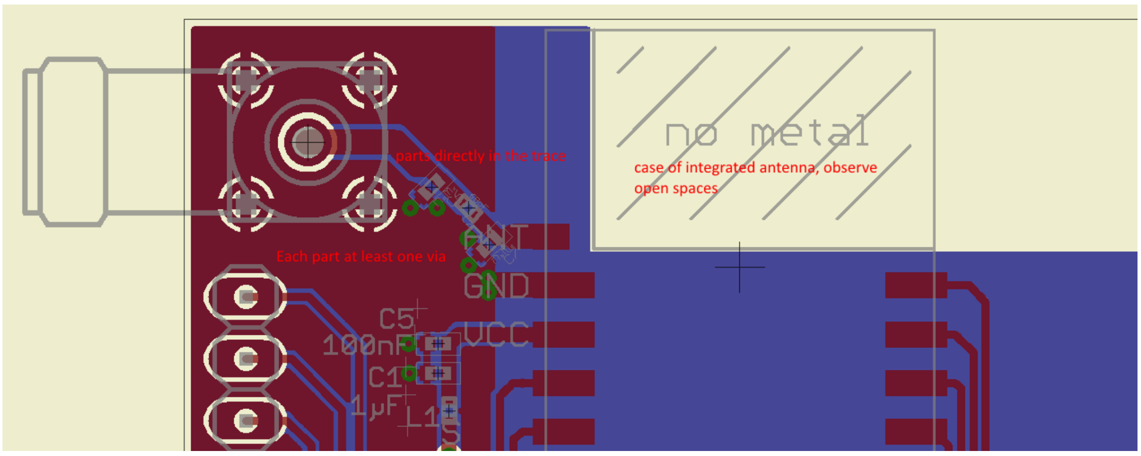

no metal
parts directly in the trace
case of integrated antenna, observe open spaces
Each part at least one via
ANT
GND
VCC
C5
100nF
C1
1µF
L1

Figure 8: Layout

- To avoid the risk of short circuits and interference there should be no routing underneath the module on the top layer of the baseboard.
- On the second layer, a ground plane is recommended, to provide good grounding and shielding to any following layers and application environment.
- In case of integrated antennas it is required to have areas free from ground. This area should be copied from the evaluation board.
- The area with the integrated antenna must overlap with the carrier board and should not protrude, as it is matched to sitting directly on top of a PCB.
- Modules with integrated antennas should be placed with the antenna at the edge of the main board. It should not be placed in the middle of the main board or far away from the edge. This is to avoid tracks beside the antenna.

- Filter and blocking capacitors should be placed directly in the tracks without stubs, to achieve the best effect.
- Antenna matching elements should be placed close to the antenna / connector, blocking capacitors close to the module.
- Ground connections for the module and the capacitors should be kept as short as possible and with at least one separate through hole connection to the ground layer.
- ESD protection elements should be placed as close as possible to the exposed areas.

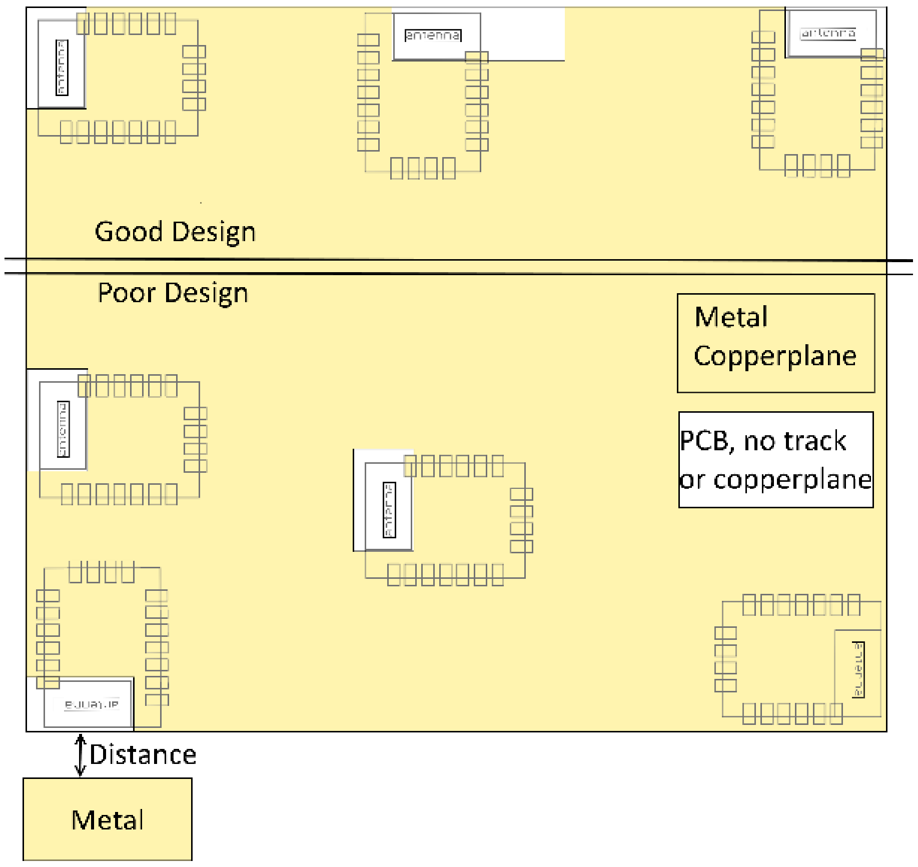


Figure 9: Placement of the module with integrated antenna

15.2. Dimensioning of the micro strip antenna line

The antenna track has to be designed as a 50Ω feed line. The width W for a micro strip can be calculated using the following equation:

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

Example:
A FR4 material with ε_r = 4.3, a height H = 1000 µm and a copper thickness of T_{met}= 18 µm

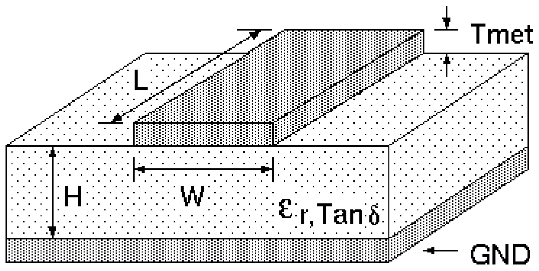


Figure 10: Dimensioning the antenna feed line as micro strip

will lead to a trace width of W $\sim$ 1.9 mm. To ease the calculation of the micro strip line (or e.g. a coplanar) many calculators can be found in the internet.

- As rule of thumb a distance of about 3×W should be observed between the micro strip and other traces / ground.
- The micro strip refers to ground, therefore there has to be the ground plane underneath the trace.
- Keep the feeding line as short as possible.

15.3. Antenna solutions

There exist several kinds of antennas, which are optimized for different needs. Chip antennas are optimized for minimal size requirements but at the expense of range, PCB antennas are optimized for minimal costs, and are generally a compromise between size and range. Both usually fit inside a housing.
Range optimization in general is at the expense of space. Antennas that are bigger in size, so that they would probably not fit in a small housing, are usually equipped with a RF connector. A benefit of this connector may be to use it to lead the RF signal through a metal plate (e.g. metal housing, cabinet).
As a rule of thumb a minimum distance of $\lambda/10$ (which is 3.5 cm @ 868 MHz and 1.2 cm @ 2.44 GHz) from the antenna to any other metal should be kept. Metal placed further away will not directly influence the behavior of the antenna, but will anyway produce shadowing.

Keep the antenna away from large metal objects as far as possible to avoid electromagnetic field blocking.

The choice of antenna might have influence on the safety requirements.

In the following chapters, some special types of antenna are described.

15.3.1. 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.

15.3.2. 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.

15.3.3. 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.

15.3.4. Antennas provided by Würth Elektronik eiSos

15.3.4.1. 2600130021 - Himalia - 2.4 GHz dipole antenna

Figure 11: 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. [℃]	-40 – +80

Special care must be taken for FCC certification when using this external antenna to fulfill the requirement of permanently attached antenna or unique coupling for example by using the certified dipole antenna in a closed housing, so that only through professional installation it is possible to remove it.

16. Reference design

Proteus-I was tested and certified on the corresponding Proteus-I evaluation board. For the compliance with the EU directive 2014/53/EU Annex I, the evaluation board serves as reference design.
This is no discrepancy due to the fact that the evaluation board itself does not fall within the scope of the EU directive 2014/53/EU Annex I as the module is tested on the evaluation board, which is also the recommended use.

Further information concerning the use of the evaluation board can be found in the manual of the Proteus-I evaluation board.

16.1. Schematic

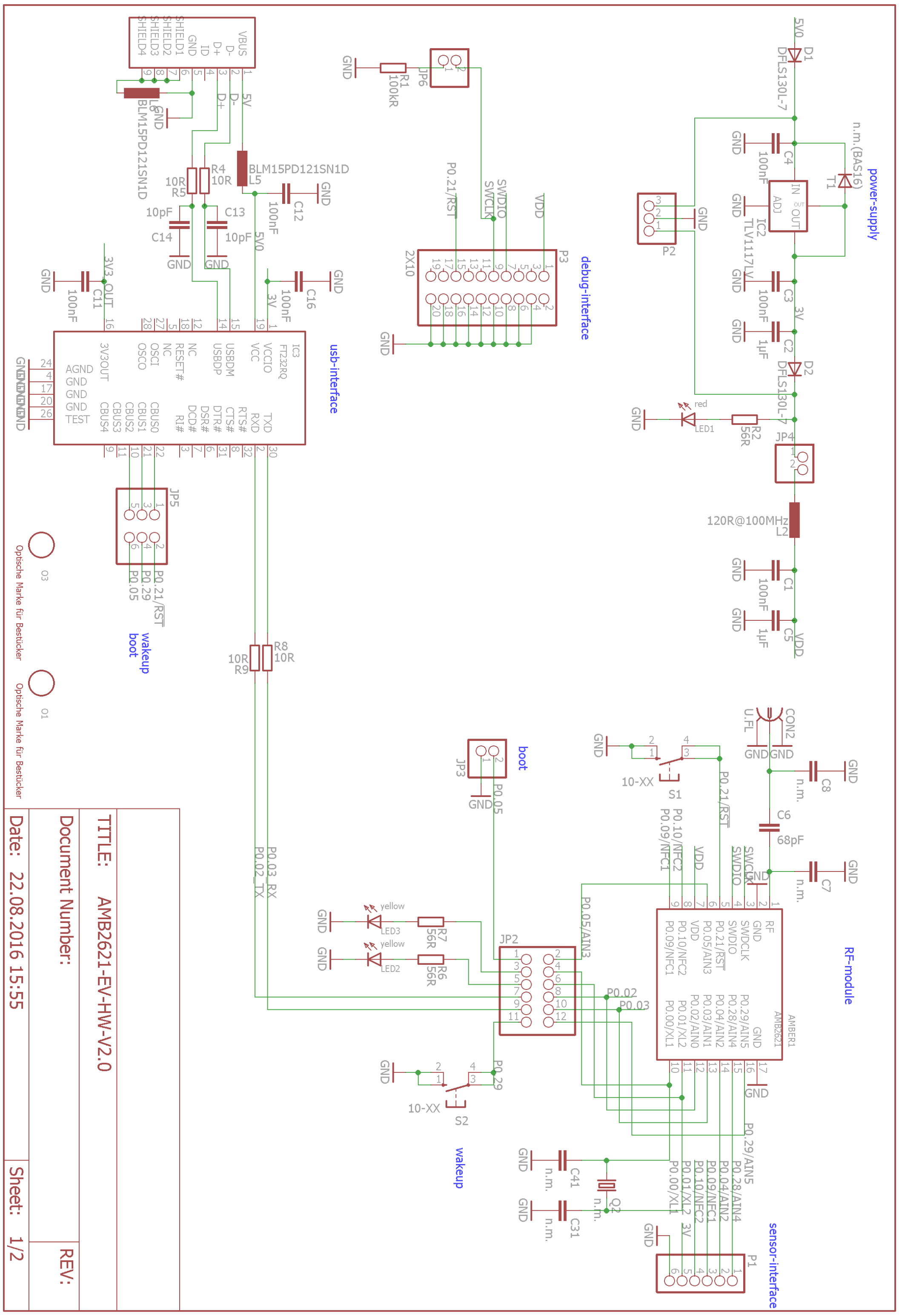


Figure 12: Circuit diagram

16.2. Layout

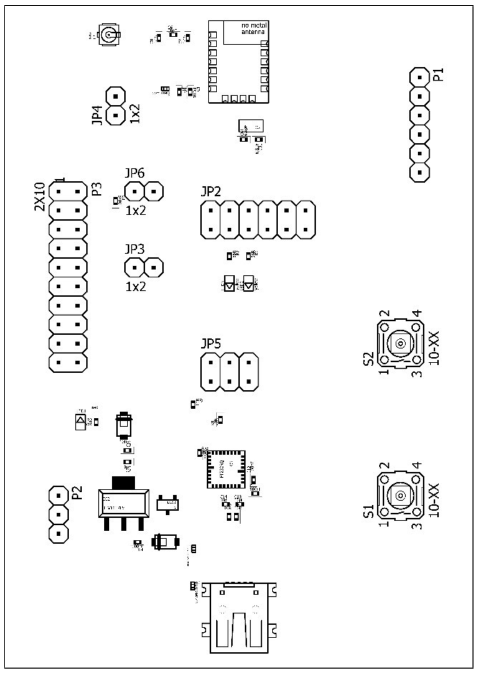


Figure 13: Assembly diagram

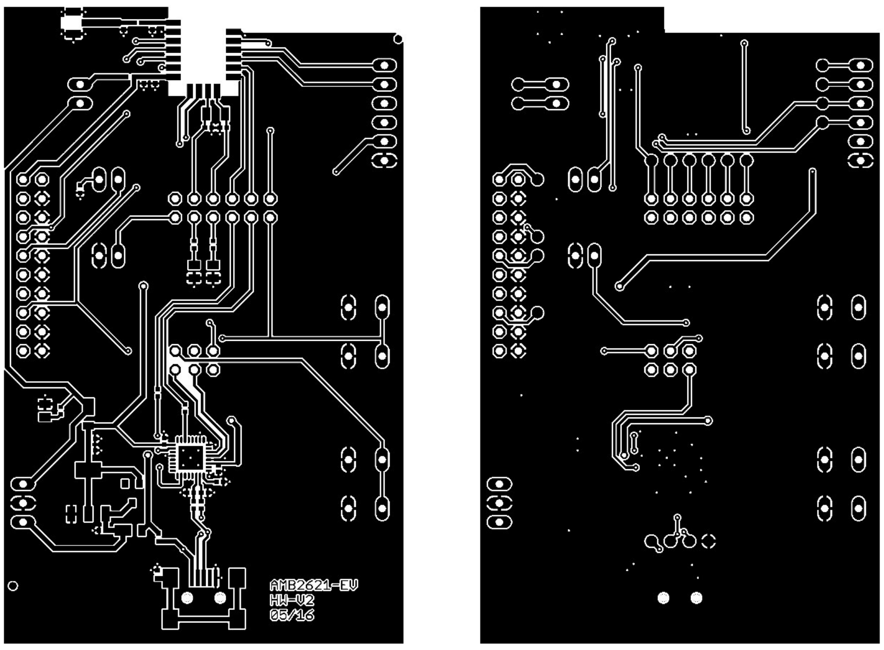


Figure 14: Top & bottom Layer

17. Manufacturing information

17.1. 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*.

17.2. Soldering

17.2.1. 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℃
Preheat temperature Max	$T_{S\ Max}$	200℃
Preheat time from $T_{S\ Min}$ to $T_{S\ Max}$	t_S	60 - 120 seconds
Ramp-up rate (T_L to T_P)		3℃ / second max.
Liquidous temperature	T_L	217℃
Time t_L maintained above T_L	t_L	60 - 150 seconds
Peak package body temperature	T_P	see table below
Time within 5℃ of actual peak temperature	t_P	20 - 30 seconds
Ramp-down Rate (T_P to T_L)		6℃ / second max.
Time 20℃ to T_P		8 minutes max.

Table 20: Classification reflow soldering profile, Note: refer to IPC/JEDEC J-STD-020E

Package thickness	Volume mm^3 <350	Volume mm^3 350-2000	Volume mm^3 >2000
< 1.6mm	260℃	260℃	260℃
1.6mm - 2.5mm	260℃	250℃	245℃
> 2.5mm	250℃	245℃	245℃

Table 21: 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.

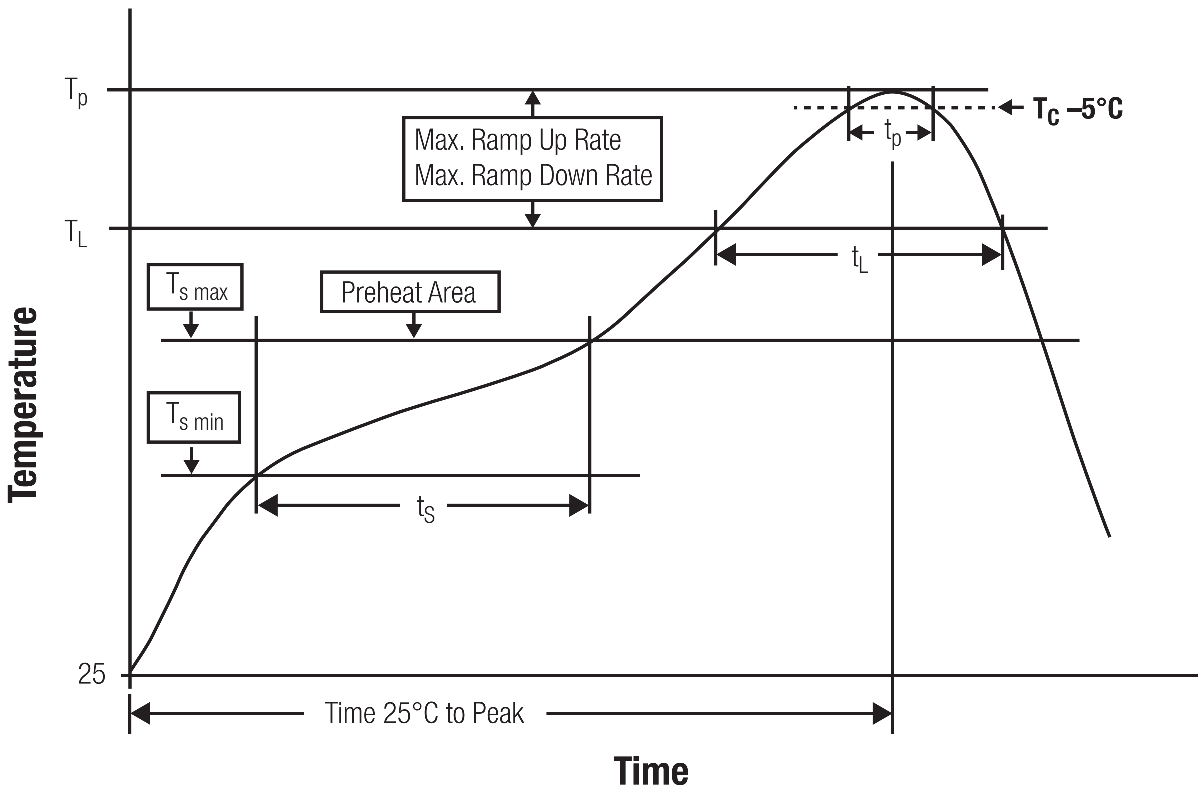


Figure 15: Reflow soldering profile

After reflow soldering, visually inspect the board to confirm proper alignment

17.2.2. 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.

17.2.3. Other notations

- Conformal coating of the product will result in the loss of warranty. The RF shields will not protect the part from low-viscosity coatings.
- Do not attempt to improve the grounding by forming metal strips directly to the EMI covers or soldering on ground cables, as it may damage the part and will void the warranty.
- Always solder every pad to the host PCB even if some are unused, to improve the mechanical strength of the module.
- The part is sensitive to ultrasonic waves, as such do not use ultrasonic cleaning, welding or other processing. Any ultrasonic processing will void the warranty.

17.3. 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.

17.4. 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.

18. Physical dimensions

18.1. Dimensions

Dimensions
11 x 8 x 2 mm

Table 22: Dimensions

18.2. Weight

Weight
<1g

Table 23: Weight

18.3. Module drawing

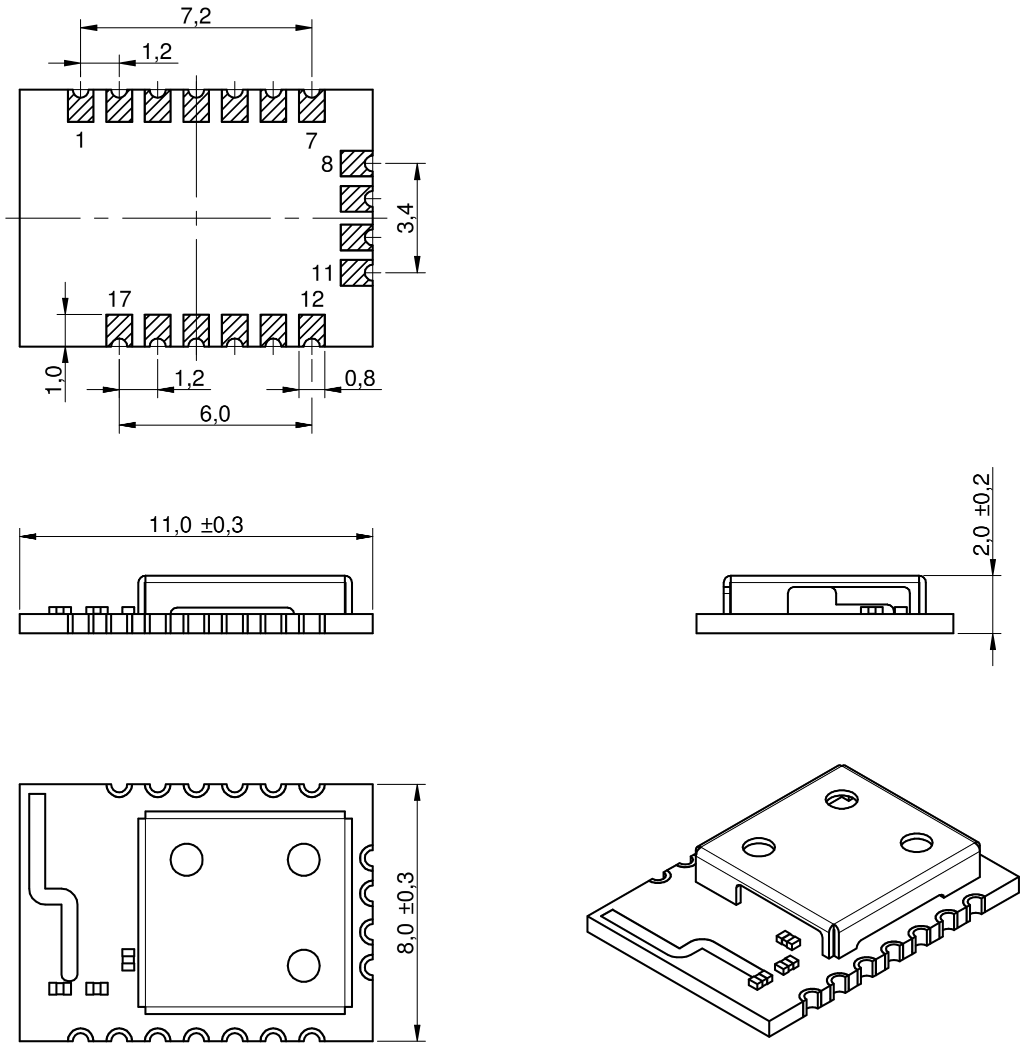


Figure 16: Module dimensions [mm]

18.4. Footprint WE-FP-4

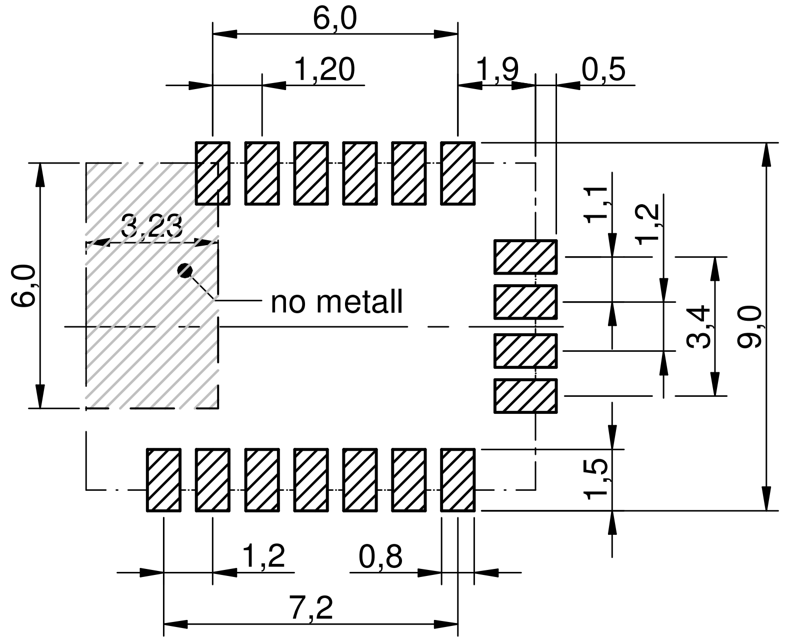


Figure 17: Footprint WE-FP-4 [mm]

18.5. Antenna free area

To avoid influence and mismatching of the antenna the recommended free area around the antenna should be maintained. As rule of thumb a minimum distance of metal parts to the antenna of λ/10 should be kept (see figure 17). Even though metal parts would influence the characteristic of the antenna, but the direct influence and matching keep an acceptable level.

19. Marking

19.1. 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.

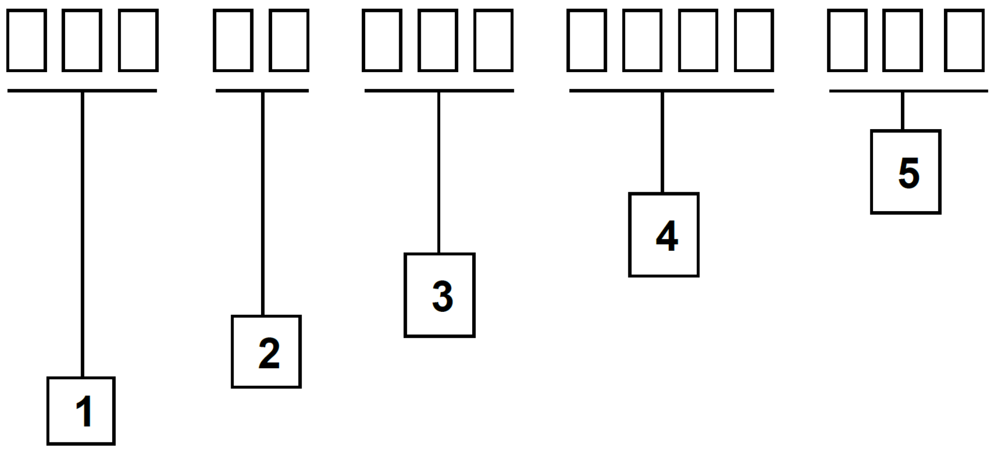


Figure 18: Lot number structure

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 24: 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.

19.2. 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.

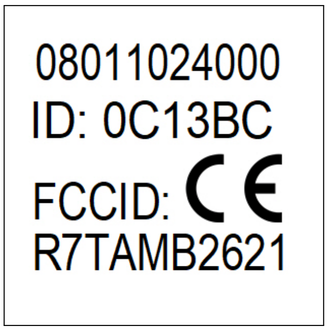

08011024000
ID: 0C13BC
FCCID: CE
R7TAMB2621

Figure 19: Label of the Proteus-I

20. Information for Ex 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 5dBm for external antenna and 0dBm for internal antenna.

- The total amount of capacitivity of all capacitors is 5.9µF.

- The total amount of inductivity of all inductors is 10.025µH.
- A DC/DC regulator is included in the chipset and used to obtain low power functionality.

21. Bluetooth SIG listing/qualification

Type	Data
Design name	AMB2621
Declaration ID	D033500
QDID	90212
Specification name	4.2
Project type	End product

Each product containing intellectual property of the Bluetooth® Special Interest Group (SIG) must be qualified by the SIG to obtain the corresponding Declaration ID. Every new Bluetooth® design must pass the qualification process, even when linking to a Bluetooth® design that is already qualified. To go through the qualification process each company must register as a member of the Bluetooth® SIG:
https://www.bluetooth.org/login/register/

Due to the qualification of the Proteus-I as end product no further Bluetooth® tests are required. The only arising expenses are those for purchasing a Bluetooth® Declaration ID.

The fees for the Declaration ID depend on your membership status:
https://www.bluetooth.org/en-us/test-qualification/qualification-overview/fees

Refer to the testing laboratory of your choice for further more detailed information regarding the qualification of your product.

21.1. Qualification steps when referencing the Proteus-I

Due to the qualification of the Proteus-I as end product, it can be referenced when starting the qualification process of any product integrating the Proteus-I. To perform the qualification process in a row, we recommend purchasing a Declaration ID before starting the new qualification. Any of the following links was valid at the time this document was created but are subject to change without further notice.
Perform the following steps:

1. Visit *https://www.bluetooth.org/tpg/QLI_SDoc.cfm* .
2. Select "Manage Declarations IDs" .
3. Push the "Purchase a Declaration ID" button and fill in the form.

The process can be finished once the invoice for the Declaration ID is paid.

To perform the qualification process of a product integrating Proteus-I, please go through the following steps:

1. Visit *https://www.bluetooth.org/tpg/QLI_SDoc.cfm* .
2. Select option "Start the Bluetooth® Qualification Process with NO Required Testing".
3. Enter the QDID (see above) and select the corresponding Proteus-I entry.
4. Select your pre-paid Declaration ID (it can be selected as soon as the Declaration ID has been paid).
5. Follow the subsequent steps to finish the qualification process.

After finishing the process, the product will be listed on the Bluetooth® website.

22. Regulatory compliance information

22.1. Important notice EU

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

22.2. Important notice FCC

The use of RF frequencies is limited by national regulations. The Proteus-I has been designed to comply with the FCC Part 15.
The Proteus-I can be operated without notification and free of charge in the area of the United States of America. However, according to the FCC Part 15, restrictions (e.g. in terms of maximum allowed RF power and antenna) may apply.

22.3. Conformity assessment of the final product

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

22.4. Exemption clause

Relevant regulation requirements are subject to change. Würth Elektronik eiSos does not guarantee the accuracy of the before mentioned information. Directives, technical standards, procedural descriptions and the like may be interpreted differently by the national authorities. Equally, the national laws and restrictions may vary with the country. In case of doubt or uncertainty, we recommend that you consult with the authorities or official certification organizations of the relevant countries. Würth Elektronik eiSos is exempt from any responsibilities or liabilities related to regulatory compliance.

Notwithstanding the above, Würth Elektronik eiSos makes no representations and warranties of any kind related to their accuracy, correctness, completeness and/or usability for customer applications. No responsibility is assumed for inaccuracies or incompleteness.

22.5. EU Declaration of conformity

EU DECLARATION OF CONFORMITY

Radio equipment: **2608011024000 & 2608011124000**

The manufacturer: Würth Elektronik eiSos GmbH & Co. KG
Max-Eyth-Straße 1
74638 Waldenburg

This declaration of conformity is issued under the sole responsibility of the manufacturer.

Object of the declaration: 2608011024000 & 2608011124000

The object of the declaration described above is in conformity with the relevant Union harmonisation legislation Directive 2014/53/EU and 2011/65/EU with its amending Annex II EU 2015/863 . Following harmonised norms or technical specifications have been applied:

EN 300 328 V2.2.2 (2019-07)
EN 301 489-1 V2.2.3 (2019-11)
EN 301 489-17 V3.2.4 (2020-09)
EN 62479 : 2010
EN 62368-1:2014 + AC:2015

[illegible]

Trier, 18th of December 2020
Place and date of issue

22.6. FCC Compliance Statement

FCC ID: R7TAMB2621

This device complies with Part 15 of the FCC Rules.
Operation is subject to the following two conditions:
(1) this device may not cause harmful interference, and
(2) this device must accept any interference received, including interference that may cause undesired operation.
(FCC 15.19)

Modifications (FCC 15.21)
Caution: Changes or modifications for this equipment not expressly approved by Würth Elektronik eiSos may void the FCC authorization to operate this equipment.

22.7. IC Compliance Statement

Certification Number: 5136A-AMB2621
PMN: AMB2621
This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.
Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

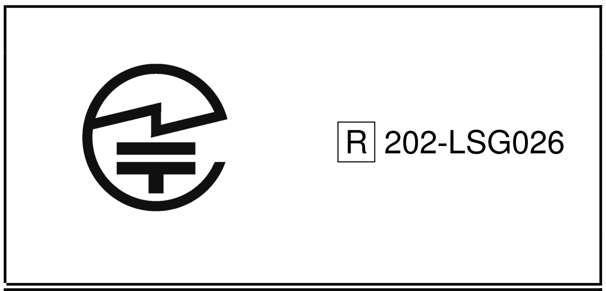
22.8. ARIB Declaration of conformity

	Japanese Radio Law Compliance. This device is granted pursuant to the Japanese Radio Law. This device should not be modified (otherwise the granted designation number will become invalid)
ID-Code (Interference provision)	The MAC address of the radio device maintains the format 00:18:DA:xx:xx:xx. The latter part xx:xx:xx of the MAC address coincides with the serial number of the device.

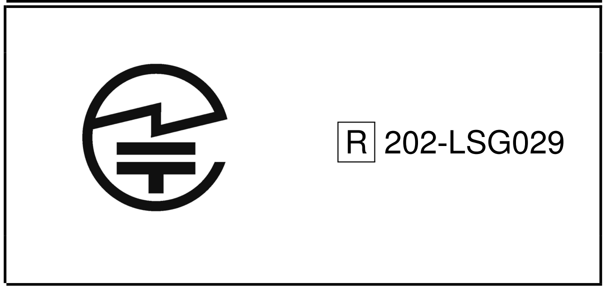
22.8.1. Label

Due to the size of the Proteus-I (AMB2621) label, the certification label of the Proteus-I is not placed onto the module label.

260801102400x
(AMB2621):



260801112400x
(AMB2621-1):



After integration of the Proteus-I (AMB2621, AMB2621-1) in the end device, the corresponding certification label must be recognized from the outside. Otherwise this information must be referenced on the housing as well as in the user manual. E labeling is allowed.

22.8.2. Certified antennas

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

Product	Certified antennas
Proteus-I (2608011024000, AMB2621)	PCB antenna included in the Proteus-I
Proteus-I (2608011124000, AMB2621-1)	260013021 (AMB1926) - 2.4 GHz dipole antenna[1] as specified in chapter `15.3.4.1`

[1] Additional, not yet certified, antennas must be re-certified without retesting. Only antenna gain and antenna characteristic diagrams must be specified. Please contact your local field sales engineer (FSE) to get support in certifying your own antenna.

23. Important notes

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

23.1. General customer responsibility

Some goods within the product range of Würth Elektronik eiSos GmbH & Co. KG contain statements regarding general suitability for certain application areas. These statements about suitability are based on our knowledge and experience of typical requirements concerning the areas, serve as general guidance and cannot be estimated as binding statements about the suitability for a customer application. The responsibility for the applicability and use in a particular customer design is always solely within the authority of the customer. Due to this fact, it is up to the customer to evaluate, where appropriate to investigate and to decide whether the device with the specific product characteristics described in the product specification is valid and suitable for the respective customer application or not. Accordingly, the customer is cautioned to verify that the documentation is current before placing orders.

23.2. Customer responsibility related to specific, in particular safety-relevant applications

It has to be clearly pointed out that the possibility of a malfunction of electronic components or failure before the end of the usual lifetime cannot be completely eliminated in the current state of the art, even if the products are operated within the range of the specifications. The same statement is valid for all software sourcecode and firmware parts contained in or used with or for products in the wireless connectivity and sensor product range of Würth Elektronik eiSos GmbH & Co. KG. In certain customer applications requiring a high level of safety and especially in customer applications in which the malfunction or failure of an electronic component could endanger human life or health, it must be ensured by most advanced technological aid of suitable design of the customer application that no injury or damage is caused to third parties in the event of malfunction or failure of an electronic component.

23.3. Best care and attention

Any product-specific data sheets, manuals, application notes, PCN's, warnings and cautions must be strictly observed in the most recent versions and matching to the products firmware revisions. This documents can be downloaded from the product specific sections on the wireless connectivity homepage.

23.4. Customer support for product specifications

Some products within the product range may contain substances, which are subject to restrictions in certain jurisdictions in order to serve specific technical requirements. Necessary information is available on request. In this case, the field sales engineer or the internal sales person in charge should be contacted who will be happy to support in this matter.

23.5. Product improvements

Due to constant product improvement, product specifications may change from time to time. As a standard reporting procedure of the Product Change Notification (PCN) according to the JEDEC-Standard, we inform about major changes. In case of further queries regarding the PCN, the field sales engineer, the internal sales person or the technical support team in charge should be contacted. The basic responsibility of the customer as per section 23.1 and 23.2 remains unaffected. All wireless connectivity module driver software ¨wireless connectivity SDK¨ and it's source codes as well as all PC software tools are not subject to the Product Change Notification information process.

23.6. Product life cycle

Due to technical progress and economical evaluation we also reserve the right to discontinue production and delivery of products. As a standard reporting procedure of the Product Termination Notification (PTN) according to the JEDEC-Standard we will inform at an early stage about inevitable product discontinuance. According to this, we cannot ensure that all products within our product range will always be available. Therefore, it needs to be verified with the field sales engineer or the internal sales person in charge about the current product availability expectancy before or when the product for application design-in disposal is considered. The approach named above does not apply in the case of individual agreements deviating from the foregoing for customer-specific products.

23.7. Property rights

All the rights for contractual products produced by Würth Elektronik eiSos GmbH & Co. KG on the basis of ideas, development contracts as well as models or templates that are subject to copyright, patent or commercial protection supplied to the customer will remain with Würth Elektronik eiSos GmbH & Co. KG. Würth Elektronik eiSos GmbH & Co. KG does not warrant or represent that any license, either expressed or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, application, or process in which Würth Elektronik eiSos GmbH & Co. KG components or services are used.

23.8. General terms and conditions

Unless otherwise agreed in individual contracts, all orders are subject to the current version of the "General Terms and Conditions of Würth Elektronik eiSos Group", last version available at *www.we-online.com*.

24. Legal notice

24.1. Exclusion of liability

Würth Elektronik eiSos GmbH & Co. KG considers the information in this document to be correct at the time of publication. However, Würth Elektronik eiSos GmbH & Co. KG reserves the right to modify the information such as technical specifications or functions of its products or discontinue the production of these products or the support of one of these products without any written announcement or notification to customers. The customer must make sure that the information used corresponds to the latest published information. Würth Elektronik eiSos GmbH & Co. KG does not assume any liability for the use of its products. Würth Elektronik eiSos GmbH & Co. KG does not grant licenses for its patent rights or for any other of its intellectual property rights or third-party rights.

Notwithstanding anything above, Würth Elektronik eiSos GmbH & Co. KG makes no representations and/or warranties of any kind for the provided information related to their accuracy, correctness, completeness, usage of the products and/or usability for customer applications. Information published by Würth Elektronik eiSos GmbH & Co. KG regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof.

24.2. Suitability in customer applications

The customer bears the responsibility for compliance of systems or units, in which Würth Elektronik eiSos GmbH & Co. KG products are integrated, with applicable legal regulations. Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of Würth Elektronik eiSos GmbH & Co. KG components in its applications, notwithstanding any applications-related in-formation or support that may be provided by Würth Elektronik eiSos GmbH & Co. KG. Customer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences lessen the likelihood of failures that might cause harm and take appropriate remedial actions. The customer will fully indemnify Würth Elektronik eiSos GmbH & Co. KGand its representatives against any damages arising out of the use of any Würth Elektronik eiSos GmbH & Co. KG components in safety-critical applications.

24.3. Trademarks

AMBER wireless is a registered trademark of Würth Elektronik eiSos GmbH & Co. KG. All other trademarks, registered trademarks, and product names are the exclusive property of the respective owners.

24.4. Usage restriction

Würth Elektronik eiSos GmbH & Co. KG products have been designed and developed for usage in general electronic equipment only. This product is not authorized for use in equipment where a higher safety standard and reliability standard is especially required or where 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.

25. License terms

This License Terms will take effect upon the purchase and usage of the Würth Elektronik eiSos GmbH & Co. KG wireless connectivity products. You hereby agree that this license terms is applicable to the product and the incorporated software, firmware and source codes (collectively, "Software") made available by Würth Elektronik eiSos in any form, including but not limited to binary, executable or source code form.
The software included in any Würth Elektronik eiSos wireless connectivity product is purchased to you on the condition that you accept the terms and conditions of this license terms. You agree to comply with all provisions under this license terms.

25.1. Limited license

Würth Elektronik eiSos hereby grants you a limited, non-exclusive, non-transferable and royalty-free license to use the software and under the conditions that will be set forth in this license terms. You are free to use the provided Software only in connection with one of the products from Würth Elektronik eiSos to the extent described in this license terms. You are entitled to change or alter the source code for the sole purpose of creating an application embedding the Würth Elektronik eiSos wireless connectivity product. The transfer of the source code to third parties is allowed to the sole extent that the source code is used by such third parties in connection with our product or another hardware provided by Würth Elektronik eiSos under strict adherence of this license terms. Würth Elektronik eiSos will not assume any liability for the usage of the incorporated software and the source code. You are not entitled to transfer the source code in any form to third parties without prior written consent of Würth Elektronik eiSos.
You are not allowed to reproduce, translate, reverse engineer, decompile, disassemble or create derivative works of the incorporated Software and the source code in whole or in part. No more extensive rights to use and exploit the products are granted to you.

25.2. Usage and obligations

The responsibility for the applicability and use of the Würth Elektronik eiSos wireless connectivity product with the incorporated Firmware in a particular customer design is always solely within the authority of the customer. Due to this fact, it is up to you to evaluate and investigate, where appropriate, and to decide whether the device with the specific product characteristics described in the product specification is valid and suitable for your respective application or not.
You are responsible for using the Würth Elektronik eiSos wireless connectivity product with the incorporated Firmware in compliance with all applicable product liability and product safety laws. You acknowledge to minimize the risk of loss and harm to individuals and bear the risk for failure leading to personal injury or death due to your usage of the product.
Würth Elektronik eiSos' products with the incorporated Firmware are not authorized for use in safety-critical applications, or where a failure of the product is reasonably expected to cause severe personal injury or death. Moreover, Würth Elektronik eiSos' products with the incorporated Firmware are neither designed nor intended for use in areas such as military, aerospace, aviation, nuclear control, submarine, transportation (automotive control, train control, ship control), transportation signal, disaster prevention, medical, public information network etc. You shall inform Würth Elektronik eiSos about the intent of such usage before

design-in stage. In certain customer applications requiring a very high level of safety and in which the malfunction or failure of an electronic component could endanger human life or health, you must ensure to have all necessary expertise in the safety and regulatory ramifications of your applications. You acknowledge and agree that you are solely responsible for all legal, regulatory and safety-related requirements concerning your products and any use of Würth Elektronik eiSos' products with the incorporated Firmware in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by Würth Elektronik eiSos. YOU SHALL INDEMNIFY WÜRTH ELEKTRONIK EISOS AGAINST ANY DAMAGES ARISING OUT OF THE USE OF WÜRTH ELEKTRONIK EISOS' PRODUCTS WITH THE INCORPORATED FIRMWARE IN SUCH SAFETY-CRITICAL APPLICATIONS.

25.3. Ownership

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

25.4. Firmware update(s)

You have the opportunity to request the current and actual Firmware for a bought wireless connectivity Product within the time of warranty. However, Würth Elektronik eiSos has no obligation to update a modules firmware in their production facilities, but can offer this as a service on request. The upload of firmware updates falls within your responsibility, e.g. via ACC or another software for firmware updates. Firmware updates will not be communicated automatically. It is within your responsibility to check the current version of a firmware in the latest version of the product manual on our website. The revision table in the product manual provides all necessary information about firmware updates. There is no right to be provided with binary files, so called "Firmware images", those could be flashed through JTAG, SWD, Spi-Bi-Wire, SPI or similar interfaces.

25.5. Disclaimer of warranty

THE FIRMWARE IS PROVIDED "AS IS". YOU ACKNOWLEDGE THAT WÜRTH ELEKTRONIK EISOS MAKES NO REPRESENTATIONS AND WARRANTIES OF ANY KIND RELATED TO, BUT NOT LIMITED TO THE NON-INFRINGEMENT OF THIRD PARTIES' INTELLECTUAL PROPERTY RIGHTS OR THE MERCHANTABILITY OR FITNESS FOR YOUR INTENDED PURPOSE OR USAGE. WÜRTH ELEKTRONIK EISOS DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT RELATING TO ANY COMBINATION, MACHINE, OR PROCESS IN WHICH THE WÜRTH ELEKTRONIK EISOS' PRODUCT WITH THE INCORPORATED FIRMWARE IS USED. INFORMATION PUBLISHED BY WÜRTH ELEKTRONIK EISOS REGARDING THIRD-PARTY PRODUCTS OR SERVICES DOES NOT CONSTITUTE A LICENSE FROM WÜRTH ELEKTRONIK EISOS TO USE SUCH PRODUCTS OR SERVICES OR A WARRANTY OR ENDORSEMENT THEREOF.

25.6. Limitation of liability

Any liability not expressly provided by Würth Elektronik eiSos shall be disclaimed.
You agree to hold us harmless from any third-party claims related to your usage of the Würth Elektronik eiSos' products with the incorporated Firmware, software and source code. Würth Elektronik eiSos disclaims any liability for any alteration, development created by you or your customers as well as for any combination with other products.

25.7. Applicable law and jurisdiction

Applicable law to this license terms shall be the laws of the Federal Republic of Germany. Any dispute, claim or controversy arising out of or relating to this license terms shall be resolved and finally settled by the court competent for the location of Würth Elektronik eiSos' registered office.

25.8. Severability clause

If a provision of this license terms is or becomes invalid, unenforceable or null and void, this shall not affect the remaining provisions of the terms. The parties shall replace any such provisions with new valid provisions that most closely approximate the purpose of the terms.

25.9. Miscellaneous

Würth Elektronik eiSos reserves the right at any time to change this terms at its own discretion. It is your responsibility to check at Würth Elektronik eiSos homepage for any updates. Your continued usage of the products will be deemed as the acceptance of the change.
We recommend you to be updated about the status of new firmware and software, which is available on our website or in our data sheet and manual, and to implement new software in your device where appropriate.
By ordering a wireless connectivity product, you accept this license terms in all terms.

List of Figures

List of Tables

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 25: 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

WÜRTH ELEKTRONIK

more than you expect

Internet of Things

Monitoring & Control

Automated Meter Reading

Contact:
Würth Elektronik eiSos GmbH & Co. KG
Division Wireless Connectivity & Sensors

Max-Eyth-Straße 1
74638 Waldenburg
Germany

Tel.: +49 651 99355-0
Fax.: +49 651 99355-69
www.we-online.com/wireless-connectivity